เอกสารประกอบการสอน
รายวิชา การเขียนโปรแกรมเชิงวัตถุ

สาธิต สุวรรณเวช

คณะวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
มหาวิทยาลัยราชภัฏรำไพพรรณี

2559

เอกสารประกอบการสอน
รายวิชา การเขียนโปรแกรมเชิงวัตถุ

สาธิต สุวรรณเวช
วท.ม. (วิทยาการคอมพิวเตอร์)

คณะวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
มหาวิทยาลัยราชภัฏรำไพพรรณี

2559

คำนำ

เอกสารประกอบการสอนรายวิชา การเขียนโปรแกรมเชิงวัตถุ รหัส 9022081 ใช้สำหรับนักศึกษาหลักสูตรเทคโนโลยีสารสนเทศของมหาวิทยาลัยราชภัฏรำไพพรรณี รายวิชานี้ผู้เขียนมุ่งเน้นให้ผู้เรียนมีความรู้ความสามารถในการพัฒนาโปรแกรมเชิงวัตถุ แนวคิดเชิงวัตถุ การออกแบบส่วนติดต่อกับผู้ใช้ การเขียนคำสั่งควบคุมการทำงานและการตัดสินใจ การประยุกต์ใช้โครงสร้างข้อมูล การจัดการกับสิ่งผิดปกติ การประยุกต์ใช้งานคลาสและเธรด

ผู้เขียนได้แบ่งเนื้อหาเอกสารประกอบการสอนเป็น 9 บท ประกอบด้วยหลักการวิเคราะห์และออกแบบระบบเชิงวัตถุ หลักการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ พื้นฐานการพัฒนาโปรแกรมด้วยภาษาวิชวลซีชาร์ป การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้และการจัดการเหตุการณ์ การใช้คำสั่งควบคุมและการตัดสินใจ โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชัน คลาสและความสัมพันธ์ระหว่างคลาส การจัดการกับสิ่งผิดปกติ และเธรด ซึ่งเป็นเนื้อหาที่สำคัญในการนำไปใช้เขียนโปรแกรมเชิงวัตถุ

เอกสารประกอบการสอนนี้คงอำนวยประโยชน์ต่อการเรียนการสอนในรายวิชาการเขียนโปรแกรมเชิงวัตถุ ผู้เขียนหวังเป็นอย่างยิ่งว่าเอกสารประกอบการสอนเล่มนี้จะเป็นประโยชน์ต่อผู้เรียน

ผู้เขียนขอขอบคุณเจ้าของเอกสาร ตำรา และหนังสือทุกท่าน ที่ใช้ศึกษาค้นคว้าและอ้างอิงในการเขียน และขอขอบคุณมหาวิทยาลัยราชภัฏรำไพพรรณีที่สนับสนุนส่งเสริมให้คณาจารย์ทำผลงานวิชาการในครั้งนี้

สาธิต สุวรรณเวช
กันยายน 2559

สารบัญ

	หน้า
คำนำ	(1)
สารบัญ	(3)
สารบัญภาพ	(7)
สารบัญตาราง	(13)
แผนบริหารการสอนประจำวิชา	(15)
แผนบริหารการสอนประจำบทที่ 1	1
บทที่ 1 หลักการวิเคราะห์และออกแบบระบบเชิงวัตถุ	3
1.1 แนวคิดและการพัฒนาระบบเชิงวัตถุ	3
1.2 แนวคิดพื้นฐานของระบบเชิงวัตถุ	4
1.3 วัตถุ	8
1.4 คลาส	9
1.5 ยูเอ็มแอล	10
1.6 สรุป	19
แบบฝึกหัดบทที่ 1	21
เอกสารอ้างอิง	23
แผนบริหารการสอนประจำบทที่ 2	25
บทที่ 2 หลักการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ	27
2.1 ภาษาโปรแกรมเชิงวัตถุ	27
2.2 แนวความคิดภาษาเชิงวัตถุ	29
2.3 คุณสมบัติภาษาเชิงวัตถุ	32
2.4 สรุป	35
แบบฝึกหัดบทที่ 2	37
เอกสารอ้างอิง	39
แผนบริหารการสอนประจำบทที่ 3	41
บทที่ 3 พื้นฐานการพัฒนาโปรแกรมด้วยภาษาวิชวลซีชาร์ป	43
3.1 เริ่มต้นเรียนรู้ภาษาวิชวลซีชาร์ป	43
3.2 คลาสที่เกี่ยวข้องกับอินพุทและเอาท์พุท	52
3.3 การเขียนโปรแกรมด้วยวิชวลซีชาร์ป	54
3.4 สรุป	58
แบบฝึกหัดบทที่ 3	59
เอกสารอ้างอิง	61
แผนบริหารการสอนประจำบทที่ 4	63
บทที่ 4 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้และการจัดการเหตุการณ์	65

สารบัญ (ต่อ)

	หน้า
4.1 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้	65
4.2 การจัดการเหตุการณ์	70
4.3 สรุป	79
แบบฝึกหัดบทที่ 4	81
เอกสารอ้างอิง	83
แผนบริหารการสอนประจำบทที่ 5	85
บทที่ 5 การใช้คำสั่งควบคุมและการตัดสินใจ	87
5.1 การตัดสินใจ	87
5.2 การทำงานซ้ำ	106
5.3 สรุป	124
แบบฝึกหัดบทที่ 5	127
เอกสารอ้างอิง	129
แผนบริหารการสอนประจำบทที่ 6	131
บทที่ 6 โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชัน	133
6.1 โครงสร้างข้อมูลแบบแถวลำดับ	133
6.2 โครงสร้างข้อมูลแบบคอลเลคชัน	149
6.3 สรุป	158
แบบฝึกหัดบทที่ 6	159
เอกสารอ้างอิง	161
แผนบริหารการสอนประจำบทที่ 7	163
บทที่ 7 คลาสและความสัมพันธ์ระหว่างคลาส	165
7.1 การสร้างคลาสและการสร้างวัตถุจากคลาส	165
7.2 ความสัมพันธ์ระหว่างคลาส	182
7.3 การใช้งานดีไซน์แพทเทิร์น	194
7.4 สรุป	202
แบบฝึกหัดบทที่ 7	203
เอกสารอ้างอิง	205
แผนบริหารการสอนประจำบทที่ 8	207
บทที่ 8 การจัดการกับสิ่งผิดปกติ	209
8.1 ประเภทของความผิดพลาด	209
8.2 การตรวจสอบข้อผิดพลาด	211
8.3 เครื่องมือสำหรับตรวจสอบข้อผิดพลาด	216
8.4 การจัดการข้อผิดพลาด	219

สารบัญ (ต่อ)

	หน้า
8.5 สรุป	226
แบบฝึกหัดบทที่ 8	227
เอกสารอ้างอิง	229
แผนบริหารการสอนประจำบทที่ 9	231
บทที่ 9 เธรด	233
9.1 ความหมายของเธรด	233
9.2 การใช้งานเธรด	236
9.3 การประยุกต์ใช้เธรดกับคอนโทรลใน Windows Forms	247
9.4 สรุป	251
แบบฝึกหัดบทที่ 9	253
เอกสารอ้างอิง	255
บรรณานุกรม	257
ภาคผนวก	261

สารบัญภาพ

ภาพที่		หน้า
1.1	องค์ประกอบของระบบเชิงวัตถุ	4
1.2	การกำหนดสาระสำคัญระบบการซื้อขาย	5
1.3	การกำหนดสาระสำคัญแผนที่ My School	5
1.4	การซ่อนรายละเอียด	6
1.5	การแยกชิ้นส่วน	6
1.6	การแบ่งลำดับชั้น	7
1.7	ลำดับชั้นของสัตว์	7
1.8	วัตถุที่เป็นกายภาพ	8
1.9	วัตถุที่เป็นแนวคิด	8
1.10	วัตถุที่เป็นซอฟต์แวร์	8
1.11	การเขียนคลาส	9
1.12	การเขียนชื่อคลาส	9
1.13	การเขียนการกระทำของคลาส	10
1.14	ไดอะแกรมของยูเอ็มแอล	11
1.15	สัญลักษณ์แอคเตอร์	11
1.16	สัญลักษณ์ยูสเคส	11
1.17	ตัวอย่างการเขียนยูสเคส	12
1.18	ความสัมพันธ์แบบโครงสร้าง	13
1.19	ความสัมพันธ์แบบแอกกรีเกชัน	14
1.20	ความสัมพันธ์แบบคอมโพสิชัน	14
1.21	ความสัมพันธ์แบบเจนเนอราไลเซชัน	15
1.22	ตัวอย่างออปเจคต์ ไดอะแกรม	15
1.23	ตัวอย่างซีเควนไดอะแกรมการเข้าสู่ระบบ	16
1.24	ตัวอย่างคอลลาโบเรชันไดอะแกรมการเข้าสู่ระบบ	16
1.25	ตัวอย่างสเตทไดอะแกรมการลงทะเบียนเรียน	17
1.26	ตัวอย่างแอคทิวิทีไดอะแกรมการซื้อขายดีวีดี (DVD)	18
1.27	ตัวอย่างคอมโพเนนท์ไดอะแกรม	19
1.28	ตัวอย่างดีพลอยเมนท์ไดอะแกรม	19
2.1	วิธีการพัฒนาโปรแกรม	28
2.2	ตัวอย่างของพรอบเบรมสเปซ	29
2.3	แผนผังการแก้ปัญหาพื้นที่สี่เหลี่ยม	29
2.4	ตัวอย่างการวิเคราะห์คลาสของสี่เหลี่ยม	30
2.5	การทำงานร่วมกันของวัตถุ	30

สารบัญภาพ (ต่อ)

ภาพที่		หน้า
2.6	วัตถุต้องเกิดจากคลาส	31
2.7	คุณสมบัติของวัตถุที่เกิดจากคลาสเดียวกัน	31
2.8	การได้รับข่าวสารของคลาส	31
2.9	การห่อหุ้ม	32
2.10	การซ่อนรายละเอียด	33
2.11	การสืบทอดคุณสมบัติ	33
2.12	การพ้องรูป	34
3.1	ความยากของวิชวลซีชาร์ป	43
3.2	หน้าจอของวิชวลสตูดิโอ 2010	43
3.3	หน้าจอการสร้าง Project	44
3.4	การสร้าง Project แบบ Console Application	45
3.5	ผลการรันแบบ Console Application	45
3.6	การสร้าง Project แบบ Windows Forms Application	46
3.7	ผลการรันแบบ Windows Application	46
3.8	โครงสร้างของภาษาวิชวลซีชาร์ป	47
3.9	โครงสร้างของภาษาวิชวลซีชาร์ปแบบ Console Application	47
3.10	โครงสร้างของภาษาวิชวลซีชาร์ปแบบ Windows Forms Application	48
3.11	การแสดงผลของโปรแกรมคำนวณพื้นที่สี่เหลี่ยม	54
3.12	การแสดงผลของโปรแกรมรับชื่อ	56
3.13	ผลของ MessageBox	57
4.1	แถบเครื่องมือ	65
4.2	การวาง Common Controls ในฟอร์ม	66
4.3	หน้าต่าง Properties ของฟอร์ม	67
4.4	ออกแบบฟอร์มโปรแกรมวิชวลซีชาร์ป	68
4.5	ออกแบบฟอร์มแสดงผลการเรียนนักศึกษา	69
4.6	ออกแบบฟอร์มแบบสอบถาม	69
4.7	การทำงานเมื่อมีเหตุการณ์เกิดขึ้นกับวัตถุ	70
4.8	การสร้างอีเวนท์สำหรับ Button	71
4.9	อีเวนท์ของวัตถุ	72
4.10	หน้าจอโปรแกรมแปลงหน่วยกิโลเมตร	72
4.11	หน้าจอโปรแกรมแปลงชั่วโมงเป็นนาที วินาที	74
4.12	หน้าจอโปรแกรมรับค่าตัวเลข	76
5.1	แผนผังการทำงานของคำสั่ง if	88

สารบัญภาพ (ต่อ)

ภาพที่		หน้า
5.2	แผนผังการทำงานของคำสั่ง if และ else	89
5.3	แผนผังการทำงานของคำสั่ง if และ else แบบหลายทางเลือก	91
5.4	หน้าจอตัวอย่าง if...else	91
5.5	หน้าจอโปรแกรมตัดเกรด	96
5.6	หน้าจอโปรแกรมแบบสำรวจที่พัก RBRU รีสอร์ท	98
5.7	แสดง MessageBox เมื่อใส่อายุน้อยกว่า 20 ปี	99
5.8	แสดง MessageBox เมื่อใส่อายุระหว่าง 20 – 50 ปี	99
5.9	แสดง MessageBox เมื่อใส่อายุมากกว่า 50 ปี	100
5.10	แสดง MessageBox เมื่อคลิ๊กที่ปุ่ม btnQuestion	100
5.11	แสดง MessageBox เมื่อคลิ๊กที่ปุ่ม Yes	100
5.12	แสดง MessageBox เมื่อคลิ๊กที่ปุ่ม No	100
5.13	แสดง MessageBox เมื่อไม่มีการแสดงความคิดเห็น	101
5.14	แสดง MessageBox เมื่อมีการแสดงความคิดเห็น	101
5.15	แผนผังการทำงานของคำสั่ง switch	104
5.16	แผนผังการทำงานของคำสั่ง for	107
5.17	หน้าจอโปรแกรมคำนวณสูตรคูณ	109
5.18	ผลลัพธ์ของโปรแกรมคำนวณสูตรคูณ	110
5.19	หน้าจอของโปรแกรมทดสอบการวนซ้ำ	112
5.20	ผลลัพธ์เมื่อกดปุ่มวนซ้ำแบบเดินหน้า	113
5.21	ผลลัพธ์เมื่อกดปุ่มวนซ้ำแบบถอยหลัง	113
5.22	หน้าจอโปรแกรมสร้างตาราง	115
5.23	ผลลัพธ์เมื่อกดสร้างตาราง	116
5.24	แผนผังการทำงานของคำสั่ง while	118
5.25	แผนผังการทำงานของคำสั่ง do...while	122
6.1	อาร์เรย์แบบ 1 มิติ	133
6.2	อาร์เรย์แบบ 2 มิติขนาด 2 x 5	135
6.3	อาร์เรย์แบบ 3 มิติขนาด 2 x 2 x 3	136
6.4	อาร์เรย์แบบ 4 มิติขนาด 2 x 2 x 2 x 2	136
6.5	หน้าจอโปรแกรมรับค่าลงอาร์เรย์ 1 มิติ	138
6.6	หน้าจอโปรแกรมเมื่อกดปุ่มรับค่าเพิ่ม	139
6.7	หน้าจอโปรแกรมเมื่อรับค่าครบ 10 ค่า	139
6.8	หน้าจอโปรแกรมเมื่อกดปุ่มแสดงผล	140
6.9	ตัวอย่างการคำนวณเมทริกซ์	142

สารบัญภาพ (ต่อ)

ภาพที่		หน้า
6.10	หน้าจอโปรแกรมคำนวณเมทริกซ์	142
6.11	การทำงานของสแตก	150
6.12	หน้าจอโปรแกรมตัวอย่างสแตก	151
6.13	หน้าจอเมื่อนำข้อมูลเข้าสู่สแตก	152
6.14	หน้าจอเมื่อนำข้อมูลออกจากสแตก	152
6.15	การทำงานคิว	154
6.16	หน้าจอโปรแกรมตัวอย่างคิว	154
6.17	หน้าจอโปรแกรมเมื่อนำข้อมูลเข้าสู่คิว	156
6.18	หน้าจอโปรแกรมเมื่อนำข้อมูลออกจากคิว	156
7.1	ขั้นตอนการสร้าง Project	165
7.2	การสร้าง Class Library	166
7.3	หน้าจอสำหรับสร้างคลาสใน Class Library	166
7.4	การสร้างคลาสใน Windows Forms Application	167
7.5	ตัวอย่างคลาส User	167
7.6	ตัวอย่างการแปลงคลาสไปเป็นส่วนติดต่อกับผู้ใช้	168
7.7	ตัวอย่างการแปลงคลาสไปเป็นฐานข้อมูล	168
7.8	โครงสร้างการเขียนคลาสในวิชวลซีชาร์ป	169
7.9	คลาสไดอะแกรม Customer	171
7.10	หน้าจอตัวอย่างการสร้างวัตถุจากคลาส Form	174
7.11	หน้าจอเมื่อโปรแกรมทำงาน	174
7.12	แสดง MessageBox หลังจากกดปุ่มสร้างฟอร์มใหม่	174
7.13	หน้าจอ Form ใหม่ที่ถูกสร้างขึ้นมา	175
7.14	หน้าจอตัวอย่างการสร้างฟอร์มสี	176
7.15	หน้าจอขณะผู้ใช้เลือกสีของฟอร์ม	177
7.16	หน้าจอฟอร์มใหม่ตามสีที่เลือก	177
7.17	คลาส Cycle	179
7.18	หน้าจอโปรแกรมคำนวณวงกลม	179
7.19	ความสัมพันธ์ระหว่างคลาสแบบแอสโซสิเอชัน	182
7.20	ความสัมพันธ์ระหว่างคลาสแบบแอกกรีเกชัน	183
7.21	ความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชัน	184
7.22	ความสัมพันธ์ระหว่างคลาสแบบการสืบทอด	186
7.23	คลาส General เมื่อผ่านการสืบทอด	186
7.24	คลาส Member เมื่อผ่านการสืบทอด	187

สารบัญภาพ (ต่อ)

ภาพที่		หน้า
7.25	ความสัมพันธ์ระหว่างคลาสแบบการพ้องรูป	188
7.26	คลาสตัวอย่างการหาพื้นที่แบบการพ้องรูป	190
7.27	หน้าจอตัวอย่างการหาพื้นที่แบบการพ้องรูป	190
7.28	คลาสของแอบสแตกแฟคทอรี	195
7.29	คลาสของซิงเกิลตอน	199
8.1	การแสดงข้อผิดพลาดทางไวยกรณ์	209
8.2	การแสดงข้อผิดพลาดระหว่างการทำงาน	210
8.3	การแสดงข้อผิดพลาดทางตรรกะ	211
8.4	กลยุทธ์ในการทดสอบโปรแกรม	212
8.5	การแบ่งพิสัยของข้อมูลตามวิธีทดสอบแบบกล่องดำ	213
8.6	ลูปอย่างง่าย	214
8.7	ลูปซ้อนลูป	214
8.8	ลูปต่อกัน	214
8.9	ลูปไม่เป็นโครงสร้าง	215
8.10	การใช้งานเบรคพอยท์	217
8.11	การใช้งาน Step Into	217
8.12	การเรียกใช้งานหน้าต่าง Error List	218
8.13	รายละเอียดที่แสดงในหน้าต่าง Error List	219
8.14	หน้าจอโปรแกรมตัวอย่างการตรวจสอบข้อผิดพลาด	221
8.15	การใส่ค่าเป็นเลขทศนิยมในโปรแกรม	222
8.16	ผลลัพธ์เมื่อใส่ค่าที่ไม่ใช่เลขจำนวนเต็ม	222
8.17	การใส่ค่าเป็นเลขจำนวนเต็ม	222
8.18	ผลลัพธ์เมื่อไม่เกิดข้อผิดพลาด	223
9.1	การทำงานแบบเธรดเดียว	233
9.2	การทำงานแบบหลายเธรด	234
9.3	การทำงานแบบกลุ่มต่อหนึ่ง	234
9.4	การทำงานแบบหนึ่งต่อหนึ่ง	235
9.5	การทำงานแบบกลุ่มต่อกลุ่ม	235
9.6	การทำงานของเธรด	237
9.7	หน้าจอตัวอย่างโปรแกรมทดสอบเธรด	237
9.8	ผลการทำงานของโปรแกรม	237
9.9	ผลการใช้เมธอด Join	240
9.10	หน้าจอโปรแกรมการบวกเลขคู่และเลขคี่	241

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

9.11 ผลการทำงานของโปรแกรมการบวกเลขคู่และเลขคี่ 241
9.12 หน้าจอโปรแกรมคำนวณ 244
9.13 ผลลัพธ์จากโปรแกรมคำนวณ 245
9.14 หน้าจอโปรแกรมเพิ่มข้อมูลลง listBox 248
9.15 ผลลัพธ์ของโปรแกรมเพิ่มข้อมูลลง listBox 249

สารบัญตาราง

ตารางที่		หน้า
1.1	การเปรียบเทียบการวิเคราะห์และออกแบบระบบวิธีเดิมกับวิธีเชิงวัตถุ	4
1.2	คุณสมบัติของผลไม้	9
2.1	การเปรียบเทียบระหว่างการพัฒนาโปรแกรมเชิงฟังก์ชันและเชิงวัตถุ	28
3.1	คำสงวน	49
3.2	เลขจำนวนเต็ม	49
3.3	เลขจำนวนเต็มบวก	49
3.4	เลขทศนิยม	50
3.5	ประเภทข้อมูลที่ไม่ใช่ตัวเลข	50
3.6	ลำดับความสำคัญของตัวดำเนินการ	51
3.7	การเรียกใช้งานปุ่มใน MessageBox	53
3.8	การเรียกใช้งาน Icon ใน MessageBox	54
4.1	Properties สำคัญของฟอร์ม	67
4.2	เมธอดสำคัญของฟอร์ม	68
4.3	อีเวนท์สำคัญของฟอร์ม	71
6.1	คลาสที่สำคัญในเนมสเปซ System.Collections	149
6.2	คลาสที่สำคัญในเนมสเปซ System.Collections.Generic	149
8.1	ปุ่มแถบเครื่องมือดีบัก	216
8.2	รายละเอียดของหน้าต่าง Error List	218
9.1	การทำงานของเมธอดเกี่ยวกับเธรด	236

แผนบริหารการสอนประจำวิชา

รหัสวิชา 9022081

ชื่อรายวิชา การเขียนโปรแกรมเชิงวัตถุ 3 (2-2-5) หน่วยกิต
Object oriented programming

คำอธิบายรายวิชา

การออกแบบและพัฒนาโปรแกรมเชิงวัตถุ การห่อหุ้ม การสืบทอดการพ้องรูป การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้และการจัดการเหตุการณ์ โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชัน การจัดการกับสิ่งผิดปกติ คลาสที่เกี่ยวข้องกับอินพุตและเอาต์พุต เธรด

วัตถุประสงค์ทั่วไป

1. เพื่อให้ผู้เรียนทราบถึงหลักการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ
2. เพื่อให้ผู้เรียนทราบถึงแนวคิดเชิงวัตถุ
3. เพื่อให้ผู้เรียนสามารถออกแบบส่วนติดต่อกับผู้ใช้ได้
4. เพื่อให้ผู้เรียนสามารถเขียนโปรแกรมโดยใช้คำสั่งควบคุมและการตัดสินใจได้
5. เพื่อให้ผู้เรียนเข้าใจและประยุกต์ใช้โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชันได้
6. เพื่อให้ผู้เรียนสามารถจัดการกับสิ่งผิดปกติในการเขียนโปรแกรมได้
7. เพื่อให้ผู้เรียนสามารถประยุกต์ใช้งานคลาสและเธรดในการเขียนโปรแกรมได้

เนื้อหา

บทที่ 1 หลักการวิเคราะห์และออกแบบระบบเชิงวัตถุ 4 ชั่วโมง
- 1.1 แนวคิดและการพัฒนาระบบเชิงวัตถุ
- 1.2 แนวคิดพื้นฐานของระบบเชิงวัตถุ
- 1.3 วัตถุ
- 1.4 คลาส
- 1.5 ยูเอ็มแอล
- 1.6 สรุป

บทที่ 2 หลักการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ 4 ชั่วโมง
- 2.1 ภาษาโปรแกรมเชิงวัตถุ
- 2.2 แนวความคิดภาษาเชิงวัตถุ
- 2.3 คุณสมบัติภาษาเชิงวัตถุ
- 2.4 สรุป

บทที่ 3 พื้นฐานการพัฒนาโปรแกรมด้วยภาษาวิชวลซีชาร์ป 4 ชั่วโมง
- 3.1 เริ่มต้นเรียนรู้ภาษาวิชวลซีชาร์ป

3.2 คลาสที่เกี่ยวข้องกับอินพุทและเอาท์พุท
3.3 การเขียนโปรแกรมด้วยวิชวลซีชาร์ป
3.4 สรุป

บทที่ 4 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้และการจัดการเหตุการณ์ 4 ชั่วโมง
4.1 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้
4.2 การจัดการเหตุการณ์
4.3 สรุป

บทที่ 5 การใช้คำสั่งควบคุมและการตัดสินใจ 8 ชั่วโมง
5.1 การตัดสินใจ
5.2 การทำงานซ้ำ
5.3 สรุป

บทที่ 6 โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชัน 8 ชั่วโมง
6.1 โครงสร้างข้อมูลแบบแถวลำดับ
6.2 โครงสร้างข้อมูลแบบคอลเลคชัน
6.3 สรุป

บทที่ 7 คลาสและความสัมพันธ์ระหว่างคลาส 8 ชั่วโมง
7.1 การสร้างคลาสและการสร้างวัตถุจากคลาส
7.2 ความสัมพันธ์ระหว่างคลาส
7.3 การใช้งานดีไซน์แพทเทิร์น
7.4 สรุป

บทที่ 8 การจัดการกับสิ่งผิดปกติ 8 ชั่วโมง
8.1 ประเภทของความผิดพลาด
8.2 การตรวจสอบข้อผิดพลาด
8.3 เครื่องมือสำหรับตรวจสอบข้อผิดพลาด
8.4 การจัดการข้อผิดพลาด
8.5 สรุป

บทที่ 9 เธรด 8 ชั่วโมง
9.1 ความหมายของเธรด
9.2 การใช้งานเธรด
9.3 การประยุกต์ใช้เธรดกับคอนโทรลใน Windows Forms
9.4 สรุป

วิธีสอนและกิจกรรมการเรียนการสอน

1. ฟังบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ศึกษาเอกสารประกอบการสอน และตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
3. ค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต และจากหนังสือหรือตำราด้วยตนเอง

4. แบ่งกลุ่มอภิปรายตามเนื้อหาที่กำหนด
5. ฝึกปฏิบัติ และทดลองทำตามโปรแกรมตัวอย่าง
6. ฝึกทำแบบฝึกหัดท้ายบท

สื่อการเรียนการสอน

1. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ
2. หนังสือค้นคว้าเพิ่มเติมที่เกี่ยวข้อง
3. สื่อมัลติมิเดีย
4. อินเทอร์เน็ตและเว็บไซต์ที่เกี่ยวข้อง
5. โปรแกรมวิชวลสตูดิโอ 2010
6. เครื่องคอมพิวเตอร์และเครื่องฉายโปรเจคเตอร์

การวัดผลและประเมินผล

การวัดผล

1. คะแนนระหว่างภาคเรียน 70 คะแนน มีรายละเอียดดังนี้
   1.1 ความสนใจและเวลาเรียน 10 คะแนน
   1.2 แบบฝึกหัดและงานค้นคว้า 20 คะแนน
   1.3 งานระหว่างเรียน 20 คะแนน
   1.4 สอบกลางภาค 20 คะแนน
2. คะแนนสอบปลายภาคเรียน 30 คะแนน

การประเมินผล

ช่วงคะแนน	80 – 100	ได้ระดับ	A
ช่วงคะแนน	75 – 79	ได้ระดับ	B+
ช่วงคะแนน	70 – 74	ได้ระดับ	B
ช่วงคะแนน	65 – 69	ได้ระดับ	C+
ช่วงคะแนน	60 – 64	ได้ระดับ	C
ช่วงคะแนน	55 – 59	ได้ระดับ	D+
ช่วงคะแนน	50 – 54	ได้ระดับ	D
ช่วงคะแนน	0 – 49	ได้ระดับ	F

แผนบริหารประจำบทที่ 1

เนื้อหา

บทที่ 1 หลักการวิเคราะห์และออกแบบระบบเชิงวัตถุ

1.1 แนวคิดและการพัฒนาระบบเชิงวัตถุ
1.2 แนวคิดพื้นฐานของระบบเชิงวัตถุ
1.3 วัตถุ
1.4 คลาส
1.5 ยูเอ็มแอล
1.6 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. อธิบายแนวคิดและการพัฒนาระบบเชิงวัตถุได้
2. อธิบายความแตกต่างระหว่างการพัฒนาระบบเชิงวัตถุและเชิงโครงสร้างได้
3. อธิบายความหมายของวัตถุและคลาสได้
4. อธิบายยูเอ็มแอลและไดอะแกรมแบบต่าง ๆ ได้
5. วิเคราะห์ระบบโดยใช้ยูเอ็มแอลได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ให้ผู้เรียนค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต
5. แบ่งผู้เรียนเป็นกลุ่มย่อยเพื่ออภิปรายการพัฒนาระบบเชิงวัตถุ
6. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 1

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ
4. แบบฝึกหัดบทที่ 1

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. ประเมินจากแบบฝึกหัดบทที่ 1
4. ประเมินการการสอบกลางภาค

บทที่ 1
หลักการวิเคราะห์และออกแบบระบบเชิงวัตถุ

การวิเคราะห์และออกแบบระบบเชิงวัตถุเป็นการวิเคราะห์เพื่อใช้ในการพัฒนาระบบที่มีอยู่ในปัจจุบัน เนื่องจากระบบในปัจจุบันมีความสลับซับซ้อนเป็นอย่างมาก ทำให้ต้องมองทุกอย่างที่มีอยู่ในระบบเป็นวัตถุ และเชื่อมโยงวัตถุต่าง ๆ ด้วยความสัมพันธ์ที่วัตถุแต่ละชนิดมีให้กัน เครื่องมือที่ใช้ในการวิเคราะห์ระบบเชิงวัตถุนี้ได้แก่ ยูเอ็มแอลซึ่งมีไดอะแกรมต่าง ๆ ที่สามารถทำให้ทราบถึงมุมมองในการวิเคราะห์ที่แตกต่างกันออกไป

1.1 แนวคิดและการพัฒนาระบบเชิงวัตถุ

แนวคิดเชิงวัตถุ (Object oriented) หมายถึง การมองทุกสิ่งในระบบเป็นวัตถุทั้งหมดและใช้วัตถุเป็นตัวหลักในการพิจารณาความเป็นจริงที่เกิดขึ้นในระบบ ซึ่งทุก ๆ กิจกรรมที่เกิดขึ้นในระบบเกิดจาก ความสัมพันธ์และปฏิสัมพันธ์ระหว่างวัตถุ ดังตัวอย่างเช่น

คนรับประทานอาหาร เกิดจากวัตถุ 2 ชนิด คือ คนและอาหาร มีปฏิสัมพันธ์กัน

นายเอเปิดตู้เย็นดื่มน้ำ เกิดจากวัตถุ 3 ชนิด คือ นายเอ ตู้เย็น และน้ำ มีปฏิสัมพันธ์กัน

การพัฒนาระบบเชิงวัตถุใช้หลักการจัดแบ่งประเภทของวัตถุออกเป็นกลุ่ม เรียกว่าคลาส (Class) ซึ่งทำงานร่วมกัน แต่ละคลาสจะมีหน้าที่ความรับผิดชอบที่แตกต่างกัน โดยแต่ละคลาสจะมีคุณสมบัติ (Attribute) และพฤติกรรม (Behavior) ตามบทบาทของตน ข้อมูลในส่วนรายละเอียดหรือคุณสมบัติที่เก็บซ่อนในคลาสจะไม่มีการปะปนกับคลาสอื่น ๆ แต่สามารถติดต่อสื่อสารหรือการร้องขอใช้บริการได้ด้วยข้อความ (Message)

แนวคิดเชิงโครงสร้างเป็นลักษณะของโครงสร้างที่โปรแกรมกับข้อมูลนั้นแยกออกจากกัน การวิเคราะห์ข้อมูลอาจจะใช้เครื่องมือที่เป็นแผนภาพกระแสข้อมูล (Data flow diagram : DFD) และอีอาร์ไดอะแกรม (ER-Diagram) เพื่อดูเส้นทางของข้อมูล แต่แนวคิดเชิงวัตถุนั้นจะมองเป็นวัตถุ และเป็นแหล่งรวบรวมข้อมูล (Data) วิธีการ (Method) โดยมีคลาสเป็นตัวกำหนดคุณสมบัติของวัตถุนั้น ซึ่งยังสามารถสืบทอด (Inheritance) คุณสมบัติต่าง ๆ ออกมาในลักษณะของคลาสย่อย (Subclass) ได้ ดังนั้นหากมีคลาสที่เป็นต้นแบบอยู่แล้ว ก็สามารถนำคุณสมบัติของคลาสต้นแบบนั้นมาใช้งานได้ทันที ซึ่งเป็นการนำกลับมาใช้ใหม่ ทำให้ช่วยลดค่าใช้จ่ายและเวลาในการพัฒนา และยังสามารถปรับปรุงแก้ไขได้ง่ายอีกด้วย

จะเห็นว่าการวิเคราะห์และออกแบบระบบเชิงวัตถุ สามารถพัฒนาระบบงานให้มีระเบียบและมองโปรแกรมที่เขียนออกเป็นส่วนต่าง ๆ ซึ่งสามารถนำส่วนเหล่านั้นมาปรับปรุงแก้ไขได้ง่ายและถ้าเปรียบเทียบกับการเขียนโปรแกรมเชิงโครงสร้าง แม้ระบบงานที่สร้างจะมีความใกล้เคียงกันแต่ส่วนต่าง ๆ ที่จะนำมาใช้งานก็จะต้องมีการปรับเปลี่ยนมากมาย ซึ่งบางครั้งอาจจะต้องเขียนใหม่ทั้งหมด ส่วนการพัฒนาซอฟต์แวร์เชิงโครงสร้างจะมีลักษณะเป็นนามธรรมจำเป็นจะต้องใช้การจินตนาการ

ดังนั้นระบบงานที่พัฒนาตามแนวคิดเชิงโครงสร้าง ก็จะเกิดจากการจินตนาการของผู้พัฒนาแต่ละคน จึงมีซอฟต์แวร์จำนวนมากที่เป็นระบบเดียวกัน แต่ไม่สามารถนำกลับมาใช้ใหม่ได้ทั้งหมดสรุปดังตารางที่ 1.1 (กิตติพงษ์ กลมกล่อม, 2552)

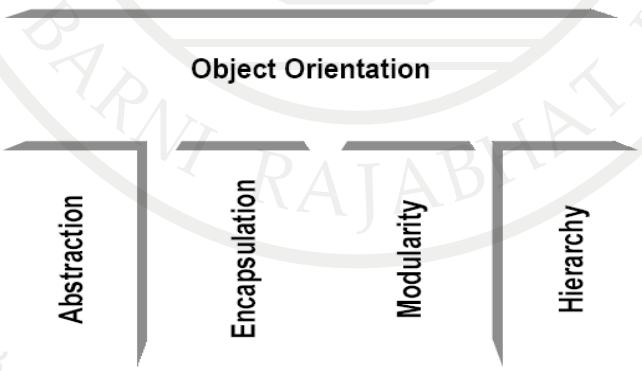
ตารางที่ 1.1 การเปรียบเทียบการวิเคราะห์และออกแบบระบบวิธีเดิมกับวิธีเชิงวัตถุ

วิธีเดิม	วิธีเชิงวัตถุ
เริ่มต้นจากการวิเคราะห์เอกสารผลลัพธ์ และการทำงานของระบบงานเดิม	เริ่มต้นการวิเคราะห์จากวัตถุที่สามารถเห็นได้ชัดเจน
แตกการทำงานออกเป็นหน่วยย่อย ๆ	แบ่งกลุ่มของวัตถุตามคุณลักษณะ
องค์ประกอบต่าง ๆ ของระบบ เช่น การประมวลผล การออกรายงาน การคำนวณ จะเกี่ยวพันกัน การเปลี่ยนแปลงจะกระทบซึ่งกันและกัน	แต่ละวัตถุเป็นอิสระต่อกัน การเปลี่ยนแปลงจะไม่กระทบกัน
การปรับเปลี่ยนระบบต้องแก้ไขชุดคำสั่ง	การปรับเปลี่ยนระบบ ทำได้โดยการเปลี่ยนคุณสมบัติ และการกระทำของวัตถุ
เครื่องมือที่สนับสนุนมีน้อยลง	เครื่องมือที่สนับสนุนมีมากขึ้น

ที่มา : (กิตติพงษ์ กลมกล่อม, 2552)

1.2 แนวคิดพื้นฐานของระบบเชิงวัตถุ

การออกแบบเชิงวัตถุจะมีขั้นตอนการกำหนดรายละเอียดของระบบก่อน หลังจากนั้นจะนำรายละเอียดที่ได้ไปวิเคราะห์เพื่อแปลงความต้องการภายในระบบออกมาเป็นคลาสและวัตถุ และกำหนดความสัมพันธ์ต่าง ๆ ของคลาส โดยใช้ไดอะแกรมในยูเอ็มแอล (UML) มาช่วยวิเคราะห์ หลังจากนั้นจึงนำระบบที่ได้มาเขียนโปรแกรมตามรายละเอียดที่วิเคราะห์ก่อนหน้านี้ ดังภาพที่ 1.1

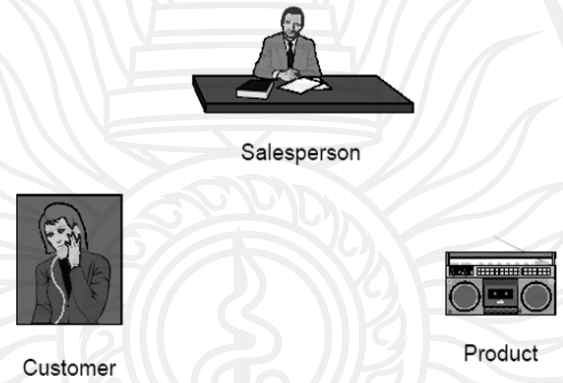


ภาพที่ 1.1 องค์ประกอบของระบบเชิงวัตถุ

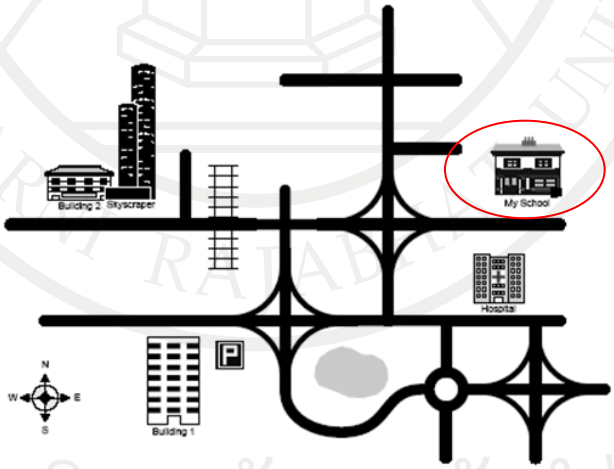
ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)

1.2.1 การกำหนดสาระสำคัญ

การกำหนดสาระสำคัญหรือแอบสแตกชัน (Abstraction) เป็นการแก้ปัญหาขั้นพื้นฐานของมนุษย์ ซึ่งถ้าปัญหาที่พบมีความซับซ้อน มนุษย์จะจดจำเฉพาะสิ่งที่สำคัญและลดการจดจำสิ่งที่เป็นรายละเอียดปลีกย่อย เนื่องจากความสามารถในการจดจำของมนุษย์มีจำกัด ซึ่งอยู่ระหว่าง 7 ± 2 หมายความว่า ถ้าคนที่มีความจำดีก็อาจจะสามารถจดจำสิ่งต่าง ๆ ได้ประมาณ 9 สิ่ง แต่สำหรับคนที่ความจำไม่ค่อยดีจะสามารถจำได้เพียงแค่ 5 สิ่ง ซึ่งในสภาพความเป็นจริงสิ่งที่อยู่รอบ ๆ ตัวเรามีรายละเอียดอยู่มากมาย การที่จะจดจำสิ่งต่าง ๆ ได้ทั้งหมดจึงเป็นไปไม่ได้ มนุษย์จึงใช้การกำหนดสาระสำคัญช่วยในการลดรายละเอียดที่ไม่จำเป็นแล้วเลือกเฉพาะส่วนที่เป็นสิ่งสำคัญ เช่น การจดจำแผนที่ ระบบการซื้อขาย ดังภาพที่ 1.2 และภาพที่ 1.3



ภาพที่ 1.2 การกำหนดสาระสำคัญระบบการซื้อขาย
ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)



ภาพที่ 1.3 การกำหนดสาระสำคัญแผนที่ My School
ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)

1.2.2 การห่อหุ้ม

การห่อหุ้มหรือเอ็นแคปซูเลชัน (Encapsulation) เป็นคุณสมบัติที่คู่กับการกำหนดสาระสำคัญ เพราะว่าการกำหนดสาระสำคัญจะมีเฉพาะรายละเอียดที่จำเป็น ส่วนการห่อหุ้มจะเป็นการซ่อนรายละเอียดไว้ภายใน ไม่ให้ผู้ใช้ทราบรายละเอียดต่าง ๆ ที่ไม่จำเป็น ทำให้การพัฒนาโปรแกรมทำได้ง่ายขึ้น เนื่องจากการใช้งานจะต้องใช้ผ่านบริการที่คลาสมีให้เท่านั้น ผู้ใช้บริการไม่สามารถเข้าถึงบริการได้โดยตรง ดังภาพที่ 1.4

ภาพที่ 1.4 การซ่อนรายละเอียด
ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)

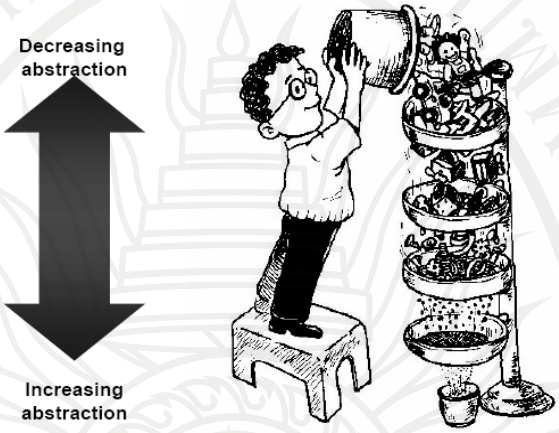
1.2.3 สภาพเป็นส่วนจำเพาะ

สภาพเป็นส่วนจำเพาะหรือโมดูลาริตี (Modularity) เป็นหลักการของการแก้ปัญหาโดยพยายามแบ่งปัญหาใหญ่ออกเป็นส่วนย่อย ๆ เพื่อให้ง่ายต่อการแก้ไข โดยการสร้างส่วนที่เป็นมาตรฐานเพื่อให้สามารถนำส่วนดังกล่าวมาใช้ในการแก้ไขปัญหาในลักษณะเดียวกันได้ ซึ่งในการพัฒนาโปรแกรมคอมพิวเตอร์ส่วนที่เป็นโมดูล สามารถนำมาใช้กับการพัฒนาโปรแกรมใหม่ ๆ ได้อีกด้วย ดังภาพที่ 1.5

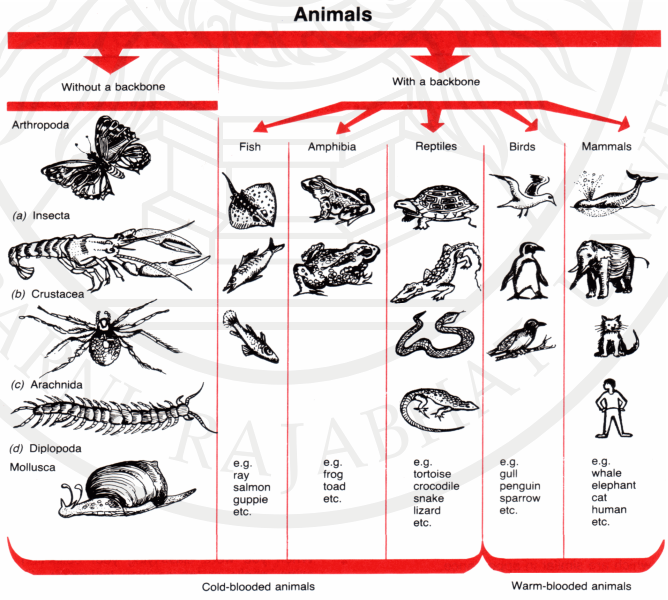
ภาพที่ 1.5 การแยกชิ้นส่วน
ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)

1.2.4 การแบ่งลำดับชั้น

การแบ่งลำดับชั้นหรือไฮราชี (Hierarchy) เป็นลักษณะของการแบ่งชั้นของรายละเอียดวัตถุ เนื่องจากวัตถุแต่ละชนิดอาจจะมีรายละเอียดที่แตกต่างกันออกไป เช่น หัวหน้างานเป็นชนิดหนึ่งของพนักงาน พนักงานเป็นชนิดหนึ่งของคน เป็นต้น ซึ่งการแบ่งลำดับชั้นจะทำให้สามารถแบ่งคลาสออกเป็นลำดับชั้นได้ง่าย และได้คลาสพื้นฐานที่ต้องการ ดังภาพที่ 1.6 และภาพที่ 1.7



ภาพที่ 1.6 การแบ่งลำดับชั้น
ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)



ภาพที่ 1.7 ลำดับชั้นของสัตว์
ที่มา : (Animal Science, n.d.)

1.3 วัตถุ

วัตถุหรืออ็อบเจกต์ (Objects) คือ สิ่งที่อยู่ในระบบซึ่งเป็นผลผลิตของคลาส ประกอบไปด้วย สถานะ (State) และพฤติกรรม (Behavior) ซ่อนอยู่ สามารถเป็นได้ทั้งสิ่งที่จับต้องได้ และจับต้องไม่ได้ ซึ่งสามารถจำแนกชนิดของวัตถุได้ 3 ชนิด (ธวัชชัย งามสันติวงศ์, 2549) ด้วยกัน คือ

1.3.1 วัตถุที่เป็นกายภาพ หมายถึง วัตถุที่อยู่รอบ ๆ ตัวเรา สามารถที่จะมองเห็นและจับต้องได้ เช่น รถยนต์ สมุด โต๊ะ คอมพิวเตอร์ เป็นต้น ดังภาพที่ 1.8

ภาพที่ 1.8 วัตถุที่เป็นกายภาพ
ที่มา : (รูปภาพความละเอียดสูงฟรี, 2556)

1.3.2 วัตถุที่เป็นแนวคิด หมายถึง วัตถุที่มีอยู่จริงสามารถสัมผัสได้ หรือเป็นวัตถุที่ไม่มีตัวตนไม่สามารถสัมผัสได้ เช่น สูตรคณิตศาสตร์ สูตรเคมี เป็นต้น ดังภาพที่ 1.9

ภาพที่ 1.9 วัตถุที่เป็นแนวคิด
ที่มา : (รูปภาพความละเอียดสูงฟรี, 2556)

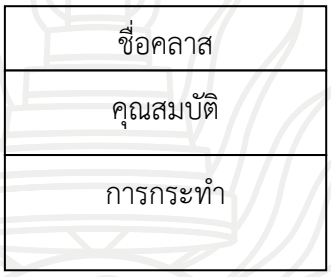
1.3.3 วัตถุที่เป็นซอฟต์แวร์ หมายถึง วัตถุที่มีพฤติกรรมและคุณสมบัติเก็บไว้ภายในเครื่องคอมพิวเตอร์ เป็นวัตถุที่ไม่มีตัวตนไม่สามารถสัมผัสได้ แต่สามารถทำงานผ่านเครื่องคอมพิวเตอร์ได้ ดังภาพที่ 1.10

ภาพที่ 1.10 วัตถุที่เป็นซอฟต์แวร์

1.4 คลาส

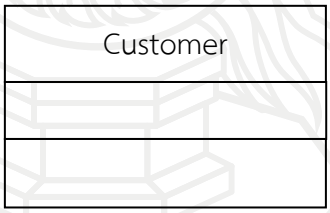
คลาส (Class) คือ กลุ่มของวัตถุที่มีโครงสร้างพื้นฐานและมีพฤติกรรมแบบเดียวกัน โดยวัตถุที่มีคุณสมบัติแบบนี้ก็จะรวมกลุ่มอยู่ในคลาสเดียวกัน เนื่องจากคลาสและวัตถุมีความคล้ายกันมากจนอาจทำให้บางคนคิดว่าเป็นสิ่งเดียวกัน แต่ในความเป็นจริงคลาสถือว่าเป็นนามธรรม ในขณะที่วัตถุเป็นรูปธรรม ซึ่งในการใช้งานจริงจะต้องสร้างวัตถุจากคลาสเสียก่อน จึงจะสามารถใช้งานได้

คลาสจะประกอบไปด้วย ชื่อคลาส (Class name) คุณสมบัติ (Attributes) และการกระทำ (Operations หรือ Methods) ซึ่งสามารถเขียนได้ตามภาพที่ 1.11



ภาพที่ 1.11 การเขียนคลาส

1.4.1 ชื่อคลาส เป็นส่วนที่บอกชื่อที่ใช้เรียกคลาสนั้น ๆ ในการตั้งชื่อคลาสจะเขียนด้วยภาษาอังกฤษ ขึ้นต้นด้วยตัวใหญ่และมีความหมาย ดังภาพที่ 1.12



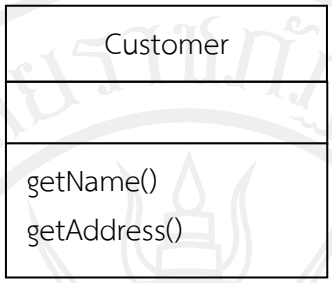
ภาพที่ 1.12 การเขียนชื่อคลาส

1.4.2 คุณสมบัติ คือ คุณสมบัติเฉพาะของกลุ่มของวัตถุซึ่งอยู่ภายในโดเมนที่สนใจ เมื่อสร้างวัตถุจากคลาสแล้ว วัตถุจะมีคุณสมบัติเฉพาะของตนเอง ดังตารางที่ 1.2

ตารางที่ 1.2 คุณสมบัติของผลไม้

ผลไม้	**คุณสมบัติ**	
	รสชาติ	**สี**
แอปเปิล	หวาน	แดง
มะนาว	เปรี้ยว	เหลือง
มะม่วง	มัน	เขียว

1.4.3 การกระทำ เป็นบริการที่คลาสนั้น ๆ มีให้ ซึ่งสามารถให้บริการภายในคลาสเดียวกันหรือต่างคลาสก็ได้ ดังภาพที่ 1.13



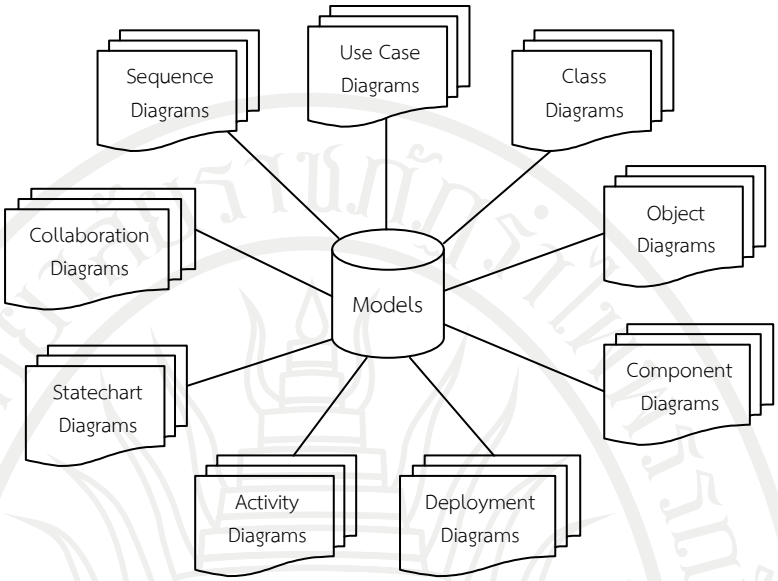
ภาพที่ 1.13 การเขียนการกระทำของคลาส

1.5 ยูเอ็มแอล

ยูเอ็มแอล (Unified modeling language : UML) เกิดมาจากการพัฒนาของ Grady Booch, Ivar Jacobson และ Jim Rumbaugh โดยได้นิยามไว้ว่า ยูเอ็มแอลเป็นสัญลักษณ์ที่ใช้อธิบาย แสดงรายละเอียดจำลองการสร้าง และจัดการเอกสารต่าง ๆ ในระบบ เพื่อให้การออกแบบซอฟต์แวร์สามารถทำได้โดยง่าย และปรับปรุงวิธีการทำงานได้ดีขึ้น ต่อมาบริษัท Rational ได้ร่วมมือให้บุคคลทั้ง 3 ทำการพัฒนาโมเดลร่วมกัน จึงเป็นที่มาของยูเอ็มแอลซึ่งเป็นโมเดลที่สื่อสารด้วยภาพ โดยแต่ละโมเดลจะแสดงมุมมองที่มีต่อระบบแตกต่างกัน ดังนั้นยูเอ็มแอลจึงเป็นระเบียบวิธี (Methodology) หนึ่งเช่นเดียวกับการวิเคราะห์ระบบเชิงโครงสร้างที่ใช้แผนภาพกระแสข้อมูลและอีอาร์ไดอะแกรม (Object Management Group, 2016)

ยูเอ็มแอลไดอะแกรมเป็นแบบจำลองทางสถาปัตยกรรมของระบบที่ประกอบไปด้วยไดอะแกรมต่าง ๆ มากมาย โดยแต่ละไดอะแกรมจะทำให้ทราบถึงมุมมองที่แตกต่างกันออกไป ทำให้เข้าใจระบบงานมากขึ้น แต่ในการพัฒนาระบบจริงอาจไม่จำเป็นต้องใช้ไดอะแกรมทั้งหมดในการอธิบายระบบก็ได้ ซึ่งสามารถพิจารณาใช้เฉพาะไดอะแกรมที่เหมาะสมและเพียงพอกับความต้องการของระบบ โดยยูเอ็มแอลไดอะแกรมประกอบด้วย (Object Oriented Analysis And Design Using UML, n.d.) ดังภาพที่ 1.14

1) ยูสเคสไดอะแกรม (Use case diagram)
2) คลาสไดอะแกรม (Class diagram)
3) อ็อบเจกต์ไดอะแกรม (Object diagram)
4) ซีเควนไดอะแกรม (Sequence diagram)
5) คอลลาโบเรชันไดอะแกรม (Collaboration diagram)
6) สเตทชาร์ทไดอะแกรม (Statechart diagram)
7) แอคทิวิทีไดอะแกรม (Activity Diagram)
8) คอมโพเนนท์ไดอะแกรม (Component diagram)
9) ดีพลอยเมนท์ไดอะแกรม (Deployment diagram)



ภาพที่ 1.14 ไดอะแกรมของยูเอ็มแอล

1.5.1 ยูสเคสไดอะแกรม

เป็นไดอะแกรมที่ถูกพัฒนาขึ้นมาใช้กับการพัฒนาระบบเชิงวัตถุ ซึ่งสามารถบอกส่วนที่สำคัญในระบบ เน้นการการพิจารณาจากมุมมองของผู้ใช้ที่อยู่ในความต้องการที่รวมรวมมาเพื่อพัฒนาระบบ ซึ่งยูสเคสจะแสดงการตอบโต้ระหว่างผู้ใช้กับระบบ โดยมีการเชื่อมโยงความสัมพันธ์กับระบบย่อยต่าง ๆ ซึ่งประกอบไปด้วย 3 ส่วน คือ

1.5.1.1 แอคเตอร์ (Actor) เป็นส่วนที่ใช้แทนผู้กระทำ เป็นได้ทั้งบุคคล หน่วยงาน ซอฟต์แวร์หรือฮาร์ดแวร์ ที่มีปฏิสัมพันธ์กับระบบ สามารถเขียนแทนด้วยสัญลักษณ์ดังภาพที่ 1.15

ภาพที่ 1.15 สัญลักษณ์แอคเตอร์

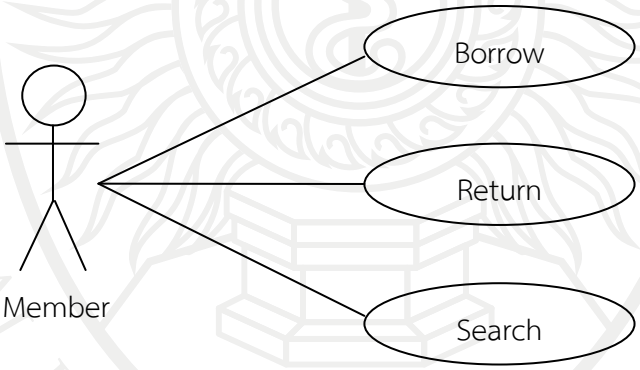
1.5.1.2 ยูสเคส (Use case) คือส่วนที่มองเป็นระบบงานย่อย ๆ ที่อยู่ภายในระบบทั้งหมด จะมีความสัมพันธ์กับแอคเตอร์เพื่อใช้ในการอธิบายระบบให้เข้าใจมากยิ่งขึ้น สามารถเขียนได้ด้วยสัญลักษณ์ดังภาพที่ 1.16

ภาพที่ 1.16 สัญลักษณ์ยูสเคส

1.5.1.3 ความสัมพันธ์ (Relationship) เป็นส่วนที่อธิบายความสัมพันธ์ระหว่างการทำงานของแอคเตอร์กับยูสเคส หรือยูสเคสกับยูสเคส หรือแอคเตอร์กับแอคเตอร์ โดยมีรูปแบบของความสัมพันธ์ดังนี้

ความสัมพันธ์เชิงโครงสร้าง (Association)
แทนด้วย ——————
ความสัมพันธ์แบบสืบทอด (Generalization)
แทนด้วย ——————▷
ความสัมพันธ์แบบพึ่งพา (Dependency)
แทนด้วย ----------

ถึงแม้ว่ายูสเคสจะมีประโยชน์ในการอธิบายภาพรวมของระบบ แต่การระบุยูสเคสที่มากเกินความจำเป็น อาจทำให้ระบบเกิดความยุ่งยากและสับสนได้เช่นกัน ดังนั้นควรมีหลักการในการพิจารณายูสเคสที่ได้จากผู้ใช้งานและสรุป โดยยูสเคสแต่ละยูสเคสนั้นจะต้องอธิบายงานที่ไม่เกี่ยวข้องกัน แต่ถ้าในกรณีที่ระบบการทำงานเหมือนกันปรากฏอยู่ในยูสเคสต่าง ๆ ก็ควรดำเนินการรวบให้อยู่ในยูสเคสเดียวกัน หากมีการนำมาใช้งานก็สามารถเรียกยูสเคสนั้นมาใช้งานได้ ดังภาพที่ 1.17



ภาพที่ 1.17 ตัวอย่างการเขียนยูสเคส

ในการเขียนยูสเคสจะมีซีนาริโอ (Scenario) เป็นตัวกำหนดการทำงานของยูสเคส โดยยูสเคสจะแสดงรายละเอียดต่าง ๆ ในลักษณะตรรกะที่ต้องทำในระบบ ด้วยการพรรณนาถึงกลุ่มความสัมพันธ์ต่าง ๆ ในซีนาริโอ

สรุปได้ว่ายูสเคส ไดอะแกรมนี้จะให้มุมมองภาพรวมของระบบงานต่าง ๆ และบุคคลที่เกี่ยวข้องที่โต้ตอบกับระบบ

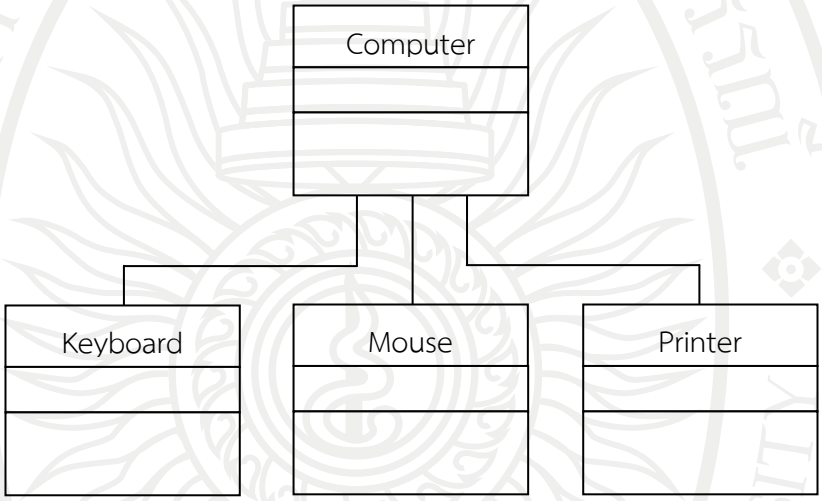
1.5.2 คลาสไดอะแกรม

คลาสไดอะแกรมเป็นไดอะแกรมที่แสดงคลาสต่างๆ และความสัมพันธ์ระหว่างคลาส โดยแต่ละคลาสจะแสดงองค์ประกอบที่มีในระบบ และมีความสัมพันธ์ (Relationship) ในลักษณะต่าง ๆ

เช่น ความสัมพันธ์แบบแอสโซสิเอชัน (Association) แอกกรีเกชัน (Aggregation) คอมโพสิชัน (Composition) และเจนเนอราไลเซชัน (Generalization)

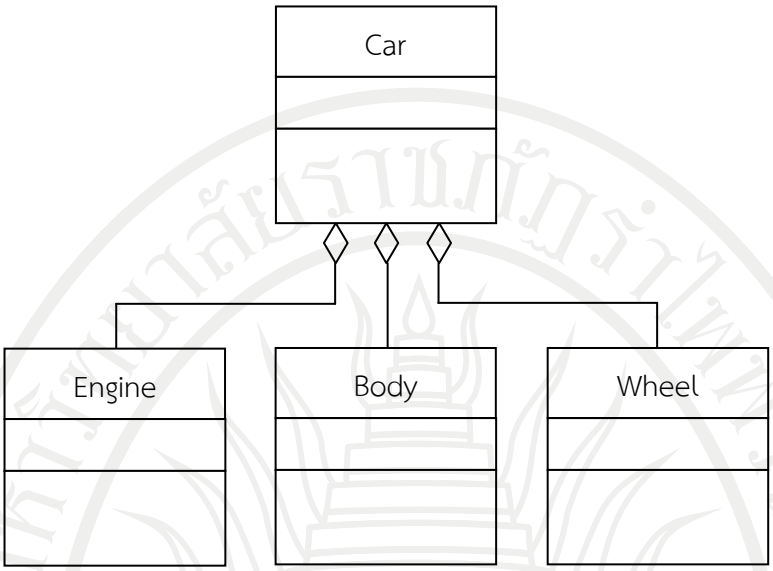
คลาสที่ปรากฏอยู่ในไดอะแกรมนั้นเมื่อนำไปพัฒนาโปรแกรมสามารถประยุกต์สร้างเป็นฐานข้อมูล ส่วนติดต่อกับผู้ใช้ หรือส่วนควบคุม ทั้งนี้ขึ้นอยู่กับการออกแบบของผู้พัฒนาระบบ

1.5.2.1 ความสัมพันธ์แบบแอสโซสิเอชันหรือแบบโครงสร้าง เป็นลักษณะความสัมพันธ์ที่มีการทำงานร่วมกันระหว่างคลาส 2 ทิศทางแบบโฮล-พาร์ท (Whole-Part) ซึ่งถ้าขาดคลาสใดคลาสหนึ่งไปจะทำให้ระบบไม่สามารถทำงานได้ดังภาพที่ 1.18 โดยแบ่งความสัมพันธ์ออกเป็นส่วนย่อย ๆ ได้อีก 2 ประเภท คือ



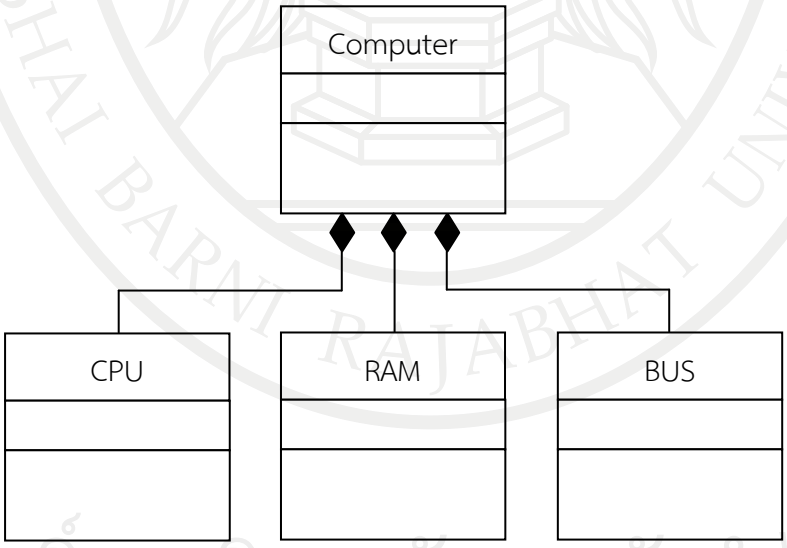
ภาพที่ 1.18 ความสัมพันธ์แบบแอสโซสิเอชัน

(1) ความสัมพันธ์แบบแอกกรีเกชัน เป็นความสัมพันธ์แบบโครงสร้างที่มีลักษณะเป็นองค์ประกอบ โดยมีคลาสที่ใหญ่ที่สุดเป็นหลักและมีคลาสอื่น ๆ เป็นส่วนประกอบทำให้สามารถสร้างคลาสขนาดใหญ่ได้ เช่น คลาสของรถยนต์ ประกอบด้วยคลาสของเครื่องยนต์ คลาสของล้อรถ คลาสของตัวถังรถ ลักษณะของความสัมพันธ์ประเภทนี้จะให้ความสำคัญกับคลาสหลักมาก กล่าวคือถ้าคลาสหลักถูกทำลาย คลาสที่เป็นส่วนประกอบย่อยจะถูกทำลายด้วย แต่ถ้าคลาสที่เป็นส่วนประกอบถูกทำลาย คลาสหลักจะยังสามารถคงอยู่ได้ดังภาพที่ 1.19



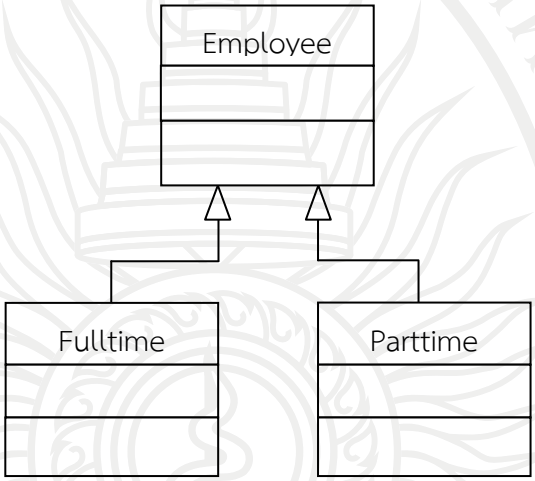
ภาพที่ 1.19 ความสัมพันธ์แบบแอกกรีเกชัน

(2) ความสัมพันธ์แบบคอมโพสิชัน เป็นความสัมพันธ์แบบโครงสร้างในลักษณะขององค์ประกอบ โดยมีความแตกต่างจากแอกกรีเกชันตรงที่คลาสที่เป็นคลาสหลักและคลาสย่อยจะมีความสำคัญเท่ากัน กล่าวคือ ไม่ว่าคลาสหลักหรือคลาสย่อยถูกทำลาย คลาสที่เป็นองค์ประกอบก็จะถูกทำลายไปด้วย เช่น คลาสของคอมพิวเตอร์ ประกอบไปด้วยคลาสของซีพียู คลาสของแรม คลาสของบัส เป็นต้น ดังภาพที่ 1.20



ภาพที่ 1.20 ความสัมพันธ์แบบคอมโพสิชัน

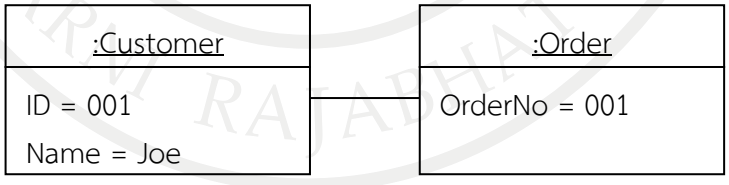
1.5.2.2 ความสัมพันธ์แบบเจนเนอราไลเซชันหรือแบบสืบทอด เป็นความสัมพันธ์ระหว่างคลาสที่อยู่ในลักษณะของการสืบทอดจากคลาสแม่ไปยังอีกคลาสลูก โดยอาจจะเรียกคลาสที่เป็นต้นแบบว่า คลาสพ่อหรือคลาสแม่ ส่วนคลาสที่สืบทอดอาจจะเรียกว่า คลาสลูก เมื่อมีการสืบทอดแล้ว คลาสลูกจะมีคุณสมบัติและการกระทำเหมือนกับคลาสแม่ทุกประการ และยังสามารถเพิ่มคุณสมบัติและการกระทำที่แตกต่างจากคลาสแม่ได้อีกด้วย การสร้างความสัมพันธ์แบบสืบทอดยังช่วยให้การพัฒนาโปรแกรมง่ายขึ้น เนื่องจากเมื่อเปลี่ยนคุณสมบัติหรือการกระทำจากคลาสแม่ คลาสลูกก็จะเปลี่ยนตามไปด้วยดังภาพที่ 1.21



ภาพที่ 1.21 ความสัมพันธ์แบบเจนเนอราไลเซชัน

1.5.3 อ็อบเจกต์ไดอะแกรม

เป็นไดอะแกรมที่คล้ายกับคลาสไดอะแกรม จะประกอบไปด้วยวัตถุและความสัมพันธ์ระหว่างวัตถุ การเขียนความสัมพันธ์ของอ็อบเจกต์ไดอะแกรมจะเขียนเหมือนกับคลาสไดอะแกรม คือ แอสโซสิเอชัน แอกกรีเกชัน คอมโพสิชัน และเจนเนอราไลเซชันดังภาพที่ 1.22

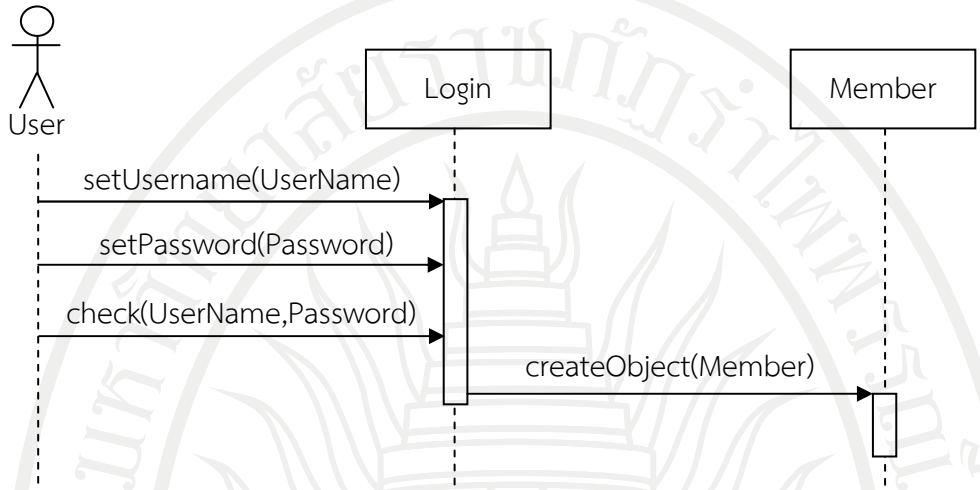


ภาพที่ 1.22 ตัวอย่างอ็อบเจกต์ไดอะแกรม

1.5.4 ซีเควนไดอะแกรม

เป็นไดอะแกรมที่ใช้ในการอธิบายขั้นตอนการทำงานของแต่ละยูสเคส ระหว่างคลาสหรืออออบเจ็กต์ต่าง ๆ ที่ส่งข้อความ (Message) ถึงกันและกัน ซึ่งจะช่วยให้ผู้เขียนโปรแกรมสามารถ

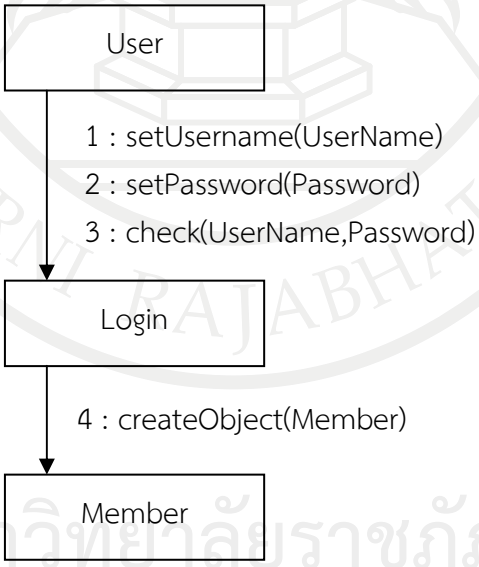
มองเห็นภาพรวมของขั้นตอนในการทำงานระหว่างวัตถุที่มีความสัมพันธ์กันในแต่ละยูสเคส ข้อความที่วัตถุส่งให้กันจะเป็นการกระทำที่วัตถุนั้น ๆ มีบริการให้ดังภาพที่ 1.23



ภาพที่ 1.23 ตัวอย่างซีเควนไดอะแกรมการเข้าสู่ระบบ

1.5.5 คอลลาโบเรชันไดอะแกรม

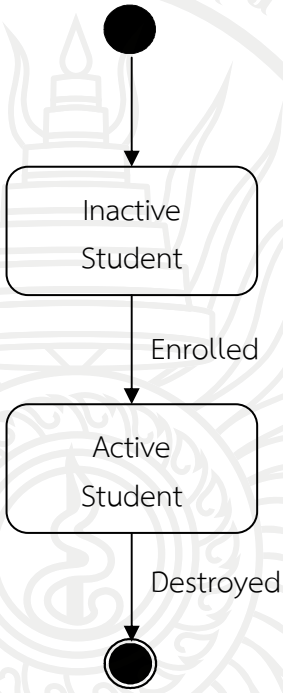
เป็นไดอะแกรมคล้ายกับซีเควนไดอะแกรม โดยซีเควนไดอะแกรมเป็นไดอะแกรมที่แสดงถึงลำดับในการทำงานโดยมีการแลกเปลี่ยนข่าวสารซึ่งกันและกันที่มีเวลาเกี่ยวข้องด้วย แต่สำหรับคอลลาโบเรชันไดอะแกรมจะแสดงความสัมพันธ์ระหว่างออบเจ็กต์และความสัมพันธ์ โดยจะแสดงลำดับการทำงานก่อนและหลังดังภาพที่ 1.24



ภาพที่ 1.24 ตัวอย่างคอลลาโบเรชันไดอะแกรมการเข้าสู่ระบบ

1.5.6 สเตทชาร์ทไดอะแกรม

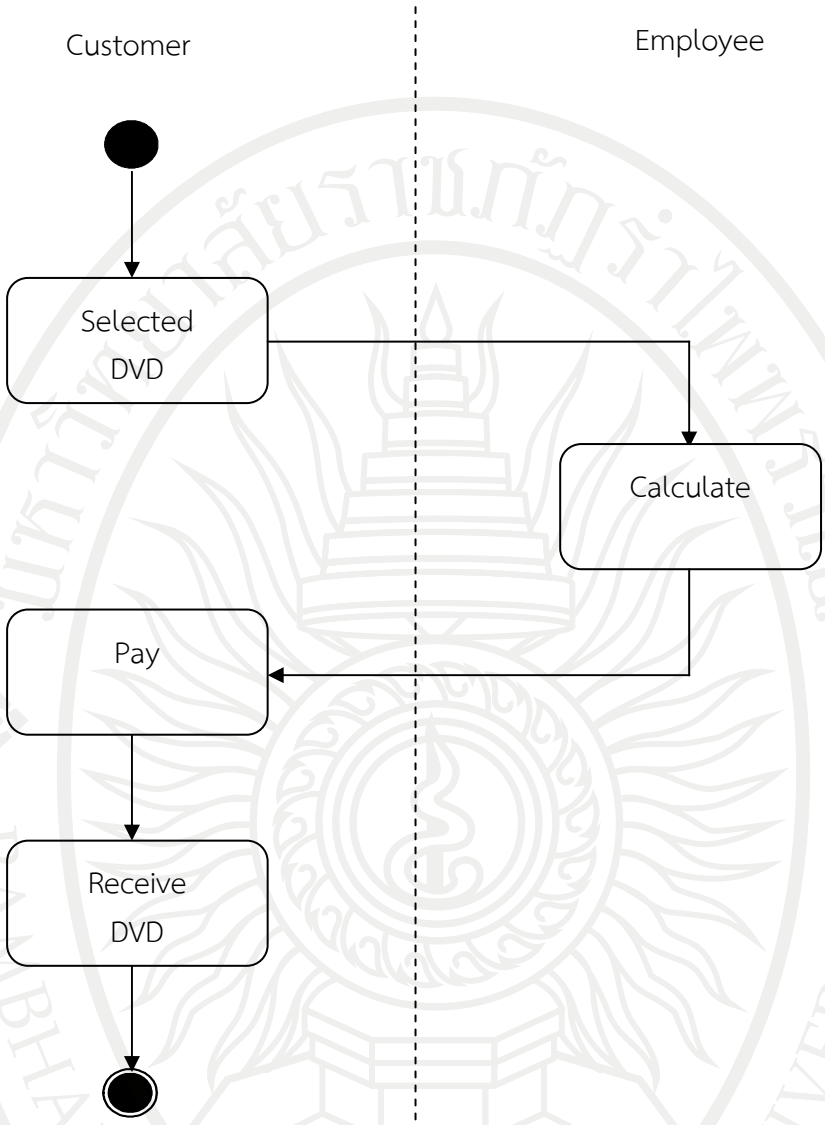
หรืออาจจะเรียกว่าสเตทไดอะแกรม (State Diagram) เป็นไดอะแกรมที่แสดงสถานะ (State) ของวัตถุที่มีอยู่ในระบบหรือโดเมนที่สนใจ การเปลี่ยนสถานะของวัตถุจะเกิดจากเหตุการณ์ต่าง ๆ ที่เกิดขึ้น การเขียนสเตทไดอะแกรมนั้นจะเขียนอธิบายเฉพาะวัตถุที่มีการเปลี่ยนสถานะเท่านั้น ดังภาพที่ 1.25



ภาพที่ 1.25 ตัวอย่างสเตทไดอะแกรมการลงทะเบียนเรียน

1.5.7 แอคทิวิทีไดอะแกรม

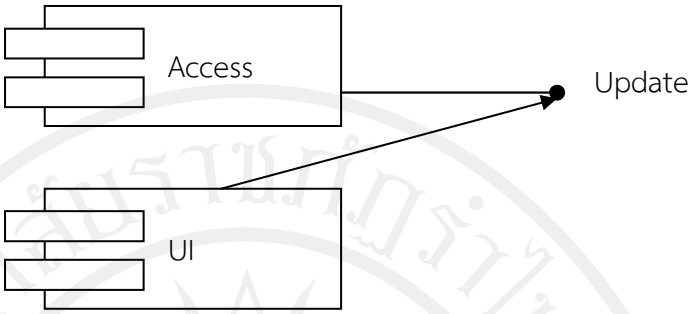
เป็นไดอะแกรมแสดงกิจกรรมหรือขั้นตอนในการทำงาน ซึ่งจะมีลักษณะคล้ายกับผังงาน (Flowchart) แต่จะแตกต่างกันตรงที่แอคทิวิทีไดอะแกรมจะสามารถบอกได้ว่ากิจกรรมเกิดขึ้นที่ใดหรือใครเป็นผู้ทำกิจกรรมนั้น และผลจากการทำงานในขั้นตอนต่าง ๆ ในระบบ ดังภาพที่ 1.26



ภาพที่ 1.26 ตัวอย่างแอคทิวิทีไดอะแกรมการซื้อขายดีวีดี (DVD)

1.5.8 คอมโพเนนท์ไดอะแกรม

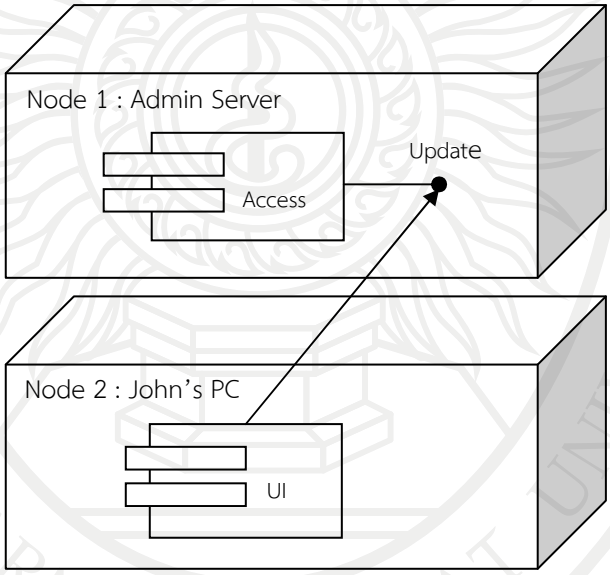
คอมโพเนนท์ไดอะแกรมเป็นไดอะแกรมแสดงลักษณะโครงสร้างทางกายภาพ (Physical) ในส่วนที่มีความเกี่ยวข้องกับองค์ประกอบของซอฟต์แวร์ (Software component) เช่น ชุดคำสั่ง (Source code) เอ็กซ์คิวโปรแกรม (Executable program) และส่วนติดต่อกับผู้ใช้ (User interface) โดยที่คอมโพเนนท์ไดอะแกรมจะแสดงองค์ประกอบต่าง ๆ ของระบบที่มีการเชื่อมโยงกันโดยใช้ความสัมพันธ์แบบพึ่งพา ดังภาพที่ 1.27



ภาพที่ 1.27 ตัวอย่างคอมโพเนนท์ไดอะแกรม

1.5.9 ดีพลอยเมนท์ไดอะแกรม

ดีพลอยเมนท์ไดอะแกรมเป็นไดอะแกรมที่แสดงที่ตั้ง (Configuration) ของส่วนประมวลผล รวมทั้งองค์ประกอบของซอฟต์แวร์ต่าง ๆ ตลอดจนความสัมพันธ์หรืออินเตอร์เฟสระหว่างฮาร์ดแวร์และซอฟต์แวร์ ทำให้ทราบถึงภาพรวมของระบบทั้งหมด ดังภาพที่ 1.28



ภาพที่ 1.28 ตัวอย่างดีพลอยเมนท์ไดอะแกรม

1.6 สรุป

แนวคิดเชิงวัตถุ หมายถึง การมองทุกสิ่งในระบบเป็นวัตถุทั้งหมดและใช้วัตถุเป็นตัวหลักในการพิจารณาสภาพความเป็นจริงที่เกิดขึ้นในระบบ ซึ่งทุก ๆ กิจกรรมที่เกิดขึ้นภายในระบบเกิดจากความสัมพันธ์และปฏิสัมพันธ์ระหว่างวัตถุ การพัฒนาระบบเชิงวัตถุเป็นการพัฒนาระบบที่ช่วยแก้ไขปัญหาความสลับซับซ้อน โดยมองทุกอย่างที่อยู่ในระบบเป็นวัตถุ และกลุ่มของวัตถุที่เรียกว่าคลาส คลาสจะทำหน้าที่เป็นแม่แบบสำหรับการสร้างวัตถุ ภายในคลาสประกอบไปด้วยชื่อคลาส คุณสมบัติ

และพฤติกรรม ที่บ่งบอกถึงลักษณะเฉพาะของแต่ละคลาส โดยคลาสสามารถเชื่อมโยงกันได้ด้วยความสัมพันธ์แบบต่าง ๆ และสามารถสืบทอดความสัมพันธ์จากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้

ในการวิเคราะห์ระบบเชิงวัตถุสามารถใช้ยูเอ็มแอลช่วยสร้างแบบจำลองในการวิเคราะห์ เพราะยูเอ็มแอลมีไดอะแกรมที่แสดงมุมมองของระบบในแบบต่าง ๆ เช่น ยูสเคสไดอะแกรม คลาสไดอะแกรม อ็อบเจกต์ไดอะแกรม ซีเควนไดอะแกรม คอลลาโบเรชันไดอะแกรม สเตทชาร์ทไดอะแกรม แอคทิวิทีไดอะแกรม คอมโพเนนท์ไดอะแกรม และสดีพลอยเมนท์ไดอะแกรม ซึ่งจะทำให้เข้าใจระบบงานมากขึ้น แต่การพัฒนาระบบนั้น ไม่จำเป็นต้องใช้ไดอะแกรมของยูเอ็มแอลทั้งหมด สามารถเลือกใช้ได้ตามความเหมาะสมของระบบ

แบบฝึกหัดบทที่ 1

1. จงอธิบายแนวคิดและการพัฒนาระบบเชิงวัตถุ
2. การพัฒนาระบบเชิงวัตถุแตกต่างจากการพัฒนาระบบเชิงโครงสร้างอย่างไร
3. วัตถุและคลาสมีความแตกต่างกันอย่างไร
4. ยูเอ็มแอลประกอบด้วยไดอะแกรมอะไรบ้าง
5. ความสัมพันธ์ระหว่างแอคเตอร์กับยูสเคสมีกี่แบบ อะไรบ้าง
6. ความสัมพันธ์แบบแอกกรีเกชันกับคอมโพสิชันต่างกันอย่างไร
7. ไดอะแกรมที่เขียนคล้ายกับการเขียนผังงานคือไดอะไรแกรมอะไร
8. ซีเควนไดอะแกรมใช้ในการอธิบายอะไร
9. คลาสไดอะแกรมและซีเควนไดอะแกรมมีความสัมพันธ์กันอย่างไร
10. จงสร้างยูสเคสไดอะแกรมเพื่ออธิบายการลงทะเบียนเรียนของนักศึกษา ซึ่งเกิดจากผลของการวิเคราะห์ความต้องการเบื้องต้น สามารถเขียนเป็นรายการได้ดังนี้
    1) ในแต่ละภาคการศึกษาจะมีการลงทะเบียนของนักศึกษา
    2) การลงทะเบียนในแต่ละครั้งจะมีการเก็บหลักฐานและค่าลงทะเบียนเรียน
    3) เจ้าหน้าที่ของสถาบันการศึกษาจะเป็นผู้จัดการในเรื่องของการจัดเก็บหลักฐานและค่าลงทะเบียนเรียนทั้งหมด และผู้จ่ายเงินต้องเป็นนักศึกษาเท่านั้น

เอกสารอ้างอิง

กิตติพงษ์ กลมกล่อม. (2552). **การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML.** กรุงเทพฯ : เคทีพี.

ธวัชชัย งามสันติวงศ์. (2549). **การวิเคราะห์และออกแบบระบบงานเชิงวัตถุ.** กรุงเทพฯ : ศูนย์หนังสือจุฬาลงกรณ์มหาวิทยาลัย.

รูปภาพความละเอียดสูงฟรี.(2556). (ออนไลน์). แหล่งที่มา : https://pixabay.com. 15 มกราคม 2559.

Object Management Group. (2016). **WHAT IS UML**. (online). Available : http://www.uml.org/what-is-uml.htm. 10 January 2016.

Animal Science. (n.d.). (online). Available : https://www.pinterest.com/libertyp/animal-science-mixed-grade/. 15 January 2016.

Object Oriented Analysis And Design Using UML. (n.d.). (online). Available : http://www.fritzsolms.net/sites/default/files/documents/ObjectOrientedAnalysisAndDesignUsingUML.pdf. 15 January 2016.

แผนบริหารประจำบทที่ 2

เนื้อหา

บทที่ 2 หลักการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ

2.1 ภาษาโปรแกรมเชิงวัตถุ
2.2 แนวความคิดภาษาเชิงวัตถุ
2.3 คุณสมบัติภาษาเชิงวัตถุ
2.4 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. อธิบายหลักการของภาษาโปรแกรมเชิงวัตถุ
2. อธิบายแนวความคิดของภาษาเชิงวัตถุ
3. อธิบายคุณสมบัติของภาษาเชิงวัตถุ

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 2

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. โปรแกรมวิชวลสตูดิโอ 2010
4. แบบฝึกหัดบทที่ 2
5. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. ประเมินจากแบบฝึกหัดบทที่ 2

บทที่ 2
หลักการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ

การพัฒนาโปรแกรมในปัจจุบันมีความซับซ้อนมาก นักพัฒนาจำเป็นจะต้องหาวิธีที่จะทำให้การพัฒนาโปรแกรมทำได้ง่ายขึ้นและเกิดประสิทธิภาพสูงสุด แนวคิดการเขียนโปรแกรมเชิงวัตถุจึงเกิดขึ้น ซึ่งมีความแตกต่างกับการพัฒนาโปรแกรมอดีต โดยมีการรวบรวมข้อดีของการพัฒนาโปรแกรมเชิงกระบวนการไว้ และยังเพิ่มการจัดการกับวิธีการโดยเน้นการทำงานให้อยู่ในรูปของการมองทุกอย่างเป็นวัตถุที่มีลักษณะเฉพาะตน เชื่อมต่อกันด้วยความสัมพันธ์ และสามารถจัดการบำรุงรักษาได้ง่าย

2.1 ภาษาโปรแกรมเชิงวัตถุ

ในการพัฒนาระบบโดยทั่วไปนั้น ระบบที่มีความซับซ้อนจะประกอบไปด้วยวัตถุเป็นจำนวนมากและมีความสัมพันธ์เกี่ยวข้องกัน การจัดการกับความซับซ้อนนี้จะต้องมีการแยกวัตถุออกจากกันโดยสร้างแบบจำลองการทำงานของวัตถุแต่ละชนิดไปเรื่อย ๆ ซึ่งการพัฒนาโปรแกรมรูปแบบนี้จะต้องสร้างโปรแกรมที่จำลองวัตถุแต่ละชนิดไปทีละขั้นตอนจนกว่าจะได้คำตอบ ทำให้เกิดแนวคิดที่เรียกว่าการพัฒนาโปรแกรมเชิงวัตถุ โดยภาษาที่ใช้ในการพัฒนาโปรแกรมเชิงวัตถุภาษาแรกคือ ภาษาซิมูลา (Simula) เกิดเมื่อปี ค.ศ.1960 ต่อมาในปี ค.ศ.1967 ภาษาซิมูลา 67 (Simula67) จึงถูกพัฒนาขึ้นที่ประเทศนอร์เวย์ เพื่อช่วยในการพัฒนาโปรแกรม และในปี ค.ศ.1970 ภาษาสมอลล์ทอล์ค (Smalltalk) ก็กำเนิดขึ้นและเข้ามาแทนที่ภาษาซิมูลาทำให้หลายคนเข้าใจว่าภาษาสมอลล์ทอร์คเป็นภาษาเชิงวัตถุอย่างแท้จริง การพัฒนาโปรแกรมคอมพิวเตอร์โดยทั่วไปจะมี 2 แบบ คือ

2.1.1 การพัฒนาโปรแกรมเชิงฟังก์ชั่น

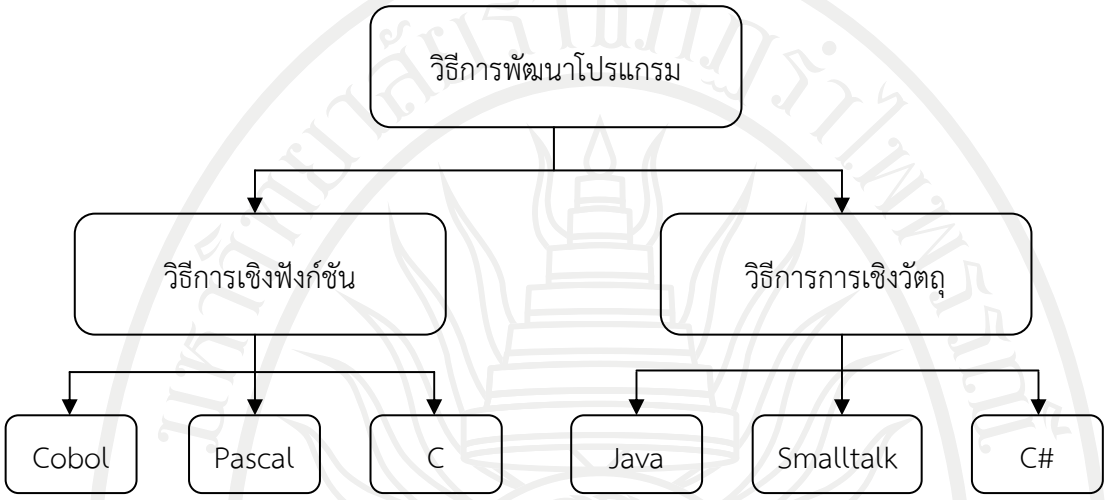
เป็นการพัฒนาโปรแกรมที่มีการเขียนคำสั่งเรียงต่อกันไปเรื่อย ๆ ทีละบรรทัด โดยที่โปรแกรมจะเริ่มทำงานตั้งแต่คำสั่งแรกไปจนถึงคำสั่งสุดท้าย ซึ่งอาจจะมีโปรแกรมย่อยที่เขียนแยกออกมาเป็นฟังก์ชัน เพื่อลดการทำงานที่ซ้ำ ๆ กันออกไป ตัวอย่างของภาษาเชิงฟังก์ชัน ได้แก่ ภาษาโคบอล (Cobol) ภาษาฟอร์แทรน (Fortran) ภาษาปาสคาล (Pascal) ภาษาซี (C) เป็นต้น

2.2.2 การพัฒนาโปรแกรมเชิงวัตถุ

เป็นการพัฒนาโปรแกรมที่มีการสมมุติสิ่งต่าง ๆ ที่อยู่ในระบบให้เป็นวัตถุขึ้นมาก่อน จากนั้นจึงเขียนคำสั่งให้วัตถุนั้นทำงานตามต้องการ คำสั่งที่อยู่ในโปรแกรมเชิงวัตถุจะเขียนเรียงต่อกันอยู่ภายในวัตถุแต่ละชนิด และโปรแกรมสามารถทำงานได้เองถ้าวัตถุนั้นถูกนิยามขึ้นมาอย่างเหมาะสม ตัวอย่างของภาษาเชิงวัตถุ ได้แก่ ภาษาซีพลัสพลัส (C++) ภาษาจาวา (Java) ภาษาซีชาร์ป (C#) เป็นต้น (ธีระพล ลิ้มศรัทธา, 2553)

การพัฒนาโปรแกรมเชิงวัตถุ ผู้พัฒนาอาจจะต้องใช้เวลาในการศึกษานานพอสมควร เนื่องจากจะต้องสร้างวัตถุสมมุติตามที่ต้องการแล้วยังต้องสร้างความสัมพันธ์ต่าง ๆ ที่เกิดขึ้นระหว่าง

วัตถุอีกด้วย ทำให้ผู้ที่ไม่มีความชำนาญพออาจจะออกแบบวัตถุได้ไม่ตรงกับความต้องการของระบบ จึงจำเป็นอย่างยิ่งที่ผู้พัฒนาควรจะศึกษาและฝึกฝนการพัฒนาโปรแกรมเชิงวัตถุให้ชำนาญ เนื่องจากโปรแกรมประยุกต์ที่ใช้งานในปัจจุบันเป็นโปรแกรมที่พัฒนาด้วยโปรแกรมเชิงวัตถุทั้งสิ้น ดังภาพที่ 2.1



ภาพที่ 2.1 วิธีการพัฒนาโปรแกรม

การพัฒนาโปรแกรมด้วยวิธีการเชิงวัตถุจะมองภาพรวมของระบบซึ่งต่างจากวิธีการเชิงฟังก์ชัน ทำให้สามารถแก้ไขความซับซ้อนของระบบได้ดีกว่าและยังทำให้การปรับปรุงแก้ไขโปรแกรมทำได้ง่ายกว่าด้วย ดังตารางที่ 2.1

ตารางที่ 2.1 การเปรียบเทียบระหว่างการพัฒนาโปรแกรมเชิงฟังก์ชันและเชิงวัตถุ

คุณสมบัติ	วิธีการเชิงฟังก์ชัน	วิธีการเชิงวัตถุ
ลักษณะทั่วไป	นำปัญหามาแตกเป็นส่วนย่อย ให้อยู่ในรูปของกระบวนการทำงาน	มองสิ่งต่าง ๆ ในระบบเป็นวัตถุที่มีความเป็นอิสระต่อกัน แต่สามารถทำงานร่วมกันได้
ลักษณะการจำแนกงาน	แตกกระบวนการทำงานออกเป็นหน่วยย่อยที่เรียกว่า ฟังก์ชัน	จำแนกวัตถุออกเป็นกลุ่มตามคุณลักษณะของวัตถุ
ความขึ้นต่อกัน	ฟังก์ชันการทำงานจะทำงานขึ้นตรงต่อกัน มีการส่งพารามิเตอร์จากฟังก์ชันหนึ่งไปยังอีกฟังก์ชันหนึ่ง	แต่ละวัตถุมีความเป็นอิสระไม่ขึ้นตรงต่อกัน และติดต่อกันผ่านทางข้อความ
ขั้นตอนการทำงาน	เริ่มต้นที่การกำหนดโครงสร้างและประเภทของข้อมูล เพื่อใช้ในการทำงาน	เริ่มต้นที่การกำหนดคุณสมบัติและพฤติกรรมของวัตถุ และสร้างความสัมพันธ์ระหว่างวัตถุให้ทำงานร่วมกัน

ที่มา : (ธวัชชัย งามสันติวงศ์, 2549)

2.2 แนวความคิดภาษาเชิงวัตถุ

ในการแก้ไขปัญหาในระบบเชิงวัตถุจะมุ่งเน้นการมองปัญหาและองค์ประกอบของปัญหาในลักษณะที่เรียกว่าพรอบเบรมสเปซ (Problem space) ซึ่งเหมือนการจำลองตามสภาพความเป็นจริงในชีวิตความเป็นอยู่ของมนุษย์ที่ประกอบไปด้วยสิ่งต่าง ๆ เช่น คน สัตว์ ต้นไม้ สิ่งของ และใช้สิ่งเหล่านั้นในการแก้ไขปัญหา ตัวอย่างของพรอบเบรมสเปซดังภาพที่ 2.2 (กิตติพงษ์ กลมกล่อม, 2552)

ภาพที่ 2.2 ตัวอย่างของพรอบเบรมสเปซ

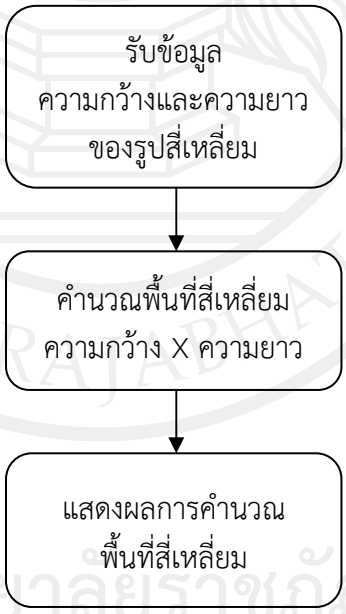
ตัวอย่าง ปัญหาและการมองปัญหาการหาพื้นที่รูปสี่เหลี่ยมผืนผ้าใด ๆ

การมองปัญหาแบบเก่า

1) การหาพื้นที่สี่เหลี่ยมผืนผ้าสามารถหาได้โดยใช้สูตร

พื้นที่สี่เหลี่ยม = กว้าง x ยาว

2) การแก้ปัญหาจะต้องทราบ ความกว้าง และ ความยาว ของสี่เหลี่ยมเสียก่อน และเมื่อนำไปพัฒนาโปรแกรม จะมีขั้นตอนกระบวนการดังภาพที่ 2.3



ภาพที่ 2.3 แผนผังการแก้ปัญหาพื้นที่สี่เหลี่ยม

การมองปัญหาแบบใหม่

1) วิเคราะห์หาวัตถุที่เกี่ยวข้องกับปัญหาที่อยู่ในระบบ เช่น สี่เหลี่ยม ความกว้าง ความยาว และพื้นที่

2) จัดกลุ่มของวัตถุที่มีลักษณะเหมือนกัน

3) วิเคราะห์ลักษณะของวัตถุ เช่น สี่เหลี่ยมควรมีส่วนประกอบที่เป็นคุณสมบัติที่บอกลักษณะเฉพาะของวัตถุ คือ ความกว้าง ความยาว และพื้นที่

4) วิเคราะห์พฤติกรรมของวัตถุ เช่น สี่เหลี่ยมต้องสามารถคำนวณพื้นที่ เส้นรอบรูป เป็นต้น

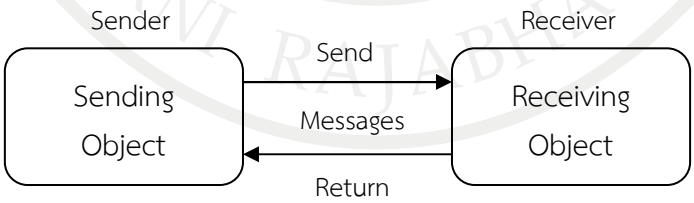
เมื่อวิเคราะห์ข้อมูลทั้งหมดของวัตถุแล้ว จะนำมาสร้างเป็นคลาสตามข้อมูลที่เกิดจากการวิเคราะห์ ดังภาพที่ 2.4

ภาพที่ 2.4 ตัวอย่างการวิเคราะห์คลาสของสี่เหลี่ยม

แนวคิดของเอลัน เคย์ (Alan Kay) ซึ่งเป็นผู้หนึ่งที่ร่วมพัฒนาภาษาสมอลล์ทอร์ค ได้ให้นิยามของภาษาเชิงวัตถุไว้ดังนี้

1) ทุกสิ่งในระบบเป็นวัตถุ หมายถึง องค์ประกอบของโปรแกรมคอมพิวเตอร์ทุก ๆ ส่วน จะต้องเป็นวัตถุ ซึ่งจากกฎข้อนี้ทำให้มีภาษาส่วนมากไม่เป็นภาษาเชิงวัตถุแบบบริสุทธิ์

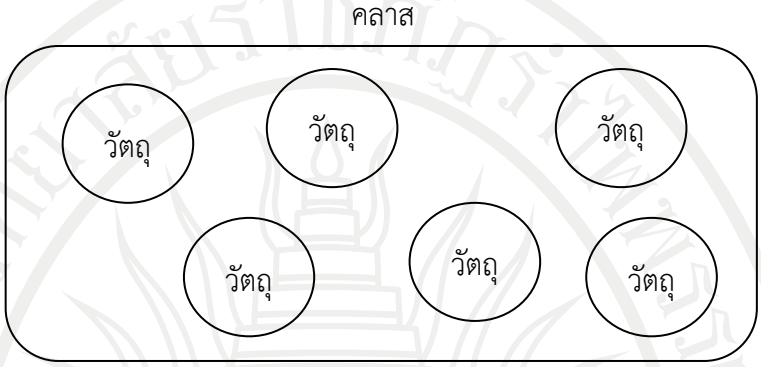
2) โปรแกรม คือ กลุ่มของวัตถุที่ส่งข้อความบอกกันเพื่อทำงาน ดังภาพที่ 2.5



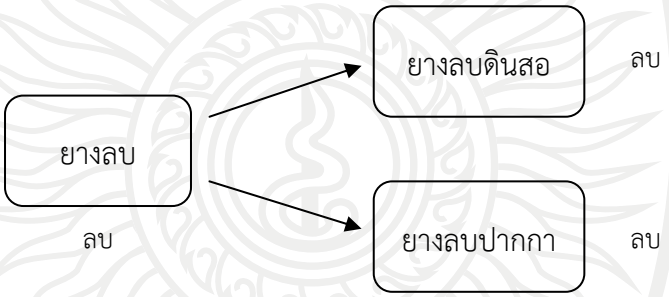
ภาพที่ 2.5 การทำงานร่วมกันของวัตถุ

3) วัตถุแต่ละชนิดมีความจำ คือ ส่วนที่ใช้ในการเก็บข้อมูลของวัตถุนั้น ๆ ซึ่งก็คือ คุณสมบัติของวัตถุนั่นเอง

4) วัตถุต้องมีชนิด ซึ่งหมายถึง คลาสนั่นเอง ในการเขียนโปรแกรมเชิงวัตถุจะต้องสร้างคลาสขึ้นมาก่อน แล้วจึงสร้างวัตถุจากคลาส ดังนั้นวัตถุจะต้องเกิดจากคลาสใดคลาสหนึ่ง วัตถุที่เกิดจากคลาสเดียวกันจะมีคุณสมบัติพื้นฐานเหมือนกัน ดังภาพที่ 2.6 และ 2.7

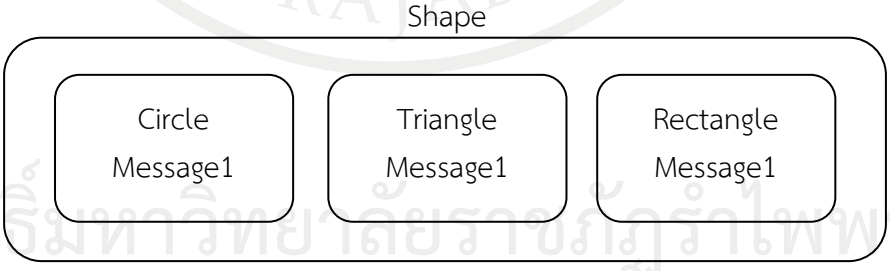


ภาพที่ 2.6 วัตถุต้องเกิดจากคลาส



ภาพที่ 2.7 คุณสมบัติของวัตถุที่เกิดจากคลาสเดียวกัน

5) วัตถุที่อยู่ในประเภทเดียวกันจะได้รับข่าวสารเหมือนกัน จากคุณสมบัติในข้อนี้ทำให้การเขียนโปรแกรมเชิงวัตถุมีประสิทธิภาพเพิ่มขึ้น เนื่องจากสามารถจัดกลุ่มของวัตถุชนิดเดียวกันได้ และผู้พัฒนาก็สามารถกำหนดชนิดย่อยของวัตถุได้อีก เช่น คลาส shape สามารถสร้างเป็นคลาส Circle Triangle และ Rectangle ได้ และคลาสทั้ง 3 ยังได้รับข่าวสารเหมือนกันอีกด้วย ดังภาพที่ 2.8

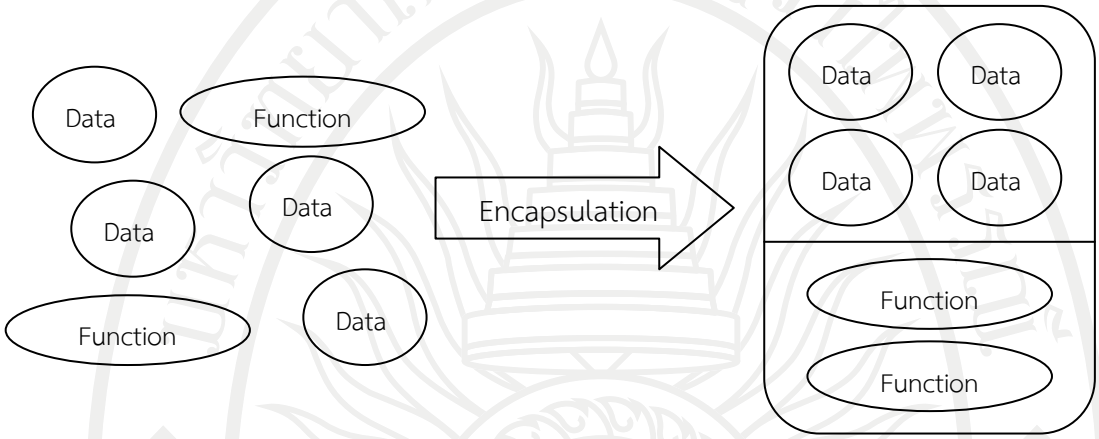


ภาพที่ 2.8 การได้รับข่าวสารของคลาส

2.3 คุณสมบัติของภาษาโปรแกรมเชิงวัตถุ

ภาษาที่เป็นใช้สำหรับการพัฒนาโปรแกรมเชิงวัตถุควรมีคุณสมบัติพื้นฐานดังต่อไปนี้ (ศุภชัย สมพานิช, 2556)

2.3.1 การห่อหุ้ม (Encapsulation) คือ การรวมคุณสมบัติและพฤติกรรมของวัตถุเข้าด้วยกัน โดยกำหนดเป็นชนิดของวัตถุ ดังภาพที่ 2.9



ภาพที่ 2.9 การห่อหุ้ม

ตัวอย่างของการเขียนโปรแกรมลักษณะของการห่อหุ้มเป็นดังนี้

```
1    class User
2    {
3        private string _Name;
4
5        public string Name
6        {
7            get { return _Name; }
8            set { _Name = value; }
9        }
10   }
```

โดยในบรรทัดที่ 3 เป็นส่วนของข้อมูลหรือดาต้า (Data) และบรรทัดที่ 5 – 10 เป็นส่วนของการทำงานหรือฟังก์ชัน (Function)

2.3.2 การซ่อนรายละเอียด (Data Hiding) เป็นการกำหนดระดับในการเข้าถึงข้อมูลเพื่อป้องกันการเข้ามาใช้ข้อมูลโดยวัตถุอื่น ๆ ที่ไม่เกี่ยวข้อง ซึ่งอาจจะทำให้เกิดความผิดเพี้ยนของข้อมูล ดังนั้นข้อมูลบางอย่างที่ไม่ยอมให้วัตถุอื่นมาใช้งานจะต้องมีการซ่อนรายละเอียดเพื่อทำให้ข้อมูลมีความปลอดภัยในการทำงาน ซึ่งสามารถกำหนดกลไกป้องกันข้อมูลและวิธีการทำงานของวัตถุได้ตามตัวอย่างภาพที่ 2.10

1) public (+) หมายถึง สามารถเข้าถึงได้โดยตรงจากคลาสภายนอก
2) private (-) หมายถึง สามารถเข้าถึงหรือถูกใช้งานจากภายในคลาสเท่านั้น
3) protected (#) หมายถึง สามารถเห็นหรือเข้าถึงได้จากภายในคลาสย่อยเท่านั้น

ภาพที่ 2.10 การซ่อนรายละเอียด

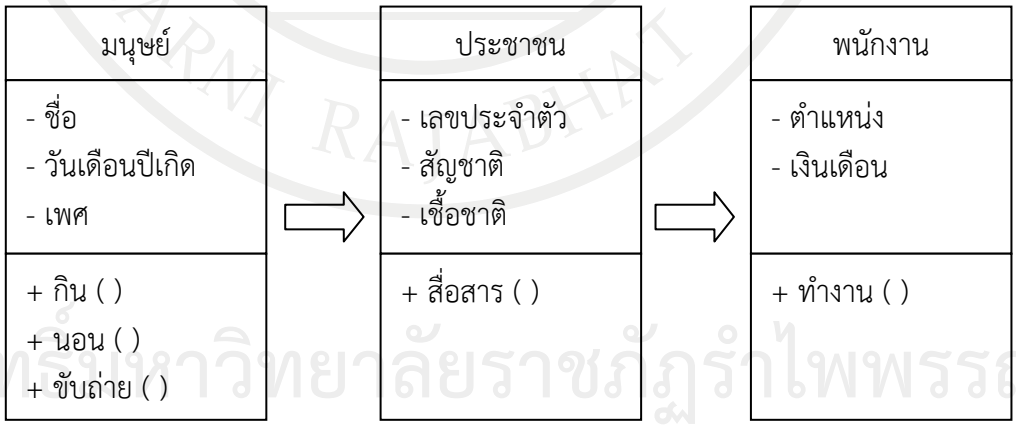
จากภาพที่ 2.10 คลาสอื่น ๆ จากภายนอกจะสามารถเข้าถึงส่วนที่เป็นการทำงานที่เรียกว่าฟังก์ชันหรือเมธอด (Method) เท่านั้น ในส่วนที่เป็นคุณสมบัติไม่สามารถเข้าถึงได้

ตัวอย่างของการเขียนโปรแกรมลักษณะของการซ่อนรายละเอียดเป็นดังนี้

```
1     public class User
2     {
3         public string Name { get; set; }
4     }
```

ในบรรทัดที่ 1 และ 3 จะมีการกำหนดระดับการเข้าถึงข้อมูลให้เป็นแบบสาธารณะ

2.3.3 การสืบทอด (Inheritance) เป็นการถ่ายทอดคุณสมบัติหรือความสามารถของวัตถุชิ้นหนึ่งไปยังวัตถุอีกชิ้นหนึ่ง โดยมีลักษณะพื้นฐานตามของเดิมและเพิ่มสิ่งใหม่เข้าไปได้ดังภาพที่ 2.11



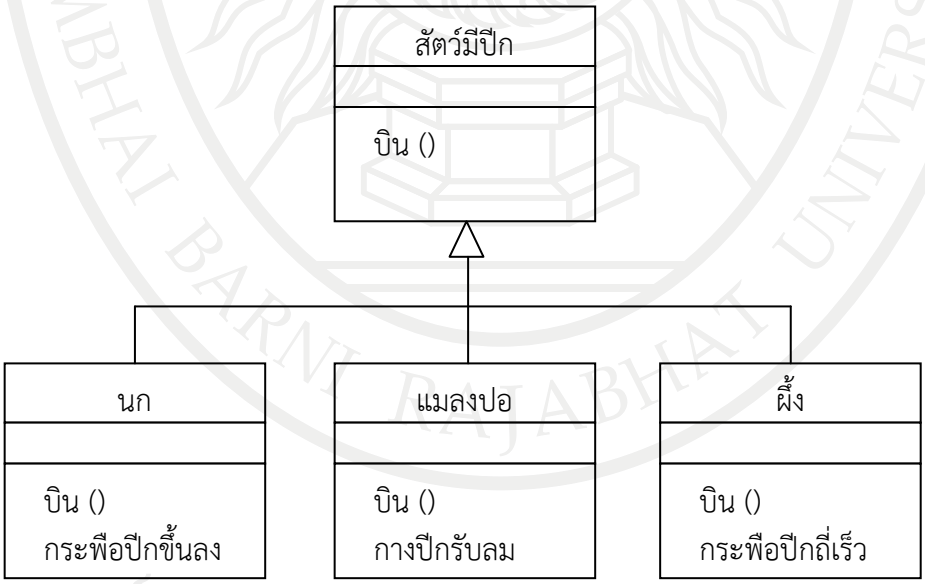
ภาพที่ 2.11 การสืบทอดคุณสมบัติ

ตัวอย่างของการเขียนโปรแกรมลักษณะของการสืบทอดเป็นดังนี้

```
1    public class Customer
2    {
3        public string Name { get; set; }
4        public string Sname { get; set; }
5    }
6
7    public class General : Customer
8    {
9        public void Register(string Username, string Password)
10       {
11       }
12   }
```

ในบรรทัดที่ 7 เป็นคำสั่งที่ใช้ในการสืบทอดจากคลาส Customer ไปยังคลาส General

2.3.4 การพ้องรูป (Polymorphism) คือ การถ่ายทอดคุณสมบัติเหมือนกับการสืบทอด เพียงแต่การทำงานของคลาสลูกแต่ละคลาสไม่เหมือนกัน จึงไม่สามารถกำหนดการทำงานไว้ในคลาสที่เป็นคลาสแม่ได้ คลาสแม่จะกำหนดเพียงแค่โครงสร้างเท่านั้น ส่วนการกำหนดรายละเอียดการทำงานของเมธอดจะให้คลาสลูกกำหนดเอง ทำให้เกิดที่มาของหนึ่งรูปหลายพฤติกรรม ดังภาพที่ 2.12



ภาพที่ 2.12 การพ้องรูป

ตัวอย่างของการเขียนโปรแกรมลักษณะของการพ้องรูปเป็นดังนี้

```
1    public class Shape
2    {
3        public virtual void Draw()
4        {
5            MessageBox.Show(“Complete”);
6        }
7    }
8    class Triangle : Shape
9    {
10       public override void Draw()
11       {
12           MessageBox.Show(“Draw Triangle”);
13           base.Draw();
14       }
15   }
```

บรรทัดที่ 3 เป็นเมธอดที่อยู่ในคลาสแม่ เมื่อมีการสืบทอดลักษณะแบบพ้องรูปจะมีการปรับการทำงานในบรรทัดที่ 10 ของคลาสลูก

2.4 สรุป

การพัฒนาโปรแกรมเชิงวัตถุเกิดจากระบบปัจจุบันมีความซับซ้อน ทำให้นักพัฒนาต้องหาวิธีการเพื่อแก้ปัญหา ภาษาเชิงวัตถุภาษาแรกคือ ภาษาซิมูลา และต่อมาภาษาสมอลล์ทอร์ค ก็เข้ามาแทนที่และเป็นภาษาที่เป็นเชิงวัตถุอย่างสมบูรณ์ และมีการพัฒนาภาษาอื่น ๆ ตามมาอีกมากมาย เช่น ภาษาซีพลัสพลัส ภาษาจาวา ภาษาซีชาร์ป เป็นต้น แนวคิดหลักของภาษาเชิงวัตถุ คือ มุ่งเน้นการแก้ปัญหาโดยมองปัญหาตามสภาพความเป็นจริงในชีวิตและสร้างแบบจำลอง เพื่อใช้สิ่งเหล่านั้นในการแก้ไขปัญหา

สำหรับคุณสมบัติของภาษาโปรแกรมเชิงวัตถุโดยทั่วไปควรมีคุณสมบัติพื้นฐานต่าง ๆ ดังนี้คือ การห่อหุ้ม คือ การรวมคุณสมบัติและพฤติกรรมของวัตถุเข้าด้วยกัน โดยกำหนดเป็นชนิดของวัตถุ การซ่อนรายละเอียด คือ การกำหนดระดับในการเข้าถึงข้อมูลเพื่อป้องกันการเข้ามาใช้ข้อมูลโดยวัตถุที่ไม่เกี่ยวข้อง ซึ่งอาจจะทำให้เกิดความผิดเพี้ยนของข้อมูล การสืบทอดเป็นการถ่ายทอดคุณสมบัติหรือความสามารถของวัตถุชิ้นหนึ่ง ไปยังวัตถุอีกชิ้นหนึ่ง โดยมีลักษณะพื้นฐานตามของเดิมและเพิ่มสิ่งใหม่เข้าไปได้ และการพ้องรูป คือ การถ่ายทอดคุณสมบัติเหมือนกับการสืบทอด เพียงแต่การทำงานของคลาสลูกแต่ละคลาสไม่เหมือนกัน จึงไม่สามารถกำหนดการทำงานไว้ในคลาสแม่ได้ คลาสแม่จะกำหนดเพียงแค่โครงสร้างเท่านั้น

แบบฝึกหัดบทที่ 2

1. จงอธิบายประวัติความเป็นมาของภาษาเชิงวัตถุ
2. จงอธิบายแนวคิดของภาษาเชิงวัตถุ
3. ภาษาเชิงฟังก์ชันแตกต่างจากภาษาเชิงวัตถุอย่างไร
4. คลาสในภาษาเชิงวัตถุหมายถึงอะไร
5. คุณสมบัติการสืบทอดและการพ้องรูปต่างกันอย่างไร
6. วิธีการเข้าถึงข้อมูลซึ่งเป็นกลไกป้องกันข้อมูลมีกี่แบบ อะไรบ้าง
7. การห่อหุ้มคืออะไร มีประโยชน์อย่างไร
8. จงอธิบายการวาดรูปเรขาคณิตเป็นภาษาเชิงวัตถุ
9. จงออกแบบคลาสของรถยนต์
10. จงออกแบบคลาสตัวอย่างของระบบซื้อขาย

เอกสารอ้างอิง

กิตติพงษ์ กลมกล่อม. (2552). **การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML**. กรุงเทพฯ : เคทีพี.

ธวัชชัย งามสันติวงศ์. (2549). **การวิเคราะห์และออกแบบระบบงานเชิงวัตถุ**. กรุงเทพฯ : ศูนย์หนังสือจุฬาลงกรณ์มหาวิทยาลัย.

ธีระพล ลิ้มศรัทธา. (2553). **การเขียนโปรแกรมเชิงวัตถุด้วย Visual Basic.NET**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

ศุภชัย สมพานิช. (2556). **คู่มือเรียนและใช้งาน Visual C#**. กรุงเทพฯ : สวัสดี ไอที.

แผนบริหารประจำบทที่ 3

เนื้อหา

บทที่ 3 พื้นฐานการพัฒนาโปรแกรมด้วยภาษาวิชวลซีชาร์ป

3.1 เริ่มต้นเรียนรู้ภาษาวิชวลซีชาร์ป
3.2 คลาสที่เกี่ยวข้องกับอินพุทและเอาท์พุท
3.3 การเขียนโปรแกรมด้วยวิชวลซีชาร์ป
3.4 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. อธิบายหลักการทำงานของภาษาวิชวลซีชาร์ปได้
2. ใช้งานคลาสที่เกี่ยวข้องกับอินพุทและเอาท์พุท
3. เขียนโปรแกรมแบบคอนโซลแอพพลิเคชันได้
4. เขียนโปรแกรมแบบวินโดว์ฟอร์มแอพพลิเคชันได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน และจากเว็บไซต์
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ผู้สอนเปิดโปรแกรมวิชวลซีชาร์ป และอธิบายการเขียนตัวอย่างโปรแกรมพร้อมกับให้ผู้เรียนทดลองทำตาม
5. ให้ผู้เรียนฝึกปฏิบัติ
6. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 3

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 3
4. ตัวอย่างโปรแกรม
5. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน

2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. ประเมินจากการฝึกปฏิบัติ
4. ประเมินจากแบบฝึกหัดบทที่ 3
5. ประเมินจากการสอบกลางภาค

บทที่ 3
พื้นฐานการพัฒนาโปรแกรมด้วยภาษาวิชวลซีชาร์ป

วิชวลซีชาร์ป (Visual C#) เป็นภาษาที่ทางบริษัทไมโครซอฟต์พัฒนาขึ้นมาเป็นส่วนหนึ่งของโปรแกรมชุดวิชวลสตูดิโอดอทเน็ต (Visual Studio.NET) เพื่อรองรับการทำงานบนดอทเน็ตโดยมีลักษณะของการเขียนโปรแกรมเชิงวัตถุสมัยใหม่ (Modern OOP) ภาษาวิชวลซีชาร์ปมีความสามารถเชิงวัตถุใกล้เคียงกับภาษาวิชวลซีพลัสพลัส แต่มีความง่ายในการเขียนโปรแกรมเหมือนกับวิชวลเบสิค ดังภาพที่ 3.1

ง่าย ยาก

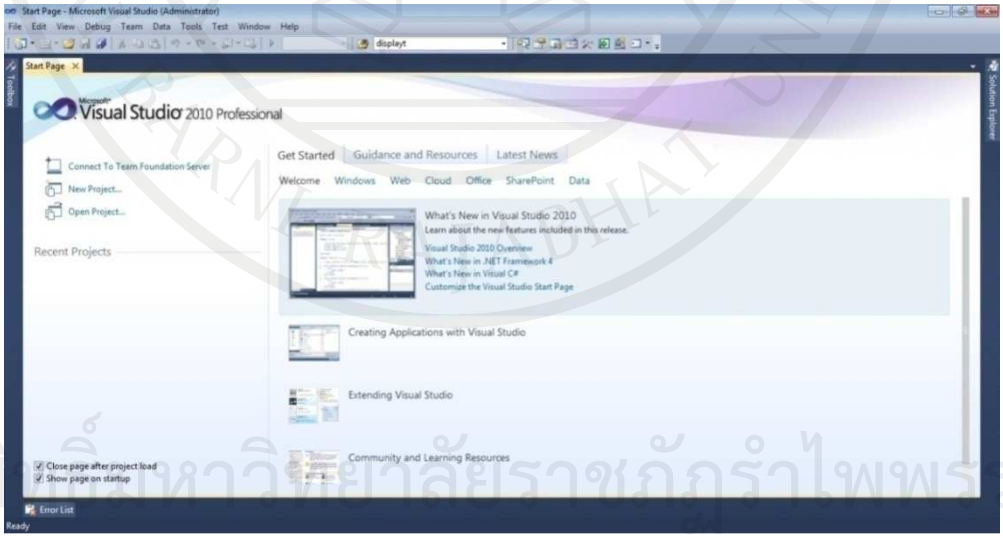
Visual Basic	Visual C#	Visual C++

ภาพที่ 3.1 ความยากของวิชวลซีชาร์ป

3.1 เริ่มต้นเรียนรู้ภาษาวิชวลซีชาร์ป

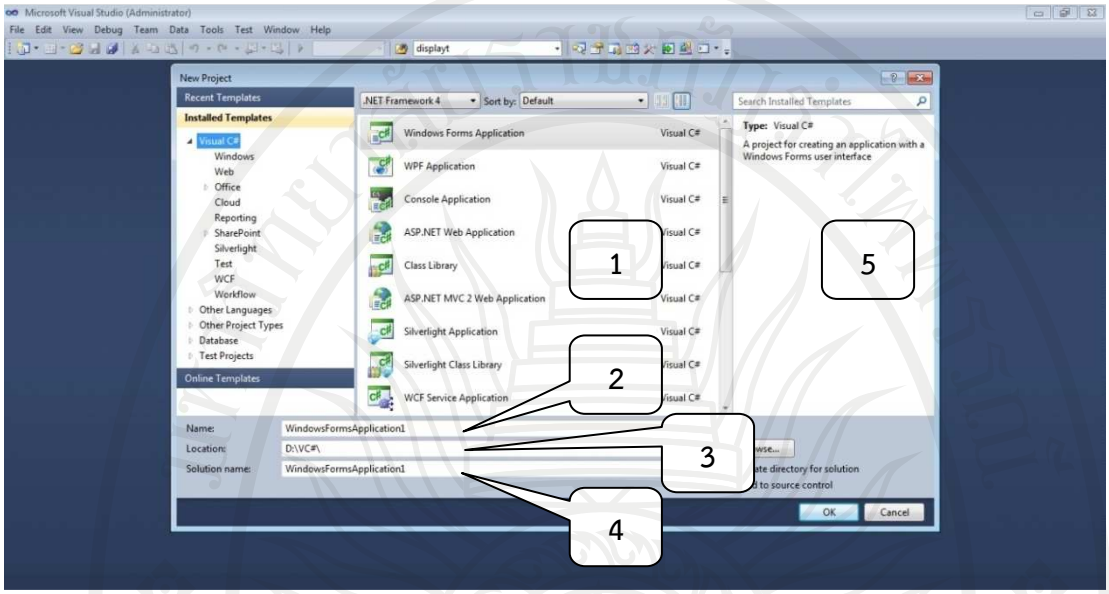
3.1.1 ทำความรู้จักกับสภาพแวดล้อมของวิชวลสตูดิโอ 2010

ภาษาวิชวลซีชาร์ปเป็นภาษาหนึ่งที่อยู่ในวิชวลสตูดิโอ 2010 (Visual Studio 2010) ดังนั้นก่อนจะเริ่มเขียนโปรแกรมจึงควรเรียนรู้สภาพแวดล้อมของการใช้งานก่อน โดยเริ่มต้นการเปิดโปรแกรมจากเมนู Start -> Microsoft Visual Studio 2010 -> Microsoft Visual Studio 2010 (ศุภชัย สมพานิช, 2556) ดังภาพที่ 3.2



ภาพที่ 3.2 หน้าจอของวิชวลสตูดิโอ 2010

จากนั้นเลือกที่เมนู File -> New -> Project หรือกด Ctrl+Shift+N แล้วเลือกวิชวลซีชาร์ปจะได้หน้าจอดังภาพที่ 3.3



ภาพที่ 3.3 หน้าจอการสร้าง Project

การสร้าง Project ในวิชวลซีชาร์ปจะมีส่วนประกอบดังนี้

หมายเลข 1 ใช้สำหรับกำหนดประเภทของ Project ที่สร้างขึ้น ในวิชาการเขียนโปรแกรมเชิงวัตถุจะใช้แค่ 2 ประเภท คือ Windows Forms Application และ Console Application

หมายเลข 2 ใช้สำหรับกำหนดชื่อของ Project ที่สร้างขึ้น

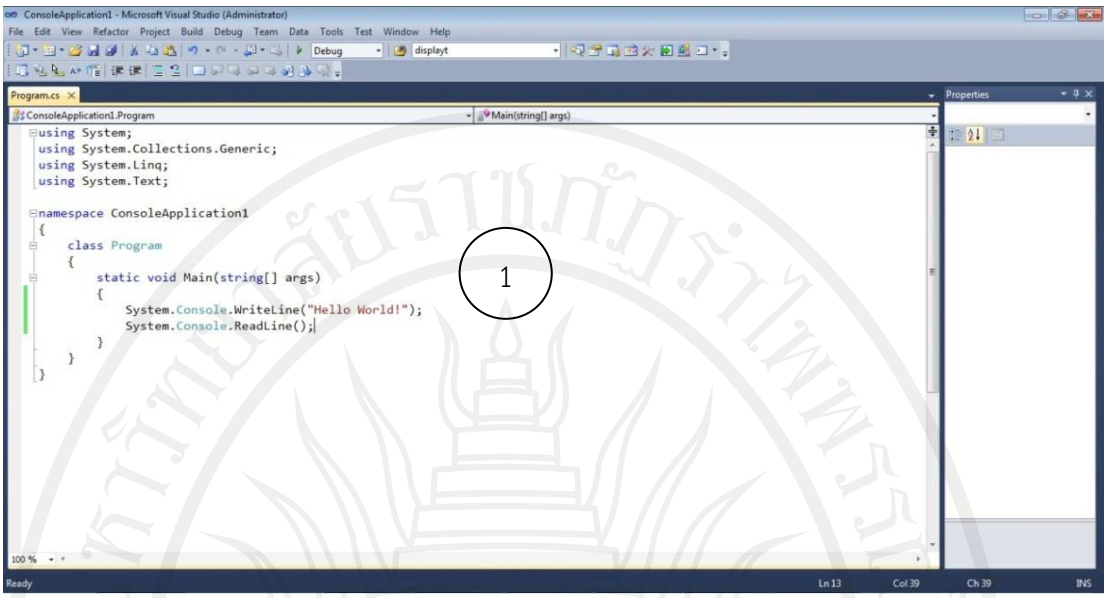
หมายเลข 3 ใช้สำหรับระบุตำแหน่งที่จัดเก็บ Project ในเครื่องคอมพิวเตอร์

หมายเลข 4 ใช้สำหรับกำหนดชื่อของ Solution ของ Project

หมายเลข 5 ใช้สำหรับอธิบายรายละเอียดของประเภท Project ที่เลือก

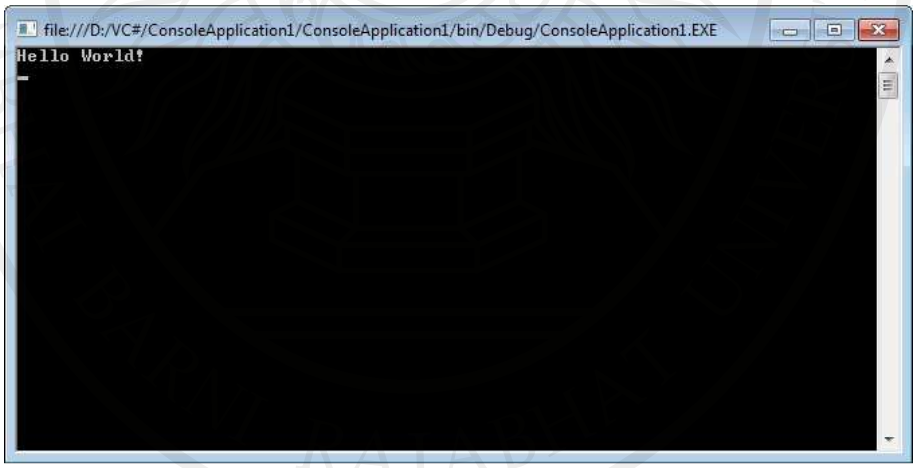
3.1.1.1 การสร้าง Project แบบ Console Application

การสร้าง Project แบบนี้เป็นการเขียนโปรแกรมลักษณะที่ให้แสดงผลแบบคอมมานด์ไลน์ (Command Line) การเขียนโดยทั่วไปจะคล้ายกับการเขียนด้วยภาษาซีหรือซีพลัสพลัส ซึ่งมีหน้าจอดังภาพที่ 3.4



ภาพที่ 3.4 การสร้าง Project แบบ Console Application

การเขียนโปรแกรมแบบ Console Application ในวิชวลซีชาร์ปจะมีพื้นที่สำหรับใช้ในการเขียนโปรแกรม คือ ตำแหน่งหมายเลข 1 ซึ่งรูปแบบการเขียนจะคล้ายกับการเขียนแบบกระบวนคำสั่ง (Procedure Programming) ดังภาพที่ 3.5

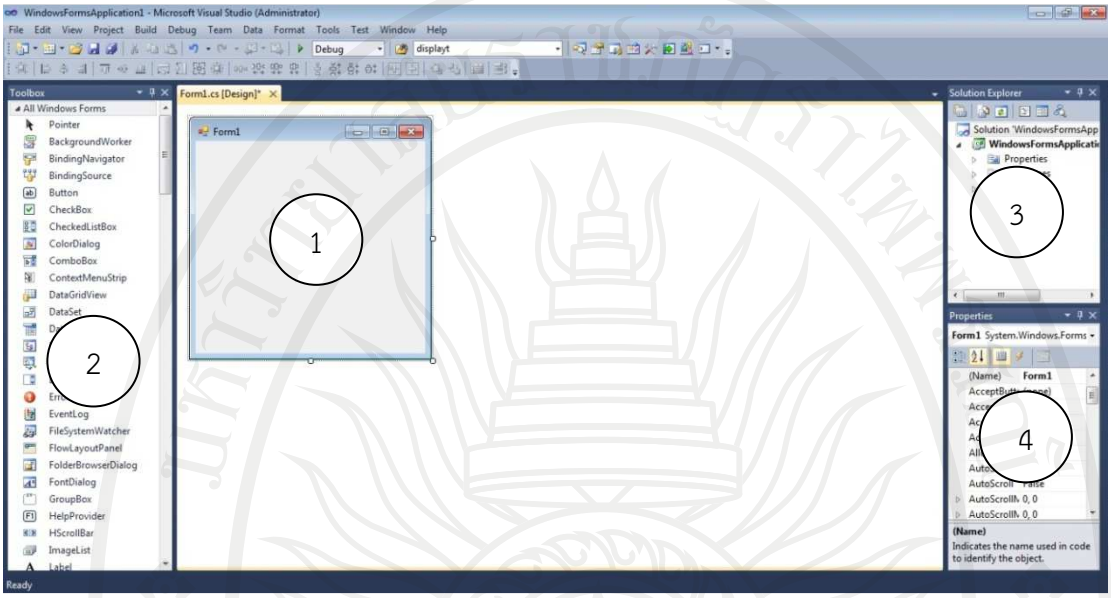


ภาพที่ 3.5 ผลการรันแบบ Console Application

3.1.1.1 การสร้าง Project แบบ Windows Forms Application

การสร้าง Project แบบนี้เป็นลักษณะการแสดงผลแบบกราฟิก (GUI) รูปแบบการทำงานจะเป็นแบบอีเวนท์ไดร์เวนโปรแกรมมิง (Event-Driven Programming) ซึ่งเป็นการเขียน

โปรแกรมโดยอาศัยเหตุการณ์ต่าง ๆ ที่เกิดขึ้นกับวัตถุเป็นตัวเลือกวิธีการทำงาน จะมีหน้าจอดังภาพที่ 3.6 และเมื่อรันจะได้ผลดังภาพที่ 3.7



ภาพที่ 3.6 การสร้าง Project แบบ Windows Forms Application

ส่วนประกอบในการเขียนโปรแกรมแบบ Windows Forms Application จะมีดังนี้

หมายเลข 1 เป็นส่วนของฟอร์มที่ใช้แสดงผลในลักษณะของวินโดว์

หมายเลข 2 เป็นส่วนของกล่องเครื่องมือ (Toolbox) ที่เก็บเครื่องมือสำหรับตกแต่งและออกแบบหน้าตาของฟอร์ม

หมายเลข 3 Solution Explorer ทำหน้าที่แสดงโครงสร้างรายการไฟล์ที่เกี่ยวข้องกับ Project

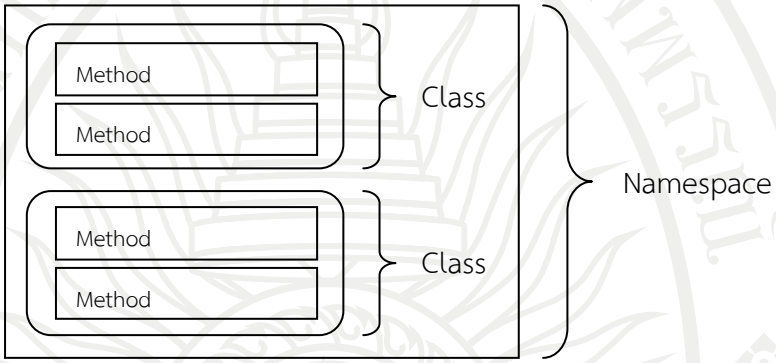
หมายเลข 4 Properties Windows ทำหน้าที่แสดงคุณสมบัติของวัตถุที่เลือก

ภาพที่ 3.7 ผลการรันแบบ Windows Application

3.1.2 โครงสร้างของโปรแกรมภาษาวิชวลซีชาร์ป

ในการเขียนโปรแกรมด้วยภาษาวิชวลซีชาร์ปจำเป็นจะต้องทราบโครงสร้างของภาษาก่อน เพื่อให้เข้าใจในการทำงานของภาษา โดยการเขียนโปรแกรมวิชวลซีชาร์ปจะต้องมีโครงสร้างและเงื่อนไขตามภาพที่ 3.8 ถึงภาพที่ 3.10 ดังนี้ (Microsoft, 2016)

1) โปรแกรมสามารถประกอบด้วยเนมสเปซ (Namespaces) อย่างน้อย 1 เนมสเปซ
2) เนมสเปซประกอบด้วยคลาส อย่างน้อย 1 คลาส
3) คลาสประกอบด้วยเมธอด อย่างน้อย 1 เมธอด



ภาพที่ 3.8 โครงสร้างของภาษาวิชวลซีชาร์ป

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine("Hello World!");
            System.Console.ReadLine();
        }
    }
}
```

ภาพที่ 3.9 โครงสร้างของภาษาวิชวลซีชาร์ปแบบ Console Application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace HelloWorld
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

ภาพที่ 3.10 โครงสร้างของภาษาวิชวลซีชาร์ปแบบ Windows Forms Application

4) คำสั่งทุกคำสั่งเมื่อจบแล้วต้องปิดท้ายด้วยเซมิโคลอน (;)

5) จำนวนช่องว่างในแต่ละแถวหรือการเว้นบรรทัดไม่มีผลต่อการทำงาน

6) ในการสร้างบล็อกของคำสั่งจะใช้เครื่องหมายปีกกา { }

7) การใส่ข้อความหมายเหตุ (comment) มี 2 แบบ คือ หมายเหตุบรรทัดเดียวใช้ //[ข้อความ] และหมายเหตุหลายบรรทัดใช้ /*[ข้อความ]*/

3.1.3 การตั้งชื่อของภาษาวิชวลซีชาร์ป

ในการเขียนโปรแกรมด้วยภาษาวิชวลซีชาร์ปจะมีข้อกำหนดในการตั้งชื่อต่าง ๆ ดังนี้

1) ตัวอักษรเล็กและตัวอักษรใหญ่มีความหมายต่างกัน

2) จะต้องขึ้นต้นด้วยตัวอักษร เครื่องหมายขีดล่าง (_) หรือ @ ก็ได้

3) ไม่สามารถใช้ตัวเลข หรือตัวอักขระ เป็นตัวเริ่มต้นได้

4) ชื่อที่ใช้จะต้องไม่เป็นคำสงวน (reserved word)

ตารางที่ 3.1 คำสงวน

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	in (generic modifier)	int
interface	internal	is	lock	long
namespace	new	null	object	operator
out	out (generic modifier)	override	params	private
protected	public	readonly	ref	sealed
short	sizeof	stackalloc	static	string
this	throw	true	try	typeof
uint	unsafe	ushort	using	virtual
void	volatile			

ที่มา : (Microsoft, 2016)

3.1.3 ชนิดของข้อมูล

ชนิดของข้อมูลใช้สำหรับการประกาศตัวแปร ซึ่งในภาษาวิชวลซีชาร์ปสามารถกำหนดชนิดของข้อมูลได้ดังตารางที่ 3.2 ถึง 3.5 (นิรันดร์ ประวิทย์ธนา, 2545)

ตารางที่ 3.2 เลขจำนวนเต็ม

Data type	ขนาด	ค่าของข้อมูล
sbyte (System.SByte)	1 byte	-128 ถึง 127
short (System.Int16)	2 bytes	-32,768 ถึง 32,767
int (System.Int32)	4 bytes	-2,147,483,648 ถึง 2,147,483,647
long (System.Int64)	8 bytes	-2^{63} ถึง (2^{63} – 1)

ตารางที่ 3.3 เลขจำนวนเต็มบวก

Data type	ขนาด	ค่าของข้อมูล
byte (System.Byte)	1 byte	0 ถึง 255
ushort (System.UInt16)	2 bytes	0 ถึง 65,535
uint (System.UInt32)	4 bytes	0 ถึง 4,294,967,295
ulong (System.UInt64)	8 bytes	0 ถึง 2^{64} -1

ตารางที่ 3.4 เลขทศนิยม

Data type	ขนาด	ค่าของข้อมูล
float (System.Single)	4 bytes	$\pm 1.5 \times 10^{-45}$ ถึง $\pm 3.4 \times 10^{38}$
double (System.Double)	8 bytes	$\pm 5.0 \times 10^{-324}$ ถึง $\pm 1.7 \times 10^{308}$
decimal (System.Decimal)	16 bytes	$\pm 1.0 \times 10^{-28}$ ถึง $\pm 7.9 \times 10^{28}$

ตารางที่ 3.5 ประเภทข้อมูลที่ไม่ใช่ตัวเลข

Data type	ขนาด	ค่าของข้อมูล
char (System.Char)	2 bytes	ตัวอักษรแบบ Unicode มีเครื่องหมาย ' (single quote) คร่อมตัวอักษร เช่น '1', 'A'
string (System.String)	ไม่แน่นอน	ตัวอักษรแบบ Unicode หลายตัวมารวมกัน มีเครื่องหมาย " (double quote) คร่อม เช่น "Welcome"
bool (System.Boolean)	1 bit	มีค่าที่เป็นไปได้ 2 ค่า คือ จริง (true) และ เท็จ (false)

3.1.4 การประกาศตัวแปร

การประกาศตัวแปรในการเขียนโปรแกรมนั้นมีความสำคัญเป็นอย่างยิ่ง เนื่องจากโปรแกรมจำเป็นจะต้องมีตัวแปรสำหรับพักค่าต่าง ๆ ที่ใช้ในการทำงาน เช่น การคำนวณ การเก็บผลลัพธ์ เป็นต้น ซึ่งในภาษาวิชวลซีชาร์ปจะมีรูปแบบในการประกาศตัวแปรดังนี้

3.1.4.1 การประกาศตัวแปรทั่วไป ใช้สำหรับประกาศตัวแปรที่สามารถเปลี่ยนแปลงค่าได้ในโปรแกรม สามารถประกาศได้ 2 วิธี ได้แก่

วิธีที่ 1 [ชนิดของข้อมูล] ชื่อตัวแปร;
วิธีที่ 2 [ชนิดของข้อมูล] ชื่อตัวแปร = [ค่าเริ่มต้นของตัวแปร];

ตัวอย่าง

```
int x;        int y, z;int Y, Z = 3;
Double d = 10.99;
String D =  "Hello World",str;
```

หมายเหตุ ไม่สามารถประกาศตัวแปรซ้ำภายในบล็อก { } เดียวกัน

3.1.4.2 การประกาศค่าคงที่ ใช้สำหรับประกาศตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าได้ในโปรแกรม สามารถประกาศได้ดังนี้

ตัวอย่าง

```
const int x = 1;
const double pi = 3.14;
```

3.1.5 ตัวดำเนินการทางคณิตศาสตร์ (Operator)

การเขียนคำสั่งในลักษณะที่มีการใช้นิพจน์ทางคณิตศาสตร์เพื่อช่วยในการคำนวณต่าง ๆ จำเป็นจะต้องมีตัวดำเนินการประกอบอยู่ในนิพจน์นั้น ซึ่งในการคำนวณตามนิพจน์จะต้องดูลำดับการทำงานของตัวดำเนินการเพื่อให้เกิดความถูกต้องในการใช้ตัวดำเนินการในนิพจน์ ดังตารางที่ 3.6

ตารางที่ 3.6 ลำดับความสำคัญของตัวดำเนินการ

ลำดับ	ตัวดำเนินการ
1	( ), x++, x--
2	+x, -x, ++x, --x, !
3	*, /, %
4	+, -
5	shift bits left: <<, shift bits right: >>
6	<, <=, >, >=, is, as
7	= =, !=
8	&, |, ^
9	&&, ||, ?:
10	=, +=, =,*=, /=, %=, <<=, >>=, &=, ^=, |=

ที่มา : (นิรันดร์ ประวิทย์ธนา, 2545)

ถ้าในกรณีที่ตัวดำเนินการมีความสำคัญเท่ากัน การดำเนินการจะเริ่มจากทางซ้ายไปทางขวาตามลำดับของตัวดำเนินการ

3.1.6 การแปลงชนิดของข้อมูล

เนื่องจากภาษาวิชวลซีชาร์ปเข้มงวดเรื่องชนิดของข้อมูลมากเพื่อป้องกันความผิดพลาดที่เกิดจากใช้ชนิดของข้อมูลที่ไม่ถูกต้อง จึงทำให้ไม่สามารถนำข้อมูลต่างชนิดมาใช้งานร่วมกันได้โดยตรง แต่จะต้องนำมาแปลงให้เป็นข้อมูลเดียวกันก่อน โดยวิธีการแปลงข้อมูลนั้นทำได้ 4 วิธี ดังนี้

3.1.6.1 การแปลงแบบอิมพลิซิท (Implicit conversion) วิธีนี้เป็นการแปลงข้อมูลโดยไม่ชัดเจน เป็นการแปลงข้อมูลประเภทเดียวกัน แต่ใช้พื้นที่การเก็บไม่เท่ากัน ซึ่งจะแปลงได้ถูกต้องก็ต่อเมื่อแปลงจากชนิดที่ใช้พื้นที่ขนาดเล็กไปสู่ชนิดที่ใช้พื้นที่ใหญ่กว่า เช่น

```
short var1 = 35;
short var2 = 18;
long total = var1 + var2;
```

3.1.6.2 การแปลงแบบเอ็กพลิซิท (Explicit conversion) วิธีนี้เป็นการแปลงข้อมูลให้เกิดความชัดเจน โดยการระบุชนิดข้อมูลไว้หน้าชื่อตัวแปรที่จะแปลง ซึ่งจะครอบด้วยวงเล็บเพื่อแยกชนิดข้อมูลชื่อกับตัวแปร ตัวอย่างเช่น

```
float num1 = 15.6f;
int num2 = (int) (num1);
```

3.1.6.3 การใช้เมธอด Parse จะใช้แปลงข้อมูลชนิดสตริง (string) ไปเป็นค่าตัวเลขที่ตรงกับข้อความนั้น ๆ ซึ่งจะทำงานตรงข้ามกับเมธอด toString ที่ใช้สำหรับแปลงค่าจากตัวเลขไปเป็นสตริง เช่น

```
string str;
string age = “17”;
int i = int.Parse(age);
MessageBox.Show(“อายุ = ”+ i.toString());
```

3.1.6.4 การใช้งานอ็อบเจกต์คอนเวิร์ท (Object Convert) จะมีเมธอดที่ใช้ในการแปลงข้อมูลชนิดหนึ่งไปสู่ข้อมูลอีกชนิดหนึ่งที่ต้องการ ดังตัวอย่าง

```
double val1 = 24.56;
long Lvar1 = Convert.ToInt16(val1);
```

3.2 คลาสที่เกี่ยวข้องกับอินพุทและเอาท์พุท

ในการเขียนโปรแกรมภาษาวิชวลซีชาร์ปจะมีคลาส Console จะมีเมธอดที่ทำงานเกี่ยวข้องกับการรับและแสดงผลข้อมูล ซึ่งในการเขียนโปรแกรมในรายวิชานี้จะกล่าวถึงคลาสที่เกี่ยวข้องกับการทำงาน 2 ส่วน คือ Console Application และ Windows Forms Application ซึ่งแต่ละส่วนจะมีการใช้งานแตกต่างกันดังนี้ (Microsoft, 2016)

3.2.1 ในส่วนของ Console Application

3.2.1.1 คำสั่งแสดงผล

วิธีที่ 1 Console.Write (ข้อความ);
วิธีที่ 2 Console.WriteLine (ข้อความ);

ตัวอย่าง

```
Console.Write ("Hello");
Console.WriteLine ("Hello World");
string str = "Hello";
Console.Write (str + " World");
Console.ReadLine ();
```

3.2.1.2 คำสั่งรับข้อมูล

วิธีที่ 1 Console.Read (); หรือ Console.ReadLine ();
วิธีที่ 2 ตัวแปร = Console.Read (); หรือ ตัวแปร = Console.ReadLine ();

ตัวอย่าง

```
Console.Write ("Where are you? : ");
string address = Console.ReadLine ( );
Console.Write ("You are at " + address);
Console.Read ( );
```

3.2.2 ในส่วนของ Windows Forms Application

3.2.1.1 แสดงผลด้วย MessageBox

สำหรับส่วนที่ใช้ติดต่อกับผู้ใช้ที่มีไว้สำหรับการตอบโต้ในขณะที่โปรแกรมทำงานก็คือ MessageBox ซึ่งเป็นฟอร์มที่จะแสดงขึ้นมาและหายไปเมื่อทำงานเรียบร้อยแล้ว โดยมีรูปแบบการใช้งานดังนี้ (Microsoft, 2016)

รูปแบบ

```
MessageBox.Show(Text, Caption, Buttons, Icon, DefaultButton);
```

Text คือ ข้อความที่ปรากฏใน MessageBox มีชนิดข้อมูลเป็น String

Caption คือ ข้อความบน TitleBar ของ MessageBox มีชนิดข้อมูลเป็น String

Buttons คือ ปุ่มที่แสดงใน MessageBox สำหรับให้ผู้ใช้งานเลือกหลังจากอ่านข้อความแล้ว มีรูปแบบข้อมูลเป็น MessageBoxButtons

Icon คือ ภาพขนาดเล็กที่ใช้แสดงประกอบกับข้อความเพื่อแสดงความรู้สึก มีรูปแบบข้อมูลเป็น MessageBoxIcon

DefaultButton คือ การกำหนดตำแหน่งของปุ่ม เพื่อตั้งให้เป็นปุ่ม Default มีรูปแบบข้อมูลเป็น MessageBoxDefaultButton

ตารางที่ 3.7 การเรียกใช้งานปุ่มใน MessageBox

การเรียกใช้งาน	ปุ่มที่ปรากฏ
MessageBoxButtons.AbortRetryIgnore	Abort Retry Ignore
MessageBoxButtons.OK	OK
MessageBoxButtons.OKCancel	OK Cancel
MessageBoxButtons.RetryCancel	Retry Cancel
MessageBoxButtons.YesNo	Yes No
MessageBoxButtons.YesNoCancel	Yes No Cancel

ตารางที่ 3.8 การเรียกใช้งาน Icon ใน MessageBox

การเรียกใช้งาน	Icon ที่ปรากฏ
MessageBoxIcon.Asterisk	
MessageBoxIcon.Error	
MessageBoxIcon.Exclamation	
MessageBoxIcon.Hand	
MessageBoxIcon.Information	
MessageBoxIcon.None	ไม่แสดงรูปภาพใด ๆ
MessageBoxIcon.Question	
MessageBoxIcon.Stop	
MessageBoxIcon.Warning	

3.3 การเขียนโปรแกรมด้วยวิชวลซีชาร์ป

3.3.1 ทดลองเขียนโปรแกรมแบบ Console Application

การเขียนโปรแกรมแบบ Console Application เป็นการเขียนโปรแกรมที่มีลักษณะการแสดงผลแบบคอมมานด์ไลน์ (Command Line) และมีรูปแบบการเขียนเป็นแบบประมวลคำสั่ง จึงต้องมีการจัดลำดับของการแสดงผลให้ชัดเจน

ตัวอย่างที่ 3.1 ให้เขียนโปรแกรมคำนวณพื้นที่สี่เหลี่ยมผืนผ้า โดยรับค่าความกว้างและความยาวเป็นเลขจำนวนเต็มบวก และแสดงผลหน้าจอดังภาพที่ 3.11

ภาพที่ 3.11 การแสดงผลของโปรแกรมคำนวณพื้นที่สี่เหลี่ยม

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections.Generic;
3       using System.Linq;
4       using System.Text;
5
6       namespace CalRect
7       {
8         class Program
9         {
10            static void Main(string[] args)
11            {
12                int _wide; //ประกาศตัวแปร
13                int _long; //ประกาศตัวแปร
14                int _area; //ประกาศตัวแปร
15                Console.Write("Enter wide : "); //รอรับค่า
16                _wide = Convert.ToInt16(Console.ReadLine()); //เก็บค่าในตัวแปร
17                Console.Write("Enter long : "); //รอรับค่า
18                _long = Convert.ToInt16(Console.ReadLine()); //เก็บค่าในตัวแปร
19                _area = _wide * _long; //คำนวณพื้นที่
20                Console.WriteLine("Area = {0}", _area); //แสดงผล
21                Console.ReadLine();
22            }
23        }
24      }
```

อธิบายในส่วนของชุดคำสั่งโปรแกรม

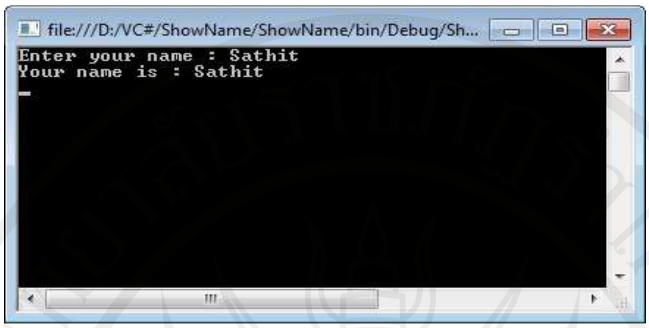
บรรทัดที่ 1 – 4 เป็นส่วนของการเรียกใช้เนมสเปซที่เกี่ยวข้องกับการทำงาน

บรรทัดที่ 6 – 10 เป็นส่วนของการกำหนดชื่อเนมสเปซ ชื่อคลาส และชื่อของเมธอดหลัก

บรรทัดที่ 12 – 14 เป็นการกำหนดตัวแปรที่ใช้ในการทำงาน

บรรทัดที่ 15 – 21 เป็นการรับค่าเพื่อใช้ในการคำนวณและแสดงผล

ตัวอย่างที่ 3.2 ให้เขียนโปรแกรมรับชื่อ และแสดงผลชื่อที่รับเข้าไป และแสดงผลดังภาพที่ 3.12



ภาพที่ 3.12 การแสดงผลของโปรแกรมรับชื่อ

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections.Generic;
3       using System.Linq;
4       using System.Text;
5       namespace ShowName
6       {
7           class Program
8           {
9               static void Main(string[] args)
10              {
11                  string _name; //ประกาศตัวแปร
12                  Console.Write("Enter your name : ");  //รอรับค่า
13                  _name = Console.ReadLine(); //เก็บค่าในตัวแปร
14                  Console.WriteLine("Your name is : " + _name); //แสดงผล
15                  Console.ReadLine();
16              }
17          }
18      }
```

อธิบายในส่วนของชุดคำสั่งโปรแกรมที่สำคัญ

บรรทัดที่ 1 – 4 เป็นส่วนของการเรียกใช้เนมสเปซที่เกี่ยวข้องกับการทำงาน

บรรทัดที่ 5 – 9 เป็นส่วนของการกำหนดชื่อเนมสเปซ ชื่อคลาส และชื่อของเมธอดหลัก

บรรทัดที่ 11 – 15 เป็นการรับค่าเพื่อใช้ในการคำนวณและแสดงผล

3.3.2 ทดลองเขียนโปรแกรมแบบ Windows Forms Application

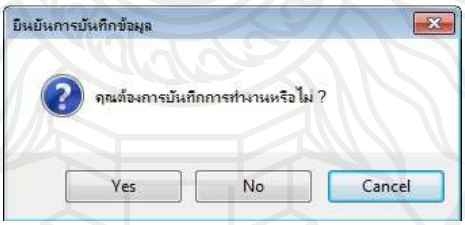
การเขียนโปรแกรมแบบ Windows Forms Application จะมีส่วนหลักคือฟอร์ม ซึ่งเป็นส่วนที่ใช้ติดต่อกับผู้ใช้งาน ภายในฟอร์มจะประกอบด้วยอ็อบเจกต์ (Object) และคอนโทรล (Control) ที่ใช้เป็นเครื่องมือติดต่อกับผู้ใช้งาน มีหน้าที่ต่างกัน เช่น Label ใช้แสดงข้อความ Textbox ใช้รับข้อความ Button ปุ่มคำสั่ง และอาจจะมีคอนโทรลบางประเภทที่ซ่อนอยู่โดยไม่แสดงบนหน้าจอ สำหรับการทำงานกับฟอร์มและคอนโทรลจะต้องทราบส่วนประกอบที่สำคัญคือ

3.3.2.1 Property คือ คุณสมบัติของอ็อบเจกต์ที่ใช้ในการกำหนดลักษณะต่าง ๆ เช่น สีของปุ่ม ข้อความบนปุ่ม ข้อความของ Label เป็นต้น

3.3.2.2 Event คือ เหตุการณ์ที่เกิดขึ้นกับอ็อบเจกต์โดยสามารถเขียนคำสั่งเพื่อกำหนดการทำงานได้ เช่น เขียนคำสั่งให้บันทึกข้อมูลเมื่อปุ่มถูกกด หรือเขียนคำสั่งเมื่อ Textbox มีการเปลี่ยนค่าจากค่าหนึ่งเป็นอีกค่าหนึ่ง เป็นต้น

3.3.2.3 Method คือ คำสั่งที่เขียนขึ้นเพื่อให้อ็อบเจกต์ทำงานเมื่อเกิด Event ขึ้นตามที่ต้องการ เช่น เมื่อปิดฟอร์มให้บันทึกค่าทั้งหมด หรือ ออกจากโปรแกรมเมื่อกดปุ่ม Exit เป็นต้น

ตัวอย่างที่ 3.3 ให้เขียนโปรแกรมแสดง MessageBox ให้แสดงผลตามภาพที่ 3.13 โดยมีเงื่อนไขดังนี้
คำถามข้อความว่า “คุณต้องการบันทึกการทำงานหรือไม่ ?”
แสดงข้อความที่ TitleBar ว่า “ยืนยันการบันทึกข้อมูล”
โดยให้แสดงปุ่ม Yes, No, Cancel และแสดง Icon รูป 



ภาพที่ 3.13 ผลของ MessageBox

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace ExMessageBox
```

```
10      {
11          public partial class frmEx3_1 : Form
12          {
13              public frmEx3_1()
14              {
15                  InitializeComponent();
16              }
17              private void ShowMessageBox_Click(object sender, EventArgs e)
18              {
19                  MessageBox.Show("คุณต้องการบันทึกการทำงานหรือไม่ ?", "ยืนยันการบันทึก
        ข้อมูล", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
20              }
21          }
22      }
```

อธิบายในส่วนของชุดคำสั่งโปรแกรมที่สำคัญ

บรรทัดที่ 17 – 20 เป็นส่วนของการทำงานในเมธอดที่เกิดเหตุการณ์ขึ้น

บรรทัดที่ 19 เป็นส่วนที่ใช้ในการแสดง MessageBox ตามเงื่อนไขที่กำหนด

3.4 สรุป

การเขียนโปรแกรมด้วยวิชวลซีชาร์ปเป็นรูปแบบการเขียนเชิงวัตถุสมัยใหม่ที่มีความยากง่ายอยู่ระหว่างวิชวลเบสิคและวิชวลซีพลัสพลัส ทำให้สามารถพัฒนาโปรแกรมได้ง่ายขึ้น การเขียนโปรแกรมในรูปแบบของ Console Application จะมีรูปแบบการแสดงผลเป็นคอมมานด์ไลน์ซึ่งสามารถรับข้อมูลได้โดยการใช้ Console.Read() หรือ Console.ReadLine() และแสดงผลข้อมูล Console.Write() หรือ Console.WriteLine ส่วนการเขียนแบบ Windows Forms Application จะมีการแสดงผลในรูปแบบของ GUI ซึ่งสามารถตอบโต้กับผู้ใช้ในลักษณะการแจ้งให้ทราบด้วย MessageBox สำหรับการประกาศตัวแปร ต้องขึ้นต้นด้วยตัวอักษร หรือเครื่องหมายขีดล่าง (_) หรือ @ ก็ได้ ซึ่งตัวอักษรเล็กและตัวอักษรใหญ่มีความหมายต่างกัน และไม่สามารถใช้ตัวเลขหรือตัวอักขระเป็นตัวเริ่มต้น และจะต้องไม่ซ้ำกับคำสงวน

การเขียนโปรแกรมแบบ Windows Forms Application จะมีส่วนหลักคือฟอร์ม ซึ่งเป็นส่วนที่ใช้ติดต่อกับผู้ใช้งาน ภายในฟอร์มจะประกอบด้วยอ็อบเจกต์และคอนโทรล ที่ใช้เป็นเครื่องมือติดต่อกับผู้ใช้งาน มีหน้าที่ต่างกัน สำหรับการทำงานกับฟอร์มและคอนโทรลจะต้องทราบส่วนประกอบที่สำคัญคือ ส่วนของ Property เป็นคุณสมบัติของอ็อบเจกต์ที่ใช้ในการกำหนดลักษณะต่าง ๆ ส่วนของ Event คือ เหตุการณ์ที่เกิดขึ้นกับอ็อบเจกต์โดยสามารถเขียนคำสั่งเพื่อกำหนดการทำงานได้ และส่วนของ Method คือ คำสั่งที่เขียนขึ้นเพื่อให้อ็อบเจกต์ทำงานเมื่อเกิด Event ขึ้นตามที่ต้องการ

แบบฝึกหัดบทที่ 3

1. จงหาผลลัพธ์จากคำสั่งต่อไปนี้
   1.1 Console.Write(3 + 2);
   1.2 Console.Write(50 – 4 * 2);
   1.3 Console.Write((40 + 4) / 4);
   1.4 Console.Write(3 / 2);
   1.5 Console.Write(4.0 / 3);
2. จงประกาศตัวแปรหรือค่าคงที่ตามที่กำหนดให้ โดยเลือกใช้ชนิดข้อมูลที่เหมาะสม
   2.1 ค่าคงที่ age เพื่อใช้แทนอายุลูกค้า
   2.2 ตัวแปร temp เพื่อเก็บอุณหภูมิ
   2.3 ค่าคงที่ PI เพื่อแทนค่า 3.1415926535
   2.4 ตัวแปร name เพื่อเก็บค่า Peter
3. ประเภทข้อมูลที่ไม่ใช่ Numeric มีอะไรบ้าง
4. การแปลงชนิดข้อมูลมีกี่วิธี
5. การทำงานแบบ Console Application และ Windows Forms Application ต่างกันอย่างไร
6. Console.Write และ Console.WriteLine ต่างกันอย่างไร
7. โครงสร้างของโปรแกรมภาษาวิชวลซีชาร์ปมีเงื่อนไขอย่างไร
8. MessageBoxButtons.OKCancel ใช้สำหรับแสดงปุ่มอะไรใน MessageBox
9. จงเขียนโปรแกรมคำนวณพื้นที่วงกลมและเส้นรอบวง โดยให้สามารถรับค่ารัศมีจากผู้ใช้ได้ โดยกำหนดค่าคงที่ชื่อ PI ใช้แทนค่า 3.141
   ตัวอย่างหน้าจอแสดงผล
   Enter Radius : 4
   Radius of Circle = 4
   Area of Circle = 50.256
   Round of Circle = 25.128
10. ให้เขียนโปรแกรมแสดง MessageBox โดยมีเงื่อนไขดังนี้
    คำถามข้อความว่า “คุณต้องการบันทึกการทำงานหรือไม่ ?”
    แสดงข้อความที่ TitleBar ว่า “ยืนยันการบันทึกข้อมูล”
    โดยให้แสดงปุ่ม Yes, No และ Cancel
    แสดง Icon รูป 

เอกสารอ้างอิง

นิรันดร์ ประวิทย์ธนา. (2545). **เก่ง C# ให้ครบสูตร**. กรุงเทพฯ : วิตตี้ กรุ๊ป.

ศุภชัย สมพานิช. (2556). **คู่มือเรียนและใช้งาน Visual C#**. กรุงเทพฯ: สวัสดี ไอที.

Microsoft. (2016). **C# Keywords**. (online). Available : https://msdn.microsoft. com/en-us/library/x53a06bb(v=vs.100).aspx. 19 January 2016.

_____. (2016). **Console Class**. (online). Available : https://msdn.microsoft. com/en-us/library/system.console(v=vs.110).aspx. 19 January 2016.

_____. (2016). **MessageBox function**. (online). Available : https://msdn.microsoft.com/en-us/library/windows/desktop/ms645505 (v=vs.85).aspx. 21 January 2016.

แผนบริหารประจำบทที่ 4

เนื้อหา

บทที่ 4 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้และการจัดการเหตุการณ์

4.1 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้

4.2 การจัดการเหตุการณ์

4.3 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. สร้างส่วนติดต่อกับผู้ใช้ได้
2. ใช้ Common Controls ในการออกแบบได้
3. ประยุกต์ใช้เมธอดที่เกี่ยวข้องกับการเขียนโปรแกรมได้
4. ใช้เมธอดและอีเวนท์ที่มีความสัมพันธ์กันได้
5. เขียนคำสั่งเพื่อจัดการกับอีเวนท์ที่เกิดขึ้นได้
6. เขียนโปรแกรมที่จัดการกับเหตุการณ์ที่เกิดขึ้นได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน
3. ให้ผู้เรียนศึกษาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต
4. ให้ผู้เรียนร่วมกันอภิปรายเนื้อหาที่เกี่ยวข้อง
5. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
6. ผู้สอนเปิดโปรแกรมวิชวลซีชาร์ปและอธิบายตัวอย่างโปรแกรม
7. ให้ผู้เรียนฝึกปฏิบัติ
8. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 4

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 4
4. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

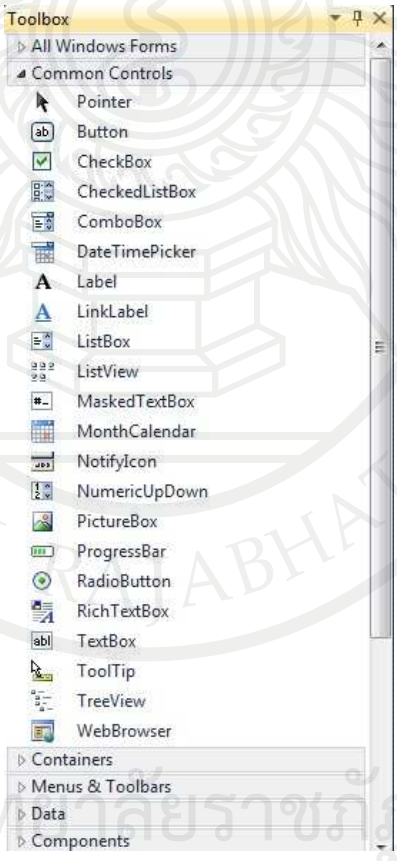
1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. ประเมินการฝึกปฏิบัติ
4. ประเมินจากแบบฝึกหัดบทที่ 4

บทที่ 4
การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้และการจัดการเหตุการณ์

การเขียนโปรแกรมที่มีการแสดงผลแบบกราฟิกมีสิ่งที่สำคัญคือการออกแบบส่วนของอินเตอร์เฟส เนื่องจากเป็นส่วนที่ใช้สำหรับติดต่อกับผู้ใช้ให้สามารถทำงานได้ ถ้าออกแบบไม่ดีอาจจะทำให้ผู้ใช้ไม่เข้าใจหรืออาจจะเกิดข้อผิดพลาดในระหว่างการทำงาน ดังนั้นจึงต้องทำความเข้าใจกับเครื่องมือที่วิชวลซีชาร์ปมีให้ และเขียนคำสั่งในการควบคุมเหตุการณ์ที่เกิดขึ้นจากการทำงานของผู้ใช้เพื่อให้สามารถออกแบบส่วนติดต่อกับผู้ใช้ได้อย่างมีประสิทธิภาพ

4.1 การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้

การเขียนโปรแกรมวิชวลซีชาร์ปในรูปแบบที่เป็น Windows Forms Application จะต้องมีการออกแบบส่วนติดต่อกับผู้ใช้ ซึ่งสามารถใช้แถบเครื่องมือในส่วนของ Common Controls สำหรับการออกแบบฟอร์มต่าง ๆ ได้ (ศุภชัย สมพานิช, 2556) ดังภาพที่ 4.1



ภาพที่ 4.1 แถบเครื่องมือ

การใช้ Common Controls เพื่อนำมาตกแต่งฟอร์มสามารถเลือกได้ตามที่ต้องการซึ่งมี Common Controls ที่สำคัญสำหรับใช้ในการออกแบบดังนี้ (นิรันดร์ ประวิทย์ธนา, 2545)

A Label เป็นส่วนที่ในการสร้างป้ายข้อความในฟอร์มเพื่อบอกให้รู้ว่าส่วนนี้เป็นส่วนใดหรือต้องการให้ใส่ข้อมูลอะไร ส่วนใหญ่จะทำงานร่วมกับ TextBox

abl TextBox เป็นส่วนที่ใช้สำหรับกรอกข้อมูลต่าง ๆ เพื่อใช้ในการประมวลผล จะทำงานร่วมกัน Label เพื่อให้รู้ว่าต้องใส่ข้อมูลอะไร

ab Button เป็นส่วนที่ใช้สร้างปุ่มที่ทำงาน โดยแต่ละปุ่มจะมีการทำงานที่ต่างกันออกไปตามการเขียนคำสั่งควบคุมการทำงาน

CheckBox เป็นส่วนที่ใช้สำหรับสร้างทางเลือกให้กับผู้ใช้ โดยสามารถเลือกได้มากกว่า 1 ตัวเลือก

RadioButton เป็นส่วนที่ใช้สร้างทางเลือกคล้ายกับ CheckBox แต่ผู้ใช้สามารถเลือกได้เพียง 1 ตัวเลือกจากกลุ่มตัวเลือกกลุ่มหนึ่ง

ListBox เป็นส่วนที่ใช้สร้างตัวเลือกให้ผู้ใช้เลื่อนดูรายการ แล้วเลือกตามที่ต้องการ

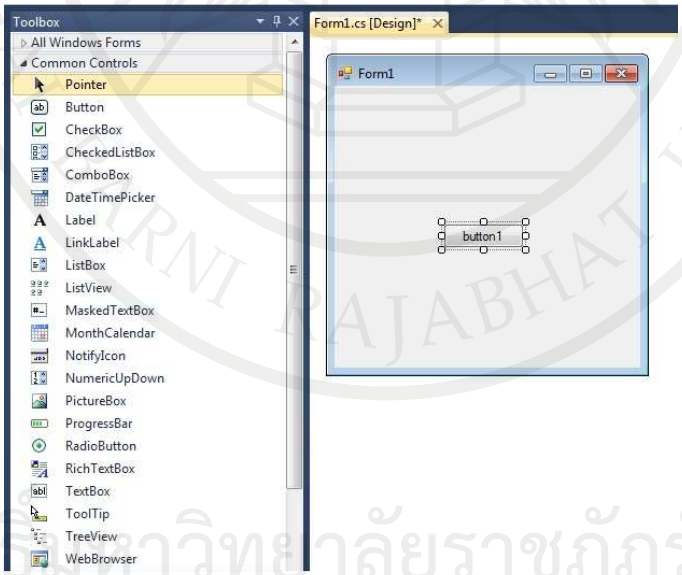
ComboBox สร้างทางเลือกให้ผู้ใช้คล้ายกับ ListBox และผู้ใช้สามารถพิมพ์ข้อมูลเพิ่มเข้าไปได้อีกด้วย

CheckListBox สร้างทางเลือกโดยให้ผู้ใช้เลือกรายการจากการเช็คที่หน้ารายการที่เลือก

ส่วนการนำมาตกแต่งฟอร์มสามารถทำได้ 2 วิธี คือ

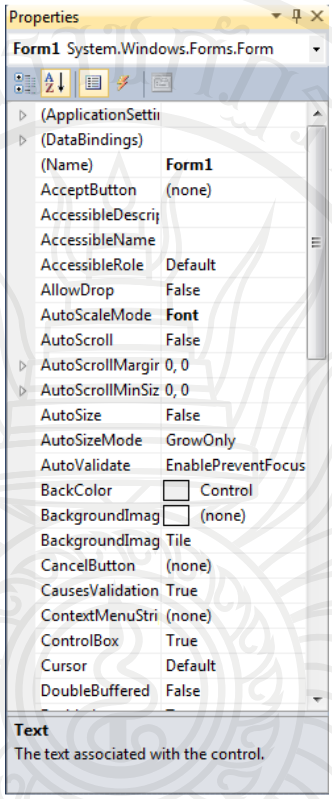
วิธีที่ 1 เลือก Common Controls ที่ต้องการโดยการคลิกเมาส์ จากนั้นเลือกตำแหน่งที่ต้องการบนฟอร์มแล้วคลิกเมาส์อีก 1 ครั้ง

วิธีที่ 2 เลือก Common Controls ที่ต้องการโดยการคลิกเมาส์ แล้วลากมาวางตำแหน่งที่ต้องการบนฟอร์ม ดังภาพที่ 4.2



ภาพที่ 4.2 การวาง Common Controls ในฟอร์ม

การกำหนดคุณสมบัติของฟอร์มหรือ Common Controls ต่าง ๆ สามารถทำได้โดยเลือกวัตถุนั้นและกำหนดในส่วนของหน้าต่าง Properties ดังภาพที่ 4.3



ภาพที่ 4.3 หน้าต่าง Properties ของฟอร์ม

สำหรับ Properties สำคัญที่ใช้ในการกำหนดค่าต่าง ๆ เพื่อให้ฟอร์มมีความสวยงามเหมาะสมกับการทำงานของโปรแกรมมีดังนี้

ตารางที่ 4.1 Properties สำคัญของฟอร์ม

ชื่อของ Properties	การทำงาน
Name	ใช้สำหรับกำหนดชื่อของฟอร์ม
Text	ใช้สำหรับกำหนดข้อความเพื่อแสดงที่แถบชื่อเรื่อง (TitleBar)
Size	ใช้สำหรับระบุความกว้างและความสูงของฟอร์ม
BackColor	ใช้สำหรับกำหนดสีพื้นของฟอร์ม
FormBorderStyle	ใช้สำหรับกำหนดรูปแบบของเส้นขอบ
Font	ใช้สำหรับกำหนดรูปแบบของตัวอักษร
ForeColor	ใช้สำหรับกำหนดสีตัวอักษรที่อยู่บนฟอร์ม

ที่มา : (Microsoft, 2016)

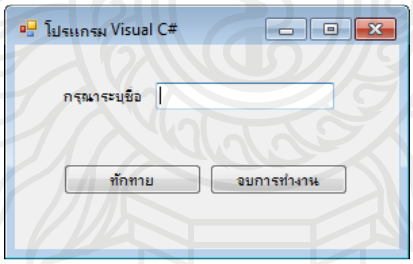
สำหรับเมธอดที่สำคัญสำหรับการทำงานของฟอร์มเพื่อใช้ในการทำงานต่าง ๆ ร่วมกับฟอร์ม มีดังตารางที่ 4.2

ตารางที่ 4.2 เมธอดสำคัญของฟอร์ม

ชื่อของเมธอด	การทำงาน
Show	เป็นเมธอดที่สำหรับเรียกฟอร์มขึ้นมาแสดงผลมีผลเช่นเดียวกับการกำหนด Property ของ Visible = True
ShowDialog	เป็นเมธอดที่เรียกฟอร์มแสดงผลแบบไดอะลอก (Dialog) เมื่อเรียกขึ้นมาแล้วต้องปิดฟอร์มนี้ก่อนจึงจะไปทำงานในฟอร์มอื่นได้
Hide	ใช้สำหรับซ่อนฟอร์ม
Activate	ใช้สำหรับเรียกฟอร์มให้พร้อมสำหรับการทำงาน
SetDesktopLocation	ใช้สำหรับกำหนดตำแหน่งพิกัดที่จะแสดงบนหน้าจอของฟอร์ม
Close	ใช้สำหรับปิดการทำงานของฟอร์ม

ที่มา : (Microsoft, 2016)

ตัวอย่างที่ 4.1 ออกแบบฟอร์มตามภาพที่ 4.4 และกำหนด Properties ต่าง ๆ ดังนี้



ภาพที่ 4.4 ออกแบบฟอร์มโปรแกรมวิชวลซีชาร์ป

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmVC
- Text กำหนดเป็น “โปรแกรม Visual C#”

กำหนด Properties สำหรับ Label
- Name กำหนดเป็น lblName
- Text กำหนดเป็น “กรุณาระบุชื่อ”

กำหนด Properties สำหรับ TextBox
- Name กำหนดเป็น txtName
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ Button1
- Name กำหนดเป็น btnGreet

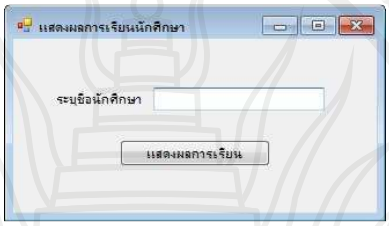
Text กำหนดเป็น “ทักทาย”

กำหนด Properties สำหรับ Button2

Name กำหนดเป็น btnExit

Text กำหนดเป็น “จบการทำงาน”

ตัวอย่างที่ 4.2 ออกแบบฟอร์มตามภาพที่ 4.5 และกำหนด Properties ต่าง ๆ ดังนี้



ภาพที่ 4.5 ออกแบบฟอร์มแสดงผลการเรียนนักศึกษา

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmReport

Text กำหนดเป็น “แสดงผลการเรียนนักศึกษา”

กำหนด Properties สำหรับ Label

Name กำหนดเป็น lblName

Text กำหนดเป็น “ระบุชื่อนักศึกษา”

กำหนด Properties สำหรับ TextBox

Name กำหนดเป็น txtName

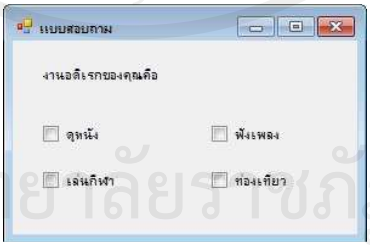
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ Button

Name กำหนดเป็น btnShow

Text กำหนดเป็น “แสดงผลการเรียน”

ตัวอย่างที่ 4.3 ออกแบบฟอร์มตามภาพที่ 4.6 และกำหนด Properties ต่าง ๆ ดังนี้



ภาพที่ 4.6 ออกแบบฟอร์มแบบสอบถาม

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmAsk

Text กำหนดเป็น “แบบสอบถาม”

กำหนด Properties สำหรับ Label

Name กำหนดเป็น lblHobby

Text กำหนดเป็น “งานอดิเรกของคุณคือ”

กำหนด Properties สำหรับ CheckBox1

Name กำหนดเป็น chkMovie

Text กำหนดเป็น “ดูหนัง”

กำหนด Properties สำหรับ CheckBox2

Name กำหนดเป็น chkMusic

Text กำหนดเป็น “ฟังเพลง”

กำหนด Properties สำหรับ CheckBox3

Name กำหนดเป็น chkSport

Text กำหนดเป็น “เล่นกีฬา”

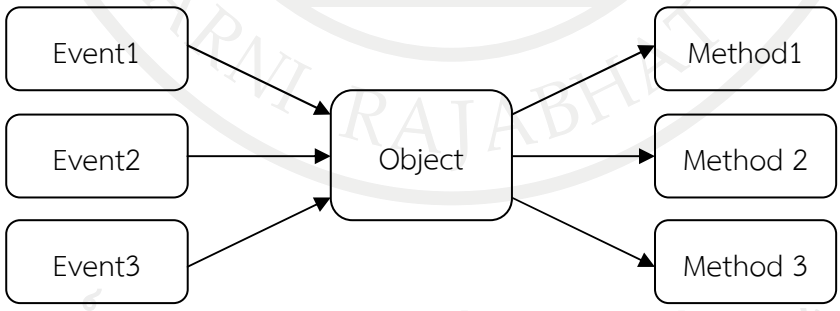
กำหนด Properties สำหรับ CheckBox4

Name กำหนดเป็น chkTour

Text กำหนดเป็น “ท่องเที่ยว”

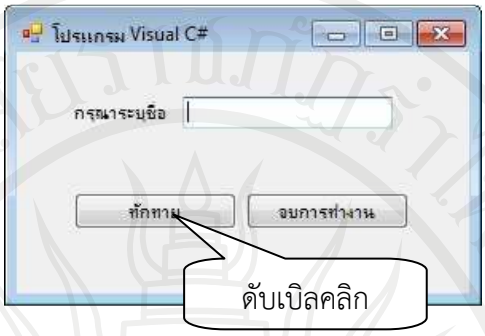
4.2 การจัดการเหตุการณ์

การทำงานต่าง ๆ บนฟอร์มจะทำงานในลักษณะที่ผู้ใช้จะเป็นคนทำให้เกิดเหตุการณ์หรือเกิดอีเวนท์ (Event) ขึ้นที่วัตถุ ซึ่งเรียกรูปแบบการทำงานนี้ว่าการเขียนโปรแกรมแบบอีเวนท์ไดร์ฟเวน (Event-Driven Programming) โดยเมื่อเกิดอีเวนท์แล้ววิชวลซีชาร์ปจะไปค้นหาว่ามีการเขียนคำสั่งสำหรับการทำงานนั้นหรือไม่ ถ้ามีก็จะทำงานตามคำสั่งที่กำหนดไว้ แต่ถ้าไม่มีก็จะเหมือนกับไม่มีอะไรเกิดขึ้น ดังภาพที่ 4.7 (ธีระพล ลิ้มศรัทธา, 2553)



ภาพที่ 4.7 การทำงานเมื่อมีเหตุการณ์เกิดขึ้นกับวัตถุ

โดยปกติการเขียนเมธอดเพื่อจัดการกับอีเวนท์สามารถทำได้โดยการดับเบิลคลิกเมาส์ที่วัตถุใด ๆ ซึ่งโปรแกรมวิชวลซีชาร์ปจะสร้างอีเวนท์ขึ้นมาให้โดยอัตโนมัติ ดังภาพที่ 4.8



ภาพที่ 4.8 การสร้างอีเวนท์สำหรับ Button

เมื่อดับเบิลคลิกแล้วโปรแกรมจะสร้างเมธอดสำหรับอีเวนท์ซึ่งสามารถเขียนคำสั่งใส่เข้าไปเพื่อให้เกิดการทำงานดังนี้

```
private void btnGreet_Click(object sender, EventArgs e)
{
  //ส่วนที่ใช้สำหรับใส่คำสั่งใน Method
}
```

จากตัวอย่างจะเห็นว่าเมื่อมีการดับเบิลคลิกที่ Button จะเกิดอีเวนท์อัตโนมัติ คือ Click หมายความว่า ถ้า Button ถูกคลิกจากผู้ใช้ โปรแกรมจะสั่งให้มาทำงานที่เมธอด btnGreet_Click ทันที แต่ถ้าต้องการสร้างอีเวนท์เพิ่มเติมสามารถเลือกได้จากปุ่ม ที่หน้าต่าง Properties สำหรับอีเวนท์ที่สำคัญของฟอร์มมีดังตารางที่ 4.3 และตัวอย่างตามภาพที่ 4.9

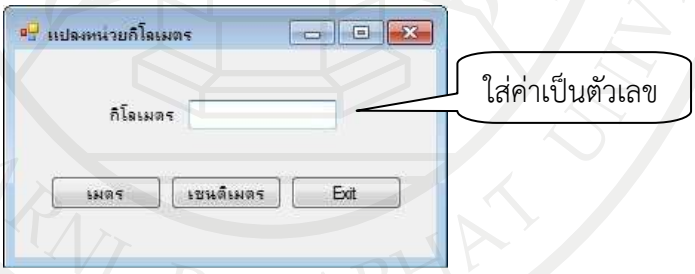
ตารางที่ 4.3 อีเวนท์สำคัญของฟอร์ม

ชื่อของ Event	เหตุการณ์
Load	เกิดขึ้นเมื่อฟอร์มถูกเรียกขึ้นมาใช้งาน
FormClosing	เกิดขึ้นเมื่อฟอร์มกำลังปิด
FormClosed	เกิดขึ้นเมื่อฟอร์มปิดแล้ว
Resize	เกิดขึ้นเมื่อฟอร์มถูกปรับขนาดให้เปลี่ยนไป
Click	เกิดขึ้นเมื่อคลิกลงไปบนฟอร์ม

ที่มา : (Microsoft, 2016)

ภาพที่ 4.9 อีเวนท์ของวัตถุ

ตัวอย่างที่ 4.4 จงเขียนโปรแกรมแปลงหน่วยกิโลเมตร โดยให้ผู้ใช้ใส่ค่าใน TextBox และแสดงผลเป็น MessageBox ตามปุ่มที่กด ดังภาพที่ 4.10



ภาพที่ 4.10 หน้าจอโปรแกรมแปลงหน่วยกิโลเมตร

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmConv

Text กำหนดเป็น “แปลงหน่วยกิโลเมตร”

กำหนด Properties สำหรับ Label

Name กำหนดเป็น lblKm

Text กำหนดเป็น “กิโลเมตร”

กำหนด Properties สำหรับ TextBox

Name กำหนดเป็น txtKm

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ Button1

Name กำหนดเป็น btnM

Text กำหนดเป็น “เมตร”

กำหนด Properties สำหรับ Button2

Name กำหนดเป็น btnCm

Text กำหนดเป็น “เซนติเมตร”

กำหนด Properties สำหรับ Button3

Name กำหนดเป็น btnExit

Text กำหนดเป็น “Exit”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex4_4
10  {
11      public partial class frmConv : Form
12      {
13          public frmConv()
14          {
15              InitializeComponent();
16          }
17          private void btnM_Click(object sender, EventArgs e)
18          {
19              int ans;
20              ans = int.Parse(textBox1.Text) * 1000;
```

```
21              MessageBox.Show(textBox1.Text + " กิโลเมตร แปลงเป็นเมตร เท่ากับ " +
    ans.ToString("#,###") + " เมตร");
22          }
23          private void btnCm_Click(object sender, EventArgs e)
24          {
25              int ans;
26              ans = int.Parse(textBox1.Text) * 1000 * 100;
27              MessageBox.Show(textBox1.Text + " กิโลเมตร แปลงเป็นเซนติเมตร เท่ากับ
    " + ans.ToString("#,###") + " เซนติเมตร");
28          }
29          private void btnExit_Click(object sender, EventArgs e)
30          {
31              this.Close();
32          }
33      }
34  }
```

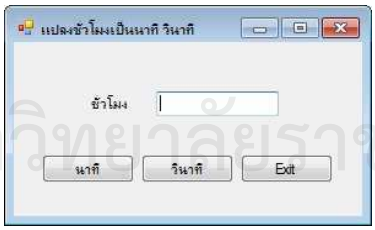
อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 4.4

บรรทัดที่ 17, 23 และ 29 เป็นส่วนของเมธอดที่ทำงานเมื่อเกิดอีเวนท์ขึ้น

บรรทัดที่ 20 และ 26 เป็นการคำนวณและแปลงค่าโดยใช้เมธอด Parse

เมื่อเกิดอีเวนท์การคลิกเมาส์ที่ปุ่ม btnM ขึ้นเมธอดชื่อ btnM_Click จะถูกเรียกขึ้นมาทำงาน โดยนำค่าจาก TextBox มาแปลงเป็นตัวเลข เนื่องจากค่าที่อยู่ใน TextBox จะเป็นข้อความไม่สามารถนำมาคำนวณได้ จึงต้องนำมาแปลงค่าให้เป็นตัวเลขเสียก่อน โดยใช้เมธอด Parse เมื่อได้ค่าตัวเลขที่เป็นกิโลเมตรแล้ว จะนำมาแปลงเป็นเมตรและแสดงผลใน MessageBox ที่มีการกำหนดรูปแบบสำหรับแสดงผลเป็นตัวเลข โดยใช้ ToString("#,###") ส่วนปุ่ม btnCm ก็จะทำลักษณะเดียวกับปุ่ม btnM คือ แปลงค่าใน TextBox เพื่อมาคำนวณหาค่าที่เป็นเซนติเมตร ปุ่ม btnExit จะใช้สำหรับการสั่งให้ฟอร์มปิดหรือหยุดการทำงานลง

ตัวอย่างที่ 4.5 จงเขียนโปรแกรมแปลงหน่วยชั่วโมง โดยให้ผู้ใช้ใส่ค่าใน TextBox และแสดงผลเป็น MessageBox ตามปุ่มที่กด ดังภาพที่ 4.11



ภาพที่ 4.11 หน้าจอโปรแกรมแปลงชั่วโมงเป็นนาที วินาที

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmHour

Text กำหนดเป็น “แปลงชั่วโมงเป็นนาที วินาที”

กำหนด Properties สำหรับ Label

Name กำหนดเป็น lblHour

Text กำหนดเป็น “ชั่วโมง”

กำหนด Properties สำหรับ TextBox

Name กำหนดเป็น txtHour

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ Button1

Name กำหนดเป็น btnMin

Text กำหนดเป็น “นาที”

กำหนด Properties สำหรับ Button2

Name กำหนดเป็น btnSec

Text กำหนดเป็น “วินาที”

กำหนด Properties สำหรับ Button3

Name กำหนดเป็น btnExit

Text กำหนดเป็น “Exit”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex4_5
10  {
11      public partial class frmHour : Form
12      {
13          public frmHour()
14          {
15              InitializeComponent();
16          }
```

```
17          private void btnMin_Click(object sender, EventArgs e)
18          {
19              int ans;
20              ans = int.Parse(textBox1.Text) * 60;
21              MessageBox.Show(textBox1.Text + " ชั่วโมง แปลงเป็นนาที เท่ากับ " +
        ans.ToString("#,###") + " นาที");
22          }
23          private void btnSec_Click(object sender, EventArgs e)
24          {
25              int ans;
26              ans = int.Parse(textBox1.Text) * 60 * 60;
27              MessageBox.Show(textBox1.Text + " ชั่วโมง แปลงเป็นวินาที เท่ากับ " +
        ans.ToString("#,###") + " วินาที");
28          }
29          private void btnExit_Click(object sender, EventArgs e)
30          {
31              this.Close();
32          }
33      }
34  }
```

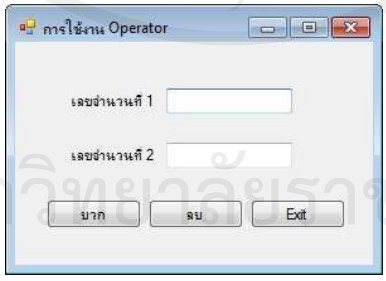
อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 4.5

บรรทัดที่ 17, 23 และ 29 เป็นส่วนของเมธอดที่ทำงานเมื่อเกิดอีเวนท์ขึ้น

บรรทัดที่ 20 และ 26 เป็นการคำนวณและแปลงค่าโดยใช้เมธอด Parse

ในตัวอย่างที่ 4.5 นี้จะมีชุดคำสั่งคล้ายกับตัวอย่างที่ 4.4 คือ รับค่าแล้วนำมาแสดงผลผ่านทาง MessageBox ตามการทำงานของแต่ละปุ่มโดยรับค่าเป็นชั่วโมง จากนั้นก็แสดงผลเป็นนาทีหรือวินาที

ตัวอย่างที่ 4.6 จงเขียนโปรแกรมรับค่าเลข 2 จำนวน แล้วดำเนินการตามปุ่มที่กด เพื่อแสดงผลลัพธ์ใน MessageBox ดังภาพที่ 4.12



ภาพที่ 4.12 หน้าจอโปรแกรมรับค่าตัวเลข

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmOpt
- Text กำหนดเป็น การใช้งาน Operator

กำหนด Properties สำหรับ Label1
- Name กำหนดเป็น lblNum1
- Text กำหนดเป็น เลขจำนวนที่ 1

กำหนด Properties สำหรับ Label2
- Name กำหนดเป็น lblNum2
- Text กำหนดเป็น เลขจำนวนที่ 2

กำหนด Properties สำหรับ TextBox1
- Name กำหนดเป็น txtNum1
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ TextBox2
- Name กำหนดเป็น txtNum2
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ Button1
- Name กำหนดเป็น btnAdd
- Text กำหนดเป็น บวก

กำหนด Properties สำหรับ Button2
- Name กำหนดเป็น btnSub
- Text กำหนดเป็น ลบ

กำหนด Properties สำหรับ Button3
- Name กำหนดเป็น btnExit
- Text กำหนดเป็น Exit

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex4_6
10  {
```

```
11          public partial class frmOpt : Form
12          {
13              public frmOpt()
14              {
15                  InitializeComponent();
16              }
17              private void btnAdd_Click(object sender, EventArgs e)
18              {
19                  int ans;
20                  ans = int.Parse(textBox1.Text) + int.Parse(textBox2.Text);
21                  MessageBox.Show("ผลลัพธ์ของการบวกเลข 2 จำนวน คือ " + ans, "แสดง
        ผลลัพธ์");
22              }
23              private void btnSub_Click(object sender, EventArgs e)
24              {
25                  int ans;
26                  ans = int.Parse(textBox1.Text) - int.Parse(textBox2.Text);
27                  MessageBox.Show("ผลลัพธ์ของการลบเลข 2 จำนวน คือ " + ans, "แสดง
        ผลลัพธ์");
28              }
29              private void btnExit_Click(object sender, EventArgs e)
30              {
31                  this.Close();
32              }
33          }
34      }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 4.6

บรรทัดที่ 17, 23 และ 29 เป็นส่วนของเมธอดที่ทำงานเมื่อเกิดอีเวนท์ขึ้น

บรรทัดที่ 20 และ 26 เป็นการคำนวณและแปลงค่าโดยใช้เมธอด Parse

เมื่อเกิดอีเวนท์ที่ปุ่ม btnAdd โดยผู้ใช้ ค่าที่อยู่ใน txtNum1 และ txtNum2 จะถูกนำมาแปลงให้เป็นตัวเลขโดยใช้เมธอด Parse หลังจากแปลงแล้วค่าจะถูกนำมาบวก ผลลัพธ์จะถูกนำมาเก็บที่ตัวแปร ans และแสดงใน MessageBox ส่วนปุ่ม btnDel จะทำงานลักษณะเดียวกับปุ่ม btnAdd คือนำค่าจาก txtNum1 และ txtNum2 มาแปลงเป็นตัวเลข และนำค่ามาลบกัน เพื่อแสดงใน MessageBox ส่วนปุ่ม btnExit ใช้สำหรับออกจากโปรแกรม

4.3 สรุป

การพัฒนาส่วนติดต่อกราฟิกกับผู้ใช้ในรูปแบบที่เป็น Windows Forms Application สามารถสร้างได้จาก Common Controls ที่วิชวลซีชาร์ปมีให้ โดยเลือกจากแถบเครื่องมือมาตกแต่งลงในฟอร์มเมื่อผู้ใช้ทำงานกับฟอร์มจะเกิดอีเวนท์ขึ้น ผู้พัฒนาจะต้องกำหนดคำสั่งภายในเมธอดเพื่อควบคุมการทำงานหลังจากเกิดอีเวนท์ที่วัตถุ

เมื่อผู้ใช้ทำให้เกิดอีเวนท์ขึ้น โปรแกรมจะตรวจสอบว่ามีคำสั่งที่อยู่ภายในเมธอดที่ควบคุมการทำงานของอีเวนท์หรือไม่ ถ้ามีโปรแกรมจะเข้าไปทำงานตามคำสั่งนั้น แต่ถ้าไม่มีก็จะเหมือนกับไม่มีอะไรเกิดขึ้น ดังนั้นในการเขียนโปรแกรมจึงจำเป็นจะต้องเข้าใจลักษณะของอีเวนท์ที่เกิดขึ้นจากการทำงานของผู้ใช้ เพื่อให้สามารถกำหนดคำสั่งที่ใช้สำหรับการทำงานของโปรแกรมเหมาะสมกับความต้องการพื้นฐานของโปรแกรม โดยปกติการเขียนคำสั่งในเมธอดเพื่อควบคุมอีเวนท์สามารถทำได้โดยการสร้างอีเวนท์ที่ต้องการจากการกระทำ เช่น ถ้าต้องการสร้างอีเวนท์การคลิกที่ปุ่ม สามารถกดดับเบิลคลิกที่ปุ่ม โปรแกรมก็จะสร้างเมธอดสำหรับการทำงานเมื่อคลิกที่ปุ่ม เป็นต้น

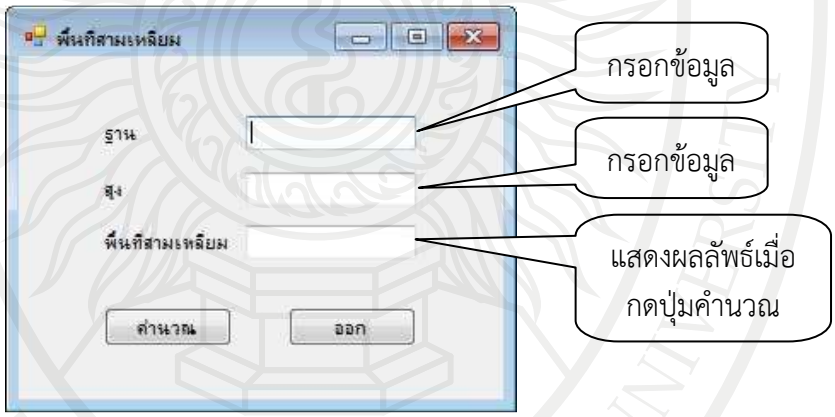
แบบฝึกหัดบทที่ 4

1. การเขียนโปรแกรมแบบอีเวนท์ไดร์ฟเวนหมายถึงอะไร
2. ถ้าต้องการสร้างฟอร์มสำหรับรับข้อมูลลูกค้า ต้องใช้ Common Controls อะไรบ้าง
3. การออกแบบสำหรับรับข้อมูลเพศ ควรใช้ Common Controls ใด
4. ในส่วนแถบชื่อเรื่องของฟอร์มสามารถตั้งค่าได้จาก Properties ใด
5. เมธอดที่สำคัญของฟอร์มมีอะไรบ้าง
6. ถ้าต้องการเขียนคำสั่งขณะฟอร์มกำลังปิด ต้องเขียนที่อีเวนท์ใด
7. ComboBox ใช้สำหรับออกแบบฟอร์มแบบใดได้บ้าง
8. อีเวนท์ Resize จะเกิดขึ้นเมื่อใด
9. จงออกแบบหน้าจอรับข้อมูลจากข้อมูลต่อไปนี้

   รับข้อมูลค่าตัวเลข 2 ตัว

   มีปุ่มสำหรับการคูณ ปุ่มสำหรับการหาร และปุ่มสำหรับออกจากโปรแกรม
10. จงเขียนโปรแกรมคำนวณพื้นที่สามเหลี่ยมโดยมีรายละเอียดดังนี้

    ออกแบบหน้าจอดังตัวอย่าง



กำหนด Properties สำหรับ Form

Name	กำหนดเป็น frmTriangle
Text	กำหนดเป็น “พื้นที่สามเหลี่ยม”

กำหนด Properties สำหรับ Label1

Name	กำหนดเป็น lblBase
Text	กำหนดเป็น “ฐาน”

กำหนด Properties สำหรับ Label2

Name	กำหนดเป็น lblHigh
Text	กำหนดเป็น “สูง”

กำหนด Properties สำหรับ Label3

Name กำหนดเป็น lblArea
Text กำหนดเป็น “พื้นที่สามเหลี่ยม”

กำหนด Properties สำหรับ TextBox1
Name กำหนดเป็น txtBase
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ TextBox2
Name กำหนดเป็น txtHigh
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ TextBox3
Name กำหนดเป็น txtArea
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ Button1
Name กำหนดเป็น btnCalculate
Text กำหนดเป็น “คำนวณ”

กำหนด Properties สำหรับ Button2
Name กำหนดเป็น btnExit
Text กำหนดเป็น “ออก”

เอกสารอ้างอิง

ธีระพล ลิ้มศรัทธา. (2553). **การเขียนโปรแกรมเชิงวัตถุด้วย Visual Basic.NET**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

นิรันดร์ ประวิทย์ธนา. (2545). **เก่ง C# ให้ครบสูตร**. กรุงเทพฯ : วิตตี้ กรุ๊ป.

ศุภชัย สมพานิช. (2556). **คู่มือเรียนและใช้งาน Visual C#**. กรุงเทพฯ: สวัสดี ไอที.

Microsoft. (2016). **C# Programming Guide**. (online). Available : https://msdn. microsoft.com/en- us/library/67ef8sbd(v=vs.100).aspx. 6 February 2016.

_____. (2016). **Form Class**. (online). Available : https://msdn.microsoft.com/ en-us/library/system.windows.forms.form(v=vs.110).aspx. 6 February 2016.

แผนบริหารประจำบทที่ 5

เนื้อหา

บทที่ 5 การใช้คำสั่งควบคุมและการตัดสินใจ

5.1 การตัดสินใจ

5.2 การทำงานซ้ำ

5.3 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. ใช้คำสั่งการตัดสินใจในการเขียนโปรแกรมได้
2. ใช้คำสั่งการทำงานซ้ำในการเขียนโปรแกรมได้
3. อธิบายรูปแบบของคำสั่งการควบคุมได้
4. ใช้คำสั่งควบคุมที่เหมาะสมในการทำงานได้
5. ประยุกต์ใช้คำสั่งควบคุมและตัดสินใจในการเขียนโปรแกรมได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน และเนื้อหาเพิ่มเติมจากอินเทอร์เน็ต
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ผู้สอนอธิบายตัวอย่างในเอกสารประกอบการสอน
5. ให้ผู้เรียนทดลองทำตามโปรแกรมตัวอย่าง
6. ให้ผู้เรียนฝึกปฏิบัติ
7. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 5

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 5
4. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน

3. ประเมินจากการซักถามของผู้เรียน
4. ประเมินจากแบบฝึกหัดบทที่ 5

บทที่ 5
การใช้คำสั่งควบคุมและการตัดสินใจ

การเขียนโปรแกรมในปัจจุบันมีการทำงานที่ซับซ้อนทั้งรูปแบบและเงื่อนไขที่แตกต่างกัน ทำให้โปรแกรมต้องสามารถเลือกทางสำหรับการทำงาน ดังนั้นเพื่อให้การทำงานของโปรแกรมไม่เกิดผิดพลาดหรือสะดุดจึงจำเป็นต้องมีการควบคุมทิศทางการทำงาน ทำให้โปรแกรมสามารถทำงานด้วยตนเองโดยผู้ใช้ไม่ต้องคอยกำหนดการทำงานเป็นขั้นตอน ผู้เขียนจะต้องระบุหน้าที่ให้โปรแกรมมีทางเลือกสำหรับการทำงานด้วยเงื่อนไข โดยการควบคุมการทำงานของโปรแกรมมีอยู่ 2 แบบ คือ

1) การตัดสินใจ (Decision)
2) การทำงานซ้ำ (Iteration)

5.1 การตัดสินใจ

การทำให้โปรแกรมสามารถทำงานแบบมีเงื่อนไข (Condition) ได้จะต้องมีการกำหนดทางเลือก โดยทางเลือกอาจจะมี 1 ทาง 2 ทาง หรือมากกว่านั้นก็ได้ แต่โปรแกรมต้องเลือกทางที่สามารถทำงานได้ถูกต้อง ซึ่งในวิชวลซีชาร์ปมีคำสั่งที่ใช้สำหรับการตัดสินใจดังนี้ (ศุภชัย สมพานิช, 2556)

5.1.1 คำสั่ง if...else

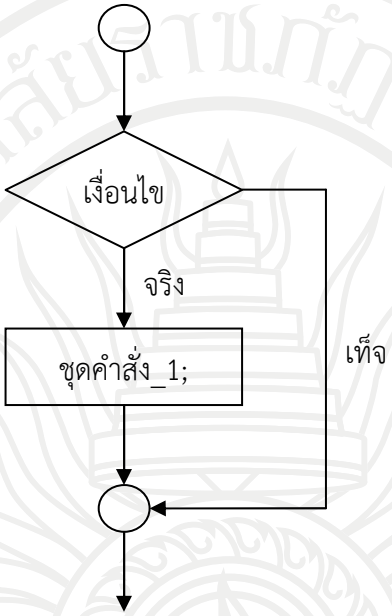
การเขียนโปรแกรมด้วยคำสั่ง if...else จะมีเงื่อนไขที่ตรงตัว คือ ถ้าเงื่อนไขที่อยู่ใน if เป็นจริงจะต้องทำคำสั่งที่อยู่หลัง if แต่ถ้าเงื่อนไขที่อยู่ใน if เป็นเท็จ จะต้องทำคำสั่งที่อยู่หลัง else แต่ในกรณีที่ไม่มี else ตามหลัง if จะแสดงว่าถ้าเงื่อนไขเป็นเท็จก็ไม่ต้องทำคำสั่งใด ๆ ซึ่งสามารถแบ่งการใช้ if...else เป็นรูปแบบต่าง ๆ ได้ดังนี้ (นิรันดร์ ประวิทย์ธนา, 2545)

รูปแบบทางเลือกเดียว

```
if ( เงื่อนไข )
{
   คำสั่ง_1 ;
   คำสั่ง_2 ;
     .....
   คำสั่ง_N ;
}
```


ในส่วนของเงื่อนไขส่วนใหญ่จะใช้การเปรียบเทียบทางตรรกะ โดยใช้ตัวดำเนินการ หมายความว่า ถ้าเงื่อนไขเป็นจริง โปรแกรมจะเข้ามาทำงานในคำสั่ง_1 คำสั่ง_2 มาเรื่อย ๆ จนจบ

บล็อกของคำสั่ง แต่ถ้าเงื่อนไขเป็นเท็จโปรแกรมจะไม่ทำงานคำสั่งในบล็อก ซึ่งสามารถอธิบายเป็นแผนผังการทำงานได้ดังภาพที่ 5.1



ภาพที่ 5.1 แผนผังการทำงานของคำสั่ง if

จากภาพที่ 5.1 จะเห็นว่าโปรแกรมจะตรวจสอบเงื่อนไขก่อนเป็นอันดับแรก ถ้าเงื่อนไขเป็นจริงโปรแกรมจะเข้ามาทำงานในส่วนของคำสั่ง ซึ่งถ้ามีคำสั่งเดียวอาจจะไม่ต้องใส่เครื่องหมาย {...} ก็ได้ แต่ถ้าเป็นชุดคำสั่งต้องใส่อยู่ใน {...} เท่านั้น ถ้าไม่ใส่ {...} โปรแกรมจะทำงานตามคำสั่งแรกเท่านั้น ส่วนถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะไม่ทำงานในชุดคำสั่ง_1 แต่จะข้ามไปทำคำสั่งถัดไป

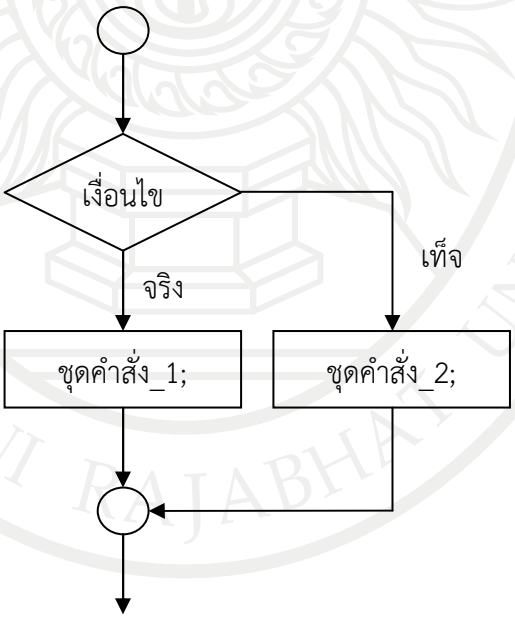
```
if ( เงื่อนไข )
   คำสั่ง_1 ;
คำสั่ง_2 ;
.....
คำสั่ง_N ;
```

จะเห็นว่าการเขียนโปรแกรมที่ไม่ใส่ {...} หลังจากตรวจสอบเงื่อนไขใน if แล้ว ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำตามคำสั่ง_1 และทำคำสั่ง_2 ไปเรื่อย ๆ จนถึงคำสั่งN แต่ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะเริ่มทำตั้งแต่คำสั่ง_2 ไปเรื่อย ๆ จนถึงคำสั่ง_N โดยเว้นการทำงานในคำสั่ง1 ซึ่งการใช้คำสั่ง if แบบ 1 ทางเลือกไม่ค่อยจะมีความชัดเจนในการเขียนโปรแกรมเท่าไร อาจจะทำให้ผู้เขียนโปรแกรมเกิดความสับสนในการที่จะให้โปรแกรมทำงานตามที่ต้องการได้

รูปแบบ 2 ทางเลือก

```
if ( เงื่อนไข )
{
   <ชุดคำสั่งที่เงื่อนไขเป็นจริง>;
}
else
{
   <ชุดคำสั่งที่เงื่อนไขเป็นเท็จ>;
}
```

ในการทำงานจะเห็นว่าเมื่อโปรแกรมตรวจสอบเงื่อนไขแล้ว ถ้าผลที่ได้เป็นจริง โปรแกรมจะเข้ามาทำงานใน <ชุดคำสั่งที่เงื่อนไขเป็นจริง> ส่วนถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะเข้ามาทำงานในคำสั่งที่อยู่หลัง else คือ <ชุดคำสั่งที่เงื่อนไขเป็นเท็จ> แทน ซึ่งคำสั่ง else จะต้องอยู่หลังคำสั่ง if เท่านั้น ไม่สามารถอยู่ก่อนได้ ถ้าในกรณีที่ไม่ใส่เครื่องหมาย {...} คำสั่งหลัง if จะมีได้เพียงแค่ 1 คำสั่งเท่านั้น ส่วนหลัง else ถ้าไม่ใส่ {...} โปรแกรมจะทำงานหลังคำสั่ง else เพียง 1 คำสั่ง ซึ่งสามารถอธิบายเป็นแผนผังการทำงานได้ดังภาพที่ 5.2



ภาพที่ 5.2 แผนผังการทำงานของคำสั่ง if และ else

```
if ( เงื่อนไข )
   คำสั่ง_1 ;
else
   คำสั่ง_2 ;
คำสั่ง_3 ;
```

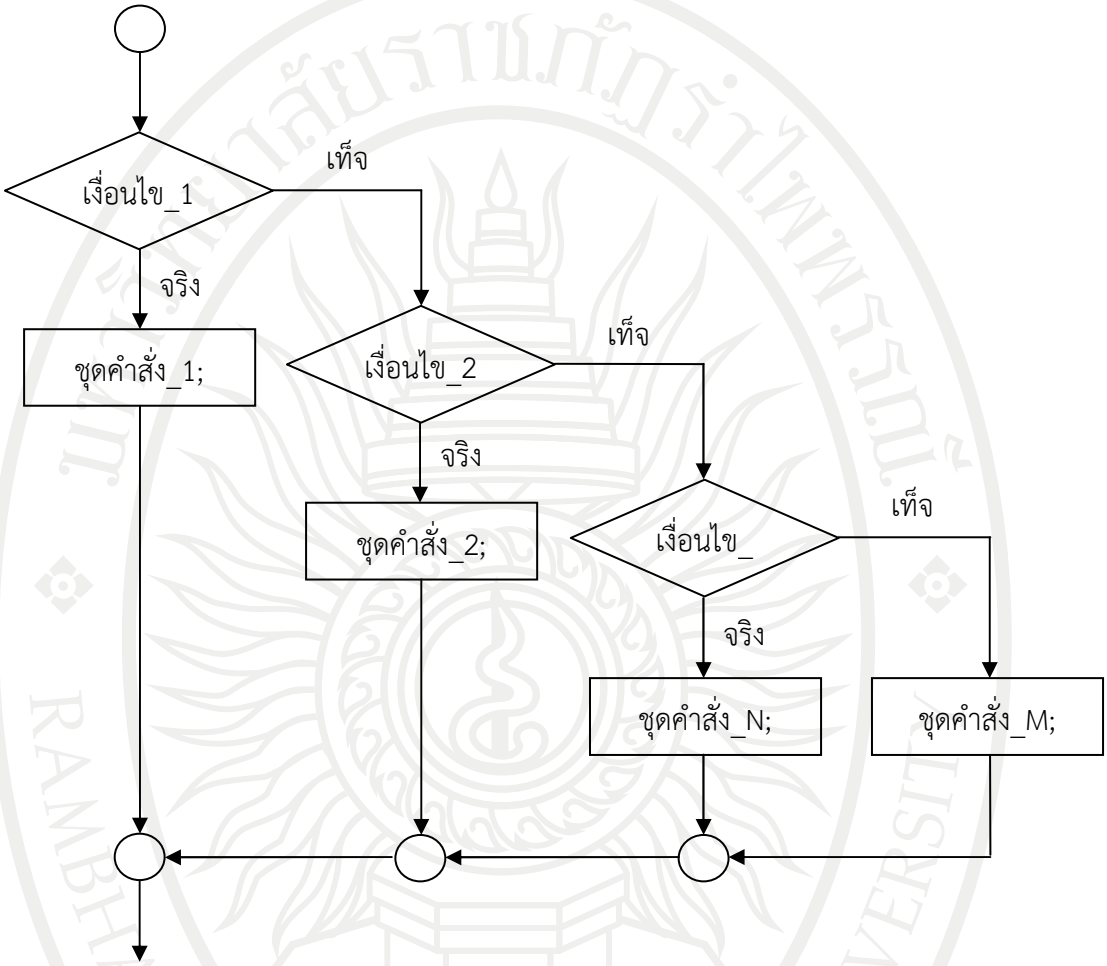
เมื่อเงื่อนไขใน if เป็นจริง โปรแกรมจะทำงานในคำสั่ง_1 แล้วลงมาทำงานที่คำสั่ง_3 ส่วนถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานในคำสั่ง_2 แล้วมาทำงานที่คำสั่ง_3

รูปแบบหลายทางเลือก

```
if ( เงื่อนไข_1 )
{
   <ชุดคำสั่งที่เงื่อนไข_1 เป็นจริง>
}
else if ( เงื่อนไข_2 )
{
   <ชุดคำสั่งที่เงื่อนไข_2 เป็นจริง>;
}
         ...
         ...
         ...
else if ( เงื่อนไข_N )
{
   <ชุดคำสั่งที่เงื่อนไข_N เป็นจริง>;
}
else
{
   <ชุดคำสั่งที่เงื่อนไข_N เป็นเท็จ>;
}
```

เมื่อโปรแกรมตรวจสอบเงื่อนไข_1 แล้วถ้าเป็นจริงโปรแกรมจะเข้ามาทำงานใน <ชุดคำสั่งที่เงื่อนไข_1 เป็นจริง> แต่ถ้าเงื่อนไข_1 เป็นเท็จ โปรแกรมจะเข้ามาตรวจสอบเงื่อนไข_2 ถ้าเป็นจริงโปรแกรมจะงานใน <ชุดคำสั่งที่เงื่อนไข_2 เป็นจริง> แต่ถ้าเงื่อนไข_2 เป็นเท็จ โปรแกรมจะตรวจสอบในเงื่อนไขต่อมาลักษณะเดียวกันจนถึงเงื่อนไข_N ถ้าเป็นจริง โปรแกรมจะทำงานใน

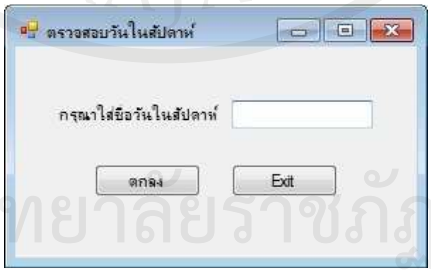
<ชุดคำสั่งที่เงื่อนไข_N เป็นจริง> แต่ถ้าเป็นเท็จโปรแกรมจะทำงานใน <ชุดคำสั่งที่เงื่อนไข_N เป็นเท็จ> แทน สามารถเขียนแผนผังการทำงานดังภาพที่ 5.3



ภาพที่ 5.3 แผนผังการทำงานของคำสั่ง if และ else แบบหลายทางเลือก

ตัวอย่างที่ 5.1 การใช้งานคำสั่ง if...else

สร้างหน้าจอตามตัวอย่างต่อไปนี้ตามภาพที่ 5.4



ภาพที่ 5.4 หน้าจอตัวอย่าง if...else

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmDay

Text กำหนดเป็น “ตรวจสอบวันในสัปดาห์”

กำหนด Properties สำหรับ label

Name กำหนดเป็น lblDay

Text กำหนดเป็น “กรุณาใส่ชื่อวันในสัปดาห์”

กำหนด Properties สำหรับ textBox

Name กำหนดเป็น txtDay

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1

Name กำหนดเป็น btnOk

Text กำหนดเป็น “ตกลง”

กำหนด Properties สำหรับ button2

Name กำหนดเป็น btnExit

Text กำหนดเป็น “Exit”

ชุดคำสั่งของโปรแกรมในกรณีที่ใช้ if แบบทางเลือกเดียว

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex5_1
10  {
11      public partial class frmDay : Form
12      {
13          public frmDay()
14          {
15              InitializeComponent();
16          }
17          private void btnOk_Click(object sender, EventArgs e)
18          {
19              string _day;
```

```
20              _day = txtDay.Text;
21              if (_day == "อาทิตย์")
22              {
23                  this.BackColor = Color.Red;
24                  MessageBox.Show("วันนี้เป็นวันอาทิตย์ สีแดง");
25              }
26              if (_day == "จันทร์")
27              {
28                  this.BackColor = Color.Yellow;
29                  MessageBox.Show("วันนี้เป็นวันจันทร์ สีเหลือง");
30              }
31              if (_day == "อังคาร")
32              {
33                  this.BackColor = Color.Pink;
34                  MessageBox.Show("วันนี้เป็นวันอังคาร สีชมพู");
35              }
36              if (_day == "พุธ")
37              {
38                  this.BackColor = Color.Green;
39                  MessageBox.Show("วันนี้เป็นวันพุธ สีเขียว");
40              }
41              if (_day == "พฤหัสบดี")
42              {
43                  this.BackColor = Color.Orange;
44                  MessageBox.Show("วันนี้เป็นวันพฤหัส สีส้ม");
45              }
46              if (_day == "ศุกร์")
47              {
48                  this.BackColor = Color.Blue;
49                  MessageBox.Show("วันนี้เป็นวันศุกร์ สีฟ้า");
50              }
51              if (_day == "เสาร์")
52              {
53                  this.BackColor = Color.Magenta;
54                  MessageBox.Show("วันนี้เป็นวันเสาร์ สีม่วง");
55              }
```

```
56              txtDay.Text = "";
57          }
58          private void btnExit_Click(object sender, EventArgs e)
59          {
60              this.Close();
61          }
62      }
63  }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.1

จากตัวอย่างจะเห็นว่าในเมธอด private void btnOk_Click(object sender, EventArgs e) ในบรรทัดที่ 17 จะมีการตรวจสอบค่าใน txtDay ซึ่งจะถูกนำมาเก็บไว้ในตัวแปร _day และนำค่ามาตรวจสอบด้วยคำสั่ง if ในบรรทัดที่ 21 ถ้าค่าในตัวแปร _day ถ้ามีค่าเท่ากับ “อาทิตย์” โปรแกรมจะทำงานในบรรทัดที่ 23 และ 24 คือ คำสั่ง this.BackColor = Color.Red; ซึ่งเป็นการกำหนดค่า Properties ของฟอร์มให้พื้นหลังเป็นสีแดง และคำสั่ง MessageBox.Show("วันนี้เป็นวันอาทิตย์ สีแดง"); แสดง MessageBox ขึ้นมา แต่ถ้าค่าตัวแปรไม่ใช่ “อาทิตย์” โปรแกรมจะตรวจสอบเงื่อนไขถัดไปในบรรทัดที่ 26 คือ “จันทร์” และทำอย่างนี้ไปเรื่อย ๆ

ปัญหาของตัวอย่างนี้ คือ โปรแกรมจะตรวจสอบเงื่อนไขทุก if ทำให้การทำงานซ้ำซ้อน ถึงแม้ว่าโปรแกรมจะทำงานตามเงื่อนไขที่เป็นจริงก่อนหน้านี้แล้วก็ตาม ถ้าต้องการให้โปรแกรมทำงานไม่ซ้ำซ้อนสามารถใช้ else มาช่วยทำให้การทำงานดีขึ้น โดยปรับการทำงานในเมธอด btnOk_Click ดังนี้

```
1           private void btnOk_Click(object sender, EventArgs e)
2           {
3               string _day;
4               _day = txtDay.Text;
5               if (_day == "อาทิตย์")
6               {
7                   this.BackColor = Color.Red;
8                   MessageBox.Show("วันนี้เป็นวันอาทิตย์ สีแดง");
9               }
10              else if (_day == "จันทร์")
11              {
12                  this.BackColor = Color.Yellow;
13                  MessageBox.Show("วันนี้เป็นวันจันทร์ สีเหลือง");
14              }
```

```
15              else if (_day == "อังคาร")
16              {
17                  this.BackColor = Color.Pink;
18                  MessageBox.Show("วันนี้เป็นวันอังคาร สีชมพู");
19              }
20              else if (_day == "พุธ")
21              {
22                  this.BackColor = Color.Green;
23                  MessageBox.Show("วันนี้เป็นวันพุธ สีเขียว");
24              }
25              else if (_day == "พฤหัสบดี")
26              {
27                  this.BackColor = Color.Orange;
28                  MessageBox.Show("วันนี้เป็นวันพฤหัส สีส้ม");
29              }
30              else if (_day == "ศุกร์")
31              {
32                  this.BackColor = Color.Blue;
33                  MessageBox.Show("วันนี้เป็นวันศุกร์ สีฟ้า");
34              }
35              else if (_day == "เสาร์")
36              {
37                  this.BackColor = Color.Magenta;
38                  MessageBox.Show("วันนี้เป็นวันเสาร์ สีม่วง");
39              }
40              else
41              {
42                  MessageBox.Show("ไม่มีวันนี้อยู่ในสัปดาห์");
43              }
44              txtDay.Text = "";
45          }
```

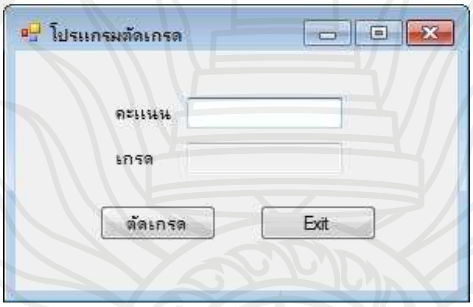
อธิบายชุดคำสั่งของโปรแกรม

จากการใส่คำสั่ง else เข้าไป จะทำให้โปรแกรมทำงานได้ดีขึ้น เนื่องจากเมื่อโปรแกรมทำงานเมื่อเงื่อนไขแรกในบรรทัดที่ 5 ถึง 9 แล้ว จะออกมาทำงานบรรทัดที่ 44 เลย แต่ถ้าเงื่อนไขในบรรทัดที่ 5 เป็นเท็จ โปรแกรมจะตรวจสอบเงื่อนไขในบรรทัดที่ 10 และทำอย่างนี้ไปเรื่อย ๆ จนกว่าจะจบ

เงื่อนไขต่าง ๆ ถ้าเงื่อนไขทุกอย่างเป็นเท็จทั้งหมดจะไม่มีคำสั่งที่อยู่ภายในคำสั่ง if ทำงานเลย และโปรแกรมจะข้ามมาทำงานในบรรทัดที่ 42

ตัวอย่างที่ 5.2 จงเขียนโปรแกรมตัดเกรด โดยมีหน้าจอดังภาพที่ 5.5 และมีเงื่อนไขดังนี้

คะแนน 0 – 59 ได้ผลการเรียนเป็น Fail
คะแนน 60 – 79 ได้ผลการเรียนเป็น Pass
คะแนน 80 – 100 ได้ผลการเรียนเป็น Good
นอกเหนือจากนั้นจะได้ผลการเรียนเป็น Error



ภาพที่ 5.5 หน้าจอโปรแกรมตัดเกรด

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmGrade
- Text กำหนดเป็น “โปรแกรมตัดเกรด”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblPoint
- Text กำหนดเป็น “คะแนน”

กำหนด Properties สำหรับ label2
- Name กำหนดเป็น lblGrade
- Text กำหนดเป็น “เกรด”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtPoint
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2
- Name กำหนดเป็น txtGrade
- ReadOnly กำหนดเป็น True
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnGrade

Text กำหนดเป็น “ตัดเกรด”

กำหนด Properties สำหรับ button2

Name กำหนดเป็น btnExit

Text กำหนดเป็น “Exit”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex5_2
10  {
11      public partial class frmGrade : Form
12      {
13          public frmGrade()
14          {
15              InitializeComponent();
16          }
17          private void btnGrade_Click(object sender, EventArgs e)
18          {
19              int grd = Convert.ToInt16(txtPoint.Text);
20              if (grd >= 0 && grd <= 59)
21                  txtGrade.Text = "Fail";
22              else if (grd >= 60 && grd <= 79)
23                  txtGrade.Text = "Pass";
24              else if (grd >= 80 && grd <= 100)
25                  txtGrade.Text = "Good";
26              else
27                  txtGrade.Text = "Error";
28          }
29          private void btnExit_Click(object sender, EventArgs e)
30          {
```

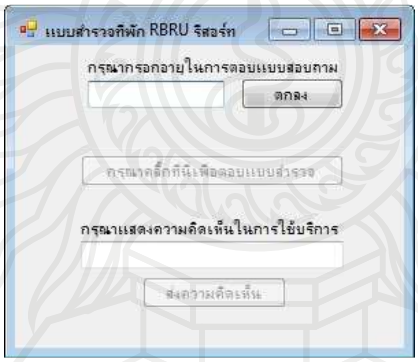
```
31                  this.Close();
32              }
33          }
34      }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.2

บรรทัดที่ 20 - 28 เป็นส่วนของการตรวจสอบเงื่อนไขในการตัดเกรดของโปรแกรมโดยใช้คำสั่ง if...else

ตัวอย่างที่ 5.3 จงเขียนโปรแกรมแบบสำรวจที่พัก RBRU รีสอร์ท โดยมีหน้าจอดังภาพที่ 5.6 และมีเงื่อนไขดังนี้

ผู้ตอบแบบสอบถามจะต้องมีอายุตั้งแต่ 20 – 50 ปี
ถ้าอายุต่ำกว่า 20 ปี ให้แจ้งว่า คุณยังเด็กเกินไป
ถ้าอายุอยู่ระหว่าง 20 – 50 ปี ให้แจ้งว่า คุณสามารถตอบแบบสอบถามได้
ถ้าอายุเกิน 50 ปี ให้แจ้งว่า คุณแก่เกินไป



ภาพที่ 5.6 หน้าจอโปรแกรมแบบสำรวจที่พัก RBRU รีสอร์ท

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmQuestion
- Text กำหนดเป็น “แบบสำรวจที่พัก RBRU รีสอร์ท”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblAge
- Text กำหนดเป็น “กรุณากรอกอายุในการตอบแบบสอบถาม”

กำหนด Properties สำหรับ label2
- Name กำหนดเป็น lblComment
- Text กำหนดเป็น “กรุณาแสดงความคิดเห็นในการใช้บริการ”

กำหนด Properties สำหรับ textBox1

Name กำหนดเป็น txtAge
Text ใส่เป็นค่าว่าง

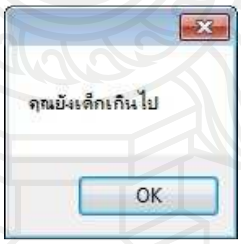
กำหนด Properties สำหรับ textBox2
Name กำหนดเป็น txtComment
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
Name กำหนดเป็น btnOk
Text กำหนดเป็น “ตกลง”

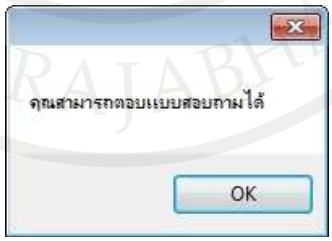
กำหนด Properties สำหรับ button2
Name กำหนดเป็น btnQuestion
Text กำหนดเป็น “กรุณาคลิ๊กที่นี่เพื่อตอบแบบสำรวจ”
Enabled กำหนดเป็น False

กำหนด Properties สำหรับ button3
Name กำหนดเป็น btnSend
Text กำหนดเป็น “ส่งความคิดเห็น”
Enabled กำหนดเป็น False

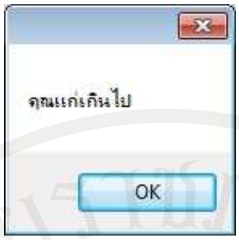
การแสดงผลเมื่อตรวจสอบค่าอายุให้แสดงผลดังภาพที่ 5.7 – 5.9



ภาพที่ 5.7 แสดง MessageBox เมื่อใส่อายุน้อยกว่า 20 ปี



ภาพที่ 5.8 แสดง MessageBox เมื่อใส่อายุระหว่าง 20 – 50 ปี

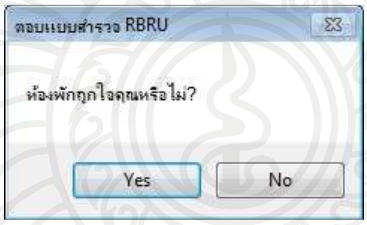


ภาพที่ 5.9 แสดง MessageBox เมื่อใส่อายุมากกว่า 50 ปี

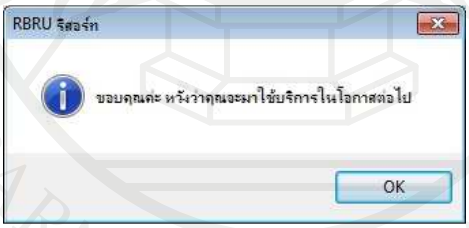
ถ้าอายุอยู่ในช่วง 20 – 50 ปี ให้กำหนด Properties สำหรับ btnQuestion และ btnSend ใหม่ดังนี้

Enabled กำหนดเป็น True

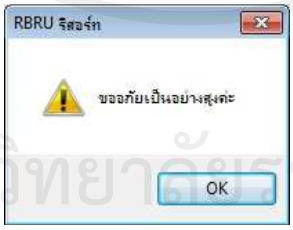
เมื่อคลิ๊กที่ปุ่ม btnQuestion โปรแกรมจะแสดง MessageBox ดังภาพที่ 5.10 เมื่อคลิ๊กปุ่ม Yes ให้แสดง MessageBox ดังภาพที่ 5.11 และเมื่อคลิ๊กปุ่ม No ให้แสดง MessageBox ดังภาพที่ 5.12



ภาพที่ 5.10 แสดง MessageBox เมื่อคลิ๊กที่ปุ่ม btnQuestion

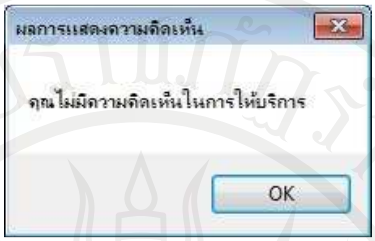


ภาพที่ 5.11 แสดง MessageBox เมื่อคลิ๊กที่ปุ่ม Yes

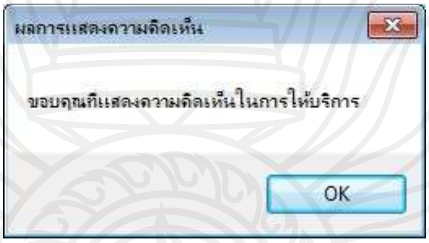


ภาพที่ 5.12 แสดง MessageBox เมื่อคลิ๊กที่ปุ่ม No

ส่วนการส่งความคิดเห็นถ้าในกรณีที่ไม่มีการใส่ความคิดเห็นให้แสดง MessageBox ดังภาพที่ 5.13 และถ้ามีการใส่ความคิดเห็นให้แสดง MessageBox ดังภาพที่ 5.14



ภาพที่ 5.13 แสดง MessageBox เมื่อไม่มีการแสดงความคิดเห็น



ภาพที่ 5.14 แสดง MessageBox เมื่อมีการแสดงความคิดเห็น

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex5_3
10      {
11          public partial class frmQuestion : Form
12          {
13              public frmQuestion()
14              {
15                  InitializeComponent();
16              }
```

```
17          private void btnOk_Click(object sender, EventArgs e)
18          {
19              int age = Convert.ToInt16(txtAge.Text);
20              if (age < 20)
21              {
22                  MessageBox.Show("คุณยังเด็กเกินไป");
23                  btnQuestion.Enabled = false;
24                  btnSend.Enabled = false;
25              }
26              else if (age <= 50)
27              {
28                  MessageBox.Show("คุณสามารถตอบแบบสอบถามได้");
29                  btnQuestion.Enabled = true;
30                  btnSend.Enabled = true;
31              }
32              else
33              {
34                  MessageBox.Show("คุณแก่เกินไป");
35                  btnQuestion.Enabled = false;
36                  btnSend.Enabled = false;
37              }
38          }
39          private void btnQuestion_Click(object sender, EventArgs e)
40          {
41              DialogResult re;
42              re = MessageBox.Show("ห้องพักถูกใจคุณหรือไม่?", "ตอบแบบสำรวจ RBRU",
    MessageBoxButtons.YesNo);
43              if (re == DialogResult.Yes)
44                  MessageBox.Show("ขอบคุณค่ะ หวังว่าคุณจะมาใช้บริการในโอกาสต่อไป",
    "RBRU รีสอร์ท", MessageBoxButtons.OK, MessageBoxIcon.Information);
45              else
46                  MessageBox.Show("ขออภัยเป็นอย่างสูงค่ะ", "RBRU รีสอร์ท",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
47          }
48          private void btnSend_Click(object sender, EventArgs e)
49          {
```

```
50              if (txtComment.Text == "")
51                  MessageBox.Show("คุณไม่มีความคิดเห็นในการให้บริการ", "ผลการแสดง
    ความคิดเห็น");
52              else
53                  MessageBox.Show("ขอบคุณที่แสดงความคิดเห็นในการให้บริการ", "ผลการ
    แสดงความคิดเห็น");
54          }
55      }
56  }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.3

บรรทัดที่ 20 - 37 เป็นส่วนของการตรวจสอบเงื่อนไขเรื่องอายุของผู้ตอบแบบสอบถามโดยใช้คำสั่ง if...else ถ้าตรงตามเงื่อนไขจะปรับค่าของปุ่ม btnSend ในบรรทัดที่ 30 ให้ใช้งานได้

บรรทัดที่ 41 เป็นส่วนของกำหนดตัวแปรสำหรับรับค่าที่ได้จากการกดปุ่มของ MessageBox

บรรทัดที่ 42 – 46 เป็นส่วนของการตรวจสอบเงื่อนไขจากการกดปุ่มของ MessageBox

บรรทัดที่ 50 – 53 เป็นส่วนของการตรวจสอบเงื่อนไขในการแสดงความคิดเห็น

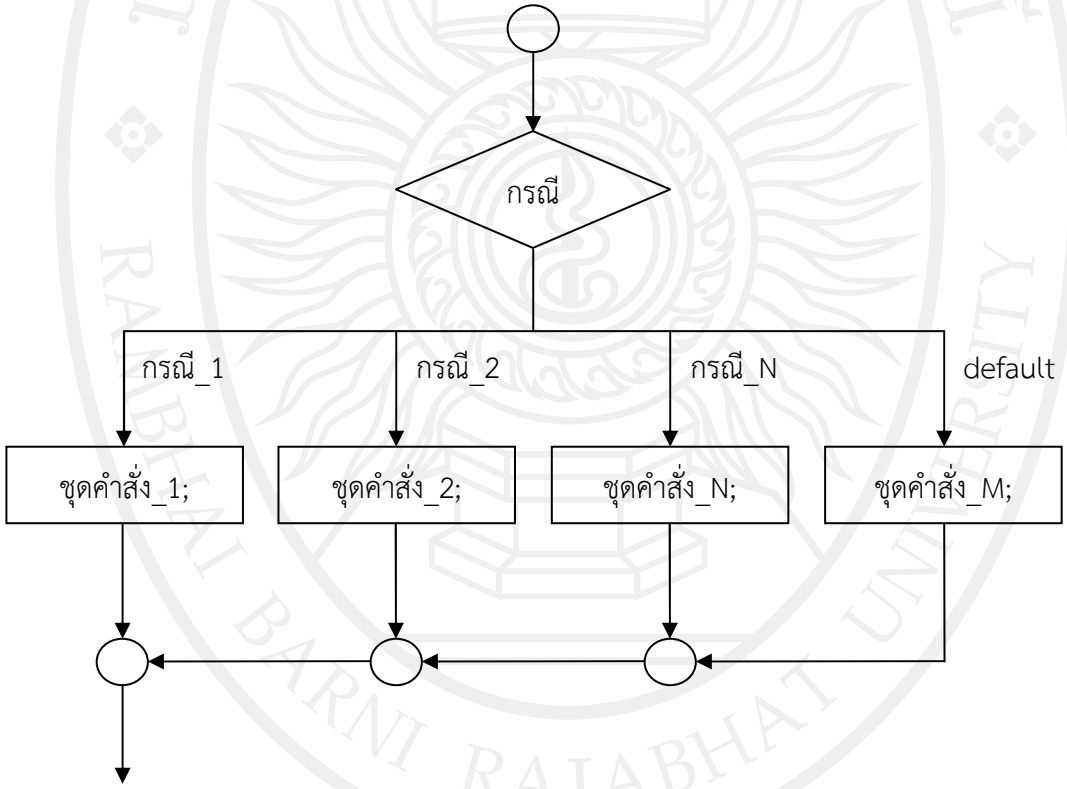
5.1.2 คำสั่ง switch

เป็นคำสั่งที่ใช้ตรวจสอบค่าที่เป็นกรณีที่เป็นเงื่อนไข ซึ่งผลที่ได้จากการตรวจสอบจะถูกนำไปเปรียบเทียบกับค่ากรณีต่าง ๆ ที่ถูกกำหนดไว้หลัง case โดยโปรแกรมจะทำงานตามคำสั่งที่หลังเครื่องหมายโคลอน (:) ทีละคำสั่งไปเรื่อย ๆ จนกว่าจะเจอคำสั่ง break และเมื่อเจอคำสั่ง break แล้วโปรแกรมจะออกจากบล็อกของคำสั่ง switch ทันที แต่ในกรณีที่การตรวจสอบเงื่อนไขไม่ตรงกับเงื่อนไขที่อยู่หลัง case เลย โปรแกรมจะมาทำงานที่คำสั่งหลัง default และทำไปเรื่อย ๆ จนกว่าจะเจอคำสั่ง break เช่นเดียวกัน (Microsoft, 2016)

```
switch ( กรณี )
{
  case กรณี_1 :
       <ชุดคำสั่ง_1>;
       break;
  case กรณี_2 :
       <ชุดคำสั่ง_2>;
       break;
       ...
       ...
       ...
```

```
        case กรณี_N :
            <ชุดคำสั่ง_N>;
            break;
        default :
            <ชุดคำสั่ง_M>;
            break;
}
```

การใช้คำสั่ง switch จะมีข้อดีคือผู้เขียนโปรแกรมไม่ต้องกำหนดทางเลือกที่เป็นจริงและเท็จ ส่วนข้อเสียคือถ้ากำหนดกรณีหลังคำสั่ง case ไม่ดี โปรแกรมอาจจะเลือกการทำงานผิดพลาดได้ ซึ่งสามารถเขียนเป็นแผนผังการทำงานได้ดังภาพที่ 5.15



ภาพที่ 5.15 แผนผังการทำงานของคำสั่ง switch

จากภาพที่ 5.15 การทำงานของ switch จะตรวจสอบกรณีที่เป็นเงื่อนไข เพื่อเลือกกรณีที่ถูกต้องเพื่อทำงาน ถ้าไม่มีกรณีใดถูกต้องตามเงื่อนไข โปรแกรมจะเข้าไปทำงานใน default แทน

ตัวอย่างที่ 5.4 การใช้คำสั่ง switch โดยแปลงจากตัวอย่างที่ 5.1 มีชุดคำสั่งดังนี้

```
1           private void btnOk_Click(object sender, EventArgs e)
2           {
3               switch(txtDay.Text)
4               {
5                   case "อาทิตย์":
6                       this.BackColor = Color.Red;
7                       MessageBox.Show("วันนี้เป็นวันอาทิตย์ สีแดง");
8                       break;
9                   case "จันทร์" :
10                      this.BackColor=Color.Yellow;
11                      MessageBox.Show("วันนี้เป็นวันจันทร์ สีเหลือง");
12                      break;
13                  case "อังคาร":
14                      this.BackColor = Color.Pink;
15                      MessageBox.Show("วันนี้เป็นวันอังคาร สีชมพู");
16                      break;
17                  case "พุธ":
18                      this.BackColor = Color.Green;
19                      MessageBox.Show("วันนี้เป็นวันพุธ สีเขียว");
20                      break;
21                  case "พฤหัส":
22                      this.BackColor = Color.Orange;
23                      MessageBox.Show("วันนี้เป็นวันพฤหัส สีส้ม");
24                      break;
25                  case "ศุกร์":
26                      this.BackColor = Color.Blue;
27                      MessageBox.Show("วันนี้เป็นวันศุกร์ สีฟ้า");
28                      break;
29                  case "เสาร์":
30                      this.BackColor = Color.Magenta;
31                      MessageBox.Show("วันนี้เป็นวันเสาร์ สีม่วง");
32                      break;
33                  default:
34                      this.BackColor = Control.DefaultBackColor;
```

```
35                      MessageBox.Show("ไม่มีวันนี้อยู่ในสัปดาห์");
36                      break;
37                  }
38              }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.4

บรรทัดที่ 3 เป็นส่วนของการตรวจสอบเงื่อนไขโดยใช้คำสั่ง switch

บรรทัดที่ 5 – 32 เป็นส่วนของกรณีที่เป็นผลจากการตรวจสอบเงื่อนไขโดยคำสั่ง switch ซึ่งจะเลือกการทำงานตามผลลัพธ์ที่ได้

บรรทัดที่ 33 เป็นส่วนที่จะทำงานเมื่อไม่มีเงื่อนไขใดถูกต้องเลย

5.2 การทำงานซ้ำ

ในการเขียนโปรแกรมจะต้องมีการทำงานบางส่วนที่ต้องทำงานซ้ำจนกว่าจะครบตามที่กำหนด เช่น การบวกเลขตั้งแต่ 1 ถึง 10 เป็นการใช้คำสั่งบวกซ้ำ ๆ โดยเพิ่มตัวเลขที่ใช้ในการบวก ซึ่งต้องเขียนคำสั่งมากมาย และเมื่อต้องการแก้ไขก็ทำให้เกิดความยุ่งยาก แต่ถ้าใช้คำสั่งที่ควบคุมการวนซ้ำจะทำให้การเขียนโปรแกรมทำได้ง่ายขึ้น

5.2.1 คำสั่ง for

คำสั่ง for เป็นคำสั่งวนรอบที่สามารถกำหนดค่าเริ่มต้น เงื่อนไข และการเพิ่มค่า โดยการวนรอบจะวนโดยอัตโนมัติจนกว่าเงื่อนไขจะเป็นเท็จ มีรูปแบบดังนี้ (Microsoft, 2016)

for (<ค่าตัวแปรเริ่มต้น>; <เงื่อนไขที่กำหนดการหยุดวนซ้ำ>; <การเพิ่มหรือลดค่าของตัวแปร>)

เช่น

```
for (i = 1; i < 10; i++)
{
   <ชุดคำสั่ง_1>;
}
```

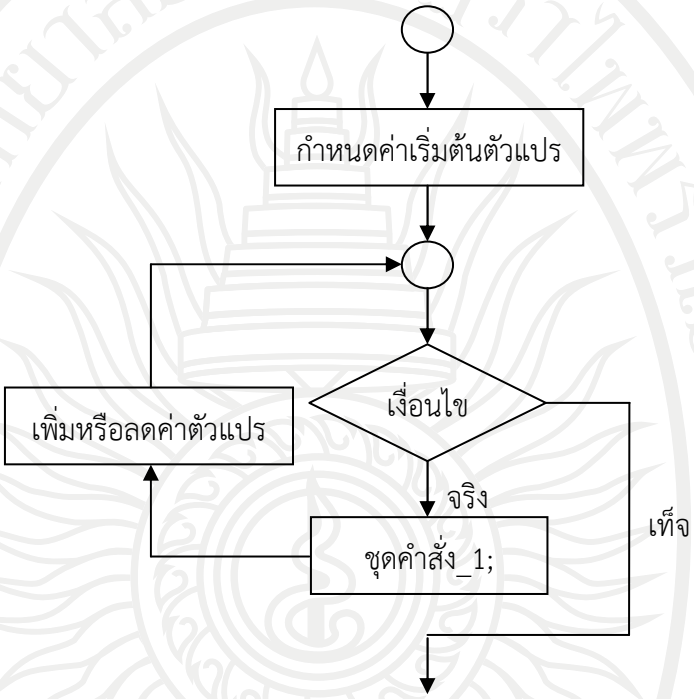
หรือ

```
for (int i = 10; i > 1; i--)
{
   <ชุดคำสั่ง_1>;
}
```

หรือ ในกรณีที่มีคำสั่งที่ต้องการวนซ้ำเพียงคำสั่งเดียว

```
for (int i = 1; i < 10; i++)
  คำสั่ง_1;
```

สามารถเขียนเป็นแผนผังการทำงานของคำสั่ง for ได้ดังภาพที่ 5.16



ภาพที่ 5.16 แผนผังการทำงานของคำสั่ง for

ในส่วนของ <การเพิ่มหรือลดค่าของตัวแปร> รูปแบบที่ใช้จะอยู่ในรูปแบบของพรีฟิกซ์ (prefix) และโพสต์ฟิกซ์ (postfix) ซึ่งเป็นการทำให้ค่าเพิ่มหรือลดลง

5.2.1.1 พรีฟิกซ์มีรูปแบบดังนี้

การเพิ่มค่า

```
++[ตัวแปร];      //มีความหมายเท่ากับ [ตัวแปร] = [ตัวแปร] + 1
```

ตัวอย่างเช่น

```
int i = 1;
++i;
Console.Write( i ) ;
```

ผลลัพธ์

```
2
```

การลดค่า

```
--[ตัวแปร];        //มีความหมายเท่ากับ [ตัวแปร] = [ตัวแปร] – 1
```

ตัวอย่างเช่น

```
int i = 1;
--i;
Console.Write( i ) ;
```

ผลลัพธ์

```
0
```

5.2.1.2 โพสต์ฟิกซ์มีรูปแบบดังนี้

การเพิ่มค่า

```
[ตัวแปร]++;       //มีความหมายเท่ากับ [ตัวแปร] = [ตัวแปร] + 1
```

ตัวอย่างเช่น

```
int i = 1;
i++;
Console.Write( i ) ;
```

ผลลัพธ์

```
2
```

การลดค่า

```
[ตัวแปร]--;       //มีความหมายเท่ากับ [ตัวแปร] = [ตัวแปร] – 1
```

ตัวอย่างเช่น

```
int i = 1;
i--;
Console.Write( i ) ;
```

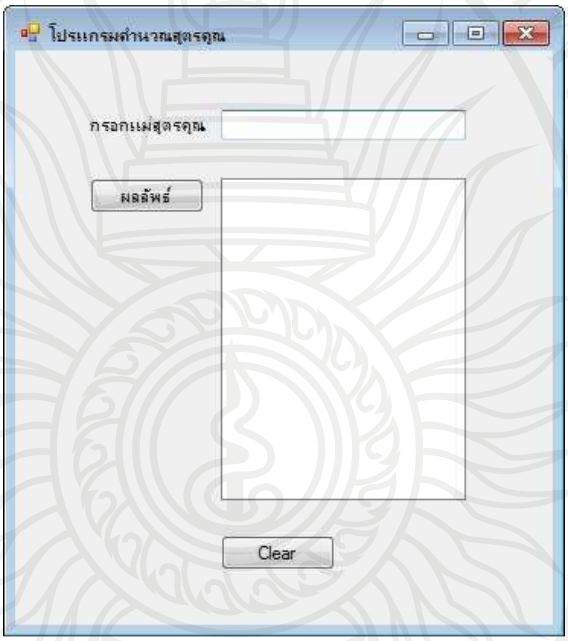
ผลลัพธ์

```
0
```

จากตัวอย่างจะเห็นว่าทั้งพรีฟิกซ์และโพสต์ฟิกซ์จะเป็นการเพิ่มค่าเหมือนกัน แต่ความแตกต่างคือ พรีฟิกซ์จะดำเนินการเพิ่มหรือลดค่าก่อน แล้วเก็บไว้ในตัวแปร ส่วนโพสต์ฟิกซ์จะเก็บค่าไว้ในตัวแปรก่อน แล้วค่อยเพิ่มหรือลดค่า

ตัวอย่างที่ 5.5 จงเขียนโปรแกรมคำนวณสูตรคูณ โดยใช้คำสั่ง for และมีเงื่อนไขดังนี้

ให้กรอกเลขแม่สูตรคูณลงใน textBox และเมื่อกดแสดงผลลัพธ์ให้แสดงสูตรคูณลงใน listBox โดยมีหน้าจอดังภาพที่ 5.17 และผลลัพธ์ตามภาพที่ 5.18



ภาพที่ 5.17 หน้าจอโปรแกรมคำนวณสูตรคูณ

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmMulti
- Text กำหนดเป็น “โปรแกรมคำนวณสูตรคูณ”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblMulti
- Text กำหนดเป็น “กรอกแม่สูตรคูณ”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtMulti
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ listBox1
- Name กำหนดเป็น lstResult
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1

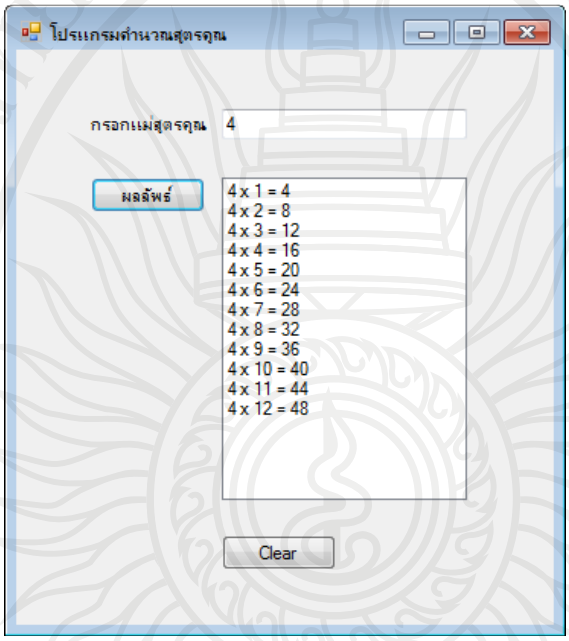
Name กำหนดเป็น btnResult

Text กำหนดเป็น “ผลลัพธ์”

กำหนด Properties สำหรับ button2

Name กำหนดเป็น btnClear

Text กำหนดเป็น “Clear”



ภาพที่ 5.18 ผลลัพธ์ของโปรแกรมคำนวณสูตรคูณ

ชุดคำสั่งของโปรแกรมเป็นดังนี้

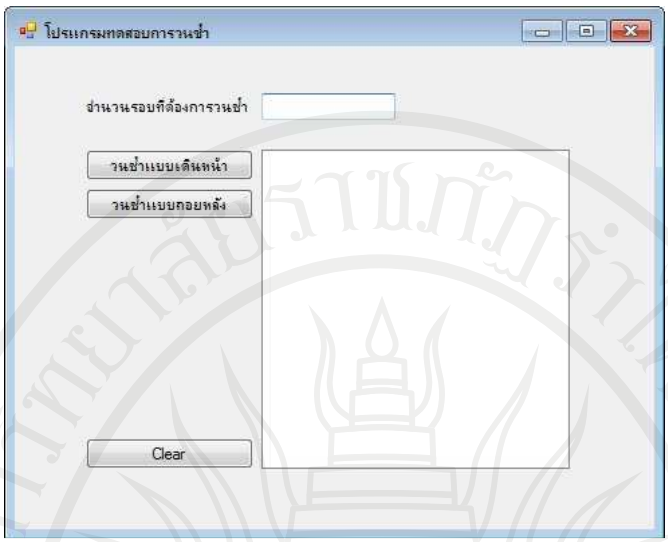
```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex5_5
10  {
11      public partial class frmMulti : Form
12      {
```

```
13          public frmMulti()
14          {
15              InitializeComponent();
16          }
17          private void btnResult_Click(object sender, EventArgs e)
18          {
19              int num;
20              num = Convert.ToInt32(textBox1.Text);
21              for (int i = 1; i <= 12; i++)
22              {
23                  lstResult.Items.Add(num + " x " + i + " = " + (num * i));
24              }
25          }
26          private void btnClear_Click(object sender, EventArgs e)
27          {
28              txtMulti.Text = "";
29              lstResult.Items.Clear();
30          }
31      }
32  }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.5

จากตัวอย่างโปรแกรมนี้จะใช้คำสั่ง for ในบรรทัดที่ 21 เพื่อวนรอบตั้งแต่ 1 ถึง 12 และมีการใช้ listBox มาแสดงผลลัพธ์ โดยในบรรทัดที่ 23 เป็นการเพิ่มข้อมูลใน listBox จะใช้เมธอด Add() ของ Items เพิ่มข้อมูลเข้าไป และในบรรทัดที่ 29 ใช้เมธอด Clear() ของ Items เพื่อเคลียร์ค่าทั้งหมดที่อยู่ใน listBox

ตัวอย่างที่ 5.6 จงเขียนโปรแกรมทดสอบการวนซ้ำ โดยใช้คำสั่ง for และมีหน้าจอภาพที่ 5.19 การทำงานของโปรแกรม คือ ให้รับค่าที่เป็นตัวเลขและเมื่อกดปุ่มวนซ้ำแบบเดินหน้าหรือถอยหลัง โปรแกรมจะแสดงผลลัพธ์ใน listBox ดังภาพที่ 5.20 และ 5.21



ภาพที่ 5.19 หน้าจอของโปรแกรมทดสอบการวนซ้ำ

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmLoop
- Text กำหนดเป็น “โปรแกรมทดสอบการวนซ้ำ”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblLoop
- Text กำหนดเป็น “จำนวนรอบที่ต้องการวนซ้ำ”

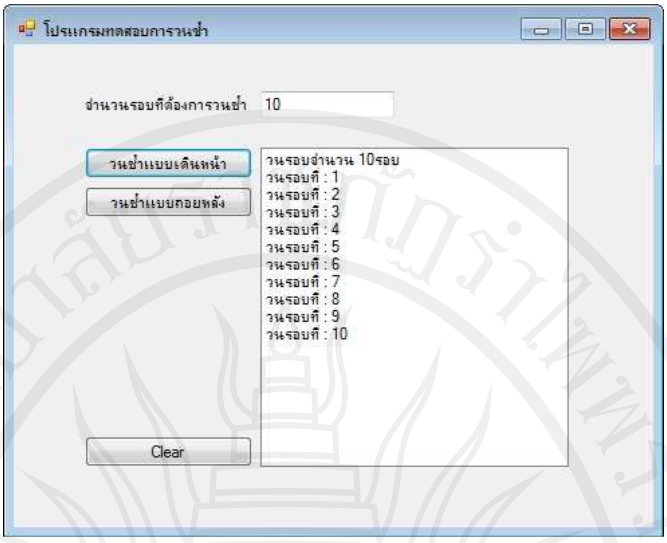
กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtLoop
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ listBox1
- Name กำหนดเป็น lstResult
- Text ใส่เป็นค่าว่าง

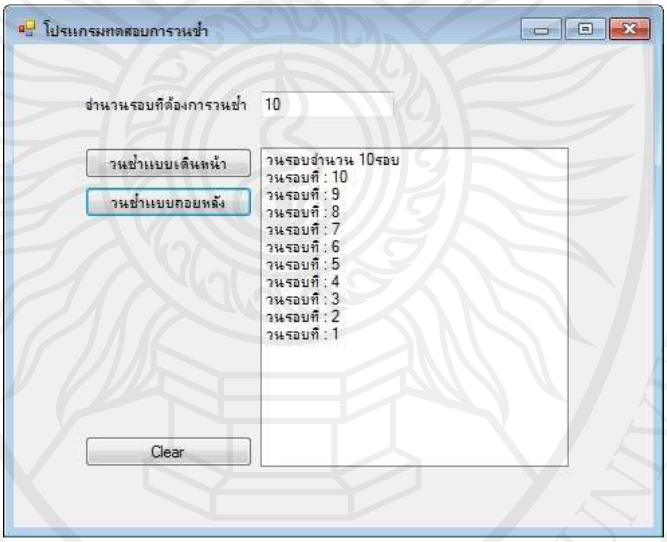
กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnForward
- Text กำหนดเป็น “วนซ้ำแบบเดินหน้า”

กำหนด Properties สำหรับ button2
- Name กำหนดเป็น btnBackward
- Text กำหนดเป็น “วนซ้ำแบบถอยหลัง”

กำหนด Properties สำหรับ button3
- Name กำหนดเป็น btnClear
- Text กำหนดเป็น “Clear”



ภาพที่ 5.20 ผลลัพธ์เมื่อกดปุ่มวนซ้ำแบบเดินหน้า



ภาพที่ 5.21 ผลลัพธ์เมื่อกดปุ่มวนซ้ำแบบถอยหลัง

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1    using System;
2    using System.Collections.Generic;
3    using System.ComponentModel;
4    using System.Data;
5    using System.Drawing;
6    using System.Linq;
7    using System.Text;
```

```
8       using System.Windows.Forms;
9       namespace Ex5_6
10      {
11          public partial class frmLoop : Form
12          {
13              public frmLoop()
14              {
15                  InitializeComponent();
16              }
17              private void btnForward_Click(object sender, EventArgs e)
18              {
19                  int num;
20                  num = Convert.ToInt32(txtLoop.Text);
21                  lstResult.Items.Clear();
22                  lstResult.Items.Add("วนรอบจำนวน " + num + "รอบ");
23                  for (int i = 1; i <= num; i++)
24                  {
25                      lstResult.Items.Add("วนรอบที่ : " + i);
26                  }
27              }
28              private void btnBackward_Click(object sender, EventArgs e)
29              {
30                  int num;
31                  num = Convert.ToInt32(txtLoop.Text);
32                  lstResult.Items.Clear();
33                  lstResult.Items.Add("วนรอบจำนวน " + num + "รอบ");
34                  for (int i = num; i >= 1; i--)
35                  {
36                      lstResult.Items.Add("วนรอบที่ : " + i);
37                  }
38              }
39              private void btnClear_Click(object sender, EventArgs e)
40              {
41                  txtLoop.Text = "";
42                  lstResult.Items.Clear();
43              }
```

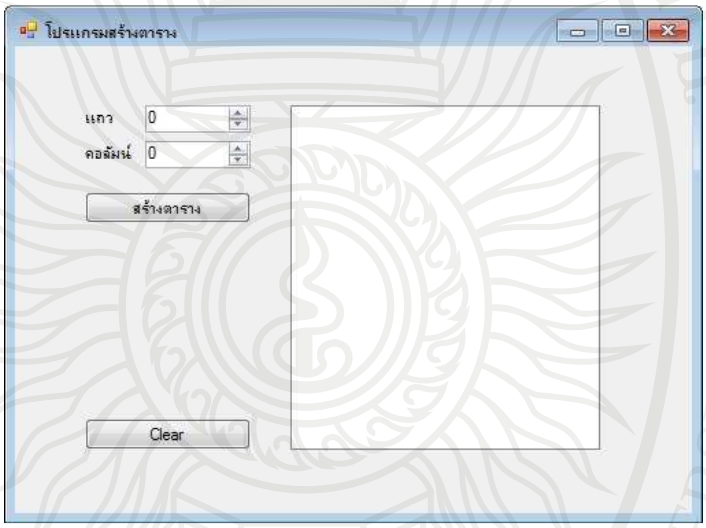
```
44          }
45      }
```

อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.6

บรรทัดที่ 17- 27 เป็นส่วนของการวนรอบแบบเดินหน้าโดยใช้การเพิ่มค่าลงใน listBox

บรรทัดที่ 18- 28 เป็นส่วนของการวนรอบแบบถอยหลังโดยใช้การเพิ่มค่าลงใน listBox

ตัวอย่างที่ 5.7 จงเขียนโปรแกรมสร้างตาราง โดยใช้คำสั่ง for และมีหน้าจอดังภาพที่ 5.22 โดยโปรแกรมจะรับค่าแถวและคอลัมน์จาก numericUpDown หลังจากนั้นเมื่อกดปุ่มสร้างตาราง โปรแกรมจะสร้างตารางใน listBox ดังภาพที่ 5.23



ภาพที่ 5.22 หน้าจอโปรแกรมสร้างตาราง

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmTable

Text กำหนดเป็น “โปรแกรมสร้างตาราง”

กำหนด Properties สำหรับ label1

Name กำหนดเป็น lblRow

Text กำหนดเป็น “แถว”

กำหนด Properties สำหรับ label2

Name กำหนดเป็น lblCol

Text กำหนดเป็น “คอลัมน์”

กำหนด Properties สำหรับ numericUpDown1

Name กำหนดเป็น numRow

กำหนด Properties สำหรับ numericUpDown2

Name กำหนดเป็น numCol

กำหนด Properties สำหรับ listBox1

Name กำหนดเป็น lstResult

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1

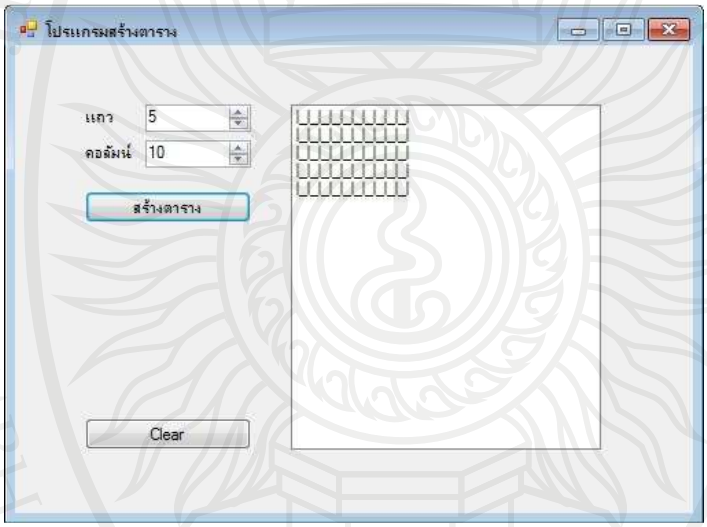
Name กำหนดเป็น btnTable

Text กำหนดเป็น “สร้างตาราง”

กำหนด Properties สำหรับ button2

Name กำหนดเป็น btnClear

Text กำหนดเป็น “Clear”



ภาพที่ 5.23 ผลลัพธ์เมื่อกดสร้างตาราง

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex5_7
```

```
10      {
11          public partial class frmTable : Form
12          {
13              public frmTable()
14              {
15                  InitializeComponent();
16              }
17              private void btnTable_Click(object sender, EventArgs e)
18              {
19                  int row;
20                  int col;
21                  string str = "|";
22                  row = Convert.ToInt32(numRow.Value);
23                  col = Convert.ToInt32(numCol.Value);
24                  lstResult.Items.Clear();
25                  for (int i = 1; i <= row; i++)
26                  {
27                      for (int j = 1; j <= col; j++)
28                      {
29                          str = str + " _|";
30                      }
31                      lstResult.Items.Add(str);
32                      str = "|";
33                  }
34              }
35              private void btnClear_Click(object sender, EventArgs e)
36              {
37                  numRow.Value = 0;
38                  numCol.Value = 0;
39                  lstResult.Items.Clear();
40              }
41          }
42      }
```


อธิบายชุดคำสั่งของโปรแกรมในตัวอย่างที่ 5.6

บรรทัดที่ 22 และ 23 เป็นส่วนของการรับค่าจำนวนแถวและคอลัมน์ของตาราง

บรรทัดที่ 25 – 33 เป็นส่วนของการสร้างตารางโดยใช้คำสั่ง for

5.2.2 คำสั่ง while

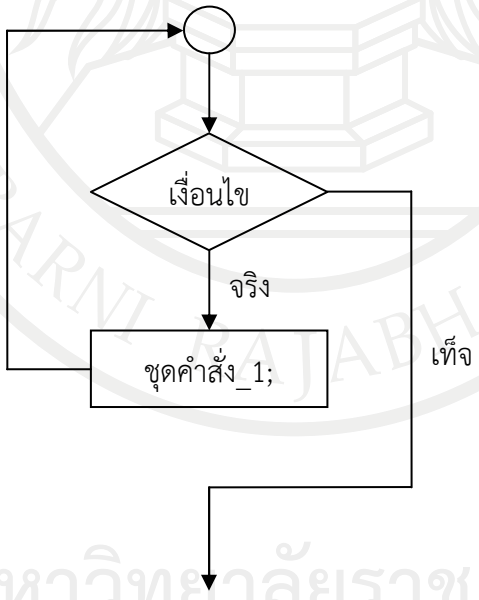
เป็นคำสั่งที่ใช้ในการวนรอบอีกรูปแบบหนึ่ง ซึ่งมีการตรวจสอบเงื่อนไขโดยไม่มีการกำหนดค่าเริ่มต้นให้ตัวแปร โดยการทำงานของคำสั่ง while จะตรวจสอบเงื่อนไขก่อน ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำงานภายใต้คำสั่ง while เมื่อทำงานเสร็จแล้วจะย้อนกลับมาตรวจสอบเงื่อนไขที่คำสั่ง while เพื่อทำงานเป็นวนรอบต่อไป (Microsoft, 2016)

```
while (เงื่อนไข)
  คำสั่ง_1;
```

หรือ

```
while (เงื่อนไข)
{
  คำสั่ง_1;
  คำสั่ง_2;
  ...
  คำสั่ง_N;
}
```

สามารถเขียนเป็นแผนผังการทำงานของคำสั่ง while ได้ดังภาพที่ 5.24



ภาพที่ 5.24 แผนผังการทำงานของคำสั่ง while

สำหรับข้อควรระวังคำสั่ง while คือ ถ้าเงื่อนไขเป็นจริงตลอด โปรแกรมจะไม่สามารถออกจากการทำงานของลูปได้เลย

จากตัวอย่างที่ 5.5 สามารถใช้คำสั่ง while แทนคำสั่ง for ในการทำงานได้ดังนี้

```
1           private void btnResult_Click(object sender, EventArgs e)
2           {
3               int num;
4               int i  = 1;
5               num = Convert.ToInt32(textBox1.Text);
6               while (i<= 12)
7               {
8                   lstResult.Items.Add(num + " x " + i + " = " + (num * i));
9                   i++;
10              }
11          }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 6 – 10 เป็นการวนรอบโดยใช้คำสั่ง while แทนคำสั่ง for ในตัวอย่างที่ 5.5

และจากตัวอย่างที่ 5.6 สามารถใช้คำสั่ง while แทนคำสั่ง for ในการทำงานได้ดังนี้

```
1           private void btnForward_Click(object sender, EventArgs e)
2           {
3               int num;
4               int i;
5               num = Convert.ToInt32(txtLoop.Text);
6               i = 1;
7               lstResult.Items.Clear();
8               lstResult.Items.Add("วนรอบจำนวน " + num + "รอบ");
9               while (i <= num)
10              {
11                  lstResult.Items.Add("วนรอบที่ : " + i);
12                  i++;
13              }
14          }
15
```

```
16          private void btnBackward_Click(object sender, EventArgs e)
17          {
18              int num;
19              int i;
20              num = Convert.ToInt32(txtLoop.Text);
21              i = num;
22              lstResult.Items.Clear();
23              lstResult.Items.Add("วนรอบจำนวน " + num + "รอบ");
24              while (i >= num)
25              {
26                  lstResult.Items.Add("วนรอบที่ : " + i);
27                  i--;
28              }
29          }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 – 13 และบรรทัดที่ 24 – 28 เป็นการวนรอบโดยใช้คำสั่ง while แทนคำสั่ง for ในตัวอย่างที่ 5.6

และจากตัวอย่างที่ 5.7 สามารถใช้คำสั่ง while แทนคำสั่ง for ในการทำงานได้ดังนี้

```
1           private void btnTable_Click(object sender, EventArgs e)
2           {
3               int row;
4               int col;
5               string str = "|";
6               int i, j;
7               row = Convert.ToInt32(numRow.Value);
8               col = Convert.ToInt32(numCol.Value);
9               lstResult.Items.Clear();
10              i = 1;
11              j = 1;
12              while (i <= row)
13              {
14                  while (j <= col)
15                  {
```

```
16                      str = str + "_|";
17                      j++;
18                  }
19                  lstResult.Items.Add(str);
20                  str = "|";
21                  i++;
22              }
23          }
```

อธิบายชุดคำสั่งของโปรแกรม

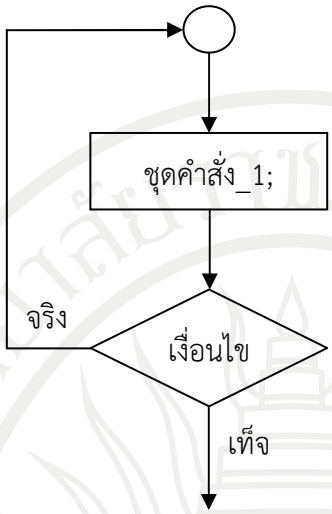
บรรทัดที่ 12 – 23 เป็นการวนรอบโดยใช้คำสั่ง while แทนคำสั่ง for ในตัวอย่างที่ 5.7

5.2.3 คำสั่ง do...while

เป็นคำสั่งในการทำงานแบบวนรอบที่คล้ายกับคำสั่ง while แต่มีจุดเด่นที่แตกต่างคือ โปรแกรมจะทำงานก่อนแล้วค่อยตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงโปรแกรมจะกลับไปที่คำสั่ง do แต่ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะออกจากการทำงานแบบวนรอบ แต่ทั้งคำสั่ง while และ do...while มีข้อควรระวังเหมือนกันคือ ต้องตรวจสอบเงื่อนไขให้ดีเสียก่อน เพราะถ้าเงื่อนไขเป็นจริงตลอดโปรแกรมจะไม่สามารถออกจากการทำงานในลูปได้เลย การเขียนคำสั่ง do...while มีรูปแบบดังนี้ (Microsoft, 2016)

```
do
{
  คำสั่ง_1;
  คำสั่ง_2;
  ...
  คำสั่ง_N;
} while (เงื่อนไข);
```

สามารถเขียนเป็นแผนผังการทำงานของคำสั่ง do...while ได้ดังภาพที่ 5.25



ภาพที่ 5.25 แผนผังการทำงานของคำสั่ง do...while

ข้อแตกต่างระหว่างคำสั่ง while และ do...while คือ คำสั่ง while จะตรวจสอบเงื่อนไขก่อนจึงจะเข้าไปทำงานในลูปส่วนคำสั่ง do...while จะทำงานในลูปก่อนและตรวจสอบเงื่อนไขทีหลัง ซึ่งการจะเลือกใช้คำสั่ง while หรือ do...while จะขึ้นอยู่กับการทำงานของโปรแกรมที่พัฒนา

จากตัวอย่างที่ 5.5 สามารถใช้คำสั่ง do...while แทนคำสั่ง for ในการทำงานได้ดังนี้

```
1          private void btnResult_Click(object sender, EventArgs e)
2          {
3              int num;
4              int i  = 1;
5              num = Convert.ToInt32(textBox1.Text);
6              do
7              {
8                  lstResult.Items.Add(num + " x " + i + " = " + (num * i));
9                  i++;
10             } while (i<= 12);
11         }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 6 – 10 เป็นการวนรอบโดยใช้คำสั่ง do...while แทนคำสั่ง for ในตัวอย่างที่ 5.5

และจากตัวอย่างที่ 5.6 สามารถใช้คำสั่ง do...while แทนคำสั่ง for ในการทำงานได้ดังนี้

```
1           private void btnForward_Click(object sender, EventArgs e)
2           {
3               int num;
4               int i;
5               num = Convert.ToInt32(txtLoop.Text);
6               i = 1;
7               lstResult.Items.Clear();
8               lstResult.Items.Add("วนรอบจำนวน " + num + "รอบ");
9               do
10              {
11                  lstResult.Items.Add("วนรอบที่ : " + i);
12                  i++;
13              } while (i <= num);
14          }
15          private void btnBackward_Click(object sender, EventArgs e)
16          {
17              int num;
18              int i;
19              num = Convert.ToInt32(txtLoop.Text);
20              i = num;
21              lstResult.Items.Clear();
22              lstResult.Items.Add("วนรอบจำนวน " + num + "รอบ");
23              do
24              {
25                  lstResult.Items.Add("วนรอบที่ : " + i);
26                  i--;
27              } while (i >= num);
28          }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 – 13 และบรรทัดที่ 23 – 27 เป็นการวนรอบโดยใช้คำสั่ง do...while แทนคำสั่ง for ในตัวอย่างที่ 5.6

และจากตัวอย่างที่ 5.7 สามารถใช้คำสั่ง do...while แทนคำสั่ง for ในการทำงานได้ดังนี้

```
1           private void btnTable_Click(object sender, EventArgs e)
```

```
2           {
3               int row;
4               int col;
5               string str = "|";
6               int i, j;
7               row = Convert.ToInt32(numRow.Value);
8               col = Convert.ToInt32(numCol.Value);
9               lstResult.Items.Clear();
10              i = 1;
11              j = 1;
12              do
13              {
14                  do
15                  {
16                      str = str + "_|";
17                      j++;
18                  } while (j <= col);
19                  lstResult.Items.Add(str);
20                  str = "|";
21                  i++;
22              } while (i <= row);
23          }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 12 – 22 เป็นการวนรอบโดยใช้คำสั่ง do...while แทนคำสั่ง for ในตัวอย่างที่ 5.7

5.3 สรุป

การเขียนคำสั่งที่ใช้ในการควบคุมการทำงานของโปรแกรมสามารถแบ่งออกเป็น 2 แบบ คือ การตัดสินใจ ซึ่งเป็นคำสั่งที่ทำให้โปรแกรมสามารถทำงานในลักษณะของทางเลือกได้ อาจจะมีทางเลือกเดียว 2 ทางเลือก หรือหลายทางเลือกตามผลลัพธ์ที่ต้องการ คำสั่งที่ใช้ในการตัดสินใจ ได้แก่ คำสั่ง if...else ซึ่งเป็นการตรวจสอบเงื่อนไขให้เกิดเป็นทางเลือกโดยดูจากผลลัพธ์ของการตรวจสอบว่าเป็นจริงหรือเป็นเท็จ ถ้าเป็นจริงจะทำงานในชุดคำสั่งหลังคำสั่ง if แต่ถ้าผลลัพธ์เป็นเท็จจะทำงานในชุดคำสั่งหลังคำสั่ง else และ คำสั่ง switch ซึ่งเป็นคำสั่งทางเลือกในการทำงานที่จะตรวจสอบเงื่อนไขและนำผลลัพธ์มาเปรียบเทียบกับกรณีต่าง ๆ เพื่อใช้ในการทำงาน

ส่วนการทำงานซ้ำ (Iteration) จะเป็นการกำหนดให้โปรแกรมสามารถทำงานบางอย่างที่ใช้คำสั่งซ้ำ ๆ โดยอาจจะมีการเปลี่ยนค่าบางอย่างที่ใช้ในการทำงาน หรือทำงานให้เกิดผลลัพธ์บางอย่างที่ต้องการ คำสั่งที่ใช้ในการวนซ้ำ ได้แก่ คำสั่ง for เป็นคำสั่งที่กำหนดการวนรอบที่แน่นอน และสามารถกำหนดระดับของการวนรอบได้ คำสั่ง while และ คำสั่ง do...while เป็นคำสั่งที่ใช้วนรอบที่ไม่แน่นอนต้องอาศัยเงื่อนไขในการตรวจสอบ ถ้าเงื่อนไขเป็นจริงจะทำงานวนรอบไปเรื่อย ๆ จนกว่าจะมีการตรวจสอบว่าเงื่อนไขเป็นเท็จซึ่งมีข้อเสียคือ อาจจะทำให้เกิดการวนรอบไม่รู้จบ

แบบฝึกหัดบทที่ 5

1. การเขียนคำสั่งควบคุมการทำงานของโปรแกรมมีกี่แบบ อะไรบ้าง
2. คำสั่ง if...else ใช้สำหรับทำอะไร
3. ถ้าต้องการเขียนโปรแกรมที่มีหลายทางเลือก สามารถใช้คำสั่งอะไรได้บ้าง
4. จงเขียนรูปแบบของคำสั่ง switch
5. คำสั่งในการวนซ้ำที่มีจำนวนรอบที่แน่นอนคือคำสั่งใด มีรูปแบบอย่างไร
6. คำสั่ง while มีรูปแบบอย่างไร
7. คำสั่ง if...else และ switch ต่างกันอย่างไร
8. จงแปลงคำสั่ง if...else ต่อไปนี้ให้อยู่ในรูปแบบ switch

```
if (num < 10)
   MessageBox.Show("Num น้อยกว่า 10");
else if (num < 50)
   MessageBox.Show("Num มีค่าตั้งแต่ 10 ขึ้นไปแต่ไม่เกิน 50");
else
   MessageBox.Show("Num มีค่าตั้งแต่ 50 ขึ้นไป");
```

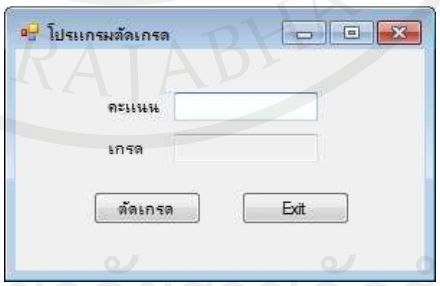
9. จงแปลงคำสั่ง for ต่อไปนี้ให้อยู่ในรูปแบบ do...while

```
for (int i = 1; i < 5; i++)
   MessageBox.Show("Hello");
```

10. จงเขียนโปรแกรมตัดเกรด โดยมีเงื่อนไขดังนี้

รับข้อมูลค่าคะแนนตั้งแต่ 0 - 100

คะแนน 0 – 50	ได้เกรด F
คะแนน 50 – 59	ได้เกรด D
คะแนน 60 – 69	ได้เกรด C
คะแนน 70 – 79	ได้เกรด B
คะแนน 80 – 100	ได้เกรด A



เอกสารอ้างอิง

นิรันดร์ ประวิทย์ธนา. (2545). **เก่ง C# ให้ครบสูตร**. กรุงเทพฯ : วิตตี้ กรุ๊ป.

ศุภชัย สมพานิช. (2556). **คู่มือเรียนและใช้งาน Visual C#**. กรุงเทพฯ: สวัสดี ไอที.

Microsoft. (2016). **do**. (online). Available : https://msdn.microsoft.com/en-us/library/370s1zax(v=vs.100).aspx. 6 February 2016.

_____. (2016). **for**. (online). Available : https://msdn.microsoft.com/en-us/library/ch45axte(v=vs.100).aspx. 6 February 2016.

_____. (2016). **switch**. (online). Available : https://msdn.microsoft.com/en-us/library/06tc147t(v=vs.100).aspx. 6 February 2016.

_____. (2016). **while**. (online). Available : https://msdn.microsoft.com/en-us/library/2aeyhxcd(v=vs.100).aspx. 6 February 2016.

แผนบริหารประจำบทที่ 6

เนื้อหา

บทที่ 6 โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชัน

6.1 โครงสร้างข้อมูลแบบแถวลำดับ

6.2 โครงสร้างข้อมูลแบบคอลเลคชัน

6.3 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. อธิบายโครงสร้างข้อมูลแบบแถวลำดับได้
2. อธิบายโครงสร้างข้อมูลแบบคอลเลคชันได้
3. อธิบายการใช้งานข้อมูลแบบอาร์เรย์ได้
4. อธิบายการใช้งานข้อมูลแบบสแตกและคิวได้
5. ประยุกต์ใช้อาร์เรย์มิติต่าง ๆ ในการเขียนโปรแกรมได้
6. ประยุกต์การเขียนโปรแกรมโดยใช้ข้อมูลแบบคอลเลคชันได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน
3. ให้ผู้เรียนแบ่งกลุ่มย่อยเพื่อศึกษาข้อมูลในอินเทอร์เน็ต
4. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
5. ผู้สอนเขียนโปรแกรมตัวอย่างและอธิบายให้กับผู้เรียน
6. ให้ผู้เรียนฝึกปฏิบัติ
7. ให้ผู้เรียนอธิบายผลจากฝึกปฏิบัติ
8. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 6

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 6
4. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. สังเกตจากการฝึกปฏิบัติของผู้เรียน
4. ประเมินจากแบบฝึกหัดบทที่ 6
5. ประเมินจากการสอบปลายภาค

บทที่ 6
โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชัน

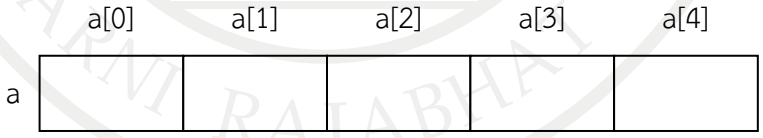
ในการเขียนโปรแกรมที่จำเป็นต้องใช้ตัวแปรเพื่อเก็บข้อมูลเป็นจำนวนมาก จะพบปัญหาในการประกาศตัวแปรเป็นจำนวนมากมาย และอาจจะทำให้เกิดความสับสนในการเรียกใช้ตัวแปร เพราะตัวแปรบางส่วนเป็นชนิดเดียวกัน ตัวอย่างเช่นคะแนนสอบของนักศึกษาในชั้นเรียนที่มีการประกาศตัวแปรเป็น int หรือชื่อของนักศึกษาในชั้นเรียนที่มีการประกาศตัวแปรเป็น string เป็นต้น และยังมีข้อมูลบางส่วนที่เป็นข้อมูลชุดเดียวกันแต่มีชนิดข้อมูลของตัวแปรต่างกัน เช่น นักศึกษาชื่อนายสมชาย ได้คะแนนสอบ 40 เป็นต้น ดังนั้นเพื่อลดปัญหาดังกล่าว จึงต้องจัดหมวดหมู่ให้กับชนิดของ ตัวแปรให้เหมาะสมกับการทำงานของโปรแกรม

6.1 โครงสร้างข้อมูลแบบแถวลำดับ

เป็นโครงสร้างข้อมูลแบบอาร์เรย์ (array) ที่มีการประกาศตัวแปรเพื่อใช้สำหรับเก็บข้อมูลที่เป็นชนิดเดียวกัน มีลักษณะโครงสร้างเป็นแบบแถวลำดับ ซึ่งสามารถกำหนดชื่อตัวแปรเดียวแต่เก็บข้อมูลได้หลายค่า โดยใช้ตัวเลขลำดับหรือดัชนี (index) เป็นตัวชี้ตำแหน่ง ทำให้สามารถเรียกใช้งานจากชื่ออ้างอิงเพียงชื่อเดียว และใช้ตัวเลขดัชนีเพื่อเข้าถึงค่าที่เก็บอยู่ในอาร์เรย์ซึ่งอาจจะเก็บอยู่ในรูปของ 1 มิติ 2 มิติ หรือหลายมิติก็ได้ การกำหนดอาร์เรย์สามารถกำหนดใช้เนื้อที่ในหน่วยความจำได้เท่ากับจำนวนหน่วยความจำที่มีอยู่ในเครื่องคอมพิวเตอร์ (วิษณุ ช้างเนียม, 2553)

6.1.1 อาร์เรย์ 1 มิติ

เป็นอาร์เรย์ที่มีการเก็บข้อมูลเพียงแถวลำดับเดียว โครงสร้างในการเก็บข้อมูลไม่ซับซ้อน สามารถเปลี่ยนแปลงแก้ไขได้ง่าย เช่น a[5] หมายถึง a เป็นตัวแปรที่มีโครงสร้างแบบอาร์เรย์1 มิติ ประกอบไปด้วยสมาชิก 5 ตัว มีดัชนีเริ่มต้นที่ 0 (นิรันดร์ ประวิทย์ธนา, 2545) ดังภาพที่ 6.1



ภาพที่ 6.1 อาร์เรย์แบบ 1 มิติ

สำหรับการประกาศอาร์เรย์ 1 มิติ สามารถทำได้ โดยมีรูปแบบดังนี้ (Microsoft, 2016)

รูปแบบที่ 1

```
<ชนิดของข้อมูล>[] <ชื่อตัวแปร array>;
<ชื่อตัวแปร array> = new <ชนิดของข้อมูล>[<จำนวน array>];
```

ตัวอย่างเช่น

```
int[] a;
a = new int[5];
```

รูปแบบที่ 2

<ชนิดของข้อมูล>[] <ชื่อตัวแปร array> = new <ชนิดของข้อมูล>[<จำนวน array>];

ตัวอย่างเช่น

```
int[] a = new int[5];
```

จากตัวอย่างทั้ง 2 รูปแบบจะเห็นว่าอาร์เรย์ที่กำหนดขึ้นมายังเป็นเพียงแค่การจองหน่วยความจำเอาไว้ใช้สำหรับเก็บข้อมูลเท่านั้น การจะใช้งานต้องใส่ข้อมูลเพิ่มเข้าไปอีกครั้ง เช่น

```
a[0] = 50;
a[1] = 60;
a[2] = 70;
a[3] = 80;
a[4] = 90;
```

หรืออาจจะใช้วิธีการประกาศตัวแปรพร้อมกับการใส่ค่าเลยก็ได้ เช่น

```
int[] num = {1, 2, 3, 4, 5} ;
```

หรือ

```
int[] num = new int[5] {1, 2, 3, 4, 5} ;
```

การประกาศตัวแปรพร้อมใส่ค่านั้น ผู้เขียนไม่จำเป็นต้องใส่ตัวเลขจำนวนของอาร์เรย์ก็ได้ เพราะโปรแกรมจะดูจากค่าที่กำหนดเข้าไปในอาร์เรย์แล้วกำหนดค่าตามจำนวนนั้น ค่าที่ได้จากวิธีประกาศพร้อมตัวแปรมีดังนี้

```
num[0] มีค่าเท่ากับ 1;
num[1] มีค่าเท่ากับ 2;
num[2] มีค่าเท่ากับ 3;
num[3] มีค่าเท่ากับ 4;
num[4] มีค่าเท่ากับ 5;
```

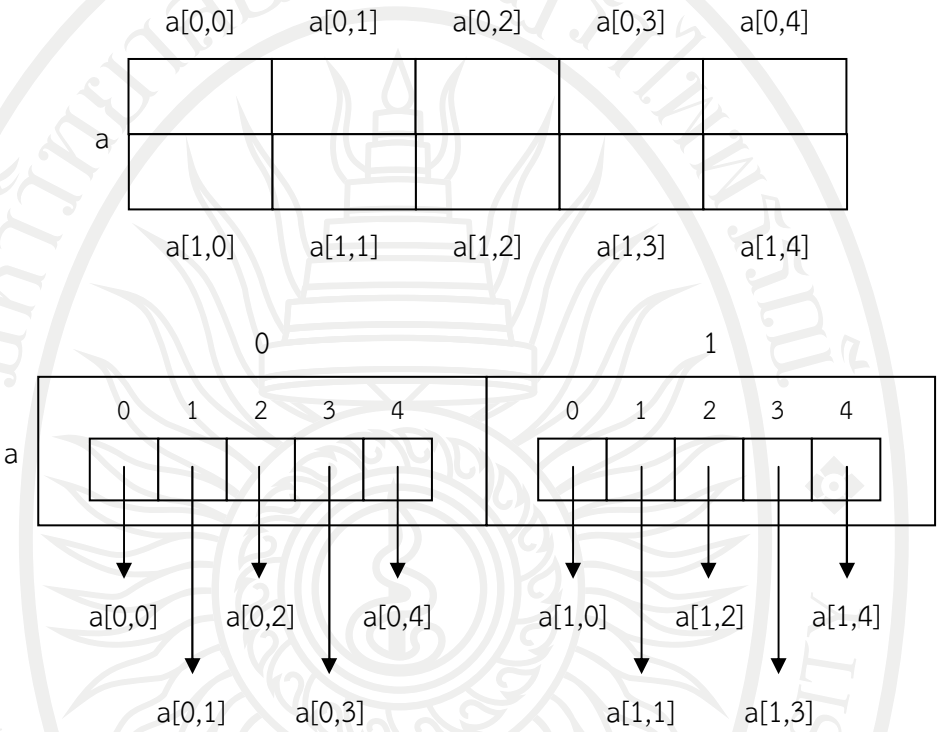
ข้อควรระวังสำหรับการประกาศตัวแปรพร้อมใส่ค่าของอาร์เรย์คือ ถ้าในกรณีที่มีการกำหนดจำนวนของอาร์เรย์แน่นอน จะต้องใส่ค่าให้เท่ากับจำนวนของอาร์เรย์นั้นด้วย

ตัวอย่างที่ไม่ถูกต้อง

```
int[] num = new int[5] {1, 2, 3} ;
int[] num = new int[5] {1, 2, 3, 4, 5, 6, 7} ;
```

6.1.2 อาร์เรย์ 2 มิติ

เป็นอาร์เรย์ที่มีการเก็บข้อมูลคล้ายกับอาร์เรย์1 มิติ เพียงแต่มีแถวเพิ่มขึ้นมาอีก 1 แถว ทำให้การอ้างอิงข้อมูลเปลี่ยนไปจากการอ้างอิงแค่ตัวเลขดัชนีเพียงอย่างเดียว แต่ต้องเพิ่มลำดับของแถวเข้าไปด้วยทำให้เกิดมิติในการเรียกใช้งานดังภาพที่ 6.2



ภาพที่ 6.2 อาร์เรย์แบบ 2 มิติขนาด 2 x 5

สำหรับการประกาศอาร์เรย์ 2 มิติ สามารถทำได้ โดยมีรูปแบบดังนี้

รูปแบบที่ 1

<ชนิดของข้อมูล>[,] <ชื่อตัวแปรอาร์เรย์>;
<ชื่อตัวแปรอาร์เรย์> = new <ชนิดของข้อมูล>[<จำนวนแถว, จำนวนคอลัมน์>];

ตัวอย่างเช่น

```
int[,] a;
a = new int[2, 5];
```

รูปแบบที่ 2

<ชนิดของข้อมูล>[,] <ชื่อตัวแปรอาร์เรย์> = new <ชนิดของข้อมูล>[<จำนวนแถว, จำนวนคอลัมน์>];

ตัวอย่างเช่น

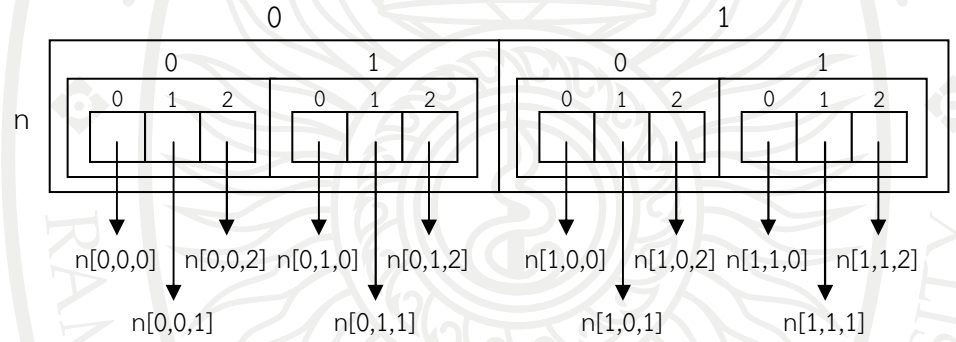
```
int[,] a = new int[2, 5];
```

การใส่ค่าพร้อมกับการประกาศอาร์เรย์ทำได้ดังนี้

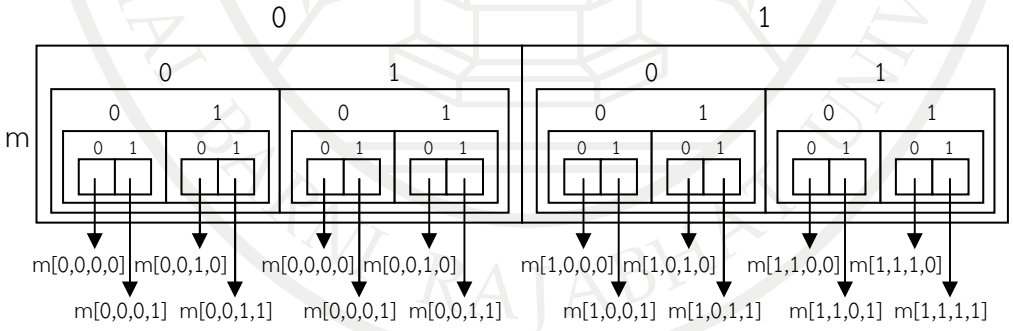
```
int[,] a;
a = new int[2, 5] { {1, 2, 3, 4, 5}, {6, 7, 8, 9, 10} };
int[,] b = { {1, 2, 3, 4, 5}, {6, 7, 8, 9, 10} };
int[,] n = new int[3, 2] { {1, 2}, {3, 4}, {5, 6} };
```

6.1.3 อาร์เรย์หลายมิติ

เป็นอาร์เรย์ที่มีขนาดมากกว่า 2 มิติขึ้นไป ทำให้สามารถใช้ในการเก็บข้อมูลได้มากขึ้น แต่การอ้างอิงตำแหน่งของข้อมูลต้องเพิ่มดัชนีเข้าไปในมิติตามลำดับด้วย ดังภาพที่ 6.3 และ 6.4



ภาพที่ 6.3 อาร์เรย์แบบ 3 มิติขนาด 2 x 2 x 3



ภาพที่ 6.4 อาร์เรย์แบบ 4 มิติขนาด 2 x 2 x 2 x 2

สำหรับการประกาศอาร์เรย์หลายมิติ จะมีรูปแบบคล้ายกันกับอาร์เรย์ 2 มิติ คือ ใช้เครื่องหมายลูกน้ำ (,) คั่นมิติ ซึ่งสามารถทำได้ โดยมีรูปแบบดังนี้

รูปแบบที่ 1

<ชนิดของข้อมูล>[<จำนวนลูกน้ำเท่ากับจำนวนมิติ -1>] <ชื่อตัวแปรอาร์เรย์>;
<ชื่อตัวแปรอาร์เรย์> = new <ชนิดของข้อมูล>[<จำนวนมิติคั่นด้วยเครื่องหมายลูกน้ำ>];

ตัวอย่างเช่น

```
int[, ,] a;
a = new int[2, 5, 2]; //array 3 มิติ
int[, , ,] b;
b = new int[5, 3, 4]; //array 4 มิติ
int[, , , ,] n;
n = new int[10, 15, 4, 5, 3]; //array 5 มิติ
```

รูปแบบที่ 2

<ชนิดของข้อมูล>[<จำนวนลูกน้ำเท่ากับจำนวนมิติ -1>] <ชื่อตัวแปรอาร์เรย์> = new <ชนิดของข้อมูล>[<จำนวนมิติคั่นด้วยเครื่องหมายลูกน้ำ>];

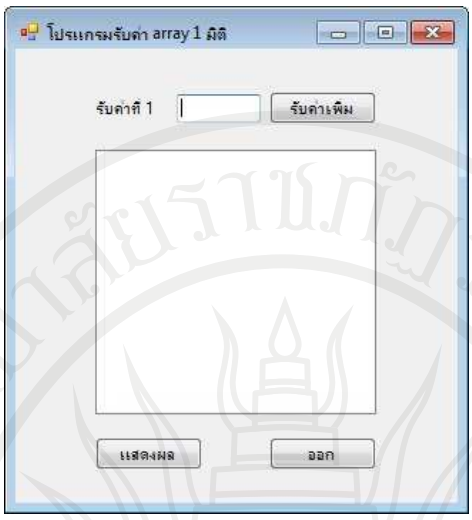
ตัวอย่างเช่น

```
int[, ,] a = new int[2, 5, 2]; //array 3 มิติ
int[, , ,] b = new int[5, 3, 4]; //array 4 มิติ
int[, , , ,] n = new int[10, 15, 4, 5, 3]; //array 5 มิติ
```

การใส่ค่าพร้อมกับการประกาศอาร์เรย์ทำได้ดังนี้

```
int[, ,] a;
a = new int[2, 3, 2] { { {1, 2}, {3, 4}, {5, 6} }, { {7, 8}, {9, 10}, {11, 12} } };
//array 3 มิติ
int[, , ,] b = { { { { 1, 2 }, { 3, 4 } }, { { 5, 6 }, { 7, 8 } } }, { { { 9, 10 }, { 11, 12 } }, { { 13, 14 }, { 15, 16 } } } }; //array 4 มิติ
int[, , , ,] n = new int[3, 2, 2, 2, 2] { { { { { 1, 2 }, { 3, 4 } }, { { 5, 6 }, { 7, 8 } } }, { { { 9, 10 }, { 11, 12 } }, { { 13, 14 }, { 15, 16 } } } }, { { { { 17, 18 }, { 19, 20 } }, { { 21, 22 }, { 23, 24 } } }, { { { 25, 26 }, { 27, 28 } }, { { 29, 30 }, { 31, 32 } } } }, { { { { 33, 34 }, { 35, 36 } }, { { 37, 38 }, { 39, 40 } } }, { { { 41, 42 }, { 43, 44 } }, { { 45, 46 }, { 47, 48 } } } } };
//array 5 มิติ
```


ตัวอย่างที่ 6.1 จงเขียนโปรแกรมรับค่าอาร์เรย์ 1 มิติจำนวน 10 ค่า และแสดงผลใน listBox โดยมีหน้าจอแสดงผลภาพที่ 6.5



ภาพที่ 6.5 หน้าจอโปรแกรมรับค่าลงอาร์เรย์ 1 มิติ

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmArray
- Text กำหนดเป็น “โปรแกรมรับค่าอาร์เรย์1 มิติ”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblInput
- Text กำหนดเป็น “รับค่าที่ 1”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtInput
- Text ใส่เป็นค่าว่าง

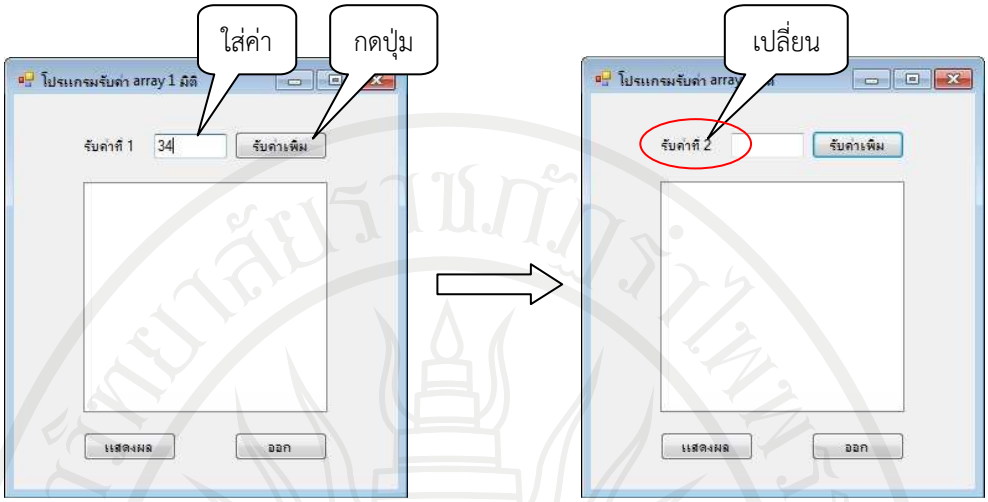
กำหนด Properties สำหรับ listBox1
- Name กำหนดเป็น lstShow
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnInput
- Text กำหนดเป็น “รับค่าเพิ่ม”

กำหนด Properties สำหรับ button2
- Name กำหนดเป็น btnShow
- Text กำหนดเป็น “แสดงผล”

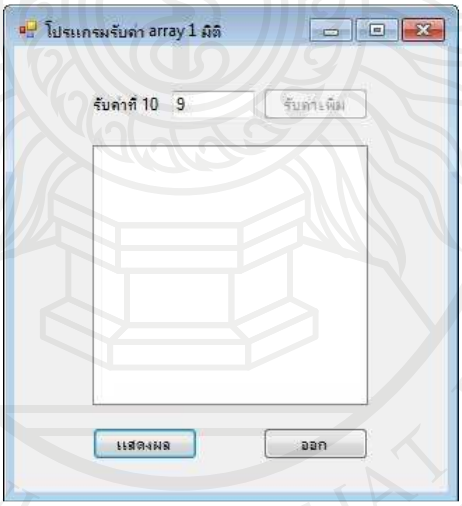
กำหนด Properties สำหรับ button3
- Name กำหนดเป็น btnExit
- Text กำหนดเป็น “ออก”

เมื่อกดปุ่มรับค่าเพิ่ม ให้ lblInput เปลี่ยน Properties ในส่วนของ Text จาก “รับค่าที่ 1” เป็น “รับค่าที่ 2” และเปลี่ยนทุกครั้งที่มีการกดปุ่มรับค่าเพิ่มดังภาพที่ 6.6

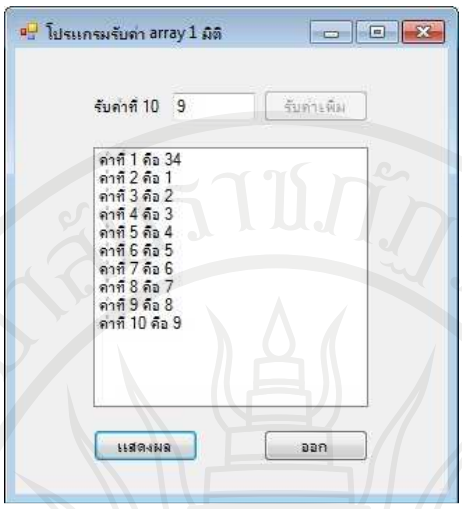


ภาพที่ 6.6 หน้าจอโปรแกรมเมื่อกดปุ่มรับค่าเพิ่ม

เมื่อรับค่าครบทั้ง 10 ค่าแล้ว ให้เปลี่ยนค่า Properties ของปุ่ม btnInput ในส่วนของ Enabled จาก True เป็น False ดังภาพที่ 6.7 และเมื่อกดปุ่มแสดงผลให้แสดงค่าในอาร์เรย์ทั้ง 10 ออกมาแสดงใน lstResult ดังภาพที่ 6.8



ภาพที่ 6.7 หน้าจอโปรแกรมเมื่อรับค่าครบ 10 ค่า



ภาพที่ 6.8 หน้าจอโปรแกรมเมื่อกดปุ่มแสดงผล

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex6_1
10      {
11          public partial class frmArray : Form
12          {
13              public frmArray()
14              {
15                  InitializeComponent();
16              }
17              int[] num = new int[10];
18              int i = 0;
19              private void btnInput_Click(object sender, EventArgs e)
20              {
21                  num[i] = Convert.ToInt32(txtInput.Text);
22                  i++;
```

```
23                  if (i < 10)
24                  {
25                      lblInput.Text = "รับค่าที่ " + (i + 1);
26                      txtInput.Text = "";
27                  }
28                  else
29                      btnInput.Enabled = false;
30              }
31              private void btnShow_Click(object sender, EventArgs e)
32              {
33                  for (int j = 0; j < 10; j++)
34                      lstShow.Items.Add("ค่าที่ " + (j + 1) + " คือ " + num[j]);
35              }
36              private void btnExit_Click(object sender, EventArgs e)
37              {
38                  this.Close();
39              }
40          }
41      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 17 – 18 เป็นส่วนการกำหนดค่าที่ใช้ร่วมกันในฟอร์มทั้งหมด

บรรทัดที่ 21 เป็นส่วนการเพิ่มค่าลงในอาร์เรย์

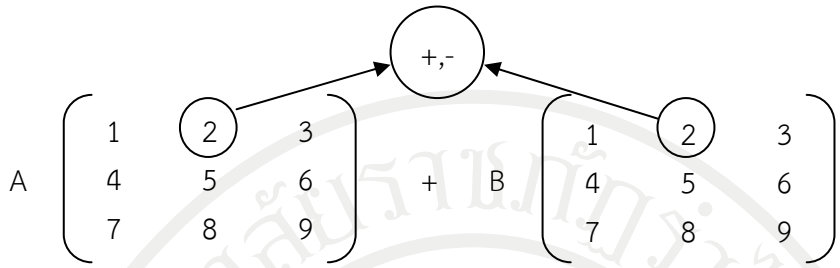
บรรทัดที่ 23 – 29 เป็นส่วนการปรับค่าในการแสดงผลของปุ่มที่ใช้เพิ่มค่า

บรรทัดที่ 33 – 34 เป็นส่วนการเพิ่มค่าลงใน listBox

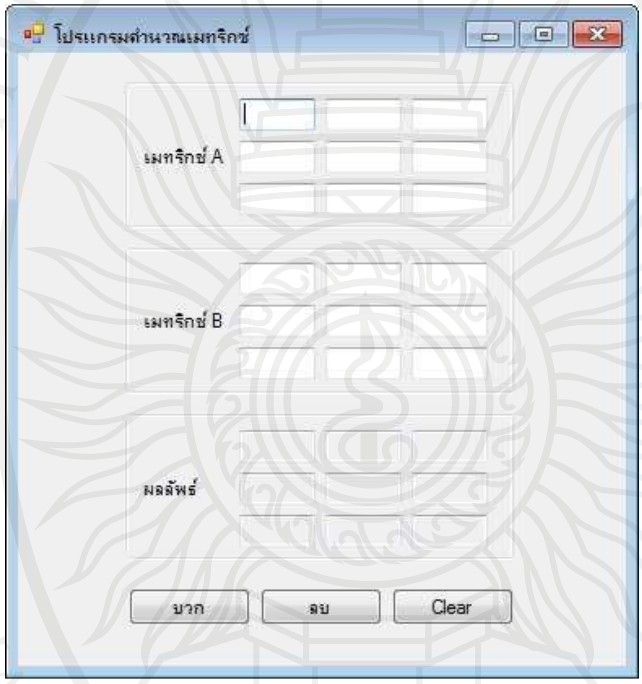
จากตัวอย่างจะเห็นว่ามีการรับค่าลงในอาร์เรย์ไปเรื่อย ๆ จนครบ 10 ค่า และแสดงผลออกใน listBox โดยใช้การทำงานแบบวนซ้ำช่วยในการแสดงผล

ตัวอย่างที่ 6.2 จงเขียนโปรแกรมบวกหรือลบเมทริกซ์ขนาด 3x3 โดยมีหน้าจอแสดงผลดังภาพที่ 6.10

การคำนวณเมทริกซ์ที่มีขนาดเท่ากันสามารถคำนวณได้โดยนำค่าของเมทริกซ์ที่อยู่ในตำแหน่งเดียวกันของเมทริกซ์ A และเมทริกซ์ B มาบวกหรือลบกันดังภาพที่ 6.9 เช่น



ภาพที่ 6.9 ตัวอย่างการคำนวณเมทริกซ์



ภาพที่ 6.10 หน้าจอโปรแกรมคำนวณเมทริกซ์

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmMatrix
- Text กำหนดเป็น “โปรแกรมคำนวณเมทริกซ์”

กำหนด Properties สำหรับ groupBox1
- Name กำหนดเป็น grp1
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ groupBox2
- Name กำหนดเป็น grp2
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ groupBox3

Name กำหนดเป็น grp3
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ label1
Name กำหนดเป็น lblMatrixA
Text กำหนดเป็น “เมทริกซ์ A”

กำหนด Properties สำหรับ label2
Name กำหนดเป็น lblMatrixB
Text กำหนดเป็น “เมทริกซ์ B”

กำหนด Properties สำหรับ label3
Name กำหนดเป็น lblMatrixC
Text กำหนดเป็น “เมทริกซ์ C”

กำหนด Properties สำหรับ textBox1
Name กำหนดเป็น txtA00
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2
Name กำหนดเป็น txtA01
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox3
Name กำหนดเป็น txtA02
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox4
Name กำหนดเป็น txtA10
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox5
Name กำหนดเป็น txtA11
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox6
Name กำหนดเป็น txtA12
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox7
Name กำหนดเป็น txtA20
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox8
Name กำหนดเป็น txtA21
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox9

Name กำหนดเป็น txtA22
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox10
Name กำหนดเป็น txtB00
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox11
Name กำหนดเป็น txtB01
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox12
Name กำหนดเป็น txtB02
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox13
Name กำหนดเป็น txtB10
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox14
Name กำหนดเป็น txtB11
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox15
Name กำหนดเป็น txtB12
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox16
Name กำหนดเป็น txtB20
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox17
Name กำหนดเป็น txtB21
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox18
Name กำหนดเป็น txtB22
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox19
Name กำหนดเป็น txtC00
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox20
Name กำหนดเป็น txtC01
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox21

Name กำหนดเป็น txtC02
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox22
Name กำหนดเป็น txtC10
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox23
Name กำหนดเป็น txtC11
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox24
Name กำหนดเป็น txtC12
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox25
Name กำหนดเป็น txtC20
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox26
Name กำหนดเป็น txtC21
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox27
Name กำหนดเป็น txtC22
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
Name กำหนดเป็น btnAdd
Text กำหนดเป็น “บวก”

กำหนด Properties สำหรับ button2
Name กำหนดเป็น btnSub
Text กำหนดเป็น “ลบ”

กำหนด Properties สำหรับ button3
Name กำหนดเป็น btnClear
Text กำหนดเป็น “Clear”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
```

```
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex6_2
10      {
11          public partial class frmMatrix : Form
12          {
13              public frmMatrix()
14              {
15                  InitializeComponent();
16              }
17              private void btnAdd_Click(object sender, EventArgs e)
18              {
19                  int[,] a = new int[3, 3];
20                  int[,] b = new int[3, 3];
21                  a[0, 0] = Convert.ToInt32(txtA00.Text);
22                  a[0, 1] = Convert.ToInt32(txtA01.Text);
23                  a[0, 2] = Convert.ToInt32(txtA02.Text);
24                  a[1, 0] = Convert.ToInt32(txtA10.Text);
25                  a[1, 1] = Convert.ToInt32(txtA11.Text);
26                  a[1, 2] = Convert.ToInt32(txtA12.Text);
27                  a[2, 0] = Convert.ToInt32(txtA20.Text);
28                  a[2, 1] = Convert.ToInt32(txtA21.Text);
29                  a[2, 2] = Convert.ToInt32(txtA22.Text);
30                  b[0, 0] = Convert.ToInt32(txtB00.Text);
31                  b[0, 1] = Convert.ToInt32(txtB01.Text);
32                  b[0, 2] = Convert.ToInt32(txtB02.Text);
33                  b[1, 0] = Convert.ToInt32(txtB10.Text);
34                  b[1, 1] = Convert.ToInt32(txtB11.Text);
35                  b[1, 2] = Convert.ToInt32(txtB12.Text);
36                  b[2, 0] = Convert.ToInt32(txtB20.Text);
37                  b[2, 1] = Convert.ToInt32(txtB21.Text);
38                  b[2, 2] = Convert.ToInt32(txtB22.Text);
39                  txtC00.Text = Convert.ToString(a[0, 0] + b[0, 0]);
40                  txtC01.Text = Convert.ToString(a[0, 1] + b[0, 1]);
41                  txtC02.Text = Convert.ToString(a[0, 2] + b[0, 2]);
```

```
42              txtC10.Text = Convert.ToString(a[1, 0] + b[1, 0]);
43              txtC11.Text = Convert.ToString(a[1, 1] + b[1, 1]);
44              txtC12.Text = Convert.ToString(a[1, 2] + b[1, 2]);
45              txtC20.Text = Convert.ToString(a[2, 0] + b[2, 0]);
46              txtC21.Text = Convert.ToString(a[2, 1] + b[2, 1]);
47              txtC22.Text = Convert.ToString(a[2, 2] + b[2, 2]);
48          }
49          private void btnSub_Click(object sender, EventArgs e)
50          {
51              int[,] a = new int[3, 3];
52              int[,] b = new int[3, 3];
53              a[0, 0] = Convert.ToInt32(txtA00.Text);
54              a[0, 1] = Convert.ToInt32(txtA01.Text);
55              a[0, 2] = Convert.ToInt32(txtA02.Text);
56              a[1, 0] = Convert.ToInt32(txtA10.Text);
57              a[1, 1] = Convert.ToInt32(txtA11.Text);
58              a[1, 2] = Convert.ToInt32(txtA12.Text);
59              a[2, 0] = Convert.ToInt32(txtA20.Text);
60              a[2, 1] = Convert.ToInt32(txtA21.Text);
61              a[2, 2] = Convert.ToInt32(txtA22.Text);
62              b[0, 0] = Convert.ToInt32(txtB00.Text);
63              b[0, 1] = Convert.ToInt32(txtB01.Text);
64              b[0, 2] = Convert.ToInt32(txtB02.Text);
65              b[1, 0] = Convert.ToInt32(txtB10.Text);
66              b[1, 1] = Convert.ToInt32(txtB11.Text);
67              b[1, 2] = Convert.ToInt32(txtB12.Text);
68              b[2, 0] = Convert.ToInt32(txtB20.Text);
69              b[2, 1] = Convert.ToInt32(txtB21.Text);
70              b[2, 2] = Convert.ToInt32(txtB22.Text);
71              txtC00.Text = Convert.ToString(a[0, 0] - b[0, 0]);
72              txtC01.Text = Convert.ToString(a[0, 1] - b[0, 1]);
73              txtC02.Text = Convert.ToString(a[0, 2] - b[0, 2]);
74              txtC10.Text = Convert.ToString(a[1, 0] - b[1, 0]);
75              txtC11.Text = Convert.ToString(a[1, 1] - b[1, 1]);
76              txtC12.Text = Convert.ToString(a[1, 2] - b[1, 2]);
77              txtC20.Text = Convert.ToString(a[2, 0] - b[2, 0]);
```

```
78              txtC21.Text = Convert.ToString(a[2, 1] - b[2, 1]);
79              txtC22.Text = Convert.ToString(a[2, 2] - b[2, 2]);
80          }
81          private void btnClear_Click(object sender, EventArgs e)
82          {
83              txtA00.Text = "";
84              txtA01.Text = "";
85              txtA02.Text = "";
86              txtA10.Text = "";
87              txtA11.Text = "";
88              txtA12.Text = "";
89              txtA20.Text = "";
90              txtA21.Text = "";
91              txtA22.Text = "";
92              txtB00.Text = "";
93              txtB01.Text = "";
94              txtB02.Text = "";
95              txtB10.Text = "";
96              txtB11.Text = "";
97              txtB12.Text = "";
98              txtB20.Text = "";
99              txtB21.Text = "";
100             txtB22.Text = "";
101             txtC00.Text = "";
102             txtC01.Text = "";
103             txtC02.Text = "";
104             txtC10.Text = "";
105             txtC11.Text = "";
106             txtC12.Text = "";
107             txtC20.Text = "";
108             txtC21.Text = "";
109             txtC22.Text = "";
110         }
111     }
112 }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 19 – 38 เป็นส่วนของการรับค่าลงอาร์เรย์ขนาด 2 มิติ

บรรทัดที่ 39 – 47 เป็นส่วนของการบวกค่าของเมทริกซ์เพื่อเก็บในอาร์เรย์ขนาด 2 มิติ

บรรทัดที่ 53 – 70 เป็นส่วนของการรับค่าลงอาร์เรย์ขนาด 2 มิติ

บรรทัดที่ 71 – 79 เป็นส่วนของการค่าของเมทริกซ์ค่าเพื่อเก็บในอาร์เรย์ขนาด 2 มิติ

บรรทัดที่ 83 – 109 เป็นส่วนของการเคลียร์ค่าใน textBox ทั้งหมด

จากตัวอย่างจะเห็นว่า การใช้อาร์เรย์ในการทำงานจะช่วยประหยัดการกำหนดชื่อตัวแปร ทำให้สามารถใช้ชื่อตัวแปรเพียงตัวเดียว แต่เก็บค่าที่เป็นชนิดเดียวกันได้หลายค่า โดยใช้ตัวเลขดัชนีเป็นตัวกำหนด

6.2 โครงสร้างข้อมูลแบบคอลเลคชัน

คอลเลคชัน (Collection) เป็นโครงสร้างข้อมูลที่มีการเก็บข้อมูลคล้ายกับอาร์เรย์เพียงแต่มีความยืดหยุ่นสูงกว่า เนื่องจากไม่จำเป็นต้องกำหนดขนาดของอาร์เรย์และสามารถเพิ่มข้อมูลได้โดยไม่จำกัด ซึ่งเหมาะสำหรับการจัดการข้อมูลแบบสแตก (stacks) คิว (queues) ลิสต์ (lists) และข้อมูลแบบตารางแฮช (hash tables) สำหรับการใช้คอลเลคชันในวิชวลซีชาร์ปสามารถใช้งานผ่านเนมสเปซ System.Collections หรือ System.Collections.Generic โดยผ่านทางคลาสดังตารางที่ 6.1 และตารางที่ 6.2

ตารางที่ 6.1 คลาสที่สำคัญในเนมสเปซ System.Collections

ชื่อของคลาส	การทำงาน
ArrayList	ใช้สำหรับเก็บค่าในรูปแบบของอาร์เรย์แต่สามารถเพิ่มข้อมูลได้เรื่อย ๆ ตามความต้องการ
Hashtable	ใช้สำหรับจัดการข้อมูลแบบแฮชที่ประกอบด้วยค่าที่เป็นคีย์และค่าที่เป็นข้อมูล
Queue	ใช้สำหรับการจัดการข้อมูลแบบคิว
SortedList	ใช้สำหรับจัดเรียงข้อมูลโดยประกอบไปด้วยค่าที่เป็นคีย์และค่าที่เป็นข้อมูล
Stack	ใช้สำหรับจัดการข้อมูลแบบสแตก

ที่มา : (Microsoft, 2016)

ตารางที่ 6.2 คลาสที่สำคัญในเนมสเปซ System.Collections.Generic

ชื่อของคลาส	การทำงาน
HashSet<T>	ใช้สำหรับแสดงเซตของข้อมูล
LinkedList<T>	ใช้สำหรับจัดการข้อมูลลิงค์ลิสต์แบบ 2 ทาง (doubly linked list)
LinkedListNode<T>	ใช้สำหรับจัดการข้อมูลที่เป็นโหนดของคลาส LinkedList<T> โดยคลาสนี้ไม่สามารถทำการสืบทอดได้

ตารางที่ 6.2 (ต่อ)

ชื่อของคลาส	การทำงาน
List<T>	ใช้สำหรับจัดการข้อมูลในรูปแบบลิสต์ เช่น การค้นหา การจัดเรียงข้อมูล ซึ่งสามารถเข้าถึงได้โดยใช้ดัชนี
Queue<T>	ใช้จัดการข้อมูลแบบคิว
SortedList<TKey, TValue>	ใช้สำหรับจัดเรียงข้อมูลในลิสต์ตามคีย์โดยทำงานร่วมกับคลาส associatedIComparer<T>
SortedSet<T>	ใช้สำหรับจัดเรียงข้อมูลตามรูปแบบที่กำหนด
Stack<T>	ใช้สำหรับจัดการข้อมูลแบบสแตก

ที่มา : (Microsoft, 2016)

การใช้งานคอลเลคชันเพื่อจัดการข้อมูลจะทำให้การทำงานง่ายขึ้น เนื่องจากคอลเลคชันในวิชวลซีชาร์ปจะมีคลาสที่ช่วยในการทำงาน โดยที่ผู้เขียนไม่จำเป็นจะต้องเขียนโครงสร้างการทำงานเอง เช่น การจัดการข้อมูลแบบสแตก หรือการจัดการข้อมูลแบบคิว

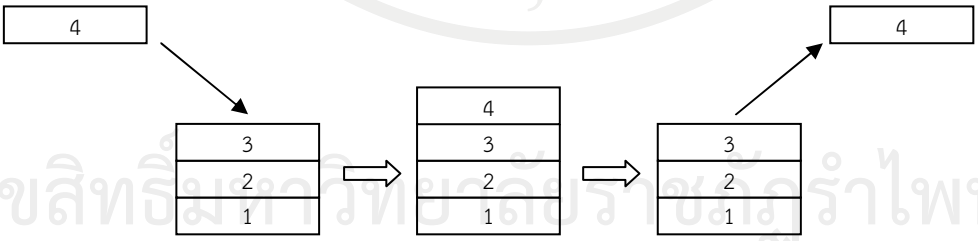
ตัวอย่างการใช้ ArrayList

```
ArrayList list = new ArrayList();
list.Add(100);
list.Add(200);
list.Add(300);
```

จากตัวอย่างจะเห็นว่าการใช้ ArrayList ไม่จะเป็นต้องกำหนดขนาดเหมือนกับอาร์เรย์ทั่วไป แต่สามารถเพิ่มข้อมูลเข้าไปได้เรื่อย ๆ

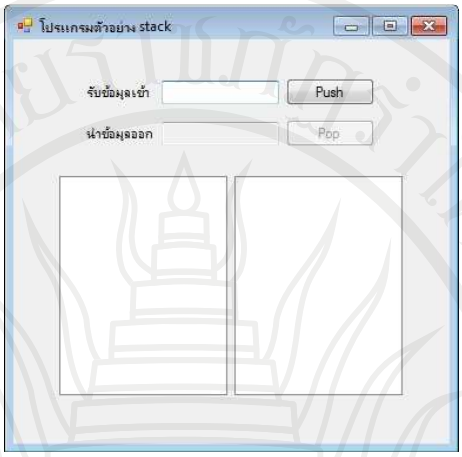
6.2.1 โครงสร้างข้อมูลแบบสแตก

สแตก (Stack) เป็นโครงสร้างข้อมูลที่เก็บข้อมูลในรูปแบบของอาร์เรย์ที่มีเก็บข้อมูลในลักษณะที่ข้อมูลที่เข้ามาก่อนจะอยู่ด้านในสุด เวลาต้องการนำข้อมูลออกมาใช้ ข้อมูลลำดับสุดท้ายจะถูกนำออกมาก่อน ส่วนข้อมูลที่เข้าเป็นลำดับแรกจะถูกนำออกมาเป็นลำดับสุดท้าย ซึ่งจะคล้ายกับการจัดเก็บหนังสือไว้ในกล่อง เวลาที่จะนำออกมาต้องหยิบหนังสือจากด้านบนก่อน ดังภาพที่ 6.11



ภาพที่ 6.11 การทำงานของสแตก

ตัวอย่างที่ 6.3 จงเขียนโปรแกรมจัดการข้อมูลแบบสแตก เพื่อรับข้อมูลเข้าสู่สแตกและนำข้อมูลออกจากสแตก โดยมีหน้าจอแสดงผลดังภาพที่ 6.12



ภาพที่ 6.12 หน้าจอโปรแกรมตัวอย่างสแตก

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmStack
- Text กำหนดเป็น “โปรแกรมตัวอย่าง Stack”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblInput
- Text กำหนดเป็น “รับข้อมูลเข้า”

กำหนด Properties สำหรับ label2
- Name กำหนดเป็น lblOutput
- Text กำหนดเป็น “นำข้อมูลออก”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtInput
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2
- Name กำหนดเป็น txtOutput
- Text ใส่เป็นค่าว่าง
- Enabled กำหนดเป็น False

กำหนด Properties สำหรับ listBox1
- Name กำหนดเป็น lstInput
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ listBox2
- Name กำหนดเป็น lstOutput

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1

Name กำหนดเป็น btnPush

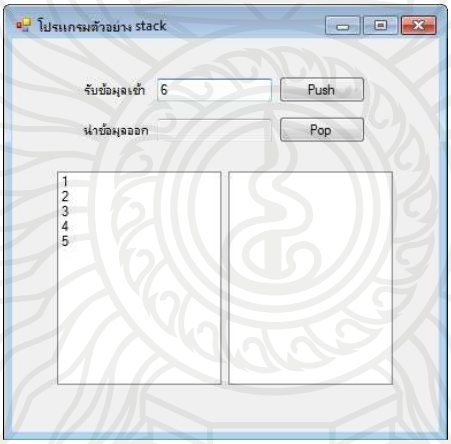
Text กำหนดเป็น “Push”

กำหนด Properties สำหรับ button2

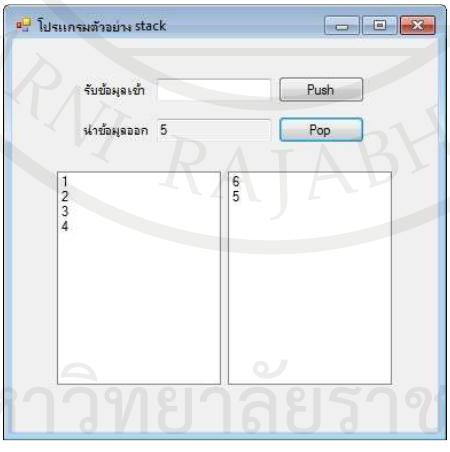
Name กำหนดเป็น btnPop

Text กำหนดเป็น “Pop”

การทำงานของโปรแกรม คือ จะรอรับค่าเพื่อนำข้อมูลเข้าสู่สแตก โดยเมื่อกดปุ่ม btnPush ค่าที่อยู่ใน txtInput จะถูกนำไปเก็บในสแตกและแสดงใน lstInput ดังภาพที่ 6.13 เมื่อนำข้อมูลออกจาก สแตก ข้อมูลใน lstInput จะถูกตัดออกแล้วนำมาแสดงใน txtOutput และ lstOutput ดังภาพที่ 6.14



ภาพที่ 6.13 หน้าจอเมื่อนำข้อมูลเข้าสู่สแตก



ภาพที่ 6.14 หน้าจอเมื่อนำข้อมูลออกจากสแตก

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections;
3       using System.Collections.Generic;
4       using System.ComponentModel;
5       using System.Data;
6       using System.Drawing;
7       using System.Linq;
8       using System.Text;
9       using System.Windows.Forms;
10      namespace Ex6_3
11      {
12          public partial class frmStack : Form
13          {
14              public frmStack()
15              {
16                  InitializeComponent();
17              }
18              Stack myStack = new Stack();
19              int i = 0;
20
21              private void btnPush_Click(object sender, EventArgs e)
22              {
23                  myStack.Push(txtInput.Text);
24                  lstInput.Items.Add(txtInput.Text);
25                  txtInput.Text = "";
26                  i++;
27                  if (i != 0)
28                      btnPop.Enabled = true;
29              }
30              private void btnPop_Click(object sender, EventArgs e)
31              {
32
33                  txtOutput.Text = Convert.ToString(myStack.Pop());
34                  i--;
35                  lstInput.Items.RemoveAt(i);
```

```
36                  lstOutput.Items.Add(txtOutput.Text);
37                  if (i == 0)
38                      btnPop.Enabled = false;
39              }
40          }
41      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 2 และ 3 เป็นส่วนของการเรียกใช้เนมสเปซเกี่ยวกับคอลเลคชัน

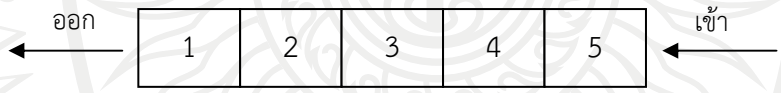
บรรทัดที่ 18 เป็นส่วนของการกำหนดตัวแปรแบบสแตก

บรรทัดที่ 23 เป็นส่วนของการเพิ่มค่าลงสแตก

บรรทัดที่ 33 เป็นส่วนของการนำค่าออกจากสแตก

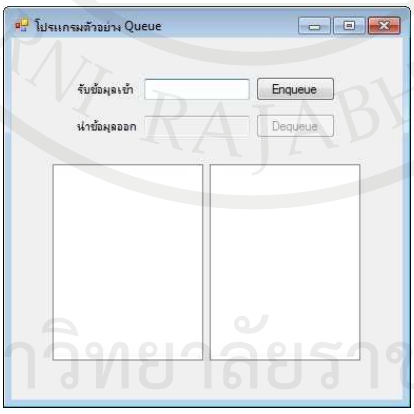
6.2.2 โครงสร้างข้อมูลแบบคิว

คิว (Queue) เป็นโครงสร้างข้อมูลที่มีการเก็บข้อมูลเรียงต่อกันเป็นลำดับ โดยข้อมูลที่เข้ามาก่อนจะอยู่ด้านในสุด ส่วนข้อมูลถัดมาจะเรียงต่อกันเป็นลำดับไปเรื่อย ๆ เวลานำข้อมูลออกจะนำข้อมูลด้านในสุดออกก่อนตามลำดับจนถึงข้อมูลสุดท้าย เปรียบเสมือนการเข้าแถวรับบริการซื้อสินค้า ผู้ที่มาก่อนจะได้รับการบริการก่อนตามลำดับ ดังภาพที่ 6.15



ภาพที่ 6.15 การทำงานคิว

ตัวอย่าง 6.4 จงเขียนโปรแกรมจัดการข้อมูลแบบคิว เพื่อนำข้อมูลเข้าสู่คิวและนำข้อมูลออกจากคิว โดยมีหน้าจอแสดงผลดังภาพที่ 6.16



ภาพที่ 6.16 หน้าจอโปรแกรมตัวอย่างคิว

กำหนด Properties สำหรับ Form

Name	กำหนดเป็น frmQueue
Text	กำหนดเป็น “โปรแกรมตัวอย่าง Queue”

กำหนด Properties สำหรับ label1

Name	กำหนดเป็น lblInput
Text	กำหนดเป็น “รับข้อมูลเข้า”

กำหนด Properties สำหรับ label2

Name	กำหนดเป็น lblOutput
Text	กำหนดเป็น “นำข้อมูลออก”

กำหนด Properties สำหรับ textBox1

Name	กำหนดเป็น txtInput
Text	ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2

Name	กำหนดเป็น txtOutput
Text	ใส่เป็นค่าว่าง
Enabled	กำหนดเป็น False

กำหนด Properties สำหรับ listBox1

Name	กำหนดเป็น lstInput
Text	ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ listBox2

Name	กำหนดเป็น lstOutput
Text	ใส่เป็นค่าว่าง

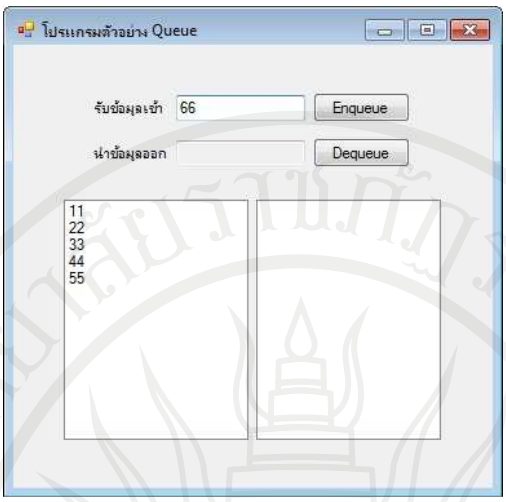
กำหนด Properties สำหรับ button1

Name	กำหนดเป็น btnEnqueue
Text	กำหนดเป็น “Enqueue”

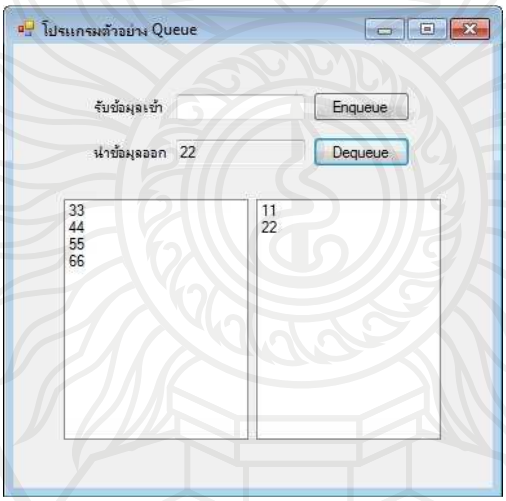
กำหนด Properties สำหรับ button2

Name	กำหนดเป็น btnDequeue
Text	กำหนดเป็น “Dequeue”

การทำงานของโปรแกรมตัวอย่างนี้ จะคล้ายกับโปรแกรมตัวอย่างสแตก คือ จะรับค่าเข้าสู่คิวโดยเมื่อกดปุ่ม btnEnqueue โปแกรมจะนำค่าจาก txtInput เข้าสู่คิวและแสดงค่าใน lstInput ดังภาพที่ 6.17 และเมื่อกดปุ่ม btnDequeue โปรแกรมจะนำค่าออกจากคิวแล้วนำมาแสดงใน txtOutput และ lstOutput ดังภาพที่ 6.18



ภาพที่ 6.17 หน้าจอโปรแกรมเมื่อนำข้อมูลเข้าสู่คิว



ภาพที่ 6.18 หน้าจอโปรแกรมเมื่อนำข้อมูลออกจากคิว

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4   using System.ComponentModel;
5   using System.Data;
6   using System.Drawing;
7   using System.Linq;
8   using System.Text;
9   using System.Windows.Forms;
```

```
10  namespace Ex6
11  {
12      public partial class Form4 : Form
13      {
14          public Form4()
15          {
16              InitializeComponent();
17          }
18          Queue myQueue = new Queue();
19          private void btnEnqueue_Click(object sender, EventArgs e)
20          {
21              myQueue.Enqueue(txtInput.Text);
22              lstInput.Items.Add(txtInput.Text);
23              txtInput.Text = "";
24              if (myQueue.Count != 0)
25                  btnDequeue.Enabled = true;
26          }
27          private void btnDequeue_Click(object sender, EventArgs e)
28          {
29              txtOutput.Text = Convert.ToString(myQueue.Dequeue());
30              lstInput.Items.RemoveAt(0);
31              lstOutput.Items.Add(txtOutput.Text);
32              if (myQueue.Count == 0)
33                  btnDequeue.Enabled = false;
34          }
35      }
36  }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 2 และ 3 เป็นส่วนของการเรียกใช้เนมสเปซเกี่ยวกับคอลเลคชัน

บรรทัดที่ 18 เป็นการกำหนดตัวแปรแบบคิว

บรรทัดที่ 21 เป็นส่วนของการเพิ่มค่าลงคิว

บรรทัดที่ 24 เป็นส่วนของการนับจำนวนข้อมูลในคิว

บรรทัดที่ 29 เป็นส่วนของการนำค่าออกจากคิว

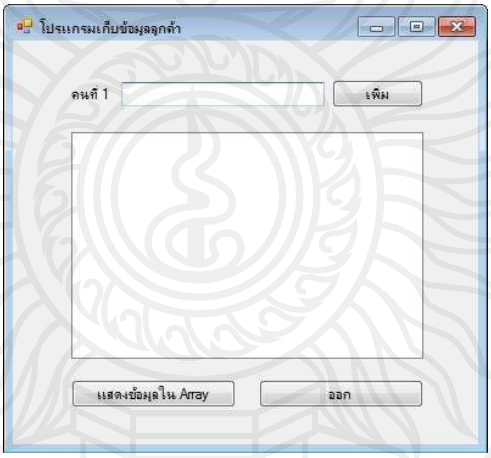
6.3 สรุป

โครงสร้างข้อมูลแบบเรียงลำดับหรืออาร์เรย์สามารถใช้จัดการกับข้อมูลที่มีชนิดเหมือนกันได้ โดยการประกาศตัวแปรเพียงชื่อเดียว และใช้เลขดัชนีเป็นตัวกำหนดตำแหน่งของข้อมูล ซึ่งโดยทั่วไป อาร์เรย์มีได้หลายมิติขึ้นอยู่กับการใช้งาน และการจองเนื้อที่สำหรับอาร์เรย์สามารถจองได้เท่ากับหน่วยความจำที่มีอยู่ในเครื่องคอมพิวเตอร์นั้นซึ่งสามารถทำงานได้เร็วกว่าการเรียกข้อมูลจากไฟล์มาประมวลผล การกำหนดอาร์เรย์สามารถทำได้ตั้งแต่ 1 มิติ 2 มิติ ไปจะถึงขนาดที่มากกว่า 2 มิติที่เรียกว่าหลายมิติ ซึ่งผู้เขียนโปรแกรมจำเป็นจะต้องเข้าใจการอ้างอิงถึงตำแหน่งข้อมูลในอาร์เรย์เป็นอย่างดี เพื่อป้องกันความผิดพลาดในการเรียกใช้ข้อมูล

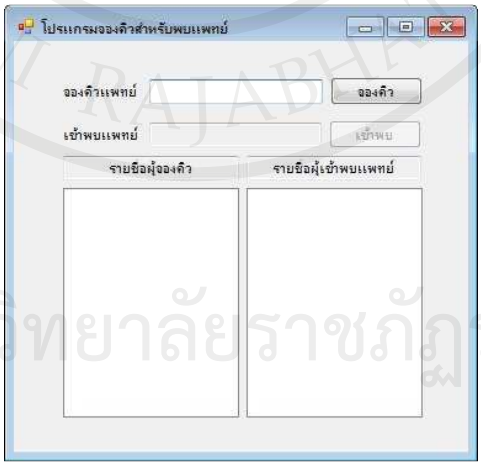
ส่วนโครงสร้างข้อมูลแบบคอลเลคชัน เป็นการจัดการข้อมูลที่คล้ายกับอาร์เรย์แต่มีความยืดหยุ่นสูงกว่า คือไม่ต้องกำหนดจำนวนที่แน่นอนสำหรับการจองหน่วยความจำที่ใช้เพื่อเก็บข้อมูลและสามารถเพิ่มข้อมูลเข้าไปได้เรื่อย ๆ เหมาะสำหรับการจัดการข้อมูลเช่น สแตก คิว หรือตารางแฮช เป็นต้น การใช้งานคอลเลคชันจำเป็นต้องเรียกใช้ผ่านเนมสเปซที่วิชวลซีชาร์ปมีให้ คือ เนมสเปซ System.Collections system.Collections.Generic;

แบบฝึกหัดบทที่ 6

1. โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชันแตกต่างกันอย่างไร
2. จงอธิบายการทำงานของโครงสร้างข้อมูลแบบ Array
3. จงกำหนดตัวแปรแบบอาร์เรย์1 มิติสำหรับเก็บข้อมูลชนิดตัวเลขจำนวน 5 ตัว
4. การกำหนด int[,] n = new int[3, 2] { {1, 2}, {3, 4}, {5, 6} }; หมายถึงอะไร
5. การเก็บข้อมูลแบบสแตกเป็นอย่างไร
6. จงกำหนดตัวแปรชื่อ myStack ให้เป็นข้อมูลแบบสแตก
7. โครงสร้างข้อมูลแบบคิวมีการทำงานอย่างไร
8. จงยกตัวอย่างของการใช้ข้อมูลแบบคิวในชีวิตประจำวัน มาสัก 3 ตัวอย่าง
9. จงเขียนโปรแกรมสำหรับเก็บชื่อลูกค้าจำนวน 10 คน และเก็บข้อมูลเป็นโครงสร้างข้อมูลชนิดอาร์เรย์โดยมีหน้าจอดังนี้



10. จงเขียนโปรแกรมจองคิวสำหรับพบแพทย์และเก็บชื่อของคนไข้ลงในโครงสร้างข้อมูลชนิดคิว โดยมีหน้าจอดังนี้



เอกสารอ้างอิง

นิรันดร์ ประวิทย์ธนา. (2545). **เก่ง C# ให้ครบสูตร**. กรุงเทพฯ : วิตตี้ กรุ๊ป.

วิษณุ ช้างเนียม. (2553). **คู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม.** กรุงเทพฯ : ไอดีซี พรีเมียร์.

Microsoft. (2016). **Arrays Tutorial**. (online). Available : https://msdn.microsoft.com/en-us/library/aa288453(v=vs.71).aspx. 10 March 2016.

_____. (2016). **System.Collections Namespace**. (online). Available : https://msdn.microsoft.com/en-us/library/system.collections(v=vs.100).aspx. 10 March 2016.

_____. (2016). **System.Collections.Generic Namespace**. (online). Available : https://msdn.microsoft.com/en-us/library/system.collections.generic(v=vs.100).aspx. 10 March 2016.

แผนบริหารประจำบทที่ 7

เนื้อหา

บทที่ 7 คลาสและความสัมพันธ์ระหว่างคลาส

7.1 การสร้างคลาสและการสร้างวัตถุจากคลาส

7.2 ความสัมพันธ์ระหว่างคลาส

7.3 การใช้งานดีไซน์แพทเทิร์น

7.4 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. สร้างคลาสจากคลาสไดอะแกรมได้
2. สร้างวัตถุจากคลาสได้
3. ออกแบบส่วนติดต่อกับผู้ใช้จากคลาสไดอะแกรมได้
4. ออกแบบฐานข้อมูลจากคลาสไดอะแกรมได้
5. เชื่อมโยงความสัมพันธ์ระหว่างคลาสได้
6. อธิบายการใช้งานดีไซน์แพทเทิร์นได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน และค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ผู้สอนเขียนโปรแกรมตัวอย่างและอธิบายให้กับผู้เรียน
5. ผู้สอนสุ่มถามความเข้าใจจากผู้เรียน
6. ให้ผู้เรียนฝึกปฏิบัติ
7. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 7

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 7
4. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. ประเมินจากการสุ่มถามผู้เรียน
4. ประเมินจากแบบฝึกหัดบทที่ 7
5. ประเมินจากการสอบปลายภาค

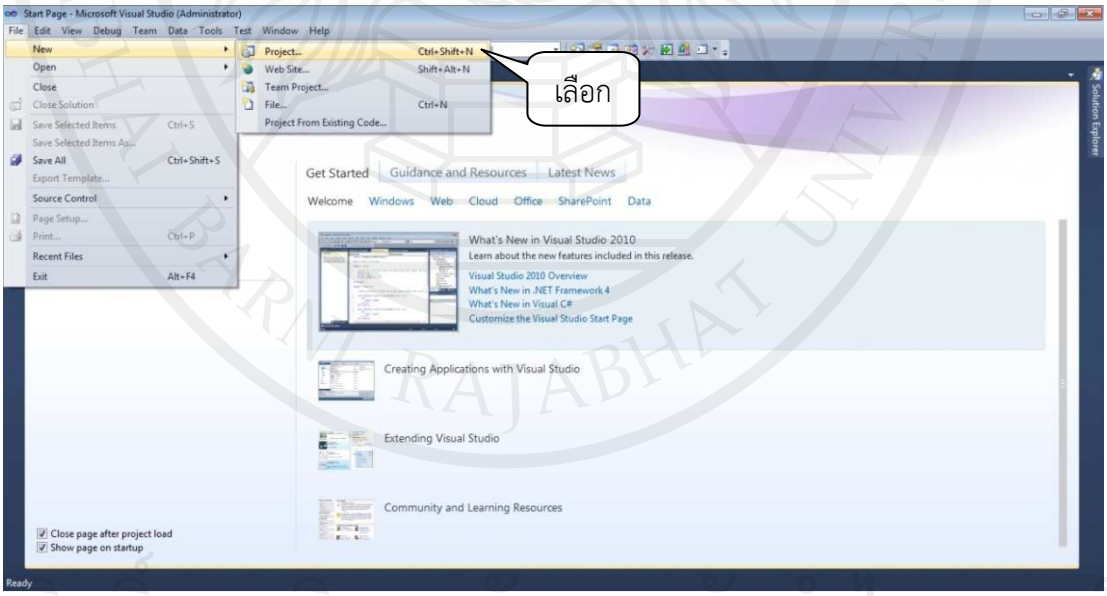
บทที่ 7
คลาสและความสัมพันธ์ระหว่างคลาส

ในการเขียนโปรแกรมเชิงวัตถุจะมีคลาสเป็นส่วนประกอบที่สำคัญ เนื่องจากเป็นส่วนที่เป็นต้นแบบในการสร้างวัตถุที่ประกอบขึ้นเป็นระบบ ดังนั้นความสัมพันธ์ระหว่างคลาสจึงเป็นความสัมพันธ์ระหว่างวัตถุด้วย คลาสที่ออกแบบมาจากยูเอ็มแอลเมื่อนำมาใช้ในการเขียนโปรแกรมสามารถปรับเปลี่ยนเป็น 3 รูปแบบ คือ สร้างเป็นฐานข้อมูล สร้างเป็นส่วนติดต่อกับผู้ใช้ และสร้างเป็นส่วนในการควบคุมการทำงานต่าง ๆ นอกจากนี้คลาสยังมีความสามารถในการห่อหุ้ม การพ้องรูป และการสืบทอดด้วย

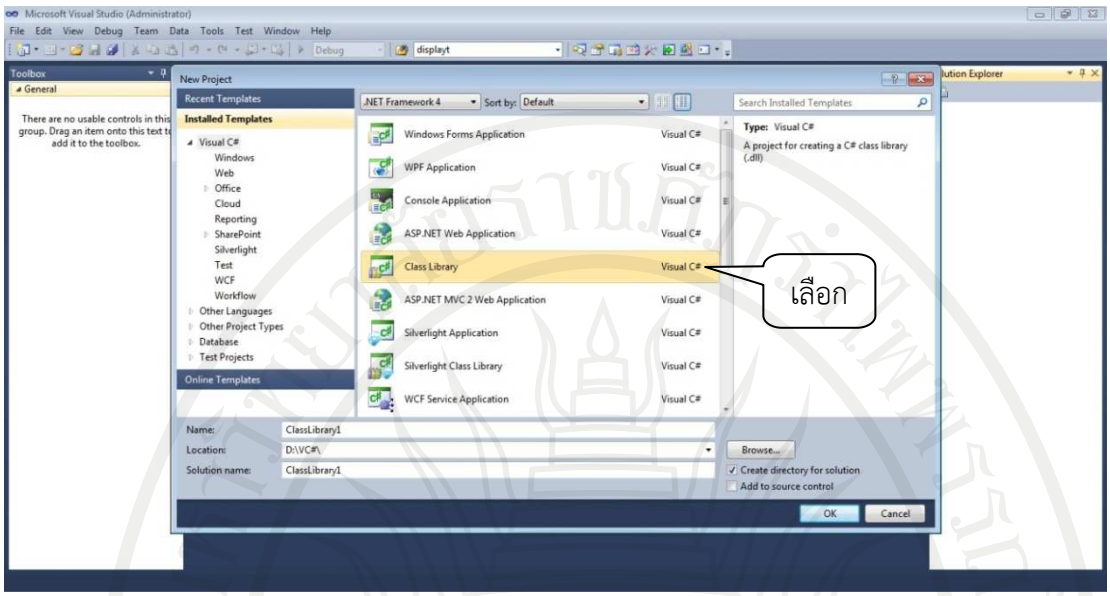
7.1 การสร้างคลาสและการสร้างวัตถุจากคลาส

7.1.1 การสร้างคลาสในวิชวลซีชาร์ป

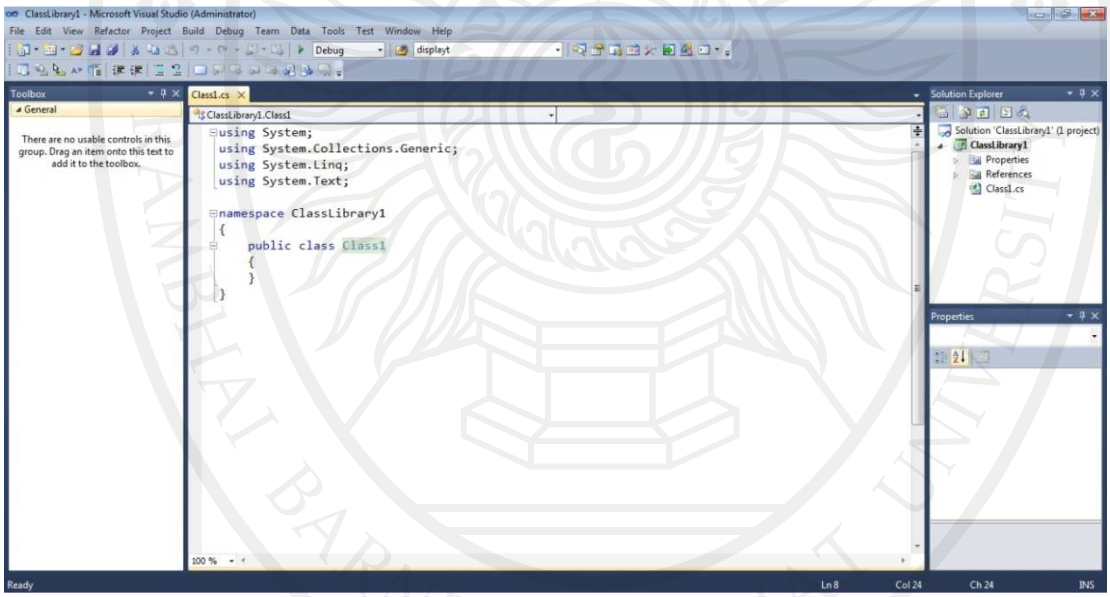
เมื่อออกแบบคลาสในระบบแล้ว เมื่อนำมาสร้างเป็นคลาสสำหรับการเขียนโปรแกรมในวิชวลซีชาร์ปสามารถทำได้ 2 วิธี คือ สร้างเป็น Class Library โดยเลือกที่เมนู File -> New -> Project หรือกด Ctrl+Shift+N ดังภาพที่ 7.1 แล้วเลือกวิชวลซีชาร์ปจะได้หน้าจอดังภาพที่ 7.2 หรือสร้างเป็นคลาสในการเขียนโปรแกรมทั่วไปใน Windows Forms Application และ Console Application โดยเลือกที่ Project -> Add Class หรือกด Shift+Alt+C ดังภาพที่ 7.3 และ 7.4



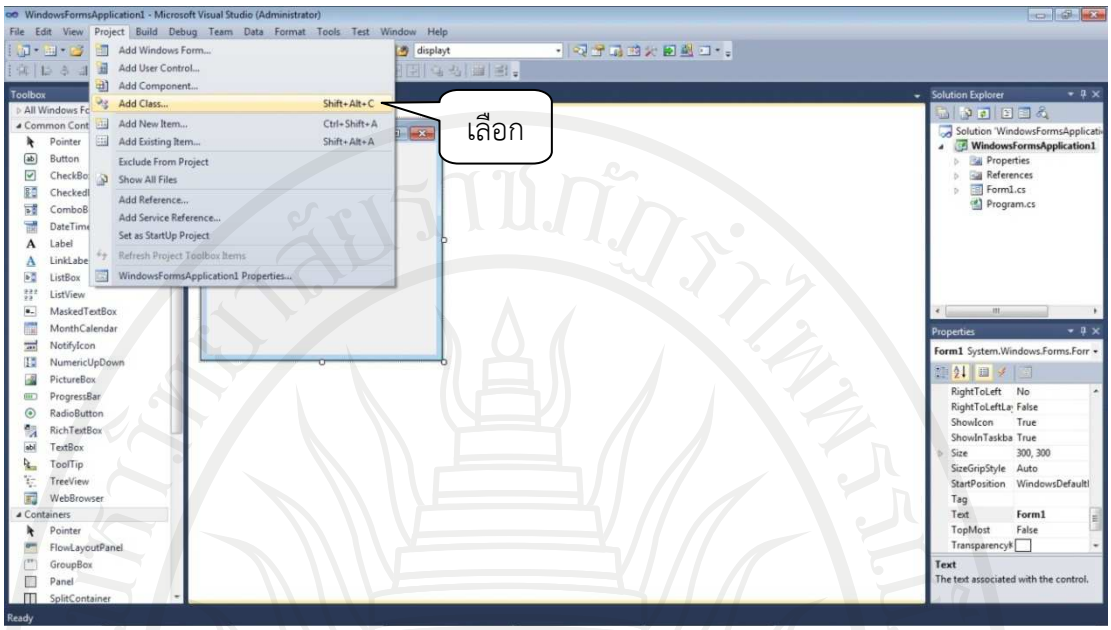
ภาพที่ 7.1 ขั้นตอนการสร้าง Project



ภาพที่ 7.2 การสร้าง Class Library

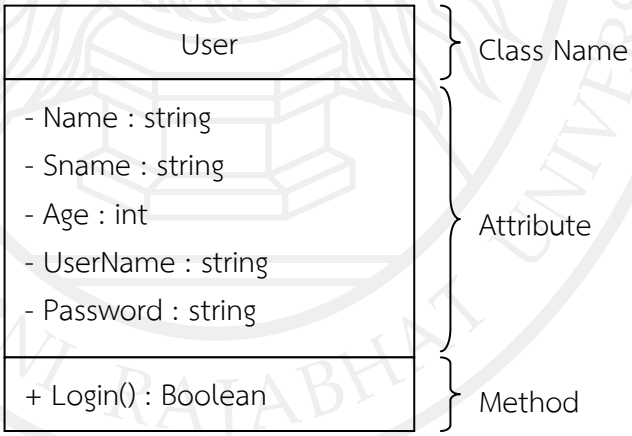


ภาพที่ 7.3 หน้าจอสำหรับสร้างคลาสใน Class Library



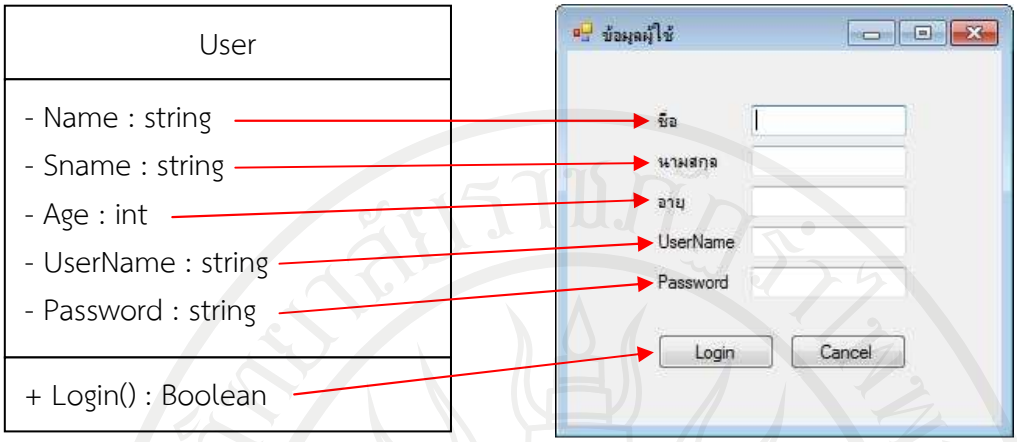
ภาพที่ 7.4 การสร้างคลาสใน Windows Forms Application

หลังจากสร้างส่วนสำหรับเขียนคลาสขึ้นมาแล้ว ผู้เขียนจะต้องนำเอาคลาสที่ได้ออกแบบตามยูเอ็มแอลมาแปลงเป็นส่วนที่ใช้ในการเขียนโปรแกรมได้ เช่น ส่วนติดต่อกับผู้ใช้ ฐานข้อมูล และการทำงานคำสั่ง โดยสามารถแปลงได้ดังภาพที่ 7.5



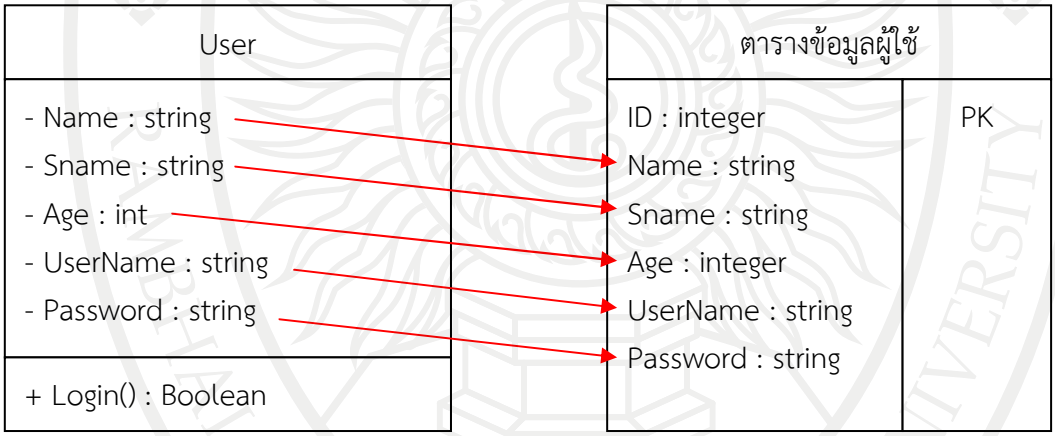
ภาพที่ 7.5 ตัวอย่างคลาส User

การแปลงไปเป็นส่วนติดต่อกับผู้ใช้หรือฟอร์มในการทำงานได้ดังภาพที่ 7.6



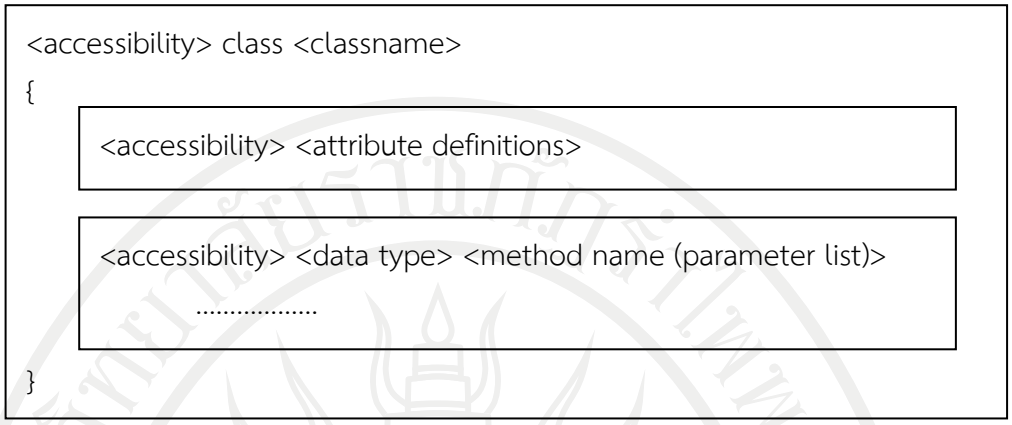
ภาพที่ 7.6 ตัวอย่างการแปลงคลาสไปเป็นส่วนติดต่อกับผู้ใช้

ส่วนการแปลงเป็นฐานข้อมูล สามารถใช้ส่วนที่เป็นคุณสมบัติของคลาสใช้ในการออกแบบได้ดังภาพที่ 7.7



ภาพที่ 7.7 ตัวอย่างการแปลงคลาสไปเป็นฐานข้อมูล

การแปลงไปเป็นรหัสคำสั่งในการเขียนโปรแกรมวิชวลซีชาร์ปจำเป็นจะต้องทราบโครงสร้างของการเขียนคลาสในวิชวลซีชาร์ปก่อน ซึ่งมีรายละเอียดดังภาพที่ 7.8



ภาพที่ 7.8 โครงสร้างการเขียนคลาสในวิชวลซีชาร์ป
ที่มา : (ธีระพล ลิ้มศรัทธา, 2553)

<accessibility> เป็นส่วนที่กำหนดการเข้าถึงข้อมูล โดยมีรายละเอียดดังนี้
public (+) หมายถึง สามารถเข้าถึงได้โดยตรงจากคลาสภายนอก
private (-) หมายถึง สามารถเข้าถึงหรือถูกใช้งานจากภายในคลาสเท่านั้น
protected (#) หมายถึง สามารถเข้าถึงได้จากภายในคลาสย่อยเท่านั้น
<classname> เป็นส่วนที่ใช้กำหนดชื่อคลาสที่ใช้ในการทำงาน
<attribute definitions> เป็นส่วนที่ใช้ประกาศตัวแปรต่าง ๆ เพื่อบอกคุณสมบัติของคลาส
<data type> เป็นส่วนกำหนดประเภทของข้อมูล ถ้าเป็น void จะไม่มีการส่งค่าคืน
<method name (parameter list)> เป็นส่วนที่ใช้กำหนดชื่อของเมธอดและค่าพารามิเตอร์

ตัวอย่างของการแปลงคลาส User ไปเป็นคลาสในโปรแกรม

```
1   public class User                   // Class Name
2   {
3       private string Name;            // - Name : string
4       private string Sname;           // - Sname : string
5       private int Age;                // - Age : int
6       private string UserName;        // - UserName : string
7       private string Password;        // - Password : string
8       public Boolean Login()          // + Login() : Boolean
9       {
10          return true;
11      }
12  }
```

7.1.2 การห่อหุ้มหรือการซ่อนรายละเอียด

สำหรับส่วนที่เป็นคุณสมบัติในคลาส จำเป็นจะต้องมีการป้องกันไม่ใช้คลาสจากภายนอกเข้ามาเปลี่ยนแปลงแก้ไขข้อมูลได้โดยตรง โดยส่วนใหญ่คุณสมบัติในคลาสจะมีการตั้งค่า <accessibility> เป็น private อยู่แล้ว แต่ในกรณีที่ต้องการแก้ไขข้อมูลสามารถใช้บริการโดยผ่านทางเมธอด get และ set ซึ่งเป็นมาตรฐานที่ใช้สำหรับการเขียนโปรแกรมเชิงวัตถุมีรูปแบบดังนี้

รูปแบบที่ 1

```
private <ชนิดข้อมูล> <ชื่อตัวแปรที่เป็นคุณสมบัติ>;
public <ชนิดข้อมูล> <ชื่อตัวแปรที่สอดคล้องกับคุณสมบัติ>
{
    get { return <ชื่อตัวแปรที่เป็นคุณสมบัติ>; }
    set { <ชื่อตัวแปรที่เป็นคุณสมบัติ> = value; }
}
```

ตัวอย่างเช่น

```
class User
{
    private string _Name;
    public string Name
    {
    get { return _Name; }
    set { _Name = value; }
    }
}
```

รูปแบบที่ 2

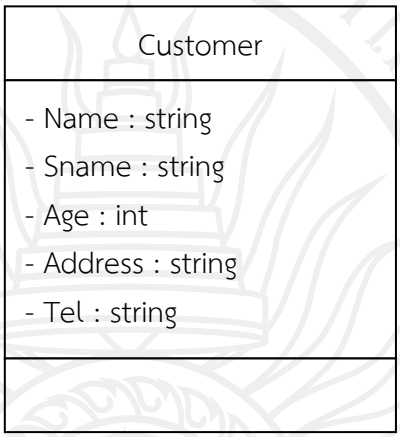
```
public <ชนิดข้อมูล> <ชื่อตัวแปรที่เป็นคุณสมบัติ> { get; set; }
```

ตัวอย่างเช่น

```
class User
{
    public string Name { get; set; }
}
```

จากรูปแบบทั้ง 2 อย่างจะเห็นว่า ในรูปแบบที่ 2 จะเขียนคำสั่งสั้นกว่ารูปแบบที่ 1 เนื่องจากเป็นการลดรูปเพื่อให้การเขียนโปรแกรมสะดวกขึ้น

ตัวอย่างที่ 7.1 จงเขียนโปรแกรมสร้างคลาสจากคลาสไดอะแกรมดังภาพที่ 7.9 โดยให้มีการซ่อนรายละเอียดในส่วนที่เป็นคุณสมบัติของคลาส



ภาพที่ 7.9 คลาสไดอะแกรม Customer

ชุดคำสั่งของโปรแกรมเป็นดังนี้

รูปแบบที่ 1

```
1       using System;
2       using System.Collections.Generic;
3       using System.Linq;
4       using System.Text;
5       namespace Ex7_1
6       {
7           class Customer
8           {
9               private string _Name;
10              private string _Sname;
11              private int _Age;
12              private string _Address;
13              private string _Tel;
14              public string Name
15              {
16                  get { return _Name; }
```

```
17                  set { _Name = value; }
18              }
19              public string Sname
20              {
21                  get { return _Sname; }
22                  set { _Sname = value; }
23              }
24              public int Age
25              {
26                  get { return _Age; }
27                  set { _Age = value; }
28              }
29              public string Address
30              {
31                  get { return _Address; }
32                  set { _Address = value; }
33              }
34              public string Tel
35              {
36                  get { return _Tel; }
37                  set { _Tel = value; }
38              }
39          }
40      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 – 13 เป็นส่วนการกำหนดค่าคุณสมบัติของคลาส

บรรทัดที่ 14 – 38 เป็นส่วนการกำหนด get และ set สำหรับปรับเปลี่ยนค่าของคุณสมบัติแต่ละตัว

รูปแบบที่ 2

```
1       using System;
2       using System.Collections.Generic;
3       using System.Linq;
4       using System.Text;
5       namespace Ex7_1
```

```
6       {
7           class Customer
8           {
9               public string Name { get; set; }
10              public string Sname { get; set; }
11              public int Age { get; set; }
12              public string Address { get; set; }
13              public string Tel { get; set; }
14          }
15      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 – 13 เป็นส่วนการกำหนดค่าคุณสมบัติของคลาสและกำหนด get และ set สำหรับปรับเปลี่ยนค่าของคุณสมบัติแต่ละตัวแบบย่อ

7.1.3 การสร้างวัตถุจากคลาส

เมื่อสร้างคลาสเรียบร้อยแล้วการใช้งานคลาสจำเป็นต้องสร้างวัตถุจากคลาสขึ้นมาก่อน เพราะโดยปกติทั่วไปจะไม่สามารถใช้งานคลาสได้โดยตรง เมื่อสร้างวัตถุจากคลาสแล้วจะสามารถใช้งานโดยผ่านทางวัตถุที่สร้างขึ้น การสร้างวัตถุจะใช้คำสั่ง new ในการสร้าง ซึ่งมีรูปแบบดังนี้

รูปแบบที่ 1

```
<classname> <ชื่อวัตถุ>;
<ชื่อวัตถุ> = new <classname>();
```

ตัวอย่างเช่น

```
Customer John;
John = new Customer();
```

รูปแบบที่ 2

```
<classname> <ชื่อวัตถุ> = new <classname>();
```

ตัวอย่างเช่น

```
Customer John = new Customer();
```

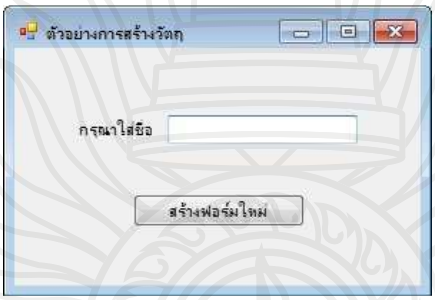
เมื่อสร้างวัตถุจากคลาสแล้ว จะสามารถใช้วัตถุทำงานได้ โดยกำหนดค่าต่าง ๆ ที่เป็นคุณสมบัติหรือใช้ Method ที่มีอยู่ในวัตถุได้ เช่น


```
Customer John = new Customer();
```

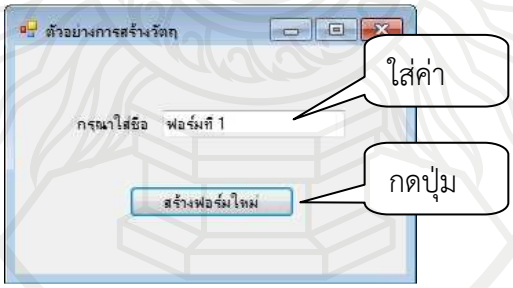
```
John.Name = "Somchai";
John.Sname = "Thongdee";
John.Age = 35;
John.Address = "Chanthaburi";
Joe.Tel = "0991234567";
```

} กำหนดคุณสมบัติ

ตัวอย่างที่ 7.2 จงเขียนโปรแกรมสร้างวัตถุจากคลาส Form โดยเมื่อใส่ชื่อในช่อง textBox แล้วกดปุ่มสร้างฟอร์มใหม่ โปรแกรมจะแสดง MessageBox แสดงชื่อ และสร้าง Form ใหม่ขึ้นมาโดยมี TitleBar แสดงเป็นชื่อที่ใส่ใน textBox ดังภาพที่ 7.10 ถึง 7.13



ภาพที่ 7.10 หน้าจอตัวอย่างการสร้างวัตถุจากคลาส Form



ภาพที่ 7.11 หน้าจอเมื่อโปรแกรมทำงาน

ภาพที่ 7.12 แสดง MessageBox หลังจากกดปุ่มสร้างฟอร์มใหม่

ภาพที่ 7.13 หน้าจอ Form ใหม่ที่ถูกสร้างขึ้นมา

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmCreate
- Text กำหนดเป็น “ตัวอย่างการสร้างวัตถุ”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblInput
- Text กำหนดเป็น “กรุณาใส่ชื่อ”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtInput
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnNew
- Text กำหนดเป็น “สร้างฟอร์มใหม่”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex7_2
10      {
11          public partial class frmCreate : Form
12          {
13              public frmCreate()
```

```
14          {
15              InitializeComponent();
16          }
17          public string txtName { get; set; }
18          private void btnNew_Click(object sender, EventArgs e)
19          {
20              txtName = txtInput.Text;
21              NewForm();
22          }
23          private void NewForm()
24          {
25              frmCreate frm = new frmCreate();
26              MessageBox.Show(txtName);
27              frm.Text = txtName;
28              frm.Show();
29          }
30      }
31  }
```

อธิบายชุดคำสั่งของโปรแกรม

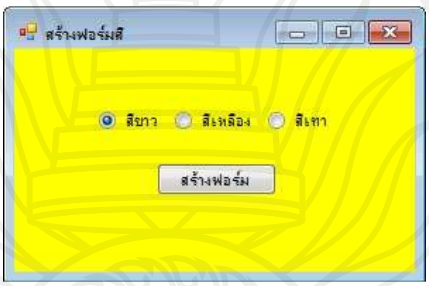
บรรทัดที่ 17 เป็นส่วนการกำหนดค่าคุณสมบัติของคลาสและกำหนด get และ set สำหรับปรับเปลี่ยนค่าของคุณสมบัติ

บรรทัดที่ 25 เป็นส่วนการสร้างวัตถุ frm จากคลาส frmCreate

ตัวอย่างที่ 7.3 จงเขียนโปรแกรมสร้างฟอร์มใหม่โดยมีสีพื้นหลังตามการเลือกของผู้ใช้ ดังภาพที่ 7.14 ถึง 7.16

ภาพที่ 7.14 หน้าจอตัวอย่างการสร้างฟอร์มสี

ภาพที่ 7.15 หน้าจอขณะผู้ใช้เลือกสีของฟอร์ม



ภาพที่ 7.16 หน้าจอฟอร์มใหม่ตามสีที่เลือก

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmColor
- Text กำหนดเป็น “สร้างฟอร์มสี”

กำหนด Properties สำหรับ radioButton1
- Name กำหนดเป็น rdoWhite
- Text กำหนดเป็น “สีขาว”

กำหนด Properties สำหรับ radioButton2
- Name กำหนดเป็น rdoYellow
- Text กำหนดเป็น “สีเหลือง”

กำหนด Properties สำหรับ radioButton3
- Name กำหนดเป็น rdoGray
- Text กำหนดเป็น “สีเทา”

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnNew
- Text กำหนดเป็น “สร้างฟอร์ม”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
```

```
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex7_3
10      {
11          public partial class frmColor : Form
12          {
13              public frmColor()
14              {
15                  InitializeComponent();
16              }
17              private void btnNew_Click(object sender, EventArgs e)
18              {
19                  frmColor frm = new frmColor();
20                  if (rdoWhite.Checked == true)
21                      frm.BackColor = Color.White;
22                  else if (rdoYellow.Checked == true)
23                      frm.BackColor = Color.Yellow;
24                  else if (rdoGray.Checked == true)
25                      frm.BackColor = Color.Gray;
26                   frm.Show();
27              }
28          }
29      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 19 เป็นส่วนการสร้างวัตถุ frm จากคลาส frmColor

บรรทัดที่ 21 – 25 เป็นส่วนการกำหนดค่าคุณสมบัติของวัตถุโดยใช้การตรวจสอบเงื่อนไขจาก radioButton

บรรทัดที่ 26 เป็นส่วนการเรียกใช้เมธอด Show ของฟอร์มมาใช้

ตัวอย่างที่ 7.4 จงเขียนโปรแกรมสำหรับการคำนวณเส้นรอบวงและพื้นที่ของวงกลม โดยใช้ค่ารัศมีเป็นข้อมูลในการคำนวณตามคลาส Cycle ดังภาพที่ 7.17 และ 7.18

สูตร เส้นรอบวง = $2\pi r$

พื้นที่วงกลม = πr^2

π มีค่าเท่ากับ 3.1415

Cycle
- radius : double
+ Girth() : double + Area() : double

ภาพที่ 7.17 คลาส Cycle

ภาพที่ 7.18 หน้าจอโปรแกรมคำนวณวงกลม

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmCycle

Text กำหนดเป็น “คำนวณวงกลม”

กำหนด Properties สำหรับ label1

Name กำหนดเป็น lblInput

Text กำหนดเป็น “รัศมี”

กำหนด Properties สำหรับ label2

Name กำหนดเป็น lblOutput

Text กำหนดเป็น “ผลลัพธ์”

กำหนด Properties สำหรับ textBox1

Name กำหนดเป็น txtInput

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2

Name กำหนดเป็น txtOutput
Text ใส่เป็นค่าว่าง
ReadOnly กำหนดเป็น True
Cursor กำหนดเป็น No

กำหนด Properties สำหรับ radioButton1
Name กำหนดเป็น rdoGirth
Text กำหนดเป็น “เส้นรอบวง”

กำหนด Properties สำหรับ radioButton2
Name กำหนดเป็น rdoArea
Text กำหนดเป็น “พื้นที่วงกลม”

กำหนด Properties สำหรับ button1
Name กำหนดเป็น btnCalculate
Text กำหนดเป็น “คำนวณ”

ชุดคำสั่งของโปรแกรมเป็นดังนี้
ส่วนที่เป็นคลาส

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   namespace Ex7_4
6   {
7       public class Cycle
8       {
9           double pi = 3.1415;
10          public double radius { get; set; }
11          public double Girth()
12          {
13              return 2 * pi * radius;
14          }
15          public double Area()
16          {
17              return pi * radius * radius;
18          }
19      }
20  }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 7 – 19 เป็นส่วนการกำหนดคลาสสำหรับการใช้งาน

บรรทัดที่ 9 – 10 เป็นส่วนการกำหนดค่าคุณสมบัติของคลาส

บรรทัดที่ 11 – 18 เป็นส่วนการกำหนดเมธอดสำหรับการคำนวณ

ส่วนที่เป็นฟอร์ม

```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex7_4
10      {
11          public partial class frmCycle : Form
12          {
13              public frmCycle()
14              {
15                  InitializeComponent();
16              }
17              private void btnCalculate_Click(object sender, EventArgs e)
18              {
19                  Cycle _cycle = new Cycle();
20                  _cycle.radius = Convert.ToDouble(txtInput.Text);
21                  if (rdoGirth.Checked == true)
22                      txtOutput.Text = Convert.ToString(_cycle.Girth());
23                  else if (rdoArea.Checked == true)
24                      txtOutput.Text = Convert.ToString(_cycle.Area());
25              }
26          }
27      }
```


อธิบายชุดคำสั่งของโปรแกรม

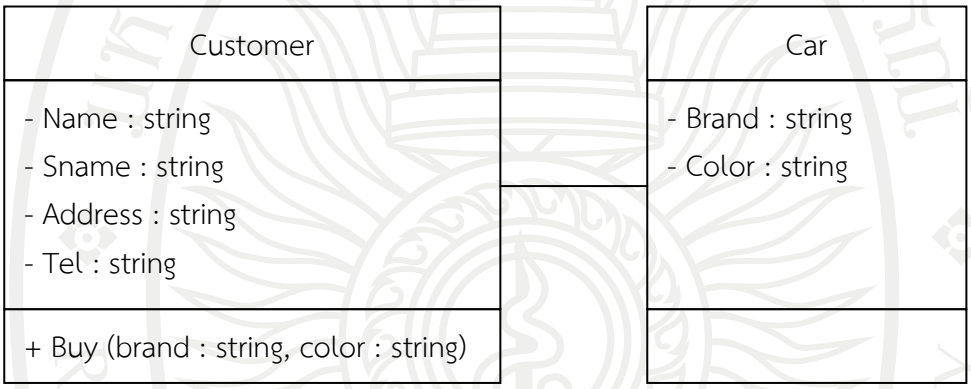
บรรทัดที่ 7 – 19 เป็นส่วนการกำหนดคลาสสำหรับการใช้งาน

บรรทัดที่ 9 – 10 เป็นส่วนการกำหนดค่าคุณสมบัติของคลาส
บรรทัดที่ 11 – 18 เป็นส่วนการกำหนดเมธอดสำหรับการคำนวณ

7.2 ความสัมพันธ์ระหว่างคลาส

การทำงานของระบบเชิงวัตถุนั้น คลาสไม่สามารถทำงานทั้งหมดได้ภายในคลาสเดียว จำเป็นจะต้องมีคลาสอื่น ๆ ที่มีความสัมพันธ์กันเข้ามาร่วมทำงานด้วย ซึ่งสามารถแบ่งความสัมพันธ์ระหว่างคลาสได้แก่ แอสโซสิเอชัน แอกกรีเกชัน คอมโพสิชัน การสืบทอด และการพ้องรูป

7.2.1 แอสโซสิเอชัน เป็นความสัมพันธ์เชิงโครงสร้างที่มีคลาสตั้งแต่ 2 คลาสขึ้นไปแบบโฮล-พาร์ท (ธวัชชัย งามสันติวงศ์, 2549) เช่น ระบบการซื้อรถยนต์ ดังภาพที่ 7.19



ภาพที่ 7.19 ความสัมพันธ์ระหว่างคลาสแบบแอสโซสิเอชัน

ชุดคำสั่งของตัวอย่างเป็นดังนี้

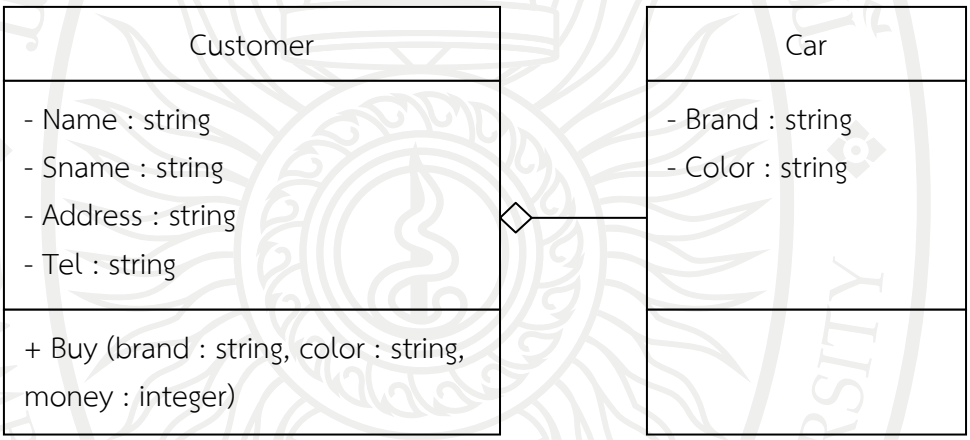
```
1       public class Customer
2       {
3           public string Name { get; set; }
4           public string Sname { get; set; }
5           public string Address { get; set; }
6           public string Tel { get; set; }
7           public void Buy(string brand, string color)
8           {
9               Car car = new Car();
10              car.Brand = brand;
11              car.Color = color;
12          }
13      }
14      public class Car
```

```
15      {
16          public string Brand { get; set; }
17          public string Color { get; set; }
18      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 7 – 12 เป็นส่วนเมธอด Buy ในคลาส Customer จำเป็นจะต้องมีวัตถุ car ที่สร้างมาจากคลาส Car เกิดขึ้น ทำให้การทำงานจะต้องมีความสัมพันธ์กันเป็นโครงสร้าง

7.2.2 แอกกรีเกชัน เป็นความสัมพันธ์เชิงโครงสร้างเช่นเดียวกัน แต่จะมีคลาสหลักและคลาสที่เป็นองค์ประกอบกันแบบโฮล-พาร์ท โดยคลาสหลักจะมีความสำคัญมากกว่าคลาสองค์ประกอบ เช่น การซื้อรถยนต์ต้องมีเงินไม่ต่ำกว่า 5 แสนบาท ดังภาพที่ 7.20



ภาพที่ 7.20 ความสัมพันธ์ระหว่างคลาสแบบแอกกรีเกชัน

ชุดคำสั่งของตัวอย่างเป็นดังนี้

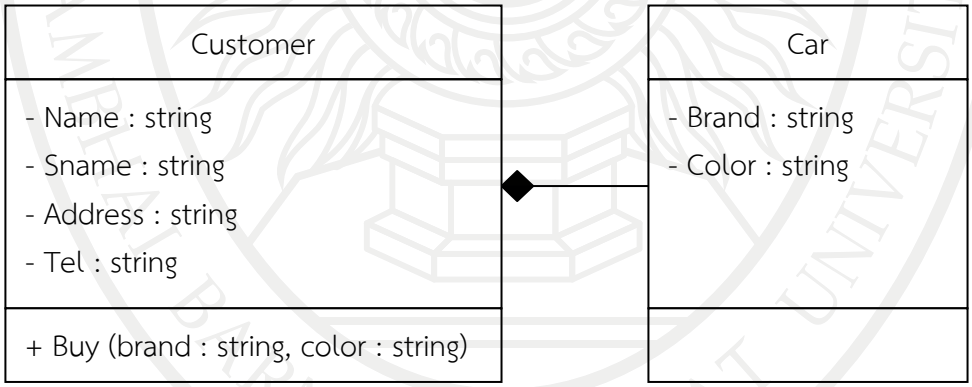
```
1       public class Customer
2       {
3           public string Name { get; set; }
4           public string Sname { get; set; }
5           public string Address { get; set; }
6           public string Tel { get; set; }
7           public void Buy(string brand, string color, int money)
8           {
9               if (money > 500000)
10              {
11                  Car car = new Car();
```

```
12              car.Brand = brand;
13              car.Color = color;
14            }
15        }
16    }
17    public class Car
18    {
19        public string Brand { get; set; }
20        public string Color { get; set; }
21    }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 7 – 16 เป็นส่วนเมธอด Buy ในคลาส Customer จะมีวัตถุ car ที่สร้างมาจากคลาส Car เกิดขึ้นแต่มีเงื่อนไขว่าต้องมีเงินไม่ต่ำกว่า 5 แสนบาท จึงทำให้วัตถุ Car เป็นองค์ประกอบที่อาจจะมีหรือไม่มีก็ได้

7.2.3 คอมโพสิชัน เป็นความสัมพันธ์เชิงโครงสร้างเช่นเดียวกัน แต่จะมีคลาสหลักและคลาสที่เป็นองค์ประกอบแบบโฮล-พาร์ทโดยคลาสหลักจะมีความสำคัญเท่ากับคลาสองค์ประกอบ เช่น ลูกค้าที่ซื้อรถยนต์ต้องมีรถยนต์อยู่แล้ว ดังภาพที่ 7.21



ภาพที่ 7.21 ความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชัน

ชุดคำสั่งของตัวอย่างเป็นดังนี้

```
1     public class Customer
2     {
3         public string Name { get; set; }
4         public string Sname { get; set; }
5         public string Address { get; set; }
```

```
6           public string Tel { get; set; }
7           private Car _car = new Car();
8           public void Buy(string brand, string color)
9           {
10              Car car = new Car();
11              car.Brand = brand;
12              car.Color = color;
13          }
14      }
15      public class Car
16      {
17          public string Brand { get; set; }
18          public string Color { get; set; }
19      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 7 เป็นส่วนของการกำหนดวัตถุ car ที่สร้างมาจากคลาส Car เกิดขึ้น โดยในตัวอย่างนี้วัตถุ car จะถูกสร้างขึ้นพร้อมกับคลาส Customer เสมอ จึงเป็นองค์ประกอบที่เกิดขึ้นพร้อมกัน ไม่สามารถแยกกันได้

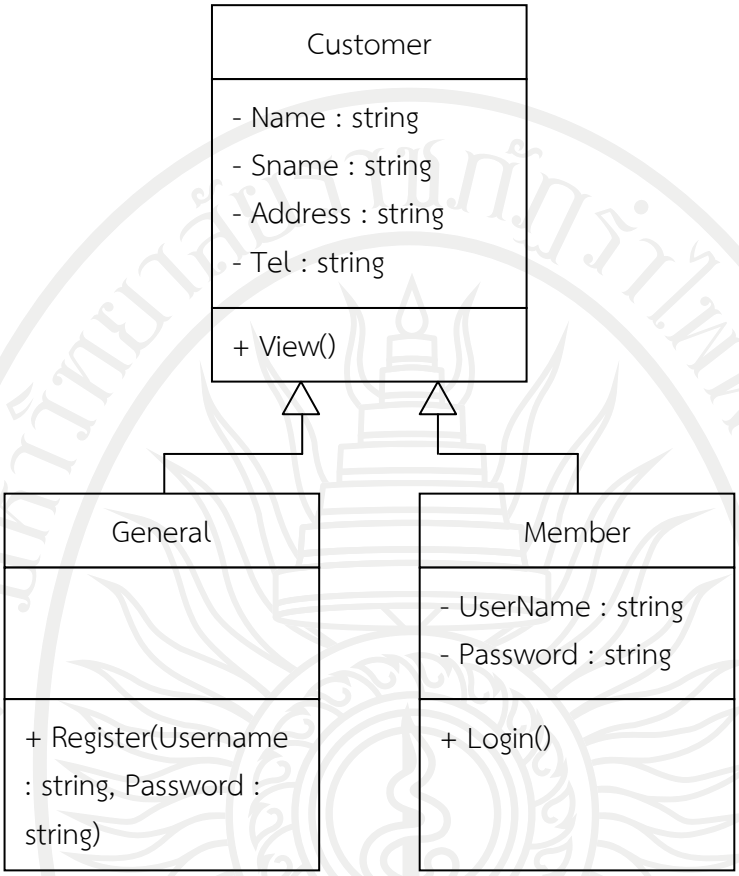
7.2.4 การสืบทอด เป็นความสัมพันธ์ในลักษณะการสืบทอดจากคลาสหนึ่งไปยังอีกคลาสหนึ่ง โดยสามารถสืบทอดคุณสมบัติและพฤติกรรมได้ (กิตติพงษ์ กลมกล่อม, 2552) ดังภาพที่ 7.22 และการสืบทอดในวิชวลซีชาร์ปจะใช้เครื่องหมายโคลอน (:) เพื่อแสดงการสืบทอด โดยมีรูปแบบดังนี้

รูปแบบการสืบทอด

<คลาสที่เป็น sub class> : <คลาสที่เป็น super class>

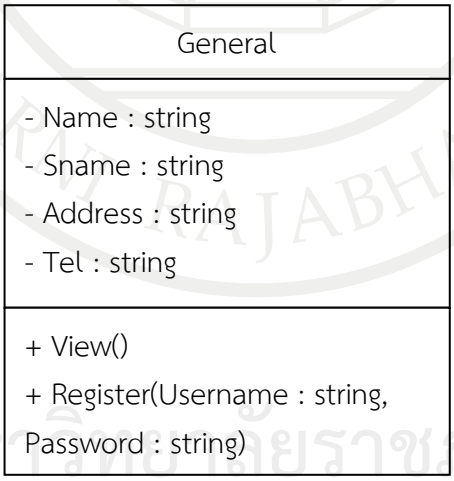
ตัวอย่างเช่น

```
public class Customer
{
}
public class General : Customer
{
}
public class Member : Customer
{
}
```

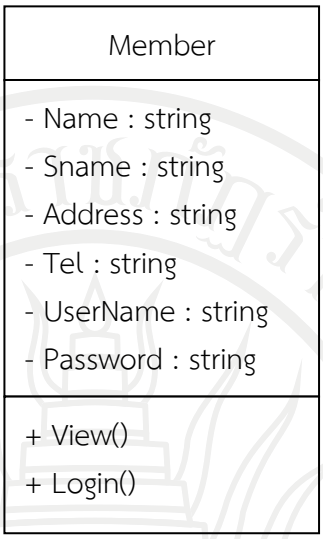


ภาพที่ 7.22 ความสัมพันธ์ระหว่างคลาสแบบการสืบทอด

เมื่อคลาส General และคลาส Member สืบทอดคุณสมบัติมาจากคลาส Customer แล้ว จะมีรายละเอียดของคลาสเป็นดังภาพที่ 7.23 และ 7.24



ภาพที่ 7.23 คลาส General เมื่อผ่านการสืบทอด



ภาพที่ 7.24 คลาส Member เมื่อผ่านการสืบทอด

ชุดคำสั่งของตัวอย่างเป็นดังนี้

```
1       public class Customer
2       {
3           public string Name { get; set; }
4           public string Sname { get; set; }
5           public string Address { get; set; }
6           public string Tel { get; set; }
7           public void View()
8           {
9           }
10      }
11      public class General : Customer
12      {
13          public void Register(string Username, string Password)
14          {
15          }
16      }
17      public class Member : Customer
18      {
19          public string Username { get; set; }
20          public string Password { get; set; }
```

```
21          public void Login()
22          {
23          }
24      }
```

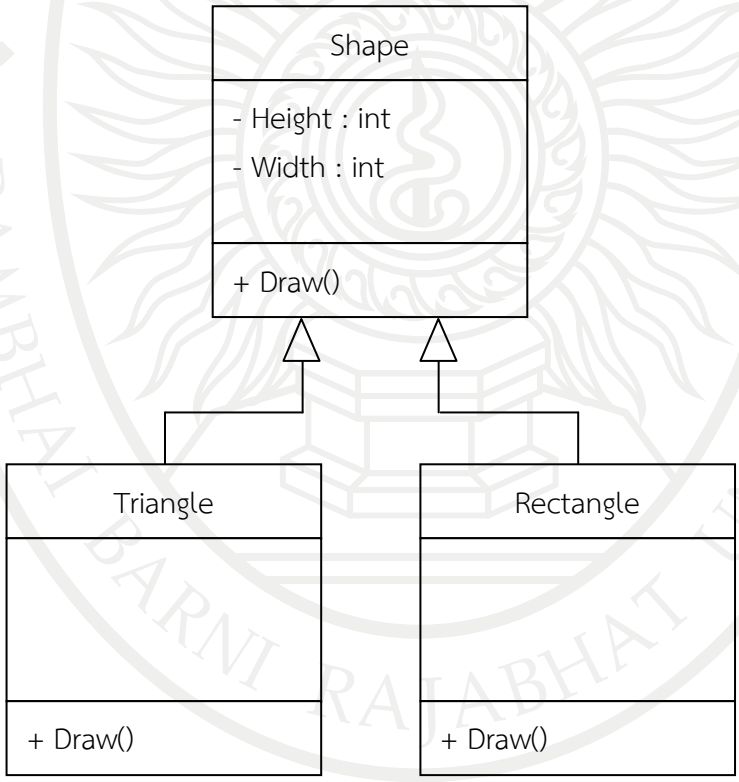
อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 1-10 เป็นส่วนของการกำหนดคลาส Customer

บรรทัดที่ 11 เป็นการกำหนดคลาส General ที่สืบทอดมาจากคลาส Customer

บรรทัดที่ 17 เป็นการกำหนดคลาส Member ที่สืบทอดมาจากคลาส Customer

7.2.5 การพ้องรูป เป็นการถ่ายทอดคุณสมบัติของการทำงาน ซึ่งในคลาสแม่และคลาสลูกจะไม่เหมือนกัน เนื่องจากมีความแตกต่างในเรื่องของพฤติกรรมการทำงาน ทำให้ถ่ายทอดมาเฉพาะส่วนของโครงสร้างเท่านั้น การพ้องรูปในวิชวลซีชาร์ปจะมีความหมายดังภาพที่ 7.25 ส่วนการใช้คำสั่งในการทำงาน จะใช้คำสั่ง override ในการทำงาน



ภาพที่ 7.25 ความสัมพันธ์ระหว่างคลาสแบบการพ้องรูป

รูปแบบ

<accessibility> virtual <data type> <Method ใน super class>

<accessibility> override <data type> <Method ใน subclass>

ตัวอย่าง

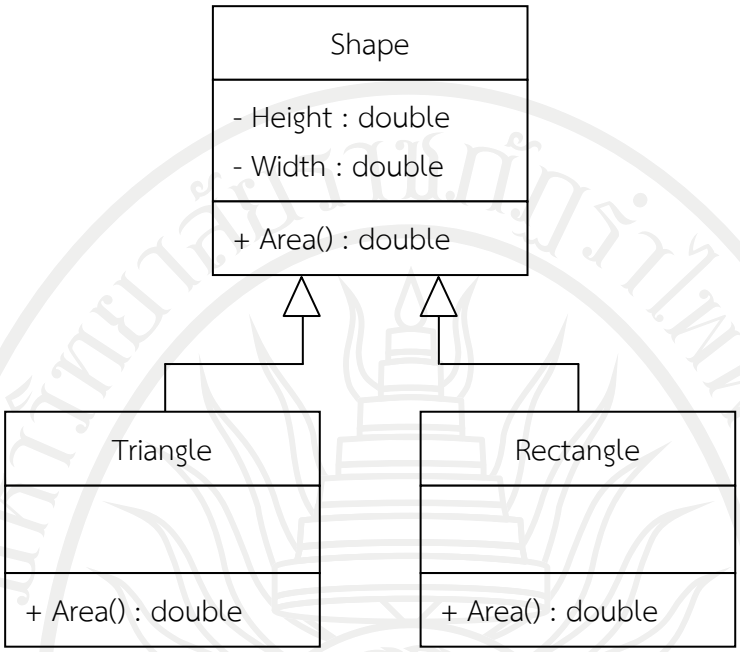
```
1   public class Shape
2   {
3       public int Height { get; set; }
4       public int Width { get; set; }
5       public virtual void Draw()
6       {
7           MessageBox.Show(“Complete”);
8       }
9   }
10  class Triangle : Shape
11  {
12      public override void Draw()
13      {
14          MessageBox.Show(“Draw Triangle”);
15          base.Draw();
16      }
17  }
18  class Rectangle : Shape
19  {
20      public override void Draw()
21      {
22          MessageBox.Show(“Draw Rectangle”);
23          base.Draw();
24      }
25  }
```

อธิบายชุดคำสั่งของโปรแกรม

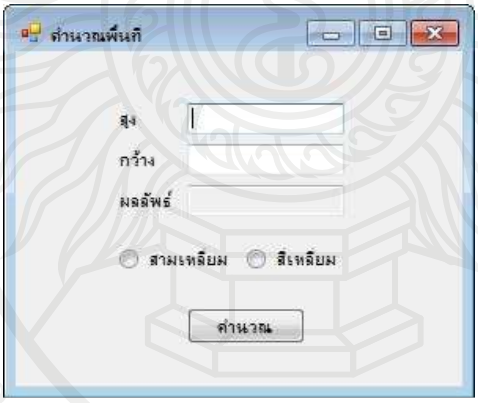
บรรทัดที่ 5 เป็นเมธอดต้นแบบของการพ้องรูป

บรรทัดที่ 12 และ 20 เป็นเมธอดที่เกิดการพ้องรูป

ตัวอย่างที่ 7.5 จงเขียนโปรแกรมคำนวณพื้นที่สามเหลี่ยมและสี่เหลี่ยมโดยใช้รูปแบบการพ้องรูป ซึ่งมีคลาสไดอะแกรมดังภาพที่ 7.26 และมีหน้าจอดังภาพที่ 7.27



ภาพที่ 7.26 คลาสตัวอย่างการหาพื้นที่แบบการพ้องรูป



ภาพที่ 7.27 หน้าจอตัวอย่างการหาพื้นที่แบบการพ้องรูป

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmArea

Text กำหนดเป็น “คำนวณพื้นที่”

กำหนด Properties สำหรับ label1

Name กำหนดเป็น lblHeight

Text กำหนดเป็น “สูง”

กำหนด Properties สำหรับ label2

Name กำหนดเป็น lblWidth

Text กำหนดเป็น “กว้าง”

กำหนด Properties สำหรับ label3

Name กำหนดเป็น lblOutput

Text กำหนดเป็น “ผลลัพธ์”

กำหนด Properties สำหรับ textBox1

Name กำหนดเป็น txtHeight

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2

Name กำหนดเป็น txtWidth

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox3

Name กำหนดเป็น txtOutput

Text ใส่เป็นค่าว่าง

ReadOnly กำหนดเป็น True

Cursor กำหนดเป็น No

กำหนด Properties สำหรับ radioButton1

Name กำหนดเป็น rdoTriangle

Text กำหนดเป็น “สามเหลี่ยม”

กำหนด Properties สำหรับ radioButton2

Name กำหนดเป็น rdoRectangle

Text กำหนดเป็น “สี่เหลี่ยม”

กำหนด Properties สำหรับ button1

Name กำหนดเป็น btnCalculate

Text กำหนดเป็น “คำนวณ”

ชุดคำสั่งของตัวอย่างเป็นดังนี้

ส่วนที่เป็นคลาส

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   namespace Ex7_5
6   {
7     class Shape
8       {
9           public double Height { get; set; }
```

```
10              public double Width { get; set; }
11              public virtual double Area()
12              {
13                  return Height * Width;
14              }
15          }
16          class Triangle : Shape
17          {
18              public override double Area()
19              {
20                  return 0.5 * base.Area();
21              }
22          }
23          class Rectangle : Shape
24          {
25              public override double Area()
26              {
27                  return base.Area();
28              }
29          }
30      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 11 เป็นเมธอดต้นแบบของการพ้องรูป

บรรทัดที่ 18 เป็นเมธอดที่เกิดการพ้องรูปในคลาส Triangle

บรรทัดที่ 25 เป็นเมธอดที่เกิดการพ้องรูปในคลาส Rectangle

ส่วนที่เป็นฟอร์ม

```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
```

```
9       namespace Ex7_5
10      {
11          public partial class frmArea : Form
12          {
13              public frmArea()
14              {
15                  InitializeComponent();
16              }
17              private void btnCalculate_Click(object sender, EventArgs e)
18              {
19                  Triangle tri = new Triangle();
20                  Rectangle rect =new Rectangle();
21                  if (rdoTriangle.Checked == true)
22                  {
23                      tri.Height = Convert.ToDouble(txtHeight.Text);
24                      tri.Width = Convert.ToDouble(txtWidth.Text);
25                      txtOutput.Text = Convert.ToString(tri.Area());
26                  }
27                  else if (rdoRectangle.Checked == true)
28                  {
29                      rect.Height = Convert.ToDouble(txtHeight.Text);
30                      rect.Width = Convert.ToDouble(txtWidth.Text);
31                      txtOutput.Text = Convert.ToString(rect.Area());
32                  }
33              }
34          }
35      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 19 เป็นการสร้างวัตถุ tri จากคลาส Triangle

บรรทัดที่ 20 เป็นการสร้างวัตถุ rect จากคลาส Rectangle

บรรทัดที่ 25 เป็นการเรียกใช้เมธอดที่เกิดการพ้องรูปในคลาส Triangle

บรรทัดที่ 31 เป็นการเรียกใช้เมธอดที่เกิดการพ้องรูปในคลาส Rectangle

7.3 การใช้งานดีไซน์แพทเทิร์น

ในการเขียนโปรแกรมเชิงวัตถุอาจจะพบเจอปัญหาที่มีความซับซ้อนเป็นอย่างมาก ทำให้การออกแบบและการเขียนโปรแกรมอาจจะมีข้อผิดพลาด ถ้าเป็นการทำงานที่มีทีมงานทำงานร่วมกัน แต่ละคนอาจจะมีทักษะในการออกแบบและเขียนโปรแกรมแตกต่างกันออกไป ทำให้เกิดปัญหาในเรื่องของรูปแบบในการพัฒนาโปรแกรม

ดีไซน์แพทเทิร์น (Design pattern) เป็นการใช้งานรูปแบบสำเร็จรูปที่ถูกพัฒนาขึ้นเพื่อใช้ในการแก้ปัญหาที่มีลักษณะคล้ายกัน โดยรูปแบบที่ใช้เป็นลักษณะเชิงวัตถุ ผู้พัฒนาโปรแกรมสามารถแก้ไขปัญหาที่พบและใช้รูปแบบที่เหมาะสมเพื่อในการพัฒนาโปรแกรม รูปแบบของดีไซน์แพทเทิร์นสามารถแบ่งออกเป็น 3 กลุ่มใหญ่ ๆ ได้ดังนี้ (Dofactory, 2016)

7.3.1 ครีเอชันนัลแพทเทิร์น (Creational Patterns) เป็นรูปแบบที่ใช้สร้างออบเจ็กต์ที่สามารถสร้างออบเจ็กต์อื่น ๆ ขึ้นมาได้ระหว่างที่กำลังทำงานอยู่ ซึ่งแพทเทิร์นในกลุ่มนี้ ได้แก่

1) แอบสแตกแฟคทอรี (Abstract Factory)
2) บิลด์เดอร์ (Builder)
3) แฟคทอรีเมธอด (Factory Method)
4) โปรโตไทป์ (Prototype)
5) ซิงเกิลตอน (Singleton)

7.3.2 สตรัคเจอร์รอลแพทเทิร์น (Structural Patterns) เป็นรูปแบบที่ทำให้ออบเจ็กต์ที่มีอยู่สามารถทำงานร่วมกันได้ สำหรับแพทเทิร์นในกลุ่มนี้ ได้แก่

1) อะแดปเตอร์ (Adapter)
2) บริดก์ (Bridge)
3) คอมโพสิท (Composite)
4) ดิโคเรเตอร์ (Decorator)
5) แฟเคด (Facade)
6) ฟลายเวท (Flyweight)
7) พรอกซี (Proxy)

7.3.3 บีฮาฟวิโอรอลแพทเทิร์น (Behavioral Patterns) เป็นรูปแบบที่ใช้ในการควบคุมการส่งข้อมูลระหว่างออบเจ็กต์ด้วยกัน สำหรับแพทเทิร์นในกลุ่มนี้ ได้แก่

1) เชนออฟเรสพ์ (Chain of Resp.)
2) คอมมานด์ (Command)
3) อินเทอร์พรีเตอร์ (Interpreter)
4) อิเทเรเตอร์ (Iterator)
5) มิเดียเตอร์ (Mediator)
6) เมเมนโต (Memento)
7) ออบเสิร์บเวอร์ (Observer)
8) สเตท (State)

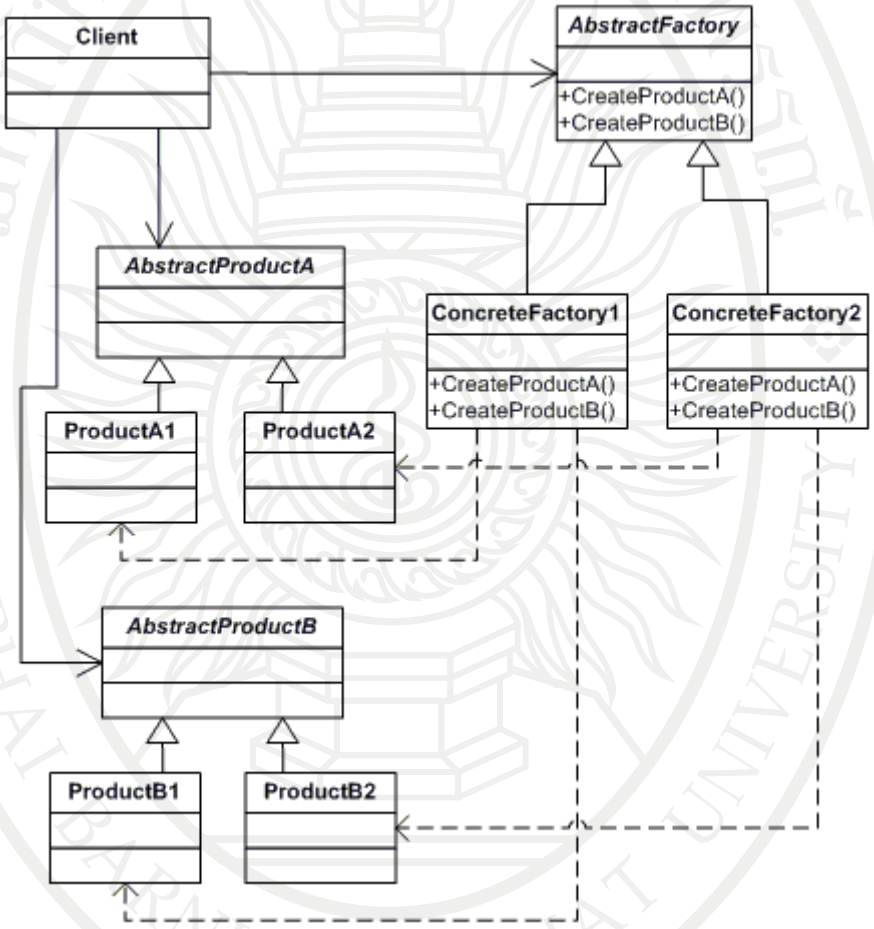
9) สเตรทิจี (Strategy)

10) เทมเพลทเมธอด (Template Method)

11) วิสิเตอร์ (Visitor)

สำหรับตัวอย่างการใช้ดีไซน์แพทเทิร์นที่สามารถพบได้บ่อยในการใช้ทั่วไป เช่น แอบสแตกแฟคทอรีและซิงเกิลตอน จะมีการเขียนชุดคำสั่งดังนี้

ตัวอย่างโครงสร้างคลาสของแอบสแตกแฟคทอรี ดังภาพที่ 7.28



ภาพที่ 7.28 คลาสของแอบสแตกแฟคทอรี
ที่มา : (dofactory, 2016)

ชุดคำสั่งตัวอย่างของแอบสแตกแฟคทอรี
ส่วนที่เป็นคลาส

```
1    using System;
2    using System.Collections.Generic;
3    using System.Linq;
```

```
4       using System.Text;
5       using System.Windows.Forms;
6       namespace Ex7
7       {
8           abstract class AbstractFactory
9           {
10              public abstract AbstractProductA CreateProductA();
11              public abstract AbstractProductB CreateProductB();
12          }
13          /// The 'ConcreteFactory1' class
14          class ConcreteFactory1 : AbstractFactory
15          {
16            public override AbstractProductA CreateProductA()
17            {
18              return new ProductA1();
19            }
20            public override AbstractProductB CreateProductB()
21            {
22              return new ProductB1();
23            }
24          }
25          /// The 'ConcreteFactory2' class
26          class ConcreteFactory2 : AbstractFactory
27          {
28            public override AbstractProductA CreateProductA()
29            {
30              return new ProductA2();
31            }
32            public override AbstractProductB CreateProductB()
33            {
34              return new ProductB2();
35            }
36          }
37          /// The 'AbstractProductA' abstract class
38          abstract class AbstractProductA
39          {
```

```
40        }
41         /// The 'AbstractProductB' abstract class
42        abstract class AbstractProductB
43        {
44          public abstract void Interact(AbstractProductA a);
45        }
46
47        /// The 'ProductA1' class
48        class ProductA1 : AbstractProductA
49        {
50        }
51        /// The 'ProductB1' class
52        class ProductB1 : AbstractProductB
53        {
54          public override void Interact(AbstractProductA a)
55          {
56              MessageBox.Show(this.GetType().Name + " interacts with " +
a.GetType().Name);
57          }
58        }
59        /// The 'ProductA2' class
60        class ProductA2 : AbstractProductA
61        {
62        }
63        /// The 'ProductB2' class
64        class ProductB2 : AbstractProductB
65        {
66          public override void Interact(AbstractProductA a)
67          {
68              MessageBox.Show(this.GetType().Name + " interacts with " +
a.GetType().Name);
69          }
70        }
71        /// The 'Client' class. Interaction environment for the products.
72        class Client
73        {
```

```
74          private AbstractProductA _abstractProductA;
75          private AbstractProductB _abstractProductB;
76          // Constructor
77          public Client(AbstractFactory factory)
78          {
79            _abstractProductB = factory.CreateProductB();
80            _abstractProductA = factory.CreateProductA();
81          }
82          public void Run()
83          {
84            _abstractProductB.Interact(_abstractProductA);
85          }
86        }
87      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 14 – 24 เป็นคลาส ConcreteFactory1 สำหรับการสร้าง ProductA1 และ ProductB1

บรรทัดที่ 26 – 36 เป็นคลาส ConcreteFactory2 สำหรับการสร้าง ProductA2 และ ProductB2

บรรทัดที่ 48 เป็นคลาส ProductA1 ที่สืบทอดมาจากคลาส AbstractProductA

บรรทัดที่ 52 เป็นคลาส ProductB1 ที่สืบทอดมาจากคลาส AbstractProductB

บรรทัดที่ 72 เป็นคลาส Client ที่สั่งให้ AbstractFactory, AbstractProductA และ AbstractProductB ทำงาน

ส่วนควบคุมที่เป็นฟอร์ม

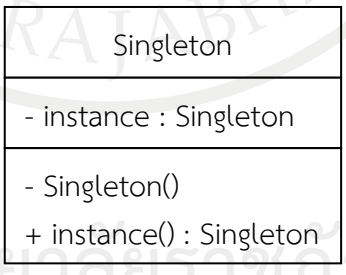
```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex7
10      {
```

```
11          public partial class frmAbstractFactory : Form
12          {
13              public frmAbstractFactory()
14              {
15                  InitializeComponent();
16              }
17              private void btnAbs_Click(object sender, EventArgs e)
18              {
19                  // Abstract factory #1
20                  AbstractFactory factory1 = new ConcreteFactory1();
21                  Client client1 = new Client(factory1);
22                  client1.Run();
23                  // Abstract factory #2
24                  AbstractFactory factory2 = new ConcreteFactory2();
25                  Client client2 = new Client(factory2);
26                  client2.Run();
27              }
28          }
29      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 20 เป็นการสร้างวัตถุ factory1 จากคลาส ConcreteFactory1
บรรทัดที่ 21 เป็นการสร้างวัตถุ client1 จากคลาส Client โดยส่ง factory1 เข้าไป
บรรทัดที่ 24 เป็นการสร้างวัตถุ factory2 จากคลาส ConcreteFactory2
บรรทัดที่ 25 เป็นการสร้างวัตถุ client2 จากคลาส Client โดยส่ง factory2 เข้าไป

ตัวอย่างโครงสร้างคลาสของซิงเกิลตอน ดังภาพที่ 7.29



ภาพที่ 7.29 คลาสของซิงเกตัน

ชุดคำสั่งตัวอย่างของซิงเกิลตอน

ส่วนที่เป็นคลาส

```
1       using System;
2       using System.Collections.Generic;
3       using System.Linq;
4       using System.Text;
5       namespace Ex7
6       {
7           class Singleton
8           {
9               private static Singleton _instance;
10              // Constructor is 'protected'
11              protected Singleton()
12              {
13              }
14              public static Singleton Instance()
15              {
16                  // Uses lazy initialization.
17                  // Note: this is not thread safe.
18                  if (_instance == null)
19                  {
20                      _instance = new Singleton();
21                  }
22                  return _instance;
23              }
24          }
25      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 7 – 25 เป็นการกำหนดคลาส Singleton

บรรทัดที่ 11 เป็นการเมธอด Singleton ให้มีการเข้าถึงแบบ protected

บรรทัดที่ 14 เป็นการกำหนดเมธอด Instance ให้มีการคืนค่าแบบ Singleton

ส่วนควบคุมที่เป็นฟอร์ม

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
```

```
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   namespace Ex7
10  {
11      public partial class frmSingleton : Form
12      {
13          public frmSingleton()
14          {
15              InitializeComponent();
16          }
17          private void btnSng_Click(object sender, EventArgs e)
18          {
19              // Constructor is protected -- cannot use new
20              Singleton s1 = Singleton.Instance();
21              Singleton s2 = Singleton.Instance();
22              // Test for same instance
23              if (s1 == s2)
24              {
25                  MessageBox.Show("Objects are the same instance");
26              }
27          }
28      }
29  }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 20 เป็นการกำหนด s1 ให้มีชนิดข้อมูลเป็นแบบ Singleton และมีค่าเท่ากับการคืนค่าของเมธอด Instance ในคลาส Singleton

บรรทัดที่ 21 เป็นการกำหนด s2 ให้มีชนิดข้อมูลเป็นแบบ Singleton และมีค่าเท่ากับการคืนค่าของเมธอด Instance ในคลาส Singleton

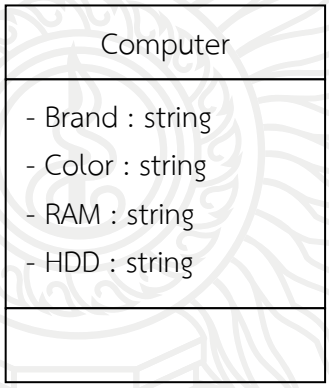
บรรทัดที่ 23 - 26 เป็นการทดสอบค่าที่ได้จาก Instance ในคลาส Singleton

7.4 สรุป

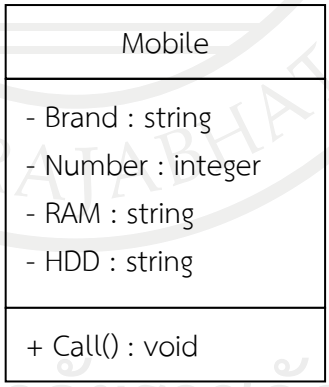
การเขียนโปรแกรมเชิงวัตถุนั้นจะมีคลาสเป็นต้นแบบที่ใช้สำหรับการทำงาน แต่ยังไม่สามารถนำคลาสเหล่านั้นมาทำงานได้โดยตรง จำเป็นจะต้องสร้างวัตถุขึ้นมาจากคลาสเสียก่อนด้วยคำสั่ง new โดยวัตถุที่สร้างขึ้นมาจากคลาสจะมีคุณสมบัติและพฤติกรรมเหมือนกับคลาสต้นแบบ ส่วนที่เป็นคุณสมบัติในคลาสจำเป็นจะต้องมีการป้องกันไม่ใช้คลาสจากภายนอกเข้ามาเปลี่ยนแปลงแก้ไขข้อมูลได้โดยตรง โดยส่วนใหญ่คุณสมบัติในคลาสจะมีการตั้งค่าการเข้าถึงข้อมูลเป็นแบบ private อยู่แล้ว แต่ในกรณีที่ต้องการแก้ไขข้อมูล สามารถใช้บริการโดยผ่านทางเมธอด get และ set ซึ่งเป็นมาตรฐานที่ใช้สำหรับการเขียนโปรแกรมเชิงวัตถุ นอกจากนี้ในระบบเชิงวัตถุยังต้องคำนึงถึงความสัมพันธ์ระหว่างคลาสในระบบด้วย เช่น การสืบทอดเป็นความสัมพันธ์ในลักษณะการสืบทอดจากคลาสหนึ่งไปยังอีกคลาสหนึ่งโดยสามารถสืบทอดคุณสมบัติและพฤติกรรมได้ และการพ้องรูปเป็นการถ่ายทอดคุณสมบัติของการทำงาน ซึ่งในคลาสแม่และคลาสลูกจะไม่เหมือนกัน เนื่องจากมีความแตกต่างในเรื่องของพฤติกรรมการทำงาน ทำให้ถ่ายทอดมาเฉพาะส่วนของโครงสร้างเท่านั้น ทำให้การออกแบบมีความซับซ้อนเป็นอย่างยิ่งซึ่งผู้พัฒนาโปรแกรมสามารถแก้ไขปัญหานี้ได้โดยการใช้ดีไซน์แพทเทิร์น ซึ่งเป็นรูปแบบมาตรฐาน ใช้สำหรับจัดการปัญหาความซับซ้อน และปัญหาโครงสร้างของการเขียนโปรแกรมเชิงวัตถุ

แบบฝึกหัดบทที่ 7

1. จงบอกรูปแบบของคำสั่งที่ใช้ในการสร้างวัตถุจากคลาส
2. คลาสไดอะแกรมสามารถแปลงเป็นอะไรได้บ้างในการเขียนโปรแกรม
3. จงอธิบายการทำงานของคำสั่ง get และ set
4. ความสัมพันธ์ระหว่างแอกกรีเกชัน (Aggregation) และคอมโพสิชัน (Composition) ต่างกันอย่างไร
5. จงเขียนคลาสแสดงความสัมพันธ์แบบเจนเนอเรไลเซชัน (Generalization) ระหว่าง รถยนต์ รถบรรทุก รถโดยสาร
6. ดีไซน์แพทเทิร์น คืออะไร
7. การใช้งานดีไซน์แพทเทิร์นแตกต่างจากการสร้างคลาสเองอย่างไร
8. ดีไซน์แพทเทิร์นสามารถแบ่งออกเป็นกี่กลุ่ม อะไรบ้าง
9. จงแปลงคลาสไดอะแกรมให้เป็นคำสั่งในโปรแกรม



10. จงเขียนคำสั่งในการสร้างวัตถุจากคลาสต่อไปนี้



เอกสารอ้างอิง

กิตติพงษ์ กลมกล่อม. (2552). **การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML.** กรุงเทพฯ : เคทีพี.

ธวัชชัย งามสันติวงศ์. (2549). **การวิเคราะห์และออกแบบระบบงานเชิงวัตถุ.** กรุงเทพฯ : ศูนย์หนังสือจุฬาลงกรณ์มหาวิทยาลัย.

ธีระพล ลิ้มศรัทธา. (2553). **การเขียนโปรแกรมเชิงวัตถุด้วย Visual Basic.NET**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

Dofactory. (2016). **.NET Design Patterns**. (online). Available : http://www.dofactory.com/net/design-patterns. 20 March 2016.

แผนบริหารประจำบทที่ 8

เนื้อหา

บทที่ 8 การจัดการกับสิ่งผิดปกติ

8.1 ประเภทของความผิดพลาด
8.2 การตรวจสอบข้อผิดพลาด
8.3 เครื่องมือสำหรับตรวจสอบข้อผิดพลาด
8.4 การจัดการข้อผิดพลาด
8.5 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. อธิบายประเภทของความผิดพลาดที่เกิดขึ้นได้
2. ตรวจสอบและแก้ไขข้อผิดพลาดที่เกิดขึ้นได้
3. ใช้วิธีการตรวจสอบข้อผิดพลาดอย่างเหมาะสม
4. จัดการกับข้อผิดพลาดที่เกิดขึ้นได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน และค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ผู้สอนเขียนโปรแกรมตัวอย่างและอธิบายให้กับผู้เรียน
5. ผู้สอนสุ่มถามความเข้าใจจากผู้เรียน
6. ให้ผู้เรียนลองอธิบายเนื้อหาตามที่เข้าใจ
7. ให้ผู้เรียนฝึกปฏิบัติ
8. ให้ผู้เรียนทำแบบฝึกหัดบทที่ 8

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 8
4. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. สังเกตจากการร่วมกิจกรรมของผู้เรียน
3. ประเมินจากการสุ่มถามผู้เรียน
4. ประเมินจากแบบฝึกหัดบทที่ 8
5. ประเมินจากการสอบปลายภาค

บทที่ 8
การจัดการกับสิ่งผิดปกติ

ข้อผิดพลาดในการพัฒนาโปรแกรมสามารถเกิดขึ้นได้ตลอดเวลาเรียกได้ว่าเป็นของคู่กัน เพราะไม่มีโปรแกรมใดที่ไม่มีข้อผิดพลาดทั้งระหว่างการพัฒนาหรือหลังจากนำไปใช้งาน เมื่อเกิดข้อผิดพลาดแล้วสิ่งที่จำเป็นคือต้องค้นหาและแก้ไขข้อผิดพลาดเพื่อไม่ให้มีผลกระทบกับการทำงานของโปรแกรม และในการพัฒนาโปรแกรมด้วยวิชวลซีชาร์ปอาจเกิดข้อผิดพลาดในระหว่างพัฒนาที่เกิดขึ้นจะมีระดับของความรุนแรงแตกต่างกันไป อาจเกิดตอนที่พัฒนาโปรแกรมหรือเกิดในระหว่างการสั่งให้โปรแกรมทำงานก็ได้ ซึ่งผู้พัฒนาสามารถใช้เครื่องมือที่วิชวลซีชาร์ปมีให้จัดการกับข้อผิดพลาดเหล่านั้น

8.1 ประเภทของความผิดพลาด

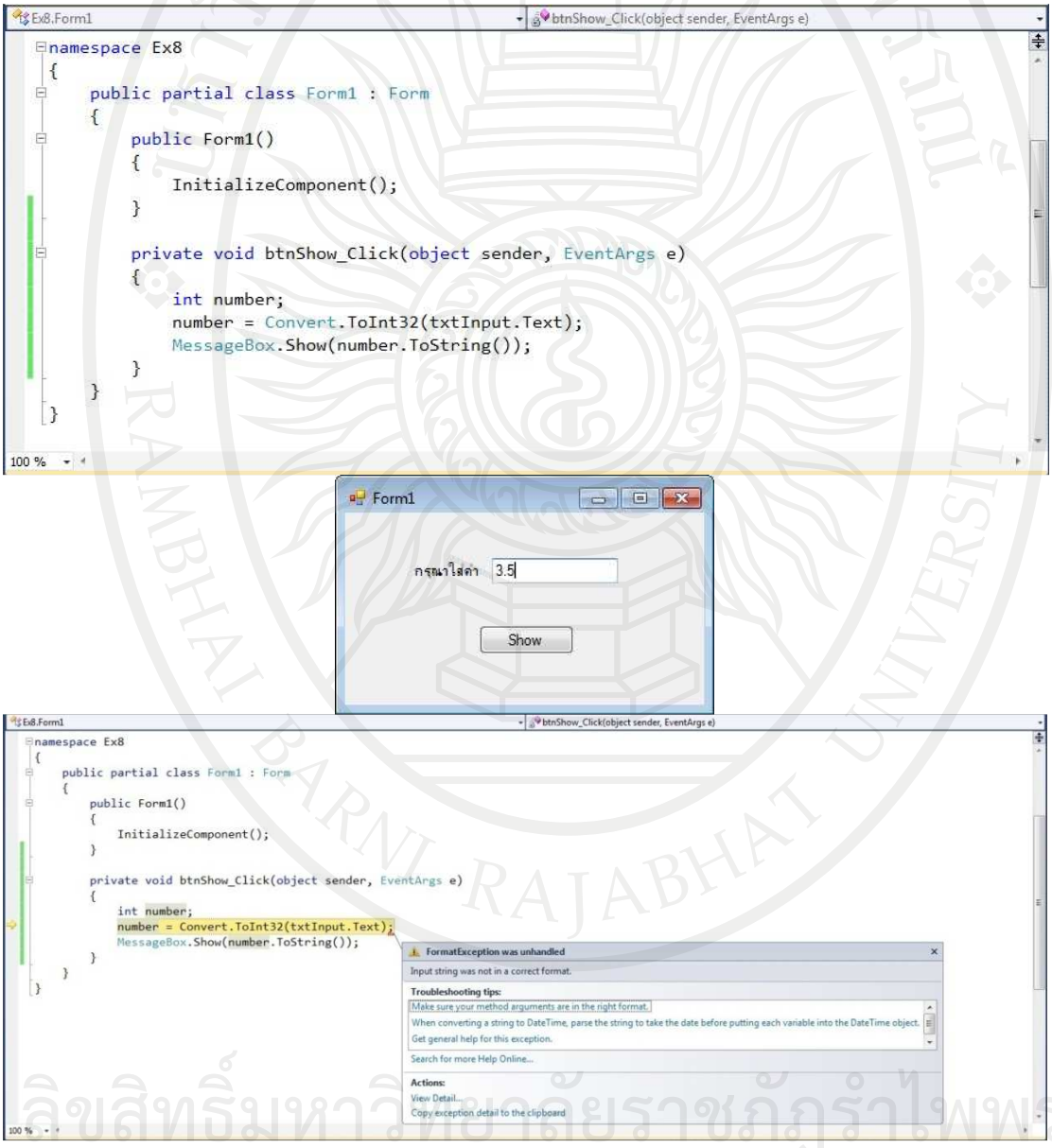
การแบ่งประเภทของข้อผิดพลาดที่เกิดขึ้น สามารถแบ่งตามลักษณะการเกิดและผลที่เกิดขึ้น ซึ่งโดยทั่วไปสามารถแบ่งออกเป็น 3 ประเภทหลัก ๆ ดังนี้ (ศุภชัย สมพานิช, 2556)

8.1.1 ข้อผิดพลาดทางไวยกรณ์ (Syntax error) เป็นข้อผิดพลาดที่เกิดขึ้นจากการพิมพ์คำสั่งผิดหลักไวยากรณ์ของภาษา สามารถพบได้ในระหว่างการพัฒนาโปรแกรม ซึ่งโปรแกรมจะแสดงผลให้ทราบโดยทันที เนื่องจากมีการตรวจสอบอยู่ตลอดเวลา ซึ่งจะมีเส้นหยักสีแดงปรากฏขึ้นมาดังภาพที่ 8.1

ภาพที่ 8.1 การแสดงข้อผิดพลาดทางไวยกรณ์

จากตัวอย่างข้างต้นจะเห็นว่าข้อผิดพลาดที่แสดงขึ้นมานั้นเกิดเนื่องจากการใช้คำสั่ง MessageBox.Show จะไม่มีตัวอักษร s ต่อท้าย และการประกาศตัวแปร number เป็น int ซึ่งเป็นจำนวนเต็ม เมื่อใส่ค่าที่เป็นจำนวนจริงก็จะเกิดข้อผิดพลาดขึ้นมา

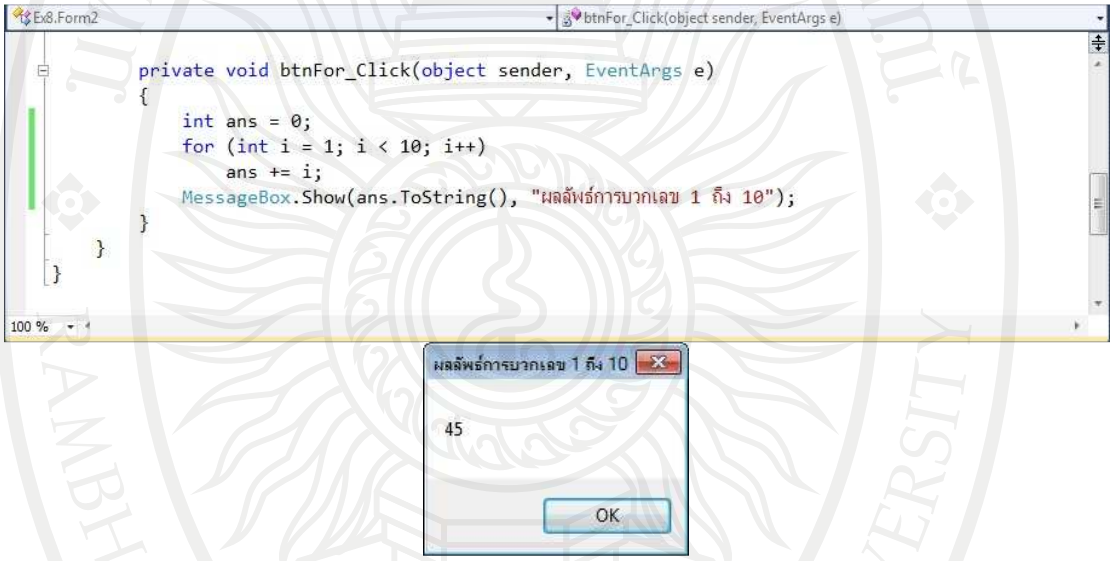
8.1.2 ข้อผิดพลาดระหว่างการทำงาน (Runtime error) เป็นข้อผิดพลาดที่เกิดขึ้นในขณะที่โปรแกรมกำลังปฏิบัติงาน (Execute) เป็นความผิดพลาดที่ทำให้เกิดความผิดปกติในระหว่างการทำงานของโปรแกรม เช่น มีการป้อนข้อมูลไม่ตรงกับชนิดตัวแปรที่กำหนด หรือทรัพยากรไม่เพียงพอสำหรับการทำงานของโปรแกรม ข้อผิดพลาดประเภทนี้ถ้าไม่เขียนคำสั่งป้องกันไว้ เมื่อเกิดข้อผิดพลาดจะทำให้โปรแกรมหยุดการทำงานกลางคัน (นิรันดร์ ประวิทย์ธนา, 2545) ดังภาพที่ 8.2



ภาพที่ 8.2 การแสดงข้อผิดพลาดระหว่างการทำงาน

จากตัวอย่างข้างตนจะเห็นว่าในการกำหนดคำสั่งในโปรแกรมจะกำหนดตัวแปร number เป็นชนิดข้อมูลแบบ int แต่ค่าที่รับตอนโปรแกรมทำงานเป็นค่าที่มีทศนิยม ทำให้เกิดข้อผิดพลาดขึ้น

8.1.3 ข้อผิดพลาดทางตรรกะ (Logical error) เป็นข้อผิดพลาดที่ทำให้ไม่ได้ผลลัพธ์ตามความต้องการของโปรแกรม เกิดจากการเขียนโปรแกรมที่มีข้อผิดพลาดทางตรรกะ ถึงแม้ว่าโปรแกรมจะไม่เกิดข้อผิดพลาดอื่นแต่ได้ผลลัพธ์ไม่เป็นไปตามต้องการ ข้อผิดพลาดประเภทนี้เป็นข้อผิดพลาดที่มีความร้ายแรงและเป็นประเภทที่ตรวจพบยากที่สุด เนื่องจากไม่มีเครื่องมือที่จะใช้ตรวจสอบข้อผิดพลาดได้เหมือนกับ 2 แบบแรก การตรวจสอบจึงต้องอาศัยความรอบคอบและประสบการณ์ของผู้เขียนโปรแกรม ตัวอย่างเช่น ต้องการให้โปรแกรมบวกเลข 1 ถึง 10 โดยใช้คำสั่ง for ตามภาพที่ 8.3



ภาพที่ 8.3 การแสดงข้อผิดพลาดทางตรรกะ

จากตัวอย่างข้างต้นจะเห็นว่าโปรแกรมต้องการบวกเลข 1 ถึง 10 แต่เมื่อโปรแกรมทำงานกลับบวกเลขตั้งแต่ 1 ถึง 9 ซึ่งเป็นความผิดพลาดในการกำหนดเงื่อนไขในคำสั่ง for ซึ่งการเขียนคำสั่งที่ถูกต้องจะต้องกำหนดให้ i มีค่าน้อยกว่าหรือเท่ากับ 10

8.2 การตรวจสอบข้อผิดพลาด

การทดสอบโปรแกรมที่สร้างขึ้นก่อนนำไปใช้งานจริงเป็นส่วนหนึ่งของการทดสอบระบบ เพราะโปรแกรมที่สร้างขึ้นนั้นอาจจะมีบางส่วนที่เป็นข้อผิดพลาดที่ไม่พบในขณะเขียนโปรแกรมหรือที่เรียกว่า ข้อผิดพลาดทางตรรกะ ทำให้ไม่สามารถตรวจสอบได้ว่าข้อผิดพลาดที่เกิดขึ้นมาจากส่วนใด และถ้าโปรแกรมที่สร้างขึ้นมีหลายฟังก์ชันหรือมีขนาดใหญ่มาก จะทำให้การค้นหาข้อผิดพลาดยากขึ้นไปอีก ดังนั้นการจะให้แน่ใจว่าโปรแกรมสามารถทำงานได้ตามที่ต้องการได้จึงต้องมีการตรวจสอบที่ดีเพื่อหาข้อผิดพลาดที่เกิดขึ้น (น้ำฝน อัศวเมฆิน, 2558)

8.2.1 การทดสอบคุณภาพของโปรแกรม

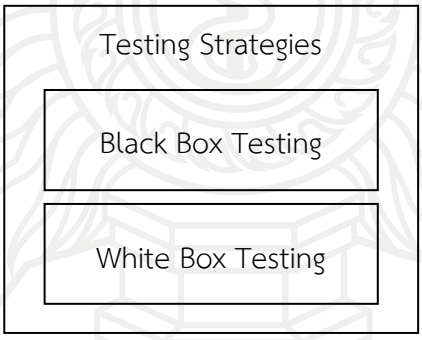
สำหรับการตรวจสอบโปรแกรมเพื่อรับประกันความถูกต้องตามหลักของวิศวกรรมซอฟต์แวร์นั้น สามารถแบ่งการตรวจสอบออกเป็น 2 ด้าน คือ

8.2.1.1 การทวนสอบ (Verification) คือ การตรวจสอบความถูกต้องของโปรแกรมโดยพิจารณาจากรายละเอียดของโปรแกรมที่พัฒนาขึ้นมา ซึ่งจะต้องตรงกับความต้องการของโปรแกรม เช่น สร้างโปรแกรมตรวจสอบเลขคู่และเลขคี่ เมื่อโปรแกรมทำงานแล้วจะต้องสามารถทำการตรวจสอบค่าของตัวเลขตามความต้องการได้ เป็นต้น

8.2.1.2 การตรวจสอบ (Validation) คือ การตรวจสอบความถูกต้องจากความต้องการของผู้ใช้งาน โดยพิจารณาจากความต้องการที่ผู้ใช้แจ้งในเอกสารก่อนการพัฒนาโปรแกรม เช่น การพัฒนาโปรแกรมสำหรับเก็บข้อมูลลูกค้า ผู้ใช้ต้องการให้บังคับการเก็บข้อมูลหมายเลขบัตรประชาชนเป็นตัวเลขจำนวน 13 หลัก ถ้าโปรแกรมไม่ได้กำหนดจำนวนที่ชัดเจนในการกรอกข้อมูล จะทำให้เกิดข้อผิดพลาดขึ้นมาได้

8.2.2 กลยุทธ์ในการทดสอบโปรแกรม

โดยทั่วไปการทดสอบข้อผิดพลาดของโปรแกรมสามารถทำได้ 2 ประเภทใหญ่ ๆ คือ การทำสอบแบบกล่องดำและการทดสอบแบบกล่องขาว ซึ่งเป็นวิธีที่ใช้ในการตรวจสอบความถูกต้องของโปรแกรม โดยทั้ง 2 วิธีจะมีการทดสอบที่แตกต่างกัน ดังภาพที่ 8.4



ภาพที่ 8.4 กลยุทธ์ในการทดสอบโปรแกรม

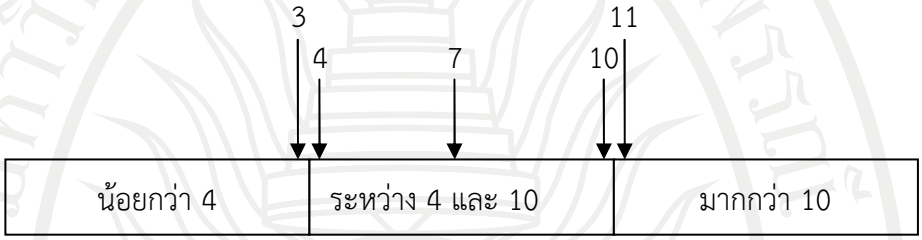
8.2.2.1 การทดสอบแบบกล่องดำ (Black Box Testing) หรืออาจเรียกอีกอย่างหนึ่งว่าการทดสอบเชิงพฤติกรรม (behavioral testing) เป็นการทดสอบที่ไม่สนใจว่าโปรแกรมจะทำงานอย่างไร โดยจะทดสอบข้อมูลเข้าและผลลัพธ์ที่ออกมาเท่านั้น โดยจะไม่มีการตรวจสอบการประมวลผลที่อยู่ภายในของโปรแกรม เพื่อทดสอบสิ่งต่อไปนี้

1) หน้าที่การทำงานผิดพลาดหรือหายไป
2) ความผิดพลาดจากโปรแกรมย่อยในการทำงานร่วมกัน
3) ความผิดพลาดในโครงสร้างข้อมูล หรือการเข้าถึงฐานข้อมูล
4) ความผิดพลาดในการประมวลผลข้อมูล
5) ความผิดพลาดจากการเริ่มต้นหรือจบโปรแกรม

การแบ่งส่วนสมมูล (Equivalence partitioning) เป็นวิธีการทดสอบแบบกล่องดำที่มีการแบ่งส่วนของข้อมูลสำหรับเข้าสู่โปรแกรมออกเป็นกลุ่ม ๆ โดยมีการแบ่งข้อมูลเป็นดังนี้

1) ค่าข้อมูลต่ำสุดของพิสัยข้อมูล (ขอบเขตข้อมูล)
2) ค่าข้อมูลสูงสุดขอพิสัยข้อมูล
3) ค่าข้อมูลตัวแทนกลุ่ม เป็นค่าที่ใกล้เคียงกับค่ากลางของพิสัยข้อมูล
4) ค่าข้อมูลเกินพิสัย หรือเกินขอบเขตข้อมูลในแต่ละช่วง

ตัวอย่างการกำหนดค่าพิสัยข้อมูลของวิธีการทดสอบแบบกล่องดำที่มีค่าตั้งแต่ 4 ถึง 10 จะกำหนดค่าที่ใช้ในการทดสอบคือ 3 4 7 10 และ 11 ดังภาพที่ 8.5



ภาพที่ 8.5 การแบ่งพิสัยของข้อมูลตามวิธีทดสอบแบบกล่องดำ

8.2.2.2 การทดสอบแบบกล่องขาว (White Box Testing) หรือเรียกอีกอย่างหนึ่งว่า การทดสอบแบบกล่องแก้ว (Glass box testing) ซึ่งเป็นวิธีการทดสอบข้อผิดพลาดทางตรรกะและถูกออกแบบมาเพื่อทดสอบสิ่งที่อยู่ในโปรแกรมต่อไปนี้

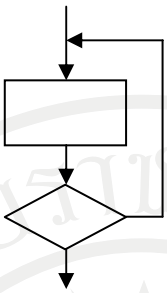
1) ทดสอบทุกเส้นทางในกระบวนการให้สามารถทำงานได้อย่างถูกต้อง
2) ทดสอบการตัดสินใจทางตรรกะทุกเส้นทาง ทั้งค่าจริงและค่าเท็จ
3) ทดสอบการทำงานภายในของการวนรอบตามจำนวนที่กำหนดการวนรอบ
4) ทดสอบโครงสร้างข้อมูลภายในก่อนส่งข้อมูลไปยังการประมวลผลอื่น

สิ่งที่วิธีการทดสอบแบบกล่องขาวทำในการทดสอบเป็นหลักคือ การตรวจสอบเส้นทางการทำงานของคำสั่งในโปรแกรมเรียกว่า การทดสอบเส้นทางมูลฐาน (Basis path testing) ซึ่งเป็นเทคนิคการทดสอบที่ช่วยให้ผู้เขียนโปรแกรมใช้ในการออกแบบกรณีทดสอบเส้นทางการทำงานต่าง ๆ และใช้ค่าที่สามารถวัดได้มาเป็นเกณฑ์ในการกำหนดชุดทดสอบของเส้นทางการทำงาน โดยการทดสอบจะพิจารณาจากเส้นทางต่าง ๆ ดังนี้

1) เส้นทางที่เรียงลำดับการทำงาน (Sequence logical path)
2) เส้นทางที่มีเงื่อนไข (If logical path)
3) เส้นทางที่มีการทำงานแบบวนรอบ (Loop logical path)

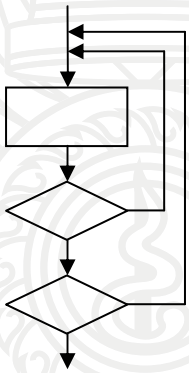
สำหรับตรวจสอบเส้นทางที่มีการทำงานแบบวนรอบจำเป็นต้องแยกลักษณะโครงสร้างของการวนรอบออกมาชัดเจน เพื่อให้สามารถวิเคราะห์ลักษณะการทำให้ถูกต้อง ซึ่งสามารถแบ่งโครงสร้างได้ดังนี้

1) ลูปอย่างง่าย (Simple loop) เป็นลูปที่มีโครงสร้างอย่างง่ายและไม่มีความซับซ้อน ซึ่งจะมีลักษณะการทำงานที่เสร็จสิ้นแค่ชั้นเดียว ดังภาพที่ 8.6



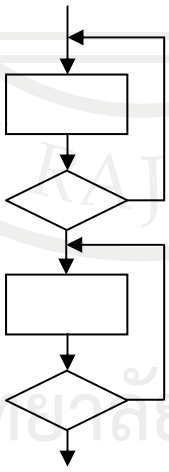
ภาพที่ 8.6 ลูปอย่างง่าย

2) ลูปซ้อนลูป (Nested loop) เป็นลูปที่มีโครงสร้างในลักษณะที่นำลูปมาซ้อนกัน ทำให้เกิดความซับซ้อนมากขึ้น ซึ่งจะมีตั้งแต่ 2 ชั้นขึ้นไป ดังภาพที่ 8.7



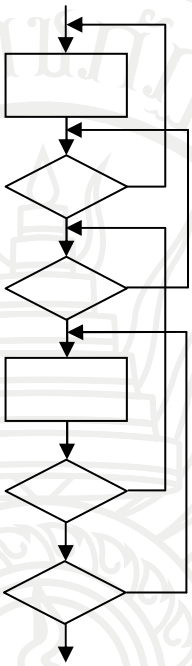
ภาพที่ 8.7 ลูปซ้อนลูป

3) ลูปต่อกัน (Concatenated loop) เป็นลูปที่มีการนำลูปมาต่อกันทำให้มีโครงสร้างใหญ่ขึ้น แต่มีความซับซ้อนไม่มากนัก ดังภาพที่ 8.8



ภาพที่ 8.8 ลูปต่อกัน

4) ลูปไม่เป็นโครงสร้าง (Unstructured loops) เป็นลูปที่มีความซับซ้อนมากที่สุด เนื่องจากไม่มีโครงสร้างของลูปที่ชัดเจน ทำให้การทดสอบมีความยุ่งยากมาก ดังภาพที่ 8.9



ภาพที่ 8.9 ลูปไม่เป็นโครงสร้าง

8.2.3 กลวิธีในการทดสอบโปรแกรม

วิธีการทดสอบแต่ละวิธีจะพยายามค้นหาข้อผิดพลาดที่เกิดขึ้น และแก้ไขให้สามารถทำงานได้อย่างถูกต้อง หลังจากทดสอบแต่ละโปรแกรมย่อยแล้วยังต้องนำมารวมกันเพื่อทดสอบการใช้งานก่อน เพื่อให้ได้ข้อสรุปว่าโปรแกรมที่สร้างขึ้นไม่เกิดข้อผิดพลาดขึ้นมาอีก ซึ่งสามารถทดสอบได้หลายระดับดังนี้

8.2.3.1 การทดสอบส่วนย่อย (Unit testing) คือ การทดสอบส่วนย่อยต่าง ๆ ของโปรแกรมก่อนที่จะนำมารวมเป็นโปรแกรมขนาดใหญ่ การทดสอบในส่วนนี้อาจจะใช้วิธีการแบบกล่องดำหรือวิธีการแบบกล่องขาวก็ได้ โดยจะพิจารณาจากองค์ประกอบของโปรแกรมหรือภายในโปรแกรมย่อย เพื่อประเมินการทำงานในส่วนต่าง ๆ ของโปรแกรม

8.2.3.2 การทดสอบการรวมกัน (Integration testing) คือ การทดสอบโดยนำเอาส่วนย่อยของโปรแกรมที่ได้หลังจากทดสอบแบบส่วนย่อยแล้วมารวมกัน โดยจะทดสอบการทำงานของกลุ่มโปรแกรมหรือส่วนของโปรแกรมย่อยที่ถูกนำมารวมเข้าไว้ด้วยกัน เพื่อค้นหาข้อผิดพลาดที่อาจจะเกิดขึ้นได้หลังจากการรวมกันของโปรแกรมย่อย

8.3 เครื่องมือสำหรับตรวจสอบข้อผิดพลาด

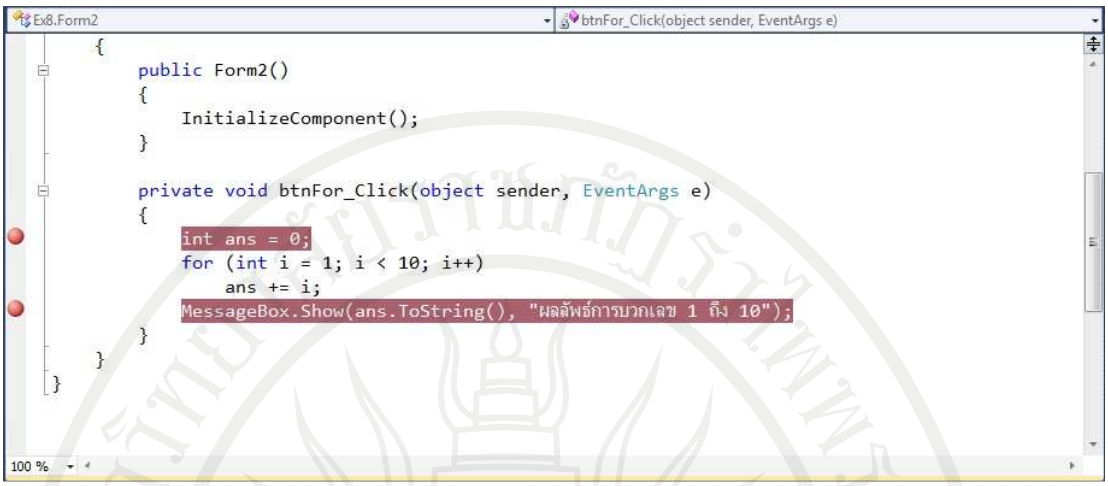
สำหรับการตรวจสอบข้อผิดพลาดที่เกิดขึ้นในการเขียนโปรแกรมด้วยวิชวลซีชาร์ปจะมีเครื่องมือที่ใช้ในการตรวจสอบข้อผิดพลาดต่าง ๆ เรียกว่า ดีบักเกอร์ (Debugger) ซึ่งสามารถค้นหาสาเหตุของข้อผิดพลาดโดยแบ่งเป็นแถบเครื่องมือดังตารางที่ 8.1

ตารางที่ 8.1 ปุ่มแถบเครื่องมือดีบัก

ปุ่ม	ชื่อ	คำอธิบาย
	Start	เริ่มต้นให้โปรแกรมทำงาน หรือทำต่อหลังจากหยุดไว้ชั่วคราว
	Break	หยุดการทำงานของโปรแกรมไว้ชั่วคราว
	End	จบการทำงานของโปรแกรม
	Restart	เริ่มต้นให้โปรแกรมทำงานใหม่ตั้งแต่ต้น
	Show Next Step	แสดงคำสั่งถัดไปที่กำลังจะทำงาน
	Step Into	ทำงานทีละคำสั่ง
	Step Over	ทำงานทีละคำสั่ง แต่ถ้ามีการเรียกใช้โปรแกรมย่อยก็ให้ทำงานในโปรแกรมย่อยรวดเดียวจบ
	Step Out	ทำงานในโปรแกรมย่อยจนเสร็จ และหยุดทำงานที่คำสั่งถัดไปหลังคำสั่งเรียกใช้โปรแกรมย่อยนั้น
Hex	Hexadecimal Display	แสดงผลลัพธ์เป็นเลขฐานสิบหก
	Show Threads in Source	แสดงการทำงานแบบเธรดในคำสั่งของโปรแกรม
	Breakpoints	แสดงหน้าต่างเบรคพอยท์ขึ้นมา

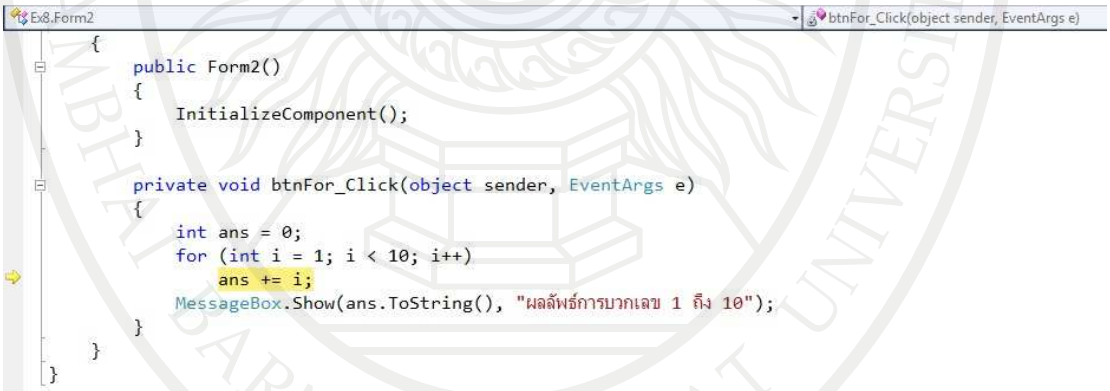
แถบเครื่องมือของดีบักที่ใช้ในการตรวจสอบข้อผิดพลาดของโปรแกรมสามารถช่วยตรวจสอบข้อผิดพลาดประเภททางตรรกะได้ ซึ่งเครื่องมือเหล่านี้จะช่วยทำให้เข้าใจการทำงานในแต่ละขั้นตอนของโปรแกรมและการเปลี่ยนแปลงของค่าตัวแปรต่าง ๆ ได้เป็นอย่างดี ทำให้ค้นหาข้อผิดพลาดได้สะดวกและรวดเร็วมากขึ้น

8.3.1 เบรคพอยท์ (Breakpoints) ใช้กำหนดตำแหน่งที่ต้องการสำหรับการทำให้โปรแกรมหยุดการทำงานชั่วคราวแล้วกลับมายังส่วนของการเขียนคำสั่งในโปรแกรม การตั้งเบรกพอยท์สามารถทำได้โดยคลิกที่แถบด้านซ้ายมือของบรรทัดนั้นหรือคลิกเมาส์ที่ปุ่มขวาเพื่อเลือกคำสั่ง Insert Breakpoints เมื่อทำแล้วจะขึ้นจุดสีแดงหน้าบรรทัดนั้น การตั้งเบรคพอยท์สามารถตั้งได้มากกว่า 1 แห่ง หากต้องการยกเลิกตำแหน่งเบรคพอยท์สามารถทำได้โดยการคลิกซ้ำที่บรรทัดที่ตั้งเบรคพอยท์ไว้ เมื่อกำหนดจดเบรคพอยท์แล้วสั่งให้โปรแกรมทำงาน โปรแกรมจะทำงานไปเรื่อย ๆ จนถึงจุดที่เป็นเบรคพอยท์ก็จะหยุดทำงานชั่วคราว จากนั้นผู้เขียนโปรแกรมก็สามารถใช้เครื่องมือต่าง ๆ ตรวจสอบการทำงานของโปรแกรมได้ ดังภาพที่ 8.10



ภาพที่ 8.10 การใช้งานเบรคพอยท์

8.3.2 การสั่งโปรแกรมทำงานแบบ Step Into เป็นการสั่งให้โปรแกรมทำงานทีละคำสั่ง ถ้าหากมีการทำงานในโปรแกรมย่อยหรือเมธอดก็จะเข้าไปทำงานในนั้นทีละคำสั่ง การใช้งาน Step Into สามารถใช้ได้โดยเลือกปุ่ม Step Into ที่แถบเครื่องมือ และกดไปเรื่อย ๆ เพื่อใช้ในการสังเกตการทำงานของโปรแกรม เมื่อใช้ Step Into คำสั่งในโปรแกรมจะแสดงลำดับการทำงานออกมาทำให้ผู้เขียนโปรแกรมทราบว่าโปรแกรมทำงานที่คำสั่งใดบ้าง ดังภาพที่ 8.11



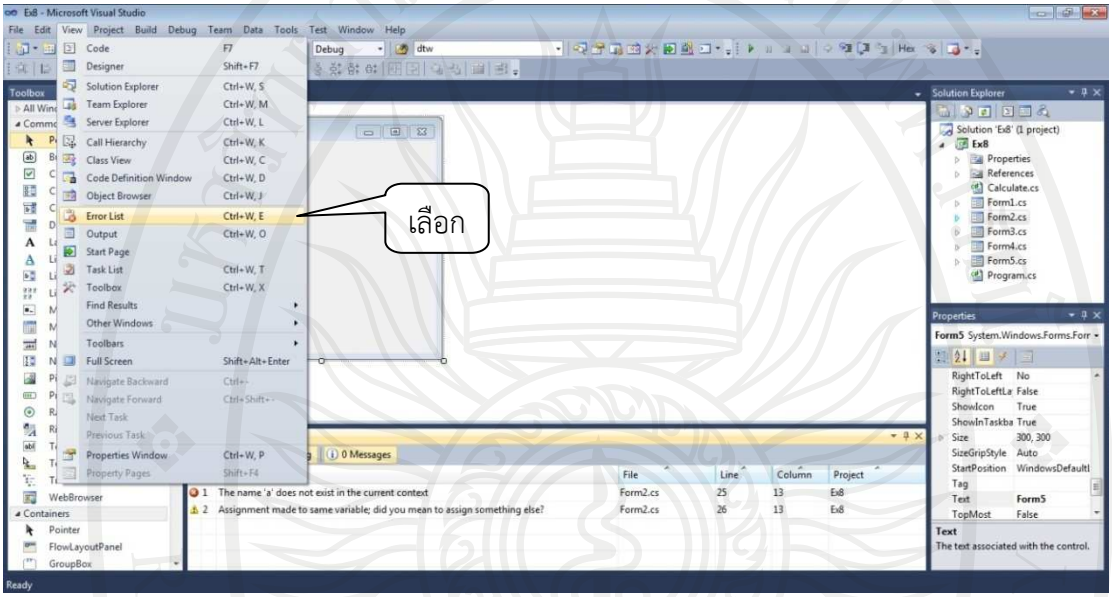
ภาพที่ 8.11 การใช้งาน Step Into

8.3.3 การสั่งโปรแกรมทำงานแบบ Step Over เป็นการสั่งให้โปรแกรมทำงานทีละคำสั่ง เช่นเดียวกับ Step Into แต่การทำงานแบบ Step Over จะมองคำสั่งที่อยู่ในโปรแกรมย่อยหรือเมธอดเป็นคำสั่งเดียว การใช้งาน Step Over สามารถใช้งานได้โดยเลือกปุ่ม Step Over ที่แถบเครื่องมือซึ่งการทำงานจะเหมือนกับ Step Into

8.3.4 การสั่งโปรแกรมทำงานแบบ Step Out เป็นการสั่งให้โปรแกรมทำงานคำสั่งในโปรแกรมย่อยที่เหลือรวดเดียว ซึ่งคำสั่งนี้จะใช้หลังจากการใช้คำสั่ง Step Into เพื่อเข้าไปดูการทำงานใน

โปรแกรมย่อยหรือเมธอดแล้ว เมื่อต้องการกลับออกมาจากโปรแกรมย่อยสามารถใช้คำสั่ง Step Out ออกมาได้

8.3.5 การใช้หน้าต่าง Error List เป็นส่วนที่ใช้ในการดูข้อผิดพลาดที่เกิดขึ้น โดยเมื่อเกิดข้อผิดพลาดขึ้นวิชวลซีชาร์ปจะแสดงข้อมูลของข้อผิดพลาดขึ้นมา การเปิดใช้หน้าต่าง Error List สามารถทำได้โดยเลือกที่เมนู View -> Error List หรือกดปุ่ม Ctrl+W, E ตามภาพที่ 8.12



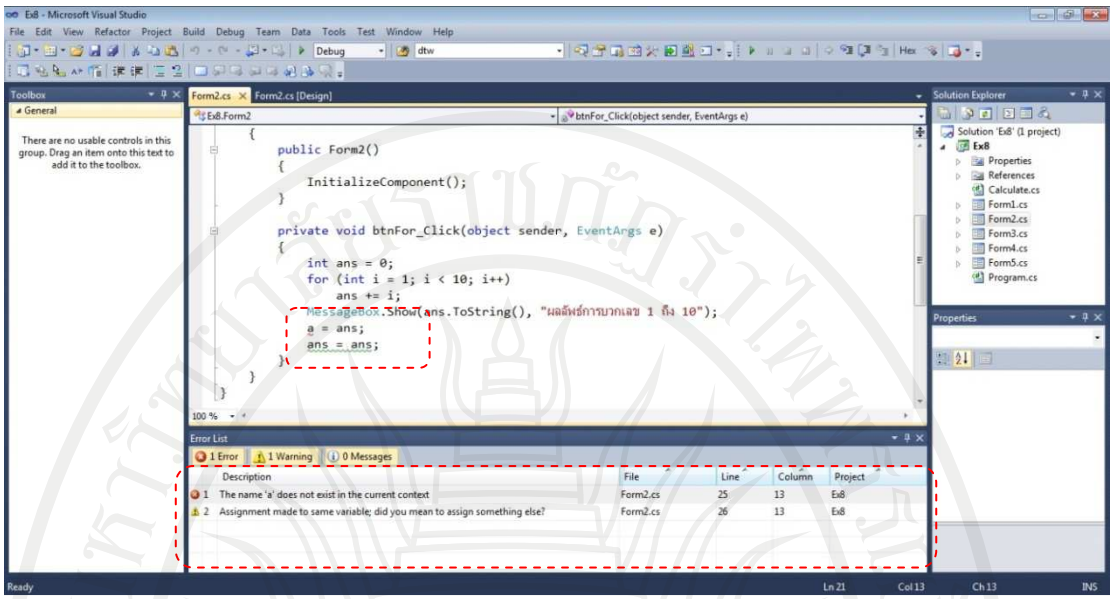
ภาพที่ 8.12 การเรียกใช้งานหน้าต่าง Error List

เมื่อเกิดข้อผิดพลาดขึ้นในระหว่างการเขียนโปรแกรม ข้อผิดพลาดทั้งหมดจะถูกแสดงที่หน้าต่าง Error List ซึ่งหน้าต่างนี้จะแสดงส่วนที่เป็นรายละเอียดของข้อผิดพลาดดังตารางที่ 8.2

ตารางที่ 8.2 รายละเอียดของหน้าต่าง Error List

หัวข้อ	**รายละเอียดของการแสดงผล**
Description	รายละเอียดของข้อผิดพลาดที่เกิดขึ้น
File	แฟ้มที่เกิดข้อผิดพลาด
Line	ตำแหน่งบรรทัดของข้อผิดพลาด
Column	ตำแหน่งของข้อผิดพลาดในบรรทัด
Project	ชื่อของโปรเจคต์ที่เกิดข้อผิดพลาด

ผู้เขียนโปรแกรมสามารถตรวจสอบรายละเอียดของข้อผิดพลาดต่าง ๆ ได้จากหน้าต่าง Error List เพื่อแก้ไขข้อผิดพลาดที่เกิดขึ้นดังภาพที่ 8.13



ภาพที่ 8.13 รายละเอียดที่แสดงในหน้าต่าง Error List

8.4 การจัดการข้อผิดพลาด

ในการเขียนโปรแกรมที่ดีจะต้องมีการตรวจสอบและป้องกันข้อผิดพลาดที่อาจจะเกิดขึ้น ความผิดพลาดสามารถเกิดขึ้นได้จากตัวโปรแกรมเองหรือจากสภาพแวดล้อมต่าง ๆ ของโปรแกรม ซึ่งเป็นสิ่งที่คาดไม่ถึงเรียกว่า สิ่งผิดปกติ (Exception) สามารถเกิดได้จากหลายสาเหตุ เช่น การกรอกข้อมูลผิดประเภท การหารด้วยค่าที่เป็นศูนย์ หรือการเปิดแฟ้มข้อมูลที่ไม่มีอยู่จริง เป็นต้น การจัดการกับสิ่งผิดปกติเหล่านี้ในวิชวลซีชาร์ปสามารถใช้คำสั่ง try...catch...finally ช่วยในการจัดการกับสิ่งที่เกิดขึ้น โดยมีรูปแบบดังนี้ (Microsoft, 2016)

รูปแบบ

```
try
{
   <คำสั่งที่ต้องการทำงาน>
}
catch
{
   <คำสั่งที่ต้องการทำเมื่อเกิดข้อผิดพลาด>
}
finally
{
   <คำสั่งที่ต้องการทำต่อไปไม่ว่าจะเกิดข้อผิดพลาดหรือไม่ก็ตาม>
}
```

ในการใช้ try...catch...finally ไม่จำเป็นจะต้องใช้ทั้งหมดในคราวเดียวกัน อาจจะใช้เป็น try...catch หรือ try...finally ก็ได้ ขึ้นอยู่กับเงื่อนไขในการเขียนโปรแกรม

ตัวอย่างการใช้คำสั่ง try...catch...finally

```
1       int number = 0;
2       try
3       {
4           number = Convert.ToInt32(txtTest.Text);
5       }
6       catch (Exception)
7       {
8           MessageBox.Show("Error", "Error");
9       }
10      finally
11      {
12          if (number < 0)
13          {
14              txtTest.Text = "0";
15          }
16      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 2 – 5 เป็นส่วนการตรวจสอบข้อผิดพลาดที่อาจจะเกิดขึ้น

บรรทัดที่ 6 – 9 เป็นส่วนการกำหนดการทำงานเมื่อเกิดข้อผิดพลาดขึ้น

บรรทัดที่ 10 – 16 เป็นส่วนการกำหนดการทำงานต่อไปไม่ว่าจะเกิดหรือไม่เกิดข้อผิดพลาด

ตัวอย่างการใช้คำสั่ง try...catch

```
1       int number = 0;
2       try
3       {
4           number = Convert.ToInt32(txtTest.Text);
5       }
6       catch (Exception)
7       {
8           MessageBox.Show("Error", "Error");
9       }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 2 – 5 เป็นส่วนการตรวจสอบข้อผิดพลาดที่อาจจะเกิดขึ้น

บรรทัดที่ 6 – 9 เป็นส่วนการกำหนดการทำงานเมื่อเกิดข้อผิดพลาดขึ้น

ตัวอย่างการใช้คำสั่ง try...finally

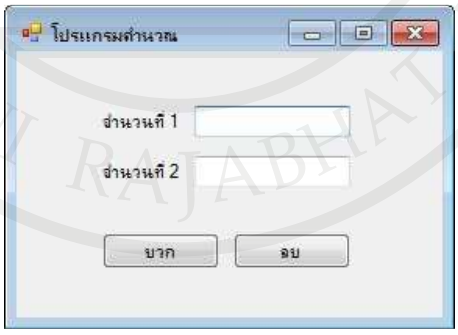
```
1       int number = 0;
2       try
3       {
4           number = Convert.ToInt32(txtTest.Text);
5       }
6       finally
7       {
8           if (number < 0)
9           {
10              txtTest.Text = "0";
11          }
12      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 2 – 5 เป็นส่วนการตรวจสอบข้อผิดพลาดที่อาจจะเกิดขึ้น

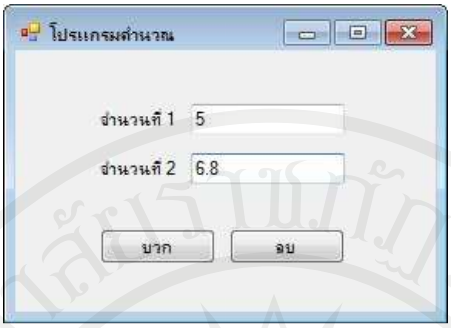
บรรทัดที่ 6 – 16 เป็นส่วนการกำหนดการทำงานต่อไปไม่ว่าจะเกิดหรือไม่เกิดข้อผิดพลาด

ตัวอย่างที่ 8.1 จงเขียนโปรแกรมบวกและลบค่าของตัวเลขจำนวนเต็ม โดยมีหน้าจอรับข้อมูลดังภาพที่ 8.14



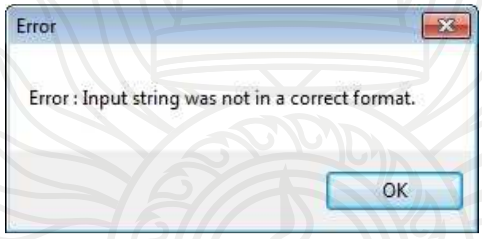
ภาพที่ 8.14 หน้าจอโปรแกรมตัวอย่างการตรวจสอบข้อผิดพลาด

เมื่อสั่งให้โปรแกรมทำงานและใส่ค่าเลขเป็นทศนิยมตามภาพที่ 8.15



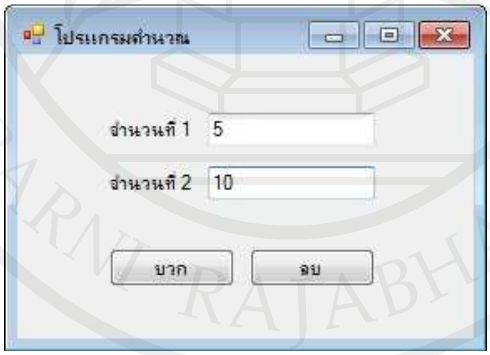
ภาพที่ 8.15 การใส่ค่าเป็นเลขทศนิยมในโปรแกรม

ผลลัพธ์ที่ได้เมื่อใส่ค่าเป็นเลขทศนิยมและกดปุ่มบวกหรือลบ ดังภาพที่ 8.16



ภาพที่ 8.16 ผลลัพธ์เมื่อใส่ค่าที่ไม่ใช่เลขจำนวนเต็ม

เมื่อใส่ค่าที่เป็นตัวเลขจำนวนเต็มตามภาพที่ 8.17 และเมื่อกดปุ่มบวกหรือลบ จะได้ผลลัพธ์ตามภาพที่ 8.18



ภาพที่ 8.17 การใส่ค่าเป็นเลขจำนวนเต็ม

ภาพที่ 8.18 ผลลัพธ์เมื่อไม่เกิดข้อผิดพลาด

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmCal
- Text กำหนดเป็น “โปรแกรมคำนวณ”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblNum1
- Text กำหนดเป็น “ตัวตั้ง”

กำหนด Properties สำหรับ label2
- Name กำหนดเป็น lblNum2
- Text กำหนดเป็น “ตัวหาร”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtNum1
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2
- Name กำหนดเป็น txtNum2
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnDiv
- Text กำหนดเป็น “หาร”

ในตัวอย่างของโปรแกรมจะมีการสร้างคลาสขึ้น 1 คลาส เพื่อใช้สำหรับการคำนวณ โดยมีชุดคำสั่งดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   namespace Ex8_1
6   {
7       class Calculate
8       {
```

```
9               public int Add(int num1, int num2)
10              {
11                  return num1 + num2;
12              }
13              public int Sub(int nun1, int num2)
14              {
15                  return nun1 - num2;
16              }
17          }
18      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 – 12 เป็นส่วนการกำหนดเมธอดสำหรับการบวกที่มีการคืนค่าเป็นจำนวนเต็ม

บรรทัดที่ 13 – 16 เป็นส่วนการกำหนดเมธอดสำหรับการลบที่มีการคืนค่าเป็นจำนวนเต็ม

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       namespace Ex8_1
10      {
11          public partial class frmCal : Form
12          {
13              public frmCal()
14              {
15                  InitializeComponent();
16              }
17              private void btnAdd_Click(object sender, EventArgs e)
18              {
19                  int number1 = 0;
20                  int number2 = 0;
21                  Calculate cal = new Calculate();
```

```
22                 try
23                 {
24                     number1 = Convert.ToInt32(txtNum1.Text);
25                     number2 = Convert.ToInt32(txtNum2.Text);
26                     MessageBox.Show(cal.Add(number1, number2).ToString()," 
       ผลลัพธ์การบวก");
27                 }
28                 catch (Exception ex)
29                 {
30                     MessageBox.Show("Error : " + ex.Message, "Error");
31                 }
32             }
33             private void btnSub_Click(object sender, EventArgs e)
34             {
35                 int number1 = 0;
36                 int number2 = 0;
37                 Calculate cal = new Calculate();
38                 try
39                 {
40                     number1 = Convert.ToInt32(txtNum1.Text);
41                     number2 = Convert.ToInt32(txtNum2.Text);
42                     MessageBox.Show(cal.Sub(number1, number2).ToString(),
       "ผลลัพธ์การลบ");
43                 }
44                 catch (Exception ex)
45                 {
46                     MessageBox.Show("Error : " + ex.Message, "Error");
47                 }
48             }
49         }
50     }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 21 เป็นสร้างวัตถุ cal จากคลาส Calculate

บรรทัดที่ 22 – 27 เป็นการตรวจสอบข้อผิดพลาดที่เกิดขึ้นจากการกรอกข้อมูล

บรรทัดที่ 28 – 31 เป็นส่วนการกำหนดการทำงานเมื่อเกิดข้อผิดพลาด

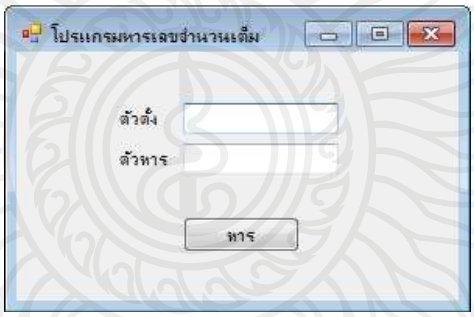
8.5 สรุป

การตรวจสอบข้อผิดพลาดของโปรแกรมเป็นสิ่งจำเป็นเพื่อให้มั่นใจว่าโปรแกรมที่ถูกสร้างขึ้นมานั้นไม่มีข้อผิดพลาดเกิดขึ้น ซึ่งการตรวจสอบคุณภาพของโปรแกรมสามารถแบ่งออกเป็น 2 ด้าน คือ การตรวจสอบความถูกต้องของโปรแกรมโดยการพิจารณาจากรายละเอียดของโปรแกรม และการตรวจสอบจากความต้องการของผู้ใช้ สำหรับกลยุทธ์ที่ใช้ในการตรวจสอบโปรแกรมเพื่อหาข้อผิดพลาดมี 2 วิธี คือ วิธีการทดสอบแบบกล่องดำ ที่จะทดสอบโดยไม่สนใจเส้นทางการทำงานของโปรแกรม และวิธีการทดสอบแบบกล่องขาว ที่ทดสอบโดยการตรวจสอบเส้นทางการทำงานของโปรแกรม นอกจากนี้ยังสามารถทดสอบโดยแบ่งระดับการทดสอบออกเป็น การทดสอบโปรแกรมย่อยก่อน และทดสอบหลังจากนำมารวมกันอีกครั้ง เพื่อให้มั่นใจว่าโปรแกรมที่ได้ไม่มีข้อผิดพลาดเกิดขึ้น

สำหรับวิชวลซีชาร์ปจะมีเครื่องมือที่ช่วยทำให้นักเขียนโปรแกรมสามารถตรวจสอบข้อผิดพลาดได้ง่ายขึ้น โดยสามารถใช้งานเบรคพอยท์ เพื่อกำหนดจุดที่จะให้โปรแกรมหยุดพักการทำงาน และใช้เครื่องมืออื่นในการค้นหาข้อผิดพลาด การแบ่งประเภทของข้อผิดพลาดที่เกิดขึ้นสามารถแบ่งตามลักษณะการเกิดและผลที่เกิดขึ้น ซึ่งโดยทั่วไปสามารถแบ่งออกเป็น 3 ประเภท คือ ข้อผิดพลาดทางไวยกรณ์เป็นข้อผิดพลาดที่เกิดจากการพิมพ์คำสั่งผิดหลักไวยากรณ์ของภาษา ข้อผิดพลาดระหว่างการทำงานเป็นข้อผิดพลาดที่เกิดขึ้นในขณะที่โปรแกรมกำลังปฏิบัติงาน และข้อผิดพลาดทางตรรกะซึ่งเป็นข้อผิดพลาดที่ทำให้ไม่ได้ผลลัพธ์ตามความต้องการของโปรแกรม โดยในประเภทสุดท้ายถือเป็นข้อผิดพลาดที่ร้ายแรงมากที่สุด

แบบฝึกหัดบทที่ 8

1. ข้อผิดพลาดมีกี่ประเภท อะไรบ้าง
2. ประเภทของข้อผิดพลาดที่มีความร้ายแรงที่สุดคืออะไร เพราะเหตุใด
3. การตรวจสอบคุณภาพแบบทวนสอบ (Verification) เป็นแบบใด
4. การตรวจสอบคุณภาพแบบตรวจสอบ (Validation) เป็นแบบใด
5. วิธีการทดสอบแบบกล่องดำและกล่องขาวแตกต่างกันอย่างไร
6. วิธีการทดสอบแบบกล่องดำ เน้นการทดสอบอะไรบ้าง
7. วิธีการทดสอบแบบกล่องขาว เน้นการทดสอบอะไรบ้าง
8. เบรกพอยท์ (Breakpoint) ในวิชวลซีชาร์ปใช้ทำอะไร
9. จงอธิบายการใช้งานคำสั่ง try...catch...finally
10. จงเขียนโปรแกรมหารเลขจำนวนเต็ม โดยป้องกันการใส่ตัวเลขที่ไม่ใช่จำนวนเต็ม และมีหน้าจอแสดงผลดังนี้



กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmDiv
- Text กำหนดเป็น “โปรแกรมหารเลขจำนวนเต็ม”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblNum1
- Text กำหนดเป็น “ตัวตั้ง”

กำหนด Properties สำหรับ label2
- Name กำหนดเป็น lblNum2
- Text กำหนดเป็น “ตัวหาร”

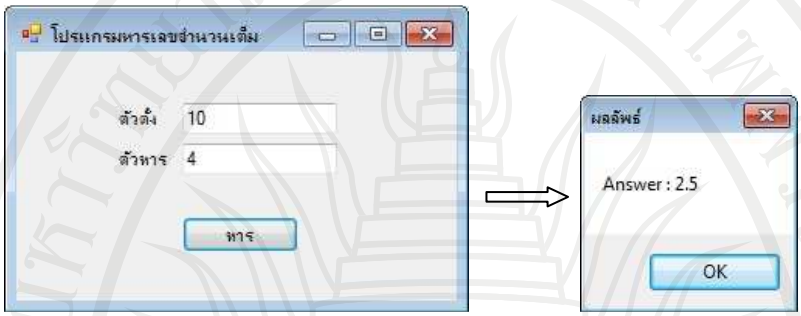
กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtNum1
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2
- Name กำหนดเป็น txtNum2
- Text ใส่เป็นค่าว่าง

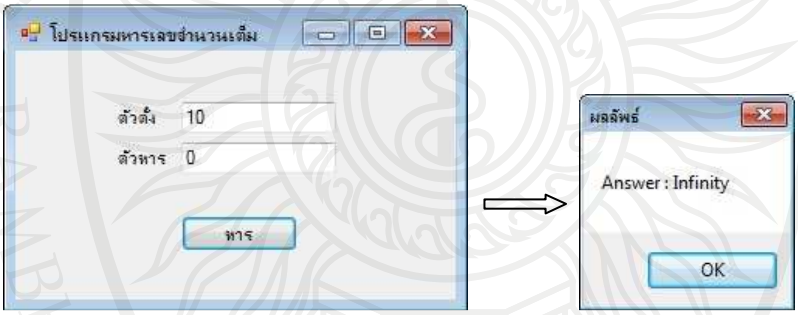
กำหนด Properties สำหรับ button1

Name	กำหนดเป็น btnDiv
Text	กำหนดเป็น “หาร”

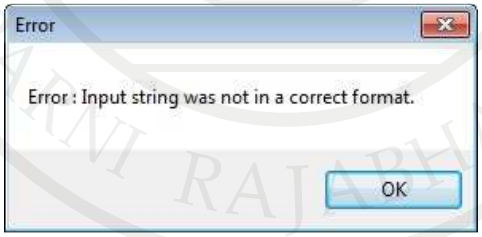
เมื่อโปรแกรมทำงานจะแสดงผลดังนี้



เมื่อใส่ค่าตัวหารที่เป็นศูนย์



เมื่อเกิดข้อผิดพลาดขึ้น โปรแกรมจะแสดงผลดังนี้



เอกสารอ้างอิง

น้ำฝน อัศวเมฆิน. (2558). **หลักการพื้นฐานของวิศวกรรมซอฟต์แวร์**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

นิรันดร์ ประวิทย์ธนา. (2545). **เก่ง C# ให้ครบสูตร**. กรุงเทพฯ : วิตตี้ กรุ๊ป.

ศุภชัย สมพานิช. (2556). **คู่มือเรียนและใช้งาน Visual C#**. กรุงเทพฯ: สวัสดี ไอที.

Microsoft. (2016). **try-catch**. (online). Available : https://msdn.microsoft.com/en-us/library/dszsf989(v=vs.100).aspx. 30 March 2016.

_____. (2016). **try-catch-finally**. (online). Available : https://msdn.microsoft.com/en-us/library/dszsf989(v=vs.100).aspx. 30 March 2016.

_____. (2016). **try-finally**. (online). Available : https://msdn.microsoft.com/en-us/library/dszsf989(v=vs.100).aspx. 30 March 2016.

แผนบริหารประจำบทที่ 9

เนื้อหา

บทที่ 9 เธรด

9.1 ความหมายของเธรด
9.2 การใช้งานเธรด
9.3 การประยุกต์ใช้เธรดกับคอนโทรลใน Windows Forms
9.4 สรุป

จุดประสงค์เชิงพฤติกรรม

เพื่อให้ผู้เรียนสามารถ

1. อธิบายความหมายของเธรดได้
2. ใช้งานเธรดได้เหมาะสมกับงาน
3. เขียนโปรแกรมแบบมัลติเธรดได้
4. ประยุกต์ใช้เธรดกับการเขียนโปรแกรมได้

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายทฤษฎีและซักถามผู้เรียน พร้อมบรรยายประกอบสื่อมัลติมิเดีย โดยใช้คอมพิวเตอร์และโปรเจคเตอร์
2. ให้ผู้เรียนศึกษาเอกสารประกอบการสอน และศึกษาข้อมูลเพิ่มจากอินเทอร์เน็ต
3. ให้ผู้เรียนตั้งคำถามเกี่ยวกับเนื้อหาที่สงสัย
4. ให้ผู้เรียนแบ่งกลุ่มย่อยเพื่ออภิปรายเนื้อหา
5. ให้ผู้เรียนฝึกปฏิบัติ และทำแบบฝึกหัดบทที่ 9

สื่อการเรียนการสอน

1. สื่อมัลติมิเดีย
2. อินเทอร์เน็ต
3. แบบฝึกหัดบทที่ 9
4. เอกสารประกอบการสอนวิชา การเขียนโปรแกรมเชิงวัตถุ

การวัดและการประเมินผล

1. สังเกตจากการซักถามผู้เรียน
2. ประเมินจากการสุ่มถามผู้เรียน
3. ประเมินจากแบบฝึกหัดบทที่ 9
4. ประเมินจากการสอบปลายภาค

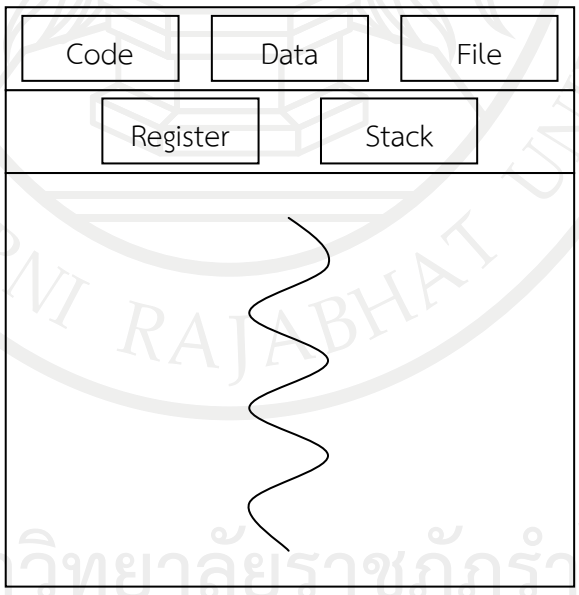
บทที่ 9
เธรด

ในปัจจุบันเครื่องคอมพิวเตอร์มีความสามารถในการประมวลผลอย่างรวดเร็ว และยังสามารถประมวลผลหลาย ๆ งานได้พร้อมกัน โดยเฉพาะการทำงานในระบบซีพียู (CPU) ที่มีแกนประมวลผลหลายแกนจะมีการรองรับการประมวลผลแบบหลายงานพร้อมกัน ซึ่งจำเป็นจะต้องมีการแบ่งการใช้งานซีพียูให้มีประสิทธิภาพสูงสุด ด้วยการแบ่งปันทรัพยากรต่าง ๆ ที่ใช้ในการประมวลผล

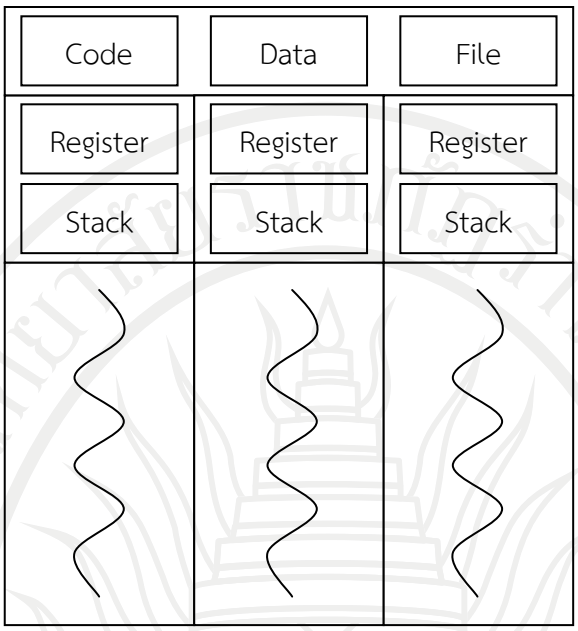
9.1 ความหมายของเธรด

ในการประมวลผลของคอมพิวเตอร์ จะมีการแบ่งเวลาในการประมวลผลงานแต่ละงานออกเป็นส่วน ๆ ในแต่ละส่วนที่เครื่องคอมพิวเตอร์ประมวลผลจะเรียกว่าโปรเซส (Process) แต่ละโปรเซสจะมีการแบ่งเป็นส่วนย่อย ๆ อีก เรียกว่า เธรด (Thread) ซึ่งโดยทั่วไป 1 โปรเซสอาจจะมีเธรดเดียว (Single thread) หรือหลายเธรด (Multi threads) ก็ได้ ดังนั้นเธรดคือส่วนย่อยของโปรเซสที่ถูกแบ่งออกมาเพื่อประมวลผล โดยมีส่วนประกอบในการทำงานดังภาพที่ 9.1 และ 9.2

1) หมายเลขเธรด (Thread ID) เป็นหมายเลขที่ใช้ระบุหมายเลขของเธรดที่อยู่ในโปรเซส
2) ตัวนับ (Program counter) ใช้นับลำดับของคำสั่งเพื่อประมวลผล
3) ชุดของรีจิสเตอร์ (Register set) ใช้เพื่อเก็บค่าตัวแปรที่กำลังทำงานอยู่
4) สแตก (Stack) ใช้เก็บข้อมูลประวัติของการประมวลผล



ภาพที่ 9.1 การทำงานแบบเธรดเดียว



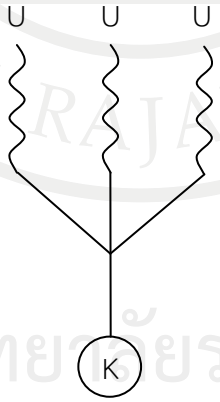
ภาพที่ 9.2 การทำงานแบบหลายเธรด

ภายในโปรเซสที่ประกอบด้วยเธรดจะมีการแบ่งปันทรัพยากรต่าง ๆ เพื่อใช้ในการประมวลผล ซึ่งถ้าเป็นโปรเซสที่ประกอบไปด้วยเธรดเพียงเธรดเดียวจะสามารถทำงานได้ทีละ 1 งาน แต่ถ้าเป็นโปรเซสที่ประกอบไปด้วยเธรดหลายเธรด จะสามารถทำงานได้หลาย ๆ งานพร้อมกัน

9.1.1 ประเภทของหลายเธรด

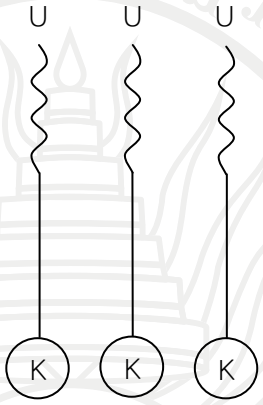
การทำงานแบบหลายเธรดหรือมัลติเธรดนั้นเป็นการทำงานร่วมกันระหว่างเคอร์เนลเธรด (Kernel thread) กับยูสเซอร์เธรด (User thread) ซึ่งสามารถแบ่งออกเป็น 3 ประเภท คือ

9.1.1.1 แบบกลุ่มต่อหนึ่ง (Many to One) เป็นรูปแบบที่มีเคอร์เนลเธรด 1 หน่วย กับยูสเซอร์เธรดหลายหน่วย ซึ่งยูสเซอร์เธรดจะสามารถเข้าใช้เคอร์เนลเธรดได้ทีละ 1 เธรดเท่านั้น ทำให้ไม่สามารถประมวลผลได้พร้อมกัน ดังภาพที่ 9.3



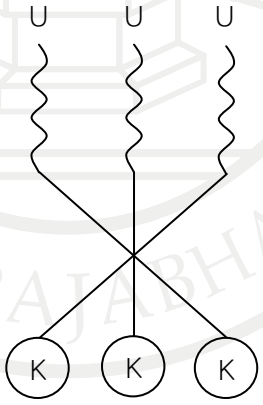
ภาพที่ 9.3 การทำงานแบบกลุ่มต่อหนึ่ง

9.1.1.2 แบบหนึ่งต่อหนึ่ง (One to One) เป็นรูปแบบที่มีเคอร์เนลเธรด 1 หน่วย กับยูสเซอร์เธรด 1 หน่วย โดยจะมีการสร้างยูสเซอร์เธรดเท่ากับจำนวนของเคอร์เนลเธรดทำให้สามารถประมวลผลแบบขนานได้ แต่รูปแบบนี้จะไม่มีการสร้างยูสเซอร์เธรดที่มากเกินกว่าเคอร์เนลเธรด ดังภาพที่ 9.4



ภาพที่ 9.4 การทำงานแบบหนึ่งต่อหนึ่ง

9.1.1.3 แบบกลุ่มต่อกลุ่ม (Many to Many) เป็นรูปแบบที่แก้ไขข้อจำกัดของ 2 แบบแรก โดยจะมีการสร้างยูสเซอร์เธรดเท่าที่จำเป็นเพื่อให้สอดคล้องกับจำนวนของเคอร์เนลเธรดทำให้สามารถประมวลผลแบบขนานได้และสามารถจัดสรรการทำงานระหว่างยูสเซอร์เธรดกับเคอร์เนลเธรดได้ ดังภาพที่ 9.5



ภาพที่ 9.5 การทำงานแบบกลุ่มต่อกลุ่ม

9.2 การใช้งานเธรด

ในวิชวลซีชาร์ปจะมีเมธอดสำหรับการสร้างและใช้งานเธรด ซึ่งทำให้สามารถสร้างเธรดได้หลายเธรด เพื่อทำให้การประมวลผลเร็วขึ้นได้ โดยมีเมธอดที่เกี่ยวข้องดังตารางที่ 9.1

ตารางที่ 9.1 การทำงานของเมธอดเกี่ยวกับเธรด

ชื่อเมธอด	การทำงาน
Start	เริ่มการทำงานของเธรด
Sleep	หยุดเธรดไว้ชั่วคราวตามเวลาที่กำหนด
Suspend	หยุดเธรดไว้ชั่วคราวเมื่อต้องการ
Abort	หยุดการทำงานของเธรดเมื่อต้องการ
Resume	เริ่มการทำงานของเธรดหลังจากหยุดชั่วคราว
Join	ให้เธรดหยุดรอเธรดอื่นทำงานให้เสร็จ ถ้ามีการกำหนดเวลาการทำงานเมธอดจะคืนค่า True เมื่อทำงานตามเวลาที่กำหนด

ที่มา : (Microsoft, 2016)

สำหรับการใช้งานเธรดในวิชวลซีชาร์ปนั้น จะต้องประกาศเรียกใช้ System.Threading ก่อน ซึ่งเป็นเนมสเปซ โดยประกาศในส่วนของ using ดังนี้ (Microsoft, 2016)

```
using System.Threading;
```

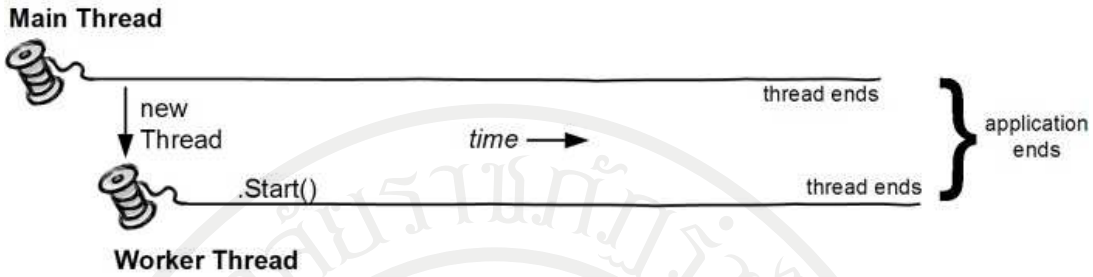
รูปแบบการใช้งานเธรด

```
Thread <ชื่อวัตถุ> = new Thread( <ชื่อเมธอดสำหรับทำงาน> );
<ชื่อวัตถุ>.<method ของเธรด>;
```

ตัวอย่างเช่น

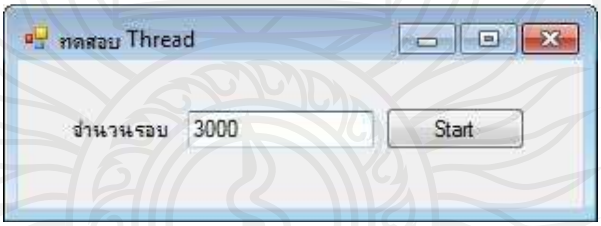
```
Thread thd = new Thread(Write_1);
thd.Start();
```

เมื่อเธรดถูกสั่งให้ทำงานจะมีการแยกการทำงานออกจากกัน ทำให้การควบคุมการทำงานของคอนโทรลใน Windows Forms ถูกแยกออกจากกันด้วย ดังภาพที่ 9.6

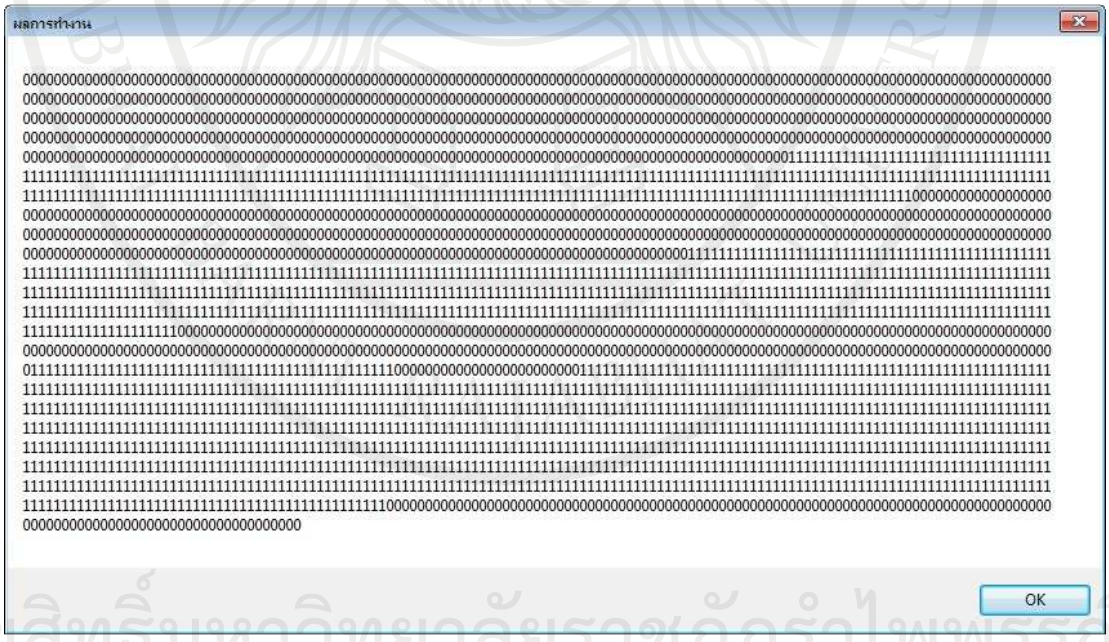


ภาพที่ 9.6 การทำงานของเธรด
ที่มา : (Joseph Albahari , 2014)

ตัวอย่างที่ 9.1 การเริ่มต้นเขียนโปรแกรมแบบหลายเธรด โดยการทำงานของโปรแกรมตัวอย่างจะพิมพ์ค่าเลข 0 และ 1 ซึ่งสร้างเป็นเธรดและจะทำงานแยกจากกัน ดังภาพที่ 9.7 และ 9.8



ภาพที่ 9.7 หน้าจอตัวอย่างโปรแกรมทดสอบเธรด



ภาพที่ 9.8 ผลการทำงานของโปรแกรม

กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmThread

Text กำหนดเป็น “ทดสอบ Thread”

กำหนด Properties สำหรับ label1

Name กำหนดเป็น lblLoop

Text กำหนดเป็น “จำนวนรอบ”

กำหนด Properties สำหรับ textBox1

Name กำหนดเป็น txtLoop

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1

Name กำหนดเป็น btnStart

Text กำหนดเป็น “Start”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   using System.Threading; //เรียกใช้งานคลาส Threading
10  namespace Ex9_1
11  {
12      public partial class frmThread : Form
13      {
14          public frmThread()
15          {
16              InitializeComponent();
17          }
18          int count;
19          string str;
20          private void btnStart_Click(object sender, EventArgs e)
21          {
22              str = "";
```

```
23              count = Convert.ToInt32(txtLoop.Text);
24              Thread thd = new Thread(Write_1);
25              thd.Start();
26              for (int i = 0; i < count; i++)
27              {
28                  str += "0";
29              }
30              MessageBox.Show(str, "ผลการทำงาน");
31          }
32          public void Write_1()
33          {
34              for (int i = 0; i < count; i++)
35              {
36                  str += "1";
37              }
38          }
39      }
40  }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 เป็นการเรียกใช้งานคลาส Threading

บรรทัดที่ 24 เป็นการสร้างวัตถุ thd จากคลาส Thread และกำหนดเมธอดในการทำงานคือ Write_1

บรรทัดที่ 25 เป็นการสั่งให้เธรด thd ทำงานโดยใช้เมธอด Start

บรรทัดที่ 32 – 38 เป็นเมธอดที่เธรด thd ใช้ทำงาน

จากตัวอย่างจะเห็นว่าการพิมพ์เลข 0 และเลข 1 จะพิมพ์แทรกกันอยู่ ซึ่งไม่สามารถกำหนดได้ว่าจะมีตำแหน่งในการพิมพ์อยู่ตรงไหน เนื่องจากการทำงานเป็นแบบขนานเพราะเธรดแต่ละเธรดจะมีการทำงานเป็นของตัวเอง

การทำงานของเธรดในตัวอย่างที่ 9.1 นั้นจะพบปัญหาบางประการคือ เมื่อเธรดมีการทำงานของตนเอง จะไม่สนใจว่าการทำงานของเธรดอื่นจะเสร็จเรียบร้อยหรือไม่ ทำแค่ของตนเองเสร็จก็พอ ทำให้เกิดปัญหาไม่สามารถกำหนดผลลัพธ์ได้ตามต้องการ ดังนั้นในการเขียนโปรแกรมที่ต้องการให้เธรดแต่ละเทรดทำงานให้เสร็จแล้วแสดงผลลัพธ์ออกมาพร้อมกันต้องใช้เมธอด Join ช่วยทำงาน โดยสามารถเพิ่มคำสั่งในปุ่ม btnStart ดังนี้

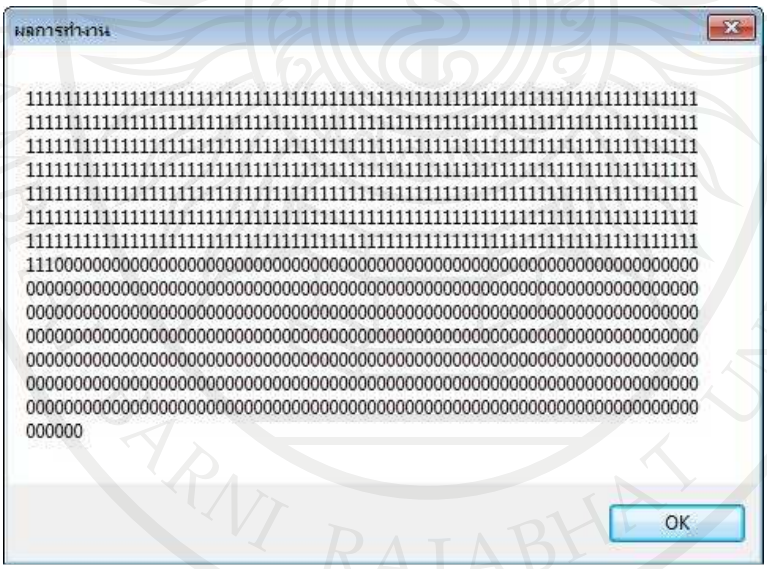

```
1           private void btnStart_Click(object sender, EventArgs e)
2           {
```

```
3                   str = "";
4                   count = Convert.ToInt32(txtLoop.Text);
5                   Thread thd = new Thread(Write_1);
6                   thd.Start();
7                   thd.Join(); //เพิ่มคำสั่ง
8                   for (int i = 0; i < count; i++)
9                   {
10                      str += "0";
11                  }
12                  MessageBox.Show(str, "ผลการทำงาน");
13              }
```

อธิบายชุดคำสั่งของโปรแกรม

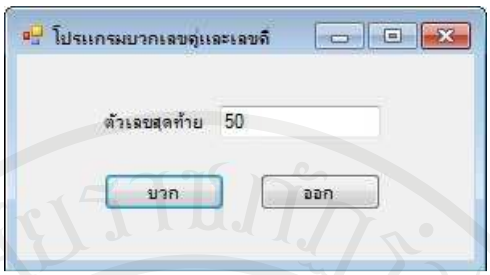
บรรทัดที่ 7 เป็นการเรียกใช้เมธอด Join มาช่วยในการทำงานของเธรด

เมื่อใส่ค่าจำนวนรอบเป็น 500 จะได้ผลลัพธ์เป็นดังภาพที่ 9.9



ภาพที่ 9.9 ผลการใช้เมธอด Join

ตัวอย่างที่ 9.2 จงเขียนโปรแกรมบวกเลขคู่และบวกเลขคี่ตั้งแต่ 1 จนถึงค่าที่อยู่ใน textbox โดยใช้เธรดแบ่งการทำงานระหว่างการบวกเลขคู่และการบวกเลขคี่ ดังภาพที่ 9.10 และ 9.11



ภาพที่ 9.10 หน้าจอโปรแกรมการบวกเลขคู่และเลขคี่

ภาพที่ 9.11 ผลการทำงานของโปรแกรมการบวกเลขคู่และเลขคี่

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmAdd
- Text กำหนดเป็น “โปรแกรมบวกเลขคู่และเลขคี่”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblInput
- Text กำหนดเป็น “ตัวเลขสุดท้าย”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtInput
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnAdd
- Text กำหนดเป็น “บวก”

กำหนด Properties สำหรับ button2
- Name กำหนดเป็น btnExit
- Text กำหนดเป็น “ออก”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1    using System;
2    using System.Collections.Generic;
```

```
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       using System.Threading;
10      namespace Ex9_2
11      {
12          public partial class frmAdd : Form
13          {
14              public frmAdd()
15              {
16                  InitializeComponent();
17              }
18              private void btnAdd_Click(object sender, EventArgs e)
19              {
20                  int odd = 0;
21                  int num = Convert.ToInt32(txtInput.Text);
22                  Thread thd = new Thread(Add_Even);
23                  thd.Start();
24                  for (int i = 1; i <= num; i += 2)
25                  {
26                      odd += i;
27                  }
28                  MessageBox.Show(Convert.ToString(odd), "ผลลัพธ์การบวกเลขคี่");
29              }
30              private void Add_Even()
31              {
32                  int even = 0;
33                  int num = Convert.ToInt32(txtInput.Text);
34                  for (int i = 0; i <= num; i += 2)
35                  {
36                      even += i;
37                  }
38                  MessageBox.Show(Convert.ToString(even), "ผลลัพธ์การบวกเลขคู่");
```

```
39              }
40              private void btnExit_Click(object sender, EventArgs e)
41              {
42                  this.Close();
43              }
44          }
45      }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 9 เป็นการเรียกใช้งานคลาส Threading

บรรทัดที่ 22 เป็นการสร้างวัตถุ thd จากคลาส Thread และกำหนดเมธอดในการทำงานคือ Add_Even

บรรทัดที่ 23 เป็นการสั่งให้เธรด thd ทำงานโดยใช้เมธอด Start

บรรทัดที่ 24 – 28 เป็นการทำงานของเธรดหลัก

บรรทัดที่ 30 – 39 เป็นเมธอดที่เธรด thd ใช้ทำงาน

จากตัวอย่างที่ 9.2 โปรแกรมจะมีการทำงานของเธรดหลักเป็นการบวกเลขคี่ ส่วนเธรดย่อยจะเป็นการบวกเลขคู่ ซึ่งทั้ง 2 เธรดจะมีการทำงานที่แยกออกจากกัน การทำงานของเธรดใดเสร็จก่อนจะแสดงผลด้วย MessageBox แต่ถ้าต้องการให้ทั้ง 2 เธรดเป็นเธรดย่อยเหมือนกัน สามารถเปลี่ยนคำสั่งในโปรแกรม ได้ดังนี้

```
1               private void btnAdd_Click(object sender, EventArgs e)
2               {
3                   Thread thd1 = new Thread(Add_Even);
4                   Thread thd2 = new Thread(Add_Odd);
5                   thd1.Start();
6                   thd2.Start();
7               }
8               private void Add_Even()
9               {
10                  int even = 0;
11                  int num = Convert.ToInt32(txtInput.Text);
12                  for (int i = 0; i <= num; i += 2)
13                  {
14                      even += i;
15                  }
16                  MessageBox.Show(Convert.ToString(even), "ผลลัพธ์การบวกเลขคู่");
```

```
17          }
18          private void Add_Odd()
19          {
20              int odd = 0;
21              int num = Convert.ToInt32(txtInput.Text);
22              for (int i = 1; i <= num; i += 2)
23              {
24                  odd += i;
25              }
26              MessageBox.Show(Convert.ToString(odd), "ผลลัพธ์การบวกเลขคี่");
27          }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 3 – 4 เป็นการสร้างวัตถุ thd1 และ thd2 จากคลาส Thread

บรรทัดที่ 5 – 6 เป็นการสั่งให้เธรด thd1 และ thd2 ทำงานโดยใช้เมธอด Start

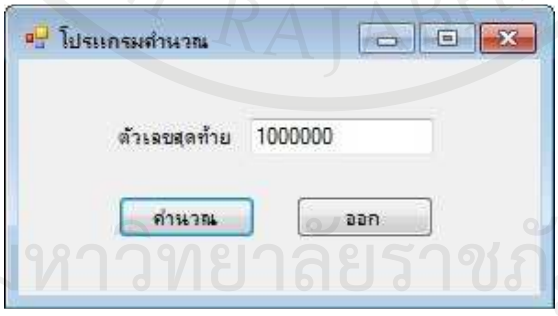
บรรทัดที่ 8 – 17 เป็นเมธอดที่ใช้สำหรับบวกเลขคู่

บรรทัดที่ 18 – 27 เป็นเมธอดที่ใช้สำหรับบวกเลขคี่

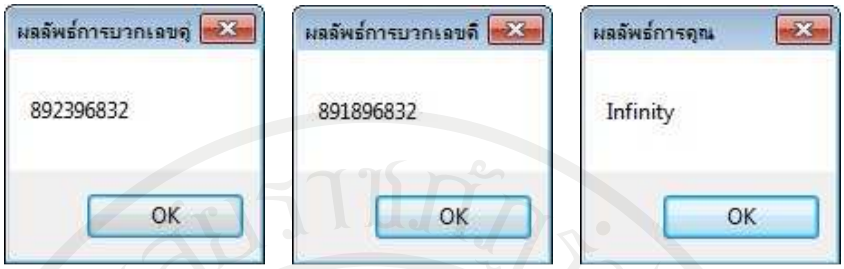
จากการแก้ไขคำสั่งในตัวอย่างที่ 9.2 จะมีการสร้างเธรดย่อย 2 เธรด คือ เธรด thd1 สำหรับการบวกเลขคู่ และเธรด thd2 สำหรับการบวกเลขคี่ ซึ่งลักษณะการทำงานแบบนี้จะทำให้แต่ละเธรดสามารถทำงานของตนเองได้อย่างอิสระ

ในการสร้างเธรดเพื่อใช้ในการประมวลผลนั้น ไม่ควรสร้างเธรดมากจนเกินไปกล่าวคือ ไม่ควรเกิน 2 เท่าของจำนวนเธรดที่อยู่ในซีพียู เช่น ซีพียูมีการทำงาน 4 เธรด สามารถสร้างเธรดได้ไม่เกิน 8 เธรดเพราะถ้าสร้างเธรดมากจนเกินไป จะทำให้ซีพียูทำงานมากจนเกินไปทำให้ไม่สามารถแบ่งเวลาทำงานอย่างมีประสิทธิภาพได้

ตัวอย่างที่ 9.3 ทดลองสร้างเธรด 3 เธรดโดยใช้ตัวอย่างที่ 9.2 มาเพิ่มการคูณเลขตั้งแต่ 1 จนถึงค่าสุดท้ายใน textbox ดังภาพที่ 9.12 และ 9.13



ภาพที่ 9.12 หน้าจอโปรแกรมคำนวณ



ภาพที่ 9.13 ผลลัพธ์จากโปรแกรมคำนวณ

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmCal
- Text กำหนดเป็น “โปรแกรมบวกเลขคู่และเลขคี่”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblInput
- Text กำหนดเป็น “ตัวเลขสุดท้าย”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtInput
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnCal
- Text กำหนดเป็น “คำนวณ”

กำหนด Properties สำหรับ button2
- Name กำหนดเป็น btnExit
- Text กำหนดเป็น “ออก”

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9   using System.Threading;
10  namespace Ex9_3
11  {
```

```
12      public partial class frmCal : Form
13      {
14          public frmCal()
15          {
16              InitializeComponent();
17          }
18          private void btnAdd_Click(object sender, EventArgs e)
19          {
20              Thread thd1 = new Thread(Add_Even);
21              Thread thd2 = new Thread(Add_Odd);
22              Thread thd3 = new Thread(Multi);
23              thd1.Start();
24              thd2.Start();
25              thd3.Start();
26          }
27          private void Add_Even()
28          {
29              int even = 0;
30              int num = Convert.ToInt32(txtInput.Text);
31              for (int i = 0; i <= num; i += 2)
32              {
33                  even += i;
34              }
35              MessageBox.Show(Convert.ToString(even), "ผลลัพธ์การบวกเลขคู่");
36          }
37          private void Add_Odd()
38          {
39              int odd = 0;
40              int num = Convert.ToInt32(txtInput.Text);
41              for (int i = 1; i <= num; i += 2)
42              {
43                  odd += i;
44              }
45              MessageBox.Show(Convert.ToString(odd), "ผลลัพธ์การบวกเลขคี่");
46          }
47          private void Multi()
```

```
48          {
49              double mul = 1;
50              int num = Convert.ToInt32(txtInput.Text);
51              for (int i = 1; i <= num; i += 2)
52              {
53                  mul *= i;
54              }
55              MessageBox.Show(Convert.ToString(mul), "ผลลัพธ์การคูณ");
56          }
57          private void btnExit_Click(object sender, EventArgs e)
58          {
59              this.Close();
60          }
61      }
62  }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 22 เป็นการสร้างวัตถุ thd3 จากคลาส Thread

บรรทัดที่ 25 เป็นการสั่งให้เธรด thd3 ทำงานโดยใช้เมธอด Start

บรรทัดที่ 47 – 56 เป็นเมธอดที่ใช้สำหรับบวกคูณ

จากตัวอย่างที่ 9.3 จะเห็นว่าเมื่อเพิ่มเธรดในการทำงานเข้าไปในโปรแกรม โปรแกรมสามารถแยกการทำงานออกจากกันได้โดยไม่กระทบกับการทำงานในเธรดอื่น

9.3 การประยุกต์ใช้เธรดกับคอนโทรลใน Windows Forms

ในการนำเธรดมาประยุกต์ใช้งานใน Windows Forms นั้นจำเป็นจะต้องอาศัยเทคนิคดีลีเกท (Delegate) เข้ามาช่วย เนื่องจากเธรดในวิชวลซีชาร์ปจะมีการแยกกันทำงานกันอย่างชัดเจน ทำให้การใช้งานร่วมกันของคอนโทรลเกิดปัญหาขึ้น

ดีลีเกทในวิชวลซีชาร์ปจะมีลักษณะคล้ายกับพอยท์เตอร์ (Pointer) ในภาษาซีหรือซีพลัสพลัส ซึ่งจะใช้สำหรับเมธอดที่มีค่าพารามิเตอร์และค่ารีเทิร์นเหมือนกัน ทำให้สามารถใช้งานออบเจ็คต์ที่มีค่าพารามิเตอร์และค่ารีเทิร์นเหมือนกันโดยไม่จำเป็นต้องทราบว่าออบเจ็คต์นั้นอยู่ในคลาสใด โดยสามารถใช้งานผ่านคำสั่ง Invoke (ธีระพล ลิ้มศรัทธา, 2553)

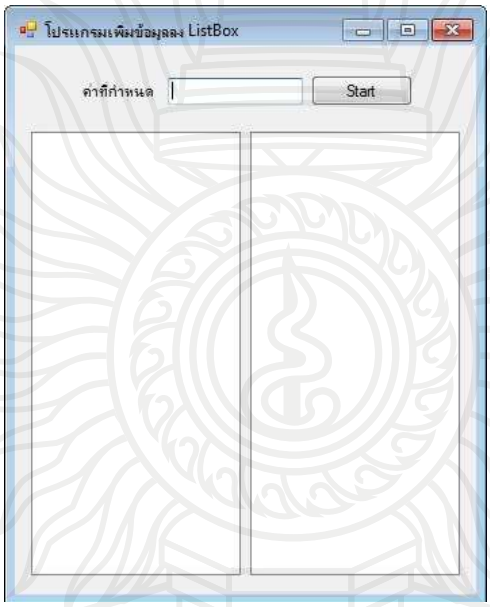
รูปแบบการใช้งานคำสั่ง Invoke

```
public object Invoke(
        Delegate method
)
```

ตัวอย่างเช่น

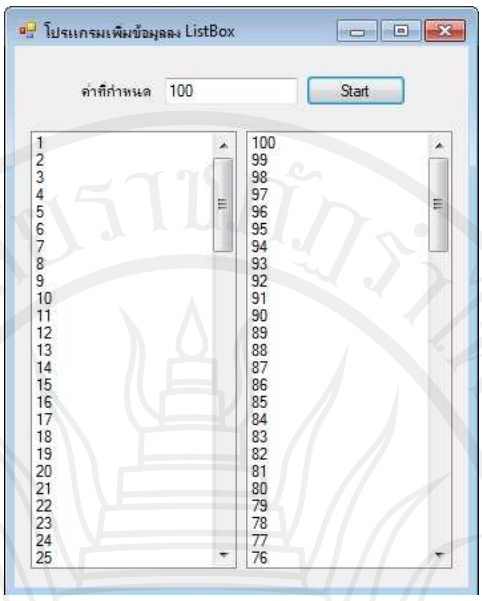
```
Invoke((MethodInvoker)delegate
        {
            MessageBox.Show("Hello");
        });
```

ตัวอย่างที่ 9.4 ทดลองเขียนโปรแกรมใส่ค่าเลขลงใน listBox โดยให้รับค่าตัวเลขใน textbox แล้วเพิ่มเลขลงใน listBox จาก 1 ถึงเลขที่อยู่ใน textBox และจากเลขที่อยู่ใน textBox ลงมาจนถึง 1 โดยมีหน้าจอดังภาพที่ 9.14



ภาพที่ 9.14 หน้าจอโปรแกรมเพิ่มข้อมูลลง listBox

เมื่อใส่ค่าใน textBox เป็น 100 แล้วกดปุ่ม Start โปรแกรมจะเพิ่มข้อมูลลง listBox ทั้งด้านซ้ายและขวาดังภาพที่ 9.15



ภาพที่ 9.15 ผลลัพธ์ของโปรแกรมเพิ่มข้อมูลลง listBox

กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmList
- Text กำหนดเป็น “โปรแกรมเพิ่มข้อมูลลง ListBox”

กำหนด Properties สำหรับ label1
- Name กำหนดเป็น lblInput
- Text กำหนดเป็น “ตัวเลขสุดท้าย”

กำหนด Properties สำหรับ textBox1
- Name กำหนดเป็น txtInput
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1
- Name กำหนดเป็น btnStart
- Text กำหนดเป็น “Start”

กำหนด Properties สำหรับ listBox1
- Name กำหนดเป็น lstUp
- Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ listBox2
- Name กำหนดเป็น lstDown
- Text ใส่เป็นค่าว่าง

ชุดคำสั่งของโปรแกรมเป็นดังนี้

```
1       using System;
```

```
2       using System.Collections.Generic;
3       using System.ComponentModel;
4       using System.Data;
5       using System.Drawing;
6       using System.Linq;
7       using System.Text;
8       using System.Windows.Forms;
9       using System.Threading;
10      namespace Ex9_4
11      {
12          public partial class frmList : Form
13          {
14              public frmList()
15              {
16                  InitializeComponent();
17              }
18              private void btnInsert_Click(object sender, EventArgs e)
19              {
20                  lstUp.Items.Clear();
21                  lstDown.Items.Clear();
22                  Thread thd1 = new Thread(CountUp);
23                  Thread thd2 = new Thread(CountDown);
24                  thd1.Start();
25                  thd2.Start();
26              }
27              private void CountUp()
28              {
29                  int num = Convert.ToInt32(txtInput.Text);
30                  Invoke((MethodInvoker)delegate
31                  {
32                      for(int i = 1; i <= num; i++)
33                      {
34                          lstUp.Items.Add(i);
35                      }
36                  });
37              }
```

```
38          private void CountDown()
39          {
40              int num = Convert.ToInt32(txtInput.Text);
41              Invoke((MethodInvoker)delegate
42              {
43                  for (int i = num; i >= 1; i--)
44                  {
45                      lstDown.Items.Add(i);
46                  }
47              });
48          }
49      }
50  }
```

อธิบายชุดคำสั่งของโปรแกรม

บรรทัดที่ 30 และ 41 เป็นการใช้ดีลีเกทผ่านทางคำสั่ง Invoke

จากตัวอย่างที่ 9.4 จะมีการใช้นำเทคนิคของดีลีเกทมาใช้เพื่อช่วยในการทำงานร่วมกับคอนโทรลของ Windows Forms โดยผ่านทางคำสั่ง Invoke

9.4 สรุป

การใช้เธรดในการเขียนโปรแกรมจะช่วยทำให้โปรแกรมทำงานได้เร็วขึ้น เนื่องจากเธรดจะมีการแยกการประมวลผลออกจากกันอย่างชัดเจน และจะทำงานจนเสร็จ แต่ข้อควรระวังสำหรับการใช้เธรดในการทำงานคือ งานที่ทำแต่ละเธรดจะต้องแยกการทำงานกันอย่างชัดเจน ไม่ใช้การแบ่งเธรดกับงานที่จำเป็นต้องรอผลลัพธ์ของกันและกัน เพราะจะทำให้เธรดไม่สามารถทำงานได้อย่างมีประสิทธิภาพ

ส่วนการนำเธรดมาประยุกต์ใช้กับ Windows Forms นั้น จำเป็นจะต้องอาศัยเทคนิคดีลีเกทเข้ามาช่วย เนื่องจากเธรดในวิชวลซีชาร์ปจะมีการแยกกันทำงานกันอย่างชัดเจน ทำให้การใช้งานร่วมกันของคอนโทรลเกิดปัญหาขึ้น

ดีลีเกทในวิชวลซีชาร์ปจะมีลักษณะคล้ายกับพอยท์เตอร์ (Pointer) ในภาษาซีหรือซีพลัสพลัส ซึ่งจะใช้สำหรับเมธอดที่มีค่าพารามิเตอร์และค่ารีเทิร์นเหมือนกัน ทำให้สามารถใช้งานออบเจ็คต์ที่มีค่าพารามิเตอร์และค่ารีเทิร์นเหมือนกันโดยไม่จำเป็นต้องทราบว่าออบเจ็คต์นั้นอยู่ในคลาสใด โดยสามารถใช้งานผ่านคำสั่ง Invoke

แบบฝึกหัดบทที่ 9

1. จงอธิบายความหมายของเธรด
2. เธรดมีส่วนประกอบอะไรบ้าง
3. มัลติเธรดมีกี่ประเภท อะไรบ้าง
4. การเริ่มต้นใช้งานเธรด จำเป็นต้องเรียกเนมสเปซ (Namespace) ใดก่อน
5. ในการสั่งให้เธรดทำงาน ต้องใช้เมธอดใดในการสั่งงาน
6. เทคนิคดีลีเกทใช้สำหรับทำอะไรเกี่ยวกับเธรด
7. มัลติเธรดไม่ควรใช้ในกรณีใดบ้าง
8. การใช้งานเธรดร่วมกับคอนโทรลใน Windows Forms ต้องทำอย่างไร
9. เมื่อต้องการหยุดการทำงานของเธรดชั่วคราวและให้ทำงานต่อ ต้องใช้เมธอดใดของเธรดทำงานบ้าง
10. คำสั่ง Invoke มีรูปแบบการทำงานอย่างไร

เอกสารอ้างอิง

ธีระพล ลิ้มศรัทธา. (2553). **การเขียนโปรแกรมเชิงวัตถุด้วย Visual Basic.NET**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

Joseph Albahari. (2016). **Threading in C#**. (online). Available : http://www.albahari.com/threading/. 13 April 2016.

Microsoft. (2016). **Multithreaded Applications**. (online). Available : https://msdn.microsoft.com/en-us/library/ck8bc5c6(v=vs.100).aspx. 13 April 2016.

_____. (2016). **Threading**. (online). Available : https://msdn.microsoft.com/en-us/library/ms173178(v=vs.100).aspx. 13 April 2016.

_____. (2016). **Thread Class**. (online). Available : https://msdn.microsoft. com/en-us/library/system.threading.thread(v=vs.100).aspx. 13 April 2016.

_____. (2016). **Thread Methods**. (online). Available : https://msdn.microsoft. com/en-us/library/system.threading.thread_methods(v=vs.100).aspx. 13 April 2016.

บรรณานุกรม

บรรณานุกรม

กิตติพงษ์ กลมกล่อม. (2552). **การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML.** กรุงเทพฯ : เคทีพี.

ธวัชชัย งามสันติวงศ์. (2549). **การวิเคราะห์และออกแบบระบบงานเชิงวัตถุ.** กรุงเทพฯ : ศูนย์หนังสือจุฬาลงกรณ์มหาวิทยาลัย.

ธีระพล ลิ้มศรัทธา. (2553). **การเขียนโปรแกรมเชิงวัตถุด้วย Visual Basic.NET**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

น้ำฝน อัศวเมฆิน. (2558). **หลักการพื้นฐานของวิศวกรรมซอฟต์แวร์**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

นิรันดร์ ประวิทย์ธนา. (2545). **เก่ง C# ให้ครบสูตร**. กรุงเทพฯ : วิตตี้ กรุ๊ป.

วิษณุ ช้างเนียม. (2553). **คู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม.** กรุงเทพฯ : ไอดีซี พรีเมียร์.

ศุภชัย สมพานิช. (2556). **คู่มือเรียนและใช้งาน Visual C#**. กรุงเทพฯ: สวัสดี ไอที.

รูปภาพความละเอียดสูงฟรี.(2556). (ออนไลน์). แหล่งที่มา : https://pixabay.com. 15 มกราคม 2559.

Dofactory. (2016). **.NET Design Patterns**. (online). Available : http://www.dofactory. com/net/design-patterns. 20 March 2016.

Joseph Albahari. (2016). **Threading in C#**. (online). Available : http://www.albahari. com/threading/. 13 April 2016.

Object Management Group. (2016). **WHAT IS UML**. (online). Available : http://www.uml.org/what-is-uml.htm. 10 January 2016.

Microsoft. (2016). **Arrays Tutorial**. (online). Available : https://msdn.microsoft. com/en-us/library/aa288453(v=vs.71).aspx. 10 March 2016.

_____. (2016). **Console Class**. (online). Available : https://msdn.microsoft. com/en-us/library/system.console(v=vs.110).aspx. 19 January 2016.

_____. (2016). **C# Keywords**. (online). Available : https://msdn.microsoft. com/en-us/library/x53a06bb(v=vs.100).aspx. 19 January 2016.

_____. (2016). **C# Programming Guide**. (online). Available : https://msdn. microsoft.com/en- us/library/67ef8sbd(v=vs.100).aspx. 6 February 2016.

_____. (2016). **do**. (online). Available : https://msdn.microsoft.com/en-us/library/ 370s1zax(v=vs.100).aspx. 6 February 2016.

_____. (2016). **for**. (online). Available : https://msdn.microsoft.com/en-us/library/ ch45axte(v=vs.100).aspx. 6 February 2016.

_____. (2016). **Form Class**. (online). Available : https://msdn.microsoft.com/ en-us/library/system.windows.forms.form(v=vs.110).aspx. 6 February 2016.

บรรณานุกรม (ต่อ)

Microsoft. (2016). **MessageBox function**. (online). Available : https://msdn.microsoft.com/en-us/library/windows/desktop/ms645505(v=vs.85).aspx. 21 January 2016.

_____. (2016). **Multithreaded Applications**. (online). Available : https://msdn.microsoft.com/en-us/library/ck8bc5c6(v=vs.100).aspx. 13 April 2016.

_____. (2016). **switch**. (online). Available : https://msdn.microsoft.com/en-us/library/06tc147t(v=vs.100).aspx. 6 February 2016.

_____. (2016). **System.Collections.Generic Namespace**. (online). Available : https://msdn.microsoft.com/en-us/library/system.collections.generic(v=vs.100).aspx. 10 March 2016.

_____. (2016). **System.Collections Namespace**. (online). Available : https://msdn.microsoft.com/en-us/library/system.collections(v=vs.100).aspx. 10 March 2016.

_____. (2016). **try-catch**. (online). Available : https://msdn.microsoft.com/en-us/library/dszsf989(v=vs.100).aspx. 30 March 2016.

_____. (2016). **try-catch-finally**. (online). Available : https://msdn.microsoft.com/en-us/library/dszsf989(v=vs.100).aspx. 30 March 2016.

_____. (2016). **try-finally**. (online). Available : https://msdn.microsoft.com/en-us/library/dszsf989(v=vs.100).aspx. 30 March 2016.

_____. (2016). **Threading**. (online). Available : https://msdn.microsoft.com/en-us/library/ms173178(v=vs.100).aspx. 13 April 2016.

_____. (2016). **Thread Class**. (online). Available : https://msdn.microsoft. com/en-us/library/system.threading.thread(v=vs.100).aspx. 13 April 2016.

_____. (2016). **Thread Methods**. (online). Available : https://msdn.microsoft.com/en-us/library/system.threading.thread_methods(v=vs.100).aspx. 13 April 2016.

_____. (2016). **while**. (online). Available : https://msdn.microsoft.com/en-us/library/2aeyhxcd(v=vs.100).aspx. 6 February 2016.

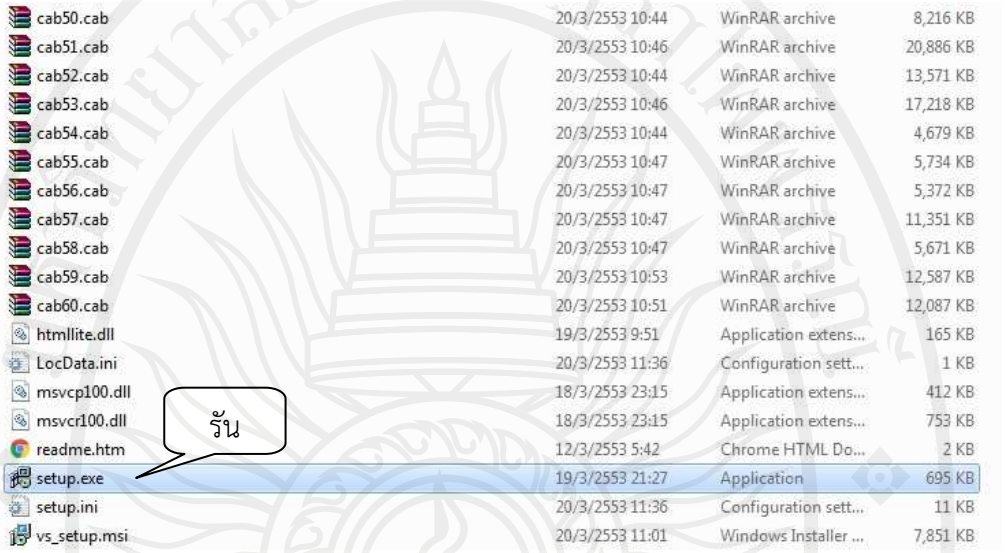
Animal Science. (n.d.). (online). Available : https://www.pinterest.com/libertyp/animal-science-mixed-grade/. 15 January 2016.

Object Oriented Analysis And Design Using UML. (n.d.). (online). Available : http://www.fritzsolms.net/sites/default/files/documents/ObjectOrientedAnalysisAndDesignUsingUML.pdf. 15 January 2016.

ภาคผนวก

การติดตั้งโปรแกรมวิชวลสตูดิโอ 2010

ในการติดตั้งโปรแกรมวิชวลสตูดิโอ 2010 สามารถติดตั้งโดยการเลือกรันไฟล์ setup.exe ที่อยู่ในโฟลเดอร์วิชวลสตูดิโอ 2010 ตามภาพที่ 1.1

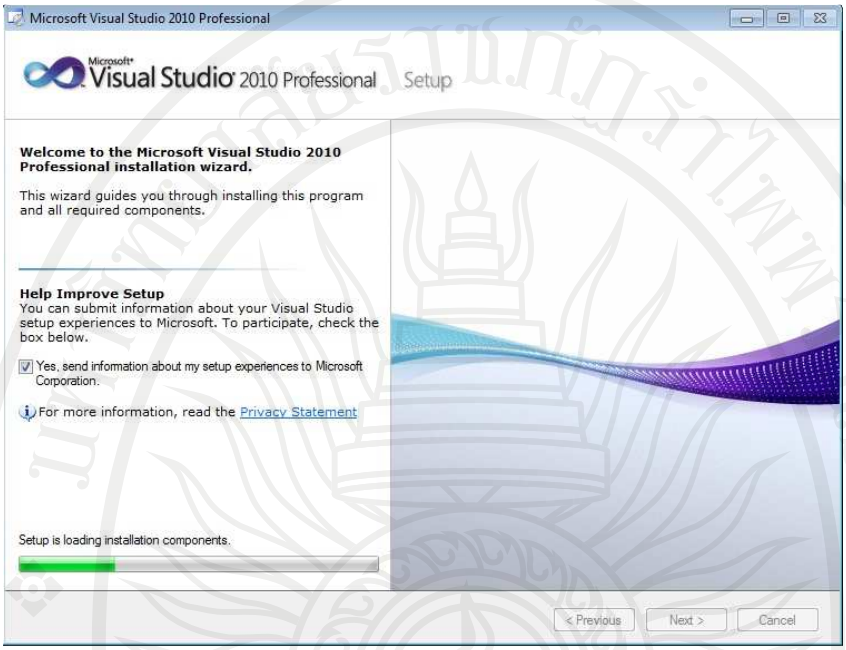


ภาพที่ 1.1 เลือกรันไฟล์ setup.exe

เมื่อรันไฟล์ setup.exe แล้ว ระบบจะดำเนินการติดตั้งวิชวลสตูดิโอ 2010 โดยแสดงตัวเลือกสำหรับใช้ในการติดตั้งดังภาพที่ 1.2

ภาพที่ 1.2 หน้าจอเริ่มการติดตั้งวิชวลสตูดิโอ 2010

เมื่อเลือก Install Microsoft Visual Studio 2010 ระบบจะดำเนินการติดตั้งโปรแกรมโดยตรวจสอบสภาพความพร้อมทั้งหมดเพื่อป้องกันข้อผิดพลาดที่จะเกิดขึ้น

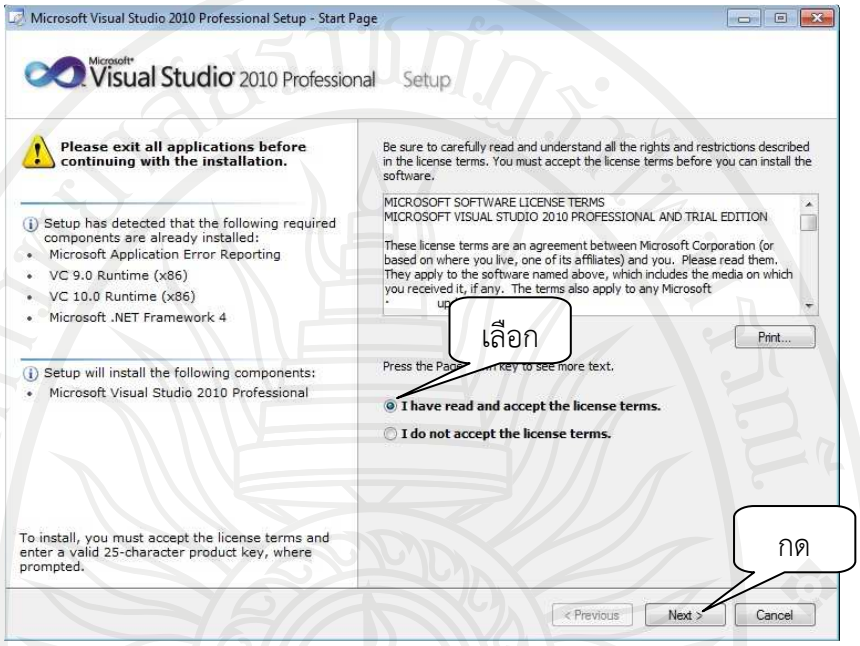


ภาพที่ 1.3 หน้าจอตรวจสอบก่อนการติดตั้ง

หลังจากนั้นให้กดปุ่ม Next เพื่อดำเนินการต่อ

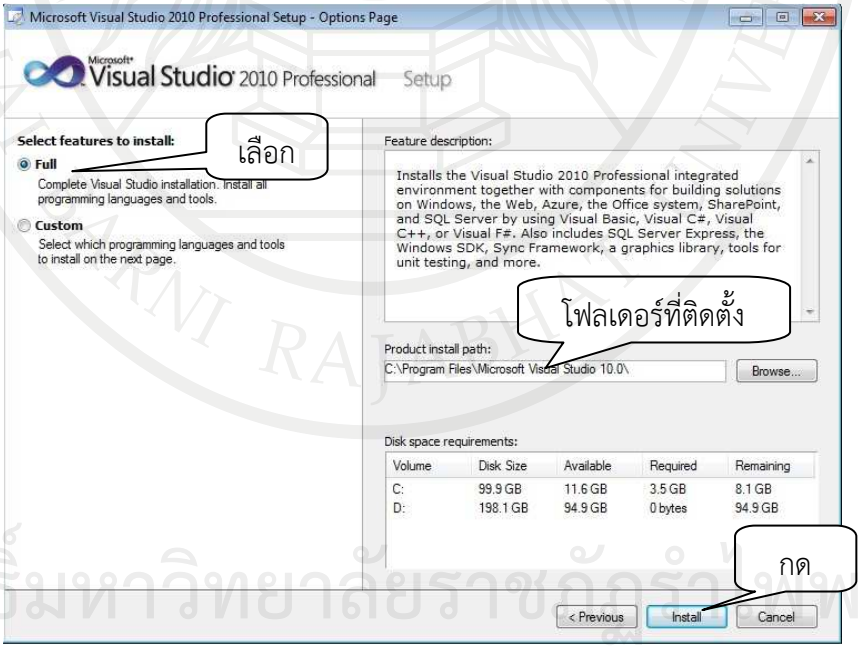
ภาพที่ 1.4 หน้าจอสำหรับดำเนินการติดตั้ง

เมื่อกดปุ่ม Next แล้วระบบติดตั้งจะมีเงื่อนไขต่าง ๆ ให้อ่าน และให้กดยืนยันเมื่ออ่านจบแล้ว ถ้าไม่ยืนยันจะไม่สามารถติดตั้งโปรแกรมต่อได้



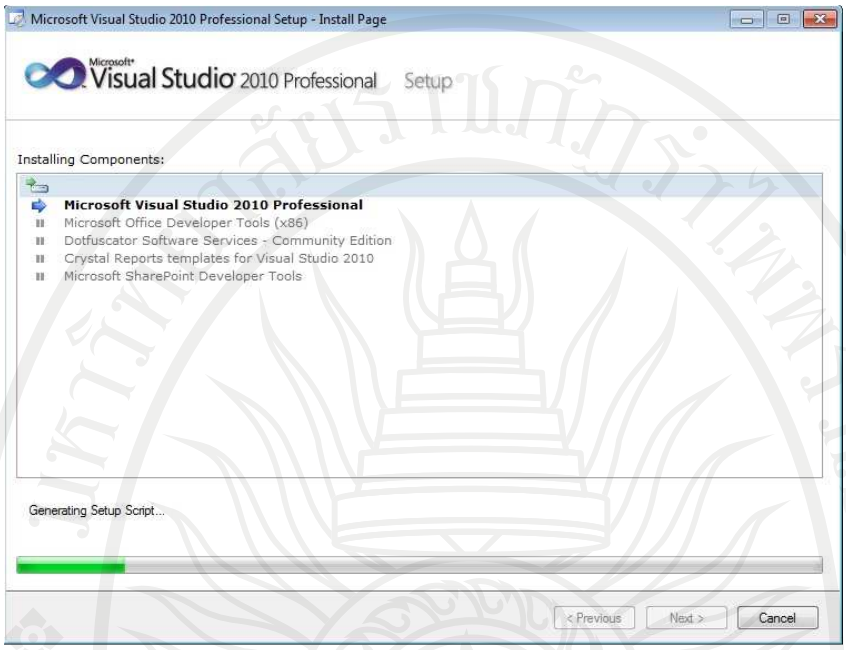
ภาพที่ 1.5 หน้าจอสำหรับยืนยันเงื่อนไข

หลังจากนั้นให้เลือกการติดตั้งแบบ Full เพื่อติดตั้งแบบตัวเต็ม

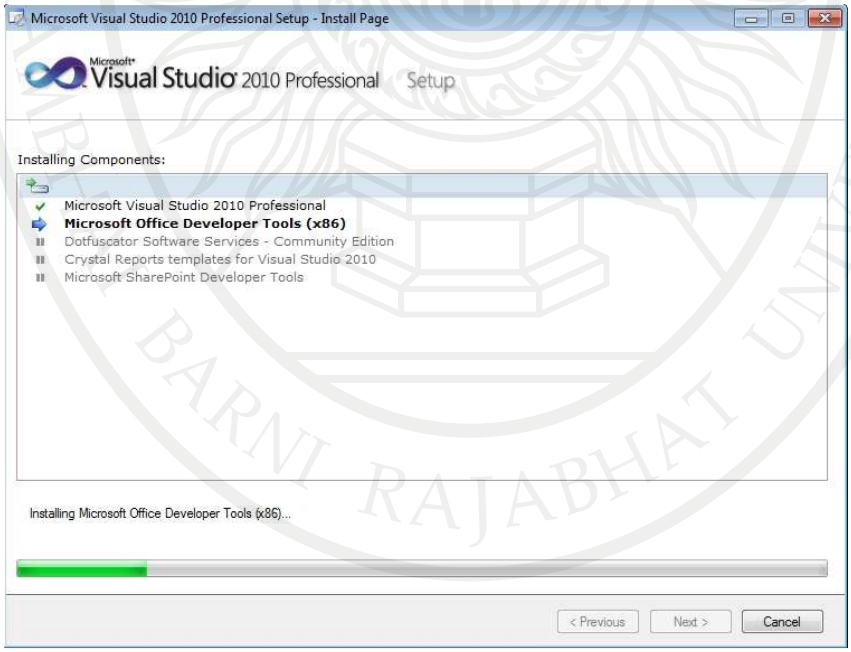


ภาพที่ 1.6 หน้าจอสำหรับเลือกการติดตั้ง

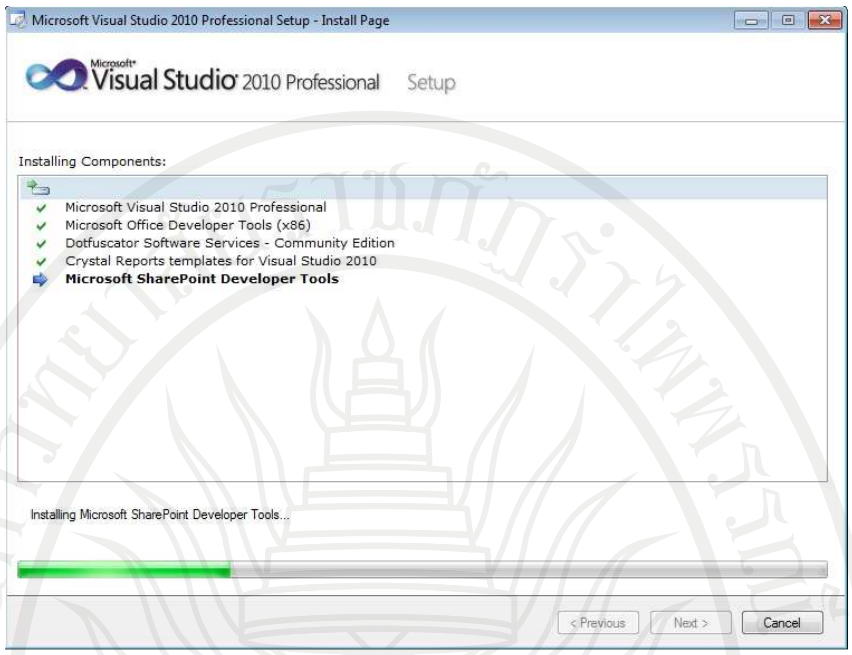
หลังจากเลือกการติดตั้งเรียบร้อยแล้วระบบจะติดตั้งโปรแกรมไปเรื่อย ๆ



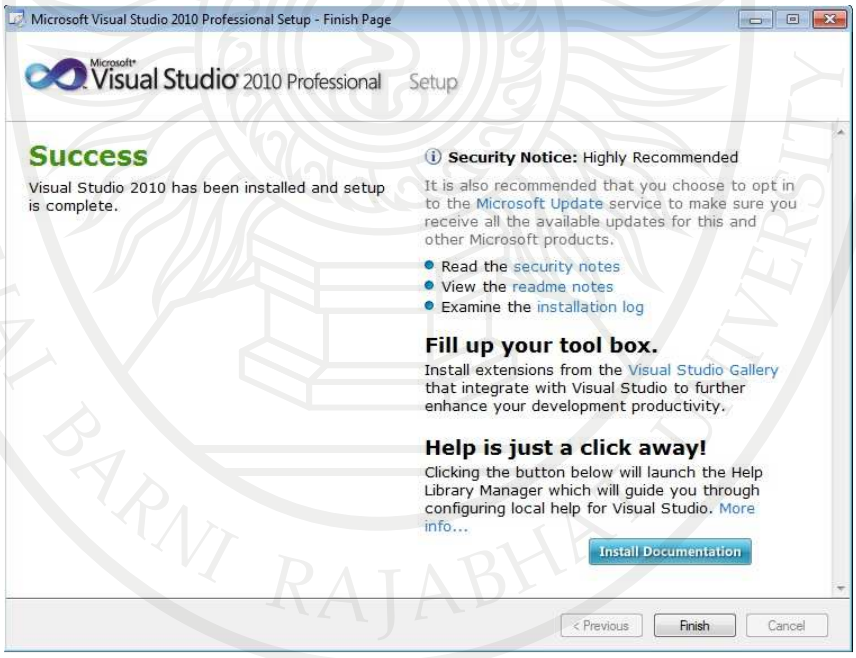
ภาพที่ 1.7 หน้าจอขณะติดตั้งโปรแกรม



ภาพที่ 1.8 หน้าจอขณะติดตั้งโปรแกรม (ต่อ)



ภาพที่ 1.9 หน้าจอการติดตั้งส่วนประกอบต่าง ๆ ของโปรแกรม



ภาพที่ 1.10 หน้าจอหลังจากติดตั้งโปรแกรมเสร็จแล้ว

หลังจากติดตั้งโปรแกรมเสร็จเรียบร้อยแล้ว สามารถใช้เปิดโปรแกรมขึ้นมาใช้งานได้ผ่านทางเมนู start ของ Windows

แนวคำตอบแบบฝึกหัดบทที่ 1

1. จงอธิบายแนวคิดและการพัฒนาระบบเชิงวัตถุ

แนวคิดเชิงวัตถุ (Object Oriented) เป็นการมองทุกสิ่งในระบบเป็นวัตถุทั้งหมด และใช้วัตถุเป็นตัวหลักในการพิจารณาความเป็นจริงที่เกิดขึ้นในระบบ ซึ่งทุก ๆ กิจกรรมที่เกิดขึ้นในระบบเกิดจาก ความสัมพันธ์และปฏิสัมพันธ์ระหว่างวัตถุ

การพัฒนาระบบเชิงวัตถุ ใช้หลักการจัดแบ่งประเภทของวัตถุออกเป็นกลุ่ม เรียกว่าคลาส (Class) ซึ่งทำงานร่วมกัน แต่ละคลาสจะมีหน้าที่ความรับผิดชอบที่แตกต่างกัน โดยแต่ละคลาสจะมีคุณสมบัติ และพฤติกรรม (Behavior) ตามบทบาทของตน ข้อมูลรายละเอียดหรือคุณสมบัติที่เก็บซ่อนในคลาสจะไม่มีการปะปนกับคลาสอื่น ๆ แต่สามารถติดต่อสื่อสารหรือการร้องขอใช้บริการได้ด้วยข้อความ (Message)

2. การพัฒนาระบบเชิงวัตถุแตกต่างจากการพัฒนาระบบเชิงโครงสร้างอย่างไร

แนวคิดเชิงโครงสร้างเป็นโครงสร้างที่โปรแกรมกับข้อมูลนั้นแยกออกจากกัน การวิเคราะห์ข้อมูลอาจจะใช้แผนภาพกระแสข้อมูล (Data Flow Diagram) และอีอาร์ไดอะแกรม (ERDiagram) เพื่อดูเส้นทางของข้อมูล แต่แนวคิดเชิงวัตถุนั้นจะมองเป็นวัตถุ และเป็นแหล่งรวบรวมข้อมูล (Data) วิธีการ (Method) โดยมีคลาสเป็นตัวกำหนดคุณสมบัติของวัตถุนั้น ซึ่งยังสามารถสืบทอด (Inheritance) คุณสมบัติต่าง ๆ ออกมาในลักษณะของคลาสย่อย (Subclass) ได้ ดังนั้นหากมีคลาสที่เป็นต้นแบบอยู่แล้ว ก็สามารถนำคุณสมบัติของคลาสต้นแบบนั้นมาใช้งานได้ทันที ซึ่งเป็นการนำกลับมาใช้ใหม่ ทำให้ช่วยลดค่าใช้จ่ายและเวลาในการพัฒนา และยังสามารถปรับปรุงแก้ไขได้ง่ายอีกด้วย

3. วัตถุและคลาสมีความแตกต่างกันอย่างไร

วัตถุ คือ สิ่งที่อยู่ในระบบซึ่งเป็นผลผลิตของคลาส ประกอบไปด้วยสถานะ (State) และพฤติกรรม (Behavior) ซ่อนอยู่ สามารถเป็นได้ทั้งสิ่งที่จับต้องได้ และจับต้องไม่ได้ ซึ่งสามารถจำแนกชนิดของวัตถุได้ 3 ชนิดด้วยกัน คือ

วัตถุที่เป็นกายภาพ หมายถึง วัตถุที่อยู่รอบ ๆ ตัวเรา สามารถที่จะมองเห็นและจับต้องได้ เช่น รถยนต์ สมุด โต๊ะ คอมพิวเตอร์ เป็นต้น

วัตถุที่เป็นแนวคิด หมายถึง วัตถุที่มีอยู่จริงสามารถสัมผัสได้ หรือเป็นวัตถุที่ไม่มีตัวตนไม่สามารถสัมผัสได้ เช่น สูตรคณิตศาสตร์ สูตรเคมี เป็นต้น

วัตถุที่เป็นซอฟต์แวร์ หมายถึง วัตถุที่มีพฤติกรรมและคุณสมบัติเก็บไว้ภายในเครื่องคอมพิวเตอร์ เป็นวัตถุที่ไม่มีตัวตนไม่สามารถสัมผัสได้ แต่สามารถทำงานผ่านเครื่องคอมพิวเตอร์ได้

คลาส คือ กลุ่มของวัตถุที่มีโครงสร้างพื้นฐานและมีพฤติกรรมแบบเดียวกัน โดยวัตถุที่มีคุณสมบัติแบบนี้ก็จะรวมกลุ่มอยู่ในคลาสเดียวกัน คลาสจะประกอบไปด้วย ชื่อคลาส (Class Name) คุณสมบัติ (Attributes) และการกระทำ (Operations หรือ Methods)

4. ยูเอ็มแอลประกอบด้วยไดอะแกรมอะไรบ้าง

ยูสเคสไดอะแกรม (Use Case Diagram)

คลาสไดอะแกรม (Class Diagram)
ออปเจคต์ไดอะแกรม (Object Diagram)
ซีเควนไดอะแกรม (Sequence Diagram)
คอลลาโบเรชันไดอะแกรม (Collaboration Diagram)
สเตทชาร์ทไดอะแกรม (Statechart Diagram)
แอคทิวิทีไดอะแกรม (Activity Diagram)
คอมโพเนนท์ไดอะแกรม (Component Diagram)
ดีพลอยเมนท์ไดอะแกรม (Deployment Diagram)

5. ความสัมพันธ์ระหว่างแอคเตอร์กับยูสเคสมีกี่แบบ อะไรบ้าง

ความสัมพันธ์เชิงโครงสร้าง (Association)
แทนด้วย ————————

- ความสัมพันธ์แบบสืบทอด (Generalization)
แทนด้วย ————▷

- ความสัมพันธ์แบบพึ่งพา (Dependency)
แทนด้วย ----------

6. ความสัมพันธ์แบบแอกกรีเกชันกับคอมโพสิชันต่างกันอย่างไร

ความสัมพันธ์แบบแอกกรีเกชันเป็นความสัมพันธ์เชิงโครงสร้างที่มีคลาสที่ใหญ่ที่สุดเป็นหลัก และมีคลาสอื่น ๆ เป็นส่วนประกอบทำให้สามารถสร้างคลาสขนาดใหญ่ได้ เช่น คลาสของรถยนต์ ประกอบด้วยคลาสของเครื่องยนต์ คลาสของล้อรถ คลาสของตัวถังรถ เป็นต้น ลักษณะของความสัมพันธ์ประเภทนี้จะให้ความสำคัญกับคลาสหลักมาก กล่าวคือถ้าคลาสหลักถูกทำลาย คลาสที่เป็นส่วนประกอบย่อยจะถูกทำลายด้วย แต่ถ้าคลาสที่เป็นส่วนประกอบถูกทำลาย คลาสหลักจะยังสามารถคงอยู่ได้

ความสัมพันธ์แบบคอมโพสิชัน เป็นความสัมพันธ์เชิงโครงสร้างเช่นกัน แต่จะแตกต่างจากแอกกรีเกชันตรงที่คลาสที่เป็นคลาสหลักและคลาสย่อยจะมีความสำคัญเท่ากัน กล่าวคือ ไม่ว่าคลาสหลักหรือคลาสย่อยถูกทำลาย คลาสที่เป็นองค์ประกอบก็จะถูกทำลายไปด้วย เช่น คลาสของคอมพิวเตอร์ ประกอบไปด้วยคลาสของซีพียู คลาสของแรม คลาสของบัส เป็นต้น

7. ไดอะแกรมที่เขียนคล้ายกับการเขียนผังงานคือไดอะไรแกรมอะไร

แอคทิวิทีไดอะแกรม (Activity Diagram) เป็นไดอะแกรมแสดงกิจกรรมหรือขั้นตอนในการทำงาน ซึ่งจะมีลักษณะคล้ายกับผังงาน (Flowchart) แต่จะแตกต่างกันตรงที่แอคทิวิทีไดอะแกรมจะสามารถบอกได้ว่ากิจกรรมเกิดขึ้นที่ใดหรือใครเป็นผู้ทำกิจกรรมนั้น และผลจากการทำงานในขั้นตอนต่าง ๆ ในระบบ

8. ซีเควนไดอะแกรมใช้ในการอธิบายอะไร

ซีเควนไดอะแกรมเป็นไดอะแกรมที่ใช้ในการอธิบายขั้นตอนการทำงานของแต่ละยูสเคสระหว่างออบเจ็กต์ต่าง ๆ ที่ส่งข้อความ (Message) ถึงกันและกัน ซึ่งจะช่วยให้ผู้พัฒนาโปรแกรมสามารถมองเห็นภาพรวมของขั้นตอนในการทำงานระหว่างวัตถุที่มีความสัมพันธ์กันในแต่ละยูสเคส ข้อความที่วัตถุส่งให้กันจะเป็นการกระทำที่วัตถุนั้น ๆ มีบริการให้

9. คลาสไดอะแกรมและซีเควนไดอะแกรมมีความสัมพันธ์กันอย่างไร

ในการเขียนซีเควนไดอะแกรมจำเป็นจะต้องมีคลาสและวัตถุเข้ามาเกี่ยวข้อง ซึ่งคลาสนั้นจะต้องมีอยู่ในคลาสไดอะแกรม ส่วนวัตถุที่เกิดขึ้นในซีเควนไดอะแกรมก็ต้องเกิดมาจากคลาสที่อยู่ในคลาสไดอะแกรม

10. จงสร้างยูสเคสไดอะแกรมเพื่ออธิบายการลงทะเบียนเรียนของนักศึกษา ซึ่งเกิดจากผลของการวิเคราะห์ความต้องการเบื้องต้น สามารถเขียนเป็นรายการได้ดังนี้

ในแต่ละภาคการศึกษาจะมีการลงทะเบียนของนักศึกษา

การลงทะเบียนในแต่ละครั้งจะมีการเก็บหลักฐานและค่าเล่าเรียน

เจ้าหน้าที่ของสถาบันการศึกษาจะเป็นผู้จัดการในเรื่องของการจัดเก็บหลักฐานและค่าเล่าเรียนทั้งหมด และผู้จ่ายเงินต้องเป็นนักศึกษาเท่านั้น

แนวคำตอบแบบฝึกหัดบทที่ 2

1. จงอธิบายประวัติความเป็นมาของภาษาเชิงวัตถุ

ในการพัฒนาระบบโดยทั่วไปนั้น ระบบที่มีความซับซ้อนจะประกอบไปด้วยวัตถุเป็นจำนวนมากและมีความสัมพันธ์เกี่ยวข้องกัน การจัดการกับความซับซ้อนนี้จะต้องมีการแยกวัตถุออกจากกันโดยสร้างแบบจำลองการทำงานของวัตถุแต่ละชนิดไปเรื่อย ๆ ซึ่งการพัฒนาโปรแกรมรูปแบบนี้จะต้องสร้างโปรแกรมที่จำลองวัตถุแต่ละชนิดไปทีละขั้นจนกว่าจะได้คำตอบ ทำให้เกิดแนวคิดที่เรียกว่า การพัฒนาโปรแกรมเชิงวัตถุ โดยภาษาที่ใช้ในการพัฒนาโปรแกรมเชิงวัตถุภาษาแรกคือ ภาษาซิมูลา (Simula) เกิดเมื่อปี ค.ศ.1960 ต่อมาในปี ค.ศ.1967 ภาษาซิมูลา 67 (Simula67) จึงถูกพัฒนาขึ้นที่ประเทศนอร์เวย์ เพื่อช่วยในการพัฒนาโปรแกรมสำหรับการจำลอง และในปี ค.ศ.1970 ภาษาสมอลล์ทอล์ค (Smalltalk) ก็กำเนิดขึ้นและเข้ามาแทนที่ภาษาซิมูลาทำให้หลายคนเข้าใจว่าภาษาสมอลล์ทอร์คเป็นภาษาเชิงวัตถุอย่างแท้จริง

2. จงอธิบายแนวคิดของภาษาเชิงวัตถุ

ในการแก้ไขปัญหาในระบบเชิงวัตถุจะมุ่งเน้นการมองปัญหาและองค์ประกอบของปัญหาในลักษณะที่เรียกว่าพรอบเบรมสเปซ (Problem space) ซึ่งเหมือนการจำลองตามสภาพความเป็นจริงในชีวิตความเป็นอยู่ของมนุษย์ที่ประกอบไปด้วยสิ่งต่าง ๆ เช่น คน สัตว์ ต้นไม้ สิ่งของ และใช้สิ่งเหล่านั้นในการแก้ไขปัญหา

3. ภาษาเชิงฟังก์ชันแตกต่างจากภาษาเชิงวัตถุอย่างไร

หลักการของภาษาเชิงฟังก์ชันและภาษาเชิงวัตถุมีดังนี้

คุณสมบัติ	วิธีการเชิงฟังก์ชัน	วิธีการเชิงวัตถุ
ลักษณะทั่วไป	นำปัญหามาแตกเป็นส่วนย่อย ให้อยู่ในรูปของกระบวนการทำงาน	มองสิ่งต่าง ๆ ในระบบเป็นวัตถุที่มีความเป็นอิสระต่อกัน แต่สามารถทำงานร่วมกันได้
ลักษณะการจำแนกงาน	แตกกระบวนการทำงานออกเป็นหน่วยย่อยที่เรียกว่า ฟังก์ชัน	จำแนกวัตถุออกเป็นกลุ่มตามคุณลักษณะของวัตถุ
ความขึ้นต่อกัน	ฟังก์ชันการทำงานจะทำงานขึ้นตรงต่อกัน มีการส่งพารามิเตอร์จากฟังก์ชันหนึ่งไปยังอีกฟังก์ชันหนึ่ง	แต่ละวัตถุมีความเป็นอิสระไม่ขึ้นตรงต่อกัน และติดต่อกันผ่านทางข้อความ
ขั้นตอนการทำงาน	เริ่มต้นที่การกำหนดโครงสร้างและประเภทของข้อมูล เพื่อใช้ในการทำงาน	เริ่มต้นที่การกำหนดคุณสมบัติและพฤติกรรมของวัตถุ และสร้างความสัมพันธ์ระหว่างวัตถุให้ทำงานร่วมกัน

4. คลาสในภาษาเชิงวัตถุหมายถึงอะไร

ในการเขียนโปรแกรมเชิงวัตถุจะต้องสร้างคลาสขึ้นมาก่อน แล้วจึงสร้างวัตถุจากคลาส ดังนั้นวัตถุจะต้องเกิดจากคลาสใดคลาสหนึ่ง วัตถุที่เกิดจากคลาสเดียวกันจะมีคุณสมบัติพื้นฐานเหมือนกัน

5. คุณสมบัติการสืบทอดและการพ้องรูปต่างกันอย่างไร

การสืบทอด (Inheritance) เป็นการถ่ายทอดคุณสมบัติหรือความสามารถของวัตถุชิ้นหนึ่ง ไปยังวัตถุอีกชิ้นหนึ่ง โดยมีลักษณะพื้นฐานตามของเดิมและเพิ่มสิ่งใหม่เข้าไปได้

การพ้องรูป (Polymorphism) คือ การถ่ายทอดคุณสมบัติเหมือนกับการสืบทอด เพียงแต่การทำงานของคลาสลูกแต่ละคลาสไม่เหมือนกัน จึงไม่สามารถกำหนดการทำงานไว้ในคลาสแม่ได้ คลาสแม่จะกำหนดเพียงแค่โครงสร้างเท่านั้น ส่วนการกำหนดรายละเอียดการทำงานของเมธอดจะให้คลาสลูกกำหนดเอง ทำให้เกิดที่มาของหนึ่งรูปหลายพฤติกรรม

6. วิธีการเข้าถึงข้อมูลซึ่งเป็นกลไกป้องกันข้อมูลมีกี่แบบ อะไรบ้าง

public (+) หมายถึง สามารถเข้าถึงได้โดยตรงจากคลาสภายนอก

private () หมายถึง สามารถเข้าถึงหรือถูกใช้งานจากภายในคลาสเท่านั้น

protected (#) หมายถึง สามารถเห็นหรือเข้าถึงได้จากภายในคลาสย่อยเท่านั้น

7. การห่อหุ้มคืออะไร มีประโยชน์อย่างไร

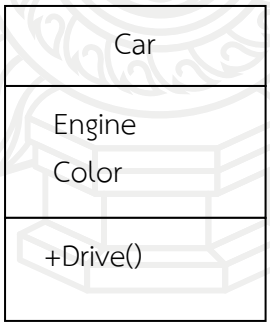
การห่อหุ้ม (Encapsulation) คือ การรวมคุณสมบัติและพฤติกรรมของวัตถุเข้าด้วยกัน โดยกำหนดเป็นชนิดของวัตถุ ทำให้สามารถกำหนดเป็นคลาสที่จะใช้ในการทำงานของโปรแกรมได้

8. จงอธิบายการวาดรูปเรขาคณิตเป็นภาษาเชิงวัตถุ

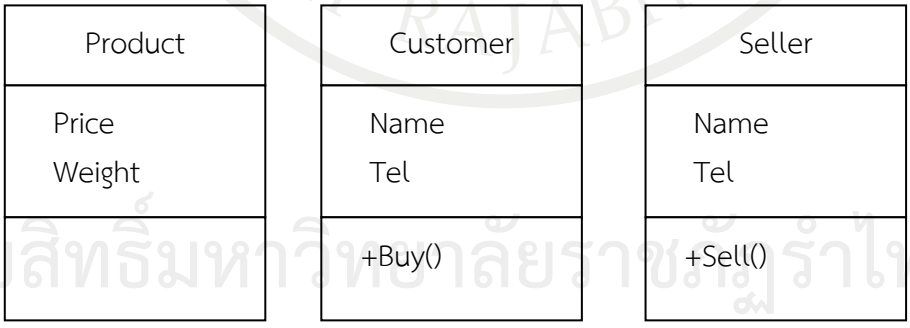
วิเคราะห์ลักษณะของรูปเรขาคณิต คือ ความกว้าง ความยาว และพื้นที่

วิเคราะห์พฤติกรรมของวัตถุ คือ คำนวณพื้นที่ คำนวณเส้นรอบรูป

9. จงออกแบบคลาสของรถยนต์



10. จงออกแบบคลาสตัวอย่างของระบบซื้อขาย



แนวคำตอบแบบฝึกหัดบทที่ 3

1. จงหาผลลัพธ์จากคำสั่งต่อไปนี้

   1.1 Console.Write(3 + 2);

   5

   1.2 Console.Write(50 – 4 * 2);

   42

   1.3 Console.Write((40 + 4) / 4);

   11

   1.4 Console.Write(3 / 2);

   1

   1.5 Console.Write(4.0 / 3);

   1.3333333

2. จงประกาศตัวแปรหรือค่าคงที่ตามที่กำหนดให้ โดยเลือกใช้ชนิดข้อมูลที่เหมาะสม

   2.1 ค่าคงที่ age เพื่อใช้แทนอายุลูกค้า

   int age;

   2.2 ตัวแปร temp เพื่อเก็บอุณหภูมิ

   double temp;

   2.3 ค่าคงที่ PI เพื่อแทนค่า 3.1415926535

   const double PI = 3.1415926535;

   2.4 ตัวแปร name เพื่อเก็บค่า Peter

   string name;

3. ประเภทข้อมูลที่ไม่ใช่ Numeric มีอะไรบ้าง

Data type	ขนาด	ค่าของข้อมูล
char (System.Char)	2 bytes	ตัวอักษรแบบ Unicode มีเครื่องหมาย ' (single quote) คร่อมตัวอักษร เช่น '1', 'A'
string (System.String)	ไม่แน่นอน	ตัวอักษรแบบ Unicode หลายตัวมารวมกัน มีเครื่องหมาย " (double quote) คร่อม เช่น "Welcome"
bool (System.Boolean)	1 bit	มีค่าที่เป็นไปได้ 2 ค่า คือ จริง (true) และ เท็จ (false)

4. การแปลงชนิดข้อมูลมีกี่วิธี

การแปลงแบบอิมพลิซิท (Implicit conversion) วิธีนี้เป็นการแปลงข้อมูลโดยไม่ชัดเจน เป็นการแปลงข้อมูลประเภทเดียวกัน แต่ใช้พื้นที่การเก็บไม่เท่ากัน ซึ่งจะแปลงได้ถูกต้องก็ต่อเมื่อแปลงจากชนิดที่ใช้พื้นที่ขนาดเล็กไปสู่ชนิดที่ใช้พื้นที่ใหญ่กว่า

การแปลงแบบเอ็กพลิซิท (Explicit conversion) วิธีนี้เป็นการแปลงข้อมูลให้เกิดความชัดเจน โดยการระบุชนิดข้อมูลไว้หน้าชื่อตัวแปรที่จะแปลง ซึ่งจะครอบด้วยวงเล็บเพื่อแยกชนิดข้อมูลชื่อกับตัวแปร

การใช้เมธอด Parse จะใช้แปลงข้อมูลชนิดสตริง (string) ไปเป็นค่าตัวเลขที่ตรงกับข้อความนั้น ๆ ซึ่งจะทำงานตรงข้ามกับเมธอด toString ที่ใช้สำหรับแปลงค่าจากตัวเลขไปเป็นสตริง

การใช้งานอ็อบเจคต์คอนเวิร์ท (Object Convert) จะมีเมธอดที่ใช้ในการแปลงข้อมูลชนิดหนึ่งไปสู่ข้อมูลอีกชนิดหนึ่งที่ต้องการ

5. การทำงานแบบ Console Application และ Windows Forms Application ต่างกันอย่างไร

การเขียนโปรแกรมแบบ Console Application เป็นการเขียนโปรแกรมที่มีลักษณะการแสดงผลแบบคอมมานด์ไลน์ (Command Line) และมีรูปแบบการเขียนเป็นแบบประมวลคำสั่ง จึงต้องมีการจัดลำดับของการแสดงผลให้ชัดเจน

การเขียนโปรแกรมแบบ Windows Forms Application จะมีส่วนหลักคือฟอร์ม ซึ่งเป็นส่วนที่ใช้ติดต่อกับผู้ใช้งาน ภายในฟอร์มจะประกอบด้วยอ็อบเจคต์ (Object) และคอนโทรล (Control) ที่ใช้เป็นเครื่องมือติดต่อกับผู้ใช้งาน มีหน้าที่ต่างกัน เช่น Label ใช้แสดงข้อความ Textbox ใช้รับข้อความ Button ปุ่มคำสั่ง และอาจจะมีคอนโทรลบางประเภทที่ซ่อนอยู่โดยไม่แสดงบนหน้าจอ

6. Console.Write และ Console.WriteLine ต่างกันอย่างไร

Console.Write (ข้อความ); ใช้สำหรับแสดงข้อมูลทางจอภาพโดยที่เคอร์เซอร์จะอยู่ต่อท้ายที่บรรทัดเดิม

Console.WriteLine (ข้อความ); ใช้สำหรับแสดงข้อมูลทางจอภาพโดยที่เคอร์เซอร์จะเลื่อนลงมาอยู่อีกบรรทัดนึง

7. โครงสร้างของโปรแกรมภาษาวิชวลซีชาร์ปมีเงื่อนไขอย่างไร

ในการเขียนโปรแกรมด้วยภาษาวิชวลซีชาร์ปจำเป็นจะต้องทราบโครงสร้างของภาษาก่อนเพื่อให้เข้าใจในการทำงานของภาษา โดยการเขียนโปรแกรมวิชวลซีชาร์ปจะต้องมีโครงสร้างและเงื่อนไขดังนี้

โปรแกรมสามารถประกอบด้วยเนมสเปซ (Namespaces) อย่างน้อย 1 เนมสเปซ
เนมสเปซประกอบด้วยคลาส อย่างน้อย 1 คลาส
คลาสประกอบด้วยเมธอด อย่างน้อย 1 เมธอด

8. MessageBoxButtons.OKCancel ใช้สำหรับแสดงปุ่มอะไรใน MessageBox

ปุ่ม OK และปุ่ม Cancel

9. จงเขียนโปรแกรมคำนวณพื้นที่วงกลมและเส้นรอบวง โดยให้สามารถรับค่ารัศมีจากผู้ใช้ได้ โดยกำหนดค่าคงที่ชื่อ PI ใช้แทนค่า 3.141

ตัวอย่างหน้าจอแสดงผล

Enter Radius : 4

Radius of Circle = 4

Area of Circle = 50.256
Round of Circle = 25.128

ชุดคำสั่งมีดังนี้

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Ex4_1
{
    class Program
    {
        static void Main(string[] args)
        {
            const double PI = 3.141;
            double radius;
            Console.Write("Enter Radius : ");
            radius = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Radius of Circle = {0}",
Convert.ToDouble(radius));
            Console.WriteLine("Area of Circle = {0}", Convert.ToString(PI *
radius * radius));
            Console.WriteLine("Round of Circle = {0}", Convert.ToString(2 * PI
* radius));
            Console.ReadKey();
        }
    }
}
```

10. ให้เขียนโปรแกรมแสดง MessageBox โดยมีเงื่อนไขดังนี้
    คำถามข้อความว่า “คุณต้องการบันทึกการทำงานหรือไม่ ?”
    แสดงข้อความที่ TitleBar ว่า “ยืนยันการบันทึกข้อมูล”
    โดยให้แสดงปุ่ม Yes, No และ Cancel
    แสดง Icon รูป 

ชุดคำสั่งมีดังนี้

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace ExMessageBox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void ShowMessageBox_Click(object sender, EventArgs e)
        {
            MessageBox.Show("คุณต้องการบันทึกการทำงานหรือไม่ ?", "ยืนยันการ
บันทึกข้อมูล", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning);
        }
    }
}
```

แนวคำตอบแบบฝึกหัดบทที่ 4

1. การเขียนโปรแกรมแบบอีเวนท์ไดร์ฟเวนหมายถึงอะไร

การทำงานต่าง ๆ บนฟอร์มจะทำงานในลักษณะที่ผู้ใช้จะเป็นคนทำให้เกิดเหตุการณ์หรือเกิดอีเวนท์ขึ้นที่วัตถุ ซึ่งเรียกรูปแบบนี้ว่าการเขียนโปรแกรมแบบอีเวนท์ไดร์ฟเวนโดยเมื่อเกิดอีเวนท์แล้ววิชวลซีชาร์ปจะไปค้นหาว่ามีการเขียนคำสั่งสำหรับการทำงานนั้นหรือไม่ ถ้ามีก็จะทำงานตามคำสั่งที่กำหนดไว้ แต่ถ้าไม่มีก็จะเหมือนกับไม่มีอะไรเกิดขึ้น

2. ถ้าต้องการสร้างฟอร์มสำหรับรับข้อมูลลูกค้า ต้องใช้ Common Controls อะไรบ้าง

Label เป็นส่วนที่ในการสร้างป้ายข้อความในฟอร์มเพื่อบอกให้รู้ว่าส่วนนี้เป็นส่วนใดหรือต้องการให้ใส่ข้อมูลอะไร ส่วนใหญ่จะทำงานร่วมกับ TextBox

TextBox เป็นส่วนที่ใช้สำหรับกรอกข้อมูลต่าง ๆ เพื่อใช้ในการประมวลผล จะทำงานร่วมกัน Label เพื่อให้รู้ว่าต้องใส่ข้อมูลอะไร

Button เป็นส่วนที่ใช้สร้างปุ่มที่ทำงาน โดยแต่ละปุ่มจะมีการทำงานที่ต่างกันออกไปตามการเขียนคำสั่งควบคุมการทำงาน

3. การออกแบบสำหรับรับข้อมูลเพศ ควรใช้ Common Controls ใด

RadioButton เป็นส่วนที่ใช้สร้างทางเลือกคล้ายกับ CheckBox แต่ผู้ใช้สามารถเลือกได้เพียง 1 ตัวเลือกจากกลุ่มตัวเลือกกลุ่มหนึ่ง

4. ในส่วนแถบชื่อเรื่องของฟอร์มสามารถตั้งค่าได้จาก Properties ใด

Text ใช้สำหรับกำหนดข้อความเพื่อแสดงที่แถบชื่อเรื่อง (TitleBar)

5. เมธอดที่สำคัญของฟอร์มมีอะไรบ้าง

ชื่อของเมธอด	การทำงาน
Show	เป็นเมธอดที่สำหรับเรียกฟอร์มขึ้นมาแสดงผลมีผลเช่นเดียวกับการกำหนด Property ของ Visible = True
ShowDialog	เป็นเมธอดที่เรียกฟอร์มแสดงผลแบบไดอะลอก (Dialog) เมื่อเรียกขึ้นมาแล้วต้องปิดฟอร์มนี้ก่อนจึงจะไปทำงานในฟอร์มอื่นได้
Hide	ใช้สำหรับซ่อนฟอร์ม
Activate	ใช้สำหรับเรียกฟอร์มให้พร้อมสำหรับการทำงาน
SetDesktopLocation	ใช้สำหรับกำหนดตำแหน่งพิกัดที่จะแสดงบนหน้าจอของฟอร์ม
Close	ใช้สำหรับปิดการทำงานของฟอร์ม

6. ถ้าต้องการเขียนคำสั่งขณะฟอร์มกำลังปิด ต้องเขียนที่อีเวนท์ใด

Close ใช้สำหรับปิดการทำงานของฟอร์ม

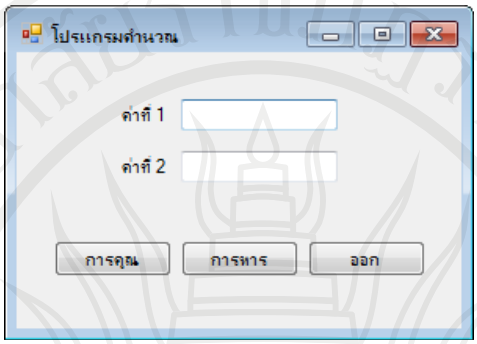
7. ComboBox ใช้สำหรับออกแบบฟอร์มแบบใดได้บ้าง

ComboBox สร้างทางเลือกให้ผู้ใช้คล้ายกับ ListBox และผู้ใช้สามารถพิมพ์ข้อมูลเพิ่มเข้าไปได้อีกด้วย เช่น ฟอร์มสำหรับการใส่คำนำหน้าชื่อ ฟอร์มสำหรับการบอกวุฒิการศึกษา เป็นต้น

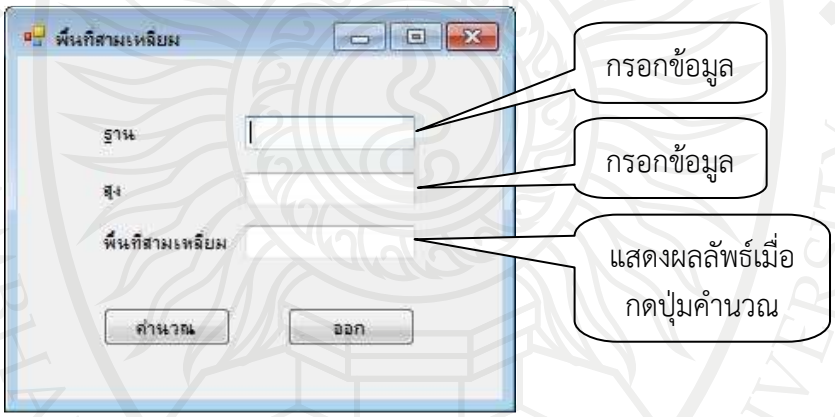
8. อีเวนท์ Resize จะเกิดขึ้นเมื่อใด

Resize เกิดขึ้นเมื่อฟอร์มถูกปรับขนาดให้เปลี่ยนไป

9. จงออกแบบหน้าจอรับข้อมูลจากข้อมูลต่อไปนี้
   รับข้อมูลค่าตัวเลข 2 ตัว
   มีปุ่มสำหรับการคูณ ปุ่มสำหรับการหาร และปุ่มสำหรับออกจากโปรแกรม



10. จงเขียนโปรแกรมคำนวณพื้นที่สามเหลี่ยมโดยมีรายละเอียดดังนี้
    ออกแบบหน้าจอดังตัวอย่าง



กำหนด Properties สำหรับ Form
- Name กำหนดเป็น frmTriangle
- Text กำหนดเป็น “พื้นที่สามเหลี่ยม”

กำหนด Properties สำหรับ Label1
- Name กำหนดเป็น lblBase
- Text กำหนดเป็น “ฐาน”

กำหนด Properties สำหรับ Label2
- Name กำหนดเป็น lblHigh
- Text กำหนดเป็น “สูง”

กำหนด Properties สำหรับ Label3
- Name กำหนดเป็น lblArea
- Text กำหนดเป็น “พื้นที่สามเหลี่ยม”

กำหนด Properties สำหรับ TextBox1
Name กำหนดเป็น txtBase
Text ใส่เป็นค่าว่าง
กำหนด Properties สำหรับ TextBox2
Name กำหนดเป็น txtHigh
Text ใส่เป็นค่าว่าง
กำหนด Properties สำหรับ TextBox3
Name กำหนดเป็น txtArea
Text ใส่เป็นค่าว่าง
กำหนด Properties สำหรับ Button1
Name กำหนดเป็น btnCalculate
Text กำหนดเป็น “คำนวณ”
กำหนด Properties สำหรับ Button2
Name กำหนดเป็น btnExit
Text กำหนดเป็น “ออก”

ชุดคำสั่งมีดังนี้

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Ex
{
    public partial class frmTriangle : Form
    {
        public frmTriangle ()
        {
            InitializeComponent();
        }

        private void btnCalculate_Click(object sender, EventArgs e)
```

```
            {
                txtArea.Text = Convert.ToString(int.Parse(txtBase.Text) *
int.Parse(txtHigh.Text));
            }

            private void btnExit_Click(object sender, EventArgs e)
            {
                this.Close();
            }
        }
    }
```

แนวคำตอบแบบฝึกหัดบทที่ 5

1. การเขียนคำสั่งควบคุมการทำงานของโปรแกรมมีกี่แบบ อะไรบ้าง

   การควบคุมการทำงานของโปรแกรมมีอยู่ 2 แบบ คือ

   การตัดสินใจ (Decision)

   การทำงานซ้ำ (Iteration)

2. คำสั่ง if...else ใช้สำหรับทำอะไร

   การเขียนโปรแกรมด้วยคำสั่ง if...else จะมีเงื่อนไขที่ตรงตัว คือ ถ้าเงื่อนไขที่อยู่ใน if เป็นจริงจะต้องทำคำสั่งที่อยู่หลัง if แต่ถ้าเงื่อนไขที่อยู่ใน if เป็นเท็จ จะต้องทำคำสั่งที่อยู่หลัง else แต่ในกรณีที่ไม่มี else ตามหลัง if จะแสดงว่าถ้าเงื่อนไขเป็นเท็จก็ไม่ต้องทำคำสั่งใด ๆ

3. ถ้าต้องการเขียนโปรแกรมที่มีหลายทางเลือก สามารถใช้คำสั่งอะไรได้บ้าง

   คำสั่ง if...else

   คำสั่ง switch

4. จงเขียนรูปแบบของคำสั่ง switch

   รูปแบบเป็นดังนี้

```
switch ( กรณี )
{
   case กรณี_1 :
        <ชุดคำสั่ง_1>;
        break;
   case กรณี_2 :
        <ชุดคำสั่ง_2>;
        break;
        ...
        ...
   case กรณี_N :
        <ชุดคำสั่ง_N>;
        break;
   default :
        <ชุดคำสั่ง_M>;
        break;
}
```

5. คำสั่งในการวนซ้ำที่มีจำนวนรอบที่แน่นอนคือคำสั่งใด มีรูปแบบอย่างไร

   คำสั่ง for มีรูปแบบดังนี้

   for (<ค่าตัวแปรเริ่มต้น>; <เงื่อนไขที่กำหนดการหยุดวนซ้ำ>; <การเพิ่มหรือลดค่าของตัวแปร>)

6. คำสั่ง while มีรูปแบบอย่างไร

มีรูปแบบดังนี้

```
while (เงื่อนไข)
  คำสั่ง_1;
```

หรือ

```
while (เงื่อนไข)
{
  คำสั่ง_1;
  คำสั่ง_2;
  ...
  คำสั่ง_N;
}
```

7. คำสั่ง if...else และ switch ต่างกันอย่างไร

การเขียนโปรแกรมด้วยคำสั่ง if...else จะมีเงื่อนไขที่ตรงตัว คือ ถ้าเงื่อนไขที่อยู่ใน if เป็นจริงจะต้องทำคำสั่งที่อยู่หลัง if แต่ถ้าเงื่อนไขที่อยู่ใน if เป็นเท็จ จะต้องทำคำสั่งที่อยู่หลัง else แต่ในกรณีที่ไม่มี else ตามหลัง if จะแสดงว่าถ้าเงื่อนไขเป็นเท็จก็ไม่ต้องทำคำสั่งใด ๆ

คำสั่ง switch เป็นคำสั่งที่ใช้ตรวจสอบค่าที่เป็นกรณีที่เป็นเงื่อนไข ซึ่งผลที่ได้จากการตรวจสอบจะถูกนำไปเปรียบเทียบกับค่ากรณีต่าง ๆ ที่ถูกกำหนดไว้หลัง case โดยโปรแกรมจะทำงานตามคำสั่งที่หลังเครื่องหมายโคลอน (:) ทีละคำสั่งไปเรื่อย ๆ จนกว่าจะเจอคำสั่ง break และเมื่อเจอคำสั่ง break แล้วโปรแกรมจะออกจากบล็อกของคำสั่ง switch ทันที แต่ในกรณีที่การตรวจสอบเงื่อนไขไม่ตรงกับเงื่อนไขที่อยู่หลัง case เลย โปรแกรมจะมาทำงานที่คำสั่งหลัง default และทำไปเรื่อย ๆ จนกว่าจะเจอคำสั่ง break เช่นเดียวกัน

8. จงแปลงคำสั่ง if...else ต่อไปนี้ให้อยู่ในรูปแบบ switch

```
if (num < 10)
  MessageBox.Show("Num น้อยกว่า 10");
else if (num < 50)
  MessageBox.Show("Num มีค่าตั้งแต่ 10 ขึ้นไปแต่ไม่เกิน 50");
else
  MessageBox.Show("Num มีค่าตั้งแต่ 50 ขึ้นไป");
```

ชุดคำสั่งเป็นดังนี้

```
switch (num)
{
  case num < 10 :
      MessageBox.Show("Num น้อยกว่า 10");
```

```
            break;
        case num < 50 :
            MessageBox.Show("Num มีค่าตั้งแต่ 10 ขึ้นไปแต่ไม่เกิน 50");
            break;
        default :
            MessageBox.Show("Num มีค่าตั้งแต่ 50 ขึ้นไป");
            break;
    }
```

9. จงแปลงคำสั่ง for ต่อไปนี้ให้อยู่ในรูปแบบ do...while

```
for (int i = 1; i < 5; i++)
  MessageBox.Show("Hello");
```

ชุดคำสั่งเป็นดังนี้

```
int i = 1;
do
{
  MessageBox.Show("Hello");
  i++;
} while (i<5);
```

10. จงเขียนโปรแกรมตัดเกรด โดยมีเงื่อนไขดังนี้

รับข้อมูลค่าคะแนนตั้งแต่ 0 100

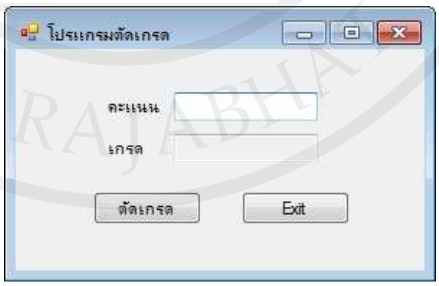
คะแนน 0 – 50 ได้เกรด F

คะแนน 50 – 59 ได้เกรด D

คะแนน 60 – 69 ได้เกรด C

คะแนน 70 – 79 ได้เกรด B

คะแนน 80 – 100 ได้เกรด A



ชุดคำสั่งเป็นดังนี้

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Ex
{
    public partial class frmEx : Form
    {
        public frmEx()
        {
            InitializeComponent();
        }

        private void btnGrade_Click(object sender, EventArgs e)
        {
            int grd = Convert.ToInt16(txtPoint.Text);
            if (grd >= 0 && grd <= 59)
                txtGrade.Text = "Fail";
            else if (grd >= 60 && grd <= 79)
                txtGrade.Text = "Pass";
            else if (grd >= 80 && grd <= 100)
                txtGrade.Text = "Good";
            else
                txtGrade.Text = "Error";
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

แนวคำตอบแบบฝึกหัดบทที่ 6

1. โครงสร้างข้อมูลแบบแถวลำดับและคอลเลคชันแตกต่างกันอย่างไร

เป็นโครงสร้างข้อมูลแบบอาร์เรย์ (array) ที่มีการประกาศตัวแปรเพื่อใช้สำหรับเก็บข้อมูลที่เป็นชนิดเดียวกัน มีลักษณะโครงสร้างเป็นแบบแถวลำดับ ซึ่งสามารถกำหนดชื่อตัวแปรเดียวแต่เก็บข้อมูลได้หลายค่า โดยใช้ตัวเลขลำดับหรือดัชนี (index) เป็นตัวชี้ตำแหน่ง ทำให้สามารถเรียกใช้งานจากชื่ออ้างอิงเพียงชื่อเดียว และใช้ตัวเลขดัชนีเพื่อเข้าถึงค่าที่เก็บอยู่ในอาร์เรย์ซึ่งอาจจะเก็บอยู่ในรูปของ 1 มิติ 2 มิติ หรือหลายมิติก็ได้ การกำหนดอาร์เรย์สามารถกำหนดใช้เนื้อที่ในหน่วยความจำได้เท่ากับจำนวนหน่วยความจำที่มีอยู่ในเครื่องคอมพิวเตอร์

เป็นโครงสร้างข้อมูลที่มีการเก็บข้อมูลคล้ายกับอาร์เรย์เพียงแต่มีความยืดหยุ่นสูงกว่า เนื่องจากไม่จำเป็นต้องกำหนดจำนวนของอาร์เรย์แต่สามารถเพิ่มข้อมูลได้เรื่อย ๆ ซึ่งเหมาะสำหรับการจัดการข้อมูลแบบสแตก (stacks) คิว (queues) ลิสต์ (lists) และตารางแฮช (hash tables)

2. จงอธิบายการทำงานของโครงสร้างข้อมูลแบบ Array

เป็นโครงสร้างข้อมูลที่มีการประกาศตัวแปรเพื่อใช้สำหรับเก็บข้อมูลที่เป็นชนิดเดียวกัน มีลักษณะโครงสร้างเป็นแบบแถวลำดับ ซึ่งสามารถกำหนดชื่อตัวแปรเดียวแต่เก็บข้อมูลได้หลายค่า โดยใช้ตัวเลขลำดับหรือดัชนี (index) เป็นตัวชี้ตำแหน่ง ทำให้สามารถเรียกใช้งานจากชื่ออ้างอิงเพียงชื่อเดียว และใช้ตัวเลขดัชนีเพื่อเข้าถึงค่าที่เก็บอยู่ในอาร์เรย์ซึ่งอาจจะเก็บอยู่ในรูปของ 1 มิติ 2 มิติ หรือหลายมิติก็ได้

3. จงกำหนดตัวแปรแบบอาร์เรย์1 มิติสำหรับเก็บข้อมูลชนิดตัวเลขจำนวน 5 ตัว

```
int[] num = new int[5];
```

4. การกำหนด int[,] n = new int[3, 2] { {1, 2}, {3, 4}, {5, 6} }; หมายถึงอะไร

เป็นการกำหนดตัวแปร n เป็นอาเรย์แบบ 3 มิติที่มีขนาดเท่ากับ 3 x 2 โดยกำหนดค่าเริ่มต้นดังนี้

ตำแหน่งที่ [0, 0] มีค่าเท่ากับ 1
ตำแหน่งที่ [0, 1] มีค่าเท่ากับ 2
ตำแหน่งที่ [1, 0] มีค่าเท่ากับ 3
ตำแหน่งที่ [1, 1] มีค่าเท่ากับ 4
ตำแหน่งที่ [2, 0] มีค่าเท่ากับ 5
ตำแหน่งที่ [2, 0] มีค่าเท่ากับ 6

5. การเก็บข้อมูลแบบสแตกเป็นอย่างไร

เป็นโครงสร้างข้อมูลที่เก็บข้อมูลในรูปแบบของอาร์เรย์ที่มีเก็บข้อมูลในลักษณะที่ข้อมูลที่เข้ามาก่อนจะอยู่ด้านในสุด เวลาต้องการนำข้อมูลออกมาใช้ ข้อมูลลำดับสุดท้ายจะถูกนำออกมาก่อน ส่วนข้อมูลที่เข้าเป็นลำดับแรกจะถูกนำออกมาเป็นลำดับสุดท้าย ซึ่งจะคล้ายกับการเก็บหนังสือไว้ในกล่อง เวลาที่จะนำออกมาต้องหยิบหนังสือจากด้านบนก่อน

6. จงกำหนดตัวแปรชื่อ myStack ให้เป็นข้อมูลแบบสแตก

```
Stack myStack = new Stack();
```

7. โครงสร้างข้อมูลแบบคิวมีการทำงานอย่างไร

เป็นโครงสร้างข้อมูลที่มีการเก็บข้อมูลเรียงต่อกันเป็นลำดับ ข้อมูลที่เข้ามาก่อนจะอยู่ด้านในสุด ส่วนข้อมูลถัดมาจะเรียงต่อกันเป็นลำดับไปเรื่อย ๆ เวลานำข้อมูลออกจะนำข้อมูลด้านในสุดออกก่อนตามลำดับจนถึงข้อมูลสุดท้าย เปรียบเสมือนการเข้าแถวรับบริการซื้อสินค้า ผู้ที่มาก่อนจะได้รับการบริการก่อนตามลำดับ

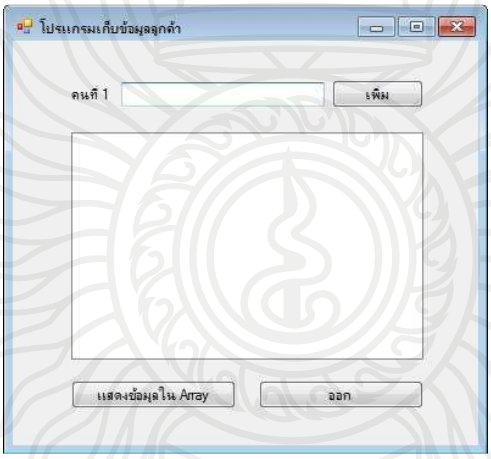
8. จงยกตัวอย่างของการใช้ข้อมูลแบบคิวในชีวิตประจำวัน มาสัก 3 ตัวอย่าง

การเข้าแถวซื้ออาหาร

การจอดรถติดไฟแดง

การเข้าแถวชำระค่าบริการ

9. จงเขียนโปรแกรมสำหรับเก็บชื่อลูกค้าจำนวน 10 คน และเก็บข้อมูลเป็นโครงสร้างข้อมูลชนิดอาร์เรย์โดยมีหน้าจอดังนี้

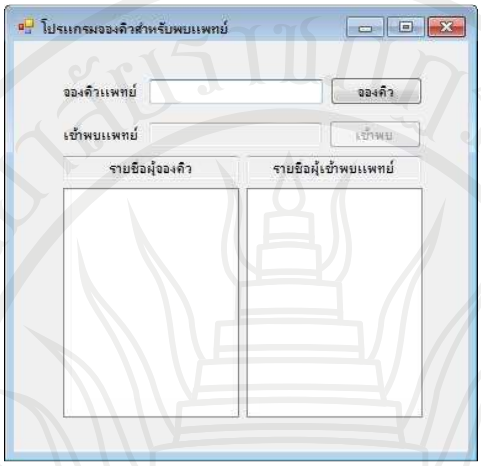


ชุดคำสั่งเป็นดังนี้

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Ex
{
    public partial class frmArray : Form
    {
```

```
    public frmArray()
    {
        InitializeComponent();
    }

    string[] name = new name[10];
    int i = 0;

    private void btnInput_Click(object sender, EventArgs e)
    {
        name[i] = txtInput.Text;
        i++;
        if (i < 10)
        {
            lblInput.Text = "รับค่าที่ " + (i + 1);
            txtInput.Text = "";
        }
        else
            btnInput.Enabled = false;
    }

    private void btnShow_Click(object sender, EventArgs e)
    {
        for (int j = 0; j < 10; j++)
            lstShow.Items.Add("ค่าที่ " + (j + 1) + " คือ " + name[j]);

    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
}
```

10. จงเขียนโปรแกรมจองคิวสำหรับพบแพทย์และเก็บชื่อของคนไข้ลงในโครงสร้างข้อมูลชนิดคิว โดยมีหน้าจอดังนี้



ชุดคำสั่งเป็นดังนี้

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Ex
{
    public partial class frmStack : Form
    {
        public frmStack()
        {
            InitializeComponent();
        }

        Stack myStack = new Stack();
        int i = 0;
```

```
        private void btnPush_Click(object sender, EventArgs e)
        {
            myStack.Push(txtInput.Text);
            lstInput.Items.Add(txtInput.Text);
            txtInput.Text = "";
            i++;
            if (i != 0)
                btnPop.Enabled = true;
        }

        private void btnPop_Click(object sender, EventArgs e)
        {

            txtOutput.Text = Convert.ToString(myStack.Pop());
            i;
            lstInput.Items.RemoveAt(i);
            lstOutput.Items.Add(txtOutput.Text);
            if (i == 0)
                btnPop.Enabled = false;
        }
    }
}
```

แนวคำตอบแบบฝึกหัดบทที่ 7

1. จงบอกรูปแบบของคำสั่งที่ใช้ในการสร้างวัตถุจากคลาส

   รูปแบบมีดังนี้

   <classname> <ชื่อวัตถุ>;

   <ชื่อวัตถุ> = new <classname>();

   หรือ

   <classname> <ชื่อวัตถุ> = new <classname>();

2. คลาสไดอะแกรมสามารถแปลงเป็นอะไรได้บ้างในการเขียนโปรแกรม

   หลังจากสร้างส่วนสำหรับเขียนคลาสขึ้นมาแล้ว ผู้เขียนจะต้องนำเอาคลาสที่ได้ออกแบบตามยูเอ็มแอลมาแปลงเป็นส่วนที่ใช้ในการเขียนโปรแกรม เช่น ส่วนติดต่อกับผู้ใช้ ฐานข้อมูล และการทำงานคำสั่ง

3. จงอธิบายการทำงานของคำสั่ง get และ set

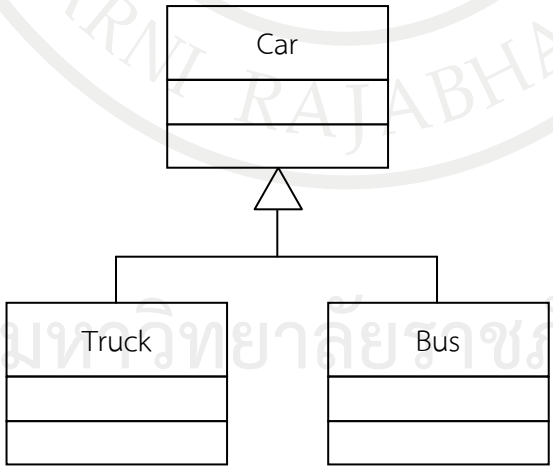
   สำหรับส่วนที่เป็นคุณสมบัติในคลาส จำเป็นจะต้องมีการป้องกันไม่ใช้คลาสจากภายนอกเข้ามาเปลี่ยนแปลงแก้ไขข้อมูลได้โดยตรง โดยส่วนใหญ่คุณสมบัติในคลาสจะมีการตั้งค่า <accessibility> เป็น private อยู่แล้ว แต่ในกรณีที่ต้องการแก้ไขข้อมูลสามารถใช้บริการโดยผ่านทาง Method get และ set ซึ่งเป็นมาตรฐานที่ใช้สำหรับการเขียนโปรแกรมเชิงวัตถุ

4. ความสัมพันธ์ระหว่างแอกกรีเกชัน (Aggregation) และคอมโพสิชัน (Composition) ต่างกันอย่างไร

   แอกกรีเกชัน เป็นความสัมพันธ์เชิงโครงสร้างเช่นเดียวกัน แต่จะมีคลาสหลักและคลาสที่เป็นองค์ประกอบแบบโฮลพาร์ทโดยคลาสหลักจะมีความสำคัญมากกว่าคลาสองค์ประกอบ

   คอมโพสิชัน เป็นความสัมพันธ์เชิงโครงสร้างเช่นเดียวกัน แต่จะมีคลาสหลักและคลาสที่เป็นองค์ประกอบแบบโฮลพาร์ทโดยคลาสหลักจะมีความสำคัญเท่ากับคลาสองค์ประกอบ

5. จงเขียนคลาสแสดงความสัมพันธ์แบบเจนเนอเรไลเซชัน (Generalization) ระหว่าง รถยนต์ รถบรรทุก รถโดยสาร



6. ดีไซน์แพทเทิร์น คืออะไร

ดีไซน์แพทเทิร์น เป็นการใช้งานรูปแบบสำเร็จรูปที่ถูกพัฒนาขึ้นเพื่อใช้ในการแก้ปัญหาที่มีลักษณะคล้ายกัน โดยรูปแบบที่ใช้เป็นลักษณะเชิงวัตถุ ผู้พัฒนาโปรแกรมสามารถแก้ไขปัญหาที่พบและใช้รูปแบบที่เหมาะสมเพื่อในการพัฒนาโปรแกรม

7. การใช้งานดีไซน์แพทเทิร์นแตกต่างจากการสร้างคลาสเองอย่างไร

คลาสที่สร้างจากดีไซน์แพทเทิร์นจะเป็นรูปแบบที่มาตรฐานกว่าคลาสที่เกิดจากการสร้างขึ้นมาเอง ซึ่งสามารถช่วยในเรื่องการพัฒนาโปรแกรมได้เป็นอย่างดี

8. ดีไซน์แพทเทิร์นสามารถแบ่งออกเป็นกี่กลุ่ม อะไรบ้าง

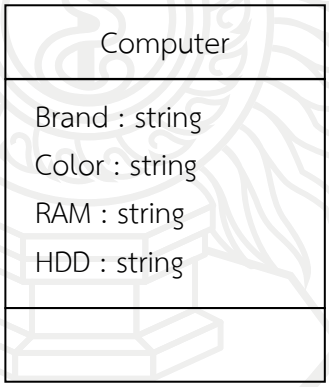
ดีไซน์แพทเทิร์นสามารถแบ่งออกเป็น 3 กลุ่มใหญ่ ๆ ได้ดังนี้

ครีเอชันนัลแพทเทิร์น (Creational Patterns) เป็นรูปแบบที่ใช้สร้างออบเจ็คต์ที่สามารถสร้างออบเจ็คต์อื่น ๆ ขึ้นมาได้ระหว่างที่กำลังทำงานอยู่

สตรัคเจอร์รัลแพทเทิร์น (Structural Patterns) เป็นรูปแบบที่ทำให้ออบเจ็คต์ที่มีอยู่สามารถทำงานร่วมกันได้

บีฮาวิโอรัลแพทเทิร์น (Behavioral Patterns) เป็นรูปแบบที่ใช้ในการควบคุมการส่งข้อมูลระหว่างออบเจ็คต์ด้วยกัน

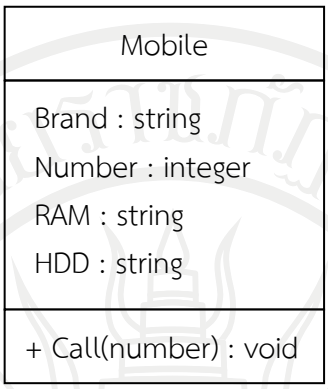
9. จงแปลงคลาสไดอะแกรมให้เป็นคำสั่งในโปรแกรม



ชุดคำสั่งเป็นดังนี้

```
class Computer
{
    public string Brand { get; set; }
    public string Color { get; set; }
    public string RAM { get; set; }
    public string HDD { get; set; }
}
```

10. จงเขียนคำสั่งในการสร้างวัตถุจากคลาสต่อไปนี้



ชุดคำสั่งเป็นดังนี้

```
class Mobile
{
    public string Brand { get; set; }
    public int Number { get; set; }
    public string RAM { get; set; }
    public string HDD { get; set; }

    public void Call()
    {
    }
}
```

แนวคำตอบแบบฝึกหัดบทที่ 8

1. ข้อผิดพลาดมีกี่ประเภท อะไรบ้าง

การแบ่งประเภทของข้อผิดพลาดที่เกิดขึ้น สามารถแบ่งตามลักษณะการเกิดและผลที่เกิดขึ้น ซึ่งโดยทั่วไปสามารถแบ่งออกเป็น 3 ประเภทหลัก ๆ ดังนี้

ข้อผิดพลาดทางไวยกรณ์ (Syntax Error)

ข้อผิดพลาดระหว่างการทำงาน (Runtime Error)

ข้อผิดพลาดทางตรรกะ (Logical Error)

2. ประเภทของข้อผิดพลาดที่มีความร้ายแรงที่สุดคืออะไร เพราะเหตุใด

ข้อผิดพลาดทางตรรกะ (Logical Error) เป็นข้อผิดพลาดที่ทำให้ไม่ได้ผลลัพธ์ตามความต้องการของโปรแกรม เกิดจากการเขียนโปรแกรมที่มีข้อผิดพลาดทางตรรกะ ถึงแม้ว่าโปรแกรมจะไม่เกิดข้อผิดพลาดอื่น แต่ได้ผลลัพธ์ไม่เป็นไปตามต้องการ ข้อผิดพลาดประเภทนี้เป็นข้อผิดพลาดที่มีความร้ายแรงและเป็นประเภทที่ตรวจพบยากที่สุด

3. การตรวจสอบคุณภาพแบบทวนสอบ (Verification) เป็นแบบใด

การตรวจสอบความถูกต้องของโปรแกรมโดยพิจารณาจากรายละเอียดของโปรแกรมที่พัฒนาขึ้นมา ซึ่งจะต้องตรงกับความต้องการของโปรแกรม เช่น สร้างโปรแกรมตรวจสอบเลขคู่และเลขคี่ เมื่อโปรแกรมทำงานแล้วจะต้องสามารถทำการตรวจสอบค่าของตัวเลขตามความต้องการได้ เป็นต้น

4. การตรวจสอบคุณภาพแบบตรวจสอบ (Validation) เป็นแบบใด

การตรวจสอบความถูกต้องจากความต้องการของผู้ใช้งาน โดยพิจารณาจากความต้องการที่ผู้ใช้แจ้งในเอกสารก่อนการพัฒนาโปรแกรม เช่น การพัฒนาโปรแกรมสำหรับเก็บข้อมูลลูกค้า ผู้ใช้ต้องการให้บังคับการเก็บข้อมูลหมายเลขบัตรประชาชนเป็นตัวเลขจำนวน 13 หลัก ถ้าโปรแกรมไม่ได้กำหนดจำนวนที่ชัดเจนในการกรอกข้อมูล จะทำให้เกิดข้อผิดพลาดขึ้นมาได้

5. วิธีการทดสอบแบบกล่องดำและกล่องขาวแตกต่างกันอย่างไร

การทดสอบแบบกล่องดำ (Black Box Testing) หรือเรียกอีกอย่างหนึ่งว่า การทดสอบเชิงพฤติกรรม (behavioral testing) เป็นการทดสอบที่ไม่สนใจว่าโปรแกรมจะทำงานอย่างไร โดยจะทดสอบข้อมูลเข้าและผลลัพธ์ที่ออกมาเท่านั้น โดยจะไม่มีการตรวจสอบการประมวลผลที่อยู่ภายในของโปรแกรม

การทดสอบแบบกล่องขาว (White Box Testing) หรือเรียกอีกอย่างหนึ่งว่า การทดสอบแบบกล่องแก้ว (Glass box testing) ซึ่งเป็นวิธีการทดสอบข้อผิดพลาดทางตรรกะ และจะตรวจสอบเส้นทางการทำงานของคำสั่งในโปรแกรมเรียกว่า การทดสอบเส้นทางมูลฐาน (Basis Path Testing) ซึ่งเป็นเทคนิคการทดสอบที่ช่วยให้ผู้เขียนโปรแกรมใช้ในการออกแบบกรณีทดสอบเส้นทางการทำงานต่าง ๆ และใช้ค่าที่สามารถวัดได้มาเป็นเกณฑ์ในการกำหนดชุดทดสอบของเส้นทางการทำงาน

6. วิธีการทดสอบแบบกล่องดำ เน้นการทดสอบอะไรบ้าง

ค่าข้อมูลต่ำสุดของพิสัยข้อมูล (ขอบเขตข้อมูล)

ค่าข้อมูลสูงสุดขอพิสัยข้อมูล

ค่าข้อมูลตัวแทนกลุ่ม เป็นค่าที่ใกล้เคียงกับค่ากลางของพิสัยข้อมูล

ค่าข้อมูลเกินพิสัย หรือเกินขอบเขตข้อมูลในแต่ละช่วง

7. วิธีการทดสอบแบบกล่องขาว เน้นการทดสอบอะไรบ้าง

ทดสอบเพื่อให้ทราบว่าทุกเส้นทางในกระบวนการสามารถทำงานได้อย่างถูกต้อง

ทดสอบการตัดสินใจทางตรรกะทุกเส้นทาง ทั้งค่าจริงและค่าเท็จ

ทดสอบการทำงานภายในของการวนรอบทุกรอบตามจำนวนที่กำหนดการวนรอบ

ทดสอบโครงสร้างข้อมูลภายในให้ถูกต้องก่อนส่งข้อมูลไปยังการประมวลผลอื่น

8. เบรกพอยท์ (Breakpoint) ในวิชวลซีชาร์ปใช้ทำอะไร

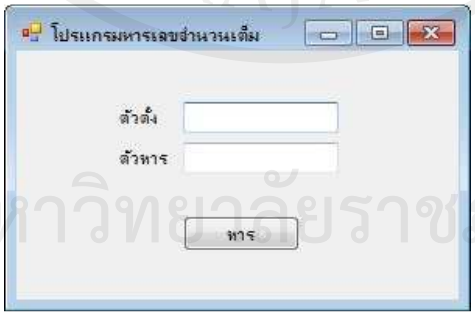
เบรคพอยท์ (Breakpoints) ใช้กำหนดตำแหน่งที่ต้องการสำหรับการทำให้โปรแกรมหยุดการทำงานชั่วคราวแล้วกลับมายังส่วนของการเขียนคำสั่งในโปรแกรม การตั้งเบรคพอยท์สามารถทำได้โดยคลิกที่แถบด้านซ้ายมือของบรรทัดนั้นหรือคลิกเมาส์ที่ปุ่มขวาเพื่อเลือกคำสั่ง Insert Breakpoints เมื่อทำแล้วจะขึ้นจุดสีแดงหน้าบรรทัดนั้น

9. จงอธิบายการใช้งานคำสั่ง try...catch...finally

ใช้สำหรับการจัดการสิ่งผิดปกติที่เกิดขึ้นในระหว่างการทำงาน โดยมีรูปแบบดังนี้

```
try
{
   <คำสั่งที่ต้องการทำงาน>
}
catch
{
   <คำสั่งที่ต้องการทำเมื่อเกิดข้อผิดพลาด>
}
finally
{
   <คำสั่งที่ต้องการทำต่อไปไม่ว่าจะเกิดข้อผิดพลาดหรือไม่ก็ตาม>
}
```

10. จงเขียนโปรแกรมหารเลขจำนวนเต็ม โดยป้องกันการใส่ตัวเลขที่ไม่ใช่จำนวนเต็ม และมีหน้าจอแสดงผลดังนี้



กำหนด Properties สำหรับ Form

Name กำหนดเป็น frmDiv

Text กำหนดเป็น “โปรแกรมหารเลขจำนวนเต็ม”

กำหนด Properties สำหรับ label1

Name กำหนดเป็น lblNum1

Text กำหนดเป็น “ตัวตั้ง”

กำหนด Properties สำหรับ label2

Name กำหนดเป็น lblNum2

Text กำหนดเป็น “ตัวหาร”

กำหนด Properties สำหรับ textBox1

Name กำหนดเป็น txtNum1

Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ textBox2

Name กำหนดเป็น txtNum2

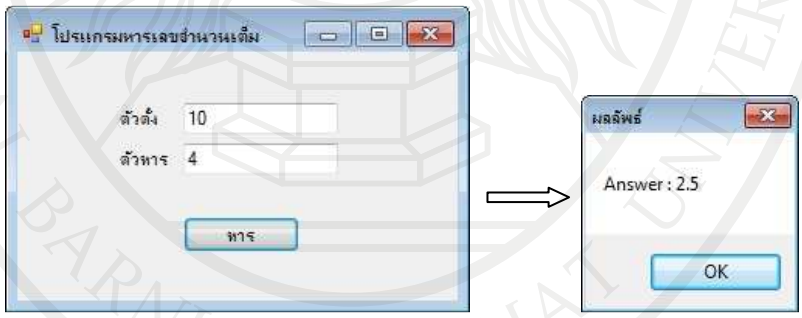
Text ใส่เป็นค่าว่าง

กำหนด Properties สำหรับ button1

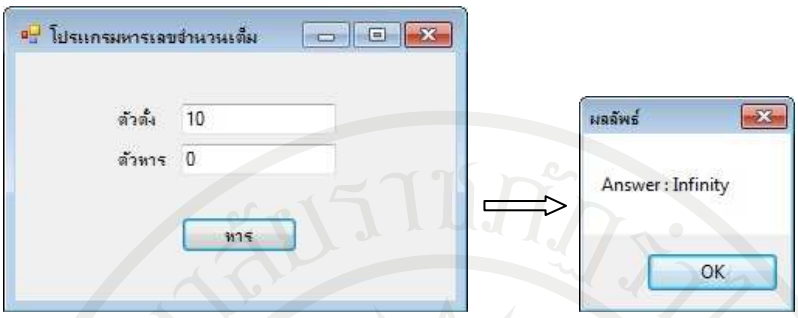
Name กำหนดเป็น btnDiv

Text กำหนดเป็น “หาร”

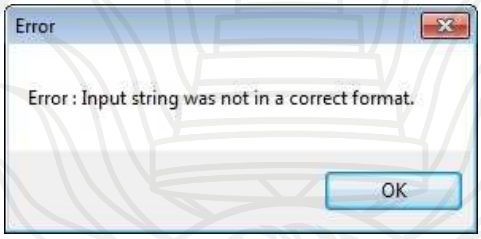
เมื่อโปรแกรมทำงานจะแสดงผลดังนี้



เมื่อใส่ค่าตัวหารที่เป็นศูนย์



เมื่อเกิดข้อผิดพลาดขึ้น โปรแกรมจะแสดงผลดังนี้



ชุดคำสั่งเป็นดังนี้

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Ex
{
    public partial class frmDiv : Form
    {
        public frmDiv ()
        {
            InitializeComponent();
        }
        private void btnDiv_Click(object sender, EventArgs e)
        {
```

```
            int num1;
            int num2;
            double ans;
            try
            {
                num1 = Convert.ToInt32(txtNum1.Text);
                num2 = Convert.ToInt32(txtNum2.Text);
                ans = (double)(num1) / (double)(num2);
                MessageBox.Show("Answer : " + ans, "ผลลัพธ์");
            }
            catch (Exception ex2)
            {
                MessageBox.Show("Error : " + ex2.Message, "Error");
            }
        }
    }
}
```

แนวคำตอบแบบฝึกหัดบทที่ 9

1. จงอธิบายความหมายของเธรด

ในการประมวลผลของคอมพิวเตอร์ จะมีการแบ่งเวลาในการประมวลผลงานแต่ละงานออกเป็นส่วน ๆ ในแต่ละส่วนที่เครื่องคอมพิวเตอร์ประมวลผลจะเรียกว่าโปรเซส (Process) แต่ละโปรเซสจะมีการแบ่งเป็นส่วนย่อย ๆ อีก เรียกว่า เธรด (Thread) ซึ่งโดยทั่วไป 1 โปรเซสอาจจะมีเธรดเดียว (Single Thread) หรือหลายเธรด (Multi Threads) ก็ได้ ดังนั้นเธรดคือส่วนย่อยของโปรเซสที่ถูกแบ่งออกมาเพื่อประมวลผล

2. เธรดมีส่วนประกอบอะไรบ้าง

หมายเลขเธรด (Thread ID) เป็นหมายเลขที่ใช้ระบุหมายเลขของเธรดที่อยู่ในโปรเซส
ตัวนับ (Program counter) ใช้นับลำดับของคำสั่งเพื่อประมวลผล
ชุดของรีจิสเตอร์ (Register set) ใช้เพื่อเก็บค่าตัวแปรที่กำลังทำงานอยู่
สแตก (Stack) ใช้เก็บข้อมูลประวัติของการประมวลผล

3. มัลติเธรดมีกี่ประเภท อะไรบ้าง

การทำงานแบบหลายเธรดนั้นเป็นการทำงานร่วมกันระหว่างเคอร์เนลเธรด (Kernel Thread) กับยูสเซอร์เธรด (User Thread) ซึ่งสามารถแบ่งออกเป็น 3 ประเภท คือ

แบบกลุ่มต่อหนึ่ง (Many to One) เป็นรูปแบบที่มีเคอร์เนลเธรด 1 หน่วย กับ ยูสเซอร์เธรดหลายหน่วย ซึ่งยูสเซอร์เธรดจะสามารถเข้าใช้เคอร์เนลเธรดได้ทีละ 1 Thread เท่านั้น ทำให้ไม่สามารถประมวลผลได้พร้อมกัน

แบบหนึ่งต่อหนึ่ง (One to One) เป็นรูปแบบที่มีเคอร์เนลเธรด 1 หน่วย กับยูสเซอร์เธรด1 หน่วย โดยจะมีการสร้างยูสเซอร์เธรดเท่ากับจำนวนของเคอร์เนลเธรดทำให้สามารถประมวลผลแบบขนานได้ แต่รูปแบบนี้จะไม่มีการสร้างยูสเซอร์เธรดที่มากเกินกว่าเคอร์เนลเธรด

แบบกลุ่มต่อกลุ่ม (Many to Many) เป็นรูปแบบที่แก้ไขข้อจำกัดของ 2 แบบแรก โดยจะมีการสร้างยูสเซอร์เธรดเท่าที่จำเป็นเพื่อให้สอดคล้องกับจำนวนของเคอร์เนลเธรดทำให้สามารถประมวลผลแบบขนานได้และสามารถจัดสรรการทำงานระหว่างยูสเซอร์เธรดกับเคอร์เนลเธรดได้

4. การเริ่มต้นใช้งานเธรด จำเป็นต้องเรียกเนมสเปซ (Namespace) ใดก่อน

using System.Threading;

5. ในการสั่งให้เธรดทำงาน ต้องใช้เมธอดใดในการสั่งงาน

Start เริ่มการทำงานของเธรด

6. เทคนิคดีลีเกทใช้สำหรับทำอะไรเกี่ยวกับเธรด

ดีลีเกทในวิชวลซีชาร์ปจะมีลักษณะคล้ายกับพอยท์เตอร์ (Pointer) ในภาษาซีหรือภาษาซีพลัสพลัส ซึ่งจะใช้สำหรับเมธอดที่มีค่าพารามิเตอร์และค่ารีเทิร์นเหมือนกัน ทำให้สามารถใช้งานออบเจ็คต์ที่มีค่าพารามิเตอร์และค่ารีเทิร์นเหมือนกันโดยไม่จำเป็นต้องทราบว่าออบเจ็คต์นั้นอยู่ในคลาสใด โดยสามารถใช้งานผ่าน Invoke

7. มัลติเธรดไม่ควรใช้ในกรณีใดบ้าง

ในการสร้างเธรดเพื่อใช้ในการประมวลผลนั้น ไม่ควรสร้างเธรดมากจนเกินไป กล่าวคือ ไม่ควรเกิน 2 เท่าของจำนวนเธรดที่อยู่ในซีพียู เช่น ซีพียูมีการทำงาน 4 เธรด สามารถสร้างเธรดได้ไม่เกิน 8 เธรดเพราะถ้าสร้างเธรดมากจนเกินไป จะทำให้ซีพียูทำงานมากจนเกินไปทำให้ไม่สามารถแบ่งเวลาทำงานอย่างมีประสิทธิภาพได้

งานที่ทำแต่ละเธรดจะต้องแยกการทำงานกันอย่างชัดเจน ไม่ใช้การแบ่งเธรดกับงานที่จำเป็นต้องรอผลลัพธ์ของกันและกัน เพราะจะทำให้เธรดไม่สามารถทำงานได้อย่างมีประสิทธิภาพ

8. การใช้งานเธรดร่วมกับคอนโทรลใน Windows Forms ต้องทำอย่างไร

ในการนำเธรดมาประยุกต์ใช้งานใน Windows Forms นั้น ต้องอาศัยเทคนิคดีลีเกทเข้ามาช่วย เนื่องจากเธรดในวิชวลซีชาร์ปจะมีการแยกกันทำงานกันอย่างชัดเจน ทำให้การใช้งานร่วมกันของคอนโทรลเกิดปัญหาขึ้น

9. เมื่อต้องการหยุดการทำงานของเธรดชั่วคราวและให้ทำงานต่อ ต้องใช้เมธอดใดของเธรดทำงานบ้าง

Suspend หยุดเธรดไว้ชั่วคราวเมื่อต้องการ

Resume เริ่มการทำงานของเธรดหลังจากหยุดชั่วคราว

10. คำสั่ง Invoke มีรูปแบบการทำงานอย่างไร

รูปแบบการใช้งาน Invoke

```
public object Invoke(
        Delegate method
)
```