

การเขียนโปรแกรมเชิงวัตถุ

Object Oriented Programming

ผู้ช่วยศาสตราจารย์ สติย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนำ

การเขียนโปรแกรมโดยการจำลองแนวคิดเชิงวัตถุเป็นอีกหนึ่งในภาษาโปรแกรมที่มีการพัฒนามาอย่างต่อเนื่องและถูกนำไปใช้ในการพัฒนาในด้านต่างๆ ในปัจจุบัน เช่น การเขียนโปรแกรมผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชันต่าง ๆ หรือแม้กระทั่งการใช้แนวความคิดในการเขียนโปรแกรมเชิงวัตถุมาเพื่อใช้พัฒนาเกมส์ต่างๆ

การเขียนโปรแกรมเชิงวัตถุ คือหนึ่งในรูปแบบการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ความสำคัญกับ วัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้วัตถุ อื่นๆที่เกี่ยวข้องเพื่อให้ทำงานต่อไป การเขียนโปรแกรมเชิงวัตถุสามารถนำมาใช้จำลองการทำงานตามโลกของความเป็นจริงได้

จากความสำคัญของการเขียนโปรแกรมโดยการใช้ความรู้ทางด้านการออกแบบเชิงวัตถุดังกล่าวผู้เขียนจึงมีความคิดว่า หากมีหนังสือที่สามารถใช้เป็นจุดเริ่มต้นในการพัฒนาความคิดและการฝึกฝนเพื่อพัฒนาแนวคิดในการเขียนโปรแกรมเชิงวัตถุจะทำให้เกิดความเข้าใจและสามารถนำไปต่อยอดเพื่อพัฒนาโปรแกรมในรูปแบบดังกล่าวต่อไปในอนาคต ด้วยเหตุนี้ผู้เขียนจึงได้เรียบเรียงความรู้จากการสอน และงานวิจัยในด้านการเขียนโปรแกรมเชิงวัตถุเพื่อนำมาพัฒนาตำราเล่มนี้

ผู้เขียนมีความคาดหวังว่า ตำราเล่มนี้จะเกิดประโยชน์ต่อผู้ที่สนใจการเขียนโปรแกรมเชิงวัตถุโดยการใช้ภาษาจาวาให้สามารถนำความรู้พื้นฐานดังกล่าวไปต่อยอดสำหรับการเขียนโปรแกรมเพื่อพัฒนาและเกิดประโยชน์ต่อองค์กรต่อไป

สารบัญ

บทที่ 1 กระบวนการพัฒนาระบบโปรแกรมเชิงวัตถุ	1
1.1 ความนำ	1
1.2 แนวคิดของการโปรแกรมแบบเน้นกรรมวิธีและการโปรแกรมเชิงวัตถุ	1
1.3 การพัฒนาโปรแกรมโดยใช้ภาษาคอมพิวเตอร์เชิงวัตถุ	2
1.4 ประวัติของภาษาจาวา	3
1.5 องค์ประกอบของเทคโนโลยีจาวา	4
1.6 แพลตฟอร์มของเทคโนโลยีจาวา	5
1.7 ลักษณะการเขียนโปรแกรมและการเขียนคู่มือโปรแกรม	5
1.8 ความผิดพลาด และการแก้ไข	5
1.9 บทสรุป	7
บทที่ 2 ชนิดข้อมูลแบบนามธรรมและชนิดข้อมูลแบบทั่วไป	8
2.1 ความนำ	8
2.2 ชนิดข้อมูลแบบนามธรรม	8
2.3 ชนิดข้อมูลแบบทั่วไป	8
2.4 ตัวแปรและชนิดข้อมูลในภาษาจาวา	9
2.5 ตัวแปรค่าคงที่	10
2.6 ค่าคงที่	11
2.7 ชนิดข้อมูลแบบพื้นฐาน ชนิดข้อมูลแบบอ้างอิง	12
2.8 ตัวดำเนินการทางคณิตศาสตร์	15
2.9 การแปลงชนิดข้อมูล	18
2.10 การเขียนชุดคำสั่งเพื่อรับค่าในรูปแบบ dos mode และในรูปแบบกราฟฟิก	20
2.11 บทสรุป	23
2.12 แบบฝึกหัดปฏิบัติการ	24
บทที่ 3 คำสั่งควบคุม	25
3.1 ความนำ	25
3.2 คำสั่งทดสอบเงื่อนไข การตัดสินใจ	25
3.3 คำสั่งทดสอบเงื่อนไข switch statement	30
3.4 คำสั่งการวนซ้ำ	32
3.5 ตัวอย่างการประยุกต์คำสั่งวนซ้ำ	38
3.6 บทสรุป	40
3.7 แบบฝึกหัดปฏิบัติการ	41
บทที่ 4 คลาสและวัตถุ	43
4.1 ความนำ	43
4.2 นิยามและความหมายของคลาสและวัตถุ	43
4.3 การประกาศคลาส	44

	4.4 วัฏจักรชีวิตของวัตถุ	46
	4.5 คอนสตรัคเตอร์	51
	4.6 การห่อหุ้มและการซ่อนข้อมูลของวัตถุ	54
	4.7 ตัวอย่างของ package ของจาวา	58
	4.8 บทสรุป	58
	4.9 แบบฝึกหัดปฏิบัติการ	60
บทที่ 5 เมธอด		65
	5.1 ความนำ	65
	5.2 การประกาศเมธอด	65
	5.3 ชนิดของเมธอด	66
	5.4 การผ่านค่าข้อมูลไปให้เมธอด	67
	5.5 ขอบเขตของตัวแปร	65
	5.6 การโอเวอร์โหลดเมธอด	70
	5.7 เมธอดของคลาส Math	71
	5.8 การส่งวัตถุไปยังเมธอด	72
	5.9 Recursive Method	73
	5.10 บทสรุป	74
	5.11 แบบฝึกหัดปฏิบัติการ	75
บทที่ 6 อาร์เรย์		80
	6.1 ความนำ	80
	6.2 นิยามของอาร์เรย์	80
	6.3 อาร์เรย์ของตัวแปรพื้นฐาน	82
	6.4 อาร์เรย์สองมิติ	87
	6.5 อาร์เรย์ลิสต์	90
	6.6 อาร์เรย์ของตัวแปรชนิดอ้างอิง	92
	6.7 การประยุกต์การใช้อาร์เรย์ในการเรียงลำดับ การค้นหาข้อมูล	93
	6.8 บทสรุป	97
	6.9 แบบฝึกหัดปฏิบัติการ	98
บทที่ 7 การสืบทอดคุณสมบัติ		109
	7.1 ความนำ	109
	7.2 นิยามของการสืบทอด (Inheritance)	109
	7.3 การใช้คีย์เวิร์ด super	111
	7.4 Constructor Chaining	113
	7.5 การเรียกใช้เมธอดของ superclass	114
	7.6 โอเวอร์ไรด์เมธอด (Overriding Methods)	115
	7.7 คลาส Object และเมธอดในคลาส Object	118
	7.8 บทสรุป	119

	7.9 แบบฝึกหัดปฏิบัติการ	120
บทที่ 8 การห่อหุ้มและการซ่อนข้อมูล		129
	8.1 ความนำ	129
	8.2 นิยามของการห่อหุ้มและการซ่อนข้อมูล	129
	8.3 การใช้ Modifiers และระดับการมองเห็น	130
	8.4 คีย์เวิร์ด final	131
	8.5 เมธอดที่สำคัญในคลาส Object	132
	8.6 Initialization block	132
	8.7 บทสรุป	135
	8.8 แบบฝึกหัดปฏิบัติการ	136
บทที่ 9 การพ้องรูป		138
	9.1 ความนำ	138
	9.2 นิยามของการพ้องรูป (Polymorphism)	138
	9.3 Dynamic binding	138
	9.4 Static binding	140
	9.5 การแปลงวัตถุ ด้วย Up casting และ Down casting	142
	9.6 บทสรุป	144
	9.7 แบบฝึกหัดปฏิบัติการ	146
บทที่ 10 คลาสนามธรรมและอินเทอร์เฟส		150
	10.1 ความนำ	150
	10.2 คลาสนามธรรม	150
	10.3 อินเทอร์เฟส	156
	10.4 บทสรุป	158
	10.5 แบบฝึกหัดปฏิบัติการ	160
บทที่ 11 การสร้างส่วนต่อประสานผู้ใช้		165
	11.1 ความนำ	165
	11.2 ความรู้เบื้องต้นเกี่ยวกับ Swing และ AWT	165
	11.3 ส่วนประกอบที่สำคัญของแพ็คเกจ Swing	167
	11.4 การสร้างส่วนต่อประสานผู้ใช้	179
	11.5 บทสรุป	181
	11.6 แบบฝึกหัดปฏิบัติการ	182
บทที่ 12 การสร้างกราฟิกในจาวา		186
	12.1 ความนำ	186
	12.2 ระบบ coordinate ของจาวา	186
	12.3 คลาส Graphics	187
	12.4 การวาดภาพกราฟิกเบื้องต้น	189
	12.5 ตัวอย่างการประยุกต์การวาดภาพกราฟิกด้วยภาษาจาวา	195

	12.6 บทสรุป	198
	12.7 แบบฝึกหัดปฏิบัติการ	199
บทที่ 13 การจัดการเหตุการณ์และการจัดการข้อผิดพลาด		201
	13.1 ความนำ	201
	13.2 การเขียนโปรแกรมในรูปแบบ Procedural และ Event-Driven	201
	13.3 อินเตอร์เฟซประเภท Listener	202
	13.4 การจัดการกับเหตุการณ์	204
	13.5 การควบคุมเหตุการณ์ที่เกิดขึ้นกับคอมโพเนนต์ประเภทต่างๆ	214
	13.6 ตัวอย่างการประยุกต์การควบคุมเหตุการณ์โดยใช้คลาส Timer	220
	13.7 การจัดการข้อผิดพลาด	223
	13.7 บทสรุป	229
	13.8 แบบฝึกหัดปฏิบัติการ	231
บทที่ 14 เรด		236
	14.1 ความนำ	236
	14.2 ความหมายของเรด	236
	14.3 การสร้างเรด	236
	14.4 การทำงานของเรด	238
	14.5 ตัวอย่างโปรแกรมโดยใช้เรด	243
	14.6 บทสรุป	245
	14.7 แบบฝึกหัดปฏิบัติการ	246
บทที่ 15 ตัวอย่างการพัฒนาเกมส์และแอปพลิเคชันโดยการเขียนโปรแกรมเชิงวัตถุ		250
	15.1 ความนำ	250
	15.2 ตัวอย่างการพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุ	250
	15.3 ตัวอย่างการพัฒนาโปรแกรมการแสดงผลแอนิเมชันโดยการเขียนโปรแกรมเชิงวัตถุ	264
	15.4 บทสรุป	268
	15.5 แบบฝึกหัดปฏิบัติการ	268

บทที่ 1 กระบวนการพัฒนาระบบโปรแกรมเชิงวัตถุ

วัตถุประสงค์

- 1.1 อธิบายความหมายและแนวคิดของการโปรแกรมแบบเน้นกรรมวิธีและการโปรแกรมเชิงวัตถุได้
- 1.2 มีความรู้ความเข้าใจเกี่ยวกับภาษาจาวา องค์ประกอบของเทคโนโลยีจาวา ขั้นตอนการทำงานของโปรแกรมภาษาจาวา แพลตฟอร์มของเทคโนโลยีจาวา
- 1.3 พัฒนาชุดคำสั่งแบบเน้นกรรมวิธีและการโปรแกรมเชิงวัตถุได้
- 1.4 สามารถอธิบายลักษณะการเขียนโปรแกรมและการเขียนคู่มือโปรแกรม ความผิดพลาด และการแก้ไข

1.1 ความนำ

การเขียนโปรแกรมโดยการจำลองแนวคิดเชิงวัตถุเป็นอีกหนึ่งในภาษาโปรแกรมที่มีการพัฒนามาอย่างต่อเนื่องและถูกนำไปใช้ในการพัฒนาในด้านต่างๆ ในปัจจุบัน เช่น การเขียนโปรแกรมผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชัน หรือแม้กระทั่งการใช้แนวความคิดในการเขียนโปรแกรมเชิงวัตถุมาเพื่อใช้พัฒนาเกมส์ การเขียนโปรแกรมเชิงวัตถุคือหนึ่งในรูปแบบการเขียนโปรแกรมคอมพิวเตอร์ที่ให้ความสำคัญกับวัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้วัตถุอื่นๆที่เกี่ยวข้อง เพื่อให้ทำงานต่อไป การเขียนโปรแกรมเชิงวัตถุสามารถนำมาใช้จำลองการทำงานตามโลกของความเป็นจริงได้

1.2 แนวคิดของการโปรแกรมเชิงโครงสร้างและการโปรแกรมเชิงวัตถุ

ภาษาเชิงโครงสร้างและภาษาเชิงวัตถุ

รูปแบบการเขียนโปรแกรม (Programming paradigm) มีรูปแบบการเขียนโปรแกรมในลักษณะแตกต่างกัน เช่น การเขียนโปรแกรมเชิงโครงสร้าง (Structured Programming) การเขียนโปรแกรมแบบฟังก์ชัน (Functional Programming) การเขียนโปรแกรมแบบตรรกะ (Logic Programming) การเขียนโปรแกรมเชิงวัตถุ (Object-oriented Programming) เป็นต้น ซึ่งรูปแบบการเขียนโปรแกรมในแต่ละรูปแบบก็มีหลักการและกระบวนการทำงานที่แตกต่างกัน ในปัจจุบันรูปแบบการเขียนโปรแกรมที่เป็นที่นิยมจะมีรูปแบบในการเขียนโปรแกรมสองรูปแบบคือ การเขียนโปรแกรมโดยใช้ภาษาเชิงโครงสร้างและการเขียนโปรแกรมโดยใช้ภาษาเชิงวัตถุ

1.2.1 ภาษาเชิงโครงสร้างเป็นการเขียนโปรแกรมที่มีลักษณะดังนี้

- ▶ โปรแกรมจะแบ่งออกเป็นส่วนย่อย ๆ ที่เรียกว่าโมดูล (module)
- ▶ แต่ละโมดูลจะต้องเป็นอิสระต่อกัน
- ▶ การออกแบบให้แต่ละโมดูลมีความเป็นอิสระต่อกันทำได้ยาก
- ▶ ต้นทุนในการพัฒนาโปรแกรมสูง

1.2.2 ภาษาเชิงวัตถุ เป็นการเขียนโปรแกรมที่มีลักษณะดังนี้

- ▶ การพัฒนาโปรแกรมเป็นการเลียนแบบการทำงานเชิงวัตถุ
- ▶ สามารถนำโปรแกรมกลับมาใช้ใหม่ได้ดีกว่าภาษาเชิงกระบวนการ

1.3 การพัฒนาโปรแกรมโดยใช้ภาษาคอมพิวเตอร์เชิงวัตถุ

การเขียนโปรแกรมเชิงวัตถุจะเป็นการมองปัญหาว่าประกอบไปด้วยวัตถุ (Object) ต่างๆ ซึ่งแนวคิดการมองปัญหาเป็นวัตถุนี้จะเข้าใกล้เคียงกับธรรมชาติของมนุษย์มากที่สุดเนื่องจากมนุษย์มองสิ่งต่างๆ รอบตัวเป็นวัตถุทั้งที่เป็นรูปธรรม เช่น ปากกา นักศึกษา หรือใบลงทะเบียน เป็นต้น และที่เป็นนามธรรมเช่น คะแนน หรือรายชื่อวิชา เป็นต้น การเขียนโปรแกรมเชิงวัตถุจะเป็นขบวนการการวิเคราะห์ปัญหา โดยการจำลองคุณลักษณะและพฤติกรรมของวัตถุในระบบจริง ให้อยู่ในรูปของโปรแกรมคอมพิวเตอร์ ตัวอย่างเช่น การพัฒนาโปรแกรมระบบทะเบียนนักศึกษาอาจแบ่งโปรแกรมให้ประกอบด้วยวัตถุต่างๆ เช่น นักศึกษา ใบลงทะเบียน และรายวิชา เป็นต้น วัตถุชนิดนักศึกษามีคุณลักษณะต่างๆ เช่น ชื่อ รหัสนักศึกษา และเกรดเฉลี่ย เป็นต้น และอาจมีพฤติกรรมที่นักศึกษสามารถกระทำได้เช่น ลงทะเบียน และเพิ่มหรือถอนวิชา เป็นต้น

ตัวอย่างที่ 1.1 ระบบทะเบียนนักศึกษา

วิธีการเชิงโครงสร้าง	วิธีการเชิงวัตถุ
ลงทะเบียนรายวิชา	นักศึกษา
ชำระเงิน	ใบลงทะเบียน
เพิ่มวิชา	รายชื่อนักศึกษา

การพัฒนาโปรแกรมโดยใช้ภาษาคอมพิวเตอร์เชิงวัตถุ จะทำให้กระบวนการพัฒนาโปรแกรมทำได้รวดเร็วขึ้นและสามารถปรับปรุงแก้ไขโปรแกรมได้ง่าย ซึ่งเหมาะกับการพัฒนาโปรแกรมขนาดใหญ่ที่จะต้องมีการปรับปรุงแก้ไขโปรแกรมอยู่ตลอดเวลา นอกจากนี้โปรแกรมเชิงวัตถุยังมีคุณลักษณะเด่นอื่นๆ อีกดังนี้

การห่อหุ้ม (Encapsulation) เป็นคุณลักษณะที่ทำให้วัตถุแต่ละตัวเป็นอิสระต่อกัน ซึ่งทำให้สามารถแบ่งการพัฒนาโปรแกรมออกเป็นส่วนๆ ได้ง่าย

การสืบทอด (Inheritance) เป็นคุณลักษณะที่ทำให้สามารถนำโปรแกรมที่พัฒนาแล้วกลับมาใช้ใหม่ได้ง่ายกว่าการเขียนโปรแกรมเชิงกระบวนการ

การมีได้หลายรูปแบบ (Polymorphism) เป็นคุณลักษณะที่ทำให้นักพัฒนาโปรแกรมสามารถเพิ่มเติมรายละเอียดของเมธอดในคลาสได้

การมองแบบนามธรรม (Abstraction) เป็นคุณลักษณะของการมองทุกสิ่งที่สนใจให้อยู่ในรูปของคลาส และวัตถุ เป็นการมองที่มองสิ่งต่าง ๆ ให้เป็นวัตถุที่ประกอบด้วยคุณสมบัติ และวิธีการ

1.4 ประวัติของภาษาจาวา

พัฒนาขึ้นโดยทีมวิจัยของบริษัทซันไมโครซิสเต็มส์ โดยพัฒนามาจากโครงการที่ต้องการพัฒนาระบบซอฟต์แวร์เพื่อควบคุมเครื่องใช้ไฟฟ้าขนาดเล็กภายในบ้าน ชื่อเดิมคือภาษา Oak ต่อมาเปลี่ยนเป็นภาษาจาวา เป็นภาษาที่ไม่ขึ้นกับแพลตฟอร์ม (Platform independent) JDK1.0 ประกาศใช้เมื่อปี 1996 โดยจุดเด่นของภาษาจาวามีดังนี้

ความง่าย (simple) ภาษาจาวาเป็นภาษาที่ง่ายต่อการศึกษาและพัฒนาโปรแกรมทั้งนี้เพราะภาษาจาวาพัฒนาโดยตัดข้อด้อยของภาษา C++ ออกไปอาทิเช่น เรื่องของการใช้ pointer

ภาษาเชิงวัตถุ (object oriented) ภาษาจาวาเป็นภาษาคอมพิวเตอร์เชิงออบเจกต์ที่สมบูรณ์โดยมีคุณลักษณะเด่นของโปรแกรมเชิงออบเจกต์คือ การสืบทอดการห่อหุ้ม และการมีได้หลายรูปแบบ

การกระจาย (distributed) ภาษาจาวามีชุดคำสั่งที่เป็นแพ็คเกจ (package) ในการจัดการกับโปรโตคอล TCP/IP ทำให้สามารถพัฒนาโปรแกรมเชิงออบเจกต์แบบกระจาย (distributed object) ผ่านระบบเครือข่ายอินเทอร์เน็ตได้ง่าย

การป้องกันการผิดพลาด (robust) ภาษาจาวาเป็นภาษาคอมพิวเตอร์ที่ออกแบบมาเพื่อให้โปรแกรมที่พัฒนาขึ้นมีความน่าเชื่อถือ โดยมีการตรวจสอบการผิดพลาดที่อาจเกิดขึ้นในขั้นตอนต่างๆ เช่น ขั้นตอนการคอมไพล์ และการรันโปรแกรม เป็นต้น

ความปลอดภัย (secure) ภาษาจาวาออกแบบมาเพื่อพัฒนาโปรแกรมบนระบบเครือข่าย และมีการกระจายการทำงานบนระบบเครือข่ายอินเทอร์เน็ต ดังนั้นจึงต้องสร้างระบบป้องกันความปลอดภัยจากไวรัสคอมพิวเตอร์ และการแก้ไขโปรแกรมจากภายนอก

สถาปัตยกรรมกลาง (architecture neutral) โปรแกรมภาษาจาวาจะคอมไพล์ได้โปรแกรมไบท์โค้ด (byte code) ซึ่งสามารถทำงานบนสถาปัตยกรรมคอมพิวเตอร์ที่มีหน่วยประมวลผลกลางและระบบปฏิบัติการต่างๆ ได้

เคลื่อนย้ายง่าย (portable) ข้อกำหนดของภาษาจาวาจะไม่ขึ้นอยู่กับระบบคอมพิวเตอร์ใดโดยเฉพาะ

อินเทอร์พรีต (interpreted) ภาษาจาวาจะใช้อินเทอร์พรีเตอร์ในการแปลโปรแกรมไบท์โค้ดให้เป็นภาษาเครื่อง ดังนั้นจึงทำให้กระบวนการพัฒนาโปรแกรมเป็นไปได้อย่างรวดเร็ว

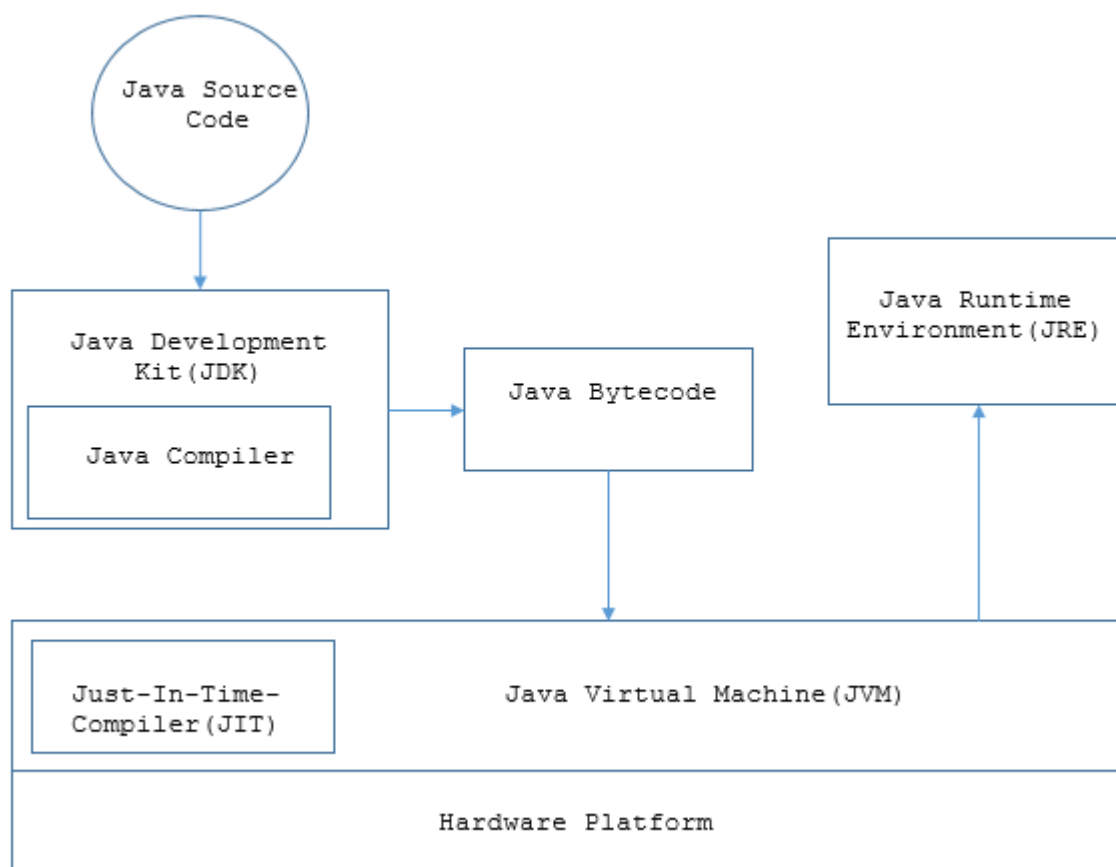
ประสิทธิภาพสูง (high performance) โดยปกติโปรแกรมอินเทอร์พรีเตอร์ที่ทำหน้าที่แปลโปรแกรมไบท์โค้ดจะทำงานช้า แต่เทคโนโลยีจาวาได้พัฒนาให้มีการแปลโปรแกรมไบท์โค้ด ในขั้นตอนการรันโปรแกรมให้เป็นภาษาเครื่องแบบทันทีทันใด (Just In Time) ที่ทำงานได้รวดเร็วเทียบเท่ากับคอมไพเลอร์ เพื่อได้โปรแกรมจาวาที่มีประสิทธิภาพในการประมวลผลสูง

มัลติเธรด (multithreaded) ภาษาจาวามีความสามารถที่จะประมวลผลหลายๆงานได้พร้อมกัน

พลวัต (dynamic) ภาษาจาวาออกแบบมาเพื่อที่จะให้ปรับเปลี่ยนเพิ่มเติมไลบรารี (library) ต่างๆได้ง่าย ซึ่งแตกต่างจากภาษาซี หรือ C++

1.5 องค์ประกอบของเทคโนโลยีจาวา

องค์ประกอบหลักของเทคโนโลยีจาวาสามารถแสดงได้ดังรูปที่ 1 โดยแบ่งเป็น 3 ส่วนที่สำคัญ ดังนี้



รูปที่ 1.1 องค์ประกอบของเทคโนโลยีจาวา

1.5.1 Java Virtual Machine (JVM) เป็นส่วนที่ทำหน้าที่ป็นอินเทอร์พรีเตอร์ (interpreter) จะทำการแปลจาวาไบต์โค้ด ให้เป็นภาษาที่เครื่องเข้าใจ โดยจาวาไบต์โค้ดสามารถรันได้ในหลายแพลตฟอร์ม ถ้าแพลตฟอร์มนั้นมี JVM

1.5.2 Java Runtime Environment (JRE) เป็นส่วนที่ใช้ในการรันโปรแกรมภาษาจาวา โดยจะทำงาน 3-ขั้นตอน ดังนี้

- โหลดไบต์โค้ดโดยใช้ **Class Loader** คือการโหลดคลาสทุกคลาสที่เกี่ยวข้องในการรันโปรแกรม
- ตรวจสอบไบต์โค้ดโดยใช้ **Byte Code Verifier** คือ การตรวจสอบว่าโปรแกรมจะต้องไม่มีคำสั่งที่ทำให้เกิดความผิดพลาดกับระบบ
- รันไบต์โค้ดโดยใช้ **Runtime Interpreter**

1.5.3 Java Development Kit (JDK) เป็นชุดพัฒนาโปรแกรมภาษาจาวาที่จะประกอบไปด้วย JVM ตัวแปลภาษาจาวา (Java Compiler) เครื่องมือ (tool) อื่นๆในการพัฒนาโปรแกรม และ API ทั้งหมดในภาษาจาวา ซึ่ง API จะเป็นมาตรฐานคำสั่งต่างๆของภาษาจาวา

1.6 แพลตฟอร์มของเทคโนโลยีจาวา

แพลตฟอร์ม (Platform) คือฮาร์ดแวร์และสภาวะแวดล้อมทางซอฟต์แวร์ (Software Environment) ที่โปรแกรมจะใช้ในการประมวลผลโดยทั่วไป แพลตฟอร์มของเทคโนโลยีจาวาจะประกอบด้วย

- Java Virtual Machine (JVM)
- Java Application Programming Interface (Java API)

1.7 ลักษณะการเขียนโปรแกรมและการเขียนคู่มือโปรแกรม

การเขียนโปรแกรมที่ดีจะมีหลักในการปฏิบัติ ดังนี้

- 1. การเขียนคอมเมนต์ที่เหมาะสม (Appropriate Comments) ผู้เขียนโปรแกรมควรเขียนสรุปรายละเอียดของการทำงานของโปรแกรม คุณสมบัติที่สำคัญของโปรแกรม โครงสร้างข้อมูล เทคนิคที่สำคัญต่างๆ ของโปรแกรมที่ส่วนต้นเมื่อเริ่มเขียนโปรแกรม นอกจากนี้ผู้เขียนโปรแกรมควรเขียนชื่อของผู้พัฒนาโปรแกรม วันที่พัฒนาโปรแกรม คำอธิบายสั้นๆ เมื่อเริ่มเขียนโปรแกรม
- 2. การตั้งชื่อต่างๆ (Naming Conventions) ผู้เขียนควรใช้ชื่อที่มีความหมายและสามารถอธิบายรายละเอียดของการทำงานได้เมื่อดูชื่อ การตั้งชื่อตัวแปรและเมธอด ให้ใช้ตัวพิมพ์เล็กเมื่อตั้งชื่อตัวแปร ถ้าชื่อที่ตั้งมีหลายคำให้รวมคำดังกล่าวเข้าด้วยกันโดยใช้ตัวพิมพ์เล็กสำหรับอักษรขึ้นต้นของคำแรก เมื่อเริ่มคำต่อไปให้ใช้ตัวพิมพ์ใหญ่เป็นตัวเริ่มต้น เช่น computeArea() ในส่วนของการตั้งชื่อคลาส ให้ใช้ตัวพิมพ์ใหญ่เป็นอักษรตัวแรกในการตั้งชื่อคลาส เช่น คลาส ComputeArea ส่วนการตั้งชื่อค่าคงที่ ให้ใช้ตัวพิมพ์ใหญ่ทั้งหมดเมื่อตั้งชื่อค่าคงที่ เช่น PI
- 3. รูปแบบของการเขียนบล็อกของคำสั่ง (Block Style) มีลักษณะของการเขียนบล็อกของคำสั่งในสองรูปแบบ ดังนี้

รูปแบบ Next-Line Style	รูปแบบ End-of-Line Style
<pre>public class Example { public static void main(String[] args) { System.out.println("Next Line Style"); } }</pre>	<pre>public class Example{ public static void main(String[] args){ System.out.println("Next Line Style"); } }</pre>

1.8 ความผิดพลาด และการแก้ไข

กระบวนการพัฒนาโปรแกรมประกอบด้วยขั้นตอนใหญ่ ๆ 4 ขั้นตอนดังนี้

- 1. Analysis & Design วิเคราะห์ปัญหาและออกแบบวิธีการแก้ปัญหา
- 2. Coding การเขียนโปรแกรม
- 3. Compile การคอมไพล์ จะมีการตรวจสอบ ว่าโปรแกรมเขียนถูกรูปแบบไวยากรณ์ของภาษาหรือไม่
- 4. Run การดำเนินการโปรแกรม เพื่อให้ได้ผลลัพธ์ตามที่ต้องการ

ขั้นตอนเหล่านี้จะทำตามลำดับ หากสำเร็จในแต่ละขั้นก็จะไปกระทำขั้นตอนต่อไป จนกว่าจะได้ผลลัพธ์ที่ต้องการ หากมีข้อผิดพลาดในแต่ละขั้นตอนก็จะมีอาการย้อนกลับไปแก้ไขในขั้นตอนก่อนหน้า เราสามารถแบ่งข้อผิดพลาดในแต่ละขั้นตอนเป็น 3 ประเภทใหญ่

1. Syntax Error รูปแบบไวยากรณ์ของโปรแกรมผิดพลาดเช่น พิมพ์ผิด หลงลืมเครื่องหมาย สาเหตุส่วนใหญ่มักเกิดจากโปรแกรมเมอร์ไม่ละเอียดรอบคอบ หรือไม่แม่นยำในรูปแบบไวยากรณ์ของภาษา เป็นข้อผิดพลาดที่ตรวจสอบได้จากขั้นตอนการคอมไพล์ โดยคอมไพเลอร์จะแสดงข้อความชี้แจงความผิดพลาดและระบุตำแหน่งที่ผิด โปรแกรมเมอร์จะต้องย้อนกลับไปแก้ไขโปรแกรมใหม่แล้วทำการคอมไพล์จนกว่าจะไม่มีข้อผิดพลาดหลงเหลืออยู่จึงจะนำไปรันได้

รูปแบบข้อความแสดง Syntax Error ของจาวา

ชื่อไฟล์ซอร์สโค้ด : เลขที่บรรทัด : ข้อความแสดงข้อผิดพลาด

โค้ดบรรทัดที่ผิดพลาด

^ ตัวชี้ตำแหน่งในบรรทัดที่ผิดพลาด

จำนวนข้อผิดพลาดที่พบ

ตัวอย่างที่ 1.2 Compilation Errors

```
public class ShowSyntaxErrors {
    public static void main(String[] args) {
        i = 30;
        System.out.println(i+4);
    }
}
```

จะปรากฏข้อผิดพลาดดังนี้

ShowSyntaxErrors.java:3: error: cannot find symbol

i = 30;

^

symbol: variable i

location: class ShowSyntaxErrors

ShowSyntaxErrors.java:4: error: cannot find symbol

System.out.println(i+4);

^

symbol: variable i

location: class ShowSyntaxErrors

2. Runtime Error ข้อผิดพลาดในขณะที่โปรแกรมทำงาน เป็นข้อผิดพลาดของการใช้คำสั่งที่ไม่ถูกต้อง สาเหตุส่วนใหญ่มักเกิดจากโปรแกรมเมอร์ใช้งานทรัพยากรระบบผิดพลาด เช่นการอ้างถึงหน่วยความจำที่ใช้ไม่ได้หรือไม่มี การเปิดไฟล์ที่ไม่มีอยู่ หรือการบันทึกไฟล์ไปยังปลายทางที่ไม่มี หรือแม้กระทั่งมีสาเหตุเกิดจากระบบผิดพลาดเช่นเครื่องคอมพิวเตอร์ติดไวรัส หน่วยความจำไม่เพียงพอ ฮาร์ดดิสก์เต็มหรือเสียหาย การแก้ไขก็ต้องตรวจสอบระบบว่ามีอะไรผิดพลาดหรือไม่ หรือย้อนกลับไปแก้ไขโปรแกรมให้ใช้ทรัพยากรระบบอย่างถูกต้อง

ตัวอย่างที่ 1.3 Runtime Errors

```
public class ShowRuntimeErrors {
    public static void main(String[] args) {
        int i = 1 / 0;
    }
}
```

จากตัวอย่างที่ 1.3 เมื่อคอมไพล์โปรแกรมจะไม่พบข้อผิดพลาดแต่หากนำโปรแกรมไปทำงานจะเกิดข้อผิดพลาดเนื่องจากการหารด้วยศูนย์ ซึ่งจะเกิด Runtime Error เกิดขึ้น

3. **Logic Error** ข้อผิดพลาดของวิธีการแก้ปัญหาที่หากนำมาเขียนโปรแกรมแล้วทำให้ได้ผลลัพธ์ไม่ตรงตามที่ต้องการ ต้องตรวจสอบว่าเราเขียนชุดคำสั่งถูกต้องตามวิธีที่ออกแบบไว้หรือไม่ หรืออีกสาเหตุหนึ่งคือเลือกใช้วิธีการแก้ปัญหาที่ผิดซึ่ง โปรแกรมเมอร์ต้องย้อนกลับไปยังขั้นตอนการวิเคราะห์ปัญหา และคิดวิธีการแก้ปัญหาใหม่และเขียนโปรแกรมให้ถูกต้องตามที่ออกแบบไว้

ตัวอย่างที่ 1.4 Logic Errors

```
public class ShowLogicErrors {
    // Determine if a number is between 1 and 100 inclusively
    public static void main(String[] args) {
        // Prompt the user to enter a number
        String input = JOptionPane.showInputDialog(null,
            "Please enter an integer:",
            "ShowLogicErrors", JOptionPane.QUESTION_MESSAGE);
        int number = Integer.parseInt(input);

        // Display the result
        System.out.println("The number is between 1 and 100, " +
            "inclusively? " + ((1 < number) && (number < 100)));

        System.exit(0);
    }
}
```

จากตัวอย่างที่ 1.4 เมื่อคอมไพล์โปรแกรมและรันโปรแกรมจะไม่พบข้อผิดพลาด แต่หากนำมาเขียนโปรแกรมแล้วทำให้ได้ผลลัพธ์ไม่ตรงตามที่ต้องการ จากตัวอย่างข้างต้นสิ่งที่ต้องการคือต้องการเช็คตัวเลขที่อยู่ระหว่าง 1 ถึง 100 โดยรวม 1 และ 100 แต่เงื่อนไขที่เขียนในโปรแกรมจะไม่รวม 1 และ 100 ทำให้เมื่อประมวลผลโปรแกรมดังกล่าวจะได้คำตอบที่ไม่ถูกต้อง

1.9 บทสรุป

การเขียนโปรแกรมเชิงวัตถุจะเป็นการมองปัญหาว่าประกอบไปด้วยวัตถุต่างๆ ซึ่งแนวคิดนี้จะเข้าใกล้เคียงกับธรรมชาติของมนุษย์มากที่สุดเนื่องจากมนุษย์มองสิ่งต่างๆ รอบตัวเป็นวัตถุทั้งที่เป็นรูปธรรม การพัฒนาโปรแกรมโดยใช้ภาษาคอมพิวเตอร์เชิงวัตถุจะทำให้ขบวนการพัฒนาโปรแกรมทำได้รวดเร็วขึ้นและสามารถปรับปรุงแก้ไขโปรแกรมได้ง่ายซึ่งเหมาะกับการพัฒนาโปรแกรมขนาดใหญ่ที่จะต้องมีการปรับปรุงแก้ไขโปรแกรมอยู่ตลอด นอกจากนี้โปรแกรมเชิงวัตถุยังมีคุณลักษณะเด่นอื่นๆ อีกดังนี้ การห่อหุ้ม (Encapsulation) การสืบทอด (Inheritance) การมีได้หลายรูปแบบ (Polymorphism) การมองแบบนามธรรม (Abstraction)

บทที่ 2 ชนิดข้อมูลแบบนามธรรมและชนิดข้อมูลแบบทั่วไป

วัตถุประสงค์

- 2.1 อธิบายความหมาย ชนิดข้อมูลแบบนามธรรมและชนิดข้อมูลแบบทั่วไป
- 2.2 พัฒนาชุดคำสั่งโดยใช้ชนิดข้อมูลแบบนามธรรมและชนิดข้อมูลแบบทั่วไป
- 2.3 สามารถอธิบาย ตัวแปรและชนิดข้อมูลในภาษาจาวา ชนิดข้อมูลแบบต่างๆ
- 2.4 สามารถเขียนโปรแกรมโดยใช้ตัวดำเนินการทางคณิตศาสตร์
- 2.5 สามารถเขียนชุดคำสั่งเพื่อรับค่าในรูปแบบ dos mode และในรูปแบบกราฟฟิก

2.1 ความนำ

ในการเขียนโปรแกรมที่มีประสิทธิภาพจำเป็นต้องมีการกำหนดรูปแบบของตัวแปรและมีการใช้ตัวดำเนินการทางคณิตศาสตร์ที่เหมาะสมเพื่อให้สามารถทำให้การดำเนินการของโปรแกรมให้ผลที่ถูกต้อง โดยทั่วไปชนิดของข้อมูลจะแบ่งออกเป็น 2 รูปแบบคือ ชนิดข้อมูลแบบนามธรรมและชนิดข้อมูลแบบทั่วไป

2.2 ชนิดข้อมูลแบบนามธรรม

ข้อมูลในระดับความคิด (Abstract data types) เป็นระดับของแบบชนิดข้อมูลประเภทนามธรรมที่สร้างขึ้นจากจินตนาการของผู้ใช้ เป็นแบบชนิดข้อมูลที่ไม่มีการรูปร่างหรือลักษณะให้เห็น การอธิบายลักษณะของข้อมูลจะใช้สัญลักษณ์ ถ้าต้องการทำให้แบบชนิดข้อมูลที่เป็นนามธรรมเป็นข้อมูลที่เป็นรูปธรรมต้องนำไปใช้จริงกับเครื่องคอมพิวเตอร์จริง (Implement) ข้อมูลในระดับความคิดแบ่งออกเป็น 2 ประเภท คือ

โครงสร้างข้อมูลแบบเชิงเส้น (Linear data structures) เช่น ลิสต์ (list) สแตก (stack) คิว (queue) เดค (deque)

โครงสร้างข้อมูลแบบไม่ใช่เชิงเส้น (Non-linear data structures) เช่น ตรี (tree) กราฟ (graph)

จากข้อมูลในแต่ละระดับที่กล่าวมา ข้อมูลในระดับระดับความคิด นักเขียนโปรแกรม (Programmer) เป็นผู้เขียนโปรแกรมสร้างข้อมูลขึ้นมาเพื่อใช้เป็นตัวเชื่อมความสัมพันธ์ไปสู่ข้อมูลในระดับโปรแกรม ส่วนโปรแกรมแปลภาษาทำหน้าที่เป็นตัวเชื่อมความสัมพันธ์จากข้อมูลในระดับโปรแกรม ไปสู่ข้อมูลในระดับเครื่องที่เครื่องบันทึกข้อมูลโดยใช้รหัสแทนข้อมูลจริง

2.3 ชนิดข้อมูลแบบทั่วไป

1. ข้อมูลเบื้องต้น (Primitive data types) เป็นข้อมูลพื้นฐานซึ่งมีโครงสร้างข้อมูลไม่ซับซ้อนจะต้องมีในภาษาคอมพิวเตอร์ทุกภาษา ตัวอย่างของข้อมูลประเภทนี้ เช่น จำนวนเต็ม (integer) จำนวนจริง (real) ตัวอักษร (character)

2. ข้อมูลโครงสร้าง (Structured data types) เป็นข้อมูลที่มีโครงสร้างสลับซับซ้อนเกิดจากการนำโครงสร้างข้อมูลเบื้องต้นมาประกอบกันเป็นโครงสร้างข้อมูลที่หลากหลายขึ้น ข้อมูลที่ใช้ในเครื่องคอมพิวเตอร์ยุคแรกเป็นข้อมูลเบื้องต้นเท่านั้น แต่ในปัจจุบันภาษาคอมพิวเตอร์เกือบทุกภาษามีข้อมูลโครงสร้างด้วยแทบทั้งสิ้น ตัวอย่างข้อมูลโครงสร้าง เช่น แถวลำดับ (array) เซต (set) ระเบียบข้อมูล (record) แฟ้มข้อมูล (file)

2.4 ตัวแปรและชนิดข้อมูลในภาษาจาวา

ส่วนประกอบหลัก ๆ ของโปรแกรมภาษาจาวาประกอบด้วย

1. Comments ข้อความที่แทรกอยู่ภายในโปรแกรมซึ่งคอมไพเลอร์จะไม่แปลข้อความนี้ให้เป็นส่วนหนึ่งของโปรแกรม
2. Reserved word คำสงวน
3. Modifiers คีย์เวิร์ดที่ใช้ในการบอกคุณลักษณะของตัวแปรหรือเมธอด
4. Statements ชุดคำสั่ง
5. Blocks กลุ่มคำสั่ง
6. Classes คลาส
7. Methods เมธอด
8. The main method เมธอดหลัก

โดยรายละเอียดของตัวแปรและชนิดข้อมูลในภาษาจาวามีรายละเอียดดังนี้

2.4.1 ชื่อที่ตั้งขึ้นในภาษาจาวา (Identifiers)

Identifier คือชื่อที่ตั้งขึ้นในภาษาจาวา ซึ่งอาจเป็นชื่อของคลาส ชื่อของตัวแปร ชื่อของเมธอด หรือชื่อของค่าคงที่ ซึ่งจะต้องเป็นไปตามกฎการตั้งชื่อ ดังนี้

identifier จะต้องขึ้นต้นด้วยอักขระ A-Z, a-z, _ หรือ \$ เท่านั้น

identifier ที่ประกอบไปด้วยตัวอักขระมากกว่าหนึ่งตัว ตัวอักขระหลังจากตัวแรกนั้นจะต้องเป็นตัวอักขระข้างต้น หรือเป็นตัวเลข 0 ถึง เท่านั้น 9

identifier จะต้องไม่ตรงกับคำสงวน

identifier จะต้องไม่ใช่ true false หรือ null

identifier ในภาษาจาวาจะถือว่าตัวอักษรพิมพ์ใหญ่และตัวอักษรพิมพ์เล็กต่างกัน(case sensitive) ดังนั้น identifier ที่ชื่อ myVariable จะแตกต่างจาก MyVariable

2.4.2 ตัวแปร (Variables)

ตัวแปรใช้สำหรับเก็บข้อมูลในโปรแกรม สามารถเปลี่ยนค่าได้

ตัวอย่างที่ 2.1 แสดงตัวอย่างการประกาศตัวแปรสำหรับการคำนวณพื้นที่ของวงกลม

```
public class ComputeArea{
    public static void main(String[] args){
        // Compute the first area
        double radius = 1.0;
        double area = radius*radius*3.14159;
        System.out.println("The area is" + area + " for radius" +radius);
        // Compute the second area
        radius = 2.0;
        area = radius*radius*3.14159;
        System.out.println("The area is" + area + "for radius" +radius);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
The area is3.14159 for radius1.0
The area is12.56636for radius2.0
```

ตัวอย่างที่ 2.1 แสดงตัวอย่างการประกาศตัวแปรสำหรับการคำนวณพื้นที่ของวงกลมโดยจากตัวอย่างข้างต้นจะเป็นการประกาศตัวแปรชื่อ radius และ area ซึ่งมีประเภทเป็นทศนิยม (double)

ขั้นตอนในการดำเนินการกับตัวแปรประกอบด้วยขั้นตอนดังนี้

1. การประกาศตัวแปร (Declaring variable) สามารถประกาศได้ตามตัวอย่างต่อไปนี้

```
int x;           // ประกาศให้ตัวแปร x เก็บเลขจำนวนเต็ม integer
double radius;  // ประกาศให้ตัวแปร radius เก็บเลขจำนวนจริง double
char a;         // ประกาศให้ตัวแปร a เก็บตัวอักษร character
```

2. การกำหนดค่าให้กับตัวแปร (Assignment Statements)

```
x = 1;          // Assign 1 to x;
radius = 1.0;   // Assign 1.0 to radius;
a = 'A';        // Assign 'A' to a;
```

3. การประกาศตัวแปรและกำหนดค่าให้กับตัวแปรใน 1 บรรทัด (Declaring and Initializing in One Step)

```
int x = 1;
double d = 1.4;
float f = 1.4;
```

2.5 ตัวแปรค่าคงที่ (Constants)

ตัวแปรค่าคงที่คือค่าที่ใช้แสดงข้อมูลที่เป็นตัวเลข ตัวอักษร ข้อความ หรือค่าทางตรรกะ ประกอบด้วย

ตรรกะ (boolean)

ตัวอักษร (character)

ตัวเลขจำนวนเต็ม (integral)

ตัวเลขทศนิยม (floating point)

ข้อความ (string)

การประกาศตัวแปรค่าคงที่จะใช้คีย์เวิร์ดคำว่า final เป็นตัวระบุว่าเป็นตัวแปรนั้นเป็นตัวแปรแบบค่าคงที่ โดยมีรูปแบบการประกาศค่าตัวแปรค่าคงที่ทำได้ดังนี้

```
final datatype CONSTANTNAME = VALUE;
```


ตัวอย่างที่ 2.2 การประกาศตัวแปรค่าคงที่

```
public class Ex2_2{
    public static void main(String[] args){
        final double PI = 3.14159;
        final int SIZE = 3;
    }
}
```

ตัวอย่างที่ 2.2 แสดงตัวอย่างการประกาศตัวแปรค่าคงที่ชื่อ PI ซึ่งมีประเภทเป็นทศนิยม (double) และกำหนดค่าเป็น 3.14159 และตัวแปรค่าคงที่ชื่อ SIZE ซึ่งมีประเภทเป็นตัวเลขจำนวนเต็ม (int) และกำหนดค่าเป็น 3

2.6 ค่าคงที่ (Number Literals)

2.6.1 ค่าคงที่ตัวเลข (Number Literals) ค่าคงที่คือตัวเลขที่ปรากฏโดยตรงในโปรแกรม เช่น 34, 1,000,000, และ 5.0 เป็นค่าคงที่ที่ปรากฏในคำสั่งหรือค่าคงที่คือข้อมูลที่เรารวบรวมให้กับคอมพิวเตอร์เพื่อให้คอมพิวเตอร์นำไปใช้ในการคำนวณตามที่เรากำลังต้องการมี 5 ชนิดหลัก ๆ คือ Integer, floating point, string, logical, character

ตัวอย่างที่ 2.3 ค่าคงที่ (Number Literals)

```
public class Ex2_3{
    public static void main(String[] args){
        int i = 34;
        long l = 1000000;
        double d = 5.0;
    }
}
```

ตัวอย่างที่ 2.3 แสดงตัวอย่างการประกาศตัวแปรที่ชื่อ i ซึ่งมีประเภทเป็นจำนวนเต็ม (int) และกำหนดค่าเป็น 34 และตัวแปรชื่อ l ซึ่งมีประเภทเป็นตัวเลขจำนวนเต็ม (long) และกำหนดค่าเป็น 1000000 และตัวแปรชื่อ d ซึ่งมีประเภทเป็นตัวเลขทศนิยม (double) และกำหนดค่าเป็น 5.0

ค่าคงที่ตัวเลข (Integer Literals) สามารถกำหนดให้กับตัวแปรตัวเลขได้ ถ้าค่าคงที่ตัวเลขดังกล่าวมีขนาดหรือจำนวนไบต์ที่สามารถเก็บไว้ในตัวแปรได้ กรณีที่ค่าคงที่ตัวเลขมีขนาดใหญ่มากกว่าตัวแปรที่จะเก็บได้จะเกิด compile error เช่น การกำหนดคำสั่ง byte b = 1000 จะเกิด compile error ไม่สามารถจะเก็บค่าคงที่ตัวเลข 1000 ในตัวแปร b ที่มีชนิดข้อมูลเป็น byte ได้เนื่องจากช่วงของการเก็บข้อมูลของชนิดข้อมูลแบบ byte จะสามารถเก็บข้อมูลในช่วง $-2^7 - 1$ ถึง $2^7 - 1$ ค่าคงที่ตัวเลขจะกำหนดให้เป็น int เมื่อค่าของตัวเลขดังกล่าวอยู่ระหว่าง -2^{31} (-2147483648) ถึง $2^{31} - 1$ (2147483647) ค่าคงที่ตัวเลขจะมีชนิดข้อมูลเป็น long เมื่อเติม L หรือ l ต่อท้าย ซึ่งจะทำให้เก็บข้อมูลได้มากขึ้น

ค่าคงที่เลขทศนิยม (Floating-Point Literals) จะเป็นค่าคงที่ตัวเลขที่เขียนในรูปของจุดทศนิยม การเก็บตัวเลขค่าคงที่ดังกล่าวจะถูกเก็บในชนิดข้อมูลแบบ double เช่น 5.0 จะถูกมองในรูปของ double ไม่ใช่ float ดังนั้นเราสามารถแปลงตัวเลขที่เป็น double ให้เป็น float ได้โดยการเติมตัวอักษร f หรือ F และกรณีที่ต้องการแปลงตัวเลขดังกล่าวให้อยู่ในรูปของ double จะทำได้โดยการเติม d หรือ D เช่น สามารถใช้ 100.2f หรือ 100.2F สำหรับ float และ 100.2d หรือ 100.2D สำหรับ double

สัญลักษณ์ทางวิทยาศาสตร์ (Scientific Notation) ค่าคงที่ทศนิยมจะสามารถนำมาเขียนในรูปของสัญลักษณ์ทางวิทยาศาสตร์ได้ เช่น 1.23456e+2, หรือ 1.23456e2, จะมีค่าเท่ากับ 123.456 และ 1.23456e-2 จะมีค่าเท่ากับ 0.0123456 โดย E หรือ e สามารถที่จะใช้แทนเลขยกกำลังโดยสามารถเขียนในรูปของตัวเล็กหรือตัวใหญ่ได้

2.7 ชนิดข้อมูลแบบพื้นฐาน ชนิดข้อมูลแบบอ้างอิง

2.7.1 ชนิดข้อมูลแบบพื้นฐาน ภาษาจาวาเป็นภาษาที่ต้องระบุชนิดข้อมูลอย่างชัดเจน (Strongly typed language) โดยประกอบด้วยชนิดข้อมูลแบบพื้นฐาน และชนิดข้อมูลแบบอ้างอิง โดยชนิดข้อมูลแบบพื้นฐาน (primitive data type) ประกอบด้วย

ชนิดข้อมูลตรรกะ

ชนิดข้อมูลอักขระ

ชนิดข้อมูลตัวเลข ซึ่งประกอบด้วยชนิดข้อมูลตัวเลขจำนวนเต็ม และชนิดข้อมูลตัวเลขทศนิยม

2.7.2 ชนิดข้อมูลแบบอ้างอิง (reference data type) ประกอบด้วย ชนิดข้อมูลที่เป็นคลาส ชนิดข้อมูลที่เป็นอาร์เรย์

รายละเอียดของชนิดข้อมูลแต่ละประเภทมีดังนี้

ข้อมูลแบบตัวเลข (number)

ชนิดข้อมูล	จำนวนบิต
byte	8 bits
short	16 bits
int	32 bits
long	64 bits
float	32 bits
double	64 bits

ในการคำนวณกับชนิดข้อมูลแบบตัวเลขจะมีการใช้เครื่องหมาย (operator) ดังนี้ คือ $+$ $-$ $*$ $/$ และ $\%$ เมื่อนำชนิดข้อมูลตัวเลขจำนวนเต็ม integer กระทบกันจะได้ผลลัพธ์เป็นชนิดข้อมูลแบบ integer เช่น $5/2$ จะได้ 2 ส่วนกรณีที่มีตัวใดตัวหนึ่งเป็น float จะทำให้ผลลัพธ์ที่ได้เป็นเลขทศนิยม เช่น $5.0/2$ จะได้ 2.5 ส่วนการหารเอาเศษ (Modulation) จะกระทบกับชนิดข้อมูลจำนวนเต็มเท่านั้น เช่น $5\%2$ จะได้ผลลัพธ์เป็น 1 การคำนวณจำนวนทศนิยมผลลัพธ์ที่ได้จะเป็นค่าโดยประมาณเนื่องจากค่าที่เก็บในตัวแปรจะเป็นค่าโดยประมาณ

ตัวอย่างที่ 2.4 การคำนวณกับชนิดข้อมูลแบบตัวเลข

```
public class Ex2_4{
    public static void main(String[] args){
        System.out.println(1 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
        System.out.println(1.0 - 0.9);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
0.50000000000000001
0.09999999999999998
```

ตัวอย่างที่ 2.4 แสดงตัวอย่างการคำนวณกับชนิดข้อมูลแบบตัวเลข

ชนิดข้อมูลแบบอักขระ (Character Data Type)

การเก็บชนิดข้อมูลแบบอักขระอาจจะเก็บในรูปของรหัสแอสกีหรือรหัส Unicode ก็ได้ หรืออาจจะเก็บตัวอักษรโดยตรง ตัวอย่างของการประกาศชนิดข้อมูลแบบตัวอักษรสามารถทำได้ดังตัวอย่างต่อไปนี้ โดยกรณีที่ต้องการประกาศตัวอักขระพิเศษสามารถประกาศได้คล้ายกัน

ตัวอย่างที่ 2.5 ชนิดข้อมูลแบบอักขระ (Character Data Type)

```
public class Ex2_5{
    public static void main(String[] args){
        char letter = 'A'; // (ASCII)
        char numChar = '4'; // (ASCII)
        char letterUni = '\u0041'; // (Unicode)
        char numCharUni = '\u0034'; // (Unicode)
        char tab = '\t';
    }
}
```

ตัวอย่างที่ 2.5 แสดงตัวอย่างการเก็บชนิดข้อมูลแบบอักขระอาจจะเก็บในรูปของรหัสแอสกีหรือรหัส Unicode ก็ได้ จากตัวอย่าง char tab = '\t'; เป็นการประกาศให้ตัวแปร tab ทำการเว้นระยะห่าง 5 ตัวอักษร หรือ 1 tab

โดยมีตัวอย่างของตัวอักขระพิเศษดังนี้

Description	Escape Sequence	Unicode
Backspace	\b	\u0008
Tab	\t	\u0009
Linefeed	\n	\u000a
Carriage return	\r	\u000d

ตารางรหัสแอสกี (ASCII Character Set)

รหัสแอสกีจะเป็นกลุ่มของรหัสที่เป็นเซตย่อยของ Unicode จาก \u0000 to \u007f โดยมีตัวอย่างของรหัสแอสกีดังนี้

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	\$	%	&	'
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	del		

การแปลงข้อมูลระหว่างชนิดข้อมูลตัวอักษรและชนิดข้อมูลตัวเลข

การแปลงชนิดข้อมูลตัวอักษรและชนิดข้อมูลตัวเลขสามารถทำได้โดยการระบุชนิดข้อมูลที่ต้องการหน้าตัวแปรที่ต้องการแปลง โดยการแปลงชนิดข้อมูลจะมีรูปแบบดังนี้คือ Explicit Conversion หมายถึง การแปลงชนิดของข้อมูลโดยต้องประกาศ Implicit Conversion หมายถึง การแปลงชนิดของข้อมูลโดยไม่ต้องประกาศ และไม่ใช่ทุกชนิดข้อมูลจะแปลงไปอีกชนิดข้อมูลได้ แต่ข้อมูลนั้นต้องสมเหตุผลในการแปลงด้วย เช่น

`int i = 'a';` มีความหมายเดียวกับ `int i = (int)'a';` โดยการเขียนคำสั่ง `int i = 'a';` เป็นการแปลงแบบ Implicit Conversion ส่วน `int i = (int)'a';` เป็นการแปลงแบบ Explicit Conversion

`char c = 97;` มีความหมายเดียวกับ `char c = (char)97;` โดยการเขียนคำสั่ง `char c = 97;` เป็นการแปลงแบบ Implicit Conversion ส่วน `char c = (char)97;` เป็นการแปลงแบบ Explicit Conversion

ชนิดข้อมูลแบบตรรกะ (boolean)

ในภาษาจาวาจะมีชนิดข้อมูลแบบตรรกะโดยชนิดข้อมูลแบบนี้เป็นชนิดข้อมูลที่มีค่าความเป็นจริงอยู่ 2 ค่าคือ จริง (true) และเท็จ (false) เช่น

`boolean lightsOn = true;` เป็นการประกาศสถานะการเปิดไฟซึ่งเก็บในตัวแปร `lightsOn` เป็นค่าจริง

`boolean lightsOn = false;` เป็นการประกาศสถานะการเปิดไฟซึ่งเก็บในตัวแปร `lightsOn` เป็นค่าเท็จ

`boolean b = (1 > 2);` เป็นการเช็คค่า `1 > 2` หรือไม่ โดยจะเก็บค่าความจริงจากการเปรียบเทียบซึ่งเก็บในตัวแปร `b` เป็นค่าเท็จ

ในกรณีที่มีการเช็คเงื่อนไขมากกว่า 1 เงื่อนไขจะใช้เครื่องหมาย `&&` (and) `||` (or) `!` (not) เช่น

กรณี `x` เป็นตัวแปรประเภทตัวเลขหากต้องการตรวจสอบว่า `x` อยู่ในช่วงระหว่าง 1 ถึง 100 หรือไม่จะใช้คำสั่ง `(1 < x) && (x < 100)`

กรณีต้องการเช็คเงื่อนไขว่าสถานะการเปิดไฟเป็นจริงหรือช่วงเวลาเป็นช่วงเวลากลางวันเป็นจริงจะใช้คำสั่ง `(lightsOn) || (isDayTime)`

2.8 ตัวดำเนินการทางคณิตศาสตร์

2.8.1 เครื่องหมายเปรียบเทียบ (Comparison Operators) เป็นเครื่องหมายที่ช่วยในการเปรียบเทียบจะมีการใช้เครื่องหมายดังนี้

Operator	Name
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

เครื่องหมายที่ช่วยในการกำหนดค่าทางตรรกะ **Boolean Operators** มีดังนี้

Operator	Name
!	not
&&	and
	or
^	exclusive or

ตารางค่าความจริงของการใช้เครื่องหมายทางตรรกะมีดังนี้

Operand1	Operand2	!operand1	Operand1 && Operand2	Operand1 Operand2	Operand1 ^ Operand2
false	false	true	false	false	false
false	true	true	false	true	true
true	false	false	false	true	true
true	true	false	true	true	false

2.8.2 ลำดับความสำคัญของโอเปอเรเตอร์ (Operator Precedence)

ลำดับความสำคัญ	โอเปอเรเตอร์
1	var++, var--
2	+, - (Unary plus and minus), ++var, --var
3	(type) Casting
4	! (Not)
5	*, /, % (Multiplication, division, and modulus)
6	+, - (Binary addition and subtraction)
7	<, <=, >, >= (Comparison)
8	==, !=; (Equality)
9	& (Unconditional AND)
10	^ (Exclusive OR)
11	(Unconditional OR)
12	&& (Conditional AND) Short-circuit AND
13	(Conditional OR) Short-circuit OR
14	=, +=, -=, *=, /=, %= (Assignment operator)

Operator Associativity

เมื่อโอเปอเรเตอร์ที่มีลำดับความสำคัญเท่ากันเขียนต่อกันส่วนมากจะมีการเรียงลำดับการทำงาน จากซ้ายไปขวา ที่เรียกว่า Left-associative ยกเว้นการใช้เครื่องหมายเท่ากับเมื่อกำหนดค่าจะทำจากขวาไปซ้ายที่เรียกว่า right-associative เช่น

$a - b + c - d$ จะมีค่าเท่ากับ $((a - b) + c) - d$

$a = b += c = 5$ จะมีค่าเท่ากับ $a = (b += (c = 5))$

ลำดับการคำนวณของโอเปอแรนด์ (Operand Evaluation Order)

กฎของ Precedence และ associative rule จะกำหนดลำดับการทำงานของโอเปอเรเตอร์ แต่ไม่ได้กำหนดลำดับของ operand ที่จะคำนวณ โดยปกติในภาษาจาวา operand จะคำนวณจากซ้ายไปขวา โอเปอแรนด์ที่อยู่ด้านซ้ายของเครื่องหมายจะถูกคำนวณก่อนโอเปอเรนด์ที่อยู่ด้านขวา กรณีที่โอเปอเรนด์มี side effect คือมีการเปลี่ยนค่าของตัวแปรก่อนค่อยนำไปคำนวณในคำสั่งเดียวกัน หากมีการเรียงลำดับเครื่องหมายในการคำนวณไม่ถูกต้องจะทำให้ค่าของตัวแปรไม่ถูกต้องตามความต้องการจริง

ตัวอย่างที่ 2.6 ลำดับการคำนวณของโอเปอเรนด์ (Operand Evaluation Order)

```
public class Ex2_6{
    public static void main(String[] args){
        int a = 0;
        int x = a + (++a);
        System.out.println("x =" + x);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

x =1

จากตัวอย่าง x จะกลายเป็น 1 เมื่อทำตามคำสั่งข้างต้น เนื่องจากเริ่มต้น a จะมีค่าเป็น 0 หลังจากนั้นจะนำค่าที่อยู่ 0 ใน a คือ 0 ไปบวกกับค่าภายหลังการเพิ่มค่า a จากคำสั่ง ++a ซึ่งจะได้ 1 ทำให้ผลที่ได้คือ 0+1 ซึ่งมีค่าเป็น 1 หลังจากนั้นจะนำเลข 1 ดังกล่าวไปเก็บในตัวแปร x

ตัวอย่างที่ 2.7 ลำดับการคำนวณของโอเปอเรนด์ (Operand Evaluation Order)

```
public class Ex2_7{
    public static void main(String[] args){
        int a = 0;
        int x = ++a + a;
        System.out.println("x = " + x);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

x =2

จากตัวอย่าง x จะกลายเป็น 2 เมื่อทำตามคำสั่งข้างต้น เนื่องจากเริ่มต้น a มีค่าเป็น 0 หลังจากนั้นจะทำคำสั่ง ++a คือเพิ่มค่า a ก่อนทำให้ a มีค่าเป็น 1 แล้วนำค่า a มาบวกต่อซึ่งกลายเป็น 1+1 จะได้ค่าเป็น 2 หลังจากนั้นจะนำ 2 ดังกล่าวไปเก็บไว้ในตัวแปร x

2.8.3 ตัวดำเนินการทางคณิตศาสตร์แบบย่อ (Shortcut Assignment Operators)

เมื่อต้องการเขียนโปรแกรมให้สั้นลงสามารถทำได้โดยการใช้ตัวดำเนินการคณิตศาสตร์แบบย่อดังนี้

Operator	Example	Equivalent
+=	i+=8	i = i+8
-=	f-=8.0	f = f-8.0
=	i=8	i = i*8
/=	i/=8	i = i/8
%=	i%=8	i = i%8

2.8.4 ตัวดำเนินการเพิ่มค่าและลดค่า (Increment and Decrement Operators) ในการเพิ่มค่าหรือลดค่าสามารถเขียนในรูปแบบ suffix หรือ prefix ได้โดย

ตัวดำเนินการเพิ่มค่าและลดค่า	รูปแบบ	ความหมาย
x++	Suffix	x=x+1
++x	prefix	x=x+1
x--	Suffix	x=x-1
--x	prefix	x=x-1

1. Suffix การเขียนคำสั่งในลักษณะนี้จะเป็นการนำค่าในตัวแปรมาใช้ก่อนแล้วค่อยเพิ่มค่าหรือลดค่า

2. Prefix การเขียนคำสั่งในลักษณะนี้จะเป็นการเพิ่มค่าของตัวแปรก่อนนำตัวแปรมาใช้งาน

ตัวอย่างที่ 2.8 ตัวดำเนินการเพิ่มค่าและลดค่า

เขียนในรูปแบบ suffix หรือ prefix	เขียนในรูปแบบปกติ
<pre>public class Example{ public static void main(String[] args){ byte i = 10; int newNum = 10*i++; System.out.println(newNum); } }</pre>	<pre>public class Example{ public static void main(String[] args){ byte i = 10; int newNum = 10*i; i++; System.out.println(newNum); } }</pre>
<pre>public class Example{ public static void main(String[] args){ byte i = 10; int newNum = 10*(++i); System.out.println(newNum); } }</pre>	<pre>public class Example{ public static void main(String[] args){ byte i = 10; i=i+1; int newNum = 10*(i); System.out.println(newNum); } }</pre>

การใช้ตัวดำเนินการเพิ่มค่าถึงแม้ว่าจะทำให้คำสั่งสั้นลงแต่จะทำให้เข้าใจยาก ให้หลีกเลี่ยงการใช้คำสั่งในรูปแบบนี้เมื่อมีการเปลี่ยนแปลงค่าตัวแปรหลายตัวหรือในคำสั่งเดียวที่มีการใช้ตัวแปรซ้ำหลายครั้ง เช่น

```
int k = ++i + i
```

2.9 การแปลงชนิดข้อมูล (Numeric Type Conversion)

ในภาษาจาวาจะมีการแปลงชนิดข้อมูลโดยมีลำดับของการแปลงชนิดข้อมูลหรือ Type casting ดังต่อไปนี้ double float long int short byte

การแปลงข้อมูลที่กว้างขึ้น (Widening conversion) คือการแปลงจากชนิดข้อมูลที่มีขนาดเล็กกว่าไปเป็นชนิดข้อมูลที่มีขนาดใหญ่กว่า

การแปลงข้อมูลที่แคบลง (narrowing conversion) คือการแปลงจากชนิดข้อมูลที่มีขนาดใหญ่กว่าไปเป็นชนิดข้อมูลที่มีขนาดเล็กลง ซึ่งอาจมีผลให้เสียความละเอียดของข้อมูลบางส่วนไป

ตัวอย่างที่ 2.9 การแปลงชนิดข้อมูล (Numeric Type Conversion)

```
public class Ex2_10{
    public static void main(String[] args){
        byte i = 100;
        long k = i*3+4;
        double d = i*3.1+k/2;
        int x=k;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Ex2_9.java:6: error: incompatible types: possible lossy conversion from long to int
        int x=k;
            ^
1 error
```

จากตัวอย่างที่ 2.9 เป็นตัวอย่างของการแปลงข้อมูล เมื่อมีการใช้คำสั่ง `int x = k;` จะไม่ถูกต้องเนื่องจาก `k` มีขนาดใหญ่กว่าชนิดข้อมูลของ `x`

ตัวอย่างที่ 2.10 ตัวอย่างของการแปลงข้อมูล

```
public class Ex2_10{
    public static void main(String[] args){
        byte i = 100;
        long k = i*3+4;
        double d = i*3.1+k/2;
        int x=10;
        long g = x;
    }
}
```

จากตัวอย่างที่ 2.10 เป็นตัวอย่างของการแปลงข้อมูล เมื่อมีการใช้คำสั่ง `long k = x;` จะถูกต้องเนื่องจากเป็นการแปลงโดยอัตโนมัติ (implicit casting)

ตัวอย่างที่ 2.11 ตัวอย่างของการแปลงข้อมูล

```
public class Ex2_11{
    public static void main(String[] args){
        double d = 3;
    }
}
```

จากตัวอย่างที่ 2.11 เป็นตัวอย่างของการแปลงข้อมูล เมื่อมีการใช้คำสั่ง `double d = 3;` จะถูกต้องเนื่องจากเป็นการแปลงโดยอัตโนมัติ(implicit casting) การแปลงข้อมูลที่กว้างขึ้น (Widening conversion) คือการแปลงจากชนิดข้อมูลที่มีขนาดเล็กกว่าไปเป็นชนิดข้อมูลที่มีขนาดใหญ่กว่า

ตัวอย่างที่ 2.12 ตัวอย่างของการแปลงข้อมูล

```
public class Ex2_12{
    public static void main(String[] args){
        int i = (int)3.0;
    }
}
```

จากตัวอย่างที่ 2.12 เป็นตัวอย่างของการแปลงข้อมูล เมื่อมีการใช้คำสั่ง `int i = (int)3.0;` จะถูกต้องเนื่องจากเป็นการแปลงแบบชัดแจ้ง (Explicit casting) เป็นการแปลงข้อมูลที่แคบลง (narrowing conversion) คือการแปลงจากชนิดข้อมูลที่มีขนาดใหญ่กว่าไปเป็นชนิดข้อมูลที่มีขนาดเล็กลง ซึ่งอาจมีผลให้เสียความละเอียดของข้อมูลบางส่วนไป

ตัวอย่างที่ 2.13 ตัวอย่างของการแปลงข้อมูล

```
public class Ex2_13{
    public static void main(String[] args){
        int x = 5/2.0;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Ex2_13.java:3: error: incompatible types: possible lossy conversion from double to int
    int x = 5/2.0;
           ^
1 error
```

จากตัวอย่างที่ 2.13 เป็นตัวอย่างของการแปลงข้อมูล เมื่อมีการใช้คำสั่ง `int x = 5/2.0;` จะไม่ถูกต้องเนื่องจากผลลัพธ์จากการหารมีขนาดใหญ่กว่าชนิดข้อมูลของ `x`

2.10 การเขียนชุดคำสั่งเพื่อรับค่าในรูปแบบ dos mode และในรูปแบบกราฟฟิก

2.10.1 การรับข้อมูลผ่านทางคีย์บอร์ด

ผู้พัฒนาโปรแกรมสามารถรับข้อมูลโดยใช้เมธอดในคลาส `Scanner` ดังตัวอย่างโปรแกรมต่อไปนี้จะให้ผู้ใช้ป้อนอัตราดอกเบี้ย จำนวนปี และยอดเงินยืมเพื่อจะคำนวณจำนวนเงินที่จะต้องจ่ายรายเดือนและผลรวมของจำนวนเงินทั้งหมดที่ต้องจ่ายพร้อมดอกเบี้ยทั้งหมดโดยใช้เมธอดในคลาส `Scanner`

ตัวอย่างที่ 2.14 คำนวณจำนวนเงินที่จะต้องจ่ายรายเดือนและผลรวมของจำนวนเงินทั้งหมดที่ต้องจ่ายพร้อมดอกเบี้ยทั้งหมดโดยใช้ เมธอดในคลาส `Scanner`

```
import java.util.Scanner;
public class Ex2_14 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter yearly interest rate: ");
        double annualInterestRate = input.nextDouble();
        double monthlyInterestRate = annualInterestRate / 1200;
        System.out.print("Enter number of years: ");
        int numberOfYears = input.nextInt();
        System.out.print("Enter loan amount: ");
        double loanAmount = input.nextDouble();

        double monthlyPayment = loanAmount * monthlyInterestRate / (1 - 1 / Math.pow(1 +
monthlyInterestRate, numberOfYears * 12));
        double totalPayment = monthlyPayment * numberOfYears * 12;
        monthlyPayment = (int)(monthlyPayment * 100) / 100.0;
        totalPayment = (int)(totalPayment * 100) / 100.0;

        System.out.println("The monthly payment is " + monthlyPayment);
        System.out.println("The total payment is " + totalPayment);
    }
}
```

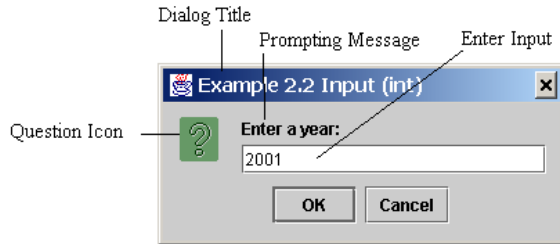
ผลลัพธ์ที่เกิดขึ้นคือ

```
Enter yearly interest rate: 10
Enter number of years: 5
Enter loan amount: 50000
The monthly payment is 1062.35
The total payment is 63741.13
```

2.10.2 การเขียนคำสั่งเพื่อรับค่าในรูปแบบกราฟฟิก

รูปแบบการรับค่าเพื่อจะนำไปคำนวณโดยใช้ในรูปแบบกราฟฟิก จะมีรูปแบบการทำงานดังนี้

```
String string = JOptionPane.showInputDialog ( null, "Prompt Message", "Dialog Title",
JOptionPane.QUESTION_MESSAGE);
```



เมื่อมีการรับข้อมูลผ่าน input dialog box ค่าที่ได้จะเป็นตัวอักษรดังนั้นจึงจำเป็นต้องมีการแปลงข้อมูลให้เป็นตัวเลขจึงจะสามารถนำเอาค่าดังกล่าวไปทำการคำนวณได้ ในการแปลงจาก String เป็น int สามารถทำได้โดยใช้ static method ที่ชื่อ parseInt ที่อยู่ในคลาส Integer เช่น

```
int intValue = Integer.parseInt(intString);
```

เมื่อ intString เป็นตัวอักขระตัวเลข เช่น “123” ในการแปลงจาก String เป็น double สามารถทำได้โดยใช้ static method ที่ชื่อ parseDouble ที่อยู่ในคลาส Double เช่น

```
double doubleValue = Double.parseDouble(doubleString);
```

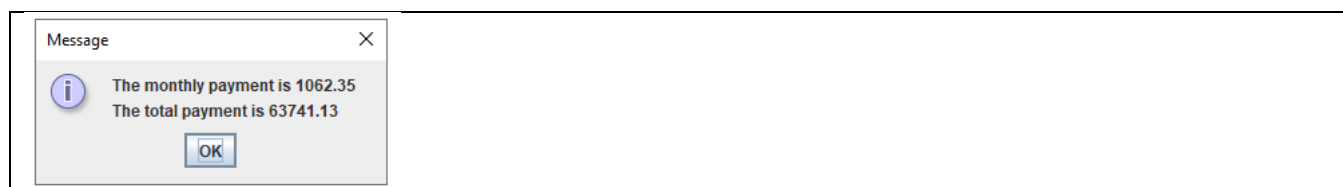
เมื่อ doubleString เป็นตัวอักขระตัวเลข เช่น “123.45”

ตัวอย่างที่ 2.15 ตัวอย่างโปรแกรมที่จะให้ผู้ใช้อัตราดอกเบี้ย จำนวนปี และยอดเงินยืมเพื่อจะคำนวณจำนวนเงินที่ต้องจ่ายรายเดือนและผลรวมของจำนวนเงินทั้งหมดที่ต้องจ่ายพร้อมดอกเบี้ยทั้งหมดโดยใช้ เมธอดในคลาส

JOptionPane

```
import javax.swing.JOptionPane;
public class Ex2_15 {
    public static void main(String[] args) {
        String annualInterestRateString = JOptionPane.showInputDialog(
            "Enter yearly interest rate, for example 8.25:");
        double annualInterestRate =
            Double.parseDouble(annualInterestRateString);
        double monthlyInterestRate = annualInterestRate / 1200;
        String numberOfYearsString = JOptionPane.showInputDialog(
            "Enter number of years as an integer, \nfor example 5:");
        int numberOfYears = Integer.parseInt(numberOfYearsString);
        String loanString = JOptionPane.showInputDialog(
            "Enter loan amount, for example 120000.95:");
        double loanAmount = Double.parseDouble(loanString);
        double monthlyPayment = loanAmount * monthlyInterestRate / (1
            - 1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12));
        double totalPayment = monthlyPayment * numberOfYears * 12;
        monthlyPayment = (int)(monthlyPayment * 100) / 100.0;
        totalPayment = (int)(totalPayment * 100) / 100.0;
        String output = "The monthly payment is " + monthlyPayment +
            "\nThe total payment is " + totalPayment;
        JOptionPane.showMessageDialog(null, output);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



ตัวอย่างที่ 2.16 ตัวอย่างโปรแกรมที่รับค่าจาก JOptionPane เพื่อทำนายวันเกิด

```
import javax.swing.JOptionPane;
public class Ex2_16 {
    public static void main(String[] args) {
        String set1 =
            " 1  3  5  7\n" +
            " 9 11 13 15\n" +
            "17 19 21 23\n" +
            "25 27 29 31";
        String set2 =
            " 2  3  6  7\n" +
            "10 11 14 15\n" +
            "18 19 22 23\n" +
            "26 27 30 31";
        String set3 =
            " 4  5  6  7\n" +
            "12 13 14 15\n" +
            "20 21 22 23\n" +
            "28 29 30 31";

        String set4 =
            " 8  9 10 11\n" +
            "12 13 14 15\n" +
            "24 25 26 27\n" +
            "28 29 30 31";

        String set5 =
            "16 17 18 19\n" +
            "20 21 22 23\n" +
            "24 25 26 27\n" +
            "28 29 30 31";

        int day = 0;

        // Prompt the user to answer questions
        int answer = JOptionPane.showConfirmDialog(null,
            "Is your birthday in these numbers?\n" + set1);

        if (answer == JOptionPane.YES_OPTION)
            day += 1;

        answer = JOptionPane.showConfirmDialog(null,
            "Is your birthday in these numbers?\n" + set2);

        if (answer == JOptionPane.YES_OPTION)
            day += 2;

        answer = JOptionPane.showConfirmDialog(null,
            "Is your birthday in these numbers?\n" + set3);

        if (answer == JOptionPane.YES_OPTION)
            day += 4;

        answer = JOptionPane.showConfirmDialog(null,
            "Is your birthday in these numbers?\n" + set4);
```

```

if (answer == JOptionPane.YES_OPTION)
    day += 8;

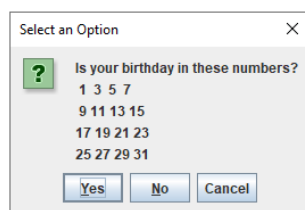
answer = JOptionPane.showConfirmDialog(null,
    "Is your birthday in these numbers?\n" + set5);

if (answer == JOptionPane.YES_OPTION)
    day += 16;

JOptionPane.showMessageDialog(null, "Your birthday is " +
    day + "!");
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



2.11 บทสรุป

ในการเขียนโปรแกรมที่มีประสิทธิภาพจำเป็นต้องมีการกำหนดรูปแบบของตัวแปรและมีการใช้ตัวดำเนินการทางคณิตศาสตร์ที่เหมาะสมเพื่อให้สามารถทำให้การดำเนินการของโปรแกรมให้ผลที่ถูกต้อง โดยทั่วไปชนิดของข้อมูลจะแบ่งออกเป็น 2 รูปแบบคือ ชนิดข้อมูลแบบนามธรรมและชนิดข้อมูลแบบทั่วไป

ข้อมูลในระดับความคิด (Abstract data types) เป็นระดับของแบบชนิดข้อมูลประเภทนามธรรมที่สร้างขึ้นจากจินตนาการของผู้ใช้ เป็นแบบชนิดข้อมูลที่ไม่รูปร่างหรือลักษณะให้เห็น การอธิบายลักษณะของข้อมูลจะใช้สัญลักษณ์ ถ้าต้องการทำให้แบบชนิดข้อมูลที่เป็นนามธรรมเป็นข้อมูลที่เป็นรูปธรรมต้องนำไปใช้จริงกับเครื่องคอมพิวเตอร์จริง (Implement) ข้อมูลในระดับความคิดแบ่งออกเป็น 2 ประเภท คือ โครงสร้างข้อมูลแบบเชิงเส้น (Linear data structures) เช่น ลิสต์ (list) สแตก (stack) คิว (queue) ดีคิว (deque) โครงสร้างข้อมูลแบบไม่ใช่เชิงเส้น (Non-linear data structures) เช่น ตรี (tree) กราฟ (graph)

ชนิดข้อมูลแบบทั่วไป 1. ข้อมูลเบื้องต้น (Primitive data types) เป็นข้อมูลพื้นฐานซึ่งมีโครงสร้างข้อมูลไม่ซับซ้อนจะต้องมีในภาษาคอมพิวเตอร์ทุกภาษา ตัวอย่างของข้อมูลประเภทนี้ เช่น จำนวนเต็ม (integer) จำนวนจริง (real) ตัวอักษร (character) 2. ข้อมูลโครงสร้าง (Structured data types) เป็นข้อมูลที่มีโครงสร้างสลับซับซ้อนเกิดจากการนำโครงสร้างข้อมูลเบื้องต้นมาประกอบกันเป็นโครงสร้างข้อมูลที่หลากหลายขึ้น ข้อมูลที่ใช้ในเครื่องคอมพิวเตอร์ยุคแรกเป็นข้อมูลเบื้องต้นเท่านั้น แต่ในปัจจุบันภาษาคอมพิวเตอร์เกือบทุกภาษามีข้อมูลโครงสร้างด้วยแทบทั้งสิ้น ตัวอย่างข้อมูลโครงสร้าง เช่น แถวลำดับ (array) เซต (set) ระเบียบข้อมูล (record) แฟ้มข้อมูล (file)

2.12 แบบฝึกหัดปฏิบัติการ

1. กำหนดให้ int a=1 และ double d=1.0 จงหาผลลัพธ์จาก expressions ต่อไปนี้

Expressions	ผลลัพธ์
a=46/9;	
a=46%9+4*4-2;	
a=45+43%5*(23*3%2);	
a%=3/a + 3;	
d=4+d*d+4;	
d+=1.5*3(++a);	
d-=1.5*3+a++;	

2. จงหาส่วนที่ผิดพลาดของโปรแกรมต่อไปนี้และแก้ไขให้ถูกต้อง

```
public class Test {
    public void main(string[] args){
        int I;
        int k=100.0;
        int j=i+1;
        System.out.println("j is"+ j+"and k is" +k);
    }
}
```

3. จงหาผลลัพธ์จากส่วนของโปรแกรมต่อไปนี้

โปรแกรม	ผลลัพธ์
int i=0; System.out.println(--i + i +i ++); System.out.println(i + ++i);	
int i=0; i=i+(i=1); System.out.println(i);	
int i=0; i=(i=1) + i System.out.println(i);	

บทที่ 3 คำสั่งควบคุม

วัตถุประสงค์

- 3.1 มีความรู้ความเข้าใจและสามารถพัฒนาชุดคำสั่งโดยการใช้คำสั่งทดสอบเงื่อนไข การตัดสินใจ
- 3.2 มีความรู้ความเข้าใจและสามารถพัฒนาชุดคำสั่งโดยการใช้คำสั่งการวนซ้ำ

3.1 ความนำ

โดยปกติการทำงานของคอมพิวเตอร์จะทำงานเรียงลำดับคำสั่งลงมาตั้งแต่ต้นโปรแกรมจนจบโปรแกรม แต่ถ้าต้องการเปลี่ยนแปลงขั้นตอนการทำงานของคำสั่ง เช่น กระโดดข้ามไปที่ คำสั่งใดคำสั่งหนึ่ง หรือให้วนกลับมาทำคำสั่งที่เคยทำไปแล้ว ลักษณะการสั่งงานแบบนี้จะต้องใช้คำสั่งควบคุม ดังนั้นคำสั่งควบคุมจึงเป็นคำสั่งที่ใช้เปลี่ยนแปลงลำดับขั้นตอนการทำงานของโปรแกรม ซึ่งแบ่งออกเป็น 3 ชนิด คือ

1. คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข (Unconditional branch Statement) ได้แก่ คำสั่ง goto ซึ่งไม่มีการใช้งานในภาษาจาวา
2. คำสั่งให้ไปทำงานโดยมีเงื่อนไข (Condition Statement) ได้แก่ คำสั่ง if, switch
3. คำสั่งให้ไปทำงานแบบเป็นวงจร (Loop Control Statement) ได้แก่ คำสั่ง while, do while, for

3.2 คำสั่งทดสอบเงื่อนไข การตัดสินใจ

3.2.1 คำสั่งทดสอบเงื่อนไข if Statements

กรณีที่ต้องการให้โปรแกรมมีการทำหรือไม่ทำคำสั่งที่ผู้ใช้กำหนดจะทำได้โดยการใช้คำสั่ง if โดยมีรูปแบบดังนี้

```
if (booleanExpression) {
    statement(s);
}
```

เช่น เมื่อโปรแกรมรับค่าจากผู้ใช้และต้องการทดสอบว่าค่า i มีค่ามากกว่า 0 และน้อยกว่า 10 ให้พิมพ์ข้อความออกมา

ตัวอย่างที่ 3.1 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_1{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        int i = input.nextInt();
        if ((i > 0) && (i < 10)) {
            System.out.println("i is an " + "integer between 0 and 10");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
9
i is an integer between 0 and 10
```

ตัวอย่างการใช้คำสั่ง if มีดังนี้

ตัวอย่างที่ 3.2 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_2{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        int radius;
        double area;
        final double PI=3.14;
        radius= input.nextInt();
        if (radius >= 0)
        { area = radius*radius*PI;
          System.out.println( "The area for the circle of radius " + radius + " is " +
area);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
10
The area for the circle of radius 10 is 314.0
```

ข้อควรระวัง หลังคำสั่ง if จะไม่ใส่ semicolon เช่นตัวอย่างที่ 3.2 หากทำคำสั่ง if (radius >= 0); เมื่อทำคำสั่งข้างต้นจะไม่เกิด compile error หรือ runtime error แต่สิ่งที่เกิดคือ logic error คือการทำงานไม่ถูกต้องตามสิ่งที่ควรจะเป็น

3.2.2 คำสั่งทดสอบเงื่อนไข if...else Statement

กรณีที่ต้องการให้โปรแกรมมีการทำหรือไม่ทำคำสั่งที่ผู้ใช้กำหนดตามเงื่อนไข เช่นกรณีที่เงื่อนไขเป็นจริงให้ทำชุดคำสั่งที่ 1 กรณีเงื่อนไขที่ทดสอบเป็นเท็จจะทำชุดคำสั่งที่ 2 จะทำได้โดยการใช้คำสั่ง if...else โดยมีรูปแบบดังนี้

```
if (booleanExpression) {
    คำสั่งกรณีที่เงื่อนไขถูกต้อง;
}
else {
    คำสั่งกรณีที่เงื่อนไขไม่ถูกต้อง;
}
```

ตัวอย่างที่ 3.3 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int radius;
        double area;
        final double PI=3.14;
        radius= input.nextInt();
        if (radius >= 0) {
            area = radius*radius*PI;
            System.out.println("The area for the " + "circle of radius " + radius + " is " +
area);
        }else {
            System.out.println("Negative input");
        }
    }
}
```


ผลลัพธ์ที่เกิดขึ้นคือ

```
15
The area for the circle of radius 15 is 706.5
```

ตัวอย่างที่ 3.3 แสดง ตัวอย่างการรับข้อมูลรัศมีของวงกลมและนำมาเก็บในตัวแปร radius จากนั้นจะทำการเช็คค่าข้อมูลรัศมีดังกล่าวมีค่ามากกว่าหรือเท่ากับศูนย์หรือไม่ หากตัวเลขดังกล่าวมากกว่าหรือเท่ากับศูนย์จะนำไปคำนวณพื้นที่ของวงกลมตามสมการ $area = radius * radius * \pi$; ในกรณีที่รัศมีของวงกลมมีค่าน้อยกว่าศูนย์จะแสดงข้อความ Negative input

ตัวอย่างที่ 3.4 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_4 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int score;
        char grade;
        System.out.println("Input score:");
        score= input.nextInt();
        if (score >= 90)
            grade = 'A';
        else
            if (score >= 80)
                grade = 'B';
            else
                if (score >= 70)
                    grade = 'C';
                else
                    if (score >= 60)
                        grade = 'D';
                    else
                        grade = 'F';
        System.out.println("Grade =" + grade);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Input score:
85
Grade =B
```

ตัวอย่างที่ 3.4 แสดงตัวอย่างการรับข้อมูลคะแนนสอบและนำมาเก็บในตัวแปร score จากนั้นจะทำการเช็คค่าคะแนนสอบดังกล่าว หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 90 จะนำตัวอักษร 'A' ไปเก็บในตัวแปร grade หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 80 จะนำตัวอักษร 'B' ไปเก็บในตัวแปร grade หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 70 จะนำตัวอักษร 'C' ไปเก็บในตัวแปร grade หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 60 จะนำตัวอักษร 'D' ไปเก็บในตัวแปร grade หากไม่ตรงกับกรณีใดๆ ข้างต้น จะนำตัวอักษร 'F' ไปเก็บในตัวแปร grade

ตัวอย่างที่ 3.5 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_5 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int score;
        char grade;
        System.out.println("Input score:");
        score= input.nextInt();
        if (score >= 90)
```

```

        grade = 'A';
    else if (score >= 80)
        grade = 'B';
    else if (score >= 70)
        grade = 'C';
    else if (score >= 60)
        grade = 'D';
    else
        grade = 'F';
    System.out.println("Grade =" + grade);
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

Input score:
85
Grade =B

```

ตัวอย่างที่ 3.5 แสดงตัวอย่างการเขียนอีกรูปแบบของการเขียนคำสั่ง if...else เหมือนตัวอย่างที่ 3.4 โดยการรับข้อมูลคะแนนสอบและนำมาเก็บในตัวแปร score จากนั้นจะทำการเช็คค่าคะแนนสอบดังกล่าว หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 90 จะนำตัวอักษร 'A' ไปเก็บในตัวแปร grade หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 80 จะนำตัวอักษร 'B' ไปเก็บในตัวแปร grade หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 70 จะนำตัวอักษร 'C' ไปเก็บในตัวแปร grade หากคะแนนสอบดังกล่าวมากกว่าหรือเท่ากับ 60 จะนำตัวอักษร 'D' ไปเก็บในตัวแปร grade หากไม่ตรงกับกรณีใดๆ ข้างต้น จะนำตัวอักษร 'F' ไปเก็บในตัวแปร grade

คำสั่ง else จะจับคู่กับ if ที่ใกล้ตัวมันมากที่สุดก่อน เช่น

ตัวอย่างที่ 3.6 คำสั่งทดสอบเงื่อนไข if Statements

```

import java.util.Scanner;
public class Ex3_6 {
    public static void main(String[] args) {
        int i = 1; int j = 2; int k = 3;
        if (i > j)
            if (i > k)
                System.out.println("A");
        else
            System.out.println("B");
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

ไม่มีผลลัพธ์

จากตัวอย่างที่ 3.6 เมื่อโปรแกรมประมวลผลจะไม่มีผลลัพธ์ออกมาเนื่องจากคำสั่ง else จะจับคู่กับ if ที่ใกล้ตัวมันมากที่สุดก่อนโดยจะมีผลการทำงานเหมือนกับตัวอย่างที่ 3.7

ตัวอย่างที่ 3.7 คำสั่งทดสอบเงื่อนไข if Statements

```

import java.util.Scanner;
public class Ex3_7 {
    public static void main(String[] args) {
        int i = 1; int j = 2; int k = 3;
        if (i > j)
            if (i > k)
                System.out.println("A");
        else
            System.out.println("B");
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

ไม่มีผลลัพธ์

ตัวอย่างที่ 3.7 เมื่อโปรแกรมประมวลผลจะไม่มีผลลัพธ์ออกมาเนื่องจากคำสั่ง else จะจับคู่กับ if ที่ใกล้ตัวมันมากที่สุดก่อน

เมื่อต้องการให้โปรแกรมทำคำสั่ง if-else ตามที่เราต้องการให้ใส่ block ของปีกกาเปิดปิดเข้าไปเพื่อจัดลำดับการทำงาน เช่น

ตัวอย่างที่ 3.8 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_8 {
    public static void main(String[] args) {
        int i = 1;
        int j = 2;
        int k = 3;
        if (i > j) {
            if (i > k)
                System.out.println("A");
        }
        else
            System.out.println("B");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

B

ตัวอย่างที่ 3.8 แสดงตัวอย่างเมื่อต้องการให้โปรแกรมทำคำสั่ง if-else ตามที่เราต้องการให้ใส่ block ของปีกกาเปิดปิดเข้าไปเพื่อจัดลำดับการทำงาน

การทดสอบเงื่อนไขสามารถใช้เครื่องหมายทดสอบเงื่อนไขแบบย่อสำหรับทดสอบเงื่อนไขได้โดยใช้ conditional expression เช่น

(booleanExp) ? exp1 : exp2

ตัวอย่างที่ 3.9 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_9 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int num;
        num= input.nextInt();
        if (num % 2 == 0)
            System.out.println(num + " is even");
        else
            System.out.println(num + " is odd");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

10

10 is even

ตัวอย่างที่ 3.9 แสดงตัวอย่างการรับตัวเลขจำนวนเต็มเก็บลงในตัวแปร num และทำการตรวจสอบตัวเลขดังกล่าวว่าเป็นเลขคู่หรือเลขคี่โดยใช้ if เพื่อทดสอบเงื่อนไข โดยเราสามารถทดสอบเงื่อนไขโดยใช้เครื่องหมายทดสอบเงื่อนไขแบบย่อสำหรับทดสอบเงื่อนไขได้ดังตัวอย่างที่ 3.10

ตัวอย่างที่ 3.10 คำสั่งทดสอบเงื่อนไข if Statements

```
import java.util.Scanner;
public class Ex3_10 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int num;
        num= input.nextInt();
        System.out.println( (num % 2 == 0)? num + " is even" : num + " is odd");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
7
7 is odd
```

ตัวอย่างที่ 3.10 แสดงตัวอย่างการรับตัวเลขจำนวนเต็มเก็บลงในตัวแปร num และทำการตรวจสอบตัวเลขดังกล่าวว่าเป็นเลขคู่หรือเลขคี่โดยใช้เครื่องหมายทดสอบเงื่อนไขแบบย่อสำหรับทดสอบเงื่อนไข

3.3 คำสั่งทดสอบเงื่อนไข switch statements

เมื่อมีตัวเลือกให้เลือกมากกว่า 1 ตัวเลือกอาจใช้คำสั่ง switch-case สำหรับทดสอบเงื่อนไขโดยมีรูปแบบดังนี้

```
switch (switch-expression) {
case value1:
    . . . // x is valueOne
break;
case value2:
    . . . // x is valueTwo
break;
default:
    . . . // other value of x
}
```

เมื่อ switch-expression คือ เงื่อนไขที่จะทดสอบค่า มีชนิดข้อมูลเป็นแบบ byte, short, int, long และ char เท่านั้น

ค่า value1 ถึง valueN จะต้องเป็นชนิดข้อมูลเดียวกันกับตัวแปรที่อยู่ใน switch-expression

คำสั่งที่อยู่ใน Block ของค่าที่ตรวจสอบในแต่ละกรณี หากตรงจะทำงานในบล็อกของคำสั่งแต่ละ case เมื่อสิ้นสุดการทำงานจะใช้คำสั่ง break เพื่อบอกจุดสิ้นสุดการทำงานในแต่ละ case โดยไม่ต้องตรวจสอบเงื่อนไขอื่น ๆ ที่เหลือ กรณีที่ไม่ใส่ break จะทำให้ case ต่อไป switch case ถูกประมวลผลโดยอัตโนมัติ

ข้อควรระวัง ควรจะใช้ break เมื่อสิ้นสุดการทำงานในแต่ละ case เพื่อให้ผลการทำงานถูกต้อง เช่น ตัวอย่างโปรแกรมต่อไปนี้ หากไม่มี switch case หาก numberofyear เป็น 15 ผลลัพธ์ที่ได้จะกำหนด เป็น 8.50 และ 9.0 และพิมพ์ wrong ในลำดับต่อไป เช่น

ตัวอย่างที่ 3.11 คำสั่งทดสอบเงื่อนไข switch statements

```
import java.util.Scanner;
public class Ex3_11 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int year;
        double annualInterestRate=0;
        numberOfyear = input.nextInt();
        switch (numberOfyear) {
            case 7:
                annualInterestRate = 7.25;
                break;
            case 15:
                annualInterestRate = 8.50;
                break;
            case 30:
                annualInterestRate = 9.0;
                break;
            default: System.out.println("Wrong number of years, enter 7, 15, or 30");
        }
        System.out.println("Number of year is " + numberOfyear + "\nAnnual Interest Rate = " +
            annualInterestRate);
    }
}
```

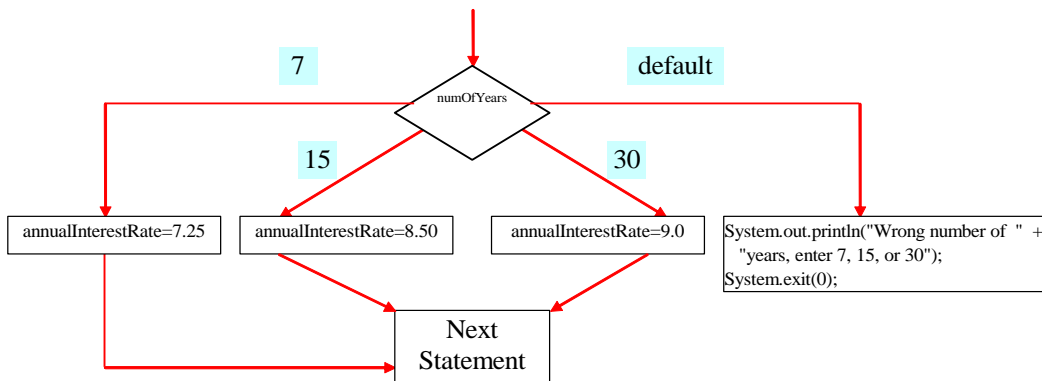
ผลลัพธ์ที่เกิดขึ้นคือ

15

Number of year is 15

Annual Interest Rate = 8.5

ตัวอย่างโฟลว์ชาร์ตของชุดคำสั่ง switch



3.4 คำสั่งการวนซ้ำ

การวนซ้ำ หมายถึง การควบคุมให้การกระทำบางคำสั่งซ้ำหลายรอบ ซึ่งจะช่วยให้การเขียนโปรแกรมทำได้ง่ายขึ้น ไม่ต้องเขียนคำสั่งเดิมหลายครั้ง ทำให้โปรแกรมมีความกระชับ สามารถตรวจสอบความผิดพลาดได้ง่าย คำสั่งการวนรอบมี 3 ชนิดหลัก ๆ คือ คำสั่ง for คำสั่ง do-while และคำสั่ง while

โดยที่แต่ละคำสั่งมีรูปแบบและวิธีการใช้งานที่แตกต่างกัน สามารถเลือกใช้ตามความเหมาะสมของการใช้งานในโปรแกรม

ส่วนประกอบของคำสั่งแบบวนซ้ำ ประกอบด้วย

ส่วนของการตรวจสอบ (loop test) เป็นเงื่อนไขเพื่อทดสอบว่าจะทำวนซ้ำอีกหรือไม่

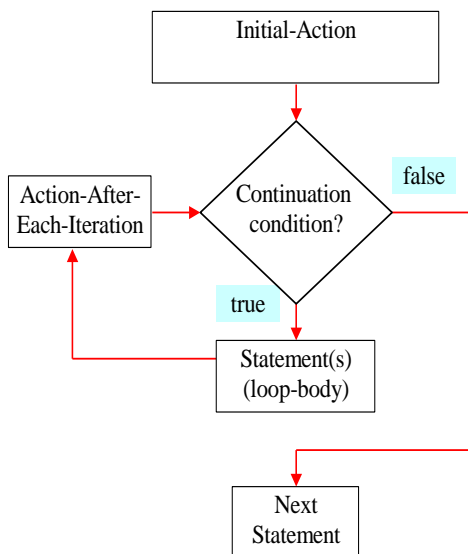
ส่วนของการทำวนซ้ำ (loop body) เป็นชุดคำสั่งที่จะถูกดำเนินการ

รายละเอียดของการสร้าง loop มีขั้นตอนการทำงานดังนี้

ระบุส่วนของการทำงานที่ต้องทำซ้ำ (loop body) ในส่วนนี้จะเป็นการระบุเงื่อนไข (loop test) ที่จะทำซ้ำ หรือเลิกทำซ้ำ

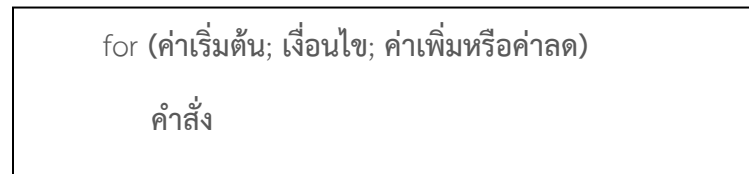
ระบุชนิดของ loop ที่จะใช้ ซึ่งชนิดของการทำซ้ำจะมีประเภทที่ต่างกันหลัก ๆ 2 ประเภทคือ Pre-test loop และ Post-test loop

3.4.1 คำสั่ง for



คำสั่ง for เป็นคำสั่งที่สั่งให้ทำคำสั่ง หรือกลุ่มของคำสั่งวนซ้ำหลายรอบ โดยมีจำนวนรอบในการวนซ้ำที่แน่นอน

รูปแบบ



โดยที่ ค่าเริ่มต้น, เงื่อนไข, ค่าเพิ่มหรือค่าลด เป็นนิพจน์

คำสั่ง หมายถึง คำสั่งที่จะถูกกระทำซ้ำ ซึ่งอาจจะมีเพียงคำสั่งเดียว หรือหลายคำสั่งก็ได้

คำสั่ง for มีขั้นตอนการทำงานดังนี้

1. คำนวณค่าเริ่มต้นของตัวแปรที่ใช้ควบคุมการวนซ้ำซึ่งอยู่ในรูปของคำสั่งกำหนดค่า
2. คำนวณผลลัพธ์จากเงื่อนไข ที่อยู่ในรูปของนิพจน์ความสัมพันธ์ ซึ่งจะให้ผลเป็นเท็จ (false) หรือจริง (true)
3. ถ้าผลลัพธ์จากข้อ 2 มีค่าเป็นเท็จหรือ false ไปที่ 7
4. ถ้าผลลัพธ์จากข้อ 2 มีค่าเป็นจริง หรือ true คำสั่งที่อยู่ภายในคำสั่ง for จะถูกกระทำ
5. คำนวณค่าใหม่ของตัวแปรที่ใช้ควบคุมการวนซ้ำ

6. กลับไปที่ข้อ 2

7. จบการกระทำการคำสั่ง for และข้อความแรกที่อยู่ถัดจากคำสั่ง for จะถูกทำในลำดับต่อไป

ตัวอย่างที่ 3.12 คำสั่ง for

```
import java.util.Scanner;
public class Ex3_12 {
    public static void main(String[] args) {
        int i;
        for (i = 0; i < 5; i++) {
            System.out.println("Welcome to Java! " + i);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Welcome to Java! 0
Welcome to Java! 1
Welcome to Java! 2
Welcome to Java! 3
Welcome to Java! 4
```

ตัวอย่างที่ 3.12 แสดงตัวอย่างการประมวลผลคำสั่ง for เพื่อวนแสดงข้อความ Welcome to Java! ต่อด้วยตัวเลข แสดงหมายเลขรอบที่โดยเริ่มที่รอบที่ 0 ถึง 4

ตัวอย่างที่ 3.13 คำสั่ง for

```
import java.util.Scanner;
public class Ex3_13 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int i=0;
        int sum=0;
        int n, temp;
        System.out.print("Input n: ");
        n= input.nextInt();
        for (i = 0; i < n; i++) {
            System.out.print("No." + (i+1)+" :");
            temp =input.nextInt();
            sum+=temp;
        }
        System.out.println("Sum =" +sum);
        System.out.println("Average =" +((float) sum/n));
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Input n: 5
No.1 :1
No.2 :5
No.3 :9
No.4 :3
No.5 :7
Sum =25
Average =5.0
```

ตัวอย่างที่ 3.13 แสดงตัวอย่างการประมวลผลคำสั่ง for โดยการรับจำนวนรอบเก็บในตัวแปร n และใช้ตัวแปร i สำหรับการนับจำนวนรอบโดยเริ่มที่ i =0 และ i < n ทำการนับจำนวนรอบทีละ 1 โดยแต่ละรอบจะรับตัวเลขจำนวนเต็ม และหาผลรวม เมื่อครบตามจำนวนรอบที่กำหนดเรียบร้อยแล้ว จะทำการหาค่าเฉลี่ยและแสดงผลลัพธ์ออกทางหน้าจอ

ตัวอย่างที่ 3.14 คำสั่ง for

```
import java.util.Scanner;
public class Ex3_14 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int i=0, j=0;
        int sum=0;
        int n, temp;
        System.out.print("Input n: ");
        n= input.nextInt();
        for (i = 1; i <= n; i++) {
            for(j = 1; j <= n; j++){
                System.out.print(String.format("%3d", (i*j)));
            }
            System.out.println("");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Input n: 5
 1  2  3  4  5
 2  4  6  8 10
 3  6  9 12 15
 4  8 12 16 20
 5 10 15 20 25
```

ตัวอย่างที่ 3.14 แสดงตัวอย่างการประมวลผลคำสั่ง for โดยการรับจำนวนรอบเก็บในตัวแปร n และใช้ตัวแปร i สำหรับการนับจำนวนรอบโดยเริ่มที่ i = 0 และ i <= n ทำการเพิ่มจำนวนรอบทีละ 1 โดยแต่ละรอบจะมีการวนซ้ำภายใน โดยใช้ตัวแปร j สำหรับการนับจำนวนรอบโดยเริ่มที่ j = 0 และ j <= n ทำการเพิ่มจำนวนรอบทีละ 1 โดยแต่ละรอบจะทำการคำนวณโดยนำ i*j ซึ่งผลลัพธ์ที่ได้จะแสดงตารางสูตรคูณดังแสดงในตัวอย่าง

ตัวอย่างที่ 3.15 คำสั่ง for

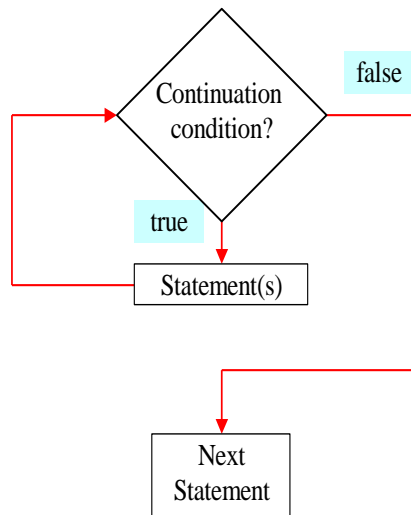
```
import java.util.Scanner;
public class Ex3_15 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int i=0, j=0;
        int sum=0;
        int n, temp;
        System.out.print("Input n: ");
        n= input.nextInt();
        for (i = 1; i <= n; i++) {
            for(j = 1; j <= i; j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Input n: 5
*
**
***
****
*****
```

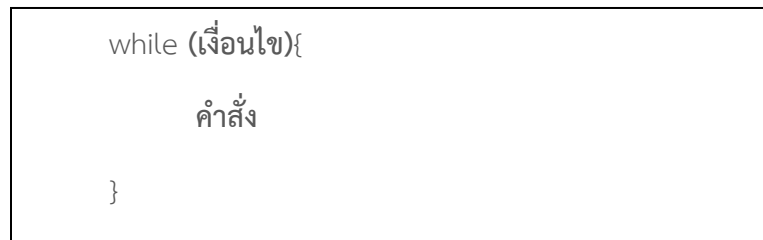

ตัวอย่างที่ 3.15 แสดงตัวอย่างการประมวลผลคำสั่ง for โดยการรับจำนวนรอบเก็บในตัวแปร n และใช้ตัวแปร i สำหรับการนับจำนวนรอบโดยเริ่มที่ $i = 0$ และ $i \leq n$ ทำการเพิ่มจำนวนรอบทีละ 1 โดยแต่ละรอบจะมีการวนซ้ำภายใน โดยใช้ตัวแปร j สำหรับการนับจำนวนรอบโดยเริ่มที่ $j = 0$ และ $j \leq n$ ทำการเพิ่มจำนวนรอบทีละ 1 โดยแต่ละรอบจะทำการพิมพ์เครื่องหมาย * ในแต่ละแถว ซึ่งผลลัพธ์ที่ได้จะแสดงผลดังตัวอย่าง

3.4.2 คำสั่ง while



คำสั่ง while เป็นคำสั่งวนซ้ำ ที่สั่งให้ทำคำสั่งที่อยู่ภายในคำสั่ง while หลายรอบจนกระทั่งเงื่อนไขเป็นเท็จ จึงจะจบการวนซ้ำ

รูปแบบ

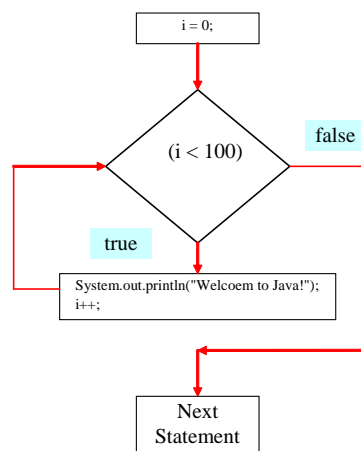


โดยที่ เงื่อนไข เป็นนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ และคำสั่งอยู่ภายในคำสั่ง while อาจมีเพียงคำสั่งเดียว หรือหลายคำสั่ง

คำสั่ง while มีขั้นตอนการทำงานดังนี้

1. คำนวณค่าของเงื่อนไข
2. ถ้าค่าของเงื่อนไข มีค่าเป็นเท็จ ไปที่ข้อ 5
3. ถ้าค่าของเงื่อนไข มีค่าเป็นจริง คำสั่งที่อยู่ภายในคำสั่ง while จะถูกกระทำ
4. กลับไปที่ข้อ 1
5. จบการกระทำคำสั่ง while และข้อความแรกที่อยู่ถัดจากคำสั่ง while จะถูกทำในลำดับต่อไป

ตัวอย่างที่ 3.16 คำสั่ง while



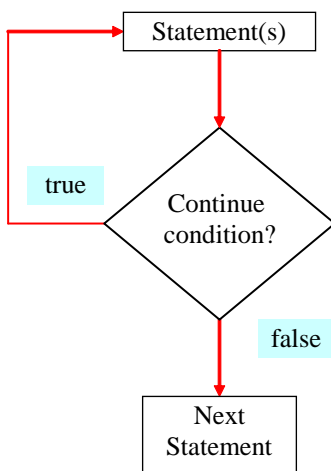
```
import java.util.Scanner;
public class Ex3_16 {
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println("Welcome to Java!");
            i++;
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Welcome to Java!
Welcome to Java!
Welcome to Java!
Welcome to Java!
Welcome to Java!
```

ตัวอย่างที่ 3.16 แสดงตัวอย่างการประมวลผลคำสั่ง while เพื่อวนแสดงข้อความ Welcome to Java! โดยการทำงานเริ่มต้นจะใช้ตัวแปร i เป็นตัวนับจำนวนรอบโดยก่อนการแสดงข้อความจะทำการเช็คค่า i<5 หรือไม่หากใช่จะแสดงข้อความ Welcome to Java! แต่หากเงื่อนไขดังกล่าวเป็นเท็จจะจบการวนลูป

3.4.3 คำสั่ง do-while



คำสั่ง do-while เป็นคำสั่งวนซ้ำ ที่สั่งให้ทำคำสั่งที่อยู่ภายในคำสั่ง do-while หนึ่งรอบ แล้วจึงจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นเท็จจะจบการทำงานทันที

รูปแบบ

```
do{
    คำสั่ง;
}while (เงื่อนไข);
```

โดยที่ เงื่อนไข เป็นนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ และคำสั่งอยู่ภายในคำสั่ง do-while อาจมีเพียงคำสั่งเดียว หรือหลายคำสั่ง

คำสั่ง do-while มีขั้นตอนการทำงานดังนี้

1. กระทำคำสั่งที่อยู่ภายในคำสั่ง do-while
2. คำนวณหาค่าของเงื่อนไข
3. ถ้าค่าของเงื่อนไข มีค่าเป็นเท็จหรือ false ไปที่ข้อ 5
4. ถ้าค่าของเงื่อนไข มีค่าเป็นจริงหรือ true กลับไปที่ข้อ 1
5. จบการกระทำคำสั่ง do-while และข้อความแรกที่อยู่ถัดจากคำสั่ง do-while จะถูกทำในลำดับต่อไป

ตัวอย่างที่ 3.17 คำสั่ง do-while

```
import java.util.Scanner;
public class Ex3_17 {
    public static void main(String[] args) {
        int i=0;
        do{
            System.out.println("i is " + i);
            i++;
        } while (i<5);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
i is 0
i is 1
i is 2
i is 3
i is 4
```

ตัวอย่างที่ 3.17 แสดงตัวอย่างการประมวลผลคำสั่ง do...while เพื่อวนแสดงข้อความ i is ตามด้วยหมายเลขลำดับรอบโดยเริ่มที่ 0 ถึง 4 โดยการทำงานเริ่มต้นจะใช้ตัวแปร i เป็นตัวนับจำนวนรอบโดยรอบแรกจะแสดงข้อความ i is ตามด้วยหมายเลขลำดับรอบโดยเริ่มที่ 0 เมื่อทำการประมวลผลรอบแรกเสร็จก่อนการแสดงข้อความรอบต่อไปจะทำการเช็คค่า i<5 หรือไม่หากใช่จะแสดงข้อความ i is ตามด้วยหมายเลขลำดับรอบ แต่ถ้าหากเงื่อนไขดังกล่าวเป็นเท็จจะจบการวนลูป

เราสามารถใช้รูปแบบของการวนซ้ำในรูปแบบใดก็ได้ คือ while, do, และ for โดยผลจากการทำงานจะมีลักษณะคล้ายกัน ให้เลือกใช้ตามที่ผู้ใช้นัดมากที่สุด โดยทั่วไป

for จะใช้ในกรณีที่ทราบจำนวนของรอบที่ต้องการประมวลผลอย่างชัดเจน

while จะใช้ในกรณีที่ไมทราบจำนวนของรอบที่ต้องการประมวลผลอย่างชัดเจน

do-while จะใช้ในกรณีที่ไมทราบจำนวนของรอบที่ต้องการประมวลผลอย่างชัดเจน และจำเป็นต้องมีการทำงานก่อนการเช็คเงื่อนไข

ข้อควรระวัง การเติม Semicolon ที่ส่วนท้ายของ for จะทำให้เกิดความผิดพลาดเกิดขึ้น ดังแสดงด้านล่าง

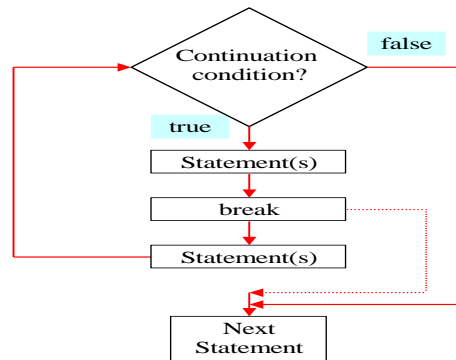
```
for (int i=0; i<10; i++);
{
    System.out.println("i is " + i);
}
```

และในทำนองเดียวกันการเติม Semicolon ที่ส่วนท้ายของ while จะทำให้เกิดความผิดพลาดเกิดขึ้น

```
int i=0;
while (i<10);
{
    System.out.println("i is " + i);
    i++;
}
```

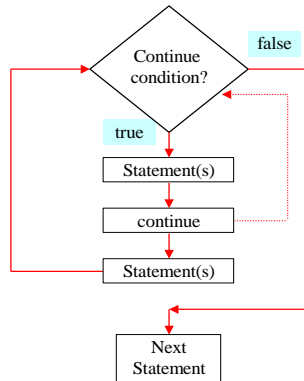
3.4.4 คำสั่ง break

เป็นคำสั่งที่ทำให้หยุดสิ้นสุดการทำงานของโครงสร้างแบบทำซ้ำ หรือเป็นคำสั่งให้สิ้นสุดการทำงานใน loop



3.4.5 คำสั่ง continue

เป็นคำสั่งที่จะข้ามการทำงานคำสั่งที่เหลือภายในบล็อก { } โดยไปเริ่มการทำซ้ำในรอบต่อไป



3.5 ตัวอย่างการประยุกต์คำสั่งการวนซ้ำ

ตัวอย่างที่ 3.18 พนักงานขายคนหนึ่งในห้างสรรพสินค้าได้เริ่มงานและได้รับเงินเดือนพื้นฐานและค่าคอมมิชชันจากการขายสินค้าโดยเงินเดือนอยู่ที่อัตรา 150000 บาท นอกจากนี้ทางห้างสรรพสินค้าได้กำหนดอัตราการจ่ายค่าคอมมิชชันให้พนักงานขายดังตารางด้านล่าง

ยอดขาย	อัตราการจ่ายค่าคอมมิชชัน
1-150,000	8 %
150,001-300,000	10 %
มากกว่าหรือเท่ากับ 300,001	12 %

พนักงานขายคนดังกล่าวตั้งเป้าหมายที่จะได้รับเงินไม่น้อยกว่า 900,000 บาท จงเขียนโปรแกรมเพื่อคำนวณยอดขายต่ำสุดที่ต้องทำเพื่อจะได้รับเงินไม่น้อยกว่า 900,000 บาท

```
import java.util.Scanner;
public class Ex3_18 {
    public static void main(String[] args) {
        final double COMMISSION_SOUGHT = 885000;
        final double INITIAL_SALES_AMOUNT = 1;
        double commission = 0;
        double salesAmount = INITIAL_SALES_AMOUNT;
        do {
            salesAmount += 1;
            if (salesAmount >= 300001)
                commission = 150000 * 0.08 + 150000 * 0.1 + (salesAmount - 300000) * 0.12;
            else if (salesAmount >= 150001)
                commission = 150000 * 0.08 + (salesAmount - 150000) * 0.10;
            else
                commission = salesAmount * 0.08;
        } while (commission < COMMISSION_SOUGHT);
        System.out.println("The sales amount " + (int)(salesAmount * 100) / 100.0 +
            "\nis needed to make a commission of " + COMMISSION_SOUGHT);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

The sales amount 7450000.0
is needed to make a commission of 885000.0

ตัวอย่างที่ 3.19 การแสดงตัวเลขแบบ Pyramid

```

      1
     2 1 2
    3 2 1 2 3
   4 3 2 1 2 3 4
  5 4 3 2 1 2 3 4 5
```

```
import java.util.Scanner;
public class Ex3_19{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number of lines: ");
        int n = input.nextInt();
        if (n < 1 || n > 15) {
            System.out.println("You must enter a number from 1 to 15");
            System.exit(0);
        }
        for (int row = 1; row <= n; row++) {
            for (int column = 1; column <= n - row; column++)
                System.out.print(" ");
            for (int num = row; num >= 1; num--)
                System.out.print((num >= 10) ? " " + num : " " + num);
            for (int num = 2; num <= row; num++)
                System.out.print((num >= 10) ? " " + num : " " + num);
            System.out.println();
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

Enter the number of lines: 5

```

      1
     2 1 2
    3 2 1 2 3
   4 3 2 1 2 3 4
  5 4 3 2 1 2 3 4 5
```

ตัวอย่างที่ 3.20 แสดงตัวเลขจำนวนเฉพาะ(Prime Numbers) 50 ตัวแรกโดยแสดงตัวเลขจำนวนเฉพาะบรรทัดละ 10 ตัวเลข

```
import java.util.Scanner;
public class Ex3_20{
    public static void main(String[] args) {
        final int nPrime = 50;
        final int nline = 10;
        int count = 0;
        int number = 2;
        System.out.println("The first 50 prime numbers are \n");
        while (count < nPrime) {
            boolean isPrime = true;
            for (int divisor = 2; divisor <= number / 2; divisor++) {
                if (number % divisor == 0) {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime) {
                count++;
                if (count % nline == 0) {
                    System.out.println(number);
                }
                else
                    System.out.print(number + " ");
            }
            number++;
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

The first 50 prime numbers are

```
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
```

3.6 บทสรุป

การทำงานของคอมพิวเตอร์จะทำงานเรียงลำดับคำสั่งลงมาตั้งแต่ต้นโปรแกรมจนจบโปรแกรม แต่ถ้าต้องการเปลี่ยนแปลงขั้นตอนการทำงานของคำสั่ง เช่น กระโดดข้ามไปที่ คำสั่งใดคำสั่งหนึ่ง หรือให้วนกลับมาทำคำสั่งที่เคยทำไปแล้ว ลักษณะการสั่งงานแบบนี้จะต้องใช้คำสั่งควบคุม ดังนั้นคำสั่งควบคุมจึงเป็นคำสั่งที่ใช้เปลี่ยนแปลงลำดับขั้นตอนการทำงานของโปรแกรม ซึ่งแบ่งออกเป็น 3 ชนิด คือ

1. คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข (Unconditional branch Statement) ได้แก่คำสั่ง goto
2. คำสั่งให้ไปทำงานโดยมีเงื่อนไข (Condition Statement) ได้แก่คำสั่ง if, switch
3. คำสั่งให้ไปทำงานแบบเป็นวงจร (Loop Control Statement) ได้แก่คำสั่ง while, do while, for

3.7 แบบฝึกหัดปฏิบัติการ

1. จงเขียนโปรแกรมที่อ่านค่า Fahrenheit degree แบบ double จากนั้นให้แปลงเป็น Celsius สูตรของการคำนวณคือ

$$\text{Celsius} = (5/9) * (\text{Fahrenheit} - 32)$$

ข้อมูลนำเข้า ค่า Fahrenheit degree แบบ double

ข้อมูลส่งออก แปลงเป็น Celsius

ข้อมูลนำเข้า	ข้อมูลส่งออก
32.0	0

2. (Financial application: compound value) กำหนดให้ฝากเงิน 100 บาท ทุกเดือนในบัญชีโดยมีอัตราดอกเบี้ยรายปีเป็น 5%. อัตราดอกเบี้ยรายเดือนจะถูกคิดหลังสิ้นเดือน ดังนั้นยอดเงินของเดือนที่ 1 จะเป็น

$$100 * (1 + 0.00417) = 100.417$$

ยอดเงินของเดือนที่ 2 จะเป็น

$$(100 + 100.417) * (1 + 0.00417) = 201.252$$

ยอดเงินของเดือนที่ 2 จะเป็น

$$(100 + 201.252) * (1 + 0.00417) = 302.507$$

จงเขียนโปรแกรมที่รับค่ายอดเงินจากผู้ใช้นั้นคำนวณยอดเงินหลังเดือนที่ 6

ข้อมูลนำเข้า ค่ายอดเงินจากผู้ใช้

ข้อมูลส่งออก คำนวณยอดเงินหลังเดือนที่ 6

ข้อมูลนำเข้า	ข้อมูลส่งออก
100	608.82

3. เขียนโปรแกรมที่รองรับการป้อนข้อมูลจากผู้ใช้โดยให้ป้อนสามจุดคือ (x1, y1), (x2, y2), (x3, y3) ของสามเหลี่ยมและคำนวณพื้นที่ของสามเหลี่ยมดังกล่าวโดยสูตรของพื้นที่สามเหลี่ยมคือ

$$\text{area} = \sqrt{s(s - \text{side1})(s - \text{side2})(s - \text{side3})}$$

$$s = (\text{side1} + \text{side2} + \text{side3}) / 2;$$

ข้อมูลนำเข้า ค่ายอดเงินจากผู้ใช้

ข้อมูลส่งออก คำนวณยอดเงินหลังเดือนที่ 6

ข้อมูลนำเข้า	ข้อมูลส่งออก
1.5 -3.4 4.6 5 9.5 -3.4	33.6

4. (Financial application: calculate interest) กรณีที่ผู้ใช้ทราบยอดคงเหลือ และอัตราดอกเบี้ยรายปี เราสามารถคำนวณดอกเบี้ยของเดือนถัดไปได้ด้วยสูตรต่อไปนี้

$$\text{interest} = \text{balance} * (\text{annualInterestRate} / 1200)$$

จงเขียนโปรแกรมเพื่อรับยอดคงเหลือ และอัตราดอกเบี้ยรายปี และคำนวณดอกเบี้ยของเดือนถัดไป

ข้อมูลนำเข้า ยอดคงเหลือ และอัตราดอกเบี้ยรายปี

ข้อมูลส่งออก ดอกเบี้ยของเดือนถัดไป

ข้อมูลนำเข้า	ข้อมูลส่งออก
1000 3.5	2.91667

5. ให้เขียนโปรแกรมภาษาจาวาเพื่อหาค่าของ $f(x, n)$ เมื่อ x สามารถเป็นจำนวนจริงใด ๆ n เป็นจำนวนเต็มที่ไม่มากกว่าหรือเท่ากับ 0 และฟังก์ชัน $f(x, n)$ มีการนิยามดังนี้

$$f(x, n) = \sum_{i=0:n} x^i$$

ข้อมูลนำเข้า รับข้อมูลเลข x และ n

ข้อมูลส่งออก ค่าของ $f(x, n)$

ข้อมูลนำเข้า	ข้อมูลส่งออก
10 3	1111

6. ให้เขียนโปรแกรมภาษาจาวาเพื่อแสดงค่าของ a_0, a_1, \dots, a_n ที่สัมพันธ์กับสมการ recurrence $a_k = k^2 a_{k-1} - a_{k-2} + 3^k$ เมื่อ $k=2, 3, 4, \dots$ โดยค่าเริ่มต้น n, a_0 และ a_1 ถูกกำหนดโดยผู้ใช้

ข้อมูลนำเข้า รับข้อมูลเลข x และ n

ข้อมูลส่งออก ค่าของ $f(x, n)$

ข้อมูลนำเข้า	ข้อมูลส่งออก
3 3 7	3 7 34 326

9. Occurrence of max number

จงเขียนโปรแกรมที่อ่านตัวเลขจำนวนจริง หาตัวเลขที่มีขนาดใหญ่มากที่สุดจากกลุ่มของตัวเลขดังกล่าว นับจำนวนครั้งของการปรากฏเลขดังกล่าว โดยให้ตัวเลข Input จบด้วยเลข 0 ตัวอย่างเช่น ถ้าป้อนตัวเลขต่อไปนี้ 3 5 2 5 5 0 โปรแกรมจะค้นหาจำนวนตัวเลขที่มากที่สุดคือ 5 และจำนวนครั้งของการเกิดเลข 5 คือ 4 เช่น

ข้อมูลนำเข้า รับข้อมูลเลขจำนวนเต็มบวก n ตัว

ข้อมูลส่งออก ตัวเลขที่มีค่ามากที่สุดและจำนวนครั้งของการปรากฏ

ข้อมูลนำเข้า	ข้อมูลส่งออก
3 5 2 5 5 0	5 4

บทที่ 4 คลาสและวัตถุ

วัตถุประสงค์

- 4.1 สามารถอธิบายความหมายของคลาสและวัตถุ
- 4.2 สามารถเขียนโปรแกรมเพื่อประกาศคลาส สร้างวัตถุ
- 4.3 สามารถอธิบายความหมายและเขียนโปรแกรมเพื่อสร้างคอนสตรักเตอร์
- 4.4 สามารถอธิบายความหมายของการห่อหุ้มของวัตถุ
- 4.5 สามารถอธิบายวัฏจักรชีวิตของวัตถุ

4.1 ความนำ

ออบเจกต์หรือวัตถุ (Object) คือสิ่งต่างๆ ที่มีอยู่ในชีวิตประจำวันแบ่งได้เป็นสองประเภทคือ สิ่งที่เป็นรูปธรรม (Tangible) คือสิ่งที่เป็นวัตถุและจับต้องได้ เช่น นักศึกษา ใบลงทะเบียน ปากกา และรถ เป็นต้น สิ่งที่เป็นนามธรรม (Intangible) คือสิ่งที่ไม่สามารถจับต้องได้ เช่น คะแนน รายชื่อวิชา บัญชีเงินฝาก และตารางเที่ยวบิน เป็นต้น โดยส่วนใหญ่ วัตถุต่างๆ จะประกอบไปด้วยคุณลักษณะ (Attribute) และพฤติกรรม (behavior)

4.2 นิยามและความหมายของคลาสและวัตถุ

คลาส คือ แม่แบบของสิ่งต่างๆ ที่เราสนใจโดยภายในคลาสนี้มีคุณลักษณะและพฤติกรรม ในการเขียนโปรแกรมเชิงวัตถุจะใช้คลาสเป็นต้นแบบสำหรับสร้างวัตถุ ตัวอย่างเช่น หากต้องการโปรแกรมสำหรับเก็บรายละเอียดของนักเรียนแต่ละคน การเขียนโปรแกรมเชิงวัตถุจะมองว่ากลุ่มของนักเรียนสามารถที่จะกำหนดเป็นคลาสได้ โดยภายในประกอบด้วยแอตทริบิวต์และเมธอด โดยส่วนของแอตทริบิวต์ของนักเรียนหรือคุณลักษณะของนักเรียนประกอบด้วยรหัสประจำตัว ชื่อ นามสกุล คณะ เกรดเฉลี่ย อายุ วันเดือนปีเกิด ในส่วนพฤติกรรมสามารถกำหนดได้เป็น การเดิน การวิ่ง การกิน การลงทะเบียนเรียน เป็นต้น ซึ่งหากผู้เขียนโปรแกรมต้องการเก็บข้อมูลของนักเรียนแต่ละคนสามารถเก็บข้อมูลของนักเรียนแต่ละคนผ่านการเขียนโปรแกรมโดยการประกาศวัตถุ เพื่อเก็บข้อมูลดังกล่าว

ออบเจกต์หรือวัตถุ (Object) คือสิ่งต่างๆ ที่มีอยู่ในชีวิตประจำวันแบ่งได้เป็นสองประเภทคือ

1. สิ่งที่เป็นรูปธรรม (Tangible) คือสิ่งที่เป็นวัตถุและจับต้องได้ เช่น นักศึกษา ใบลงทะเบียน ปากกา และรถ เป็นต้น
2. สิ่งที่เป็นนามธรรม (Intangible) คือสิ่งที่ไม่สามารถจับต้องได้ เช่น คะแนน รายชื่อวิชา บัญชีเงินฝาก และตารางเที่ยวบิน เป็นต้น

โดยส่วนใหญ่วัตถุต่างๆ จะประกอบไปด้วยคุณลักษณะ (attribute) และพฤติกรรม (behavior) โดยที่คุณลักษณะก็คือข้อมูลของวัตถุ ส่วนพฤติกรรมหรือเรียกว่าเมธอด (method) คือสิ่งที่วัตถุสามารถทำได้ ซึ่งในโปรแกรมเชิงวัตถุก็คือคำสั่งในการทำงาน โปรแกรมเชิงวัตถุจะประกอบด้วยวัตถุต่างๆ หลายวัตถุ ซึ่งแต่ละวัตถุจะมีคุณลักษณะต่างๆ ที่เป็นข้อมูลของวัตถุเอง จากที่กล่าวมาข้างต้นจะพบว่า คลาสเป็นการจัดกลุ่มของวัตถุที่มีคุณลักษณะและพฤติกรรมบางอย่างเหมือนกัน คลาสเปรียบเสมือนพิมพ์เขียวสำหรับสร้างวัตถุ

4.3 การประกาศคลาส

ในการประกาศคลาสสำหรับการใช้งานจะประกอบด้วยกลุ่มของแอทริบิวต์ (attribute) หรือตัวแปรของวัตถุ (instance variable) และกลุ่มของพฤติกรรม (method) จากนิยามข้างต้นหากต้องการสร้างคลาสที่เก็บข้อมูลของนักเรียนจะสามารถทำได้โดยกำหนดส่วนของแอทริบิวต์และเมธอด โดยส่วนของแอทริบิวต์ที่ควรมีสำหรับการนิยามในคลาสนักเรียน เช่น รหัสประจำตัว ชื่อ นามสกุล คณะ เกรตเฉลี่ย อายุ วันเดือนปีเกิด ในส่วนพฤติกรรมสามารถกำหนดได้เป็น การเดิน การวิ่ง การกินการลงทะเบียนเรียน

คลาสของภาษาจาวามีสมาชิก (class member) ได้ 3 ประเภทดังนี้

1. กลุ่มของแอทริบิวต์ (data member) หรือ variable สำหรับเก็บค่าสถานะ (state) ของ instance ของคลาสนั้น แอทริบิวต์หรือ data member ประกอบด้วย instance variable และ class variable
2. กลุ่มของพฤติกรรม (method member) คือฟังก์ชันซึ่งเป็นพฤติกรรมของ instance ของคลาสนั้น เมธอด (method) เป็นชุดคำสั่งย่อยที่ทำงานกับข้อมูล ตัวเมธอดอาจจะมีหรือไม่มีพารามิเตอร์ก็ได้ โดยที่เมธอดอาจจะมีการคืนค่าออกมาภายหลังจากที่ทำงานเสร็จ ประเภทของเมธอด ประกอบด้วย
 - a. Constructor คอนสตรัคเตอร์เป็นเมธอดชนิดหนึ่งที่มีชื่อเดียวกับชื่อคลาส ถูกเรียกเพื่อสร้างวัตถุและกำหนดค่าเริ่มต้นให้กับวัตถุ คอนสตรัคเตอร์ที่ไม่มีพารามิเตอร์จะเรียกว่า default constructor คอนสตรัคเตอร์จะต้องมีชื่อเดียวกับชื่อคลาส การสร้างคอนสตรัคเตอร์จะไม่มีส่วนของการคืนค่ากลับ หน้าที่ของคอนสตรัคเตอร์คือการกำหนดค่าเริ่มต้นให้กับวัตถุ
 - b. Mutator methods (setter) ทำหน้าที่เปลี่ยนข้อมูลภายในวัตถุ
 - c. Accessor methods (getter) ทำหน้าที่ดึงข้อมูลจากวัตถุ
 - d. เมธอดอื่น ๆ
3. กลุ่มของคลาส (class member/ inner class) คือคลาสที่ถูกกำหนดขึ้นภายในคลาสนั้น ซึ่งอาจจะถูกกำหนดไว้เพื่อใช้งานภายในคลาสนั้นเอง หรือจากคลาสนอกคลาสนั้นก็ได้

ตัวอย่างที่ 4.1 ตัวอย่างการประกาศคลาส Calculation

```
public class Calculation{
    int x; //กลุ่มของแอทริบิวต์
    public void f() { } //กลุ่มของเมธอด
    class A{} //กลุ่มของคลาส
}
```

ตัวอย่างที่ 4.1 แสดงตัวอย่างการสร้างคลาสที่ชื่อ Calculation โดยภายในคลาสดังกล่าวมีแอทริบิวต์ 1 แอทริบิวต์ คือ x นอกจากนี้ในคลาส Calculation มีเมธอด 1 เมธอดชื่อ f() นอกจากแอทริบิวต์และเมธอดแล้วภายในคลาสนี้ยังจะมีคลาสอื่นอยู่ภายในด้วย

การนำเสนอคลาสสามารถนำเสนอเพื่อให้เข้าใจต่อความเข้าใจโดยการใช้คลาสไดอะแกรม โดยคลาสไดอะแกรม เป็นแผนภาพที่ใช้แสดงคลาสและความสัมพันธ์ในแง่ต่างๆ ระหว่างคลาสเหล่านั้น โดยความสัมพันธ์นั้นเป็นแบบ Static ไม่ใช่แบบ Dynamic ตัวอย่างความสัมพันธ์แบบ Static เช่น เจ้าของบัญชี เป็นเจ้าของบัญชีเงินฝาก ตัวอย่างความสัมพันธ์แบบ Dynamic เช่น เจ้าของบัญชี ฝากเงินเข้า/ถอนเงินจากบัญชีเงินฝาก ก่อนอื่นควรรู้จักวัตถุ (Object) และ คลาส (Class)

วัตถุ (Objects) เป็นสิ่งที่อยู่ในระบบซอฟต์แวร์ มีความเป็นเอกเทศใช้เป็นตัวแทนของข้อมูล บรรจุสถานะ รวมถึงบริการ ซึ่งจะอธิบายการทำงานของวัตถุนั้นในระบบ ไม่มีการแชร์ข้อมูลระหว่างกัน แต่สื่อสารกันโดยใช้ message passing

การจะจัดวัตถุให้อยู่ใน class เดียวกัน ขึ้นอยู่กับผู้ออกแบบระบบว่ามองที่คุณสมบัติใดเป็นหลัก ถ้ามองวัตถุในแง่ความกว้าง ความสูง และเป็นวัตถุที่แข็ง กำแพงและประตู อยู่คลาสเดียวกัน ถ้ามองวัตถุในแง่การเปิดปิดได้ประตูและหน้าต่าง อยู่คลาสเดียวกัน

วิธีการเขียนคลาสไดอะแกรม

คลาสไดอะแกรมมีคุณสมบัติมากมายที่ต้องพิจารณาขณะเขียน ประเด็นที่สำคัญดังนี้ ชื่อของ Class diagram ควรมีความหมายเพื่ออธิบายลักษณะของระบบควรระบุความรับผิดชอบ (คุณสมบัติและวิธีการ) ของแต่ละคลาสอย่างชัดเจน สำหรับแต่ละคลาสควรระบุจำนวนคุณสมบัติขั้นต่ำเนื่องจากคุณสมบัติที่ไม่จำเป็นจะทำให้ไดอะแกรมมีความซับซ้อน

สัญลักษณ์ต่างๆ ในคลาสไดอะแกรม

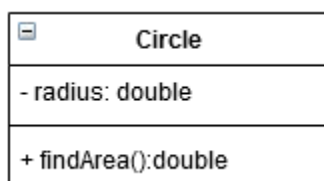
คลาสไดอะแกรมแผนภาพแบ่งออกเป็นสามส่วน ส่วนด้านบนใช้เพื่อกำหนดชื่อคลาส ส่วนที่สองใช้เพื่อแสดงคุณสมบัติของคลาส ส่วนที่สามใช้เพื่ออธิบายการดำเนินการที่ดำเนินการโดยคลาส

ความสัมพันธ์ แสดงให้เห็นว่าวัตถุมีองค์ประกอบที่เกี่ยวข้องกันและการเชื่อมโยงกันอย่างไรอธิบายการทำงานของระบบมีความสัมพันธ์ลักษณะด้วยกันคือ

1. Dependency คือความสัมพันธ์ระหว่างสองสิ่งที่มีการเปลี่ยนแปลงในองค์ประกอบหนึ่งส่งผลกระทบต่ออีกอันเปลี่ยนไปด้วย
2. Association เป็นชุดของความสัมพันธ์ที่เชื่อมต่อองค์ประกอบของโมเดล
3. Generalization เป็นการกำหนดความสัมพันธ์ที่เชื่อมต่อองค์ประกอบพิเศษกับองค์ประกอบทั่วไป โดยทั่วไปจะอธิบายความสัมพันธ์การสืบทอดของวัตถุ
4. Realization เป็นการกำหนดความสัมพันธ์ซึ่งองค์ประกอบทั้งสองเชื่อมต่อกัน องค์ประกอบหนึ่งอธิบายถึงความรับผิดชอบบางอย่างซึ่งไม่ได้นำไปปฏิบัติและองค์ประกอบอื่นใช้ความรับผิดชอบนั้น ความสัมพันธ์นี้มีอยู่ในกรณีของอินเทอร์เฟซ

คลาสไดอะแกรมเป็นไดอะแกรมแบบคงที่และใช้เพื่อสร้างแบบจำลองมุมมองแบบคงที่ของระบบ มุมมองแบบโครงสร้างของระบบ คลาสไดอะแกรมถือเป็นพื้นฐานสำหรับส่วนประกอบ ใช้ในการมองเห็นมุมมองแบบคงที่ของระบบหรือใช้เพื่อสร้างรหัสคำสั่ง

ตัวอย่างการสร้างคลาสสำหรับวงกลมซึ่งประกอบด้วยแอตทริบิวต์ radius และเมธอด findArea() โดยสามารถเขียนในรูปคลาสไดอะแกรม และชุดคำสั่งได้ดังต่อไปนี้



ตัวอย่างที่ 4.2 ตัวอย่างการประกาศคลาส Circle

```

class Circle {
    double radius = 1.0;
    double findArea(){
        return radius * radius * 3.14159;
    }
}
  
```

ตัวอย่างที่ 4.2 แสดงตัวอย่างการประกาศคลาส Circle ที่ประกอบด้วยแอตทริบิวต์ radius และเมธอด findArea() สำหรับคำนวณพื้นที่ของวงกลมที่มีรัศมีตามที่ผู้ใช้กำหนด

ตัวอย่างที่ 4.3 ตัวอย่างการประกาศคลาส Person

```
class Person {
    public String name;
    public String surname;
    public int age;
    int heartRate = 80 ;
    void setHeartRate (float heartRate) {
        heartRate = heartRate;
    }
    float getHeartRate () {
        return heartRate;
    }
    void breath () {
        System.out.println(name + " is breath with " + heartRate);
    }
}
```

ตัวอย่างที่ 4.3 แสดงตัวอย่างการประกาศคลาส Person ที่ประกอบด้วยแอตทริบิวต์ดังนี้ ชื่อ (name) นามสกุล (surname) อายุ (age) อัตราการเต้นของหัวใจ (heartRate) และมีเมธอดดังนี้

void setHeartRate (float heartRate) เป็นเมธอดสำหรับการกำหนดอัตราการเต้นของหัวใจ

float getHeartRate () เป็นเมธอดสำหรับการดึงอัตราการเต้นของหัวใจ

void breath () เป็นเมธอดสำหรับแสดงชื่อ และอัตราการเต้นของหัวใจ ของแต่ละคน

4.4 วัฏจักรชีวิตของวัตถุ

เมื่อผู้เขียนโปรแกรมได้นิยามคลาสเรียบร้อยแล้ว ในขั้นตอนต่อไปหากต้องการสร้างวัตถุ (object) สำหรับทำงานที่ต้องการมีขั้นตอนในการสร้างวัตถุดังนี้

4.4.1 การประกาศตัวแปรอ้างอิงวัตถุ (Declaring Object Reference Variables)

ตัวแปรอ้างอิงวัตถุเป็นตัวแปรที่เก็บที่อยู่หรืออ้างอิงไปยังวัตถุ รูปแบบการประกาศตัวแปรอ้างอิงวัตถุมีรูปแบบคือ

```
ClassName objectReference;
```

ตัวอย่าง

```
Circle myCircle;
```

4.4.2 การสร้างวัตถุ (Creating Objects)

ภายหลังการประกาศตัวแปรอ้างอิงวัตถุแล้วหลังจากนั้นจะสร้างวัตถุภายในหน่วยความจำโดยใช้คำสั่ง new และตามด้วยคอนสตรักเตอร์ที่จะใช้สร้างวัตถุ เช่น new ClassName() และเก็บที่อยู่ดังกล่าวไว้ในตัวแปรอ้างอิงวัตถุที่ประกาศไว้ก่อนหน้านี้ โดยมีรูปแบบดังนี้

```
objectReference = new ClassName();
```

ตัวอย่าง

```
myCircle = new Circle();
```

วัตถุที่ถูกอ้างอิง (object reference) จะถูกกำหนดให้กับตัวแปรอ้างอิงวัตถุ (object reference variable) เช่น myCircle เป็นตัวแปรที่เก็บที่อยู่ของวัตถุที่สร้างจากคลาส Circle

การประกาศวัตถุและการสร้างวัตถุในขั้นตอนเดียว

```
ClassName objectReference = new ClassName();
```

ตัวอย่าง

```
Circle myCircle = new Circle();
```

4.4.3 การเข้าถึงข้อมูลของวัตถุ (Accessing Objects)

การอ้างอิงแอตทริบิวต์หรือข้อมูลภายในวัตถุสามารถทำได้โดยใช้รูปแบบดังต่อไปนี้

`objectReference.data`

เช่น

```
myCircle.radius;
```

การอ้างอิงเมธอดภายในวัตถุสามารถทำได้โดยใช้รูปแบบดังต่อไปนี้

`objectReference.method`

เช่น

```
myCircle.findArea();
```

ตัวอย่างการเขียนโปรแกรมเพื่อสร้างคลาสและสร้างวัตถุสามารถแสดงตัวอย่างได้ดังนี้

ตัวอย่างที่ 4.5 ตัวอย่างการประกาศคลาส Animal และการสร้างวัตถุจากคลาส Animal

```
class Animal{
    private String name;
    private String color;
    protected int leg;
    public void setName(String name){
        this.name= name;
    }
    public void setColor(String color){
        this.color= color;
    }
    public void setLeg(int leg){
        this.leg= leg;
    }
    public String getName(){
        return this.name;
    }
    public String getColor(){
        return this.color;
    }
    public int getLeg(){
        return this.leg;
    }
    public void animalRun(){
        System.out.println(this.name+" run with "+this.leg+ " legs");
    }
    public static void main(String[] args){
        Animal tiger = new Animal();
        tiger.setName("Triger");
        tiger.setColor("Yellow");
        tiger.setLeg(4);
        tiger.animalRun();
    }
}
```

Triger run with 4 legs

ตัวอย่างที่ 4.5 แสดงตัวอย่างการประกาศคลาส Animal ซึ่งเป็นคลาสสำหรับเก็บข้อมูลของสัตว์ โดยภายในคลาสประกอบด้วยแอทริบิวต์ดังนี้ ชื่อ (name) สี (color) จำนวนขา (leg) และมีเมธอดดังนี้

public void setName(String name) เป็นเมธอดสำหรับการกำหนดชื่อให้สัตว์

public void setColor(String color) เป็นเมธอดสำหรับการกำหนดสีให้สัตว์

public void setLeg(int leg) เป็นเมธอดสำหรับการกำหนดจำนวนขาให้สัตว์

public String getName() เป็นเมธอดสำหรับการดึงชื่อสัตว์

public String getColor() เป็นเมธอดสำหรับการดึงสีของสัตว์

public int getLeg() เป็นเมธอดสำหรับการดึงจำนวนขาของสัตว์

public void animalRun() เป็นเมธอดสำหรับการกำหนดการแสดงความเคลื่อนไหวเมื่อสัตว์แต่ละตัววิ่ง

เมื่อประกาศคลาสเรียบร้อยแล้วโปรแกรมจะยังไม่ทำงาน เมื่อต้องการเริ่มการทำงานของโปรแกรม จะเริ่มประมวลผลที่ public static void main(String[] args) เป็นเมธอดเริ่มต้นโดยภายในเมธอดมีการสร้างตัวแปรอ้างอิงวัตถุ ชื่อ tiger ซึ่งมีชนิดข้อมูลเป็น Animal และสร้างวัตถุจากคอนสตรัคเตอร์ของคลาสโดยใช้คำสั่ง new Animal(ซึ่งคำสั่งนี้ ()) จะเป็นการสร้างวัตถุในหน่วยความจำโดยภายในวัตถุนี้จะมีแอทริบิวต์และเมธอดตามที่ประกาศไว้ในคลาส Animal เมื่อสร้างวัตถุจากคลาส Animal เสร็จแล้วจะนำ Reference ของวัตถุนี้ไปเก็บในตัวแปร tiger ซึ่งเป็นตัวแปรอ้างอิงวัตถุ เมื่อ

สร้างวัตถุเสร็จแล้วเมื่อต้องการกำหนดชื่อ สี จำนวนขา ให้กับวัตถุที่สร้างขึ้นจะเรียกผ่านตัวแปรอ้างอิงวัตถุที่ชื่อ tiger โดยมีการเรียกดังนี้

```
tiger.setName("Triger");
```

```
tiger.setColor("Yellow");
```

```
tiger.setLeg(4);
```

ซึ่งเมื่อกำหนดค่าต่างๆ เรียบร้อยหากต้องการแสดงข้อมูลภายในวัตถุที่สร้างขึ้นจะทำได้โดยการเรียกเมธอดผ่านตัวแปรอ้างอิงวัตถุที่ชื่อ tiger โดยมีการเรียกดังนี้ tiger.animalRun(); ซึ่งผลลัพธ์ที่ได้โปรแกรมจะแสดงข้อความดังนี้ Triger run with 4 legs

ตัวอย่างที่ 4.6 ตัวอย่างการประกาศคลาส Circle และการสร้างวัตถุจากคลาส Circle

```
class Circle {
    private double radius = 1;
    private static int numberOfObjects = 0;
    public Circle() {
        numberOfObjects++;
    }
    public Circle(double newRadius) {
        radius = newRadius;
        numberOfObjects++;
    }
    public double getRadius() {
        return radius;
    }
    public void setRadius(double newRadius) {
        radius = (newRadius >= 0) ? newRadius : 0;
    }
    public static int getNumberOfObjects() {
        return numberOfObjects;
    }
    public double getArea() {
        return radius * radius * Math.PI;
    }
}
public class Ex4_6 {
    public static void main(String[] args) {
        Circle myCircle = new Circle(5.0);
        System.out.println("The area of the circle of radius " + myCircle.getRadius() + " is "
+ myCircle.getArea());
        myCircle.setRadius(myCircle.getRadius() * 1.1);
        System.out.println("The area of the circle of radius " + myCircle.getRadius() + " is "
+ myCircle.getArea());
    }
}
```

The area of the circle of radius 5.0 is 78.53981633974483

The area of the circle of radius 5.5 is 95.03317777109125

ตัวอย่างที่ 4.6 แสดงตัวอย่างการประกาศคลาส Circle ซึ่งเป็นคลาสจำลองวงกลม โดยภายในคลาสประกอบด้วยแอตทริบิวต์ดังนี้ รัศมี (radius) จำนวนของวงกลมที่สร้างจากคลาส (numberOfObjects) และมีเมธอดดังนี้

public Circle() เป็นคอนสตรัคเตอร์ซึ่งเป็นเมธอดชนิดหนึ่งที่มีชื่อเดียวกับชื่อคลาส ถูกเรียกเพื่อสร้างวัตถุและกำหนดค่าเริ่มต้นให้กับวัตถุ

`public Circle(double newRadius)` เป็นคอนสตรักเตอร์ซึ่งสร้างวัตถุและกำหนดค่าเริ่มต้นให้กับวัตถุโดยจะมีการกำหนดค่ารัศมีให้กับวงกลมที่สร้างขึ้น

`public double getRadius()` เป็นเมธอดสำหรับดึงค่ารัศมีของวงกลม

`public double getArea()` เป็นเมธอดสำหรับดึงค่าพื้นที่ของวงกลม

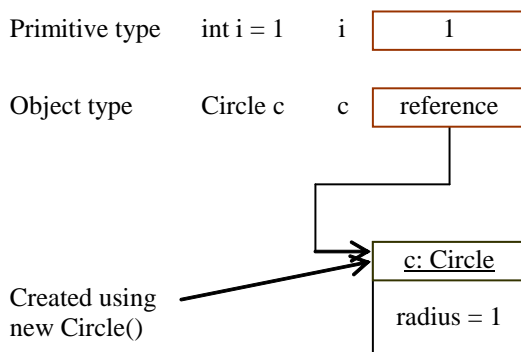
เมื่อประกาศคลาสเรียบร้อยแล้วตัวโปรแกรมจะยังไม่ทำงาน เมื่อต้องการเริ่มการทำงานของโปรแกรม จะเริ่มประมวลผลที่ `public static void main(String[] args)` เป็นเมธอดเริ่มต้นโดยภายในเมธอดมีการสร้างตัวแปรอ้างอิงวัตถุชื่อ `myCircle` ซึ่งมีชนิดข้อมูลเป็น `Circle` และสร้างวัตถุจากคอนสตรักเตอร์ของคลาสโดยใช้คำสั่ง `new Circle(5.0)` ซึ่งคำสั่งนี้จะเป็นการสร้างวัตถุในหน่วยความจำและกำหนดค่าเริ่มต้นให้กับวงกลมโดยจะมีการกำหนดค่ารัศมีให้กับวงกลมที่สร้างขึ้นเป็น 5.0 โดยภายในวัตถุนี้จะมีแอตทริบิวต์และเมธอดตามที่ประกาศไว้ในคลาส `Circle` เมื่อสร้างวัตถุจากคลาส `Circle` เสร็จแล้วจะนำ Reference ของวัตถุนี้ไปเก็บในตัวแปร `myCircle` ซึ่งเป็นตัวแปรอ้างอิงวัตถุ เมื่อสร้างวัตถุเสร็จแล้วเมื่อต้องการดึงค่ารัศมี และคำนวณพื้นที่ของวงกลมที่สร้างขึ้นจะเรียกผ่านตัวแปรอ้างอิงวัตถุที่ชื่อ `myCircle` โดยมีการเรียกดังนี้

```
System.out.println("The area of the circle of radius " + myCircle.getRadius() + " is " + myCircle.getArea());
```

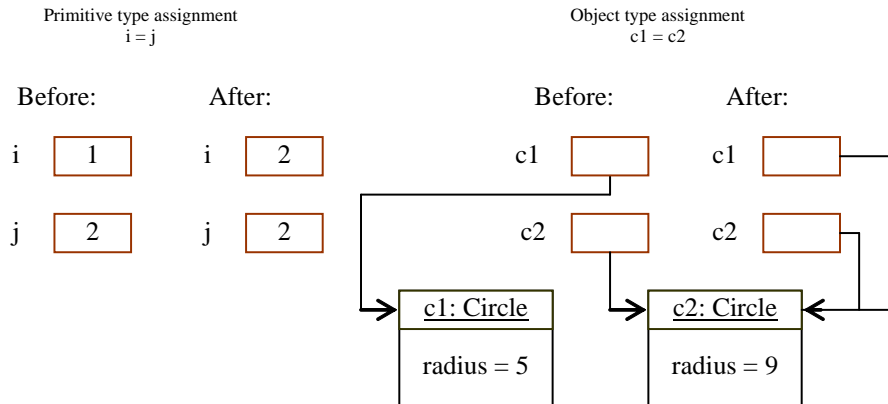
ซึ่งผลลัพธ์ที่ได้โปรแกรมจะแสดงข้อความดังนี้ The area of the circle of radius 5.0 is 78.53981633974483

4.4.4 การคัดลอกข้อมูลของ Primitive Data Types และ Object Types

ตัวแปรอ้างอิงวัตถุ (Object Types) และตัวแปรของชนิดข้อมูลธรรมดา (Primitive Data Types) จะมีรูปแบบการทำงานที่แตกต่างกัน ดังรูป



การคัดลอกข้อมูลของ Primitive Data Types และ Object Types



4.4.5 Garbage Collection

จากรูปก่อนหน้าจะพบว่าหลังจากกำหนดให้ `c1 = c2` จะพบว่า `c1` ชี้ไปยังตำแหน่งเดียวกับ `c2` จากเหตุการณ์ดังกล่าวจะพบว่า `c1` จะไม่ถูกใช้ซึ่งทำให้วัตถุดังกล่าวกลายเป็นขยะหรือ garbage เมื่อเกิดเหตุการณ์ดังกล่าว garbage จะถูกเก็บโดย JVM ซึ่งก็คือการคืนหน่วยความจำที่ใช้นั่นเอง

4.5 คอนสตรัคเตอร์ (Constructor)

คอนสตรัคเตอร์เป็นเมธอดชนิดหนึ่งที่มีชื่อเดียวกับชื่อคลาส ถูกเรียกเพื่อสร้างวัตถุและกำหนดค่าเริ่มต้นให้กับวัตถุ คอนสตรัคเตอร์ที่ไม่มีพารามิเตอร์จะเรียกว่า Default constructor คอนสตรัคเตอร์จะต้องมีชื่อเดียวกับชื่อคลาส การสร้างคอนสตรัคเตอร์จะไม่มีส่วนของการคืนค่ากลับ หน้าที่ของคอนสตรัคเตอร์คือการกำหนดค่าเริ่มต้นให้กับวัตถุ (initializing objects) เช่น

ตัวอย่างที่ 4.7 ตัวอย่างการเขียนคอนสตรัคเตอร์ภายในคลาส Animal

```
class Animal{
    private String name;
    private String color;
    protected int leg;
    public void setName(String name){
        this.name= name;
    }
    public void setColor(String color){
        this.color= color;
    }
    public void setLeg(int leg){
        this.leg= leg;
    }
    public String getName(){
        return this.name;
    }
    public String getColor(){
        return this.color;
    }
    public int getLeg(){
        return this.leg;
    }
    public void animalRun(){
        System.out.println(this.name+" run with "+this.leg+ " legs");
    }
}
```

```

public static void main(String[] args){
    Animal tiger = new Animal();
    tiger.setName("Triger");
    tiger.setColor("Yellow");
    tiger.setLeg(4);
    tiger.animalRun();
}
}

```

เมื่อต้องการสร้างวัตถุจะเรียกใช้คอนสตรัคเตอร์พร้อมกำหนดค่าเริ่มต้นให้กับวัตถุในคอนสตรัคเตอร์ เช่น

```
Animal tiger = new Animal();
```

คอนสตรัคเตอร์เป็นโปรแกรมย่อยสำหรับกำหนดค่าเริ่มต้นให้แก่วัตถุ มีคุณสมบัติพิเศษดังนี้ คือ

1. จะถูกดำเนินการเมื่อมีการสร้างวัตถุเสมอไม่สามารถละเลย หรือหลงลืมได้ ทุกครั้งที่มีการเรียกคำสั่ง new จะต้องมีการเรียกใช้คอนสตรัคเตอร์เสมอ
2. ต้องมีชื่อเมธอดเหมือนกับชื่อคลาส ถ้าเราลองสังเกตให้ดีหลังคำสั่ง new จะมีชื่อคลาสแล้วตามด้วยวงเล็บ
3. การประกาศโปรแกรมย่อยคอนสตรัคเตอร์ไม่ต้องระบุ void และไม่ต้องระบุ return type เช่นรูปแบบการประกาศคอนสตรัคเตอร์แบบดีฟอลท์

ตัวอย่างที่ 4.8 ตัวอย่างการเขียนคอนสตรัคเตอร์ภายในคลาส Student

```

class Student {
    String id;
    String name;
    short startYear;
    float gpa = 0.0F;
    Student() {
    }
    Student(String newId, String newName, short newStartYear) {
        id = newId;
        name = newName;
        startYear = newStartYear;
    }
    void print() {
        System.out.println("id : "+ id);
        System.out.println("name : "+ name);
        System.out.println("year : "+ startYear);
        System.out.println("gpa : "+ gpa);
        System.out.println("=====");
    }
}

```

ตัวอย่างที่ 4.8 แสดงตัวอย่างการประกาศคลาส Student และการเขียนคอนสตรัคเตอร์ภายในคลาส Student ซึ่งเป็นคลาสสำหรับเก็บข้อมูลของนักเรียน โดยภายในคลาสประกอบด้วยแอตทริบิวต์ดังนี้ ชื่อ (name) รหัสประจำตัวนักเรียน (id) ปีที่เข้าศึกษา (startYear) เกรดเฉลี่ย (gpa) และมีเมธอดดังนี้

void print() เป็นเมธอดสำหรับการแสดงรายละเอียดของนักเรียน

นอกจากนี้ภายในคลาสจะมีการกำหนดคอนสตรัคเตอร์ซึ่งเป็นเมธอดชนิดหนึ่งที่มีชื่อเดียวกับชื่อคลาส ถูกเรียกเพื่อสร้างวัตถุและกำหนดค่าเริ่มต้นให้กับวัตถุ โดยภายในคลาส Student จากตัวอย่างข้างต้นจะมี คอนสตรัคเตอร์ 2 คอนสตรัคเตอร์ คือ Student(){} ซึ่งเป็นการประกาศคอนสตรัคเตอร์แบบดีฟอลท์และ Student(String newId, String newName, short newStartYear){ } ซึ่งเป็นการประกาศคอนสตรัคเตอร์โดยมีการรับพารามิเตอร์เป็น String newId, String newName, short newStartYear

หรือเราสามารถประกาศคอนสตรัคเตอร์ที่มีพารามิเตอร์ตามที่ต้องการแทนที่คอนสตรัคเตอร์แบบดีฟอลท์โดยการประกาศพารามิเตอร์และกำหนดค่าให้กับตัวแปรได้

ตัวอย่างที่ 4.9 ตัวอย่างการเขียนคอนสตรัคเตอร์ภายในคลาส Student และการสร้างวัตถุจากคลาส Student โดยการใช้คอนสตรัคเตอร์ในรูปแบบต่างๆ

```
class Student {
    String id;
    String name;
    short startYear;
    float gpa = 0.0F;
    Student() {
    }
    Student(String newId, String newName, short newStartYear) {
        id = newId;
        name = newName;
        startYear = newStartYear;
    }
    void print() {
        System.out.println("id : "+ id);
        System.out.println("name : "+ name);
        System.out.println("year : "+ startYear);
        System.out.println("gpa : "+ gpa);
        System.out.println("=====");
    }
}

class TestStudent {
    public static void main(String[] args) {
        Student std1;
        std1 = new Student("46F0000", "Tom", 2546);
        Student std2;
        std2 = new Student("46F0999", "Jerry", 2546);
        std1.print();
        std2.print();
    }
}
```

ตัวอย่างที่ 4.9 แสดงตัวอย่างการประกาศคลาส Student และการเขียนคอนสตรัคเตอร์ภายในคลาส Student

เมื่อประกาศคลาสเรียบร้อยแล้วโปรแกรมจะยังไม่ทำงาน เมื่อต้องการเริ่มการทำงานของโปรแกรม จะเริ่มประมวลผลที่ public static void main(String[] args) เป็นเมธอดเริ่มต้นโดยภายในเมธอดมีการสร้างตัวแปรอ้างอิงวัตถุชื่อ std1 ซึ่งมีชนิดข้อมูลเป็น Student และสร้างวัตถุจากคอนสตรัคเตอร์ของคลาสโดยใช้คำสั่ง new Student("46F0000", "Tom", 2546) ซึ่งคำสั่งนี้จะเป็นการสร้างวัตถุในหน่วยความจำและกำหนดค่าเริ่มต้นให้กับนักเรียนโดยจะมีการกำหนดค่ารหัสประจำตัว ชื่อ ปีที่เข้าศึกษาให้กับนักเรียนที่สร้างขึ้นเป็น "46F0000", "Tom", 2546 ตามลำดับ

เมื่อเราประกาศคอนสตรัคเตอร์ของตัวเองขึ้นมาจะทำให้ไม่สามารถเรียกใช้คอนสตรัคเตอร์แบบดีฟอลต์ได้อีกต่อไป ทำให้การสร้างวัตถุจากคลาส Student ทุกครั้งจะต้องระบุ พารามิเตอร์ รหัสและชื่อนักศึกษาอย่างครบถ้วน หากไม่ใส่พารามิเตอร์หรือใส่ไม่ครบหรือผิดประเภทคอมไพเลอร์จะตรวจสอบได้และแจ้งข้อผิดพลาดให้เราทราบ เมื่อประกาศคอนสตรัคเตอร์ ภายในคอนสตรัคเตอร์ควรจะมีการกำหนดค่าเริ่มต้นให้กับตัวแปรของวัตถุให้ครบทุกตัวตอนสร้างวัตถุ

4.6 การห่อหุ้มและการซ่อนข้อมูลของวัตถุ

4.6.1 ระดับการเข้าถึงวัตถุ (Visibility Modifiers and Accessor Methods)

ในหัวข้อนี้เราจะมาสรุปคำที่ใช้เขียนเกี่ยวกับการประกาศคุณสมบัติหรือเมธอดภายในวัตถุเพื่อกำหนดระดับการเข้าถึงวัตถุทั้งหมด (Visibility Modifier) ประกอบด้วย

1. ระดับ default เป็นระดับการเข้าถึงโดยปริยาย เมื่อวัตถุมีการประกาศคุณสมบัติและเมธอด โดยไม่กำหนดระดับการเข้าถึง เมื่อมีการกำหนดดังกล่าว คลาส ตัวแปรหรือข้อมูลสามารถที่จะเข้าถึงได้โดยคลาสทุก ๆ คลาสที่อยู่ใน package เดียวกัน

2. ระดับ private เป็นระดับการเข้าถึงเมื่อมีการประกาศ โดยคำว่า private เมื่อมีการกำหนดดังกล่าว คลาส ตัวแปร ข้อมูล หรือ เมธอดสามารถจะเข้าถึงได้เฉพาะภายในคลาส การเข้าถึงข้อมูลดังกล่าวต้องทำผ่านเมธอด get หรือ set เท่านั้น

3. ระดับ protected เป็นระดับการเข้าถึงเมื่อมีการประกาศ โดยคำว่า protected การอ้างถึงต้องใช้คำสั่ง this หรือ super เท่านั้น

4. ระดับ public เป็นระดับการเข้าถึง เมื่อมีการประกาศ ด้วยคำว่า public การกำหนดคำดังกล่าวกับคลาส ตัวแปร ข้อมูล หรือ เมธอด จะทำให้คลาส ตัวแปร ข้อมูลหรือเมธอดดังกล่าวสามารถที่จะเข้าถึงได้โดยคลาสทุก ๆ คลาสที่อยู่ใน package เดียวกันหรือต่าง package

ตัวอย่างที่ 4.10 ตัวอย่างระดับการเข้าถึงวัตถุ

```
public class PersonInformation {
    private String name;
    protected String surname;
    private Date birthdate;
    public String occupation;
    public PersonInformation(String name, String surname, Date birthdate,String ocupation) {
        this.name = name;
        this.surname = surname;
        this.birthdate = birthdate;
        this.ocupation = ocupation;
    }
    public String getName() {
        return name;
    }
    public Date getBirthdate() {
        return birthdate;
    }
    public String getSurname() {
        return surname;
    }
    public Date getOccupation() {
        return ocupation;
    }
    public void setName(String name) {
        this.name=name;
    }
    public Date setBirthdate(Date birthdate) {
        this.birthdate=birthdate;
    }
    public String setSurname(String surname) {
        this.surname=surname;
    }
    public Date setOccupation(String ocupation) {
        this.ocupation=ocupation;
    }
}
```

}

ตัวอย่างที่ 4.10 แสดงตัวอย่างการประกาศคลาส PersonInformation โดยภายในคลาสมีการกำหนดการเข้าถึงแอตทริบิวต์ด้วยระดับต่าง ๆ เช่น

```
private String name;
protected String surname;
private Date birthdate;
public String occupation;
```

4.6.2 ตัวแปรของวัตถุและเมธอดของวัตถุ (Instance Variables and Methods)

ตัวแปรของวัตถุ (Instance variables) จะเป็นตัวแปรที่อยู่ภายในวัตถุ ส่วนเมธอดของวัตถุจะถูกเรียกโดยวัตถุของคลาสโดยทั้งตัวแปรวัตถุ และเมธอดของวัตถุจะต้องสร้างวัตถุขึ้นมาก่อนจึงจะสามารถใช้ได้

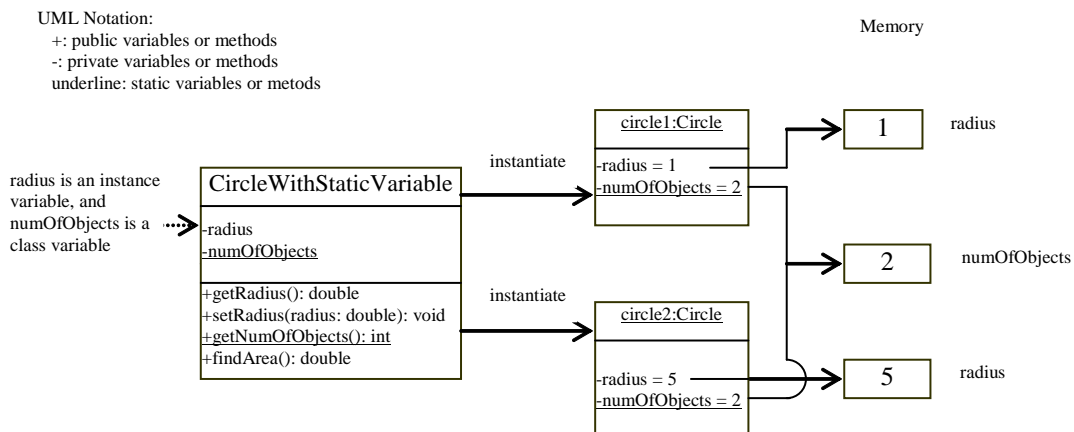
ตัวแปรของคลาส ค่าคงที่ และเมธอดของคลาส (Class Variables, Constants, and Methods)

Class variables จะถูกใช้งานได้โดย instance ของคลาสทุก ๆ instance เป็นแอตทริบิวต์ที่ไม่จำเป็นต้องสร้างอินสแตนซ์ของวัตถุขึ้นมาก่อน

Class methods ไม่ถูกผูกกับวัตถุใด ๆ

Class constants เป็น final variables ที่จะแชร์โดย instance ทุก ๆ instances ของคลาส

เราสามารถกำหนด class variables, constants และ methods ได้โดยการใช้คำ static



ตัวอย่างที่ 4.11 การใช้ Instance variables/Class Variables/Instance Method/Class Method

```

public class TestCircleWithStaticMembers {
    public static void main(String[] args) {
        System.out.println("Before creating objects");
        System.out.println("The number of Circle objects is " +
CircleWithStaticMembers.numberOfObjects);
        CircleWithStaticMembers c1 = new CircleWithStaticMembers();
        System.out.println("\nAfter creating c1");
        System.out.println("c1: radius (" + c1.radius + ") and number of Circle objects (" +
c1.numberOfObjects + ")");
        CircleWithStaticMembers c2 = new CircleWithStaticMembers(5);
        c1.radius = 9;
        System.out.println("\nAfter creating c2 and modifying c1");
        System.out.println("c1: radius (" + c1.radius + ") and number of Circle objects (" +
c1.numberOfObjects + ")");
        System.out.println("c2: radius (" + c2.radius + ") and number of Circle objects (" +
c2.numberOfObjects + ")");
    }
}

class CircleWithStaticMembers {
    double radius;
    static int numberOfObjects = 0;
    CircleWithStaticMembers() {
        radius = 1.0;
        numberOfObjects++;
    }
    CircleWithStaticMembers(double newRadius) {
        radius = newRadius;
        numberOfObjects++;
    }
    static int getNumberOfObjects() {
        return numberOfObjects;
    }
    double getArea() {
        return radius * radius * Math.PI;
    }
}

```

```

Before creating objects
The number of Circle objects is 0

After creating c1
c1: radius (1.0) and number of Circle objects (1)

After creating c2 and modifying c1
c1: radius (9.0) and number of Circle objects (2)
c2: radius (5.0) and number of Circle objects (2)

```

ตัวอย่างที่ 4.11 แสดงตัวอย่างการประกาศคลาส CircleWithStaticMembers โดยภายในคลาสมีการกำหนด Instance variables/Class Variables/Instance Method/Class Method เช่น

double radius เป็นการกำหนด Instance variables

static int numberOfObjects = 0; เป็นการกำหนด Class Variables

static int getNumberOfObjects() เป็นการกำหนด Class Method

double getArea() เป็นการกำหนด Instance Method

4.6.3 ขอบเขตของตัวแปร

ขอบเขตของ Instance variable และ class variable จะสามารถใช้ได้ทั้งคลาส และสามารถที่จะประกาศที่ตำแหน่งใด ๆ ก็ได้ภายในคลาส

ขอบเขตของตัวแปรที่เป็นแบบท้องถิ่น Local variable จะเริ่มจาก ณ ตำแหน่งที่ประกาศและต่อเนื่องไปจนกระทั่งสิ้นสุดบล็อกที่เก็บตัวแปรนั้น โดยตัวแปรแบบท้องถิ่นจะต้องประกาศก่อนถึงจะใช้งานได้

การใช้คีย์เวิร์ด this

ใช้สำหรับอ้างอิงวัตถุปัจจุบัน นอกจากนี้ยังสามารถใช้เรียก constructors ของวัตถุได้

4.6.4 Class Abstraction

Class abstraction หมายถึงการแยกส่วนของ class implementation ออกจาก use of the class คนที่สร้างคลาสจะให้รายละเอียดเกี่ยวกับ class ถึงวิธีการใช้งานคลาสเช่นสามารถใช้เมธอดใดได้บ้างและต้องส่งพารามิเตอร์ใดเข้าไปยังเมธอดนั้น ๆ โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ว่าการทำงานภายในมีการทำงานอย่างไรการทำงานภายในคลาสจะถูกปิด (encapsulated) และซ่อน (hidden) ไม่ให้ผู้ใช้เห็นรายละเอียดภายใน

ตัวอย่างที่ 4.12 ตัวอย่างการสร้างคลาสพนักงาน

```
class Employee{
    private String name;
    private String location;
    private double salary;
    public Employee(String name,String location, double salary){
        this.name=name;
        this.location=location;
        this.salary=salary;
    }
    public String getName(){
        return name;
    }
    public String getLocation(){
        return location;
    }
    public double getSalary(){
        return salary;
    }
    public void setName(String name){
        this.name=name;
    }
    public void setLocation(String location){
        this.location=location;
    }
    public void setSalary(double salary){
        this.salary=salary;
    }
    public void increaseEmpSalary(double percentage){
        salary = salary * (1+percentage);
    }
    public String toString(){
        return "Employee information: name : " +name+ "Location : "+location + "Salary : "+
        salary ;
    }
}
```

4.7 ตัวอย่างของ Package ของจาวา

ในการเขียนโปรแกรมภาษาจาวานอกจากผู้ใช้จะสามารถสร้างคลาสเพื่อทำงานตามที่ต้องการแล้วนั้น ผู้เขียนโปรแกรมยังสามารถเรียกใช้คลาสต่างๆ ที่อยู่ภายใน **Package** ของจาวาได้ยกตัวอย่างเช่น

ชื่อ Package	การใช้งาน
java.lang	จัดเตรียมคลาสที่เป็นพื้นฐานในการเขียนโปรแกรมจาวา คลาสที่สำคัญที่สุดคือคลาส Object ซึ่งเป็นคลาสแม่ของลำดับชั้นของคลาสต่างๆ
java.awt	มีคลาสสำหรับการเขียนกราฟิก
java.applet	มีคลาสสำหรับการเขียนแอปเพล็ต
java.io	มีคลาสสำหรับอินพุตและเอาต์พุตสตรีมไฟล์
java.util	ประกอบด้วยคลาสอรรถประโยชน์ต่าง ๆ เช่น วันที่
java.net	ประกอบด้วยคลาสสำหรับรองรับการสื่อสารบนเครือข่าย
java.awt.image	มีคลาสสำหรับการจัดการภาพบิตแมป
java.awt.peer	การใช้ GUI เฉพาะแพลตฟอร์ม
java.sql	จัดเตรียม API สำหรับการเข้าถึงและประมวลผลข้อมูลที่จัดเก็บในฐานข้อมูล
java.rmi	Java Remote Method Invocation (Java RMI) เป็น Java API ที่ดำเนินการเรียกใช้เมธอดรีโมต ซึ่งเทียบเท่ากับการเรียกโพรซีเจอร์ระยะไกล (RPC)

สามารถเข้าไปดูรายละเอียดของเมธอดต่าง ๆ ของภาษาจาวาได้ที่

<http://docs.oracle.com/javase/7/docs/api/index.html>

<http://docs.oracle.com/javase/tutorial/index.html>

4.8 บทสรุป

คลาส คือ แม่แบบของสิ่งต่างๆ ที่เราสนใจโดยภายในคลาสมีคุณลักษณะและพฤติกรรมในการเขียนโปรแกรมเชิงวัตถุจะใช้คลาสเป็นต้นแบบสำหรับสร้างวัตถุ ตัวอย่างเช่น หากต้องการโปรแกรมสำหรับเก็บรายละเอียดของนักเรียนแต่ละคน การเขียนโปรแกรมเชิงวัตถุจะมองว่ากลุ่มของนักเรียนสามารถที่จะกำหนดเป็นคลาสได้ โดยภายในประกอบด้วยแอตทริบิวต์และเมธอด โดยส่วนของแอตทริบิวต์ของนักเรียนหรือคุณลักษณะของนักเรียนประกอบด้วยรหัสประจำตัว ชื่อ นามสกุล คณะ เกรดเฉลี่ย อายุ วันเดือนปีเกิด ในส่วนพฤติกรรมสามารถกำหนดได้เป็น การเดิน การวิ่ง การกิน การลงทะเบียนเรียน เป็นต้น ซึ่งหากผู้เขียนโปรแกรมต้องการเก็บข้อมูลของนักเรียนแต่ละคนสามารถเก็บข้อมูลของนักเรียนแต่ละคนผ่านการเขียนโปรแกรมโดยการประกาศวัตถุ เพื่อเก็บข้อมูลดังกล่าว

ในการประกาศคลาสสำหรับการใช้งานจะประกอบด้วยกลุ่มของแอตทริบิวต์ (Attribute) หรือ Instance variable และกลุ่มของพฤติกรรม (method) จากนั้นตามขั้นตอนหากต้องการสร้างคลาสที่เก็บข้อมูลของนักเรียนจะสามารถทำได้โดยกำหนดส่วนของแอตทริบิวต์และเมธอด โดยส่วนของแอตทริบิวต์ที่ควรมีสำหรับการนิยามในคลาสนักเรียนเช่น รหัสประจำตัว ชื่อ นามสกุล คณะ เกรดเฉลี่ย อายุ วันเดือนปีเกิด ในส่วนพฤติกรรมสามารถกำหนดได้เป็น การเดิน การวิ่ง การกิน การลงทะเบียนเรียน

คลาสของภาษาจาวามีสมาชิก(Class member) ได้ 3 ประเภทดังนี้

กลุ่มของแอทริบิวต์ (data member) หรือ variable สำหรับเก็บค่าสถานะ (state) ของ instance ของคลาสนั้น แอทริบิวต์หรือ data member ประกอบด้วย instance variable และ class variable

กลุ่มของพฤติกรรม (method member) คือฟังก์ชันซึ่งเป็นพฤติกรรมของ instance ของคลาสนั้น เมธอด (method) เป็นชุดคำสั่งย่อยที่ทำงานกับข้อมูล ตัวเมธอดอาจจะมีหรือไม่มีพารามิเตอร์ก็ได้ โดยที่เมธอดอาจจะมีการคืนค่าออกมาหลังจากที่ทำงานเสร็จ ประเภทของเมธอด ประกอบด้วย

a. Constructor คอนสตรัคเตอร์เป็นเมธอดชนิดหนึ่งที่มีชื่อเดียวกับชื่อคลาส ถูกเรียกเพื่อสร้างวัตถุและกำหนดค่าเริ่มต้นให้กับวัตถุ คอนสตรัคเตอร์ที่ไม่มีพารามิเตอร์จะเรียกว่า Default constructor คอนสตรัคเตอร์จะต้องมีชื่อเดียวกับชื่อคลาส การสร้างคอนสตรัคเตอร์จะไม่มีส่วนของการคืนค่ากลับ หน้าที่ของคอนสตรัคเตอร์คือการกำหนดค่าเริ่มต้นให้กับวัตถุ

b. Mutator methods (setter) ทำหน้าที่เปลี่ยนข้อมูลภายในวัตถุ

c. Accessor methods (getter) ทำหน้าที่ดึงข้อมูลจากวัตถุ

d. เมธอดอื่น ๆ

กลุ่มของคลาส (Class member) คือคลาสที่ถูกกำหนดขึ้นภายในคลาสนั้น ซึ่งอาจจะถูกกำหนดไว้เพื่อใช้งานภายในคลาสนั้นเอง หรือจากคลาสมานนอกคลาสนี้ก็ได้

4.9 แบบฝึกหัดปฏิบัติการ

1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสร้างคลาสจากตัวอย่างต่อไปนี้

ต้องการสร้างคลาสของเลขเชิงซ้อนที่ประกอบไปด้วยส่วนจริง(r) และส่วนจินตภาพ (i) เช่น $5+3i$

- โดยตัวเลขเชิงซ้อน 1 ตัวจะถือว่าเป็น 1 Object
- การทำงานของตัวเลขเชิงซ้อนที่เราจะสามารถเรียกเพื่อให้งานทำได้คือ
 - การบวก (add) โดยการส่งตัวเลขเชิงซ้อนอีก ตัวเข้ามาบวก 1
 - และการแสดงผลตัวเลข (print) โดยนำส่วนจริงและส่วนจินตภาพมาแสดงผลในรูปแบบ $r+ci$

เมื่อต้องการสร้างตัวเลขเชิงซ้อนขึ้นมา 1 ตัวจะต้องระบุรูปแบบของตัวเลขที่จะสร้าง(Type/class) และตามด้วยตัวแปรอ้างอิงวัตถุ (ตัวแปร) ที่จะเก็บที่อยู่ของวัตถุจริง ๆ และตามด้วยการสร้างวัตถุในหน่วยความจำที่ประกอบด้วยแอมริบิต์และเมธอดตามที่กำหนดไว้ในคลาส โดยการใช้คำสั่ง new และตามด้วย Constructor ที่ทำหน้าที่กำหนดค่าเริ่มต้นให้วัตถุ เช่น

```
Complex a = new Complex(1.0, 2.0);
```

เมื่อสร้างวัตถุด้วยคำสั่งดังกล่าวจะมีรูปแบบการเก็บข้อมูลในหน่วยความจำคือ

```
// ComplexTest.java
class Complex {
    private double r, i;
    Complex(double r, double i) {
        this.r = r; this.i = i;
    }
    Complex(Complex c) {
        this(c.r, c.i);
    }
    public void add(Complex c) {
        r += c.r;
        i += c.i;
    }
    public void print() {
        System.out.println(r + "+ i" + i);
    }
}
class ComplexTest {
    public static void main(String args[]) {
        Complex a = new Complex(1.0, 2.0);
        Complex b = new Complex(3.0, 4.0);
        Complex c = new Complex(a);
        c.add(b);
        c.print();
    }
}
```

จากโปรแกรมดังกล่าวให้เพิ่มความสามารถของ Object ให้สามารถลบ คูณ และหารได้

2. จงหาข้อผิดพลาดของส่วนของโปรแกรมต่อไปนี้ โดยวงกลมตำแหน่งที่ผิด เขียนหมายเลขกำกับแต่ละตำแหน่งและอธิบายเหตุผลด้านล่างว่าเหตุใดตำแหน่งดังกล่าวจึงผิด ถ้าไม่มีข้อผิดพลาดให้ตอบว่าไม่มีข้อผิดพลาด

ส่วนของโปรแกรม	ผลลัพธ์
<pre>public class ShowErrors{ public static void main(String[] args){ ShowErrors t= new ShowErrors(5); } }</pre>	
<pre>public class ShowErrors{ public static void main(String[] args){ ShowErrors t= new ShowErrors(); t.x(); } }</pre>	
<pre>public class ShowErrors{ public void method1(){ Circle c; System.out.println("What is radius "+ c.getRadius()); c=new Circle; } }</pre>	
<pre>public class ShowErrors{ public static void main(String[] args){ C c = new C(5.0); System.out.println(c.value); } } class C{ int value=2; }</pre>	

3. จงอธิบายการทำงานของโปรแกรมต่อไปนี้

<pre>public class Test{ public static void main(String[] args){ Count myCount = new Count(); int times=0; for (int i=0; i<100 ;i++) increment(myCount, times); System.out.println("count is "+myCount.count); System.out.println("times is "+times); } public static void increment(Count c, int times){ c.count++; times++; } }</pre>
<pre>public class Count{ public int count; public Count(int c){ count =c; } public Count(){ count =1; } }</pre>
แสดงผลลัพธ์จากโปรแกรม

3.1 ข้อแตกต่างของการส่งค่าพารามิเตอร์ของ Primitive type และการส่งค่าพารามิเตอร์ของ reference type จากโปรแกรมต่อไปนี้

3.2 ระบุจุดที่มีการส่งค่าพารามิเตอร์ของ Primitive type และการส่งค่าพารามิเตอร์ของ reference type

4. จงแสดง output จากโปรแกรมต่อไปนี้พร้อมอธิบายการทำงานในแต่ละบรรทัด

ส่วนของโปรแกรม	ผลลัพธ์
<pre>public class Test { public static void main(String[] args) { int[] a = {1, 2}; swap(a[0], a[1]); System.out.println("a[0] = " + a[0] + " a[1] = " + a[1]); } public static void swap(int n1, int n2) { int temp = n1; n1 = n2; n2 = temp; } }</pre>	
<pre>public class Test { public static void main(String[] args) { T t1 = new T(); T t2 = new T(); System.out.println("t1's i = " + t1.i + " and j = " + t1.j); System.out.println("t2's i = " + t2.i + " and j = " + t2.j); } class T { static int i = 0; int j = 0; T() { i++; j = 1; } } }</pre>	

(ฝึกเขียนโปรแกรมตาม Concept OOP)

5. [Algebra: solve 2x2 equation] ให้สร้างคลาส [LinearEquation](#) สำหรับการแก้สมการระบบสมการเชิงเส้น ตัว 2 แปรได้ตามสมการต่อไปนี้

$$\begin{matrix} ax + by = e \\ cx + dy = f \end{matrix} \quad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$

โดยภายในคลาสประกอบด้วยส่วนประกอบต่อไปนี้

- ตัวแปรประเภท Private [a](#), [b](#), [c](#), [d](#), [e](#), และ [f](#).
- Constructor ที่มีการรับค่า arguments สำหรับ [a](#), [b](#), [c](#), [d](#), [e](#), and [f](#).
- [get](#) methods สำหรับตัวแปร [a](#), [b](#), [c](#), [d](#), [e](#), and [f](#).
- method ชื่อ [isSolvable\(\)](#) ที่คืนค่า true ถ้า $ad - bc$ ไม่ใช่ 0.

- method ชื่อ `getX()` and `getY()` ที่คืนค่าผลลัพธ์ของสมการ

หลังจากนั้นให้เขียนโปรแกรมเพื่อทดสอบโดย รับค่า a b c d e f จากผู้ใช้หลังจากนั้นคำนวณผลลัพธ์
ตัวอย่าง

Enter a b c d e f: 21.0- 6.0- 5.0- 3.0 4.0 9.0

x is -2.0 and y is 3.0

ตัวอย่าง

Enter a b c d e f: 1 0.2 0.2 0.45.0 4.0 0.

The equation has no solution

6. (APPLICATION) จงเขียนโปรแกรมเพื่อแสดงรายละเอียดของคลาส Account ที่ประกอบด้วยสมาชิกต่อไปนี้

- ตัวแปร private ชนิดข้อมูล int ชื่อ id สำหรับเก็บหมายเลขบัญชี
- ตัวแปร private ชนิดข้อมูล double ชื่อ balance สำหรับเก็บยอดเงินคงเหลือ
- ตัวแปร private ชนิดข้อมูล double ชื่อ annualInterestRate สำหรับเก็บอัตราดอกเบี้ย
- ตัวแปร private ชนิดข้อมูล Date ชื่อ dateCreated สำหรับเก็บวันที่ที่บัญชีถูกสร้าง
- constructor ที่ไม่มี argument สำหรับการสร้างบัญชีแบบ default
- constructor ที่มี argument สำหรับการสร้างบัญชีแบบระบุเลขที่บัญชี และยอดเงินเริ่มต้น
- accessor method(get) และ mutator method(set) สำหรับตัวแปร id, balance, annualInterestRate, dateCreated
- เมธอด getMonthlyInterestRate() ที่คืนอัตราดอกเบี้ยรายเดือน
- เมธอด getMonthlyInterest() ที่คืนดอกเบี้ยรายเดือน
- เมธอด withdraw() ที่ถอนเงินตามจำนวนที่ระบุ
- เมธอด deposit() ที่ฝากเงินตามจำนวนที่ระบุ

วาดคลาสไดอะแกรมและเขียนส่วนของ Client สำหรับเรียกใช้คลาส Account โดยสร้างอ็อบเจกต์ของบัญชีเลขที่ (ID) 1122 ยอดเงินเปิดบัญชีคือ 20000 และอัตราดอกเบี้ยคือ 4.5 % หลังจากนั้นให้ใช้ withdraw method สำหรับถอนเงิน 2500 บาท และใช้ deposit method สำหรับฝากเงิน 3000 บาท และโปรแกรมสามารถแสดงยอดเงินคงเหลือและอัตราดอกเบี้ยรายเดือนได้

7. เขียนโปรแกรมที่รับค่าจากผู้ใช้ที่ประกอบด้วยพิกัดของจุดศูนย์กลาง x y ความกว้าง ความสูงของสี่เหลี่ยมและเช็คสี่เหลี่ยมรูปที่สองอยู่ในสี่เหลี่ยมรูปแรก หรือซ้อนทับกับรูปแรก ดังแสดงในภาพ

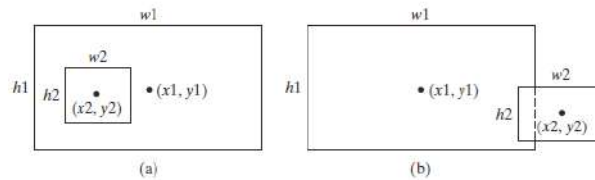


FIGURE 3.11 (a) A rectangle is inside another one. (b) A rectangle overlaps another one.

Here are the sample runs:

```
Enter r1's center x-, y-coordinates, width, and height: 2.5 4 2.5 43
Enter r2's center x-, y-coordinates, width, and height: 1.5 5 0.5 3
r2 is inside r1
```

8. [Maximum Prime Number] จงเขียนโปรแกรมแบบ OOP โดยมีคลาส เมธอด และแอทริบิวต์ ที่อ่านตัวเลขจำนวนจริง หาตัวเลขที่เป็นเลขจำนวนเฉพาะและมีค่ามากที่สุดจากกลุ่มของตัวเลขดังกล่าว โดยให้ตัวเลข Input จบด้วยเลข 0 ตัวอย่างเช่น ถ้าป้อนตัวเลขต่อไปนี้ 3 5 2 5 5 0 โปรแกรมจะหาตัวเลขที่เป็นเลขจำนวนเฉพาะและมีค่ามากที่สุดคือ 5 ถ้าหากไม่มีข้อมูลที่เป็นจำนวนเฉพาะในข้อมูลที่ให้มาเลยให้แสดงค่า -1 เช่น

ข้อมูลนำเข้า รับข้อมูลเลขจำนวนเต็มบวก n ตัว

ข้อมูลส่งออก ตัวเลขที่มีค่ามากที่สุดและจำนวนครั้งของการปรากฏ

ข้อมูลนำเข้า	ข้อมูลส่งออก
3 5 2 5 5 0	5
0 111 59 53 24 9 8 4 2 103	103
4 8 6 10 12 14 0	-1

บทที่ 5 เมธอด

วัตถุประสงค์

- 5.1 สามารถอธิบายความหมายของเมธอด ชนิดของเมธอด
- 5.2 สามารถเขียนโปรแกรมเพื่อผ่านค่าข้อมูลไปให้เมธอด
- 5.3 สามารถอธิบายความหมายและเขียนโปรแกรมเพื่อกำหนดขอบเขตของตัวแปร
- 5.4 สามารถอธิบายความหมายของการโอเวอร์โหลดเมธอด
- 5.5 สามารถอธิบายและเขียนโปรแกรมแบบ *Recursive Method*

5.1 ความนำ

เมธอด (Method) คือ กลุ่มคำสั่งที่ถูกกำหนดขึ้นเพื่อให้ได้ผลลัพธ์อย่างใดอย่างหนึ่ง เป็นส่วนประกอบหนึ่งของคลาสเปรียบเสมือนกับฟังก์ชัน (function) ในการพัฒนาโปรแกรมแบบโครงสร้าง (structured programming)

5.2 การประกาศเมธอด

รูปแบบการประกาศเมธอดมีดังนี้

```
access_modifier return_type methodName ( parameter_list ) {
// method body
return type;
}
```

access_modifier เป็นตัวกำหนดคุณสมบัติในการเรียกใช้เมธอด ทำให้ผู้พัฒนาโปรแกรมสามารถที่จะควบคุมได้ว่าเมธอดนั้นๆ จะสามารถถูกเรียกใช้จากคลาสอื่นได้หรือไม่ ซึ่งจัดเป็นคุณสมบัติหนึ่งของหลักการโปรแกรมเชิงวัตถุ นั่นคือคุณสมบัติการซ่อนข้อมูล (encapsulation) ค่าของ access_modifier สำหรับเมธอดมีทั้งหมด 7 ค่า ดังแสดงด้านล่าง

ตารางแสดง access_modifier สำหรับเมธอด

Access_modifier	การเรียกใช้งาน
static	สามารถเรียกใช้งานโดยไม่จำเป็นต้องสร้างวัตถุขึ้นมาก่อน
public	สามารถเรียกประมวลผลโดยไม่มีข้อจำกัด
private	เรียกประมวลผลได้เฉพาะภายในคลาสที่ประกาศเมธอดนี้เท่านั้น
<blank> -default	สามารถเรียกประมวลผลโดยคลาสอื่นได้ แต่คลาสนั้นๆ จะต้องอยู่ภายใน package เดียวกับคลาสที่ประกาศเมธอด
abstract	เป็นเมธอดที่ยังไม่ได้เขียนขั้นตอนการทำงาน จะมีเพียงชื่อของเมธอด เท่านั้น ผู้ใช้จะต้องกำหนดขั้นตอนการทำงานต่างๆ ให้กับเมธอดนี้เอง
final	เป็นเมธอดที่ไม่อนุญาตให้มีการประกาศซ้ำ หรือ overload ได้
synchronized	เป็นเมธอดที่มีการประมวลผลที่ข้อมูลชุดเดียวกันกับเมธอดอื่น หรือเป็น เธรด (thread) มีการประมวลผลพร้อมกัน เธรดคือระบบของจาวาสำหรับการสนับสนุนการทำงานแบบ multi-tasking ที่ให้โปรแกรมสามารถทำงานพร้อมกันได้

return_type เป็นประเภทของตัวแปรที่ต้องการส่งค่าคืน หากมีการกำหนด return_type ภายในเมธอดต้องมีคำสั่ง return ค่ากลับด้วย

parameter_list คือชุดของตัวแปรที่ใช้ในการส่งค่ามายังเมธอดดังกล่าว parameter_list นี้สามารถใช้ในการส่งค่าคืนได้ด้วย แต่ทั้งนี้ขึ้นอยู่กับประเภทของตัวแปร ถ้าเป็นตัวแปรพื้นฐานแล้วเมธอดจะไม่สามารถเปลี่ยนแปลงค่าตัวแปร หรือส่งคืนค่ากลับผ่านทาง parameter_list นี้ได้

5.3 ชนิดของเมธอด

เมธอดในภาษา Java มี 2 ประเภท ดังนี้

1. เมธอดที่เป็นสมาชิกของคลาส (class method) เมธอดประเภทนี้จะมีเพียงชุดเดียวเท่านั้น และไม่จำเป็นต้องสร้างวัตถุเพื่อเรียกใช้เมธอดกล่าวคือ เมธอดนี้สามารถเรียกใช้ผ่านชื่อคลาส ได้ access_modifier ที่ทำให้ เมธอดเป็นสมาชิกของคลาสคือ static เมธอดประเภทนี้ เหมาะสำหรับเมธอดที่ไม่มีการเปลี่ยนแปลงค่าไปตามวัตถุ เช่น สูตรในการคำนวณทางคณิตศาสตร์ ซึ่งเมธอดดังกล่าวจะรับค่าที่ผู้ใช้ต้องการคำนวณ และทำการคำนวณแล้วส่งผลลัพธ์กลับไปยังผู้ใช้

2. เมธอดที่เป็นสมาชิกของอินสแตนซ์ (instance method) เมธอดที่เป็นสมาชิกของ instance จะต้องเกิดจากตัวดำเนินการ new กล่าวคือ เมื่อจะเรียกใช้เมธอดประเภทนี้ จะต้องมีการสร้างวัตถุขึ้นมาก่อน และค่าของข้อมูลจะเปลี่ยนไปตามวัตถุที่สร้างขึ้น

ตัวอย่างที่ 5.1 ตัวอย่าง class method

```
class X {
    public static int methodA(int a) {
        a += 1;
        return a;
    }
}
class TestClassMember {
    public static void main (String args[] ) {
        int num = 0;
        num = num + X.methodA(num);
        System.out.println("num="+num);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
num=1
```

ตัวอย่างที่ 5.2 ตัวอย่าง instance method

```
class X {
    int methodA(int a) {
        a += 1;
        return a;
    }
}
class TestClassMember1 {
    public static void main (String args[]) {
        int num = 0;
        X numX = new X();
        num = num + numX.methodA(num);
        System.out.println("num="+num);
    }
}
```


ผลลัพธ์ที่เกิดขึ้นคือ

num=1

โดยสรุป เมธอดที่เป็น class method นั้นต้องมี modifier แบบ static และสามารถเรียกประมวลผลผ่านชื่อของคลาส เช่น X.methodA(); ในส่วนของ instance method นั้นไม่ต้องมี modifier เป็น static แต่ก่อนที่จะเรียกใช้งานจะต้องมีการสร้างวัตถุขึ้นมาก่อน เช่น X numX = new X(); และการเรียกประมวลผลต้องเรียกผ่านชื่อของวัตถุนั้น เช่น numX.methodA(); การเรียกใช้ class method ที่อยู่ภายในคลาสเดียวกันสามารถที่จะเรียกใช้ผ่านชื่อเมธอดนั้นได้

ผลลัพธ์ที่เกิดขึ้นคือ

num=1

ตัวอย่างที่ 5.3 ตัวอย่าง class method

```
public class Demonstrate {
    public static void main (String args[]) {
        int script = 6, acting=9, directing=8;
        System.out.print("The rating of the movie is ");
        System.out.println(movieRating(script, acting, directing));
    }
    public static int movieRating(int s, int a, int d) {
        return s+a+d;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

The rating of the movie is 23

5.4 การผ่านค่าข้อมูลไปให้เมธอด

รูปแบบของการเขียนโปรแกรมในการรับส่งค่าตัวแปร มีรูปแบบดังนี้

```
access_modifier return_type MethodName(parameter_list) {
    // Method body
    return type;
}
```

ในรูปแบบที่กำหนดเราสามารถละ modifier และ return_type และไม่จำเป็นต้องมี parameter_list ถ้ามีการส่งค่ากลับในส่วนของเมธอดต้องมีคำสั่งในการส่งค่าคืน เช่น return type; โดยสรุปจะมีรูปแบบวิธีในการเขียนแต่ละรูปแบบ และวิธีการใช้งาน ดังนี้

1. เมธอดที่ไม่มีการรับส่งค่าตัวแปรใดๆ

โดยปกติจะเป็นการเรียกเมธอดของคลาสหรือวัตถุที่เมธอดดังกล่าวจะทำการประมวลผล นั่นคือ ไม่มีการเปลี่ยนค่าของตัวแปรของคลาสถ้าไม่มีการส่งค่าคืน ต้องระบุ void ในส่วนของ return_type และถ้าไม่มีการรับค่า ส่วนของ parameter_list ไม่ต้องมีการระบุ

2. เมธอดที่ส่งค่าตัวแปร

จำนวนค่าที่จะทำการส่งผ่านไปยังเมธอดใดๆ จะต้องมีค่าเท่ากับจำนวนของตัวแปรที่อยู่ใน parameter_list ตำแหน่งของค่าจะสอดคล้องกับตำแหน่งของตัวแปรใน parameter_list ด้วย

3. เมธอดที่มีการส่งค่าคืน

ในการส่งค่าคืน ต้องมีการระบุประเภทของข้อมูลที่ต้องการส่งคืน การส่งค่าคืนที่เป็นค่าพื้นฐานจะสามารถส่งคืนได้ครั้งละ 1 ตัวแปรเท่านั้น ถ้าต้องการส่งคืนมากกว่า 1 ตัวแปร ต้องอยู่ในรูปของคลาส สำหรับการผ่านค่าของข้อมูลไปให้เมธอดสามารถทำได้แบบเดียวกับภาษาคอมพิวเตอร์ทั่วไปคือ ผ่านค่าโดยส่งเฉพาะข้อมูล (pass-by-value) และผ่านค่าแบบอ้างอิง (pass-by-reference) หรือส่งไปเฉพาะที่อยู่ของข้อมูล ดังที่ได้กล่าวมาแล้วว่าภาษา Java กำหนดให้ข้อมูลพื้นฐาน (primitive data type) ผ่านค่าข้อมูลแบบส่งเฉพาะข้อมูลเท่านั้น (pass-by-value) ซึ่งหมายความว่าตัวแปรที่ประกาศชนิดของข้อมูลแบบพื้นฐานเมธอดที่รับค่าข้อมูลจากตัวแปรดังกล่าว จะไม่สามารถเปลี่ยนแปลงข้อมูลได้เลย ถ้าต้องการเปลี่ยนแปลงค่าของข้อมูลจะต้องผ่านค่าแบบอ้างอิงเท่านั้น ซึ่งการผ่านค่าแบบอ้างอิง จะใช้ได้กับทุกประเภทที่เป็นแบบคลาสเท่านั้น

ตัวอย่างที่ 5.4 โปรแกรมผ่านค่าแบบ pass-by-value

```
class A {
    int x = 10, y = 5;
    void methodA (int a, int b) {
        a = x;
        b = y;
    }
}
class TestPassByValue {
    public static void main (String args[]) {
        int var1, var2;
        A objA = new A();
        var1 = 5;
        var2 = 5;
        System.out.println("var1(before) = " + var1);
        System.out.println("var2(before) = " + var2);
        objA.methodA(var1, var2);
        System.out.println("var1(after) = " + var1);
        System.out.println("var2(after) = " + var2);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
var1(before) = 5
var2(before) = 5
var1(after) = 5
var2(after) = 5
```

ตัวอย่างที่ 5.5 โปรแกรมผ่านค่าแบบ Pass-by-reference

```
class A {
    int x = 10, y = 10;
    void methodA (B objB) {
        objB.var1 = x;
        objB.var2 = y;
    }
}
class B {
    int var1, var2;
}
class TestPassByValue1 {
    public static void main(String args[]) {
        B objB = new B();
        A objA = new A();
        objB.var1 = 5;
        objB.var2 = 5;
        System.out.println("var1(before)="+objB.var1);
        System.out.println("var2(before)="+objB.var2);
        objA.methodA(objB);
    }
}
```

```

        System.out.println("var1 (after)="+objB.var1);
        System.out.println("var2 (after)="+objB.var2);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

var1 (before)=5
var2 (before)=5
var1 (after)=10
var2 (after)=10

```

จากตัวอย่างโปรแกรมที่ผ่านมา ในการผ่านค่าแบบ Pass-by-value จะเห็นว่า method A ใน class A มีการส่งค่าข้อมูลผ่านตัวแปร var1 และ var2 ไปยังตัวแปร a และ b ซึ่งต้องการเปลี่ยนค่าข้อมูลโดยกำหนดให้เท่ากับ x และ y ของ class A แต่จะเห็นว่า เมื่อส่งค่าคืน ตัวแปร var1 และ var2 ยังมีค่าเท่าเดิม ถ้าต้องการเปลี่ยนค่าของ var1 และ var2 ให้เป็นค่าของ x และ y ต้องให้ตัวแปรที่ผ่านค่าของ methodA เป็นแบบ object ดังตัวอย่างการผ่านค่าแบบ pass-by-reference ซึ่งในโปรแกรมมีการเพิ่ม class B เพื่อเก็บตัวแปร var1 และ var2 และต้องสร้าง object เพื่อเก็บข้อมูลเบื้องต้น และเมื่อส่งตัวแปรไปที่ methodA จะเห็นว่าในครั้งนี้โปรแกรมเปลี่ยนค่าให้ตามที่ต้องการ ()

5.5 ขอบเขตของตัวแปร

ในภาษา Java สามารถตั้งชื่อ class method และตัวแปรซ้ำกันได้ กฎของขอบเขตของตัวแปรของคลาสสามารถสรุปได้ดังนี้

1. ตัวแปร ไม่จำเป็นต้องประกาศต่อจากการประกาศคลาสหรือเมธอดสามารถประกาศเมื่อใดก็ได้ที่ต้องการใช้งาน
2. สามารถประกาศใช้ตัวแปรซ้ำกันได้ ถ้าไม่อยู่ใน block เดียวกัน หรือ block ที่ซ้อนกัน
3. ตัวแปรจะถูกเรียกใช้ (live) อยู่ภายใน block เท่านั้น ถ้าออกนอก block ถือว่าหน่วยความจำสำหรับตัวแปรตัวนี้ ได้ยกเลิกและคืนให้กับระบบ
4. กรณีที่ตัวแปรซ้ำกัน ถ้าไม่มีการระบุที่ชัดเจน จะเป็นการเรียกตัวแปรภายใน block นั้นมาใช้งาน
การใช้คำสั่ง this

คำสั่ง this ใช้ในการเรียกวัตถุที่กำลังประมวลผลอยู่ การใช้คำสั่งดังกล่าวมีวัตถุประสงค์ เพื่ออ้างอิงตัวแปรที่ซ้ำกันใน block ของคำสั่ง เช่น เมธอดมีการรับตัวแปรที่ชื่อซ้ำกับตัวแปรของวัตถุ เพื่อลดความสับสนการใช้ตัวแปรชื่อซ้ำจึงใช้คำสั่ง this ในการจำแนกว่าเป็นตัวแปรของวัตถุ

ตัวอย่างที่ 5.6 การใช้คำสั่ง this

```

class First {
    int r = 5;
    { // class block
        int r = 6;
        System.out.println ("First->Class Block-> r =" + r);
        System.out.println ("First->Class Block->this.r = " + this.r);
    }
    void methodA () {
        int r = 10;
        int s = 20;
        System.out.println("First->methodA->r =" + r);
        System.out.println("First->methodA->this.r =" + this.r);
        { // Block 1
            //int r = 20;

```

```

        int t = 5;
        System.out.println("First->methodA->Block1->t" + t);
        System.out.println("First->methodA->Block1->r" + r);
    }
    { // Block 2
        int t = 8;
        System.out.println("First->methodA->Block2->t" + t);
    }
}

void methodB() {
    int t = 30;
    System.out.println("First->methodB-> t is " + t);
    //System.out.println("First->methodB->s is " + s);
}
}

class TestScope {
    public static void main (String args[]) {
        First var1 = new First();
        var1.methodA();
        var1.methodB();
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

First->Class Block-> r =6
First->Class Block->this.r = 5
First->methodA->r =10
First->methodA->this.r =5
First->methodA->Block1->t5
First->methodA->Block1->r10
First->methodA->Block2->t8
First->methodB-> t is 30

```

5.6 การโอเวอร์โหลดเมธอด

การโอเวอร์โหลดเมธอด (Overloading) คือ การเขียนชุดของเมธอดที่มีชื่อเหมือนกัน แต่มีพารามิเตอร์ของเมธอดที่แตกต่างกัน ซึ่งถือว่าเป็นคุณลักษณะพิเศษหนึ่งของการเขียนโปรแกรมเชิงวัตถุ คือ polymorphism ตัวอย่างต่อไปนี้ เป็นการแสดงโปรแกรมที่ไม่มีการ overload และมีการ overload

No-overloading	Overloading
<pre> ... if type = 1 DrawTriangle (a, b, c); else DrawSquare (a, b, c, d); ... function DrawTriangle(int a, int b, int c) {...} function DrawSquare(int a, int b, int c, int d) {...} </pre>	<pre> class MainTask { Draw(a,b,c) // draw triangle Draw(a,b,c,d) // draw square ... } class Drawable { void Draw(int a, int b, int c) {...} void Draw(int a, int b, int c, int d) {...} } </pre>

การเขียนโปรแกรมที่ตัวแปรภาษามีคุณสมบัติของ Overloading จะช่วยลดเวลาในการพัฒนาโปรแกรมเนื่องจากไม่จำเป็นต้องแก้ไขโปรแกรมที่เคยพัฒนาไว้ หากแต่สามารถเพิ่มฟังก์ชันต่างๆ ได้ แต่เมธอดที่จะเพิ่มต้องมีรูปแบบที่แตกต่างกันซึ่งตัวแปรภาษาจะพิจารณาจากรูปแบบของตัวแปรและใช้เมธอดที่มีตัวแปรสอดคล้องกัน ข้อควรระวังในการเรียก method overloading คือ ความคลุมเครือในการเรียกเมธอดมาใช้งาน

ตัวอย่างที่ 5.7 การเขียนโปรแกรมแบบ Method overloading

```
public class DemoOverloading {
// add 1
    static int add (int x, int y) {
        System.out.print("add1 int, int " + x + "+" + y + "=");
        return (x+y);
    }
// add 2
    static int add (int x, float y) {
        System.out.print("add2 int, float" + x + "+" + y + "=");
        return (x + (int)y);
    }
// add 3
    static int add (float x, int y) {
        System.out.print("add3 float, int " + x + "+" + y + "=");
        return ((int)x+y);
    }
    public static void main (String args[]) {
        System.out.println(add(5,9));
        System.out.println(add(5,2.8F));
        System.out.println(add(2.8F, 5));
        //System.out.println(add(2.8F, 5.2F)); // This line is ambiguous
        System.out.println(add(2.8F, (int)5.2));
        System.out.println(add((int)2.8F,5));
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
add1 int, int 5+9=14
add2 int, float5+2.8=7
add3 float, int 2.8+5=7
add3 float, int 2.8+5=7
add1 int, int 2+5=7
```

ตัวอย่างข้างต้น จะมีบรรทัดหนึ่งที่เขียน comment ไว้ว่า This line is ambiguous เนื่องจากว่า โปรแกรมไม่สามารถหาเมธอดที่มี parameter list สอดคล้องกับการเรียกเมธอดนี้ กล่าวคือ ไม่มีการประกาศเมธอดที่มีค่า parameter list เป็นชนิด float ทั้ง 2 ตัวแปร ดังนั้นหลังจากการ compile โปรแกรมนี้จะพบการแจ้ง error จากบรรทัดนั้นนั่นเอง

5.7 เมธอดของคลาส Math

ในการเขียนโปรแกรมที่มีการคำนวณทางคณิตศาสตร์ ภาษา Java ได้จัดเตรียมเมธอดสำหรับงานดังกล่าวไว้ที่คลาสที่ชื่อว่า Math ซึ่งสามารถสรุปเมธอดที่ใช้กันเป็นส่วนมากไว้ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 5.8 การเขียนโปรแกรมที่มีการคำนวณทางคณิตศาสตร์

```
public class Math_method {
    public static void main (String args[]) {
        System.out.println( "Natural logarithm of 10 : "+Math.log(10) );
        System.out.println("Exponential of 1.0 : " + Math.exp(1.0) );
        System.out.println("The third power of 2 : " + Math.pow(2, 3) );
        System.out.println("Square root of 7 : " + Math.sqrt(7) );
        System.out.println( "Absolute value of -10 : "+Math.abs(-10) );
        System.out.println("Ceiling value of 9.2 : " + Math.ceil(9.2) );
        System.out.println("Floor value of 9.2 : " + Math.floor(9.2) );
        System.out.println( "Maximum of 2 and 3 : " + Math.max(2,3) );
        System.out.println("Minimum of 2 and 3 : " + Math.min(2,3) );
        System.out.println("Sine of 0 radian : " + Math.sin(0.0) );
        System.out.println("Cosine of 0 radian : " + Math.cos(0.0) );
        System.out.println("Tangent of 0 radian : " + Math.tan(0.0) );
        System.out.println("Pi : " + Math.PI );
        System.out.println("Random number (0.0 to 1.0) : " + Math.random() );
    }
}
```

```

    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

Natural logarithm of 10 : 2.302585092994046
Exponential of 1.0 : 2.718281828459045
The third power of 2 : 8.0
Square root of 7 : 2.6457513110645907
Absolute value of -10 : 10
Ceiling value of 9.2 : 10.0
Floor value of 9.2 : 9.0
Maximum of 2 and 3 : 3
Minimum of 2 and 3 : 2
Sine of 0 radian : 0.0
Cosine of 0 radian : 1.0
Tangent of 0 radian : 0.0
Pi : 3.141592653589793
Random number (0.0 to 1.0) : 0.8711276341104848

```

5.8 การส่งวัตถุไปยังเมธอด (Passing Objects to Methods)

การส่งค่าไปยังเมธอดจะเป็นการส่งโดยค่า (Passing by value) ซึ่งค่าที่ส่งไปจะเป็นค่าที่อ้างอิงไปยังวัตถุ ตัวอย่างของการส่งค่าวัตถุเป็นอาร์กิวเมนต์ เช่น

ตัวอย่างที่ 5.9 การส่งวัตถุไปยังเมธอด (Passing Objects to Methods)

```

public class TestPassObject {
    public static void main(String[] args) {
        Circle myCircle = new Circle(1);
        int n = 5;
        printAreas(myCircle, n);
        System.out.println("\n" + "Radius is " + myCircle.getRadius());
        System.out.println("n is " + n);
    }
    public static void printAreas(Circle c, int times) {
        System.out.println("Radius \t\tArea");
        while (times >= 1) {
            System.out.println(c.getRadius() + "\t\t" + c.getArea());
            c.setRadius(c.getRadius() + 1);
            times--;
        }
    }
}

class Circle {
    private double radius = 1;
    private static int numberOfObjects = 0;
    public Circle() {
        numberOfObjects++;
    }
    public Circle(double newRadius) {
        radius = newRadius;
        numberOfObjects++;
    }
    public double getRadius() {
        return radius;
    }
    public void setRadius(double newRadius) {
        radius = (newRadius >= 0) ? newRadius : 0;
    }
}

```

```

    }
    public static int getNumberOfObjects() {
        return numberOfObjects;
    }
    public double getArea() {
        return radius * radius * Math.PI;
    }
}

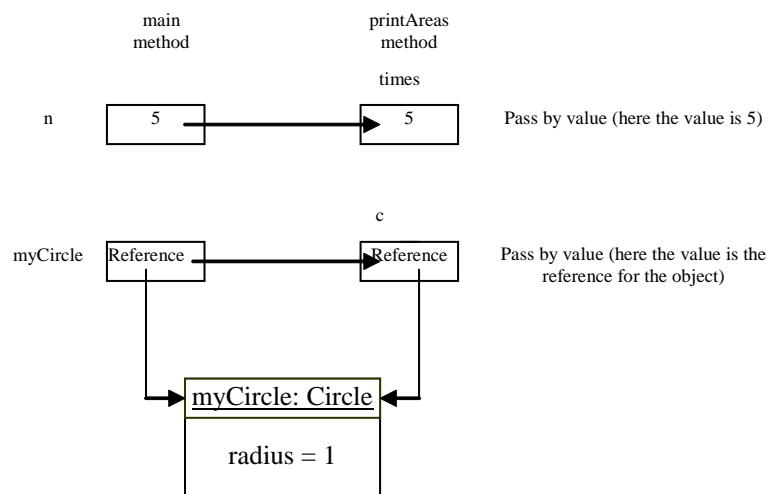
```

ผลลัพธ์ที่เกิดขึ้นคือ

Radius	Area
1.0	3.141592653589793
2.0	12.566370614359172
3.0	28.274333882308138
4.0	50.26548245743669
5.0	78.53981633974483

Radius is 6.0

n is 5



5.9 Recursive Method

Recursive method คือ เมธอดที่ถูกเรียกจากตัวเมธอดเอง ซึ่งเทคนิคนี้ได้ถูกนำมาใช้เพื่อให้โปรแกรมกระชับ และหลีกเลี่ยงการใช้ Loop แบบต่างๆ งานที่จะสามารถเขียนแบบ recursive ได้ จะต้องมียุติสิ้นสุดของการเรียกซ้ำอย่างชัดเจน ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 5.10 Recursive Method

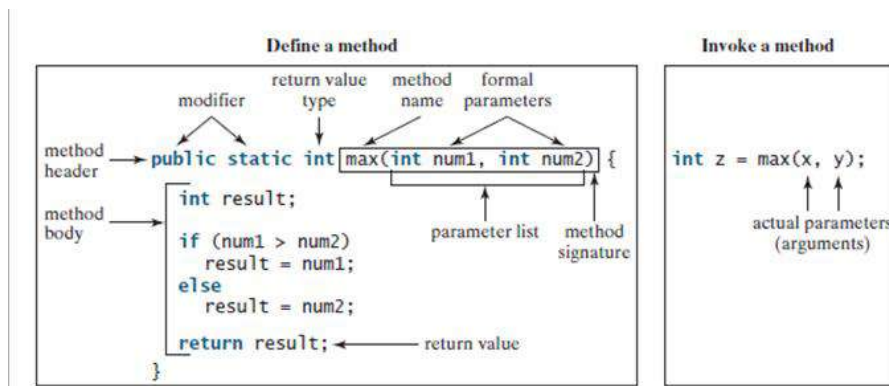
```
public class Demonstrate {
    public static void main (String args[] ) {
        System.out.print( "2 to the 3rd power is " );
        System.out.println( recursivePowerOf2(3) );
        System.out.print( "2 to the 10th power is " );
        System.out.println( recursivePowerOf2(10) );
    }
    public static int recursivePowerOf2 (int n) {
        if (n==0) {
            return 1;
        }
        else {
            return 2 * recursivePowerOf2(n-1);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

2 to the 3rd power is 8
2 to the 10th power is 1024

5.10 บทสรุป

เมธอดคือกลุ่มคำสั่งที่ถูกกำหนดขึ้นเพื่อให้ได้ผลลัพธ์อย่างใดอย่างหนึ่ง เป็นส่วนประกอบหนึ่งของคลาส เปรียบเสมือนกับฟังก์ชันในการพัฒนาโปรแกรมแบบโครงสร้าง (Structured programming) ตัวอย่างการสร้าง (เมธอดภายในคลาสมีรูปแบบดังนี้



เมธอดในภาษา Java มี 2 ประเภท ดังนี้

เมธอดที่เป็นสมาชิกของคลาส (class method) เมธอดประเภทนี้จะมีเพียงชุดเดียวเท่านั้น และไม่จำเป็นต้องสร้างวัตถุเพื่อเรียกใช้เมธอดกล่าวคือ เมธอดนี้สามารถเรียกใช้ผ่านชื่อคลาส ได้ `access_modifier` ที่ทำให้ เมธอดเป็นสมาชิกของคลาสคือ `static` เมธอดประเภทนี้ เหมาะสำหรับเมธอดที่ไม่มีการเปลี่ยนแปลงค่าไปตามวัตถุ เช่น สูตรในการคำนวณทางคณิตศาสตร์ ซึ่งเมธอดดังกล่าวจะรับค่าที่ผู้ใช้ต้องการคำนวณ และทำการคำนวณแล้วส่งผลลัพธ์กลับไปยังผู้ใช้

เมธอดที่เป็นสมาชิกของอินสแตนซ์ (instance method) เมธอดที่เป็นสมาชิกของ instance จะต้องเกิดจากตัวดำเนินการ `new` กล่าวคือ เมื่อจะเรียกใช้เมธอดประเภทนี้ จะต้องมีการสร้างวัตถุขึ้นมาก่อน และค่าของข้อมูลจะเปลี่ยนไปตามวัตถุที่สร้างขึ้น

Overloading คือ การเขียนชุดของเมธอดที่มีชื่อเหมือนกัน แต่มีพารามิเตอร์ที่แตกต่างกัน ซึ่งถือว่าเป็นคุณลักษณะพิเศษหนึ่งของการเขียนโปรแกรมเชิงวัตถุ คือ polymorphism

5.11 แบบฝึกหัดปฏิบัติการ

1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสร้างเมธอดจากตัวอย่างต่อไปนี้

ข้อ	Code	ผลลัพธ์
1	<pre> Class A { int x = 10, y = 5; void methodA (int a, int b) { a = x; b = y; } } class TestPassByValue { public static void main (String args[]) { int var1, var2; A objA = new A(); var1 = 5; var2 = 5; System.out.println("var1(before) = " +var1); System.out.println("var2(before) = " +var2); objA.methodA(var1, var2); System.out.println("var1(after) = " + var1); System.out.println("var2(after) = " + var2); } } </pre>	ให้อธิบายการทำงานของการใช้คำสั่ง objA.methodA(var1, var2);
2	<pre> class A { int x = 10, y = 10; void methodA (B objB) { objB.var1 = x; objB.var2 = y; } } class B { int var1, var2; } class TestPassByValue1 { public static void main(String args[]) { B objB = new B(); A objA = new A(); objB.var1 = 5; objB.var2 = 5; System.out.println("var1(before)="+objB.var1); System.out.println("var2(before)="+objB.var2); objA.methodA(objB); System.out.println("var1(after)="+objB.var1); System.out.println("var2(after)="+objB.var2); } } </pre>	ให้อธิบายการทำงานของการใช้คำสั่ง objA.methodA(objB);

2. จงอธิบายว่าเหตุใดโปรแกรมด้านล่างจึง Compiles ไม่ผ่าน

โปรแกรม	เหตุผล
<pre>public class TestProgram{ public static void main(String[] args){ int i=f(2, 3); } public static int f(int a, int b){ return Math.pow(a, b) + Math.pow(b, a); } }</pre>	
<pre>public class Test { public static method1(int n, m) { n += m; method2(3.4); } public int method2(int n) { if (n > 0) return 1; else if (n == 0) return 0; else if (n < 0) return -1; } }</pre>	
<pre>public class Test { public static void main(String[] args) { nPrintln (5, "Welcome to Java!"); } public static void nPrintln(String message, int n) { int n = 1; for (int i = 0; i < n; i++) System.out.println(message); } }</pre>	

3. จงเขียน method header สำหรับ method m() ในแต่ละข้อต่อไปนี้

- `int i=m(1, 1);`
- `float f=m(Math.exp(5));`
- `String s = m(2f, 8d);`
- `CsStudent l = m("John", "K.", "Wick");`
- `for(double d=1; d<=256; d *=2) m(d);`

4. จงอธิบาย output ที่ได้จาก main() ฟังก์ชันต่อไปนี้

โปรแกรม	เหตุผล
<pre> public class TestProgram{ public static void main(String[] args){ System.out.println(g("A")); } public static String f(){ System.out.println("A"); return "A"; } public static String g(String s){ return f()+s; } } </pre>	
<pre> public class Test { public static void main(String[] args) { int max = 0; max(1, 2, max); System.out.println(max); } public static void max(int value1, int value2, int max) { if (value1 > value2) max = value1; else max = value2; } } </pre>	
<pre> public class Test { public static void main(String[] args) { int i = 0; while (i <= 4) { method1(i); i++; } System.out.println("i is " + i); } public static void method1(int i) { do { if (i % 3 != 0) System.out.print(i + " "); i--; } while (i >= 1); System.out.println(); } } </pre>	

(ฝึกเขียนโปรแกรมตาม Concept OOP)

5. [RoachPopulation] ให้เขียนคลาส RoachPopulation ที่จำลองการเจริญเติบโตของประชากรแมลงสาบ โดยที่ constructor จะรับขนาดของประชากรแมลงสาบเริ่มต้น นอกจากนี้มีเมธอด wait ที่จำลองช่วงเวลา que ที่ประชากรแมลงสาบจะเพิ่มขึ้น 2 เท่า เมธอด spray จำลองการฉีดพ่นยาฆ่าแมลง ทำให้ประชากรของแมลงสาบลดลง 10% เมธอด getRoaches คืนค่าจำนวนประชากรแมลงสาบที่มีในปัจจุบัน ให้สร้างคลาสนี้และโปรแกรมทดสอบเพื่อจำลองห้องครัวที่มีแมลงสาบเริ่มต้น n ตัว ทำ wait, spray และพิมพ์ค่าจำนวนแมลงสาบตามจำนวนรอบที่ผู้ใช้กำหนด

ข้อมูลนำเข้า ขนาดของประชากรแมลงสาบเริ่มต้น จำนวนรอบของการทำ ทำ wait, spray

ข้อมูลส่งออก จำนวนประชากรแมลงสาบที่มีในปัจจุบัน

ข้อมูลนำเข้า	ข้อมูลส่งออก
3 10	60
2 100	334

6. [Palindrome] Palindrome คือคำหรือประโยคที่เมื่ออ่านอักษรจากหน้าไปหลังหรือหลังไปหน้าที่ค่าเท่ากัน โดยไม่พิจารณาช่องว่างหรืออักษรพิเศษ เช่น RADAR หรือ MADAM I'M ADAM ให้เขียนโปรแกรมที่มี static method ชื่อ isPalindrome(String s) คืนค่าบูลีนซึ่งเมธอดนี้ทำหน้าที่ในการตรวจสอบว่าคำหรือประโยคที่รับจากผู้ใช้เป็น palindrome หรือไม่

ข้อมูลนำเข้า คำหรือประโยคที่รับจากผู้ใช้นี้

ข้อมูลส่งออก ตรวจสอบว่าคำหรือประโยคที่รับจากผู้ใช้นี้เป็น palindrome หรือไม่

ข้อมูลนำเข้า	ข้อมูลส่งออก
RADAR	0
MADAM I'M ADAM	1

7. (MyTriangle) ให้สร้างคลาส MyTriangle ที่ประกอบด้วย method ต่อไปนี้

1. public boolean isValid(double side1, double side2, double side3) เป็นเมธอดที่คืนค่า true ถ้าผลรวมของสองด้านมีค่ามากกว่าด้านที่สาม

2. public double area(double side1, double side2, double side3) เป็นเมธอดที่คืนค่าพื้นที่ของสามเหลี่ยม $area = \sqrt{s(s - side1)(s - side2)(s - side3)}$

$$s = (side1 + side2 + side3)/2;$$

ให้เขียนโปรแกรมทดสอบที่อ่านค่าด้านสามด้านของสามเหลี่ยมและตรวจสอบว่าด้านดังกล่าวเป็นด้านของสามเหลี่ยมหรือไม่และคำนวณพื้นที่ของสามเหลี่ยมกรณีที่ด้านสามด้านดังกล่าว Valid

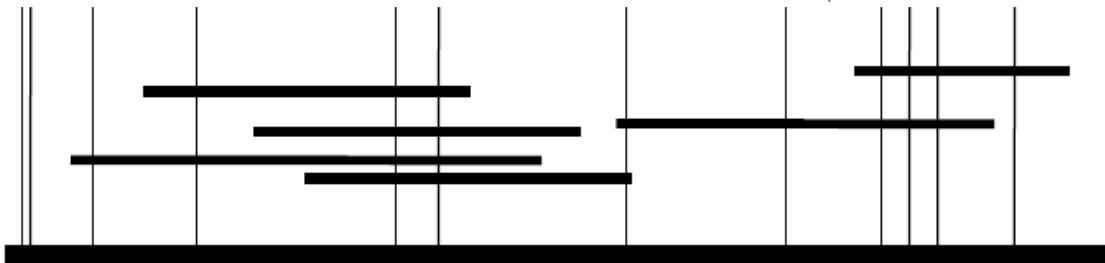
ข้อมูลนำเข้า ค่าด้านสามด้านของสามเหลี่ยม

ข้อมูลส่งออก ด้านดังกล่าวเป็นด้านของสามเหลี่ยมหรือไม่และคำนวณพื้นที่ของสามเหลี่ยม

กรณีที่ด้านสามด้านดังกล่าว Valid ให้พิมพ์ และพิมพ์พื้นที่ของสามเหลี่ยม 1
กรณีที่ด้านสามด้านดังกล่าว ไม่ Valid ให้พิมพ์ 0

ข้อมูลนำเข้า	ข้อมูลส่งออก
3 2 1	0
4 3 2	2.90

8. [Light] ในโลกสองมิติ ยานบินอวกาศมากมายลอยเป็นเส้นขนานกับแกน x อยู่บนอวกาศ เข้าวันหนึ่งเกิดปรากฏการณ์ประหลาดเกิดขึ้น คือมีลำแสงส่องขึ้นมาจากพื้นที่ตำแหน่งต่าง ๆ ลำแสงนี้ส่องทะลุยานบินจนถึงขอบฟ้า ดังรูป



ให้เขียนโปรแกรมรับตำแหน่งของยานบิน และตำแหน่งที่เกิดแสงจากพื้น แล้วคำนวณว่าจุดที่แสงส่องผ่านยานบินมีทั้งหมดกี่จุด ในกรณีที่แสงวิ่งผ่านหัวหรือท้ายยานพอดีจะไม่นับ

ข้อมูลนำเข้า บรรทัดแรกระบุจำนวนเต็มสองจำนวน N M โดย N แทนจำนวนยานบิน และ M แทนจำนวนจุดที่เกิดแสงจากพื้น

จากนั้น N บรรทัด แต่ละบรรทัดจะระบุตำแหน่งของยานบินโดยจะเป็นพิกัดในแนวแกน x ณ จุดเริ่มต้นและจุดสิ้นสุดของยาน และ M บรรทัดถัดไปจะระบุตำแหน่งที่แสงเกิด

ข้อมูลส่งออก

จุดที่แสงส่องผ่านยานบินมีทั้งหมดกี่จุด

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
4 5 10 30 5 15 1 15 5 50 5 12 15 40 80	8

บทที่ 6 อาร์เรย์

วัตถุประสงค์

- 6.1 สามารถอธิบายความหมายของอาร์เรย์ อาร์เรย์ของตัวแปรพื้นฐาน อาร์เรย์ของตัวแปรชนิดอ้างอิง
- 6.2 สามารถเขียนโปรแกรมเพื่อประยุกต์การใช้อาร์เรย์ในการเรียงลำดับ การค้นหาข้อมูล

6.1 ความนำ

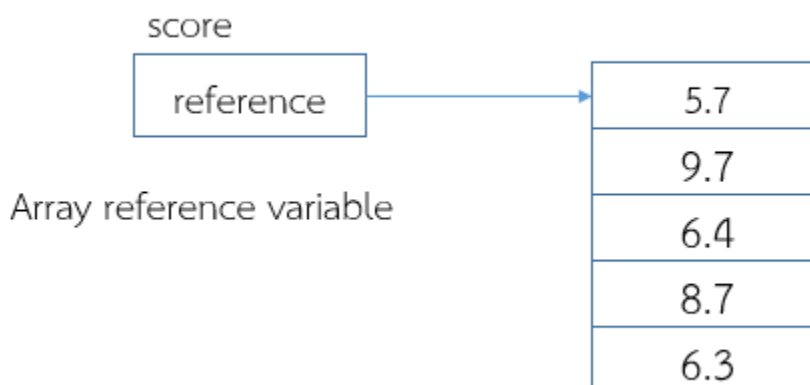
ตัวแปรชุด (array) เป็นชนิดข้อมูลประเภทหนึ่งที่น่าเอาชนิดข้อมูลข้อมูลพื้นฐาน เช่น ตัวอักษร (char) ชนิดข้อมูลแบบเลขจำนวนเต็ม (int) และชนิดข้อมูลแบบเลขจำนวนจริง (float) มาประยุกต์เป็นชนิดข้อมูลประเภทนี้ เมื่อประกาศโครงสร้างข้อมูลแบบอาร์เรย์ (array) จะเก็บข้อมูลต่างจากชนิดข้อมูลพื้นฐานทั่วไป คือ สามารถเก็บค่าภายในตัวแปรชนิดนี้ได้มากกว่า 1 ค่าซึ่งจำนวนค่าที่เก็บนั้นขึ้นอยู่กับขนาดของอาร์เรย์ที่ได้กำหนดไว้

6.2 นิยามของอาร์เรย์

อาร์เรย์คือโครงสร้างข้อมูลที่มีชนิดข้อมูลประเภทเดียวกัน การประกาศอาร์เรย์ของชนิดข้อมูลพื้นฐาน มีรูปแบบดังนี้

```
double[] score = new double[5];
```

โดยการประกาศอาร์เรย์ดังกล่าวข้างต้นสามารถแสดงได้ดังรูปต่อไปนี้



รูปที่ 6.1 การประกาศอาร์เรย์ของชนิดข้อมูลพื้นฐาน

ประเภทของตัวแปรชุด อาจแบ่งตามลักษณะของจำนวนตัวเลขของดัชนีคือ

1. ตัวแปรชุด 1 มิติ (one dimension arrays หรือ single dimension arrays) เป็นตัวแปรชุดที่มีตัวเลขแสดงขนาดเป็นเลขตัวเดียว เช่น word[20] num[25] x[15]
2. ตัวแปรชุดหลายมิติ (multi-dimension arrays) เป็นตัวแปรชุดที่ชื่อมีตัวเลขแสดงขนาดเป็นตัวเลขหลายตัว ที่นิยมใช้กันมี 2 มิติกับ 3 มิติ
 - 2.1 ตัวแปรชุด 2 มิติมีเลขแสดงขนาด 2 ตัว เช่น a[3][5] name[5][6]
 - 2.2 ตัวแปรชุด 3 มิติมีเลขแสดงขนาด 3 ตัว เช่น a[3][5][6] name[5][6][8]

6.2.1 การประกาศและสร้างตัวแปรอาเรย์

ในการประกาศและสร้างตัวแปรอาเรย์ ประกอบด้วยขั้นตอนดังต่อไปนี้

1. การประกาศตัวแปรอาเรย์สามารถประกาศดังนี้ `datatype[] arrayRefVar`; เช่น `double[] myList`;
2. การสร้างอาเรย์สามารถประกาศดังนี้ `arrayRefVar = new datatype[arraySize]`; เช่น `myList = new double[10]`; โดย `myList[0]` อ้างถึงสมาชิกตัวแรกในอาเรย์ `myList[9]` อ้างถึงสมาชิกในอาเรย์ตัวสุดท้าย
3. การประกาศและสร้างอาเรย์ในหนึ่งขั้นตอนสามารถประกาศดังนี้

```
datatype[] arrayRefVar = new datatype[arraySize];
```

เช่น `double[] myList = new double[10];`

6.2.2 การกำหนดค่าเริ่มต้นให้กับอาเรย์(Array Initializers)

การประกาศ สร้าง และกำหนดค่าให้อาเรย์ในหนึ่งขั้นตอน เช่น `double[] myList = {1.9, 2.9, 3.4, 3.5}`; ตัวอย่างของโปรแกรมที่แสดงการประกาศ สร้าง และกำหนดค่าให้อาเรย์ในหนึ่งขั้นตอนและพิมพ์สมาชิกที่อยู่ในอาเรย์สามารถแสดงได้ดังตัวอย่างที่ 6.1

ตัวอย่างที่ 6.1 การประกาศ สร้าง และกำหนดค่าให้อาเรย์ในหนึ่งขั้นตอนและพิมพ์สมาชิกที่อยู่ในอาเรย์

```
class ExArray1 {
    public static void main (String args[] ) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};
        for(int i=0;i<myList.length;i++){
            System.out.println(myList[i]);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
1.9
2.9
3.4
3.5
```

โดยในการประกาศ สร้าง และกำหนดค่าให้อาเรย์ในหนึ่งขั้นตอนดังตัวอย่างข้างต้นจะมีผลลัพธ์เหมือนกับตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.2 การประกาศ สร้าง และกำหนดค่าให้อาเรย์

```
class ExArray2 {
    public static void main (String args[] ) {
        double[] myList = new double[4];
        myList[0] = 1.9;
        myList[1] = 2.9;
        myList[2] = 3.4;
        myList[3] = 3.5;
        for(int i=0;i<myList.length;i++){
            System.out.println(myList[i]);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
1.9
2.9
3.4
3.5
```

อาร์เรย์เมื่ออาร์เรย์ถูกประกาศแล้วจะไม่สามารถเปลี่ยนขนาดของอาร์เรย์นอกจากนี้ยังสามารถหาขนาดของอาร์เรย์ได้จาก `arrayRefVar.length` เมื่ออาร์เรย์ถูกสร้าง จะมีการกำหนดค่าเริ่มต้นให้กับอาร์เรย์ด้วยค่าเริ่มต้น โดยการกำหนด 0 สำหรับชนิดข้อมูลแบบตัวเลข `"\u0000"` สำหรับชนิดข้อมูลแบบตัวอักษร `false` สำหรับชนิดข้อมูลแบบ boolean

สมาชิกของอาร์เรย์สามารถเข้าถึงได้ด้วย index โดย index ของอาร์เรย์เริ่มที่ 0 ถึง ขนาดของอาร์เรย์ลบ 1 (`arrayRefVar.length-1`) เช่น `arrayRefVar[index]`;

6.3 อาร์เรย์ของตัวแปรพื้นฐาน

6.3.1 การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท integer

เช่น `int a=new int[10]`; เป็นการประกาศตัวแปร array ชื่อ a เป็น array ของข้อมูลประเภท integer มีสมาชิกได้ จำนวน 10 ตัว คือ `a[0]` `a[1]` `a[2]` `a[3]` ... `a[9]` โดยสมาชิกแต่ละตัวจะใช้เนื้อที่เท่ากับตัวแปรประเภท integer ที่ไม่ได้ อยู่ใน array คือ 4 ไบต์ ต่อ ตัวแปร 1 ตัว ดังนั้นเนื้อที่หน่วยความจำที่ใช้ทั้งหมดจึงเท่ากับจำนวนสมาชิก คูณ ด้วย 4 ไบต์ การกำหนดค่าให้แก่ตัวแปร array อาจกำหนดพร้อมกับการประกาศ เช่น `int[] num1 ={56,25,89}`; เป็นการประกาศว่าตัวแปร `num1` เป็น array ประเภท integer มีสมาชิก 3 ตัวโดย `num1[0] = 56`; `num1[1]=25`; `num1[2]=89`;

หรือสามารถประกาศได้อีกรูปแบบดังนี้ `int a[]={200,230}`; เป็นการประกาศว่า a เป็นตัวแปร array ประเภท integer ที่มีสมาชิก 2 ตัว โดย `a[0]` มีค่าเป็น 200 `a[1]` มีค่าเป็น 230

แต่ไม่สามารถประกาศว่า `int value[]`; โดยถ้าจะไม่ระบุจำนวนสมาชิก ต้องระบุค่าของแต่ละสมาชิกที่ถูก ล้อมรอบด้วย { } โดยระหว่างสมาชิกคั่นด้วยเครื่องหมาย , (คอมม่า) ดังตัวอย่าง

ตัวอย่างที่ 6.3 การประกาศตัวแปรอาร์เรย์และพิมพ์สมาชิกในอาร์เรย์

```
class ExArray3 {
    public static void main (String args[] ) {
        double[] myList;
        myList = {1.9, 2.9, 3.4, 3.5};

        for(int i=0;i<myList.length;i++){
            System.out.println(myList[i]);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
ExArray3.java:4: error: illegal start of expression
    myList = {1.9, 2.9, 3.4, 3.5};
```

หรือสามารถประกาศตัวแปรโดยยังไม่กำหนดค่า เช่น `int money[5]`; แล้วไปกำหนดค่าให้สมาชิกแต่ละตัวในภายหลัง เช่น `money[0] = 250`; `money[4] = 500`;

6.3.2 การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท float

เช่น `float b=new float[10]`; เป็นการประกาศตัวแปร array ของ ตัวแปรจำนวนที่มีทศนิยมได้ คือ float ในชื่อ b ซึ่งมีสมาชิกได้ 5 ตัว คือ `b[0]` `b[1]`... `b[9]` โดยสมาชิกแต่ละตัวใช้หน่วยความจำ 4 ไบต์ ดังนั้นทั้งหมดจะใช้หน่วยความจำ

4 คูณ 5 คือ 40 ไบต์ การกำหนดค่าของตัวแปร array ประเภท float เป็นไปในลักษณะเดียวกับ array ประเภท integer ประกาศพร้อมกับกำหนดค่าให้เลยโดยล้อมรอบด้วย { } และค่าของสมาชิกแต่ละตัวคั่นด้วย , เช่น

```
float[] num = {2.00,1.25,5.36,6.32,246.10};
```

เป็นการประกาศว่าตัวแปร num เป็น array ประเภท float มีสมาชิก 5 ตัวโดย

```
num[0] = 2.00 num[1] = 1.25
```

```
num[2] = 5.36
```

```
num[3] = 6.32
```

```
num[4] = 246.10
```

หรือประกาศตัวแปรก่อนแล้วไปกำหนดค่าภายหลัง เช่น float salary[10]; เป็นการประกาศว่าตัวแปร salary เป็น array ประเภท float มีสมาชิก 10 ตัวโดยในการกำหนดค่าสามารถกำหนดได้ด้วยคำสั่งต่อไปนี้

```
salary[0] = 25000.00;
```

```
salary[9] = 55600.00;
```

6.3.3 การประมวลผลด้วยอาร์เรย์

6.3.3.1 การหาผลรวมภายในอาร์เรย์ สามารถแสดงการทำงานดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.4 การหาผลรวมภายในอาร์เรย์

```
public class ExArray4 {
    public static void main(String[] args) {
        int[] values = new int[5];
        for (int i = 1; i < 5; i++) {
            values[i] = i + values[i-1];
        }
        values[0] = values[1] + values[4];
        for(int i=0;i<values.length;i++){
            System.out.println(values[i]);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
11
1
3
6
10
```

ตัวอย่างที่ 6.5 การหาผลรวมภายในอาร์เรย์

```
import java.util.Scanner;
public class ExArray5 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int sum = 0;
        int[] values = new int[5];
        for (int i = 0; i < 5; i++) {
            values[i] = input.nextInt();
        }
        for (int i = 0; i < values.length; i++) {
            sum += values[i];
        }
        System.out.println("Sum= " + sum);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
5
1
2
3
4
5
Sum= 15
```

6.3.3.2 การหาค่าที่มากที่สุดหรือน้อยที่สุดในอาร์เรย์ สามารถแสดงการทำงานดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.6 การหาค่าที่มากที่สุดภายในอาร์เรย์

```
import java.util.Scanner;
public class ExArray6 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int sum = 0;
        int[] values = new int[5];
        for (int i = 0; i < 5; i++) {
            values[i] = input.nextInt();
        }
        int max = values[0];
        for (int i = 0; i < values.length; i++) {
            if (max < values[i]) {
                max = values[i];
            }
        }
        System.out.println("Max= " + max);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
5
9
7
8
15
2
Max= 15
```

6.3.3.3 การคัดลอกข้อมูลภายในอาร์เรย์ สามารถแสดงการทำงานดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.7 การคัดลอกข้อมูลภายในอาร์เรย์

```
public class ExArray7 {
    public static void main(String[] args) {
        int[] sourceArray = {2, 3, 1, 5, 10};
        int[] targetArray = new int[sourceArray.length];
        for (int i = 0; i < sourceArray.length; i++){
            targetArray[i] = sourceArray[i];
        }
        System.out.println("Source Array:");
        for(int i=0; i < sourceArray.length;i++){
            System.out.print(sourceArray[i]+" ");
        }
        System.out.println("\nTarget Array:");
        for(int i=0;i < targetArray.length;i++){
            System.out.print(targetArray[i]+" ");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Source Array:
2 3 1 5 10
Target Array:
2 3 1 5 10
```

ตัวอย่างที่ 6.8 การคัดลอกข้อมูลภายในอาร์เรย์

```
public class ExArray8 {
    public static void main(String[] args) {
        int[] sourceArray = {2, 3, 1, 5, 10};
        int[] targetArray = new int[sourceArray.length];
        System.arraycopy(sourceArray, 0, targetArray, 0, sourceArray.length);

        System.out.println("Source Array:");
        for(int i=0; i < sourceArray.length;i++){
            System.out.print(sourceArray[i]+" ");
        }
        System.out.println("\nTarget Array:");
        for(int i=0;i < targetArray.length;i++){
            System.out.print(targetArray[i]+" ");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Source Array:
2 3 1 5 10
Target Array:
2 3 1 5 10
```

6.3.3.4 การส่งอาร์เรย์ไปยังเมธอด (Passing Arrays to Methods) สามารถแสดงการทำงานดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.9 การส่งอาร์เรย์ไปยังเมธอด

```
public class ExArray9 {
    public static void main(String[] args) {
        int[] list = {3, 1, 2, 6, 4, 2};
        printArray(list);
    }
    public static void printArray(int[] array) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

3 1 2 6 4 2

ตัวอย่างที่ 6.10 การส่งอาร์เรย์ไปยังเมธอด

```
public class ExArray10 {
    public static void main(String[] args) {
        printArray(new int[]{3, 1, 2, 6, 4, 2});
    }
    public static void printArray(int[] array) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

3 1 2 6 4 2

ตัวอย่างที่ 6.11 การส่งอาร์เรย์ไปยังเมธอด

```
public class ExArray11 {
    public static void main(String[] args) {
        int x = 1; // x represents an int value
        int[] y = new int[10]; // y represents an array of int values

        m(x, y); // Invoke m with arguments x and y

        System.out.println("x is " + x);
        System.out.println("y[0] is " + y[0]);
    }
    public static void m(int number, int[] numbers) {
        number = 1001; // Assign a new value to number
        numbers[0] = 5555; // Assign a new value to numbers[0]
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

x is 1

y[0] is 5555

6.3.3.5 การคืนค่าอาเรย์จากเมธอด (Returning an Array from a Method) สามารถแสดงการทำงานดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.12 การคืนค่าอาเรย์จากเมธอด

```
public class ExArray12 {
    public static void main(String[] args) {
        int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
        int[] list2 = reverse(list1);
        for (int i = 0; i < list2.length; i++) {
            System.out.print(list2[i] + " ");
        }
    }

    public static int[] reverse(int[] list) {
        int[] result = new int[list.length];
        for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {
            result[j] = list[i];
        }
        return result;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

6 5 4 3 2 1

6.4 อาเรย์สองมิติ

1. การประกาศตัวแปรอาเรย์สองมิติสามารถประกาศดังนี้

```
dataType[][] refVar;
```

เช่น double[][] myList;

2. การสร้างอาเรย์

```
refVar = new dataType[10][10];
```

เช่น myList = new double[10][10];

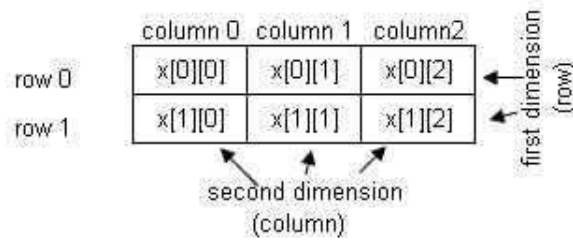
3. การประกาศและสร้างอาเรย์ในหนึ่งขั้นตอน

```
dataType[][] refVar = new dataType[10][10];
```

หรือสามารถประกาศด้วยคำสั่งต่อไปนี้

```
dataType refVar[][] = new dataType[10][10];
```

อาเรย์ 2 มิติ มีการจัดการจัดเก็บเปรียบเทียบกับ ตาราง 2 มิติ โดยมิติที่ 1 เปรียบเหมือนแถว (row) ของตาราง มิติที่ 2 เปรียบคล้ายกับสดมภ์ (column) ของตาราง ดังรูป



ตัวอย่างที่ 6.13 การประกาศและสร้างอาร์เรย์ 2 มิติ

```
public class ExArray13 {
    public static void main(String[] args) {
        int[][] matrix = new int[10][10];
        matrix[0][0] = 3;
        for (int i = 0; i < matrix.length; i++){
            for (int j = 0 ; j < matrix[i].length; j++){
                matrix[i][j] = (int) (Math.random() * 1000);
            }
        }
        for (int i = 0; i < matrix.length; i++){
            for (int j = 0; j < matrix[i].length; j++){
                System.out.print(matrix[i][j]+" ");
            }
            System.out.println("");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
720 170 641 973 444 142 532 299 951 173
574 847 404 272 913 64 234 438 904 338
732 170 243 63 873 146 788 490 721 980
782 563 522 998 148 950 58 110 802 959
145 971 480 46 207 784 390 184 698 317
139 456 787 913 746 835 122 534 243 281
653 183 307 955 514 347 630 576 664 480
198 658 548 919 511 685 354 25 808 45
511 824 733 909 984 860 538 94 549 514
816 356 691 558 979 481 969 301 172 23
```

เราสามารถประกาศ สร้าง และกำหนดค่าให้อาร์เรย์สองมิติได้ ดังตัวอย่าง

ตัวอย่างที่ 6.14 การประกาศและสร้างอาร์เรย์ 2 มิติ

```
public class ExArray14 {
    public static void main(String[] args) {
        int[][] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9},
            {10, 11, 12}
        };
        for (int i = 0; i < array.length; i++){
            for (int j = 0; j < array[i].length; j++){
                System.out.print(array[i][j]+" ");
            }
            System.out.println("");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
1 2 3
```

```

4 5 6
7 8 9
10 11 12

```

ตัวอย่างที่ 6.15 การประกาศและสร้างอาร์เรย์ 2 มิติ

```

public class ExArray15 {
    public static void main(String[] args) {
        int[][] array = new int[4][3];
        array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
        array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
        array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
        array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
        for (int i = 0; i < array.length; i++){
            for (int j = 0; j < array[i].length; j++){
                System.out.print(array[i][j]+" ");
            }
            System.out.println("");
        }
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

1 2 3
4 5 6
7 8 9
10 11 12

```

เมื่อประกาศอาร์เรย์ 2 มิติแล้วเราสามารถหาขนาดของอาร์เรย์ได้ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6.16 การประกาศและสร้างอาร์เรย์ 2 มิติ

```

public class ExArray16 {
    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3, 4, 5},
            {2, 3, 4, 5},
            {3, 4, 5},
            {4, 5},
            {5}
        };
        System.out.println("matrix.length is "+matrix.length);
        System.out.println("matrix[0].length is "+matrix[0].length);
        System.out.println("matrix[1].length is "+matrix[1].length);
        System.out.println("matrix[2].length is "+matrix[2].length);
        System.out.println("matrix[3].length is "+matrix[3].length);
        System.out.println("matrix[4].length is "+matrix[4].length);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

```

ตัวอย่างที่ 6.17 การประกาศและสร้างอาร์เรย์ 2 มิติและการหาผลรวมของข้อมูลในอาร์เรย์

```
import java.util.Scanner;
public class ExArray17{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int[][] m = new int[3][4];
        System.out.println("Enter " + m.length + " rows and " + m[0].length + " columns: ");
        for (int i = 0; i < m.length; i++){
            for (int j = 0; j < m[i].length; j++){
                m[i][j] = input.nextInt();
            }
        }
        System.out.println("\nSum of all elements is " + sum(m));
    }
    public static int sum(int[][] m) {
        int total = 0;
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[i].length; j++) {
                total += m[i][j];
            }
        }
        return total;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Enter 3 rows and 4 columns:
7 8 9 5
3 5 7 8
9 4 7 2
Sum of all elements is 74
```

6.5 อาร์เรย์ลิสต์ (ArrayList)

ArrayList เป็นโครงสร้างข้อมูลประเภทหนึ่งใช้จัดเก็บชุดของข้อมูลที่เป็นแบบ Array ทั้งรูปแบบชนิดเดียวกัน และต่างชนิดกัน โดยที่ไม่ทราบขนาดหรือ Size ที่แท้จริง

โดย ArrayList เป็นคลาสใน java.util.ArrayList เมื่อต้องการใช้งานจะต้องทำการ import เข้ามาในคลาสก่อน การใช้งานคล้ายกับ Array โดยที่เราสามารถเพิ่ม สมาชิกให้กับ ArrayList ผ่านเมธอดที่มีชื่อว่า add() สามารถเพิ่มสมาชิกหรือข้อมูลใน ArrayList ตามความต้องการของข้อมูลชุดเหล่านั้น และสามารถเพิ่มปรับลดขนาดของ ArrayList ในตำแหน่ง index ต่าง ๆ ได้ ใน ArrayList สามารถรับข้อมูลได้หลากหลายประเภท ที่อยู่ในชุดประเภทเดียวกัน รองรับข้อมูลที่ซ้ำกันได้ รวมทั้งที่เป็นค่าว่าง (null) โดยสมาชิกของ ArrayList จะถูกจัดเก็บตามลำดับที่เพิ่มเข้ามา และ ArrayList ยังเป็นชนิดที่เป็น unsynchronized สามารถเข้ามาใช้ข้อมูลใน ArrayList ได้ในเวลาเดียวกันหลาย ๆ ครั้ง โดยไม่มีปัญหาจะต้องรอคิวว่าให้ Thread อื่นที่เรียกใช้ ArrayList ตัวนั้น ๆ ทำงานจนเสร็จสิ้นเสียก่อน

ตัวอย่างที่ 6.18 ตัวอย่างการสร้างอาร์เรย์ลิสต์

```
import java.util.Scanner;
import java.util.ArrayList;
public class ExArray18{
    public static void main(String[] args) {
        ArrayList<String> myArrList = new ArrayList<String>();
        myArrList.add("Dog");
        myArrList.add("Cat");
        myArrList.add("Tiger");
        myArrList.add("Lion");
        System.out.println("Array list size is "+myArrList.size());
        for (int i = 0; i < myArrList.size(); i++) {
            System.out.println(myArrList.get(i));
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Array list size is 4
Dog
Cat
Tiger
Lion
```

ตัวอย่างที่ 6.19 ตัวอย่างการสร้างอาร์เรย์ลิสต์

```
import java.util.Scanner;
import java.util.ArrayList;
public class ExArray19{
    public static void main(String[] args) {
        ArrayList<String> myArrList = new ArrayList<String>();
        String animal = "Dog,Cat,Tiger,Lion";
        String[] animalArray = animal.split(",");
        for(String temp: animalArray)
        {
            myArrList.add(temp);
        }
        System.out.println("Array list size is "+myArrList.size());
        for (int i = 0; i < myArrList.size(); i++) {
            System.out.println(myArrList.get(i));
        }
    }
}
```

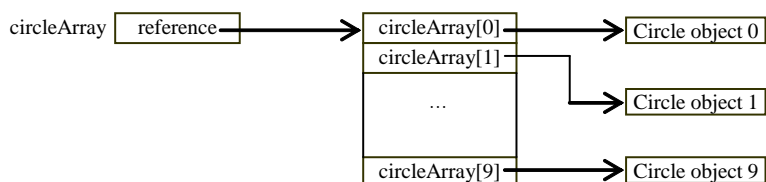
ผลลัพธ์ที่เกิดขึ้นคือ

```
Array list size is 4
Dog
Cat
Tiger
Lion
```

6.6 อาร์เรย์ของตัวแปรชนิดอ้างอิง

อาร์เรย์ของวัตถุหรือเรียกอีกอย่างว่าอาร์เรย์ของตัวแปรอ้างอิงวัตถุ (array of reference variables) ตัวอย่างของการประกาศอาร์เรย์ของวัตถุ เช่น `Circle[] circleArray = new Circle[10];`

เมื่อประกาศ `Circle[] circleArray = new Circle[10];` จะมีรูปแบบของการอ้างอิงดังนี้



การเรียกใช้ `circleArray[1].findArea()` จะทำงานใน 2 ขั้นตอนคือ

1. การประกาศ `circleArray` จะเป็นการอ้างอิงไปยังอาร์เรย์ทั้งหมด
2. ส่วน `circleArray[1]` จะอ้างอิงไปยังวัตถุของคลาส `Circle` ที่อยู่ที่ตำแหน่งดัชนี 1

ตัวอย่างที่ 6.20 การประกาศอาร์เรย์ของวัตถุ

```

public class ExArray20 {
    public static void main(String[] args) {
        Circle[] circleArray;
        circleArray = createCircleArray();
        printCircleArray(circleArray);
    }
    public static Circle[] createCircleArray() {
        Circle[] circleArray = new Circle[5];
        for (int i = 0; i < circleArray.length; i++) {
            circleArray[i] = new Circle((double)i);
        }
        return circleArray;
    }
    public static void printCircleArray(Circle[] circleArray) {
        System.out.println("Radius\t\t" + "Area");
        for (int i = 0; i < circleArray.length; i++) {
            System.out.print(circleArray[i].getRadius() + "\t\t" +
                circleArray[i].getArea() + '\n');
        }
        System.out.println("-----");
        System.out.println("The total areas of circles is \t" +
            sum(circleArray));
    }
    public static double sum(Circle[] circleArray) {
        double sum = 0;
        for (int i = 0; i < circleArray.length; i++)
            sum += circleArray[i].getArea();
        return sum;
    }
}

class Circle {
    private double radius;
    Circle() {
    }
    Circle(double r) {
        this.radius=r;
    }
    double getRadius() {
        return this.radius;
    }
    void setRadius(double radius) {
  
```

```

    this.radius=radius;
}
double getArea(){
    return radius * radius * 3.14159;
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

Radius	Area
0.0	0.0
1.0	3.14159
2.0	12.56636
3.0	28.27431
4.0	50.26544

The total areas of circles is 94.2477

6.5 การประยุกต์การใช้อาเรย์ในการเรียงลำดับ การค้นหาข้อมูล

6.5.1 การประยุกต์การใช้อาเรย์ในการค้นหาข้อมูล เมื่อประกาศอาเรย์และหากต้องการค้นหาข้อมูลที่ต้องการจะ สามารถทำได้ด้วยการนำข้อมูลที่ต้องการค้นหาหรือเรียกว่าคีย์ ทำการเปรียบเทียบกับสมาชิกทุกตัวที่อยู่ในอาเรย์ เราเรียก วิธีการค้นหาด้วยการเปรียบเทียบคีย์กับข้อมูลอาเรย์แต่ละตัวด้วยวิธีการดังกล่าวว่า **linear search** ดังแสดงดังรูป

Key	List
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8

การค้นหาแบบลำดับ หรือเรียกว่าการค้นหาข้อมูลแบบเชิงเส้น (Linear Search) เป็นวิธีพื้นฐานมากที่สุด ที่มีความ เรียบง่ายและไม่ซับซ้อน โดยมักนำไปใช้งานบนลิสต์ที่ไม่ได้มีการเรียงลำดับข้อมูล ซึ่งโดยปกติทั่วไปแล้ว เทคนิคการค้นหา ข้อมูลแบบลำดับนี้ สามารถนำไปใช้งานได้เหมาะสมกับลิสต์ที่มีขนาดเล็ก หรือใช้งานบนลิสต์ที่มักไม่ค่อยใช้เพื่อการ ค้นหาข้อมูลอยู่บ่อย ๆ โดยหลักการค้นหา จะเริ่มต้นค้นหาจากตำแหน่งแรกภายในลิสต์ และทำการตรวจสอบข้อมูลภายใน ลิสต์ว่าตรงกับค่าที่ต้องการค้นหา (Target) หรือไม่ ถ้าไม่ตรง ก็จะดำเนินการค้นหาตัวถัดไป และจะดำเนินการเปรียบเทียบ ค่าเช่นนี้ไปเรื่อย ๆ โดยผลลัพธ์จากการค้นหาข้อมูลจะมีความเป็นไปได้อยู่ 2 กรณีด้วยกันคือ

1. พบตำแหน่งข้อมูลที่ต้องการภายในลิสต์ (Successful Search)
2. ไม่พบตำแหน่งข้อมูลที่ต้องการภายในลิสต์ (Unsuccessful Search)

ตัวอย่างที่ 6.21 การค้นหาข้อมูลแบบเชิงเส้น (Linear Search)

```
public class ExArray21 {
    public static void main(String[] args) {
        int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
        System.out.println(linearSearch(list, 4));
        System.out.println(linearSearch(list, -4));
        System.out.println(linearSearch(list, -3));
    }
    public static int linearSearch(int[] list, int key) {
        for (int i = 0; i < list.length; i++)
            if (key == list[i])
                return i;
        return -1;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
1
-1
5
```

การค้นหาข้อมูลแบบลำดับเป็นวิธีที่เรียบง่ายไม่ซับซ้อนแต่วิธีก็หากข้อมูลมีปริมาณมากจะเสียเวลาในการค้นหามากขึ้นตามลำดับดังนั้นในการค้นหาข้อมูลนอกจากการค้นหาเชิงลำดับแล้วจะมีการค้นหาด้วยการแบ่งค้นหาข้อมูลที่ละครึ่ง เช่น การค้นหาข้อมูลแบบไบนารี โดยการค้นหาแบบไบนารีจะนำไปใช้กับการค้นหาข้อมูลที่มีการเรียงลำดับแล้วเท่านั้น และเหมาะกับการนำไปค้นหาข้อมูลที่มีปริมาณข้อมูลจำนวนมาก โดยหลักการค้นหาด้วยวิธีนี้ จะดำเนินการแบ่งข้อมูลในอาร์เรย์ออกเป็น 2 ส่วน โดยค่าที่ต้องการค้นหานั้นจะอยู่ในส่วนครึ่งแรกหรือส่วนครึ่งหลังของอาร์เรย์ โดยหากค่าที่ต้องการค้นหาอยู่ส่วนครึ่งแรกของลิสต์ เราก็ไม่จำเป็นต้องสนใจลิสต์ส่วนครึ่งหลังอีกต่อไป ในทำนองเดียวกัน หากค่าที่ต้องการค้นหาอยู่ส่วนครึ่งหลังของลิสต์ ก็ไม่จำเป็นต้องสนใจลิสต์ส่วนครึ่งแรก เป็น

การค้นหาตำแหน่งกึ่งกลางของอาร์เรย์จะใช้ตัวแปร 3 ตัวด้วยกันคือ

1. ตัวแปร low เป็นตัวแปรที่ใช้สำหรับกำหนดตำแหน่งเริ่มต้นของอาร์เรย์ในแต่ละรอบ
2. ตัวแปร mid เป็นตัวแปรที่ใช้สำหรับกำหนดตำแหน่งกึ่งกลางของอาร์เรย์ในแต่ละรอบ โดยการคำนวณตำแหน่งกึ่งกลางสามารถคำนวณได้ดังนี้

$$\text{mid} = [(\text{low} + \text{high}) / 2]$$

3. ตัวแปร high เป็นตัวแปรที่ใช้สำหรับกำหนดตำแหน่งท้ายสุดของอาร์เรย์ในแต่ละรอบ

วิธีการค้นหาด้วยการค้นหาแบบแบบไบนารีดังแสดงดังรูป

Key	List
8	1 2 3 4 6 7 8 9
8	1 2 3 4 6 7 8 9
8	1 2 3 4 6 7 8 9

โดยสมมติว่ามีชุดตัวเลขที่เรียงลำดับแล้วภายในลิสต์ดังนี้ คือ {1, 2, 3, 4, 5, 6, 7, 8, 9} และค่าคีย์ที่ต้องการค้นหาคือค่า 8

เริ่มต้นจากการคำนวณหาตำแหน่งกึ่งกลาง ซึ่งจะได้เท่ากับ $(0+7)/2=3$ (ปัดเศษทิ้ง) ดังนั้นตำแหน่ง 3 ภายในอาร์เรย์จึงเป็นจุดกึ่งกลางที่ใช้สำหรับแบ่งกลุ่ม เมื่อทำการเปรียบเทียบค่าที่ตำแหน่งดังกล่าวจะพบว่าค่าคีย์ 8 มีค่ามากกว่า 4 จึงทำการปรับค่า low mid และ high เพื่อค้นหาข้อมูลในตำแหน่งต่อไป

ตัวอย่างที่ 6.22 การค้นหาข้อมูลแบบไบนารี

```
public class ExArray22 {
    public static void main(String[] args) {
        int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
        int ans = binarySearch (list, 7);
        if (ans != -1) {
            System.out.println("Found");
        }
        else {
            System.out.println("Not Found");
        }
    }

    public static int binarySearch(int[] list, int key) {
        int low = 0;
        int high = list.length - 1;
        while (high >= low) {
            int mid = (low + high) / 2;
            if (key < list[mid])
                high = mid - 1;
            else if (key == list[mid])
                return mid;
            else
                low = mid + 1;
        }
        return -1 - low;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

Found

6.5.2 การประยุกต์การใช้ arrays ในการเรียงลำดับข้อมูล เมื่อประกาศอาร์เรย์และหากต้องการเรียงลำดับข้อมูลที่ต้องการจะสามารถทำได้ด้วยวิธีการดังตัวอย่างด้านล่าง

ตัวอย่างที่ 6.23 การเรียงลำดับข้อมูล

```
import java.util.*;
public class ExArray23 {
    public static void main(String args[]) {
        double[] list = {1, 4, 4, 2, 5, -3, 6, 2};
        System.out.println("Before sorting:");
        for (int i=0; i<list.length; i++) {
            System.out.print(list[i]+" ");
        }
        selectionSort(list);
        System.out.println("\nAfter sorting:");
        for (int i=0; i<list.length; i++) {
            System.out.print(list[i]+" ");
        }
    }
}
```

```

public static void selectionSort(double[] list) {
    for (int i = list.length - 1; i >= 1; i--) {
        double currentMax = list[0];
        int currentMaxIndex = 0;
        for (int j = 1; j <= i; j++) {
            if (currentMax < list[j]) {
                currentMax = list[j];
                currentMaxIndex = j;
            }
        }
        if (currentMaxIndex != i) {
            list[currentMaxIndex] = list[i];
            list[i] = currentMax;
        }
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

Before sorting:

1.0 4.0 4.0 2.0 5.0 -3.0 6.0 2.0

After sorting:

-3.0 1.0 2.0 2.0 4.0 4.0 5.0 6.0

ตัวอย่างที่ 6.24 การเรียงลำดับข้อมูล

```

import java.util.*;
public class ExArray24{
    public static void main(String args[]){
        int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
        System.out.println("Before sorting:");
        for(int i=0;i<list.length;i++){
            System.out.print(list[i]+" ");
        }
        insertionSort(list);
        System.out.println("\nAfter sorting:");
        for(int i=0;i<list.length;i++){
            System.out.print(list[i]+" ");
        }
    }
    public static void insertionSort(int array[]) {
        int n = array.length;
        for (int j = 1; j < n; j++) {
            int key = array[j];
            int i = j-1;
            while ( (i > -1) && ( array [i] > key ) ) {
                array [i+1] = array [i];
                i--;
            }
            array[i+1] = key;
        }
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

Before sorting:

1 4 4 2 5 -3 6 2

After sorting:

-3 1 2 2 4 4 5 6

ตัวอย่างที่ 6.25 การเรียงลำดับข้อมูล

```
import java.util.*;
public class ExArray25{
    public static void main(String args[]){
        double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
        java.util.Arrays.sort(numbers);
        for(int i=0;i< numbers.length;i++){
            System.out.print(numbers[i]+" ");
        }
        System.out.println("");
        char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
        java.util.Arrays.sort(chars);
        for(int i=0;i< chars.length;i++){
            System.out.print(chars [i]+" ");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
1.9 2.9 3.4 3.5 4.4 6.0
4 A D F P a
```

6.6 บทสรุป

อาเรย์คือโครงสร้างข้อมูลที่มีชนิดข้อมูลประเภทเดียวกัน ประเภทของตัวแปรชุด อาจแบ่งตามลักษณะของจำนวนตัวเลขของดัชนีคือ

1. ตัวแปรชุด 1 มิติ (one dimension arrays หรือ single dimension arrays) เป็นตัวแปรชุดที่มีตัวเลขแสดงขนาดเป็นเลขตัวเดียว เช่น word[20] ,num[25] , x[15]
2. ตัวแปรชุดหลายมิติ (multi-dimension arrays) เป็นตัวแปรชุดที่ชื่อมีตัวเลขแสดงขนาดเป็นตัวเลขหลายตัว ที่นิยมใช้กันมี 2 มิติกับ 3 มิติ

2.1 ตัวแปรชุด 2 มิติมีเลขแสดงขนาด 2 ตัว เช่น a[3][5] , name[5][6]

2.2 ตัวแปรชุด 3 มิติมีเลขแสดงขนาด 3 ตัว เช่น a[3][5][6] , name[5][6][8]

6.7 แบบฝึกหัดปฏิบัติการ

1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสร้างอาร์เรย์จากตัวอย่างต่อไปนี้

```
public class MinMaxDemo{
    public static void main(String[] args){
        int a[]={-128, 65, -235, 99, 0, 26};
        int minIndex= findMinIdx(a);
        //int maxIndex= findMaxIdx(a);
        System.out.println("min value is a["+minIndex+"]="+a[minIndex]);
        //System.out.println("max value is a["+maxIndex+"]="+a[maxIndex]);

    }
    public static int findMinIdx(int[] a){
        int k, minIdx=0;
        for(k=1;k<a.length;k++){
            if(a[k]<a[minIdx])
            {
                minIdx=k;
            }
        }
        return minIdx;
    }
}
```

1.1 ผลลัพธ์ของโปรแกรมคือ

1.2 ให้อธิบายการทำงานของ public static int findMinIdx(int[] a)

1.3 ให้เพิ่มการทำงานของเมธอด public static int findMaxIdx(int[] a) สำหรับหาตำแหน่ง index ของอาร์เรย์ที่มีค่ามากที่สุด

2. จงอธิบายว่าเหตุใดโปรแกรมด้านล่างจึง compiles ไม่ผ่าน

โปรแกรม	ผลลัพธ์ของโปรแกรม
<pre> public class Test { public static void main(String[] args) { double[100] r; for (int i = 0; i < r.length(); i++); r(i) = Math.random * 100; } } </pre>	
<pre> public class Test { public static void main(String[] args) { int list[] = {1, 2, 3, 4, 5, 6}; for (int i = 1; i < list.length; i++) list[i] = list[i - 1]; for (int i = 0; i < list.length; i++) System.out.print(list[i] + " "); } } </pre>	
<pre> public class Test { public static void main(String[] args) { int number = 0; int[] numbers = new int[1]; m(number, numbers); System.out.println("number is " + number + " and numbers[0] is " + numbers[0]); } public static void m(int x, int[] y) { x = 3; y[0] = 3; } } </pre>	
<pre> public class Test { public static void main(String[] args) { int[] list = {1, 2, 3, 4, 5}; reverse(list); for (int i = 0; i < list.length; i++) System.out.print(list[i] + " "); } public static void reverse(int[] list) { int[] newList = new int[list.length]; for (int i = 0; i < list.length; i++) newList[i] = list[list.length - 1 - i]; list = newList; } } </pre>	
<pre> public class Test { public static void main(String[] args) { int[][] array = {{1, 2}, {3, 4}, {5, 6}}; for (int i = array.length - 1; i >= 0; i--) { for (int j = array[i].length - 1; j >= 0; j--) System.out.print(array[i][j] + " "); System.out.println(); } } } </pre>	

```

public class Test {
    public static void main(String[] args) {
        int[][] array = {{1, 2, 3, 4}, {5, 6, 7, 8}};
        System.out.println(m1(array)[0]);
        System.out.println(m1(array)[1]);
    }
    public static int[] m1(int[][] m) {
        int[] result = new int[2];
        result[0] = m.length;
        result[1] = m[0].length;
        return result;
    }
}

```

3. จากโปรแกรมต่อไปนี้ให้แสดงผลลัพธ์ของโปรแกรมในแต่ละรอบของการทำงานของ Loop i พร้อมแสดงผลลัพธ์สุดท้าย

```

1 class Test2{
2     public static void main(String[] args){
3         int[] list= {1,9,3,7,2};
4         list=dosomething(list);
5     }
6     public static int[] dosomething(int[] input){
7         int temp;
8         for (int i = 1; i < input.length; i++) {
9             for(int j = i ; j > 0 ; j--){
10                if(input[j] < input[j-1]){
11                    temp = input[j];
12                    input[j] = input[j-1];
13                    input[j-1] = temp;
14                }
15            }
16        }
17        for(int i=0;i<input.length;i++){
18            System.out.print(input[i]+" ");
19        }
20        return input;
21    }
22 }

```

ค่าในอาร์เรย์ list เริ่มต้น

ภายในเมธอด public static int[] dosomething(int[] input(

i=

ค่าในอาร์เรย์ input

--	--	--	--	--

i=

ค่าในอาร์เรย์ input

--	--	--	--	--

i=

ค่าในอาร์เรย์ input

--	--	--	--	--

i=

ค่าในอาร์เรย์ input

--	--	--	--	--

i=

ค่าในอาร์เรย์ input

--	--	--	--	--

ผลลัพธ์สุดท้ายของโปรแกรม

--

4. กำหนดให้ A[0...n-1] เป็นชุดข้อมูลที่เป็นจำนวนเต็ม n จำนวน จงเขียนโปรแกรมแบบ OOP ในการเรียงข้อมูล A[0...n-1] จากน้อยไปมาก พร้อมทั้งหาความถี่สะสมของข้อมูลแต่ละตัว ตัวอย่างเช่น A= [9 5 9 5 8]

ข้อมูลที่เรียงจากน้อยไปมาก	5	8	9
ความถี่ของข้อมูลแต่ละตัว	2	3	5

ให้นักศึกษานิยาม Class ชื่อ AscendSortFreq ที่ประกอบไปด้วย

Data fields ชนิด double[] A ซึ่งแสดงชุดข้อมูล จำนวน n ตัว

Constructor ที่กำหนดค่าให้แก่ double[] A จำนวน n ตัว

Method AscendSort(double[] A) ที่ return array B ที่เรียงจากน้อยไปมาก

Method SortCommuFreq(double[] B) ที่ return array C ความถี่ของข้อมูลแต่ละตัว

5. [MatrixMultiplication] ให้เขียน class MatrixMultiplication ซึ่งทำการคำนวณหาผลคูณของสองเมทริกซ์ การคูณเมทริกซ์ทำได้เมื่อจำนวนคอลัมน์ของเมทริกซ์แรกมีค่าเท่ากับจำนวนแถวของเมทริกซ์ที่สอง ดังนั้นถ้าเมทริกซ์ A เป็นเมทริกซ์ขนาด NxL และเมทริกซ์ B มีขนาดเป็น LxM แล้วผลลัพธ์ของการคูณเมทริกซ์คือ C=AxB จะมีขนาดเป็น NxM โดยที่สมาชิกแต่ละตัวของเมทริกซ์ C มีค่าดังสมการต่อไปนี้

$$c_{ik} = a_{i1}b_{1k} + a_{i2}b_{2k} + a_{i3}b_{3k} + \dots + a_{iL}b_{Lk}$$

ตัวอย่าง ให้ $A = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 3 & 2 \end{bmatrix}, B = \begin{bmatrix} 1 & 5 & 2 \\ -2 & 0 & 1 \end{bmatrix}$

วิธีทำ $AB = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 1 & 5 & 2 \\ -2 & 0 & 1 \end{bmatrix}$

ใช้หลักการแถว คูณ หลัก

$$= \begin{bmatrix} (1)(1) + (2)(-2) & (1)(5) + (2)(0) & (1)(2) + (2)(1) \\ (-1)(1) + (0)(-2) & (-1)(5) + (0)(0) & (-1)(2) + (0)(1) \\ (3)(1) + (2)(-2) & (3)(5) + (2)(0) & (3)(2) + (2)(1) \end{bmatrix}$$

$$= \begin{bmatrix} -3 & 5 & 4 \\ -1 & -5 & -2 \\ -1 & 15 & 8 \end{bmatrix}$$

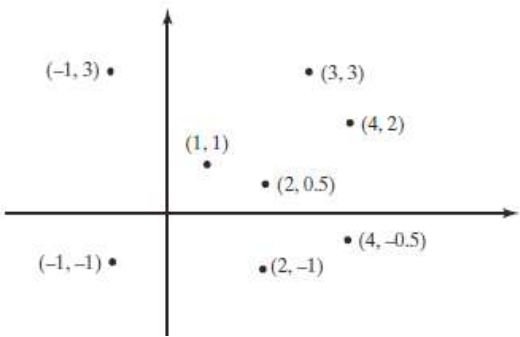
ข้อมูลนำเข้า ขนาดของเมทริกซ์ A และข้อมูลในเมทริกซ์ A

ขนาดของเมทริกซ์ B และข้อมูลในเมทริกซ์ B

ข้อมูลส่งออก ผลคูณของสองเมทริกซ์

ข้อมูลนำเข้า	ข้อมูลส่งออก
3 2	-3 5 4
1 2	-1 -5 -2
-1 0	-1 15 8
3 2	
2 3	
1 5 2	
-2 0 1	

6. จงเขียนโปรแกรมเพื่อคำนวณหาคู่จุดที่ใกล้เคียงกันมากที่สุด



ข้อมูลนำเข้า บรรทัดแรกรับจำนวนจุด n จุด
 n บรรทัดถัดไปแสดงพิกัดของจุดแต่ละจุด

ข้อมูลส่งออก ระยะทางคู่จุดที่ใกล้เคียงกันมากที่สุด

ข้อมูลนำเข้า	ข้อมูลส่งออก
8 3 3 3 1- 1 1 2 4 0.5 2 1- 1- 0.5- 4 2 -1	1.12

7. minTwoSet

กำหนดให้ จำนวนเต็ม n จำนวน โดยมีอยู่ m จำนวนอยู่ในกลุ่ม A และ $n-m$ คนอยู่ในกลุ่ม B จงเขียนโปรแกรมหาคำตอบของผลต่างของผลรวมในแต่ละกลุ่มน้อยที่สุดที่สามารถหาได้จากสมการ

$$\min \left(\left| \sum_{i=1, i \in A}^m w_i - \sum_{j=1, j \in B}^{n-m} w_j \right| \right)$$

โดย $w_i \in n, w_j \in n$ และ $| \quad |$ คือ ค่าสัมบูรณ์ เช่น $|-3| = 3$ หรือ $|3| = 3$

เขียนโปรแกรมหาค่าผลต่างของผลรวมในแต่ละกลุ่มน้อยที่สุด (เขียนแบบ OOP)

ข้อมูลนำเข้า อ่านจาก Standard Input

บรรทัดแรก ระบุ จำนวน n โดย $3 \leq n < 1000$

บรรทัดสอง ระบุค่า w_i จำนวนเต็มบวก n จำนวนที่ไม่ซ้ำกัน โดยแต่ละจำนวนคั่นด้วยช่องว่าง

ข้อมูลส่งออก ส่งออกไปยัง Standard Output

แสดงผลลัพธ์ของผลต่างของผลรวมในแต่ละกลุ่มน้อยที่สุด

ตัวอย่างข้อมูล

ข้อมูลนำเข้า	ข้อมูลส่งออก
4 3 101 99 1	0
5 14 16 47 25 2	6
6 7 2 1 3 6 0	1

8. [Find Pokemon] ญาญ่าผู้เล่นหน้าใหม่ของเกมสไปเกมอนต้องการจับตัวปิกาจู เนื่องจากปิกาจูเป็นตัวที่หาได้ยากมาก ดังนั้นญาญ่าจึงไปรวบรวมสถิติการเกิดตัวโปเกมอน ณ พิกัดต่าง ๆ โดยพบว่าในแต่ละพิกัดจะมีความถี่ของการเกิดตัวโปเกมอนตัวต่าง ๆ ที่ไม่เท่ากัน โดยมีการเก็บข้อมูลเป็นรูปขนาด $H \times W$ ช่อง โดยญาญ่าต้องการหาปิกาจูจากรูปนี้ ตัวอย่างของรูปขนาด 4×5 แสดงเป็นตารางด้านล่าง กำหนดตารางชื่อ A ตัวเลขในแต่ละช่องแสดงความถี่ของการเกิดตัวโปเกมอนในช่องนั้น

5	1	2	10	4
4	30	3	0	100
3	25	10	4	10
3	20	4	8	5

ในการหาตำแหน่งของปิกาจูจะมีเงื่อนไข 3 ข้อดังนี้

1. ปิกachuจะปรากฏเป็น 2 ช่องติดกันพอดี
2. สองช่องที่เป็นบริเวณที่มีปิกachuควรมีค่าความถี่ของการเกิดตัวโปเกมอนในช่องนั้น ต่างกันไม่เกิน 10
3. ตำแหน่งของปิกachuน่าจะเป็นตำแหน่งที่มีความถี่ของการเกิดตัวโปเกมอนสูงก็ต้องเป็นสองช่องที่มีผลรวมของค่าความถี่ของการเกิดตัวโปเกมอนมากที่สุด

จากตารางตำแหน่งที่ตรงตามเงื่อนไขคือ $A[2][2]$ และ $A[3][2]$ จึงเขียนโปรแกรมที่รับตารางแสดงค่าความถี่ของการเกิดตัวโปเกมอน จากนั้นให้หาตำแหน่งมุมบนซ้ายของช่องที่น่าจะเกิดปิกachuมากที่สุด โดยระบุแถวและคอลัมน์ช่องนั้น

ข้อมูลนำเข้า

บรรทัดแรก ระบุขนาดตาราง HxW

บรรทัดที่ 2 ถึง H+1 แสดงค่าความถี่ของการเกิดตัวโปเกมอนในแถวที่ i โดยระบุเป็นจำนวนเต็มจำนวน W ตัว จำนวนที่ j จะเป็นค่าความถี่ของการเกิดตัวโปเกมอนในช่องที่อยู่ในคอลัมน์ j

ข้อมูลส่งออก

มีบรรทัดเดียว คือ มุมบนซ้ายของช่องที่น่าจะเกิดตัวปิกachuมากที่สุดโดยระบุแถวและคอลัมน์ช่องนั้น

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
4 5 5 1 2 10 4 4 30 3 0 100 3 25 10 4 10 3 20 4 8 5	2 2
4 4 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0	3 2

9. จงเขียนเมธอด `isConsecutiveFour(int[][] values)` เพื่อตรวจสอบอาร์เรย์สองมิติต่อไปนี้ว่ามีตัวเลขตัวเดียวกันเรียงต่อกันครบสี่ตัวหรือไม่ ในแนวตั้ง แนวนอน หรือแนวทะแยง

```
public static boolean isConsecutiveFour(int[][] values)
```

จากนั้นให้เขียนโปรแกรมทดสอบที่ผู้ใช้สามารถป้อนจำนวนแถวและคอลัมน์ของอาร์เรย์สองมิติและป้อนค่าภายในอาร์เรย์ โดยเมื่อป้อนค่าไปแล้วให้เรียกใช้เมธอด `isConsecutiveFour(int[][] values)` โดยการคืนค่า `true` กรณีที่ภายในอาร์เรย์มีตัวเลข ตัวเรียงต่อกัน กรณีอื่นให้คืน `false`

ตัวอย่าง เมื่อส่งอาร์เรย์ต่อไปนี้เข้าไปที่เมธอดจะคืนค่า `true` ออกมา

0 1 0 3 1 6 1	0 1 0 3 1 6 1	0 1 0 3 1 6 1	0 1 0 3 1 6 1
0 1 6 8 6 0 1	0 1 6 8 6 0 1	0 1 6 8 6 0 1	0 1 6 8 6 0 1
5 6 2 1 8 2 9	5 5 2 1 8 2 9	5 6 2 1 6 2 9	9 6 2 1 8 2 9
6 5 6 1 1 9 1	6 5 6 1 1 9 1	6 5 6 6 1 9 1	6 9 6 1 1 9 1
1 3 6 1 4 0 7	1 5 6 1 4 0 7	1 3 6 1 4 0 7	1 3 9 1 4 0 7
3 3 3 3 4 0 7	3 5 3 3 4 0 7	3 6 3 3 4 0 7	3 3 3 9 4 0 7

ข้อมูลนำเข้า จำนวนแถวและคอลัมน์ของอาร์เรย์สองมิติและป้อนค่าภายในอาร์เรย์

ข้อมูลส่งออก คินค่า 1 กรณีที่ภายในอาร์เรย์มีตัวเลข 0 ตัวเรียงต่อกัน กรณีอื่นให้คินค่าเป็น 4

ข้อมูลนำเข้า	ข้อมูลส่งออก
7 6 0 1 0 3 1 6 1 0 1 6 8 6 0 1 5 6 2 1 8 2 9 6 5 6 1 1 9 1 1 3 6 1 4 0 7 3 3 3 3 4 0 7	1

10. (Car) ขับรถหลบสิ่งกีดขวาง

ในการแสดงขับรถผาดโผนบนถนนที่มีเลนทั้งหมด m เลน โดยให้หมายเลขประจำเลนจากซ้ายไปขวามีค่าตั้งแต่ 1 จนถึง m ตามลำดับ นักแสดงขับรถผาดโผนต้องบังคับรถให้แล่นไปบนถนนดังกล่าวให้ปลอดภัยตลอดระยะเวลา t หน่วย การแสดงเริ่มต้น ณ เวลา 0 ถึง t นักแสดงขับรถผาดโผนอยู่ในเลนที่ k ในแต่ละ 1 หน่วยเวลา อาจมีสิ่งกีดขวางตกลงมายังถนนบางเลน ทำให้เขาต้องบังคับรถเพื่อหลีกเลี่ยงสิ่งกีดขวาง ซึ่งมีทางเลือกในการบังคับรถอยู่ 3 แบบ ได้แก่ 1 หมายถึง การเปลี่ยนเลนไปทางซ้าย 1 เลนในเวลาถัดไปไปยังเลนที่มีหมายเลขประจำเลนน้อยกว่า 2 หมายถึงการเปลี่ยนเลนไปทางขวา 1 เลนในเวลาถัดไป (ไปยังเลนที่มีหมายเลขประจำเลนมากกว่า) และ 3 หมายถึง การขับอยู่ในเลนเดิม กำหนดให้ถนนเป็นเส้นตรงตลอดทาง จงเขียนโปรแกรมเพื่อบังคับให้รถแล่นไปตามเส้นทางนี้โดยปลอดภัย โดยชุดข้อมูลทดสอบจะมีคำตอบที่ถูกต้องเพียง 1 คำตอบเสมอ

ข้อมูลนำเข้า

1. บรรทัดแรกระบุจำนวนเลน m โดยที่ $2 \leq m \leq 40$
2. บรรทัดที่สองระบุหมายเลขเลนเริ่มต้น $1 \leq n \leq m$
3. บรรทัดที่สามระบุระยะเวลา t โดยที่ $1 \leq t \leq 100$
4. บรรทัดที่สี่ถึงบรรทัดที่ $t+3$ แสดงสถานะของถนน ณ เวลา $t=1, 2, \dots, K$ ตามลำดับ แต่ละบรรทัดระบุตัวเลข m ตัวเลขแต่ละตัวแสดงสถานะของถนนตั้งแต่เลนที่ 1 ถึงเลนที่ m โดยเลข 0 หมายถึงเลนนั้นไม่มีสิ่งกีดขวาง และเลข 1 หมายถึงมีสิ่งกีดขวางอยู่

ข้อมูลส่งออก

มีอยู่ t บรรทัด แต่ละบรรทัดมีตัวเลข 1 ตัวเพื่อแสดงถึงทางเลือกในการบังคับรถของนักแสดงขับรถผาดโผน ในแต่ละช่วงเวลา บรรทัดที่ i หมายถึงการเปลี่ยนเลนจากเวลาที่ $i-1$ ไปยังเวลาที่ i เมื่อ $i=1, 2, \dots, t$ โดยที่เลข 1 จะหมายถึงขับไปทางซ้าย 1 เลน, เลข 2 หมายถึงขับไปทางขวา 1 เลน, และเลข 3 หมายถึงขับอยู่ในเลนเดิม

ตัวอย่างที่ 1

ข้อมูลนำเข้า	ข้อมูลส่งออก
7	1
5	1
5	1
0 0 0 0 0 0 0	1
0 0 0 0 0 0 0	2
0 0 0 0 0 0 0	
0 1 1 0 0 0 0	
1 0 1 1 1 1 1	

11. [Oil Deposits] บ่อน้ำมัน

นักธรณีวิทยาต้องการสำรวจหาแหล่งน้ำมันบนพื้นที่สี่เหลี่ยมขนาดใหญ่เรียกว่า GRID โดยบริษัทได้ตีเส้นใน GRID ให้อยู่ในรูปของตาราง และทำการสำรวจโดยใช้เครื่องมือสำหรับตรวจจับน้ำมันว่าพื้นที่สี่เหลี่ยมที่สำรวจมีน้ำมันอยู่หรือไม่ บริเวณที่มีน้ำมันจะเรียกว่า pocket ถ้ามี pocket ในตารางแต่ละช่องต่อกันไม่ว่าจะเป็นแนวตั้ง แนวนอน แนวทแยง จะถือว่าเป็นบ่อน้ำมันบ่อเดียวกัน งานที่ต้องทำคือให้ตอบคำถามว่าในพื้นที่ พื้นที่ขนาดใหญ่ 1GRID มีบ่อน้ำมันกี่บ่อ

ข้อมูลนำเข้า

ข้อมูลนำเข้าประกอบด้วย 1 หรือมากกว่า 1GRID แต่ละ GRID จะเริ่มต้นด้วยตัวเลข m และ n ซึ่งแทนด้วยจำนวนแถวและจำนวนคอลัมน์ใน GRID กรณีที่ $m=0$ แสดงว่าไม่มีการรับข้อมูลต่อ กำหนดให้ $1 \leq m \leq 100$ และ

$1 \leq n \leq 100$ แถวต่อไปจำนวน m แถวจะมีแถวละ n ตัวอักษร แต่ละตัวอักษรแทนด้วยสถานะของน้ำมัน ณ ตารางดังกล่าวโดย '*' แทนไม่มีน้ำมัน '@' แทน oil pocket.

ข้อมูลส่งออก

สำหรับแต่ละ GRID จะแสดงจำนวนของบ่อน้ำมันที่ค้นพบ

ข้อมูลนำเข้า	ข้อมูลส่งออก
1 1 *	0
3 5 * @ * @ * **@** *@**@	1
1 8 @@*****@	2
5 5 *****@ *@@*@ *@**@ @@@*@ @@**@	2

บทที่ 7 การสืบทอดคุณสมบัติ

วัตถุประสงค์

- 7.1 สามารถอธิบายความหมายของการสืบทอด คีย์เวิร์ด *super Constructor Chaining*
- 7.2 สามารถอธิบายความหมายของ โอเวอร์โหลดดิ้ง
- 7.3 สามารถเขียนโปรแกรมเพื่อประยุกต์การสืบทอดคุณสมบัติ

7.1 ความนำ

รูปแบบการเขียนโปรแกรมเชิงวัตถุจะอนุญาตให้สามารถสร้างคลาสใหม่จากคลาสเดิมที่เคยมีอยู่แล้วซึ่งเรียกเทคนิคนี้ว่าการสืบทอด (Inheritance)

การสืบทอด (Inheritance) เป็นเทคนิคที่สำคัญมากสำหรับการนำซอฟต์แวร์กลับมาใช้ใหม่ (Reuse software) เช่น สมมติต้องการคลาสวงกลม สี่เหลี่ยมและสามเหลี่ยม คลาสต่าง ๆ เหล่านี้จะมีคุณลักษณะบางอย่างที่คล้ายกัน การออกแบบคลาสเพื่อไม่ให้ใช้ตัวแปรซ้ำ และทำให้ง่ายต่อการแก้ไข วิธีที่ดีที่สุดคือการใช้หลักการของการสืบทอดข้อมูล

7.2 นิยามของการสืบทอด (Inheritance)

เทคนิคการสืบทอดข้อมูลอนุญาตให้กำหนดคลาสทั่วไปที่เรียกว่า General class หรือ Super class และภายหลังสามารถขยาย (extends) คลาสดังกล่าวไปเป็นคลาสที่มีความจำเพาะเจาะจงได้ (specialized class หรือ sub class)

เราสามารถใช้คลาสเพื่อจำลองวัตถุที่มีชนิดข้อมูลเดียวกันได้ คลาสต่าง ๆ กันอาจจะมีคุณสมบัติและพฤติกรรมที่เหมือนกัน ที่สามารถทำให้อยู่ในรูปทั่วไปได้และสามารถใช้ร่วมกับคลาสอื่น ๆ ได้ เรายังสามารถกำหนดคลาสที่มีคุณสมบัติจำเพาะ ที่สืบทอดหรือขยายมาจากคลาสทั่วไปหรือ Generalized class ได้โดยที่คลาสที่มีความจำเพาะเหล่านั้นสืบทอดคุณสมบัติและเมธอดมาจากคลาสทั่วไป เช่น ต้องการสร้างคลาสที่มีความสามารถในการคำนวณตัวเลข โดยสามารถบวกและลบตัวเลขได้ เราสามารถสร้างคลาส Calculation โดยมีเมธอดที่ใช้ในการบวก `public void addition(int x, int y)` เมธอดที่ใช้ในการลบ `public void subtraction(int x,int y)` อยู่ภายในคลาสดังกล่าว หากปรากฏว่าในภายหลังผู้ใช้ต้องการสร้างคลาสใหม่โดยเป็นคลาสที่ใช้ในการคำนวณโดยมีการบวกลบและการคูณอยู่ภายใน ผู้เขียนโปรแกรมสามารถสร้างคลาสใหม่โดยการนำความสามารถของคลาสเดิมมาขยายการทำงานเพิ่มมากขึ้นโดยสามารถสร้างคลาส `My_Calculation extends Calculation` และเพิ่มเมธอด `public void multiplication(int x, int y)` ภายในคลาสดังกล่าวได้โดยภายในคลาส `My_Calculation` จะมีเมธอดที่สามารถเรียกใช้เพื่อบวก ลบ คูณ ได้จากความสามารถของการสืบทอดคลาสดังกล่าว

ตัวอย่างที่ 7.1 การสร้างคลาสใหม่โดยการนำความสามารถของคลาสเดิมมาขยายการทำงานเพิ่มมากขึ้น

```
class Calculation{
    int z;
    public void addition(int x, int y){
        z=x+y;
        System.out.println("The sum of the given numbers:"+z);
    }
    public void Substraction(int x,int y){
        z=x-y;
        System.out.println("The difference between the given numbers:"+z);
    }
}
public class My_Calculation extends Calculation{
    public void multiplication(int x, int y){
        z=x*y;
        System.out.println("The product of the given numbers:"+z);
    }
    public static void main(String args[]){
        int a=20, b=10;
        My_Calculation demo = new My_Calculation();
        demo.addition(a, b);
        demo.Substraction(a, b);
        demo.multiplication(a, b);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

The sum of the given numbers:30
The difference between the given numbers:10
The product of the given numbers:200

7.3 การใช้คีย์เวิร์ด super และการใช้คีย์เวิร์ด this

คีย์เวิร์ด super ใช้อ้างถึงแอตทริบิวต์และเมธอดของ super class และสามารถใช้สำหรับเรียกคอนสตรักเตอร์ของ super class ได้ โดยที่ subclass สามารถสืบทอดข้อมูลและเมธอดที่สามารถเข้าถึงได้จาก super class เราสามารถใช้ super ในการอ้างอิงโดยมีรูปแบบการใช้งานหลัก ๆ รูปแบบคือ 2

1. ใช้เรียกคอนสตรักเตอร์ของ super class
2. ใช้เรียกเมธอดของ super class

การเรียกใช้คอนสตรักเตอร์ของ superclass โดยใช้คีย์เวิร์ด super ต้องเรียกใช้เป็นบรรทัดแรกภายในคอนสตรักเตอร์ของ subclass หากคอนสตรักเตอร์ของ subclass ไม่มีการเรียกใช้คอนสตรักเตอร์ของ superclass โปรแกรมจะเรียกใช้ super โดยอัตโนมัติ ตัวอย่างของการเรียกใช้ super มีรายละเอียดดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 7.2 การใช้คีย์เวิร์ด super และการใช้คีย์เวิร์ด this

```
public class Point {
    private int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public Point() {
        x = 0;
        y = 0;
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
    public String toString() {
        return "(" + x + "," + y + ")";
    }
}

class Point3D extends Point {
    private int z;
    public Point3D() {
        super();
        z = 0;
    }
    public Point3D(int x, int y, int z) {
        super(x, y);
        this.z = z;
    }
    public int getZ() {
        return z;
    }
    public void setZ(int z) {
        this.z = z;
    }
    @Override
    public String toString() {
        return "(" + super.getX() + "," + super.getY() + "," + z + ")";
    }
}
```

```

    }
}
class TestPoint3D {
    public static void main(String[] args) {
        Point3D p1 = new Point3D(1, 2, 3);
        System.out.println(p1);
        System.out.println(p1.getX());
        System.out.println(p1.getY());
        System.out.println(p1.getZ());
        p1.setX(4);
        p1.setY(5);
        p1.setZ(6);
        System.out.println(p1);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

(1, 2, 3)
1
2
3
(4, 5, 6)

```

การใช้คีย์เวิร์ด this

คีย์เวิร์ด this ใช้อ้างถึงแอตทริบิวต์และเมธอดของคลาส และสามารถใช้สำหรับเรียกโอเวอร์โหลดคอนสตรักเตอร์ของคลาสได้

ตัวอย่างที่ 7.3 การใช้คีย์เวิร์ด this

```

class Test
{
    int x = 1;
    public void print()
    {
        int x = 2;
        System.out.println(this.x);
        System.out.println(x);
    }
    public static void main(String[] args)
    {
        new Test().print();
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

1
2

```

ตัวอย่างที่ 7.4 การใช้คีย์เวิร์ด this

```

class Person
{
    int id;
    String name;
    public Person()
    {
        this(0, "Unknown");
    }

    public Person(int id, String name)
    {
        this.id = id;
        this.name = name;
    }

    public void print()

```

```

{
    System.out.println("ID: " + this.id);
    System.out.println("Name: " + this.name);
    System.out.println();
}

public static void main(String[] args)
{
    Person p1 = new Person();
    Person p2 = new Person(001, "Yaya");
    p1.print();
    p2.print();
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

ID: 0
Name: Unknown

ID: 1
Name: Yaya

```

7.4 Constructor Chaining

คอนสตรัคเตอร์ (Constructor) ใช้สำหรับสร้าง instance หรือวัตถุของคลาส คอนสตรัคเตอร์ของ super class จะไม่ถูกสืบทอดโดย subclass แต่สามารถเรียกคอนสตรัคเตอร์ของ superclass จาก subclass โดยการใช้ super โดยรูปแบบในการเรียกใช้คอนสตรัคเตอร์ของ super class คือ

super () หรือ **super (parameter)**

ในขณะที่กำลังสร้าง Instance ของคลาสจะมีการเรียกคอนสตรัคเตอร์ของ superclass ทั้งหมดที่อยู่ในโครงสร้างการสืบทอดเดียวกันเราเรียกลักษณะการทำงานรูปแบบนี้ว่า constructor chaining

ตัวอย่างที่ 7.5 การเรียกคอนสตรัคเตอร์ของ superclass ทั้งหมดที่อยู่ในโครงสร้างการสืบทอดเดียวกัน

```

public class Faculty extends Employee {
    public static void main(String[] args) {
        new Faculty();
    }
    public Faculty() {
        System.out.println("(4) Faculty's no-arg constructor is invoked");
    }
}
class Employee extends Person {
    public Employee() {
        this("(2) Invoke Employee's overloaded constructor");
        System.out.println("(3) Employee's no-arg constructor is invoked");
    }
    public Employee(String s) {
        System.out.println(s);
    }
}
class Person {
    public Person() {
        System.out.println("(1) Person's no-arg constructor is invoked");
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

- (1) Person's no-arg constructor is invoked
- (2) Invoke Employee's overloaded constructor
- (3) Employee's no-arg constructor is invoked
- (4) Faculty's no-arg constructor is invoked

7.5 การเรียกใช้เมธอดของ superclass

เราสามารถใส่คีย์เวิร์ด `super` สำหรับอ้างอิงถึงเมธอดนอกเหนือจากการเรียกคอนสตรัคเตอร์ใน superclass โดยมีรูปแบบในการเขียนดังนี้

```
super.method(parameters);
```

ตัวอย่างที่ 7.6 การเรียกใช้เมธอดของ superclass

```
class Student
{
    int id;
    String name;
    double gpa;
    public Student(int id, String name)
    {
        setId(id);
        setName(name);
        this.gpa = 0.0;
    }
    public Student(int id, String name, double gpa)
    {
        setId(id);
        setName(name);
        setGpa(gpa);
    }
    public void setId(int id)
    {
        if (id > 0)
            this.id = id;
    }
    public void setName(String name)
    {
        if (name != null && name.length() > 0)
            this.name = name;
    }
    public void setGpa(double gpa)
    {
        if (gpa >= 0.0 && gpa <= 4.00)
            this.gpa = gpa;
    }
}

class BachelorStudent extends Student
{
    String studentClub;
    public BachelorStudent(int id, String name)
    {
        super(id, name);
    }
    public BachelorStudent(int id, String name, String studentClub)
    {
        super(id, name);
        setStudentClub(studentClub);
    }
    public BachelorStudent(int id, String name, String studentClub, double gpa)
    {

```



```

        super(id, name, gpa); // called overloaded constructor in superclass
        setStudentClub(studentClub);
    }
    public void setStudentClub(String studentClub)
    {
        if (studentClub != null && studentClub.length() > 0)
            this.studentClub = studentClub;
    }
}
public class Main
{
    public static void main(String[] args)
    {
        BachelorStudent cherprang = new BachelorStudent(64334952, "Cherprang Jaidee",
"Bridge Club", 4.00);
        System.out.println("ID: " + cherprang.id);
        System.out.println("Name: " + cherprang.name);
        System.out.println("Bridge Club: " + cherprang.studentClub);
        System.out.printf("GPA: %.2f%n", cherprang.gpa);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

ID: 64334952
Name: Cherprang Jaidee
Bridge Club: Bridge Club
GPA: 4.00

```

7.6 โอเวอร์โหลดและโอเวอร์โหลดดิ่ง

7.6.1 โอเวอร์โหลดดิ่ง (Overriding Methods)

Subclass จะสืบทอดข้อมูลจาก superclass บางครั้งที่ subclass ต้องการแก้ไขรายละเอียดการทำงานของเมธอดที่ประกาศไว้แล้วที่ superclass เราเรียกเทคนิคนี้ว่า Overriding Methods

ตัวอย่าง

```

public class Circle extends GeometricObject {
    //Other methods are omitted
    /**Override the toString method defined in GeometricObject */
    public String toString(){
        return super.toString()+"\nradius is "+radius;
    }
}

```

โอเวอร์โหลดดิ่งเกิดจากคลาสสองคลาสสืบทอดข้อมูลกัน โดยมีเมธอดชื่อเดียวกันพารามิเตอร์เหมือนกันทุกประการแต่อยู่คนละคลาสซึ่ง overridden method นี้เราสามารถแก้ไขชุดคำสั่งภายในเมธอดได้

ตัวอย่างที่ 7.7 โอเวอร์โหลดดิ่ง (Overriding Methods)

```

class A
{
    public void print(String msg)
    {
        System.out.println(msg);
    }
}
class B extends A
{
    @Override
    public void print(String msg)
    {

```

```

        System.out.println("Message: " + msg);
    }
}
public class Ex72
{
    public static void main(String[] args)
    {
        new A().print("A");
        new B().print("B");
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

A
Message: B

```

7.6.2 โอเวอร์โหลดดิ่ง (Overloading Methods)

โอเวอร์โหลดดิ่งเกิดจากการที่มีเมธอดชื่อเดียวกันภายในคลาสเดียวกันแต่มีพารามิเตอร์ต่างกัน ซึ่งอาจจะต่างกันเพราะจำนวนหรือชนิดของพารามิเตอร์ก็ได้ และเนื่องจากคอนสตรัคเตอร์เป็นเมธอดชนิดหนึ่ง ดังนั้นจึงสามารถใช้คุณสมบัติโอเวอร์โหลดดิ่งได้เช่นเดียวกัน ซึ่งการเรียกใช้ overloaded constructor จะใช้คีย์เวิร์ด this

ตัวอย่างที่ 7.8 โอเวอร์โหลดดิ่ง (Overloading Methods)

```

class Student {
    int id;
    String name;
    String surname;
    double gpa;
    Student(int id,String name, String surname) {
        this.id=id;
        this.name=name;
        this.surname=surname;
    }
    Student(int id,String name, String surname,double gpa) {
        this.id=id;
        this.name=name;
        this.surname=surname;
        this.gpa=gpa;
    }
    public void editStudentDetails(int id){
        this.id=id;
    }
    public void editStudentDetails(double gpa){
        this.gpa=gpa;
    }
    public void editStudentDetails(String name, String surname){
        this.name=name;
        this.surname=surname;
    }
    public void editStudentDetails(int id, String name, String surname){
        this.id=id;
        this.name=name;
        this.surname=surname;
    }
    public void editStudentDetails(int id,String name, String surname,double gpa){
        this.id=id;
        this.name=name;
        this.surname=surname;
        this.gpa=gpa;
    }
}

```

ตัวอย่างที่ 7.9 โอเวอร์ไรด์เมธอด (Overriding Methods)

```
public class Ex76
{
    public static void main(String[] args)
    {
        B b = new B();
        b.p(10);
        b.p(10.0);
    }
}
class A{
    public void p(double i){
        System.out.println(i*2);
    }
}
class B extends A{
    public void p(double i){
        System.out.println(i);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
10.0
10.0
```

ตัวอย่างที่ 7.10 โอเวอร์โหลดเมธอด (Overloading Methods)

```
public class Ex78
{
    public static void main(String[] args)
    {
        B b = new B();
        b.p(10);
        b.p(10.0);
    }
}
class A{
    public void p(double i){
        System.out.println(i*2);
    }
}
class B extends A{
    public void p(int i){
        System.out.println(i);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
10
20.0
```

7.7 คลาส Object และเมธอดภายในคลาส Object

ทุก ๆ คลาสในภาษาจาวาจะถูกสืบทอดมาจากคลาส `java.lang.Object` ตัวอย่างคลาสที่สืบทอดมาจากคลาส `Object` เช่น `String`, `StringBuffer`, `StringTokenizer` ภายในคลาส `Object` จะมีเมธอดที่สำคัญที่มีการเรียกใช้งานบ่อย ๆ เช่น

```
public Boolean equals(Object object)
public int hashCode
public String toString()
```

<pre>public class Circle { ... }</pre>	<p>Equivalent</p> <hr style="border: none; border-top: 1px solid black; height: 3px;"/>	<pre>public class Circle extends Object { ... }</pre>
--	---	---

ตัวอย่างการทำงานของเมธอด `toString()` ในคลาส `Object`

เมธอด `toString()` จะคืนค่า `String` จากวัตถุ ค่าดีฟอลต์ที่คืนออกมาคือ `string` ที่อธิบายชื่อ `class` ที่วัตถุถูกสร้างขึ้นหรือคลาสที่วัตถุเป็น instance ตามด้วยเครื่องหมาย `@` และตัวเลขที่แสดงวัตถุนั้น ๆ เช่น

```
Loan loan = new Loan();
System.out.println(loan.toString());
```

ค่าที่คืนออกมาคือ `Loan@15037e5`

ตัวอย่างการทำงานของเมธอด `equals()` ในคลาส `Object`

เมธอด `equals()` จะเปรียบเทียบข้อมูลภายในวัตถุ รายละเอียดของเมธอด `equals` ที่เขียนในคลาส `Object` คือ

```
public boolean equals(Object obj) {
    return (this == obj);
}
public boolean equals(Object o) {
    if (o instanceof Circle) {
        return radius == ((Circle)o).radius;
    }
    else
        return false;
}
```

การเปรียบเทียบโดยใช้เครื่องหมาย `==` จะใช้เปรียบเทียบข้อมูลภายในชนิดข้อมูลแบบ `primitive` หรือตรวจสอบว่าวัตถุ 2 วัตถุมีการอ้างอิงที่ตำแหน่งเดียวกันหรือไม่ ส่วนการใช้เมธอด `equals` จะตรวจสอบว่าวัตถุ 2 วัตถุมีข้อมูลภายในเหมือนกันหรือไม่

การเปรียบเทียบโดยใช้เครื่องหมาย `==` จะมีน้ำหนักมากกว่าเมธอด `equals` เนื่องจากเครื่องหมาย `==` จะตรวจสอบว่าตัวแปรอ้างอิงวัตถุอ้างอิงไปยังวัตถุเดียวกันหรือไม่

7.8 บทสรุป

การเขียนโปรแกรมเชิงวัตถุ จะอนุญาตให้สามารถสร้างคลาสใหม่จากคลาสเดิมที่เคยมีอยู่แล้วซึ่งเรียกเทคนิคนี้ว่าการสืบทอด (Inheritance) การสืบทอด (Inheritance) เป็นเทคนิคที่สำคัญมากสำหรับการนำซอฟต์แวร์กลับมาใช้ใหม่ (Reuse software)

Super class และ Sub class

เทคนิคการสืบทอดข้อมูลอนุญาตให้กำหนดคลาสทั่วไปที่เรียกว่า General class หรือ Super class และภายหลังสามารถขยาย (extends) คลาสดังกล่าวไปเป็นคลาสที่มีความจำเพาะเจาะจงได้ specialized class หรือ sub class เราสามารถใช้คลาสเพื่อจำลองวัตถุที่มีชนิดข้อมูลเดียวกันได้ คลาสต่าง ๆ กันอาจจะมีคุณสมบัติและพฤติกรรมที่เหมือนกัน ที่สามารถทำให้อยู่ในรูปทั่วไปได้และสามารถใช้ร่วมกับคลาสอื่น ๆ ได้ เรายังสามารถกำหนดคลาสที่มีคุณสมบัติจำเพาะ ที่สืบทอดหรือขยายมาจากคลาสทั่วไปหรือ Generalized class ได้โดยที่คลาสที่มีความจำเพาะเหล่านั้นสืบทอดคุณสมบัติและเมธอดมาจากคลาสทั่วไป ตัวอย่าง สมมติต้องการคลาสวงกลม สีเหลี่ยมและสามเหลี่ยม คลาสต่าง ๆ เหล่านี้จะมีคุณลักษณะบางอย่างที่คล้ายกัน การออกแบบคลาสเพื่อไม่ให้ใช้ตัวแปรซ้ำ และทำให้ง่ายต่อการแก้ไข วิธีที่ดีที่สุดคือการใช้หลักการของการสืบทอดข้อมูล โดยการสร้างคลาสที่คลาสทุกคลาสมีคุณสมบัติเดียวกันเรียกว่า Superclass ซึ่งจากตัวอย่างคือ GeometricObject หลังจากนั้นเมื่อต้องการสร้างวงกลม Circle ก็ให้สืบทอดคุณสมบัติจากคลาส Geometric และเพิ่มคุณสมบัติเฉพาะตัวของวงกลมเข้าไป

7.9 แบบฝึกหัดปฏิบัติการ

1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสืบทอดคลาสจากตัวอย่างต่อไปนี้

```
public abstract class GeometricObject {
    private String color = "white";

    private boolean filled;

    /** Default constructor */
    protected GeometricObject() {
    }

    /** Convenience constructor */
    protected GeometricObject(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }

    /**
     * Return the current color value
     * @return color
     */
    public String getColor() {
        return color;
    }

    /**
     * Set a new color value
     */
    public void setColor(String color) {
        this.color = color;
    }

    /**
     * Return the current filled value
     * @return filled
     */
    public boolean isFilled() {
        return filled;
    }

    /**
     * Set a new filled value
     */
    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    /**
     * Calculate the area of the GeometricObject
     * Abstract method subclasses should implement
     * @return area of object
     */
    public abstract double findArea();

    /**
     * Calculate the perimeter of the GeometricObject
     * Abstract method subclasses should implement
     * @return perimeter of object
     */
    public abstract double findPerimeter();
}

public class Circle extends GeometricObject {
    private double radius;

    /** Default constructor */
```

```

public Circle() {
    this(1.0);
}

/** Radius convenience constructor */
public Circle(double radius) {
    this(radius, "white", false);
}

/** Convenience constructor for all properties */
public Circle(double radius, String color, boolean filled) {
    super(color, filled);
    this.radius = radius;
}

/**
 * Return the radius
 * @return radius Current radius of Circle
 */
public double getRadius() {
    return radius;
}

/**
 * Set the radius of the circle
 */
public void setRadius(double radius) {
    this.radius = radius;
}

/**
 * Returns the area of the current circle
 * Implementation of abstract method in GeometricObject
 * @return area of the circle
 */
public double findArea() {
    return radius * radius * Math.PI;
}

/**
 * Returns the perimeter of the current circle
 * Implementation of abstract method in GeometricObject
 * @return perimeter of the circle
 */
public double findPerimeter() {
    return 2 * radius * Math.PI;
}

/**
 * Provide a string representation of the object
 */
public String toString() {
    return "Circle: radius = " + radius;
}
}

```

1.1 ให้อธิบายลำดับการเรียกใช้ constructor เมื่อสร้างวัตถุจากคลาส Circle

1.2 ให้สร้างคลาส Rectangle ตาม UML Class diagram ด้านบน

2. จงหาข้อผิดพลาดของส่วนของโปรแกรมต่อไปนี้

```
public class Circle{
    private double radius;

    public Circle(double radius){
        radius=radius;
    }
    public double getRadius(){
        return radius;
    }
    public double findArea(){
        return radius*radius*Math.PI;
    }
}
class Cylinder extends Circle{
    private double length;

    Cylinder(double radius, double length){
        Circle(radius);
        length=length;
    }
}
```

3. จงหาผลลัพธ์การรันโปรแกรมต่อไปนี้

3.1

```
class A{
    public A(){
        System.out.println("The no-arg constructor of A is
invoked");
    }
}
class B extends A{
    public B(){
    }
}
public class C{
    public static void main(String[] args){
        B b =new B();
    }
}
```

ผลลัพธ์การรันโปรแกรม

3.2

<pre> Animal.java: 01 public class Animal { 02 public Animal() { 03 System.out.println("A new animal has been created!"); 04 } 05 06 public void sleep() { 07 System.out.println("An animal sleeps..."); 08 } 09 10 public void eat() { 11 System.out.println("An animal eats..."); 12 } 13 } </pre>	<pre> Bird.java: 01 public class Bird extends Animal { 02 public Bird() { 03 super(); 04 System.out.println("A new bird has been created!"); 05 } 06 07 @Override 08 public void sleep() { 09 System.out.println("A bird sleeps..."); 10 } 11 12 @Override 13 public void eat() { 14 System.out.println("A bird eats..."); 15 } 16 } </pre>
<pre> Dog.java: 01 public class Dog extends Animal { 02 public Dog() { 03 super(); 04 System.out.println("A new dog has been created!"); 05 } 06 07 @Override 08 public void sleep() { 09 System.out.println("A dog sleeps..."); 10 } 11 12 @Override 13 public void eat() { 14 System.out.println("A dog eats..."); 15 } 16 } </pre>	<pre> MainClass.java: 01 public class MainClass { 02 public static void main(String[] args) { 03 Animal animal = new Animal(); 04 Bird bird = new Bird(); 05 Dog dog = new Dog(); 06 07 System.out.println(); 08 09 animal.sleep(); 10 animal.eat(); 11 12 bird.sleep(); 13 bird.eat(); 14 15 dog.sleep(); 16 dog.eat(); 17 } 18 } </pre>
<p>ผลลัพธ์การรันโปรแกรม</p>	

4. จากโปรแกรมด้านล่างต่อไปนี้ ข้อใดคือการใช้คุณสมบัติ Overriding ที่สามารถนำมาเติมลงในบรรทัดที่ 9 ของโปรแกรมได้โดยไม่ทำให้เกิดข้อผิดพลาดขึ้นกับการทำงานของโปรแกรม อธิบายเหตุผลประกอบ

```

1 class SuperClass{
2     private int num=1;
3     protected int getNumber(){
4         return num;
5     }
6 }
7 class Subclass extends SuperClass{
8     private int num=10;
9     //overriding method
10    public static void main(String[] args){
11        Subclass s= new Subclass();
12        System.out.println(s.getNumber());
13    }
14 }

```

โปรแกรม	สามารถนำมาเติมลงในบรรทัดที่ 9 ของโปรแกรมได้หรือไม่	อธิบายเหตุผลประกอบ
protected int getNumbers(){ return num+5; }		
protected long getNumber(){ return num+5; }		
protected int getNumber(){ return num+5; }		
public int getNumber(){ return num+5; }		
protected int getNumber)int num({ return num+5; }		
int getNumber(){ return num+5; }		
private int getNumber(){ return num+5; }		

5. [Application] จงเขียนโปรแกรมเพื่อแสดงรายละเอียดของคลาส Account ที่ประกอบด้วยสมาชิกต่อไปนี้

- ตัวแปร private ชนิดข้อมูล int ชื่อ id สำหรับเก็บหมายเลขบัญชี
- ตัวแปร private ชนิดข้อมูล double ชื่อ balance สำหรับเก็บยอดเงินคงเหลือ
- ตัวแปร private ชนิดข้อมูล double ชื่อ annualInterestRate สำหรับเก็บอัตราดอกเบี้ย
- ตัวแปร private ชนิดข้อมูล Date ชื่อ dateCreated สำหรับเก็บวันที่ที่บัญชีถูกสร้าง
- constructor ที่ไม่มี argument สำหรับการสร้างบัญชีแบบ default
- constructor ที่มี argument สำหรับการสร้างบัญชีแบบระบุเลขที่บัญชี และยอดเงินเริ่มต้น
- accessor method(get) และ mutator method(set) สำหรับตัวแปร id, balance, annualInterestRate,

dateCreated

- เมธอด getMonthlyInterestRate() ที่คืนอัตราดอกเบี้ยรายเดือน
- เมธอด getMonthlyInterest() ที่คืนดอกเบี้ยรายเดือน
 - เมธอด withdraw() ที่ถอนเงินตามจำนวนที่ระบุ
- เมธอด deposit() ที่ฝากเงินตามจำนวนที่ระบุ

วาดคลาสไดอะแกรมและเขียนส่วนของ Client สำหรับเรียกใช้คลาส Account โดยสร้างอ็อบเจกต์ของบัญชีเลขที่ (ID) 1122 ยอดเงินเปิดบัญชีคือ 20000 และอัตราดอกเบี้ยคือ 4.5 % หลังจากนั้นให้ใช้ withdraw method สำหรับถอนเงิน 2500 บาท และใช้ deposit method สำหรับฝากเงิน 3000 บาท และโปรแกรมสามารถแสดงยอดเงินคงเหลือและอัตราดอกเบี้ยรายเดือนได้

6. [Application] จากคลาส Account ให้เพิ่มสมาชิกของคลาส Account เพิ่มเติมดังต่อไปนี้

- ตัวแปร private ชนิดข้อมูล Date ชื่อ dateCreated สำหรับเก็บวันที่ที่บัญชีถูกสร้าง โดยชนิดข้อมูล Date ให้สร้างเองโดยประกอบด้วยสมาชิกภายในดังนี้

- Constructor ที่ไม่มี argument สำหรับการสร้างวันที่แบบ default
- Constructor ที่มี argument สำหรับการสร้างวันที่ที่ระบุวัน เดือน ปี
- ตัวแปร private ชนิดข้อมูล int ชื่อ day
- ตัวแปร private ชนิดข้อมูล String ชื่อ month
- ตัวแปร private ชนิดข้อมูล int ชื่อ year

- ตัวแปร private ชนิดข้อมูล Person ชื่อ objPerson สำหรับเก็บประวัติของลูกค้า โดยชนิดข้อมูล Person ให้สร้างเองโดยกำหนดให้ประกอบด้วยสมาชิกภายในดังนี้

- Constructor ที่ไม่มี argument สำหรับการสร้างลูกค้าแบบ default
- Constructor ที่มี argument สำหรับการสร้างลูกค้าที่ระบุชื่อ นามสกุล

- ตัวแปร private ชนิดข้อมูล String ชื่อ name
- ตัวแปร private ชนิดข้อมูล String ชื่อ surname
- ตัวแปร private ชนิดข้อมูล int ชื่อ age
- ตัวแปร private ชนิดข้อมูล Date ชื่อ bDate
- สร้าง accessor method(get) และ mutator method(set) สำหรับตัวแปรให้เหมาะสมในแต่ละคลาส
- เมธอด transferMoney(Account acc1, amount) ที่ทำหน้าที่ในการโอนเงินจากบัญชีปัจจุบันไปบัญชี acc1
- แก๊ซเมธอด getMonthlyInterest() ที่คืนดอกเบี้ยรายเดือนโดยคิดจากวันเปิดบัญชี
- เมธอด toString() สำหรับแสดงรายละเอียดภายในของคลาสแต่ละคลาส

จากนั้นให้สร้างคลาสใหม่เพิ่มอีก 2 คลาส ชื่อ SavingAccount และ FixAccount โดยสืบทอดมาจากคลาส Account โดยให้แก๊ซเมธอด transferMoney() ในคลาส SavingAccount โดยให้เพิ่มค่าธรรมเนียมที่จะต้องจ่ายให้กับทางธนาคาร 20 บาทต่อ 1 รายการ

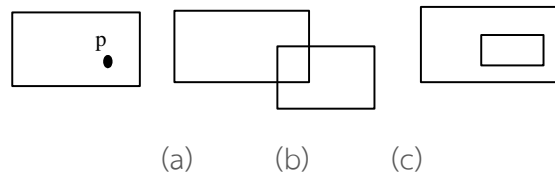
แก๊ซเมธอด transferMoney() ในคลาส FixAccount โดยบัญชีประเภทดังกล่าวจะไม่สามารถโอนเงินได้ โดยระบบต้องแจ้งต่อผู้ใช้

6.1 วาดคลาสไดอะแกรมและเขียนส่วนของ Client สำหรับเรียกใช้คลาส SavingAccount โดยสร้างออบเจ็คของบัญชีเลขที่ (ID) 1123 ยอดเงินเปิดบัญชีคือ 20000 และอัตราดอกเบี้ยต่อปีคือ 4.5 % หลังจากนั้นให้ป้อนข้อมูลชื่อ นามสกุล อายุ วันเดือนปีเกิด ของผู้เปิดบัญชี หลังจากนั้นให้ใช้ withdraw() สำหรับถอนเงิน 2500 บาท และใช้ deposit() สำหรับฝากเงิน 3000 บาท จากนั้นให้ทำการโอนเงินโดยใช้ () สำหรับการโอนเงินจากบัญชี ID 1123 ไปยังบัญชี ID 1100 และโปรแกรมสามารถแสดงยอดเงินคงเหลือและดอกเบี้ยรายเดือนได้

6.2 วาดคลาสไดอะแกรมและเขียนส่วนของ Client สำหรับเรียกใช้คลาส FixAccount โดยสร้างออบเจ็คของบัญชีเลขที่ (ID) 1124 ยอดเงินเปิดบัญชีคือ 20000 และอัตราดอกเบี้ยต่อปีคือ 7 % หลังจากนั้นให้ป้อนข้อมูลชื่อ นามสกุล อายุ วันเดือนปีเกิด ของผู้เปิดบัญชี หลังจากนั้นให้ใช้ withdraw() สำหรับถอนเงิน 2500 บาท และใช้ deposit() สำหรับฝากเงิน 3000 บาท จากนั้นให้ทำการโอนเงินโดยใช้ transferMoney() สำหรับการโอนเงินจากบัญชี ID 1124 ไปยังบัญชี ID 1100 และโปรแกรมสามารถแสดงยอดเงินคงเหลือและดอกเบี้ยรายเดือนได้ ในการถอนเงินโปรแกรมจะต้องเช็คค่าปีที่ถอนต้องมากกว่าปีที่ฝาก 1 ปีจึงจะทำการถอนได้ ส่วนเมื่อเรียกใช้ transferMoney() โปรแกรมจะต้องแจ้งผู้ใช่ว่าไม่สามารถโอนเงินได้

7. [Algorithms] จากคลาส Rectangle ในตัวอย่าง ให้สร้างคลาส MyRectangle2D ที่ประกอบไปด้วยรายละเอียดต่อไปนี้:

- ตัวแปรชนิด double ชื่อ x และ y ที่กำหนดจุดศูนย์กลางของสี่เหลี่ยม โดยมีเมธอด get และ set สำหรับกำหนดค่าและดึงค่า
- ตัวแปรชนิด double ชื่อ width and height โดยมีเมธอด get และ set สำหรับกำหนดค่าและดึงค่า
- no-arg constructor สำหรับการสร้างสี่เหลี่ยมแบบ default rectangle ที่มีการกำหนด (0, 0) สำหรับตัวแปร (x, y) และ 1 สำหรับ width and height.
- constructor สำหรับการสร้างสี่เหลี่ยมแบบกำหนดค่า $x, y, \text{width}, \text{and height}$.
- เมธอด getArea() ที่คืนพื้นที่ของสี่เหลี่ยม.
- เมธอด getPerimeter() ที่คืนเส้นรอบวงของสี่เหลี่ยม.
- เมธอด contains(double $x, \text{double } y$) ที่คืนค่า true ถ้ามีจุด point (x, y) อยู่ภายในสี่เหลี่ยมดังรูปที่ 1(a).
- เมธอด contains(MyRectangle2D r) ที่คืนค่า true ถ้ามีสี่เหลี่ยม rectangle อยู่ภายในสี่เหลี่ยม ดังรูปที่ 1(b).
- เมธอด overlaps(MyRectangle2D r) ที่คืนค่า true ถ้ามีสี่เหลี่ยม rectangle) ที่มีบางส่วนซ้อนทับกัน ดังรูปที่ 1(c).



รูปที่ 1

(a) จุดอยู่ในสี่เหลี่ยม. (b) สี่เหลี่ยมอยู่ภายในสี่เหลี่ยมตัวอื่น. (c) สี่เหลี่ยมที่มีบางส่วนซ้อนทับกัน

วาดคลาสไดอะแกรมของคลาส MyRectangle2D.

สร้างเมธอด getArea(), getPerimeter(), contains(double $x, \text{double } y$), contains(MyRectangle2D r), และ overlaps(MyRectangle2D r).

8. ให้นักศึกษา นิยาม Class ชื่อ Rectangle และ Line โดยแต่ละ class ประกอบไปด้วย

Rectangle	Line
Attributes width, height, และ คู่ลำดับ (x, y) แสดง มุมซ้ายมือด้านบนของ Rectangle	Attributes คู่ลำดับ (x1,y1) และ (x2,y2) แสดง จุดเริ่มต้นและจุดสุดท้ายของ Line
No-arg constructor	No-arg constructor
Constructor ที่กำหนดค่าให้แก่ width, height, (x, y)	Constructor ที่กำหนดค่าให้แก่ (x1,y1) และ (x2,y2)
Method getArea(Rectangle a) // return พื้นที่ของ Rectangle	Method getLong(Line a) // return ความยาวของ Line

class Rectangle { }	class Line { }
---	--

ให้นักศึกษาเขียนรายละเอียดของเมธอดต่อไปนี้

Method contains(Line a, Rectangle b) ที่ return 1 ถ้าค่า object a อยู่ภายใน object b และ return 0 ถ้าค่า object a ไม่อยู่ภายใน object b

Method cross(Line a, Line b) ที่ return 1 ถ้า object a overlap กับ object b และ return 0 ถ้า object a ไม่ overlap กับ object b

Method overlaps(Rectangle a, Rectangle b) ที่ return 1 ถ้า object a overlap กับ object b และ return 0 ถ้า object a ไม่ overlap กับ object b

บทที่ 8 การห่อหุ้มและการซ่อนข้อมูล

วัตถุประสงค์

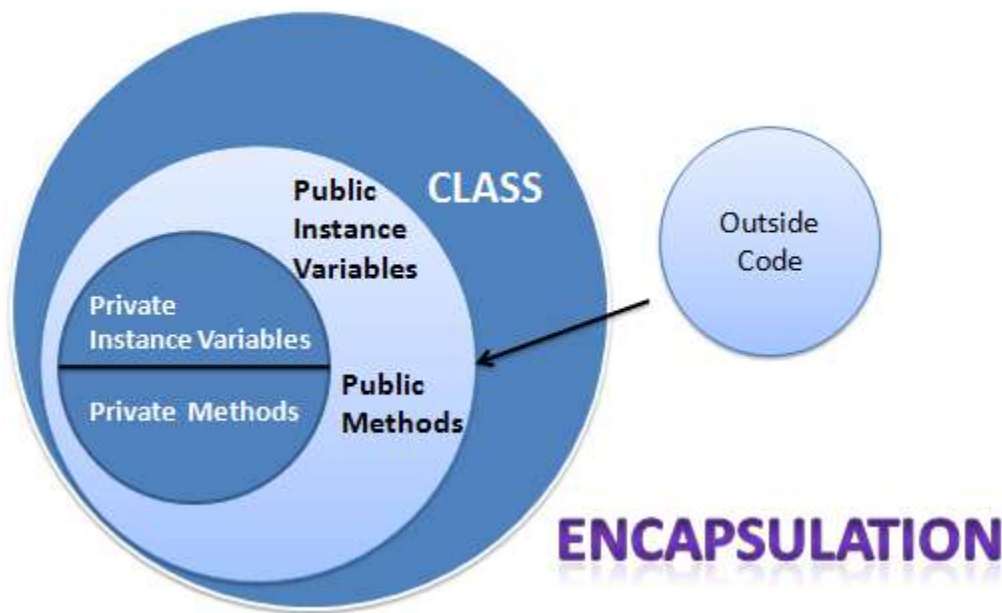
- 8.1 สามารถอธิบายความหมายของการห่อหุ้มและการซ่อนข้อมูล
- 8.2 สามารถเขียนโปรแกรมเพื่อประยุกต์การห่อหุ้มและการซ่อนข้อมูล

8.1 ความนำ

การเขียนโปรแกรมเชิงวัตถุในบางส่วนของโปรแกรมจะมีการป้องกันไม่ให้มีการเปลี่ยนแปลงโดยไม่ถูกต้อง หรือเกิดความไม่ปลอดภัยขึ้นและอาจไม่จำเป็นที่ต้องแสดงให้รู้และเห็นรายละเอียดของโปรแกรมห่อหุ้มไว้ได้ ภาษาจาวาได้สร้างกลไกในการป้องกันเรียกว่า การห่อหุ้ม เพื่อซ่อนข้อมูลบางอย่างที่ไม่ต้องการให้เปลี่ยนแปลงหรือเข้าถึงได้โดยไม่จำเป็น เรียกว่า การซ่อนข้อมูล (information Hiding) ทำให้อุปเจกต์หนึ่งสามารถติดต่อกับออปเจกต์ภายนอกผ่านเมธอดที่เป็นส่วนของส่วนของอินเตอร์เฟส (interface) เท่านั้น อีกอย่างหนึ่งคือการพัฒนาโปรแกรมเชิงวัตถุจะสามารถกำหนดให้อุปเจกต์แต่ละออปเจกต์มีความเป็นอิสระต่อกัน เรียกว่าความเป็นเป็นโมดูล (modularity)

8.2 นิยามของการห่อหุ้มและการซ่อนข้อมูล

การห่อหุ้มและการซ่อนข้อมูล หรือ encapsulation คือการปกปิดหรือควบคุมการเข้าถึงข้อมูลของวัตถุจากภายนอก โดยกลไกที่ใช้ในการควบคุมคือ การสร้างเมธอดและกำหนดสิทธิการทำงานของเมธอดเพื่อทำงานกับข้อมูลนั้น ในภาษาจาวาและภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุภาษาอื่นๆ สามารถใช้คุณสมบัตินี้ในการปกปิดส่วนประกอบภายในคลาสได้



รูปแสดงแนวคิดของ Encapsulation [<http://marcuscode.com/lang/java/encapsulation>]

คุณสมบัติการห่อหุ้ม (Encapsulation)

1. แนวคิดของ encapsulation ที่มีตัวแปรที่ถูกปกปิดไว้ภายใน

2. แนวคิดการใช้ encapsulation ได้ถูกใช้ในการเขียนโปรแกรมเชิงวัตถุแล้วแต่ตัวผู้เขียนโปรแกรมอาจจะไม่ทราบว่าได้นำคุณสมบัติดังกล่าวมาใช้ในตัวโปรแกรมที่เขียนแล้วยกตัวอย่างเช่น เมธอด `println()` ซึ่งมีเบื้องหลังการทำงานที่จะต้องทำการติดต่อกับจอภาพ เราใช้มันแค่เพียงคำสั่งเดียวและได้ผลลัพธ์ออกมา

3. encapsulation ในภาษา Java และแนวคิดการทำงานมีประโยชน์ในการใช้สำหรับพัฒนาโปรแกรมเป็นทีมหรือพัฒนาไลบรารีที่ให้อื่นใช้งาน ซึ่งสามารถใช้ได้โดยไม่ต้องกังวลว่ามันทำงานยังไง แต่ได้ผลลัพธ์ที่ต้องการนักพัฒนาโปรแกรมสามารถเรียกใช้งานเมธอดของคลาสเพื่อวัตถุประสงค์บางอย่าง เพียงแค่ทราบวิธีการเรียกใช้ และผลลัพธ์ที่ได้

8.3 การใช้ Modifiers และระดับการมองเห็น

Access Modifiers เป็นคำสั่งในการควบคุมระดับการเข้าถึงของตัวแปรหรือเมธอดที่อยู่ภายในคลาส Access Modifiers ยังเป็นคำสั่งที่ใช้ในการกำหนดการเข้าถึงของออบเจกต์ต่างๆ ใน Package เช่น คลาสและอินเทอร์เฟซ เป็นต้น

การใช้ Modifiers และระดับการมองเห็น

คีย์เวิร์ด `private` `protected` และ `public` เป็นระดับการมองเห็น (visibility) หรือ คีย์เวิร์ดบ่งบอกระดับการเข้าถึงของข้อมูลโดยเป็นคำที่บอกว่าคลาสและสมาชิกของคลาสตัวไหนบ้างที่สามารถเข้าถึงได้โดยมีระดับของการเข้าถึงข้อมูลดังนี้


ตัวอย่างของการใช้ Modifiers และระดับการมองเห็น เช่นคลาส `Account` ที่ประกอบด้วยสมาชิกต่าง ๆ ภายในคลาสเราสามารถกำหนดระดับของการมองเห็นได้ดังตัวอย่างต่อไปนี้

ชื่อ	ชนิดข้อมูล	ตัวแปร	Access Modifiers
id	int	หมายเลขบัญชี	private
balance	double	ยอดเงินคงเหลือ	Private
annualInterestRate	double	อัตราดอกเบี้ย	private
dateCreated	date	วันที่ที่บัญชีถูกสร้าง	Private

ในภาษาจาวามีคำสั่งในการควบคุมระดับการเข้าถึงอยู่ 4 ระดับด้วยกัน คือ `public` `protected` `private` และ `no modifier` (ไม่ต้องกำหนด) โดยแต่ละรูปแบบนั้นมีความหมายดังนี้

Access Modifiers	ระดับการเข้าถึง
public	คลาสหรือสมาชิกสามารถเข้าถึงได้ทุกส่วนของโปรแกรม
protected	คำนำหน้า <code>Protected</code> สามารถใช้กับ <code>method</code> ในคลาสได้ <code>attribute</code> หรือ <code>method</code> ที่นำหน้าด้วย <code>protected</code> สามารถเข้าถึงได้โดยคลาสต่าง ๆ ที่อยู่ package เดียวกันหรือในคลาสลูกที่อยู่ package เดียวกันหรือในคลาสลูก ที่อยู่ต่าง package ก็ได้
no modifier (ไม่ต้องกำหนด)	คลาสหรือสมาชิกสามารถเข้าถึงได้ภายใน package เดียวกัน และภายในคลาสเดียวกัน
private	คลาสหรือสมาชิกสามารถเข้าถึงได้ภายในคลาสเดียวกันเท่านั้น

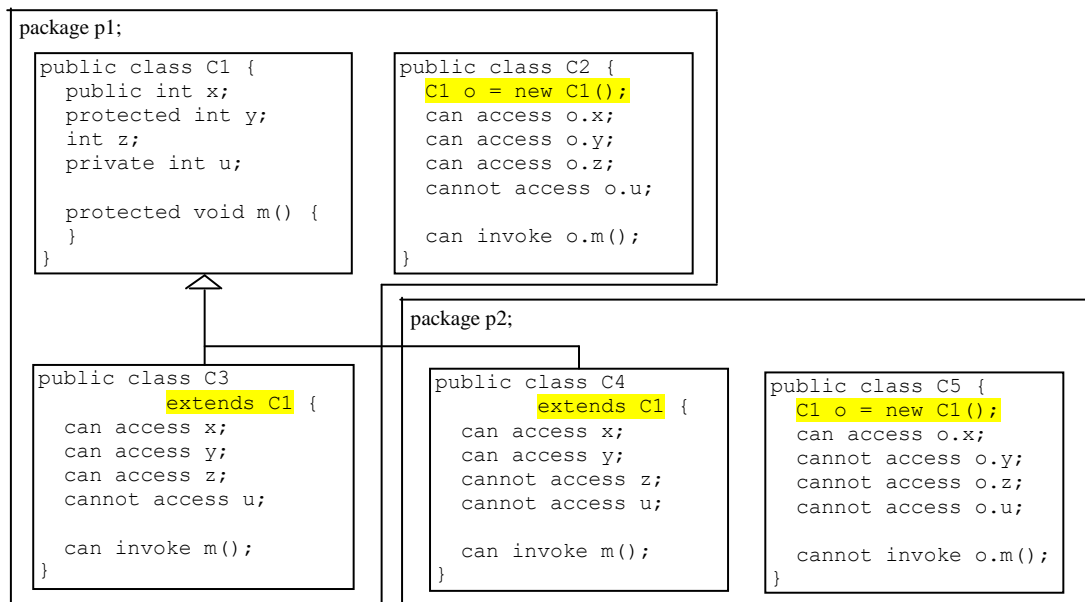
จากคำอธิบายข้างต้น สามารถนำมาเขียนเป็นตารางการเข้าถึงของคำสั่งทั้งหมดได้ดังนี้ โดยที่ public เป็นการเข้าถึงที่เป็นสาธารณะมากที่สุด และ private นั้นปิดกั้นที่สุดดังนี้

Visibility increases

 private, none (if no modifier is used), protected, public

ตารางแสดงระดับการเข้าถึงข้อมูลสามารถแสดงรายละเอียดได้ดังนี้

Access Modifiers	เข้าถึงได้ภายในคลาสเดียวกัน	เข้าถึงได้ภายใน package เดียวกัน	เข้าถึงได้จากคลาสลูก (subclass)	เข้าถึงได้จากต่าง package
public	✓	✓	✓	✓
protected	✓	✓	✓	
no modifier(ไม่ต้องกำหนด)	✓	✓		
private	✓			

รูปแบบการเข้าถึงข้อมูลเมื่อมีการใช้คีย์เวิร์ด public protected และ private ในคลาส C1 C2 C3 และ C4



8.4 คีย์เวิร์ด final

เมื่อมีการระบุด้วยคีย์เวิร์ด final หน้าตัวแปรจะเป็นการประกาศค่าคงที่ กรณีที่เราไม่ต้องการให้คลาสสามารถสืบทอดได้จะใช้คีย์เวิร์ด final ในการระบุ เราสามารถกำหนดให้เมธอดเป็น final ได้ในกรณีที่ไม่ต้องการให้คลาสลูกสามารถ overridden ได้เช่นกัน

8.5 เมธอดที่สำคัญในคลาส Object

8.5.1 เมธอด finalize ()

ถูกเรียกโดย gc บนวัตถุเมื่อ object กลายเป็น garbage และไม่สามารถเข้าถึงได้ต่อไป โดย default เมธอดนี้ไม่ได้ทำอะไรแต่ Subclass สามารถโอเวอร์ไรด์(override) finalize method เพื่อให้สามารถทำงานอย่างอื่นได้

8.5.2 เมธอด clone()

ในบางกรณีหากเราต้องการคัดลอกวัตถุ เราสามารถใช้เครื่องหมาย = ในรูปแบบต่อไปนี้ได้

```
newObject=someObject
```

ในกรณีข้างต้นไม่ได้สร้างวัตถุขึ้นอีกตัวแต่เป็นการเปลี่ยนการอ้างอิง โดยให้ newObject ชี้ไปที่เดียวกับ someObject ในการสร้างวัตถุใหม่ที่ตำแหน่งใหม่ในหน่วยความจำ ให้ใช้เมธอด clone() แทนโดยมีรูปแบบดังนี้
newObject = someObject.Clone();

วัตถุบางตัวอาจจะ clone ไม่ได้ สำหรับวัตถุที่ต้องการให้สามารถ clone ได้ ต้อง implements อินเตอร์เฟส java.lang.Cloneable ในทำนองเดียวกันเราสามารถ clone อาร์เรย์ก็ได้

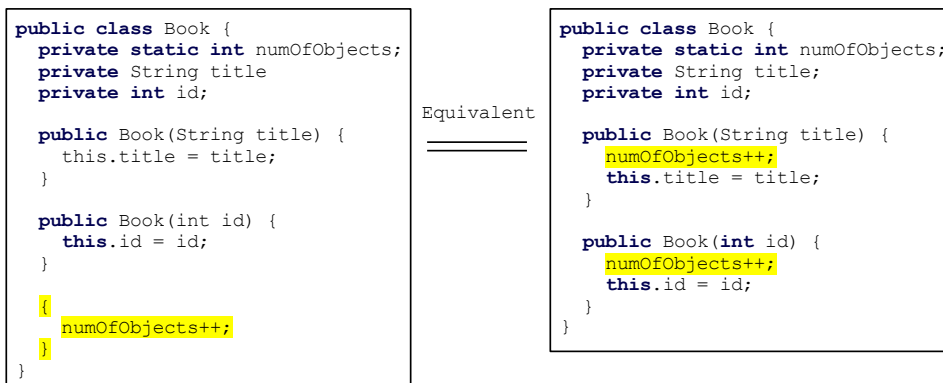
8.5.3 เมธอด getClass()

คลาสต้องถูกโหลดเพื่อมาใช้งานโดย JVM เมื่อ JVM โหลดคลาสจะสร้างวัตถุตัวหนึ่งที่มีข้อมูลของคลาส เช่น ชื่อคลาส constructor และเมธอด วัตถุดังกล่าวเป็นอินสแตนซ์ของ java.lang.Class มีลักษณะเป็น meta-object เพราะอธิบายรายละเอียดให้กับคลาสในช่วง runtime ทุกๆ วัตถุสามารถใช้เมธอด getClass เพื่อคืน meta object ตัวนี้ได้

8.6 Initialization block

สามารถใช้เพื่อกำหนดค่าเริ่มต้นของวัตถุโดยถูกใช้พร้อมกับ constructor ตัว initialization block จะปรากฏในส่วนของการประกาศคลาสแต่ไม่ได้อยู่ภายในเมธอดใด ๆ ทั้งสิ้น หรือไม่ได้อยู่ในส่วนของ constructor จะถูก execute ก่อน constructor ในคลาส

Initialization block ทำให้คลาสเขียนได้ง่ายขึ้นเนื่องจากมีการ share code บางส่วนร่วมกันแต่มีข้อจำกัดคือต้องไม่มี constructor ที่เรียก constructor ตัวอื่น เราสามารถวาง sourcecode ที่จะถูกใช้งานร่วมกันโดย สามารถวางไว้ที่ initialization block ดังตัวอย่างดังรูป



Initialization block ดังรูปข้างต้นจะเป็น instance initialization block เนื่องจาก block ดังกล่าวจะถูกเรียกเมื่อ instance ของคลาสถูกสร้างขึ้น

Static initialization block จะเหมือนกับ instance initialization block ยกเว้นเมื่อประกาศเป็น static จะสามารถอ้างถึง static member ในคลาสได้อย่างเดียวเท่านั้น และถูกเรียกเมื่อคลาสถูกโหลด JVM จะโหลดคลาสโดยอัตโนมัติเมื่อจำเป็น superclass จะถูกโหลดก่อน subclass ลำดับการทำงานสามารถสรุปได้ดังนี้

1. เมื่อคลาสถูกเรียกใช้ครั้งแรกจะโหลดคลาส กำหนดค่าของ static data fields และ execute static initialization block ของคลาส
2. เมื่อวัตถุของคลาสถูกสร้างขึ้น constructor ของคลาสจะถูกเรียกโดยมี ขั้นตอนคือ 3
 - 2.1 เรียก constructor ของ superclass
 - 2.2 กำหนดค่าเริ่มต้นให้ instance datafields และ instance initialization block
 - 2.3 Execute ภายใน constructor

ตัวอย่างที่ 81. การใช้ Initialization block และ Static initialization block

```
public class InitializationDemo {
    public static void main(String[] args) {
        new InitializationDemo();
    }
    public InitializationDemo() {
        new M();
    }
    {
        System.out.println("(2) InitializationDemo's instance block");
    }
    static {
        System.out.println("(1) InitializationDemo's static block");
    }
}
class M extends N {
    M() {
        System.out.println("(8) M's constructor body");
    }
    {
        System.out.println("(7) M's instance initialization block");
    }
    static {
        System.out.println("(4) M's static initialization block");
    }
}
class N {
    N() {
        System.out.println("(6) N's constructor body");
    }
    {
        System.out.println("(5) N's instance initialization block");
    }
    static {
        System.out.println("(3) N's static initialization block");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

- (1) InitializationDemo's static block
- (2) InitializationDemo's instance block
- (3) N's static initialization block
- (4) M's static initialization block
- (5) N's instance initialization block

- (6) N's constructor body
- (7) M's instance initialization block
- (8) M's constructor body

ตัวอย่างที่ 8.2 การใช้ Initialization block และ Static initialization block

```
public class Test {
    public static void main(String[] args) {
        new Test();
    }
    public Test() {
        new Parrot();
    }
    {
        System.out.println("(2) Test's instance block "+ "is invoked");
    }
    static {
        System.out.println("(1) Test's static block "+ "is invoked");
    }
}

class Parrot extends Birds {
    Parrot() {
        System.out.println("(8) Parrot's constructor body "+ "is invoked");
    }
    {
        System.out.println("(7) Parrot's instance initialization block "+ "is invoked");
    }
    static {
        System.out.println("(4) Parrot's static initialization block "+ "is invoked");
    }
}

class Birds {
    Birds() {
        System.out.println("(6) Birds's constructor body "+ "is invoked");
    }
    {
        System.out.println("(5) Birds's instance initialization block "+ "is invoked");
    }
    static {
        System.out.println("(3) Birds's static initialization block "+ "is invoked");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

- (1) Test's static block is invoked
- (2) Test's instance block is invoked
- (3) Birds's static initialization block is invoked
- (4) Parrot's static initialization block is invoked
- (5) Birds's instance initialization block is invoked
- (6) Birds's constructor body is invoked
- (7) Parrot's instance initialization block is invoked
- (8) Parrot's constructor body is invoked

โดยลำดับการทำงานมีดังนี้

1. คลาส Test จะถูกเรียกเป็นครั้งแรก ดังนั้น static initialization block จะถูกเรียก
2. Constructor ของคลาส Test จะถูกเรียก ดังนั้น initialization block ของ instance ของคลาส test จะถูกเรียก
3. เมื่อประมวลผลคำสั่ง new Parrot() คลาส Parrot จำเป็นต้องถูกโหลดทำให้คลาสแม่ของคลาส Parrot ถูกเรียกเป็นลำดับแรก ดังนั้น static initialization block ของคลาส Birds จะถูกเรียก
4. คลาส Parrot จะถูกโหลดดังนั้น static initialization block ของคลาส Parrot จะถูกเรียก
5. เมื่อเรียก Constructor ของคลาส Parrot จะทำให้ constructor ที่ไม่มีพารามิเตอร์ของ Superclass ของคลาส parrot คือคลาส Birds จะถูกเรียกลำดับแรก ดังนั้น instance initialization block ของคลาส Birds จะถูกเรียก
6. Code ของ Constructor ที่ไม่มีพารามิเตอร์ของคลาส Birds จะถูกเรียก หลัง instance initialization block ของคลาส Birds ถูกเรียก
7. หลังจาก Constructor ที่ไม่มีพารามิเตอร์ของคลาส Birds ถูกเรียก constructor ที่ไม่มีพารามิเตอร์ของคลาส Parrot จะถูกเรียกซึ่งทำให้ instance initialization block ของคลาส Parrot จะถูกเรียกเป็นลำดับแรก
8. Code ของ Constructor ที่ไม่มีพารามิเตอร์ของคลาส Parrot จะถูกเรียกหลัง instance initialization block ของคลาส Parrot ซึ่งถูกเรียกไปแล้ว

8.6 บทสรุป

การห่อหุ้มและการซ่อนข้อมูล หรือ encapsulation คือการปกปิดหรือควบคุมการเข้าถึงข้อมูลของวัตถุจากภายนอก โดยกลไกที่ใช้ในการควบคุมคือ การสร้างเมธอดและกำหนดสิทธิการทำงานของเมธอดเพื่อทำงานกับข้อมูลนั้น ในภาษาจาวาและภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุภาษาอื่นๆ สามารถใช้คุณสมบัตินี้ในการปกปิดส่วนประกอบภายในคลาสได้

Access Modifiers เป็นคำสั่งในการควบคุมระดับการเข้าถึงของตัวแปรหรือเมธอดที่อยู่ภายในคลาส Access Modifiers ยังเป็นคำสั่งที่ใช้ในการกำหนดการเข้าถึงของออบเจ็กต์ต่างๆ ใน Package เช่น คลาสและอินเทอร์เฟส เป็นต้น คีย์เวิร์ด private protected และ public เป็นระดับการมองเห็น (visibility) หรือ คีย์เวิร์ดบ่งบอกระดับการเข้าถึงของข้อมูลโดยเป็นคำที่บอกว่าคลาสและสมาชิกของคลาสตัวไหนบ้างที่สามารถเข้าถึงได้โดยมีระดับของการเข้าถึงข้อมูล

8.7 แบบฝึกหัดปฏิบัติการ

1. จงเขียนโปรแกรมเพื่อสร้างคลาส `SetOfInteger` ที่ใช้แทนเซตของตัวเลขจำนวนเต็ม โดยจะเก็บไว้ในอาร์เรย์ที่เรียงจากน้อยไปมากเสมอ และตัวเลขในอาร์เรย์ต้องไม่ซ้ำกัน โดยสร้างเมธอด แอทธิบิตต์ตามรายละเอียดต่อไปนี้

- ประกาศแอทธิบิตต์แบบ `public` ที่เป็นอาร์เรย์ประเภทจำนวนเต็มชื่อ `iSet`
- กำหนดตัวสร้างแบบที่รับพารามิเตอร์ 1 ตัวที่เป็นอาร์เรย์ชนิดจำนวนเต็มเพื่อนำมาหาค่าเริ่มต้นให้กับแอทธิบิตต์ `iSet` โดยจะต้องมีการเรียงจากน้อยไปมากและจำกัดสมาชิกที่ซ้ำกันเสมอ ดังตัวอย่าง {7, 5,3,5,3} จะได้เป็น 3 5 7
- กำหนดตัวสร้างแบบที่รับพารามิเตอร์ 1 ตัวที่เป็นชนิด `SetOfInteger` เพื่อนำมาหาค่าเริ่มต้นให้กับแอทธิบิตต์ `iSet`
- เมธอด `sort` ใช้สำหรับเรียงข้อมูลจากน้อยไปมาก
- เมธอด `removeDuplicatedMember` สำหรับใช้จำกัดสมาชิกตัวที่ซ้ำกัน
- เมธอด `union` ใช้สำหรับคำนวณหาเซตที่เกิดจากการนำเอา `iSet` ในคลาสนี้กับ `iSet` ที่รับเข้ามาทางพารามิเตอร์มายูเนียนกันแล้วคืนคำตอบชนิด `SetOfInteger`

2. จงออกแบบและเขียน class ต่อไปนี้

กระแสการเล่นเกมส์ออนไลน์กำลังเป็นที่นิยมเป็นอย่างมากในปัจจุบัน นักเล่นเกมสบางส่วนโหยหาถึงเกมส์ในอดีตอย่าง Ragnarok ซึ่งเป็นเกมส์ออนไลน์โดยในการพัฒนาเกมส์เริ่มต้นมีแค่ตัวละครเดียวคือ เด็กฝึกหัด (Novice) ซึ่งตัวละครดังกล่าวจะมี พลังชีวิต พลังโจมตี ประสบการณ์ที่จะใช้ในการอัปเลเวล และสามารถ เดิน, นั่ง, โจมตี และตายได้

เมื่อพัฒนาไปได้สักพักเกมส์ที่เขียนเป็นที่นิยมเป็นอย่างมาก เลยทำให้เราต้องเพิ่มตัวละครแบบอื่นๆเข้าไปบ้าง นั่นคือ นักดาบ (Swordman) และ พระ (Acolyte) เพื่อให้เกมส์มีความหลากหลายขึ้น และมีเงื่อนไขเพิ่มมานั่นคือ

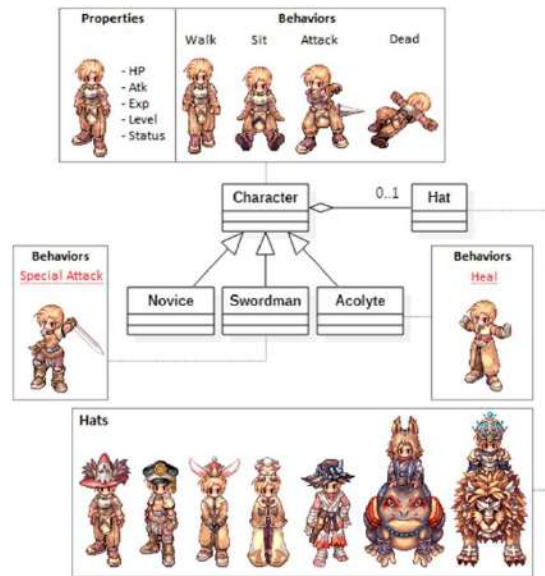
นักดาบจะมีพลังโจมตีเริ่มต้นที่ 10 หน่วย และสามารถใช้ ท่าโจมตีพิเศษ ได้อีกด้วย

พระมีพลังโจมตีเริ่มต้นที่ 5 หน่วย และสามารถใช้ การรักษาให้กับตัวเองได้ด้วย

โดยเมื่อมีตัวละครใหม่ 2 ตัวเพิ่มขึ้นมา ในขณะที่ตัว เด็กฝึกหัด (Novice) ก็ยังต้องใช้งานได้เหมือนเดิม นอกจากนี้

พระสามารถรักษาให้กับตัวเองและผู้เล่นคนอื่นได้ด้วย

เมื่อเกมส์เล่นไปในระยะหนึ่งผู้เล่นอยากแต่งตัวละครสวยๆ ได้ ทางบริษัทเลยคิดว่า ทุกตัวละครสามารถใส่หมวกบนหัวได้ และ หมวกแต่ละแบบก็จะเพิ่มความสามารถต่างๆให้กับผู้เล่นได้ด้วย



โดยรายละเอียดและความสามารถต่างๆ สามารถสรุปเป็นแผนภาพได้ตามรูปภาพด้านบน

จากรายละเอียดของเกมดังกล่าวให้นักศึกษาออกแบบคลาส เมธอด แอททริบิวต์ คอนสตรักเตอร์ และรายละเอียดภายในเมธอดที่จำเป็นสำหรับการเขียนเกมส์ตามเงื่อนไขดังกล่าว

บทที่ 9 การพ้องรูป

วัตถุประสงค์

- 9.1 สามารถอธิบายความหมายของการพ้องรูป
- 9.2 สามารถอธิบายการแปลงวัตถุ
- 9.3 สามารถอธิบายการทำงานของ *Dynamic binding* และ *Static binding*
- 9.4 สามารถเขียนโปรแกรมเพื่อประยุกต์ใช้คุณสมบัติการพ้องรูป

9.1 ความนำ

โพลีมอร์ฟิซึม (Polymorphism) หรือ การพ้องรูป คือการที่วัตถุหรือออบเจกต์สามารถมีได้หลายรูปแบบ ซึ่งเกิดจากการสืบทอดจาก super class โดยที่ออบเจกต์ที่สืบทอดจาก super class ดังกล่าวรักษาคูณสมบัติของ super class ไว้ได้ โพลีมอร์ฟิซึมเป็นการสร้างออบเจกต์โดยออบเจกต์ดังกล่าวถูกสร้างจากคลาสที่มี super class เดียวกัน โดยใน subclass นั้นได้มีการกำหนดการทำงานใหม่ให้กับเมธอดให้ตรงกับวัตถุประสงค์ของคลาสนั้น หรือการโอเวอร์ไรด์เมธอด (Override method) หลังจากนั้นเราสามารถให้ super class สำหรับการประกาศตัวแปรของออบเจกต์ (class instance) ที่สร้างออบเจกต์จาก subclass ได้

9.2 นิยามของการพ้องรูป (Polymorphism)

ความสัมพันธ์แบบการสืบทอดคุณสมบัติ อนุญาตให้คลาสสามารถสืบทอดคุณสมบัติจากคลาสแม่ (superclass) ได้ คลาสลูก (subclass) สามารถเพิ่มคุณสมบัติใหม่ ๆ ได้ โดย subclass จะเป็นคลาสที่เจาะจงลงไปจากคลาสแม่ ทุก ๆ วัตถุที่สร้างจาก subclass จะถือว่าเป็น instance ของ superclass ด้วย เช่น ทุก ๆ วัตถุที่สร้างจากคลาส circle ถือเป็นวัตถุของคลาส Object แต่ทุก ๆ วัตถุของคลาส Object ไม่ได้เป็นวัตถุชนิด circle ดังนั้นกรณีที่ในเมธอดมีการประกาศรับพารามิเตอร์เป็นแบบ instance ของ superclass เราสามารถส่ง instance ของ subclass ไปเป็นพารามิเตอร์ของ superclass ได้

9.3 Dynamic binding

การประกาศรับพารามิเตอร์เป็นแบบ instance ของ super class เราสามารถส่ง instance ของ subclass ไปเป็นพารามิเตอร์ของ super class ได้ โดยสามารถแสดงตัวอย่างการส่งพารามิเตอร์ในลักษณะดังกล่าว ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 9.1 ตัวอย่างการทำงานแบบ Dynamic binding

```
public class PolymorphismDemo {
    public static void main(String[] args) {
        m(new GraduateStudent());
        m(new Student());
        m(new Person());
        m(new Object());
    }
    public static void m(Object x) {
        System.out.println(x.toString());
    }
}
class GraduateStudent extends Student {
}
class Student extends Person {
```



```

public String toString() {
    return "Student";
}
}
class Person extends Object {
    public String toString() {
        return "Person";
    }
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

Student
Student
Person
java.lang.Object@15db9742

```

จากชุดคำสั่งข้างต้น เมธอด `m(Object x)` รับพารามิเตอร์ชนิดข้อมูล `Object` เราสามารถเรียกเมธอด `m(Object x)` โดยส่งวัตถุต่าง ๆ เช่น `new GraduateStudent()` หรือ `new Student()` เข้าไปยังเมธอดดังกล่าวได้ วัตถุของคลาสลูกสามารถใช้ได้ในทุก ๆ ส่วนของ code ที่ออกแบบให้ใช้ได้กับวัตถุของ superclass โดยเรียกคุณสมบัตินี้ว่า **polymorphism** ซึ่งหมายถึงการมีได้หลายรูป

เมื่อเมธอด `m(Object x)` ถูกประมวลผล เมธอด `toString()` ของตัวแปรอ้างอิงวัตถุ `x` จะถูกเรียก โดยขึ้นอยู่กับว่าตัวแปรอ้างอิงวัตถุ `x` เป็นตัวแปรที่อ้างอิงไปถึงวัตถุของคลาสใด เช่น `x` อาจจะเป็นอินสแตนซ์ของ `GraduateStudent` หรือ `Student` โดยที่ภายในคลาสดังกล่าวจะมีการเขียนรายละเอียดของเมธอด `toString()` ที่แตกต่างกัน ดังนั้นเมธอด `toString()` ตัวไหนในคลาสอะไรที่จะถูกเรียกใช้จะถูกตัดสินใจแบบไดนามิกโดย JVM ในช่วงของการรัน โดยเราเรียกคุณสมบัติแบบนี้ว่า **dynamic binding**



Dynamic binding มีการทำงานดังนี้

กำหนดให้วัตถุ `o` เป็นอินสแตนซ์ของคลาส `C1 C2... Cn-1 Cn` เมื่อ `C1` เป็นคลาสลูกของ `C2` และ `C2` เป็นคลาสลูกของ `C3` และ ...`Cn-1` เป็นคลาสลูกของ `Cn` ดังปรากฏในรูปกล่าวคือ `Cn` จะเป็นคลาสที่มีความเป็นทั่วไปมากที่สุด)most general class และ (`C1` เป็นคลาสที่มีความเจาะจงมากที่สุด)most specific class โดยคลาส (`Cn` ในภาษาจาวาคือคลาส `Object`

ถ้า `o` เป็นอินสแตนซ์ของคลาส `C1` โดยที่ `o` เรียกเมธอด `p` แล้ว JVM จะทำการค้นหาเมธอดที่ตรงกับที่ต้องการมากที่สุดโดยจะค้นหาเมธอด `p` ในคลาส `C1 C2... Cn-1 Cn` ตามลำดับจนกว่าจะพบ

หลักการของ Polymorphism จะอนุญาตให้เราสามารถประกาศเมธอดแบบ general ได้โดยอาร์กิวเมนต์ของเมธอดสามารถประกาศให้รับชนิดข้อมูลใด ๆ ก็ได้ที่อยู่โครงสร้างเดียวกัน เรียกรูปแบบนี้ว่า Generic programming ถ้าประกาศให้พารามิเตอร์ของเมธอดเป็น superclass ของคลาสนั้น เช่น ในคลาส `A` ซึ่งเป็นคลาสลูกของคลาส `Object` เราสามารถประกาศเมธอดที่รับพารามิเตอร์ที่มีชนิดเป็น `Object` หลังจากนั้นเมื่อสร้างอินสแตนซ์ของ subclass เราสามารถส่งวัตถุที่สร้างจาก subclass ส่งวัตถุดังกล่าวไปเป็นพารามิเตอร์ของเมธอดที่ประกาศพารามิเตอร์ก่อนหน้านี้ได้ เมื่อวัตถุถูกใช้ในเมธอดการทำงานของเมธอดภายในวัตถุนั้นจะถูกเรียกได้โดยอัตโนมัติ

9.4 Static binding

เราสามารถโอเวอร์ไรด์ instance method แต่ไม่สามารถ override ข้อมูลทั้งของ instance และ static หรือ static method ถ้ามีการประกาศ static method หรือ datafield ใน subclass โดยตัวแปรดังกล่าวหรือเมธอดดังกล่าวมีชื่อเดียวกันกับ superclass ตัวแปรดังกล่าวใน superClass จะถูกซ่อนแต่จะยังมีอยู่ โดยเราสามารถเรียกตัวแปรหรือ static method เหล่านั้นโดยการใช้คีย์เวิร์ด super ใน subclass ตัวแปรที่ถูกซ่อนหรือเมธอดที่ถูกซ่อนสามารถเข้าถึงได้ด้วยตัวแปรอ้างอิงวัตถุของ superclass (Object ของ superclass)

เมื่อเรียก Instance method จากตัวแปรอ้างอิงวัตถุ (reference variable) คลาสที่แท้จริงของตัวแปรอ้างอิงวัตถุจะถูกตัดสินใจเมื่อ runtime เมื่อเข้าถึง datafield หรือ static method ชนิดข้อมูลของตัวแปรอ้างอิงวัตถุที่กำหนดไว้จะถูกใช้ในช่วงของ compile time พิจารณาตัวอย่างต่อไปนี้

ตัวอย่างที่ 9.2 ตัวอย่างการทำงานแบบ Dynamic binding และ Static binding

```
public class Ex92{
    public static void main(String[] args){
        Animal x= new Tiger();
        System.out.println("1. x.news is "+ x.news);
        System.out.println("2. ((Tiger)x).news is "+ ((Tiger)x).news);
        System.out.println("3. x.smile() is "+ x.smile());
        System.out.println("4. ((Tiger)x).smile() is "+ ((Tiger)x).smile());
        System.out.println("5. x.getNews() is "+ x.getNews());
        System.out.println("6. x.getMessage() is "+ x.getMessage() );
    }
}
class Animal{
    public String news= "Animal's news";
    public String message="Animal's message";
    public static String smile(){
        return "smile from Animal";
    }
    public String getNews(){
        return news;
    }
    public String getMessage(){
        return message;
    }
}
class Tiger extends Animal{
    public String news= "Tiger's news";
    public String message="Tiger's message";
    public static String smile(){
        return "smile from Tiger";
    }
    //@override
    public String getNews(){
        return news;
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

1. x.news is Animal's news
2. ((Tiger)x).news is Tiger's news
3. x.smile() is smile from Animal
4. ((Tiger)x).smile() is smile from Tiger
5. x.getNews() is Tiger's news
6. x.getMessage() is Animal's message

จากตัวอย่างที่ 9.2 เป็นตัวอย่างของการใช้หลักการของการพ้องรูป (Polymorphism) ในการเขียนโปรแกรม การประกาศ `Animal x = new Tiger();` ใน `public static void main(String[] args)` เป็นการประกาศตัวแปรอ้างอิงวัตถุที่ชื่อ `x` โดยมีชนิดข้อมูลเป็น `Animal` โดยเก็บ Reference ของวัตถุที่สร้างจากคลาส `Tiger` เมื่อเรียก `x.news` จะทำการเรียกตัวแปร `news` ที่มาจากคลาส `Animal` เนื่องจากตัวแปรอ้างอิงวัตถุ `x` มีชนิดข้อมูลเป็น `Animal` โดยเป็นการผูกชนิดข้อมูลในช่วงของการคอมไพล์ ซึ่งเป็นการทำงานแบบ `Static binding` จะได้ผลลัพธ์เป็น `x.news is Animal's news`

เมื่อเรียก `((Tiger)x).news` จะทำการเรียกตัวแปร `news` ที่มาจากคลาส `Tiger` เนื่องจากตัวแปรอ้างอิงวัตถุ `x` มีการ Casting ข้อมูลเป็น `Tiger` โดยเป็นการผูกชนิดข้อมูลในช่วงของการคอมไพล์ ซึ่งเป็นการทำงานแบบ `Static binding` จะได้ผลลัพธ์เป็น `((Tiger)x).news is Tiger's news`

เมื่อเรียก `x.smile()` จะทำการเรียกเมธอด `smile()` ตามชนิดข้อมูลที่ตัวแปรอ้างอิงวัตถุ `x` ได้ประกาศไว้ก่อนโดยมีชนิดข้อมูลเป็น `Animal` ดังนั้นตัวคอมไพเลอร์จะทำการเช็คที่คลาส `Animal` มีเมธอด `smile()` หรือไม่ซึ่งพบว่ามี และการประกาศเป็นแบบ `static` ดังนั้นคอมไพเลอร์ จะเรียกเมธอด `smile()` จากคลาส `Animal` โดยเป็นการผูกเมธอดในช่วงของการคอมไพล์ ซึ่งเป็นการทำงานแบบ `Static binding` จะได้ผลลัพธ์เป็น `x.smile() is smile from Animal`

เมื่อเรียก `((Tiger)x).smile()` จะทำการเรียกเมธอด `smile()` ตามชนิดข้อมูลที่ตัวแปรอ้างอิงวัตถุ `x` ได้ประกาศไว้ก่อนโดยมีชนิดข้อมูลเป็น `Tiger` ดังนั้นตัวคอมไพเลอร์จะทำการเช็คที่คลาส `Tiger` มีเมธอด `smile()` หรือไม่ซึ่งพบว่ามี ดังนั้นคอมไพเลอร์ จะเรียกเมธอด `smile()` จากคลาส `Tiger` โดยเป็นการผูกเมธอดในช่วงของการรัน ซึ่งเป็นการทำงานแบบ `Dynamic binding` จะได้ผลลัพธ์เป็น `((Tiger)x).smile() is smile from Tiger`

เมื่อเรียก `x.getNews()` จะทำการเรียกเมธอด `getNews()` ตามชนิดข้อมูลที่ตัวแปรอ้างอิงวัตถุ `x` ได้ประกาศไว้ก่อนโดยมีชนิดข้อมูลเป็น `Tiger` ดังนั้นตัวคอมไพเลอร์จะทำการเช็คที่คลาส `Tiger` มีเมธอด `getNews()` หรือไม่ซึ่งพบว่ามี ดังนั้นคอมไพเลอร์ จะเรียกเมธอด `getNews()` จากคลาส `Tiger` โดยเป็นการผูกเมธอดในช่วงของการรัน ซึ่งเป็นการทำงานแบบ `Dynamic binding` จะได้ผลลัพธ์เป็น `x.getNews() is Tiger's news`

เมื่อเรียก `x.getMessage()` จะทำการเรียกเมธอด `getMessage()` ตามชนิดข้อมูลที่ตัวแปรอ้างอิงวัตถุ `x` ได้ประกาศไว้ก่อนโดยมีชนิดข้อมูลเป็น `Tiger` ดังนั้นตัวคอมไพเลอร์จะทำการเช็คที่คลาส `Tiger` มีเมธอด `getMessage()` หรือไม่ซึ่งพบว่าไม่มี ดังนั้นคอมไพเลอร์ จะเรียกเมธอด `getMessage()` จากคลาส `Animal` โดยเป็นการผูกเมธอดในช่วงของการรัน ซึ่งเป็นการทำงานแบบ `Dynamic binding` จะได้ผลลัพธ์เป็น `x.getMessage() is Animal's message`

9.5 การแปลงวัตถุด้วย Upcasting และ Downcasting

9.5.1 การแปลงวัตถุและโอเปอเรเตอร์ Instanceof

การแปลงหรือ Casting สามารถใช้แปลงวัตถุของคลาสประเภทหนึ่งไปเป็นอีกประเภทหนึ่งภายในโครงสร้างการสืบทอดเดียวกัน เช่น `m(new Student());` เป็นการกำหนดให้สร้างวัตถุของคลาส `Student` เป็นพารามิเตอร์ส่งไปยังเมธอด `m(Object x)` ที่มีอาร์กิวเมนต์ประเภทคลาส `Object` ในเมธอด โดยการทำงานจะคล้ายกับการการเขียนชุดคำสั่งต่อไปนี้

```
Object o=new Student();
m(o);
```

ซึ่งจากข้างต้นเป็นการแปลงแบบ Implicit หรือการแปลงโดยปริยาย

พิจารณาคำสั่ง `Student b=o` และ `Object o=new Student();`

จากตัวอย่างจะพบว่า `Student b=o` เมื่อ `o` มีชนิดข้อมูลเป็น `Object` จะเกิด Error แต่ `Object o=new Student()` ทำงานได้โดยไม่เกิด Error เนื่องจากอินสแตนซ์ที่สร้างจากคลาส `Student` จะถือว่าเป็นอินสแตนซ์ของคลาส `Object` เสมอ แต่อินสแตนซ์ของคลาส `Object` ไม่จำเป็นต้องเป็นอินสแตนซ์ของคลาส `Student` แม้ว่าเราจะรู้ว่าจริง ๆ แล้ว `o` เป็นอินสแตนซ์ของคลาส `Object` แต่คอมพิวเตอร์ไม่ฉลาดพอที่จะรู้ การจะบอกว่า `o` เป็นอินสแตนซ์ของคลาส `Student` จะต้องมีการบอกอย่างแน่ชัดหรือ explicit casting โดยวิธีการแปลงคือใช้คลาสที่ต้องแปลงอยู่ภายใต้วงเล็บหน้าวัตถุที่ต้องการแปลง เช่น

```
Student b= (Student) o;
```

9.5.2 Upcasting และ Downcasting

มีความเป็นไปได้ที่จะแปลงอินสแตนซ์ของคลาสลูกไปเป็นตัวแปรของคลาสแม่เรียกว่า Upcasting เนื่องจากอินสแตนซ์ของคลาสลูกมักจะเป็นอินสแตนซ์ของคลาสแม่ด้วย

เมื่อแปลงอินสแตนซ์ของคลาสแม่เป็นตัวแปรของคลาสลูกจะเรียกว่า Downcasting การแปลงในรูปแบบนี้จะเป็นการแปลงแบบ explicit โดยใช้รูปแบบ

```
(subclassName) variable
```

การแปลงให้สำเร็จได้จะต้องแน่ใจว่าวัตถุที่ต้องการแปลงเป็นอินสแตนซ์ของ Subclass จริง ๆ จากตัวอย่างข้างต้น ถ้าวัตถุของ Superclass ไม่ได้เป็น instance ของคลาส `Student` จะไม่สามารถแปลง `o` ไปเป็นตัวแปรของคลาส `Student` ได้ ดังนั้นควรเช็คก่อนว่าวัตถุนั้น ๆ เป็นอินสแตนซ์วัตถุที่ต้องการแปลงจริง ๆ เราสามารถใช้เมธอด `instanceof()` สำหรับตรวจสอบว่าวัตถุเป็น instance ที่เกิดจากคลาสใดกันแน่ ดังตัวอย่างด้านล่าง

ตัวอย่างที่ 9.3 Upcasting และ Downcasting

```
Object myObject = new Circle();
... // Some lines of code
/** Perform casting if myObject is an instance of Circle */
if (myObject instanceof Circle) {
    System.out.println("The circle diameter is " +
        ((Circle)myObject).getDiameter());
    ...
}
```

การแปลงหรือ Casting มีความสำคัญมาก ชนิดข้อมูลของตัวแปรจะถูกพิจารณาในช่วงของการคอมไพล์ จากตัวอย่างข้างต้น ถ้าตัวแปร myObject ถูกประกาศให้มีชนิดเป็น Object การใช้ myObject.findVolume () จะทำให้เกิด error เพราะคลาส Object ไม่มีเมธอด findVolume () ดังนั้นจำเป็นต้องแปลง myObject ไปเป็นประเภท Cylinder ก่อน จึงจะสามารถเรียกใช้เมธอด findVolume () ได้

คำถามเกิดขึ้นคือ แล้วทำไมไม่ประกาศ myObject เป็น Cylinder เลยตั้งแต่ตอนแรก คำตอบคือ เพื่อให้หลักการของ Generic programming ทำงานได้ จึงจำเป็นต้องประกาศตัวแปรเป็น superclass ไว้ก่อนซึ่งส่งผลให้สามารถรับตัวแปรจาก subclass ทุก ๆ ตัวได้

ตัวอย่างที่ 9.4 Upcasting และ Downcasting

```
public class TestPolymorphismCasting {
    public static void main(String[] args) {
        Object object1 = new Circle(1);
        Object object2 = new Rectangle(1, 1);
        displayObject(object1);
        displayObject(object2);
    }
    public static void displayObject(Object object) {
        if (object instanceof Circle) {
            System.out.println("The circle area is " +
                ((Circle)object).getArea());
            System.out.println("The circle diameter is " +
                ((Circle)object).getDiameter());
        }
        else if (object instanceof Rectangle) {
            System.out.println("The rectangle area is " +
                ((Rectangle)object).getArea());
        }
    }
}

class Circle {
    double radius = 1.0;
    Circle(){}
    Circle(double radius ){
        this.radius=radius;
    }
    double getArea(){
        return radius * radius * 3.14159;
    }
    double getDiameter(){
        return 2*radius * 3.14159;
    }
}

class Rectangle {
    double width = 1.0;
    double high = 1.0;
    Rectangle(double width, double high ){
        this.width = width;
    }
}
```

```

        this.high = high;
    }
    double getArea(){
        return width * high ;
    }
    double getDiameter(){
        return 2*width * high;
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

cle area is 3.14159
The circle diameter is 6.28318
The rectangle area is 1.0

```

จากตัวอย่างที่ 9.4 เป็นตัวอย่างของการใช้หลักการของการพ้องรูป (Polymorphism) ในการเขียนโปรแกรม ภายในโปรแกรมหกกล่าวว่ามีคลาสที่สร้างขึ้นคือ Circle และ Rectangle โดยคลาสสองคลาสดังกล่าวถือเป็นคลาสลูกของคลาส Object โดยอัตโนมัติ

การประกาศ Object object1 = new Circle(1); ใน public static void main(String[] args) เป็นการประกาศตัวแปรอ้างอิงวัตถุที่ชื่อ object1 โดยมีชนิดข้อมูลเป็น Object โดยเก็บ Reference ของวัตถุที่สร้างจากคลาส Circle

การประกาศ Object object2 = new Rectangle(1, 1); ใน public static void main(String[] args) เป็นการประกาศตัวแปรอ้างอิงวัตถุที่ชื่อ object2 โดยมีชนิดข้อมูลเป็น Object โดยเก็บ Reference ของวัตถุที่สร้างจากคลาส Rectangle

ภายในคลาส TestPolymorphismCasting นอกจากจะมีเมธอด public static void main(String[] args) แล้ว ภายในคลาสดังกล่าวยังมีเมธอด public static void displayObject(Object object) โดยการทำงานของเมธอดจะรับวัตถุที่มีชนิดข้อมูลเป็น Object ทำให้เมธอดนี้สามารถรับวัตถุต่าง ๆ ที่มีชนิดเป็น Object และชนิดที่เป็นคลาสลูกของคลาส Object เข้ามาทำงานภายในเมธอดนี้ได้ ซึ่งเป็นตัวอย่างของการทำโพลีมอร์ฟิซึม

ดังนั้นจะพบว่าเราสามารถเรียกเมธอด public static void displayObject(Object object) โดยการส่งผ่าน object1 และ object2 เข้าไปภายในเมธอดนี้ได้ดังคำสั่ง

```

displayObject(object1);
displayObject(object2);

```

9.6 บทสรุป

หลักการโพลีมอร์ฟิซึมมีดังต่อไปนี้

1. ความสัมพันธ์แบบการสืบทอดคุณสมบัติ อนุญาตให้คลาสสามารถสืบทอดคุณสมบัติจากคลาสแม่ (Superclass) ได้
2. คลาสลูก(Subclass) สามารถเพิ่มคุณสมบัติใหม่ ๆ ได้ โดย Subclass จะเป็นคลาสที่เจาะจงลงไปจากคลาสแม่
3. ทุก ๆ วัตถุที่สร้างจาก Subclass จะถือว่าเป็น instance ของ superclass ด้วย เช่น
 - a. ทุก ๆ วัตถุที่สร้างจากคลาส circle ถือเป็นวัตถุของคลาส Object
 - b. แต่ทุก ๆ วัตถุของคลาส Object ไม่ได้เป็นวัตถุชนิด circle

4. กรณีที่ในเมธอดมีการประกาศรับพารามิเตอร์เป็นแบบ instance ของ superclass เราสามารถส่ง instance ของ subclass ไปเป็นพารามิเตอร์ของ superclass ได้
5. ตัวแปรอ้างอิงของซูเปอร์คลาสสามารถชี้ไปยังอินสแตนซ์ของซับคลาส
6. แต่ตัวแปรอ้างอิงของซับคลาสไม่สามารถชี้ไปยังอินสแตนซ์ของซูเปอร์คลาสหรืออินสแตนซ์ของคลาสที่มีซูเปอร์คลาสเดียวกันได้ ถ้าจะลองนำไปชี้คลาสอื่นๆที่ไม่มีความเกี่ยวข้องกันเลยยิ่งไม่ได้
7. การที่ตัวแปรของซูเปอร์คลาสสามารถชี้ไปยังอินสแตนซ์ของซับคลาสได้ เป็นการเพิ่มความยืดหยุ่นในการเขียนโปรแกรม ลักษณะนี้เรียกว่า โพลิมอร์ฟิซึม

Dynamic Binding มีการทำงานดังนี้ กำหนดให้วัตถุ o เป็นอินสแตนซ์ของคลาส $C_1, C_2, \dots, C_{n-1}, C_n$ เมื่อ C_1 เป็นคลาสลูกของ C_2 และ C_2 เป็นคลาสลูกของ C_3, \dots และ C_{n-1} เป็นคลาสลูกของ C_n ดังปรากฏในรูปกล่าวคือ C_n จะเป็นคลาสที่มีความเป็นทั่วไปมากที่สุด (most general class) C_1 เป็นคลาสที่มีความเจาะจงมากที่สุด (most specific class) โดยคลาส C_n ในภาษาจาวาคือคลาส Object ถ้า o เป็นอินสแตนซ์ของคลาส C_1 โดยที่ o เรียกเมธอด p แล้ว JVM จะทำการค้นหาเมธอดที่ตรงกับที่ต้องการมากที่สุดโดยจะค้นหาเมธอด p ในคลาส $C_1, C_2, \dots, C_{n-1}, C_n$ ตามลำดับจนกว่าจะพบ

Static Binding คือการผูกตัวแปรอ้างอิงกับชนิดข้อมูลช่วง Compile time

9.7 แบบฝึกหัดปฏิบัติการ

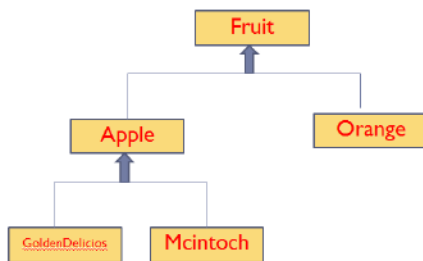
1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสืบทอดคลาสจากตัวอย่างต่อไปนี้

<pre>class A{ void f(){ System.out.println("A"); } } class B extends A{ void f(){ System.out.println("B"); } } class Test38{ static void t(A a){ a.f(); } public static void main(String[] args){ t(new A()); t(new B()); } }</pre>	<p>ให้อธิบายผลลัพธ์ของโปรแกรมพร้อมอธิบายเหตุใดจึงได้คำตอบดังกล่าว</p>
<pre>class A{ int x=1; void inc(){x++;} int get(){return x;} } class B extends A{ int x=2; void inc(){x++;} int get(){return x;} } class Test{ public static void main(String[] args){ B b =new B(); A a= b; a.inc(); b.inc(); System.out.println(a.x+","+b.x); System.out.println(a.get()+" "+b.get()); } }</pre>	<p>ให้อธิบายถึงส่วนที่มึการทำงานเป็น Static binding และ Dynamic binding</p>
<pre>public class Test{ public static void main(String[] args){ Object fruit = new Fruit(); Object apple= new (Apple)fruit; } } class Apple extends Fruit{ } class Fruit{ }</pre>	<p>ให้หาส่วนที่ผิดของโปรแกรม</p>

2. จงอธิบายผลลัพธ์ของ Output ต่อไปนี้

<pre>public class Test { public static void main(String[] args) { Object a1 = new A(); Object a2 = new Object(); System.out.println(a1); System.out.println(a2); } } class A { int x; public String toString() { return "A's x is " + x; } }</pre>	
<pre>public class Test { public static void main(String[] args) { Object a1 = new A(); Object a2 = new A(); System.out.println(a1.equals(a2)); } } class A { int x; public boolean equals(A a) { return this.x == a.x; } }</pre>	
<pre>public class Test { public static void main(String[] args) { A a1 = new A(); A a2 = new A(); System.out.println(a1.equals(a2)); } } class A { int x; public boolean equals(A a) { return this.x == a.x; } }</pre>	

3. กำหนดให้ Fruit, Apple, Orange, Golden Delicious Apple, และ Macintosh Apple มีโครงสร้างการสืบทอดและการประกาศตัวแปรดังนี้ ให้เขียนโปรแกรมเพื่อจำลองการทำงานของโครงสร้างของคลาสดังกล่าวพร้อมตอบคำถามต่อไปนี้



กำหนดการทำงานภายในmain() มีการสร้าง Object ดังนี้

```
Fruit fruit = new GoldenDelicious();
```

```
Orange orange = new Orange();
```

จงตอบคำถามต่อไปนี้

1. Is fruit instanceof Orange?
2. Is fruit instanceof Apple?
3. Is fruit instanceof GoldenDelicious?
4. Is fruit instanceof Macintosh?
5. Is orange instanceof Orange?
6. Is orange instanceof Fruit?
7. Is orange instanceof Apple?
8. Suppose the method makeApple is defined in the Apple class. Can fruit invoke this method?
Can orange invoke this method?
9. Suppose the method makeOrangeJuice is defined in the Orange class. Can orange invoke this method? Can fruit invoke this method?

4. พิจารณาส่วส่วนของโปรแกรมต่อไปนี้

```

1 public class Test{
2     public static void main(String[] args){
3         Animal x= new Tiger();
4         System.out.println("1. x.news is "+ x.news);
5         System.out.println("2. ((Tiger)x).news is "+ ((Tiger)x).news);
6         System.out.println("3. x.smile() is "+ x.smile());
7         System.out.println("4. ((Tiger)x).smile() is "+ ((Tiger)x).smile());
8         System.out.println("5. x.getNews() is "+ x.getNews());
9         System.out.println("6. x.getMessage() is "+ x.getMessage());
10
11     }
12 }
13
14 class Animal{
15     public String news= "Animal's news";
16     public String message="Animal's message";
17     public static String smile(){
18         return "smile from Animal";
19     }
20     public String getNews(){
21         return news;
22     }
23     public String getMessage(){

```

```

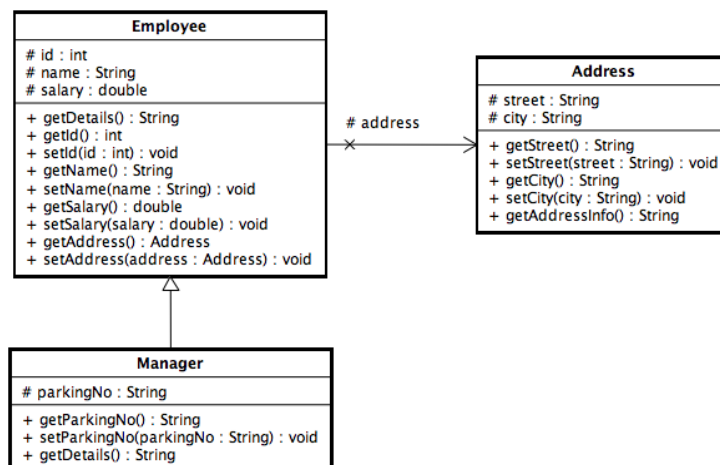
24     return message;
25 }
26 }
27 class Tiger extends Animal{
28     public String news= "Tiger's news";
29     public String message="Tiger's message";
30     public static String smile(){
31         return "smile from Tiger";
32     }
33     //@override
34     public String getNews(){
35         return news;
36     }
37 }

```

ผลลัพธ์ที่จะเกิดขึ้นจากการใช้โปรแกรมต่อไปนี้คืออะไร

อธิบายเหตุผลประกอบคำตอบ

5. กำหนดความสัมพันธ์ของคลาส Employee และ Manager ในรูปของไดอะแกรม ดังนี้



ให้เขียนโปรแกรมเพื่อจำลองการทำงานของโครงสร้างของคลาสดังกล่าว และประยุกต์ใช้ Polymorphism สำหรับการสร้างวัตถุชนิด **Manager** โดยกำหนดให้สร้างวัตถุในคลาส Main ด้วยคำสั่ง `Employee emp = new Manager();` โดยให้กำหนดที่อยู่ เงินเดือนและรายละเอียดของพนักงานได้พร้อมทั้งสามารถพิมพ์รายละเอียดของวัตถุดังกล่าวได้

บทที่ 10 คลาสนามธรรมและอินเทอร์เฟส

วัตถุประสงค์

10.1 สามารถอธิบายความหมายของคลาสนามธรรมและอินเทอร์เฟส

10.2 สามารถเขียนโปรแกรมเพื่อประยุกต์การใช้คุณสมบัติคลาสนามธรรมและอินเทอร์เฟส

10.1 ความนำ

ภาษาจาวากำหนดให้เราสามารถที่จะนิยามเมธอดต่างๆภายในคลาส ขึ้นมาก่อนได้โดยไม่ต้องกำหนดคำสั่งต่างๆภายในเมธอดเหล่านั้น แต่จะให้มีการกำหนดคำสั่งภายในคลาสอื่นๆ ที่จะสืบทอด วิธีการนี้จะช่วยให้เราสามารถจะกำหนดเมธอดที่ไม่สามารถนิยามคำสั่งไว้ล่วงหน้าได้ โดยภาษาจาวาได้กำหนดคลาสลักษณะนี้ไว้สองแบบคือคลาสแบบ abstract และอินเทอร์เฟส

10.2 คลาสนามธรรม (Abstract Class)

คลาสที่มี modifier เป็น abstract จะหมายความว่าคลาสนั้นยังเป็นคลาสที่ไม่สมบูรณ์ โดยมีเมธอดแบบ abstract ซึ่งเป็นเมธอดที่ยังไม่สมบูรณ์อย่างน้อยหนึ่งเมธอดอยู่ในคลาสหรือเราสามารถกำหนดคลาสที่เป็น abstract class ที่ไม่มี abstract method ได้โดยมีเจตนาเพื่อให้คลาสดังกล่าวใช้เพื่อการสืบทอดไปยังคลาสลูกเท่านั้น

เมธอดแบบ abstract จะมีรูปแบบการประกาศดังนี้

```
[modifier] abstract return_type methodName([arguments]);
```

ทั้งนี้เมธอดแบบ abstract เป็นเมธอดที่ยังไม่สมบูรณ์ เนื่องจากไม่มีบล็อกคำสั่งอยู่ภายในเมธอด

ตัวอย่างที่ 10.1 คลาสนามธรรม (Abstract Class)

```
public abstract class Student{
    protected String name;
    protected String surname;
    protected String grade;
    protected double gpa;
    protected int[] score;
    public Student(){
        this("No Name", "No Surname");
    }
    public Student(String name,String surname){
        this.name = name;
        this.surname = surname;
        this.score = new int[3];
        this.grade = "";
    }
    public abstract void calculateGrade();
    public String getName(){
        return this.name;
    }
    public String getSurName(){
        return this.surname;
    }
    public String getGrade(){
        return this.grade;
    }
    public double getGPA(){
        return this.gpa;
    }
}
```

```

public int[] getScore(){
    return this.score;
}
public void setName(String name){
    this.name=name;
}
public void setSurName(String surname){
    this.surname=surname;
}
public void setGrade(String grade){
    this.grade=grade;
}
public void setGPA(double gpa){
    this.gpa=gpa;
}
public void setScore(int index,int testScore){
    score[index - 1] = testScore;
}
}

```

จากตัวอย่างข้างต้นเมธอด `public abstract void calculateGrade();` ถือเป็นเมธอดแบบ `abstract method` เนื่องจากยังไม่มีเขียนรายละเอียดการทำงาน ทำให้คลาส `Student` ถือเป็น `Abstract Class`

คลาสแบบ `abstract` กำหนดขึ้นมาเพื่อให้คลาสอื่นสืบทอด โดยคลาสที่มาสืบทอดจะต้องกำหนดคำสั่งในเมธอดที่ยังไม่สมบูรณ์ ทั้งนี้เราจะไม่สามารถสร้างวัตถุของคลาสแบบ `abstract` ได้

อนึ่งถึงแม้ว่าเราจะไม่สามารถสร้างวัตถุของคลาสแบบ `abstract` ได้แต่เราสามารถประกาศวัตถุของคลาสแบบ `abstract` และสร้างวัตถุของคลาสที่ `extends` คลาสแบบ `abstract` ได้ตามหลักการของการมีได้หลายรูปแบบดังนี้

```
AbstractClass s = new ExtendsClass();
```

ตัวอย่างที่ 10.2 การประกาศวัตถุของคลาสแบบ abstract และสร้างวัตถุของคลาสที่ extends คลาสแบบ abstract

```

public abstract class Student{
    protected String name;
    protected String surname;
    protected String grade;
    protected double gpa;
    protected int[] score;
    public Student(){
        this("No Name", "No Surname");
    }
    public Student(String name,String surname){
        this.name = name;
        this.surname = surname;
        this.score = new int[3];
        this.grade = "";
    }
    public abstract void calculateGrade();
    public String getName(){
        return this.name;
    }
    public String getSurName(){
        return this.surname;
    }
    public String getGrade(){
        return this.grade;
    }
    public double getGPA(){
        return this.gpa;
    }
    public int[] getScore(){
        return this.score;
    }
    public void setName(String name){
        this.name=name;
    }
    public void setSurName(String surname){
        this.surname=surname;
    }
    public void setGrade(String grade){
        this.grade=grade;
    }
    public void setGPA(double gpa){
        this.gpa=gpa;
    }
    public void setScore(int index,int testScore){
        score[index - 1] = testScore;
    }
}

class BachelorStudent extends Student {
    public BachelorStudent(){
        super("No Name", "No Surname");
    }
    public BachelorStudent(String name,String surname){
        super(name,surname);
    }
    public void calculateGrade(){
        int sum = 0;
        for(int i = 0;i < score.length; i++){
            sum += score[i];
        }
    }
}

```

```

    }
    if(sum >= 80){
        grade = "A";
    }else if(sum >= 75) {
        grade = "B+";
    }else if(sum >= 70) {
        grade = "B";
    }else if(sum >= 65) {
        grade = "C+";
    }else if(sum >= 60) {
        grade = "C";
    }else if(sum >= 55) {
        grade = "D+";
    }else if(sum >= 50) {
        grade = "D";
    }else{
        grade = "F";
    }
}
}

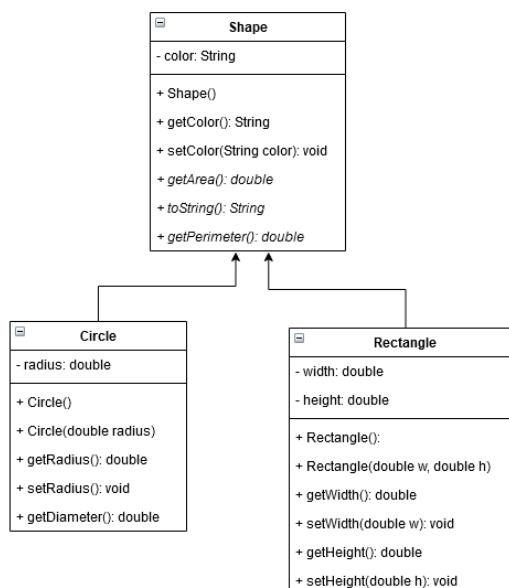
```

จากตัวอย่างข้างต้นคลาส Student ถูกประกาศเป็น Abstract Class ภายในคลาสมีเมธอด public abstract void calculateGrade(); ถือเป็นเมธอดแบบ abstract method เนื่องจากยังไม่มีเขียนรายละเอียดการทำงาน ทำให้คลาส Student ถือเป็น Abstract Class ในกรณีที่ต้องสร้างคลาสใหม่ซึ่งเป็นคลาสลูกของคลาส Student ชื่อ BachelorStudent จะทำได้โดยการเขียนคำสั่งต่อไปนี้

```
class BachelorStudent extends Student
```

เนื่องจากคลาส Student เป็น Abstract Class ดังนั้นในคลาส BachelorStudent หากต้องการให้คลาสดังกล่าวสามารถนำไปสร้างวัตถุได้จะต้องทำการเขียนรายละเอียดของเมธอด public abstract void calculateGrade() { } ดังตัวอย่างข้างต้น

ตัวอย่างที่ 10.3 การประกาศวัตถุของคลาสแบบ abstract และสร้างวัตถุของคลาสที่ extends คลาสแบบ abstract



```
public abstract class Shape {
    private String color = "white";
    protected Shape() {
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public abstract double getArea();
    public abstract String toString();
    public abstract double getPerimeter();
}

class Circle extends Shape {
    private double radius;
    public Circle() {
    }
    public Circle(double radius) {
        this.radius = radius;
    }
    public double getRadius() {
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }
    public double getArea() {
        return radius * radius * Math.PI;
    }
    public double getDiameter() {
        return 2 * radius;
    }
    public double getPerimeter() {
        return 2 * radius * Math.PI;
    }
    public String toString() {
        return "The circle's radius is " + radius;
    }
}

class Rectangle extends Shape {
    private double width;
    private double height;
    public Rectangle() {
    }
    public Rectangle(double w, double h) {
        this.width = w;
        this.height = h;
    }
    public double getWidth() {
        return width;
    }
    public void setWidth(double width) {
        this.width = width;
    }
    public double getHeight() {
        return height;
    }
    public void setHeight(double height) {
        this.height = height;
    }
}
```



```

    }
    public double getArea() {
        return width * height;
    }
    public double getPerimeter() {
        return width * height;
    }
    public String toString() {
        return "The Rectangle's width is " + width+" Rectangle's height is " +height;
    }
}

```

จากตัวอย่างที่ 10.3 คลาส Shape ถูกประกาศเป็น Abstract Class ภายในคลาสมีเมธอด

```
public abstract double getArea();
```

```
public abstract String toString();
```

```
public abstract double getPerimeter();
```

ถือเป็นเมธอดแบบ abstract method เนื่องจากยังไม่มีวิธีการเขียนรายละเอียดการทำงาน ทำให้คลาส Shape ถือเป็น Abstract Class

ในกรณีที่ต้องสร้างคลาสใหม่ซึ่งเป็นคลาสลูกของคลาส Shape ชื่อ Circle และ Rectangle จะทำได้โดยการเขียนคำสั่งต่อไปนี้

```
class Circle extends Shape{ }
```

```
class Rectangle extends Shape{ }
```

เนื่องจากคลาส Shape เป็น Abstract Class ดังนั้นในคลาส Circle และ Rectangle หากต้องการให้คลาสดังกล่าวสามารถนำไปสร้างวัตถุได้จะต้องทำการเขียนรายละเอียดของเมธอด

```
public abstract double getArea(){ }
```

```
public abstract String toString(){ }
```

```
public abstract double getPerimeter(){ }
```

ดังตัวอย่างข้างต้น

10.3 อินเทอร์เฟซ (Interface)

ในภาษา Java ใช้ interface เป็นตัวเชื่อมประสานระหว่างคลาสที่ไม่มีความสัมพันธ์กัน แต่ต้องการที่จะติดต่อหรือส่งข้อมูลให้กับอีกรหัสของอีกคลาสหนึ่ง การสร้าง interface จะคล้ายกับการสร้างคลาสแต่การประกาศจะแตกต่างจากการสร้างคลาสและมีข้อจำกัดมากกว่า ดังนี้

1. Interface ไม่สามารถมีตัวแปรอ้างอิงวัตถุ แต่สามารถมีตัวแปรค่าคงที่ได้ การประกาศตัวแปรค่าคงที่ไม่จำเป็นต้องประกาศเป็น public static final เพราะ compiler จะเติมให้โดยอัตโนมัติ
2. เมธอดต่างๆ ใน interface จะต้องเป็น abstract method เท่านั้น กล่าวคือ มีแค่ชื่อตัวแปรที่ส่งผ่านค่า ซึ่งจะแตกต่างจาก abstract class ที่เมธอดใน abstract class ไม่จำเป็นต้องเป็น abstract method ทุกเมธอด
3. เมธอดต่างๆ ต้องมี access modifier เป็น public อย่างเดียวเท่านั้น หากไม่ระบุเป็น public จะใช้ได้เฉพาะภายใน package เท่านั้น

รูปแบบ

```
public interface InterfaceName extends SuperInterfaces {
    constant declarations;
    method signatures;
}
```

จากรูปแบบการสร้าง interface สามารถสรุปลักษณะของ interface ได้ดังนี้

1. interface สามารถขยายเพิ่มจาก interface อื่นได้ เช่น SuperInterfaces และสามารถเพิ่มได้หลายๆ interface ในคราวเดียวกัน ซึ่งจะแตกต่างจากคลาสที่สืบทอดได้เพียงคลาสเดียวเท่านั้น การสืบทอดของ interface จะใช้หลักการเดียวกันกับการสืบทอดของคลาสแต่สิ่งที่สืบทอดได้คือ ตัวแปรค่าคงที่ และ abstract method เท่านั้น

2. คำสั่งในส่วนของ interface กรณีเป็น abstract method จะมีรูปแบบดังนี้

```
access_modifier return_type methodName( parameterList );
```

3. ไม่จำเป็นต้องใส่เครื่องหมาย { }

access_modifier ของเมธอดโดยปกติจะเป็น public abstract โดยอัตโนมัติในการใช้งาน interface

ในภาษา Java นั้นคลาสต่างๆ สามารถสืบทอดจาก superclass ได้เพียงคลาสเดียว ส่วนการใช้ interface สามารถที่จะใช้ได้ไม่จำกัด

รูปแบบ class SubClassName implements InterfaceName1, InterfaceName2, ... { ... }

ตัวอย่างที่ .104 ตัวอย่างการใช้งานอินเทอร์เฟซ

```
public interface Edible {
    public abstract String howToEat();
}
abstract class Animal {
    public abstract String sound();
}
class Chicken extends Animal implements Edible {
    @Override
    public String howToEat() {
        return "Chicken: Fry it";
    }
    @Override
    public String sound() {
        return "Chicken: cock-a-doodle-doo";
    }
}
class Tiger extends Animal {
    @Override
```

```

    public String sound() {
        return "Tiger: RROOAAARR";
    }
}
abstract class Fruit implements Edible {
}

class Apple extends Fruit {
    @Override
    public String howToEat() {
        return "Apple: Make apple cider";
    }
}
class Orange extends Fruit {
    @Override
    public String howToEat() {
        return "Orange: Make orange juice";
    }
}
class TestEdible {
    public static void main(String[] args) {
        Object[] objects = {new Tiger(), new Chicken(), new Apple()};
        for (int i = 0; i < objects.length; i++) {
            if (objects[i] instanceof Edible)
                System.out.println(((Edible)objects[i]).howToEat());
            if (objects[i] instanceof Animal) {
                System.out.println(((Animal)objects[i]).sound());
            }
        }
    }
}

```

จากตัวอย่างที่ 10.4 ภายในโปรแกรมได้สร้าง interface Edible ซึ่งภายในมีเมธอดที่ยังไม่สมบูรณ์ชื่อ public abstract String howToEat(); ถือเป็นเมธอดแบบ abstract method เนื่องจากยังไม่มีกรเขียนรายละเอียดการทำงาน

นอกจากนี้ภายในโปรแกรมมี abstract class ที่ชื่อ Animal และมี abstract method ชื่อ public abstract String sound(); เนื่องจากยังไม่มีกรเขียนรายละเอียดการทำงานคลาสแบบ abstract กำหนดขึ้นมาเพื่อให้คลาสอื่นสืบทอด โดยคลาสที่มาสืบทอดจะต้องกำหนดคำสั่งในเมธอดที่ยังไม่สมบูรณ์ ทั้งนี้เราจะไม่สามารถจะสร้างวัตถุของคลาสแบบ abstract ได้

ในกรณีที่ต้องสร้างคลาสใหม่ซึ่งเป็นคลาสลูกของคลาส Animal และ interface Edible ชื่อ Chicken จะทำได้โดยการเขียนคำสั่งต่อไปนี้

```
class Chicken extends Animal implements Edible{ }
```

เนื่องจากคลาส Animal เป็น Abstract Class และ interface Edible ยังมีเมธอดที่ยังไม่สมบูรณ์ดังนั้นในคลาส Chicken หากต้องการให้คลาสดังกล่าวสามารถนำไปสร้างวัตถุได้จะต้องทำการเขียนรายละเอียดของเมธอด

```
    public String howToEat() { }
```

```
    public String sound() { }
```

ดังตัวอย่างข้างต้น

หมายเหตุ @Override เป็น annotation ใช้เพื่อแสดงว่าเป็นการโอเวอร์ไรด์เมธอดจากคลาสแม่ที่คลาสลูกสืบทอดมา มีประโยชน์ในกรณีที่ต้องการแสดงว่าการโอเวอร์ไรด์ดังกล่าวใส่ชื่อ เมธอด พารามิเตอร์ ชนิดข้อมูลส่งกลับตรงตามคลาสแม่

การประยุกต์ใช้งาน Interface สามารถประยุกต์ได้ 2 แบบ คือ

1. ประยุกต์โดยใช้คำสั่ง `. implements` ตามรูปแบบข้างต้น การเขียนโปรแกรมด้วยคำสั่งข้างต้นทำให้ต้องเขียน method ต่างๆ ของ interface ใน class นั้นทั้งหมด ถึงแม้จะไม่ใช้งานก็ตาม

2. การประยุกต์ใช้ `. interface` แบบ inner class ในการประยุกต์แบบนี้เมธอดของ interface ไม่สามารถเรียกประมวลผลได้ ยกเว้นคลาสนั้นจะเขียนเมธอดประมวลผลเมธอดของ interface ให้ไม่ว่าจะประยุกต์แบบใดก็ตาม การเรียกใช้งานต้องมีการสร้างวัตถุก่อน

10.4 บทสรุป

เราสามารถที่จะนิยามเมธอดต่างๆ ภายในคลาสขึ้นมาก่อนได้โดยไม่ต้องกำหนดคำสั่งต่างๆ ภายในเมธอดเหล่านั้น แต่จะให้มีการกำหนดคำสั่งภายในคลาสอื่นๆ ที่จะสืบทอด วิธีการนี้จะช่วยให้เราสามารถจะกำหนดเมธอดที่ไม่สามารถนิยามคำสั่งไว้ล่วงหน้าได้ ภาษาจาวาได้กำหนดคลาสลักษณะนี้ไว้สองแบบคือ คลาสแบบ Abstract และ อินเตอร์เฟส

คลาสแบบ Abstract

คลาสที่มี modifier เป็น `abstract` จะหมายความว่าคลาสนั้นยังเป็นคลาสที่ไม่สมบูรณ์ โดยมีเมธอดแบบ `abstract` ซึ่งเป็นเมธอดที่ยังไม่สมบูรณ์อย่างน้อยหนึ่งเมธอดอยู่ในคลาสหรือเราสามารถกำหนดคลาสที่เป็น `abstract class` ที่ไม่มี `abstract method` ได้โดยมีเจตนาเพื่อให้คลาสดังกล่าวใช้เพื่อการสืบทอดไปยังคลาสลูกเท่านั้น

`abstract method` จะต้องอยู่ในคลาสที่เป็น `abstract class`

ถ้าคลาสลูกของ คลาสแม่ที่เป็น `abstract` ไม่เขียนรายละเอียดของ `abstract methods` ทั้งหมด ตัวของคลาสลูกเองต้องกำหนดเป็น `abstract`

คลาสลูกที่ไม่ได้เป็น `abstract` ที่สืบทอดมาจากคลาสแม่ที่เป็น `abstract` ทุก ๆ `abstract methods` จะต้องถูกเขียนรายละเอียดการทำงานแม้ว่าจะไม่ได้ใช้ใน `subclass` ก็ตาม

ไม่สามารถสร้างวัตถุจาก `abstract class` ด้วยโอเปอเรเตอร์ `new` แต่สามารถใช้ `abstract class` เพื่อเป็น `data type` เช่น `GeometricObject[] geo = new GeometricObject[10];`

สามารถที่จะสร้าง `constructors` ได้โดยสามารถถูกเรียกได้จาก `constructors` ของคลาสลูก เช่น `constructors` ของ `GeometricObject` สามารถถูกเรียกจากคลาส `Circle` และคลาส `Rectangle` ได้

สามารถสร้าง `abstract class` ที่ไม่มี `abstract methods` เลยก็ได้ในกรณีนี้เราไม่สามารถสร้างวัตถุจากคลาสดังกล่าวโดยการใช้โอเปอเรเตอร์ `new` โดยคลาสดังกล่าวจะถูกใช้เป็น `base class` เพื่อสร้าง `subclass` ตัวใหม่

คลาสลูกสามารถเป็น `abstract` ได้แม้ว่า `superclass` จะเป็นคลาสปกติ (`concrete class`) เช่น

```
abstract class GeometricObject extends Object
```

คลาสลูกสามารถโอเวอร์ไรต์เมธอดจากคลาสแม่เพื่อให้เป็น `abstract method` ก็ได้

อินเทอร์เฟส

อินเทอร์เฟส คือโครงสร้างที่มีลักษณะคล้ายคลาส ส่วนประกอบของอินเทอร์เฟสจะมีเฉพาะค่าคงที่ constants abstract methods

จุดประสงค์ของการสร้าง interface จะคล้าย abstract class แต่จุดประสงค์หลักจริง ๆ คือการกำหนดพฤติกรรม behavior สำหรับ objects เช่น Comparable, Edible, Cloneable

interface เป็นคลาสพิเศษในภาษาจาวาโดย ทุกๆ interface จะถูกคอมไพล์เป็น bytecode เหมือนคลาสปกติ ไม่สามารถสร้างวัตถุจาก interface โดยใช้ new แต่สามารถใช้สำหรับการประกาศชนิดข้อมูลให้กับตัวแปร

10.5 แบบฝึกหัดปฏิบัติการ

1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสร้างอินเทอร์เฟซจากตัวอย่างต่อไปนี้

```

1 public class House implements Cloneable, Comparable<House> {
2     private int id;
3     private double area;
4     private java.util.Date whenBuilt;
5     public House(int id, double area) {
6         this.id = id;
7         this.area = area;
8         whenBuilt = new java.util.Date();
9     }
10    public int getId() {
11        return id;
12    }
13    public double getArea() {
14        return area;
15    }
16    public java.util.Date getWhenBuilt() {
17        return whenBuilt;
18    }
19    @Override
20    public Object clone() {
21        try {
22            return super.clone();
23        }
24        catch (CloneNotSupportedException ex) {
25            return null;
26        }
27    }
28    }
29    @Override // Implement the compareTo method defined in Comparable
30    public int compareTo(House o) {
31        if (area > o.area)
32            return 1;
33        else if (area < o.area)
34            return -1;
35        else
36            return 0;
37    }
38 }

```

1.1 ให้อธิบายการทำงานของโปรแกรมต่อไปนี้

บรรทัดที่ 1

บรรทัดที่ 4-2

บรรทัดที่ 9-5

บรรทัดที่ 18-10

บรรทัดที่ 28-21

บรรทัดที่ 37-30

1.2 สร้างคลาสทดสอบ และเพิ่มคำสั่งต่อไปนี้ใน main() ผลลัพธ์จะมีผลลัพธ์อย่างไร

```

House house1 = new House(1, 1750.50);
House house2 = (House)house1.clone();

}

```

2. [Application] จงใช้ Employee ที่ให้มาต่อไปนี้เป็นคลาสแม่

```
public abstract class Employee{
    private String firstname;
    private String lastname;
    private String id;
    public Employee(String firstname,String lastname,String id){
        this.firstname=firstname;
        this.lastname=lastname;
        this.id=id;
    }
    public double earning(){}
    public double bonus(int year){}
}
```

2.1 จงสร้างคลาสลูกของ Employee ชื่อ SalariedEmployee ซึ่งมีรายละเอียดดังนี้

1. มี Instance variable เพิ่มจาก Employee ชื่อ salary สำหรับบันทึกค่าเงินเดือนของพนักงาน
2. มี Constructor ที่รับข้อมูลดังนี้

```
public SalariedEmployee(String firstname,String lastname,String id, double sal)
```

เพื่อส่งพารามิเตอร์สามตัวแรกไปยัง constructor ใน employee ให้ทำการกำหนดค่าเบื้องต้นให้กับ

firstname lastname id ส่วนพารามิเตอร์ตัวสุดท้ายใช้สำหรับกำหนดค่าให้กับ salary

3. ให้ method bonus (int year) มีการประมวลผลโดยรับพารามิเตอร์ year ซึ่งเป็นระยะเวลาการทำงานของพนักงานเข้ามา เพื่อทำการตรวจสอบว่าถ้าระยะเวลาเกิน 5 ปี จะคืนค่าโบนัสเป็น 12 เท่าของเงินเดือน ไม่เช่นนั้นจะคืนค่าโบนัสเป็น 6 เท่าของเงินเดือน
4. ให้ Method earning () มีการประมวลผลโดยนำเงินเดือนของพนักงานแต่ละคนมาทำการหักภาษี 5% ของเงินเดือน แล้ว return เงินเดือนที่หักภาษีแล้ว

2.2 จงสร้างคลาสลูกของ Employee ชื่อ ComEmployee ซึ่งมีรายละเอียดดังนี้

1. มี Instance variable เพิ่มจาก Employee ชื่อ grossSale สำหรับบันทึกยอดขายที่ได้รับ และ ComRate ที่มีชนิดข้อมูลเป็นเลขจำนวนจริง เพื่อบันทึกเปอร์เซ็นต์ของคอมมิชชันที่ได้
2. มี Constructor ที่รับข้อมูลดังนี้

```
public ComEmployee(String firstname,String lastname,String id, double sales, double percent)
```

เพื่อส่งพารามิเตอร์สามตัวแรกไปยัง constructor ใน employee ให้ทำการกำหนดค่าเบื้องต้นให้กับ firstname lastname id ส่วนพารามิเตอร์ sale และ percent ใช้สำหรับกำหนดค่าให้กับ grossSale และ ComRate

3. ให้ Method bonus (int year) มีการประมวลผลโดยรับพารามิเตอร์ year ซึ่งเป็นระยะเวลาการทำงาน of พนักงานเข้ามา เพื่อทำการตรวจสอบว่าถ้าระยะเวลาเกิน 5 ปี จะคืนค่าโบนัสเป็น 6 เท่าของยอดขาย ไม่เช่นนั้นจะคืนค่าโบนัสเป็น 3 เท่าของยอดขาย
4. ให้ Method earning () มีการประมวลผลโดยนำยอดขาย ของพนักงานแต่ละคนมารวมกับค่าคอมมิชชัน (Comission=grossSale*ComRate) แล้ว return ผลรวมที่ได้

2.3. จงสร้าง test class ชื่อ Final.java ซึ่งมีรายละเอียดต่อไปนี้

1. ให้สร้าง arrayList ขึ้นมา 1 arrayList เพื่อเก็บ ข้อมูลของพนักงานทุกประเภทไว้ด้วยกันให้เป็นพนักงานชนิด SalariedEmployee 2 คน และพนักงานชนิด ComEmployee 2 คน

2. ให้สร้าง method ชื่อ printEmp(ArrayList a) โดยรับพารามิเตอร์ที่เป็นข้อมูลชนิด ArrayList แล้วทำการคำนวณหา earning และ bonus ของแต่ละคนโดยใช้คำสั่งเดียวกันคือ r.earning() และ r.bonus() จากนั้นทำการสร้าง array ชื่อ arrayEarn สำหรับเก็บข้อมูล earning ของพนักงานทุกคน และสร้าง array ชื่อ arrayBonus สำหรับเก็บข้อมูลโบนัสของพนักงานทุกคน ท้ายสุดให้ method printEmp(ArrayList a) นี้ทำการพิมพ์รายงานที่มีรูปแบบดังด้านล่างนี้ออกทาง dialog box

Firsr name	Last name	Earning	Bonus
------------	-----------	---------	-------

3. กำหนดให้ GeometricObject.java มีรายละเอียดดังนี้

```
public abstract class GeometricObject {
    private String color = "white";
    private boolean filled;
    protected GeometricObject() {
    }
    protected GeometricObject(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public boolean isFilled() {
        return filled;
    }
    public void setFilled(boolean filled) {
        this.filled = filled;
    }
    public abstract double getArea();
    public abstract double getPerimeter();
}
```

3.1 ให้สร้างคลาส Circle ที่ extend GeometricObject และ implements อินเตอร์เฟส Comparable ให้ทำการ override เมธอด equals() ของคลาส Object และ implements เมธอด compareTo() เพื่อทำการเปรียบเทียบวงกลมจากรัศมีของวงกลม โดยที่วงกลมสองวงจะเท่ากันก็ต่อเมื่อมีรัศมีเท่ากัน วาดคลาสไดอะแกรมและเขียนโปรแกรมทดสอบคลาสดังกล่าว

3.2 ให้สร้างคลาส Rectangle ที่ extends GeometricObject และ implements อินเตอร์เฟส Comparable ให้ทำการ override เมธอด equals() ของคลาส Object และ implements เมธอด compareTo() เพื่อทำการเปรียบเทียบ Rectangle จากพื้นที่ของ Rectangle โดยที่สี่เหลี่ยมจะเท่ากันก็ต่อเมื่อมีพื้นที่เท่ากัน วาดคลาสไดอะแกรมและเขียนโปรแกรมทดสอบคลาสดังกล่าว

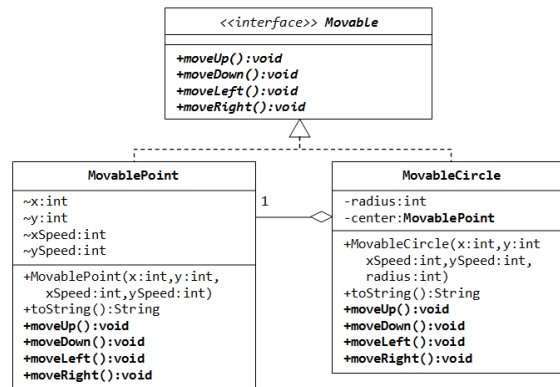
3.3 ให้สร้างคลาส Octagon ที่ extends GeometricObject และ implements อินเตอร์เฟส Comparable และ Cloneable ให้ทำการ override เมธอด equals() ของคลาส Object เมธอด compareTo() เพื่อทำการเปรียบเทียบ Octagon จากพื้นที่ของ Octagon โดยที่รูปแปดเหลี่ยมจะเท่ากันก็ต่อเมื่อมีพื้นที่เท่ากัน โดย พื้นที่ของ Octagon คือ $area = (2 + 4/\sqrt{2}) * side * side$

วาดคลาสไดอะแกรมและเขียนโปรแกรมทดสอบคลาสดังกล่าวโดยสร้าง Object จากคลาส Octagon ที่มี side=5 ให้แสดงพื้นที่และเส้นรอบวง หลังจากนั้นให้สร้าง object ใหม่โดยใช้เมธอด clone และเปรียบเทียบว่า object ทั้งสองดังกล่าวเท่ากันหรือไม่

3.4 สร้างคลาสที่ชื่อว่า ComparableCircle ที่สืบทอดมาจากคลาส Circle และ implements อินเตอร์เฟซ Comparable วาด UML class diagram และ implements เมธอด compareTo() เพื่อทำการเปรียบเทียบวงกลมจากพื้นที่ของวงกลม เขียน Test class ที่ค้นหาวงกลมที่ใหญ่ที่สุดจาก วงกลมที่เป็น 2object ของ ComparableCircle

4. Suppose that we have a set of objects with some common behaviors: they could move up, down, left or right. The exact behaviors (such as how to move and how far to move) depend on the objects themselves. One common way to model these common behaviors is to define an *interface* called Movable, with abstract methods moveUp(), moveDown(), moveLeft() and moveRight(). The classes that implement the Movable interface will provide actual implementation to these abstract methods.

Let's write two concrete classes - MovablePoint and MovableCircle - that implement the Movable interface.



5. [Algorithms] รถไฟฟ้าใต้ดิน (subway)

พ.ศ.2560 รัฐบาลได้ดำเนินโครงการก่อสร้างรถไฟฟ้าใต้ดินซึ่งเป็นโครงการเมกะโปรเจกต์จนเสร็จสิ้น ทำให้กรุงเทพฯ กลายเป็นเมืองที่มีเครือข่ายรถไฟฟ้าใต้ดินที่ใหญ่ที่สุดแห่งหนึ่งของโลก ประกอบด้วยเส้นทางรถไฟฟ้าใต้ดินหลายร้อยสาย และสถานีอีกนับล้านสถานี คุณต้องการเดินทางโดยรถไฟฟ้าใต้ดินจากสถานีหนึ่งไปยังอีกสถานีหนึ่ง โดยในระหว่างทาง สามารถทำการเปลี่ยนสายรถไฟฟ้าได้โดยการไปลงที่บางสถานีแล้วขึ้นรถไฟฟ้าสายอื่นที่ผ่านสถานีนั่นต่อ แต่การเปลี่ยนสายรถไฟฟ้าแต่ละครั้งก็ทำให้เสียเวลาเป็นอย่างมาก คุณจึงต้องการเดินทางโดยเปลี่ยนสายรถไฟฟ้าให้น้อยครั้งที่สุดเท่าที่จะทำได้

จงเขียนโปรแกรมเพื่อตอบคำถามทั้งหมด Q คำถามว่า การเดินทางจากสถานี A_i ไปยังสถานี B_i จะต้องทำการเปลี่ยนสายรถไฟฟ้าอย่างน้อยกี่ครั้ง

ข้อมูลนำเข้า

บรรทัดแรกระบุจำนวนเต็ม N และ M ($2 \leq N \leq 1,000,000$; $1 \leq M \leq 500$) แทนจำนวนสถานีทั้งหมดและจำนวนสายของรถไฟฟ้าใต้ดิน

อีก M บรรทัดต่อมา ในบรรทัดที่ $i+1$ ($1 \leq i \leq M$) ระบุจำนวนเต็มตัวแรกคือ S_i ($2 \leq S_i \leq 2,000$) แทนจำนวนสถานีที่รถไฟฟ้าสายที่ i ผ่าน และจำนวนเต็มอีก S_i จำนวนถัดมา ระบุหมายเลขของสถานีที่รถไฟฟ้าสายดังกล่าวผ่าน เรียงตามลำดับจากปลายทางข้างหนึ่งไปจนถึงปลายทางอีกข้างหนึ่ง

บรรทัดถัดมา ระบุจำนวนเต็ม Q ($2 \leq Q \leq 1,000,000$) แทนจำนวนคำถามทั้งหมด

อีก Q บรรทัดถัดมา ในบรรทัดที่ $i+M+2$ ($1 \leq i \leq Q$) ระบุจำนวนเต็ม A_i และ B_i ($1 \leq A_i, B_i \leq N$) แสดงถึงคำถามที่ i สถานีแต่ละสถานีจะมีรถไฟใต้ดินผ่านไม่เกิน 20 สาย โดยที่บางสถานีอาจไม่มีรถไฟใต้ดินผ่านเลยแม้แต่สายเดียวก็ได้ นอกจากนี้เส้นทางของรถไฟใต้ดินแต่ละสายอาจผ่านบางสถานีนี้นอกเหนือจากหนึ่งครั้งก็ได้

ข้อมูลส่งออก

มีทั้งหมด Q บรรทัด ในบรรทัดที่ i ($1 \leq i \leq Q$) ให้พิมพ์จำนวนครั้งของการเปลี่ยนสายรถไฟที่น้อยที่สุดที่ต้องใช้ในการเดินทางจากสถานี A_i ไปยังสถานี B_i แต่ถ้าไม่สามารถเดินทางโดยรถไฟใต้ดินจากสถานี A_i ไปยังสถานี B_i ได้ ให้พิมพ์คำว่า impossible

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
6 2 3 1 2 3 3 2 4 5 3 1 3 1 4 2 6	0 1 impossible
15 5 6 1 2 3 4 2 5 2 6 7 4 1 6 8 9 4 10 11 12 13 3 14 11 15 6 9 2 10 13 10 5 3 7 6 14 15 12	1 0 impossible 2 impossible 1

บทที่ 11 การสร้างส่วนต่อประสานผู้ใช้

วัตถุประสงค์

11.1 สามารถอธิบายส่วนประกอบภายในแพ็คเกจ *Swing* และ *AWT*

11.2 สามารถเขียนโปรแกรมเพื่อสร้างส่วนต่อประสานผู้ใช้

11.1 ความนำ

ระบบปฏิบัติการส่วนใหญ่จะมีส่วนติดต่อกับผู้ใช้เป็นแบบกราฟิก (Graphical User Interface เรียกว่า GUI) ทั้งนี้เนื่องจากใช้งานง่ายกว่า ภาษาจาวาจะสนับสนุนการพัฒนาโปรแกรม GUI ที่สามารถใช้งานได้หลายแพลตฟอร์มโดยการใช้คำสั่งเดียวกัน Java Foundation Class (JFC) ประกอบด้วยแพ็คเกจต่าง ๆ ดังนี้

- Abstract Window Toolkit (AWT)
- Swing
- Java 2D
- Accessibility
- Drag and Drop

11.2 ความรู้เบื้องต้นเกี่ยวกับ Swing และ AWT

คลาสที่ทำหน้าที่ในการจัดการเกี่ยวกับหน้าจอหรือส่วนต่อประสานผู้ใช้ (User interface) จะอยู่ในแพ็คเกจของจาวาที่ชื่อ Abstract Windows Toolkit (AWT) ทุก ๆ platform ที่รันภาษาจาวา components ของ package AWT จะถูกเรียกอัตโนมัติโดยการเลือกคอมโพเนนต์ที่ตรงกับระบบปฏิบัติการแต่ละประเภท (platform-specific components through their respective agents) ที่เรียกว่า peers

แพ็คเกจ AWT จะช่วยในการสร้างโปรแกรม GUI ประเภท Look and Feel ที่ขึ้นอยู่กับแพลตฟอร์มที่ใช้งาน ในแพ็คเกจ java.awt มีคลาสและอินเตอร์เฟสที่สำคัญดังนี้

- ▶ Component
- ▶ Container
- ▶ LayoutManager
- ▶ Graphics
- ▶ Color
- ▶ Font
- ▶ AWTEvent

การเขียนโปรแกรมเกี่ยวกับหน้าจอหรือส่วนต่อประสานผู้ใช้ (GUI) นั้นจะเป็นการสร้างออบเจ็กต์ต่าง ๆ ที่เป็นออบเจ็กต์ของคลาสที่เป็นส่วนประกอบกราฟิก (Graphic component) โดยคลาสที่เป็นส่วนประกอบกราฟิกจะสืบทอดมาจากคลาสที่ชื่อว่า Component คลาสที่เป็น Subclass ของคลาส Component จะแบ่งเป็นสองกลุ่มคือ

1. Container class เป็นคลาสที่ใช้ในการใส่ส่วนประกอบกราฟิกต่าง ๆ โปรแกรม GUI จะต้องมีการสร้างออบเจ็คของคลาสประเภท Container อย่างน้อย 1 ออบเจ็คขึ้นมาก่อน เพื่อใช้ในการใส่ออบเจ็คของคลาสที่เป็นส่วนประกอบกราฟิกอื่น ๆ คลาสประเภท Container ที่อยู่ในแพ็คเกจ AWT มีดังนี้ Frame Panel Dialog Applet

คลาสประเภท Container เป็นคลาสที่สืบทอดมาจากคลาสที่ชื่อ Component

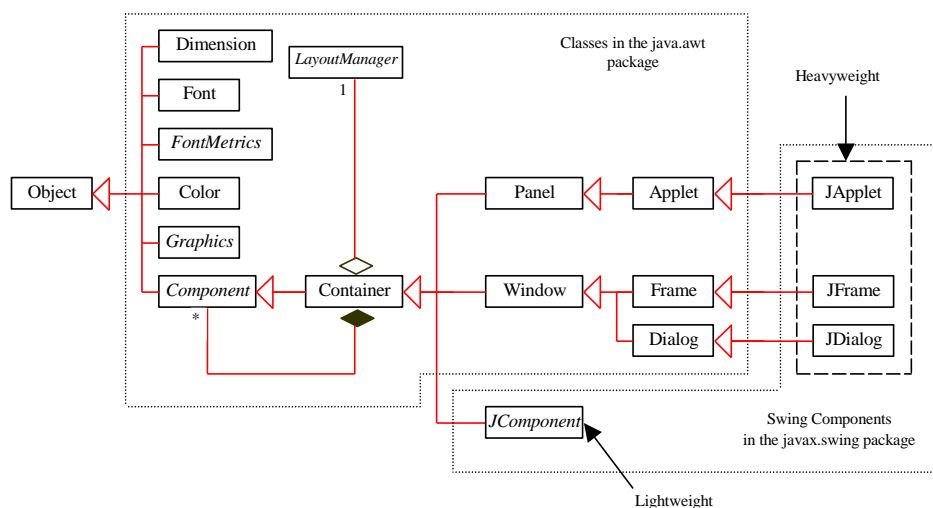
2. Component class เป็นคลาสที่เป็นส่วนประกอบกราฟิกอื่น ๆ เช่น Button Choice และ List เป็นต้น Class Component จะมี method ที่ใช้กับ object ต่าง ๆ ที่สร้างจาก class นี้ได้แก่ paint () และ repaint () ส่วน Class Container เป็นส่วนหนึ่งของพื้นที่ที่ใช้ในการแสดงผลทาง graphics เมธอดที่ใช้กันบ่อย ได้แก่ add () สำหรับใช้ในการเพิ่ม graphical object อีก method หนึ่งได้แก่ setLayout () สำหรับใช้ในการกำหนดโครงสร้างและช่วยในการกำหนดตำแหน่งและขนาดของ components ในส่วนของแพ็คเกจ Swing จะมีคลาส JComponent เป็น super-class ของ Swing component

11.2.1 การใช้งาน Swing และ AWT

คลาสในแพ็คเกจ AWT เหมาะกับการพัฒนาหน้าจอที่มีการออกแบบง่าย ๆ แต่ไม่สามารถพัฒนาหน้าจอที่มีกราฟฟิกยาก ๆ ได้ดีเท่าที่ควร นอกจากนี้ AWT มีแนวโน้มที่จะเกิดข้อผิดพลาดที่เกิดจาก platform-specific เพราะว่าเทคนิค peer-based จะทำงานขึ้นอยู่กับ platform ที่เจาะจง ดังนั้น AWT user-interface components จึงถูกแทนที่ด้วย component ที่มีความแข็งแกร่งมากขึ้น มี library ที่หลากหลายและมีความยืดหยุ่นที่เรียกว่า Swing components

Swing components จะถูกวาดโดยตรงบน canvases ยกเว้น components ที่เป็น subclasses ของ java.awt.Window หรือ java.awt.Panel ที่ต้องวาดโดยการใช้ GUI ที่เจาะจงกับ platform

Swing components จะผูกติดกับ platform และใช้ทรัพยากรของ GUI ที่จำเพาะน้อยกว่า AWT เราเรียกการไม่จำเพาะเจาะจงกับ native GUI ว่า lightweight components ส่วน AWT components จะเรียกว่า heavyweight components



การสร้าง GUI component เช่น button, label, menu bar, text field จัดเป็นส่วนหนึ่งของ Swing ส่วนของ GUI components ซึ่งอยู่ใน package javax.swing นี้จะสามารถประมวลผลได้เฉพาะ jdk1.2 หรือสูงกว่าเท่านั้น Source code ของ GUI component เมื่อสมัยที่ยังไม่มี swing จะมาจาก Abstract Windowing Toolkit package (java.awt) ตัว Swing เป็น components ที่เขียนขึ้นโดยใช้ภาษา Java และมีความง่ายในการใช้งานมากกว่าตัวต้นฉบับที่มาจาก package java.awt

11.3 ส่วนประกอบที่สำคัญของแพ็คเกจ Swing

11.3.1 คลาส JFrame เป็นหน้าต่างที่ไม่อยู่ในหน้าต่างอื่น เป็นหน้าจอที่จะใช้สำหรับบรรจุ User interface components ตัวอื่น เราสามารถเรียกใช้ผ่านคลาส JFrame คลาส JFrame จะสืบทอดมาจากคลาส Frame โดยมี constructor ที่สำคัญดังนี้

```
public JFrame()
public JFrame(String title)
```

ออบเจกต์ของคลาส JFrame แตกต่างกับ Frame ตรงที่มีหน้าต่าง pane อยู่ 4 หน้าต่างดังนี้ Root pane Layer pane Glass pane Content pane เราไม่สามารถที่จะใส่ส่วนประกอบกราฟิกลงใน Frame ได้โดยตรง แต่จะต้องใส่ลงหน้าต่างที่เป็น content pane แทน เราสามารถที่จะเรียกออบเจกต์ของคลาสประเภท Container ดังกล่าวมาได้โดยใช้เมธอดที่ชื่อ getContentPane () และสามารถที่จะใส่ส่วนประกอบกราฟิกลงใน object ดังกล่าวได้โดยใช้เมธอด add() เช่น

```
Container c = fr.getContentPane();
c.setLayout (new BorderLayout());
c.add(bn1, BorderLayout.SOUTH);
```

ตัวอย่างที่ 11.1 ตัวอย่างการสร้างหน้าจอจากคลาส JFrame

```
import javax.swing.*;
public class Ex11_1 {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Test Frame");
        frame.setSize(400, 300);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



ตัวอย่างที่ 11.2 ตัวอย่างการสร้างหน้าต่างจากคลาส JFrame

```
import javax.swing.*;
public class EX11_2 {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My Frame With Components");
        JButton jbtOK = new JButton("OK");
        frame.getContentPane().add( new JButton("OK"));
        frame.add(jbtOK);
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setVisible(true);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



11.3.2 คลาส Layout Managers โดยทั่วไป User Interface Components จะวางใน containers โดยที่ container จะมีรูปแบบการจัดวาง component ของตัวเองโดยมีส่วนที่เรียกว่า layout manager ใช้สำหรับการจัดเรียง UI components ภายใน container ส่วนของ Layout managers จะกำหนดให้กับ containers โดยการใช้เมธอด setLayout (LayoutManager) ที่อยู่ใน container ประเภทของ layout Managers มีดังนี้

FlowLayout เป็นค่า default สำหรับ java.applet.Applet, java.awt.Panel และ javax.swing.JPanel เป็นการ จัดเรียง component ต่างๆ เป็นลำดับก่อน หลังขึ้นกับลำดับการ-add object โดยจะจัดเรียงจากซ้าย ไปยังขวา

GridLayout เป็นการ จัดเรียง component ในแนวแถว (row) และสดมภ์ (column)

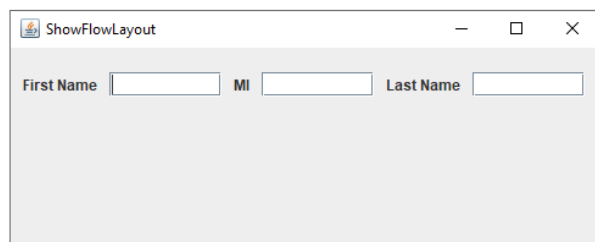
BorderLayout เป็นค่า default สำหรับ JFrame และ JApplet จะจัดเรียง component ในพื้นที่ 5 ส่วนคือ North, South, East, West และ Center

11.3.2.1 คลาส FlowLayout เป็นค่า default สำหรับ java.applet.Applet, java.awt.Panel และ javax.swing.JPanel เป็นการ จัดเรียง component ต่างๆ เป็นลำดับก่อน หลังขึ้นกับลำดับการ-add object โดยจะ จัดเรียงจากซ้าย ไปยังขวา ตัวอย่างโปรแกรมที่เพิ่ม labels 3 ขึ้น และ text fields ลงไปที่ content pane ของ frame โดยใช้ FlowLayout manager มีดังนี้

ตัวอย่างที่ 11.3 ตัวอย่างโปรแกรมที่เพิ่ม labels 3 ขึ้น และ text fields ลงไปที่ content pane ของ frame โดยใช้ FlowLayout manager

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;
public class Ex11_3 extends JFrame {
    public Ex11_3() {
        setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(8));
        add(new JLabel("Last Name"));
        add(new JTextField(8));
    }
    public static void main(String[] args) {
        Ex11_3 frame = new Ex11_3();
        frame.setTitle("ShowFlowLayout");
        frame.setSize(500, 200);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

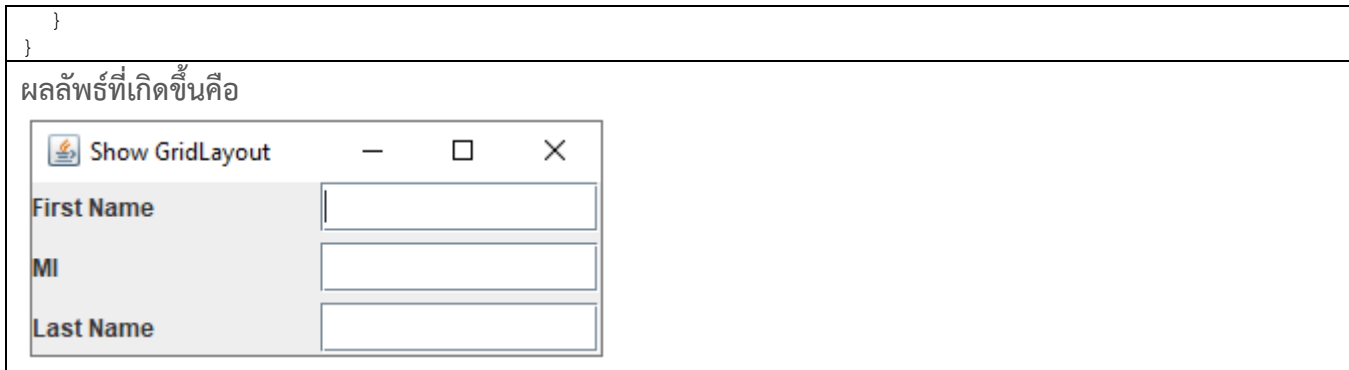
ผลลัพธ์ที่เกิดขึ้นคือ



11.3.2.2 คลาส GridLayout เป็นการจัดเรียง component ในแนวแถว (row) และสดมภ์ (column)

ตัวอย่างที่ 11.4 คลาส GridLayout เป็นการจัดเรียง component ในแนวแถว (row) และสดมภ์ (column)

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.GridLayout;
public class Ex11_4 extends JFrame {
    public Ex11_4() {
        setLayout(new GridLayout(3, 2, 5, 5));
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(8));
        add(new JLabel("Last Name"));
        add(new JTextField(8));
    }
    public static void main(String[] args) {
        Ex11_4 frame = new Ex11_4();
        frame.setTitle("Show GridLayout");
        frame.setSize(300, 125);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



11.3.2.3 คลาส BorderLayout เป็นค่า default สำหรับ JFrame และ JApplet จะจัดเรียง

Component ในพื้นที่ 5 ส่วนคือ North, South, East, West และ Center โดย BorderLayout manager จะแบ่ง container ออกเป็น 5 พื้นที่ คือ East South West North และ Center

โดยที่ Components จะถูกบรรจุลง BorderLayout โดยเมธอด add ในรูปแบบต่อไปนี้ ()

```
add(Component, constraint)
```

เมื่อconstraint คือ

BorderLayout.EAST

BorderLayout.SOUTH

BorderLayout.WEST

BorderLayout.NORTH

BorderLayout.CENTER

ตัวอย่างที่ 11.5 คลาส BorderLayout

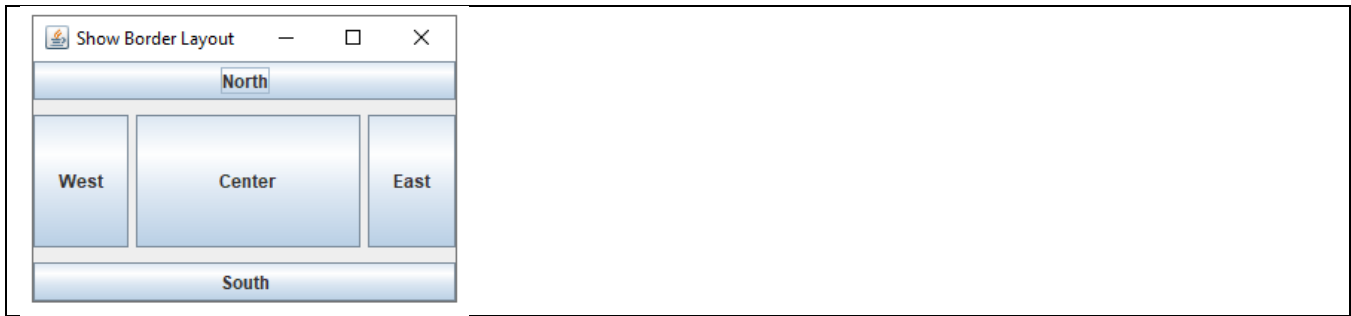
```

import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.BorderLayout;

public class Ex11_5 extends JFrame {
    public Ex11_5() {
        setLayout(new BorderLayout(5, 10));
        add(new JButton("East"), BorderLayout.EAST);
        add(new JButton("South"), BorderLayout.SOUTH);
        add(new JButton("West"), BorderLayout.WEST);
        add(new JButton("North"), BorderLayout.NORTH);
        add(new JButton("Center"), BorderLayout.CENTER);
    }
    public static void main(String[] args) {
        Ex11_5 frame = new Ex11_5();
        frame.setTitle("Show Border Layout");
        frame.setSize(300, 200);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



11.3.2.4 คลาส Color เราสามารถกำหนดสีให้กับส่วนประกอบต่าง ๆ ของหน้าจอได้โดยใช้คลาส `java.awt.Color` โดยที่สีจะประกอบไปด้วยการผสมกันของสามสี คือสีแดง (red) สีเขียว (green) และสีน้ำเงิน (blue) โดยที่แต่ละสีจะถูกกำหนดโดยตัวเลข (byte value) ที่อธิบายความเข้ม (intensity) ของสีโดยการเรียงจาก 0 (darkest shade) ถึง 255 (lightest shade) เรียกว่าโมเดล RGB

```
Color c = new Color(r, g, b);
```

ตัวอย่าง

```
Color c = new Color(228, 100, 255);
```

สีมาตรฐาน 13 สี (black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, yellow) ได้ถูกกำหนดในคลาส `java.awt.Color` โดยการประกาศเป็นค่าคงที่ คือ BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, and YELLOW.

การกำหนดสีให้กับ Component เราสามารถใช้เมธอดต่อไปนี้ในการกำหนดสีพื้นหลัง background และพื้นหน้า foreground ให้กับ component

```
setBackground(Color c);
```

```
setForeground(Color c);
```

ตัวอย่าง

```
jbt.setBackground(Color.yellow);
```

```
jbt.setForeground(Color.red);
```

สีที่ใช้ในภาษา Java จะมาจาก class `Color` ซึ่งมี method รวมทั้งค่าคงที่ต่างๆ เกี่ยวกับสี ให้ผู้เขียนโปรแกรมสามารถเลือกใช้ได้สีที่กำหนดไว้แล้วจาก `Color` สามารถสรุปได้ดังนี้

Colour Constant	Colour	RGB value
<code>public final static Color orange</code>	Orange	255, 200, 0
<code>public final static Color pink</code>	Pink	255, 175, 175
<code>public final static Color cyan</code>	Cyan	0, 255, 255
<code>public final static Color magenta</code>	Magenta	255, 0, 255
<code>public final static Color yellow</code>	Yellow	255, 255, 0
<code>public final static Color black</code>	Black	0, 0, 0
<code>public final static Color white</code>	White	255, 255, 255
<code>public final static Color gray</code>	Gray	128, 128, 128
<code>public final static Color lightGray</code>	Light gray	192, 192, 192

public final static Color darkGray	Dark gray	64, 64, 64
public final static Color red	Red	255, 0, 0
public final static Color green	Green	0, 255, 0
public final static Color blue	Blue	0, 0, 255

การใช้งานเมธอดที่ใช้งานในคลาส **Color** สามารถทำได้โดยการใช้เมธอดต่อไปนี้

Method	Description
public Color(int r, int g, int b)	เป็นการสร้างสีจากการผสมระหว่างสี แดง เขียว และ น้ำเงิน ค่าตัวเลขของแต่ละสีจะเป็นจำนวนเต็ม มีค่า 0 ถึง 255
public Color(float r, float g, float b)	เป็นการสร้างสีเช่นกัน แต่ค่าตัวเลขจะเป็นจำนวนจริง และมีค่า 0.0 ถึง 1.0
public int getRed()	Color class จะ return ค่าของสีแดง ซึ่งมีค่าตั้งแต่ 0 ถึง 255
public int getGreen()	Color class Return ค่าของสีเขียว ซึ่งมีค่าอยู่ระหว่าง 0 ถึง 255
public int getBlue()	Color class Return ค่าของสีน้ำเงิน ซึ่งมีค่าอยู่ระหว่าง 0 ถึง 255
public Color getColor()	Graphics class Return Color object ที่แสดงสีปัจจุบันที่ใช้งานอยู่
public void setColor(Color c)	Graphics class เป็นการกำหนดสีเพื่อที่จะใช้วาด object graphics

ตัวอย่างที่ 11.6 การใช้งานเมธอดที่ใช้งานในคลาส **Color**

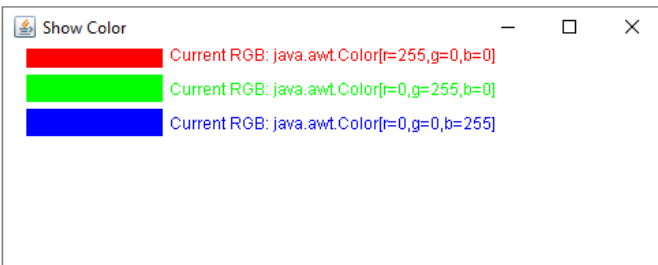
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Ex11_6 extends JFrame {
    public Ex11_6() {
        super ("Using colours");
        setSize(400, 130);
        show();
    }
    public void paint( Graphics g ) {
        g.setColor( new Color(255, 0, 0) );
        g.fillRect( 25, 25, 100, 20 );
        g.drawString("Current RGB: "+g.getColor(), 130, 40);
        g.setColor( new Color(0.0f, 1.0f, 0.0f) );
        g.fillRect( 25, 50, 100, 20 );
        g.drawString("Current RGB: "+g.getColor(), 130, 65);
        g.setColor( Color.blue );
        g.fillRect( 25, 75, 100, 20 );
        g.drawString("Current RGB: "+g.getColor(), 130, 90);
    }
    public static void main(String[] args) {
        Ex11_6 frame = new Ex11_6();
    }
}
```

```

frame.setTitle("Show Color");
frame.setSize(500, 200);
frame.setLocationRelativeTo(null); // Center the frame
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



11.3.2.5 คลาส Font มีรูปแบบในการใช้งานคือ

Font myFont = new Font(name, style, size);

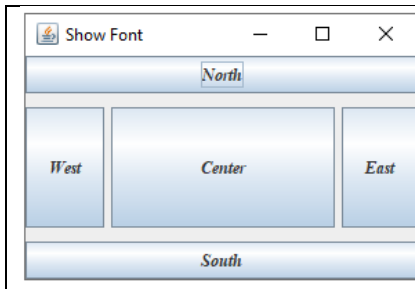
ตัวอย่างที่ 11.7 คลาส Font

```

import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.BorderLayout;
import java.awt.Font;
public class Ex11_7 extends JFrame {
    public Ex11_7() {
        //Font myFont = new Font("SansSerif ", Font.BOLD, 16);
        Font myFont = new Font("Serif", Font.BOLD+Font.ITALIC, 12);
        JButton jbtEast = new JButton("East");
        JButton jbtSouth = new JButton("South");
        JButton jbtWest = new JButton("West");
        JButton jbtNorth = new JButton("North");
        JButton jbtCenter = new JButton("Center");
        jbtEast.setFont(myFont);
        jbtSouth.setFont(myFont);
        jbtWest.setFont(myFont);
        jbtNorth.setFont(myFont);
        jbtEast.setFont(myFont);
        jbtCenter.setFont(myFont);
        setLayout(new BorderLayout(5, 10));
        add(jbtEast, BorderLayout.EAST);
        add(jbtSouth, BorderLayout.SOUTH);
        add(jbtWest, BorderLayout.WEST);
        add(jbtNorth, BorderLayout.NORTH);
        add(jbtCenter, BorderLayout.CENTER);
    }
    public static void main(String[] args) {
        Ex11_7 frame = new Ex11_7();
        frame.setTitle("Show Font");
        frame.setSize(300, 200);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



การค้นหาคือ **Font** ที่สามารถนำมาใช้ มีรูปแบบการประกาศ ดังนี้

ตัวอย่างที่ 11.8 การค้นหาคือ **Font**

```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.GraphicsEnvironment;
public class Ex11_8 extends JFrame {
    public Ex11_8() {
        GraphicsEnvironment e = GraphicsEnvironment.getLocalGraphicsEnvironment();
        String[] fontnames = e.getAvailableFontFamilyNames();
        for (int i = 0; i < fontnames.length; i++)
            System.out.println(fontnames[i]);
    }
    public static void main(String[] args) {
        Ex11_8 frame = new Ex11_8();
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

Agency FB
 Algerian
 Angsana New
 AngsanaUPC
 Arial
 Arial Black
 Arial Narrow
 Arial Rounded MT Bold
 Bahnschrift

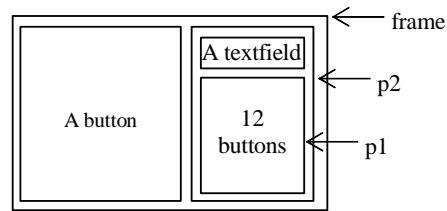
11.3.2.6 คลาส Panels

Panels ทำหน้าที่เป็น sub-containers สำหรับรวมกลุ่มของ user interface components ใน container ตัวอื่น เราสามารถรวม user interface components ใน panels ก่อนที่จะเอาไปใส่ frame หรือสามารถเพิ่มหลาย ๆ panel ใน panel ก็ได้ ในการ add ตัวคอมโพเนนต์ลงใน JFrame เราสามารถเพิ่มใน content pane ของ JFrame การเพิ่มคอมโพเนนต์ลงใน panel เราสามารถเพิ่มได้โดยตรงลงใน panel โดยใช้เมธอด add การใช้งาน Panel สามารถใช้ new JPanel() สำหรับสร้าง panel โดยมี Layout manager คือ FlowLayout หรือจะกำหนด Layout manager ตามความต้องการโดยใช้ new JPanel(LayoutManager) หลังจากนั้นสามารถใช้ add(Component) สำหรับเพิ่ม component ลงไปที่ panel โดยใช้คำสั่ง

```
JPanel p = new JPanel();
p.add(new JButton("OK"));
```

ตัวอย่าง ต้องการสร้าง Panel ที่มีการจัดหน้าจอดังนี้ สามารถเขียนโปรแกรมได้ดังตัวอย่างที่ 11.9

ตัวอย่างที่ 11.9 ต้องการสร้าง Panel ที่มีการจัดหน้าจอดังนี้

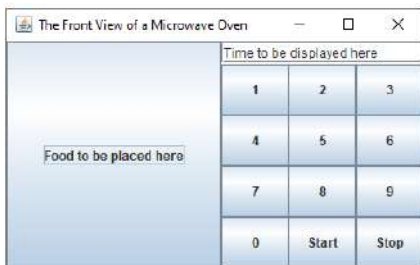


```

import java.awt.*;
import javax.swing.*;

public class Ex11_9 extends JFrame {
    public Ex11_9() {
        JPanel p1 = new JPanel();
        p1.setLayout(new GridLayout(4, 3));
        for (int i = 1; i <= 9; i++) {
            p1.add(new JButton(" " + i));
        }
        p1.add(new JButton(" " + 0));
        p1.add(new JButton("Start"));
        p1.add(new JButton("Stop"));
        JPanel p2 = new JPanel(new BorderLayout());
        p2.add(new JTextField("Time to be displayed here"),
            BorderLayout.NORTH);
        p2.add(p1, BorderLayout.CENTER);
        add(p2, BorderLayout.EAST);
        add(new JButton("Food to be placed here"),
            BorderLayout.CENTER);
    }
    public static void main(String[] args) {
        Ex11_9 frame = new Ex11_9();
        frame.setTitle("The Front View of a Microwave Oven");
        frame.setSize(400, 250);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
  
```

ผลลัพธ์ที่เกิดขึ้นคือ



11.3.2.7 คลาส Borders เราสามารถกำหนด Border ให้กับวัตถุต่าง ๆ ของคลาส JComponent ในการกำหนด titled border สามารถกำหนดได้โดย

```
new TitledBorder(String title)
```

การกำหนด line border กำหนดได้โดย

```
new LineBorder(Color color, int width)
```

โดยที่ width กำหนดความหนาของเส้น

ตัวอย่าง

```
JPanel panel = new JPanel();
```

```
panel.setBorder(new TitledBorder("My Panel"));
```

11.3.2.8 คลาส Image Icons

เราสามารถกำหนด icon ให้กับหน้าจอได้โดยการใช้คลาส javax.swing.ImageIcon โดย icon คือ รูปภาพที่มีขนาดจำกัดโดยทั่วไปจะมีขนาดเล็กใช้ตกแต่ง component โดยรูปแบบของการกำหนด Image icon ทำได้โดย new ImageIcon(filename) เช่น

```
ImageIcon icon = new ImageIcon("image/us.gif");
```

11.3.2.9 คลาส JLabel

Label เป็นส่วนที่ใช้แสดงข้อความแบบบรรทัดเดียว (single line) บน GUI ซึ่งถูก define ด้วย class JLabel ซึ่งเป็น sub-class ของ JComponent คอนสตรักเตอร์ที่ทำหน้าที่สำหรับกำหนดค่าเริ่มต้นให้กับ label มีดังนี้

```
JLabel()
```

```
JLabel(String text, int horizontalAlignment)
```

```
JLabel(String text)
```

```
JLabel(Icon icon)
```

```
JLabel(Icon icon, int horizontalAlignment)
```

```
JLabel(String text, Icon icon, int horizontalAlignment)
```

11.3.2.10 คลาส JTextField และ JPasswordField

JTextField และ JPasswordField เป็น component ที่ใช้ในการรับข้อความจากผู้ใช้ โดยที่ JTextField จะแสดงอักขระที่ผู้ใช้พิมพ์เข้าไป แต่ JPasswordField จะแสดงอักขระที่พิมพ์เข้าไปเป็น '*' เมธอดที่สำคัญของ JTextField มีรายละเอียดดังต่อไปนี้

getText() จะคืนข้อความจาก Text field.

setText(String text) ใส่ข้อความไปยัง Text field.

setEditable(boolean editable) กำหนดให้ Text field แก้ไขได้หรือไม่

setColumns(int) กำหนดจำนวนของคอลัมน์ใน Text field โดยความยาวของ text field สามารถเปลี่ยนแปลง

ได้

11.3.2.11 คลาส JButton

Button เป็นคลาสที่ใช้ในการสร้างวัตถุที่แสดงเป็นปุ่ม โดยจะมีข้อความ (label) ปรากฏอยู่บนปุ่ม เมธอดทั่ว ๆ ไปที่ใช้กับ button จะอยู่ในคลาส javax.swing.AbstractButton JButton จะสืบทอดข้อมูลและเมธอดต่าง ๆ มาจากคลาส AbstractButton โดยมีคอนสตรัคเตอร์ต่าง ๆ สำหรับสร้างปุ่ม เมธอดต่อไปนี้คือ Constructors สำหรับสร้าง button

JButton()

JButton(String text)

JButton(String text, Icon icon)

JButton(Icon icon)

นอกจากนี้ JButton ยังมีคุณสมบัติต่าง ๆ ที่สามารถกำหนดให้กับ button ได้คือ

- ▶ text
- ▶ icon
- ▶ mnemonic
- ▶ horizontalAlignment
- ▶ verticalAlignment
- ▶ horizontalTextPosition
- ▶ verticalTextPosition
- ▶ iconTextGap

11.3.2.12 คลาส JCheckBox and JRadioButton

JCheckBox จะสืบทอดข้อมูลต่าง ๆ จากเช่นคุณสมบัติ text, icon, mnemonic, verticalAlignment, horizontalAlignment, horizontalTextPosition, verticalTextPosition, และ selected จากAbstractButton โดยมีคอนสตรัคเตอร์รูปแบบต่าง ๆ สำหรับสร้าง check boxes

JCheckBox()

JCheckBox(Action a)

JCheckBox(Icon icon)

JCheckBox(Icon icon, boolean selected)

JCheckBox(String text)

JCheckBox(String text, boolean selected)

JCheckBox(String text, Icon icon)

JCheckBox(String text, Icon icon, boolean selected)

Radio buttons เป็นอีกรูปแบบหนึ่งของ check boxes เพียงแต่ปุ่มเรดิโอจะรวมเป็นกลุ่มและจะอนุญาตให้ปุ่มใดปุ่มหนึ่งถูกเรียก ตัวอย่าง

```
ButtonGroup btg = new ButtonGroup();
btg.add(jrb1);
btg.add(jrb2);
```

11.3.2.13 คลาส JTextArea

กรณีที่ต้องการอนุญาตให้ user สามารถป้อนข้อมูลได้หลายบรรทัดจะใช้ JTextArea เมธอดต่อไปนี้คือ

Constructors สำหรับสร้าง JTextArea

JTextArea(int rows, int columns) สร้าง text area โดยระบุจำนวนแถวและคอลัมน์

JTextArea(String s, int rows, int columns) สร้าง Text area โดยระบุจำนวนแถวและคอลัมน์ และกำหนดข้อความเริ่มต้น

11.3.2.14 คลาส JComboBox

combo box เป็นรายการข้อมูลที่ใช้ผู้ใช้สามารถเลือกได้ครั้งละ 1 รายการ การเพิ่มรายการเข้าไป JComboBox jcbo จะทำได้โดยการใช้เมธอด addItem เช่น jcbo.addItem(Object item)

การดึงรายการจาก JComboBox jcbo จะทำได้โดย jcbo.getItem()

เมื่อรายการข้อมูลถูกเลือกหรือไม่เลือก เมธอด itemStateChanged() สำหรับ ItemEvent จะถูกเรียก

```
public void itemStateChanged(ItemEvent e) {
    // Make sure the source is a combo box
    if (e.getSource() instanceof JComboBox)
        String s = (String)e.getItem();
}
```

11.3.2.14 คลาส JList

list มีการทำงานคล้าย combo box แต่จะอนุญาตให้ผู้ใช้เลือกข้อมูล 1 รายการหรือมากกว่า 1 รายการ เมธอดต่อไปนี้คือ constructors สำหรับสร้าง JList:

JList() สร้าง list ว่าง

JList(Object[] stringItems) สร้าง list พร้อมกำหนดค่าเริ่มต้น

นอกจากนี้ JList ยังมีคุณสมบัติต่าง ๆ ที่สามารถกำหนดให้กับ list ได้คือ

selectedIndex

selectedIndices

selectedValue

selectedValues

selectionMode

visibleRowCount

11.3.2.15 คลาส JScrollBar

Scroll bar เป็นคอนโทรลที่อนุญาตให้ผู้ใช้เลือกค่าข้อมูลจากช่วงที่กำหนดโดยลักษณะของ scrollbar จะปรากฏใน รูปแบบ คือ 2 horizontal และ vertical

11.4 การสร้างส่วนต่อประสานผู้ใช้

ตัวอย่างการสร้าง Multiple Windows

ขั้นตอนที่ 1 สร้าง subclass ของ JFrame (เรียกว่า SubFrame)

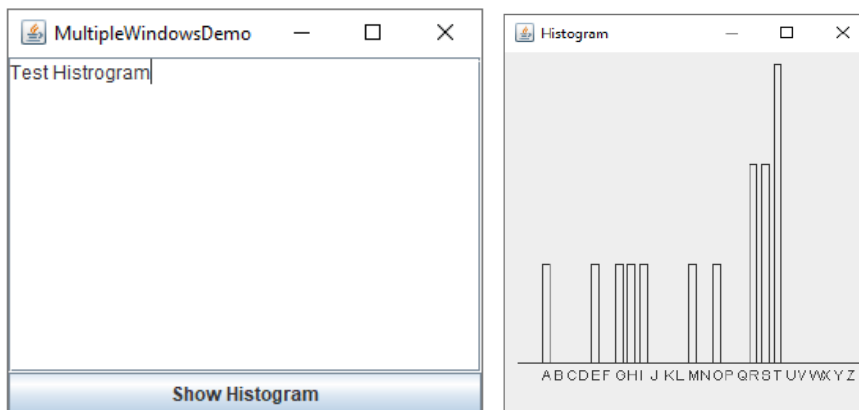
ขั้นตอนที่ 2 สร้างวัตถุของ SubFrame ในคลาสที่ต้องการเปิดหน้าจอใหม่

ขั้นตอนที่ 3 สร้างปุ่มสำหรับเรียกวัตถุของ SubFrame มาทำงาน

ขั้นตอนที่ 4 โอเวอร์ไรด์เมธอด actionPerformed()

```
SubFrame subFrame = new SubFrame("SubFrame Title");
add(new JButton("Activate SubFrame"));
public actionPerformed(ActionEvent e) {
    String actionCommand = e.getActionCommand();
    if (e.target instanceof JButton) {
        if ("Activate SubFrame".equals(actionCommand)) {
            subFrame.setVisible(true);
        }
    }
}
```

ในกรณีนี้ที่ผู้พัฒนาโปรแกรมต้องการสร้างหน้าต่างหลักซึ่งให้ผู้ใช้กรอกข้อความใน TextField และมีปุ่มที่มีข้อความ Show Histogram เมื่อผู้ใช้กดปุ่มดังกล่าว หน้าจอใหม่จะปรากฏและโชว์ฮิสโทแกรมที่นับจำนวนตัวอักษรที่ปรากฏใน TextArea ดังตัวอย่างต่อไปนี้



ตัวอย่างที่ 11.10 การสร้างส่วนต่อประสานผู้ใช้

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class MultipleWindowsDemo extends JFrame {
    private JTextArea jta;
    private JButton jbtShowHistogram = new JButton("Show Histogram");
    private Histogram histogram = new Histogram();
    private JFrame histogramFrame = new JFrame();
    public MultipleWindowsDemo() {
        JScrollPane scrollPane = new JScrollPane(jta = new JTextArea());
        scrollPane.setPreferredSize(new Dimension(300, 200));
        jta.setWrapStyleWord(true);
        jta.setLineWrap(true);
        add(scrollPane, BorderLayout.CENTER);
        add(jbtShowHistogram, BorderLayout.SOUTH);
        jbtShowHistogram.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```

        int[] count = countLetters();
        histogram.showHistogram(count);
        histogramFrame.setVisible(true);
    }
});
histogramFrame.add(histogram);
histogramFrame.pack();
histogramFrame.setTitle("Histogram");
}

private int[] countLetters() {
    int[] count = new int[26];
    String text = jta.getText();
    for (int i = 0; i < text.length(); i++) {
        char character = text.charAt(i);

        if (character >= 'A' && character <= 'Z') {
            count[(int)character - 65]++;
        }
        else if (character >= 'a' && character <= 'z') {
            count[(int)character - 97]++;
        }
    }

    return count;
}

public static void main(String[] args) {
    MultipleWindowsDemo frame = new MultipleWindowsDemo();
    frame.setLocationRelativeTo(null);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setTitle("MultipleWindowsDemo");
    frame.pack();
    frame.setVisible(true);
}

}

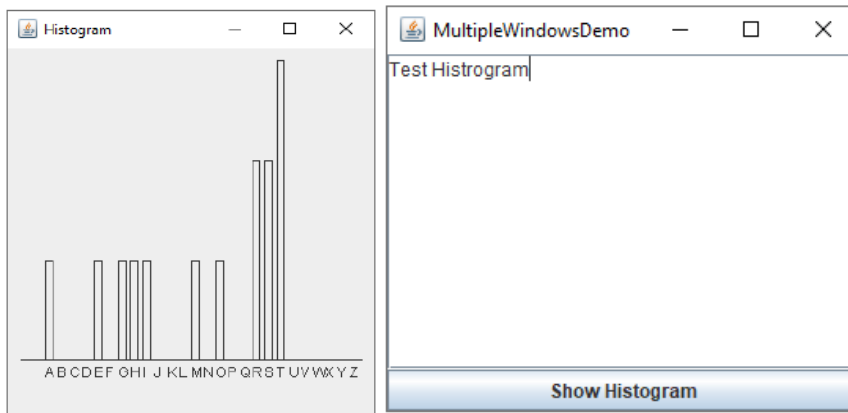
class Histogram extends JPanel {
    private int[] count;
    public void showHistogram(int[] count) {
        this.count = count;
        repaint();
    }

    @Override
    protected void paintComponent(Graphics g) {
        if (count == null) return;
        super.paintComponent(g);
        int width = getWidth();
        int height = getHeight();
        int interval = (width - 40) / count.length;
        int individualWidth = (int)((width - 40) / 24 * 0.60);
        int maxCount = 0;
        for (int i = 0; i < count.length; i++) {
            if (maxCount < count[i])
                maxCount = count[i];
        }
        int x = 30;
        g.drawLine(10, height - 45, width - 10, height - 45);
        for (int i = 0; i < count.length; i++) {
            int barHeight =
                (int)((double)count[i] / (double)maxCount * (height - 55));
            g.drawRect(x, height - 45 - barHeight, individualWidth,
                barHeight);
            g.drawString((char)(65 + i) + "", x, height - 30);
            x += interval;
        }
    }
}

```

```
@Override
public Dimension getPreferredSize() {
    return new Dimension(300, 300);
}
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

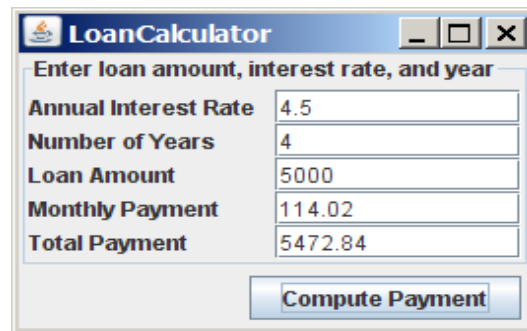


11.5 บทสรุป

คลาสที่ทำหน้าที่ในการจัดการเกี่ยวกับหน้าจอหรือ User interface จะอยู่ในแพ็คเกจของจาวาที่ชื่อ Abstract Windows Toolkit (AWT) ทุก ๆ platform ที่รันภาษาจาวา components ของ package AWT จะถูกเรียกอัตโนมัติโดยการเลือกคอมโพเนนต์ที่ตรงกับระบบปฏิบัติการแต่ละประเภท แพคเกจ AWT จะช่วยในการสร้างโปรแกรม GUI ประเภท Look and Feel ที่ขึ้นอยู่กับแพลตฟอร์มที่ใช้งาน ในแพคเกจ java.awt มีคลาสและอินเตอร์เฟสที่สำคัญดังนี้ Component Container LayoutManager Graphics Color Font การเขียนโปรแกรม GUI นั้นจะเป็นการสร้างออบเจกต์ต่าง ๆ ที่เป็นออบเจกต์ของคลาสที่เป็นส่วนประกอบกราฟฟิก (Graphic component) โดยคลาสที่เป็นส่วนประกอบกราฟฟิกจะสืบทอดมาจากคลาสที่ชื่อว่า Component คลาสที่เป็น Subclass ของคลาส Component จะแบ่งเป็นสองกลุ่มคือ คลาสที่เป็นคลาสประเภท Container เป็นคลาสที่ใช้ในการใส่ส่วนประกอบกราฟฟิกต่าง ๆ คลาสที่เป็นส่วนประกอบกราฟฟิกอื่น ๆ เช่น Button Choice และ List เป็นต้น ใน Class Component จะมี method ที่ใช้วาด object ต่าง ๆ ที่สร้างจาก class นี้ได้แก่ paint () และ repaint () ในส่วนของแพคเกจ Swing จะมีคลาส JComponent เป็น super-class ของ Swing component ส่วน Class Container เป็นส่วนหนึ่งของพื้นที่ที่ใช้ในการแสดงผลทาง graphics เมธอดที่ใช้กันบ่อยได้แก่ add () สำหรับใช้ในการเพิ่ม graphical object และ setLayout () สำหรับใช้ในการกำหนดโครงสร้างและช่วยในการกำหนดตำแหน่งและขนาดของ components โปรแกรม GUI จะต้องมีการสร้างออบเจกต์ของคลาสประเภท Container อย่างน้อย 1 ออบเจกต์ขึ้นมาก่อน เพื่อใช้ในการใส่ออบเจกต์ของคลาสที่เป็นส่วนประกอบกราฟฟิกอื่น ๆ คลาสประเภท Container ที่อยู่ในแพคเกจ AWT มีดังนี้ Frame Panel Dialog Applet

11.6 แบบฝึกหัดปฏิบัติการ

1. ให้ศึกษาและทดลองพิมพ์ตัวอย่างการสร้าง GUI จากตัวอย่างต่อไปนี้ ให้เขียนอธิบายการทำงานในแต่ละบรรทัด



```

1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4 import javax.swing.border.TitledBorder;
5 public class LoanCalculator extends JFrame {
6     // Create text fields for interest rate, years
7     // loan amount, monthly payment, and total payment
8     private JTextField jtfAnnualInterestRate = new JTextField();
9     private JTextField jtfNumberOfYears = new JTextField();
10    private JTextField jtfLoanAmount = new JTextField();
11    private JTextField jtfMonthlyPayment = new JTextField();
12    private JTextField jtfTotalPayment = new JTextField();
13
14    // Create a Compute Payment button
15    private JButton jbtComputeLoan = new JButton("Compute Payment");
16    public LoanCalculator() {
17        // Panel p1 to hold labels and text fields
18        JPanel p1 = new JPanel(new GridLayout(5, 2));
19        p1.add(new JLabel("Annual Interest Rate"));
20        p1.add(jtfAnnualInterestRate);
21        p1.add(new JLabel("Number of Years"));
22        p1.add(jtfNumberOfYears);
23        p1.add(new JLabel("Loan Amount"));
24        p1.add(jtfLoanAmount);
25        p1.add(new JLabel("Monthly Payment"));
26        p1.add(jtfMonthlyPayment);
27        p1.add(new JLabel("Total Payment"));
28        p1.add(jtfTotalPayment);
29        p1.setBorder(new
30            TitledBorder("Enter loan amount, interest rate, and years"));
31        // Panel p2 to hold the button
32        JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
33        p2.add(jbtComputeLoan);
34        // Add the panels to the frame
35        add(p1, BorderLayout.CENTER);
36        add(p2, BorderLayout.SOUTH);
37
38        // Register listener
39        jbtComputeLoan.addActionListener(new ButtonListener());
40    }
41    /** Handle the Compute Payment button */
42    private class ButtonListener implements ActionListener {
43        @Override
44        public void actionPerformed(ActionEvent e) {
45            // Get values from text fields

```

```

46     double interest =
47         Double.parseDouble(jtfAnnualInterestRate.getText());
48     int year = Integer.parseInt(jtfNumberOfYears.getText());
49     double loanAmount =
50         Double.parseDouble(jtfLoanAmount.getText());
51
52     // Create a loan object
53     Loan loan = new Loan(interest, year, loanAmount);
54
55     // Display monthly payment and total payment
56     jtfMonthlyPayment.setText(String.format("%.2f",
57         loan.getMonthlyPayment()));
58     jtfTotalPayment.setText(String.format("%.2f",
59         loan.getTotalPayment()));
60 }
61 }
62 public static void main(String[] args) {
63     LoanCalculator frame = new LoanCalculator();
64     frame.pack();
65     frame.setTitle("LoanCalculator");
66     frame.setLocationRelativeTo(null); // Center the frame
67     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
68     frame.setVisible(true);
69 }
70 }

```

```

1 public class Loan {
2     private double annualInterestRate;
3     private int numberOfYears;
4     private double loanAmount;
5     private java.util.Date loanDate;
6
7     /** Default constructor */
8     public Loan() {
9         this(2.5, 1, 1000);
10    }
11
12    /** Construct a loan with specified annual interest rate,
13        number of years, and loan amount
14        */
15    public Loan(double annualInterestRate, int numberOfYears,
16        double loanAmount) {
17        this.annualInterestRate = annualInterestRate;
18        this.numberOfYears = numberOfYears;
19        this.loanAmount = loanAmount;
20        loanDate = new java.util.Date();
21    }
22
23    /** Return annualInterestRate */
24    public double getAnnualInterestRate() {
25        return annualInterestRate;
26    }
27
28    /** Set a new annualInterestRate */
29    public void setAnnualInterestRate(double annualInterestRate) {
30        this.annualInterestRate = annualInterestRate;
31    }
32    /** Return numberOfYears */
33    public int getNumberOfYears() {
34        return numberOfYears;
35    }
36    /** Set a new numberOfYears */

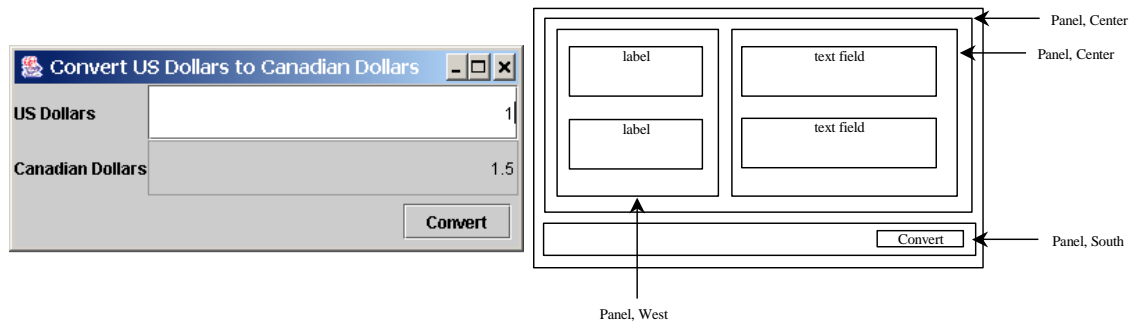
```

```

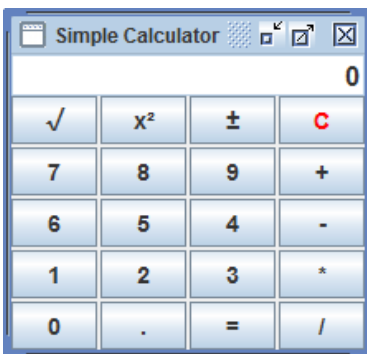
37 public void setNumberOfYears(int numberOfYears) {
38     this.numberOfYears = numberOfYears;
39 }
40 }

```

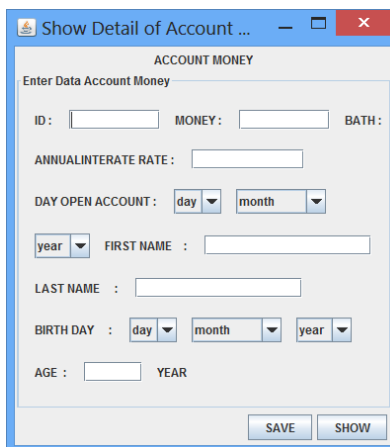
2. จาก GUI ของโปรแกรมสำหรับการแปลงเงิน US Dollars เป็น Canadian Dollars จงเขียนโปรแกรมสำหรับการแปลงเงิน US Dollars เป็น Canadian Dollars ต่อไปนี้



3. จาก GUI ของเครื่องคิดเลข ให้เพิ่มคำสั่งที่ทำให้เครื่องคิดเลขสามารถคำนวณได้




4. [Application] จาก GUI สำหรับระบบเก็บชื่อผู้เปิดบัญชีธนาคารจากคลาส Account ให้เพิ่มคำสั่งที่ทำให้โปรแกรมสามารถบันทึกข้อมูลได้



5. ให้ออกแบบคลาส AngryBirds และออกแบบ GUI ของเกมส์ตามหน้าจอดังนี้

SCENE 1: At Tokyo

SCORE



Bird Position in y-axis m

Shooting speed m/s

Angle degree

บทที่ 12 การสร้างกราฟิกในจาวา

วัตถุประสงค์

12.1 สามารถอธิบายระบบ *coordinate* ของจาวา

12.2 สามารถอธิบายส่วนประกอบของคลาส *Graphics* การทำงานของเมธอด *paintComponent*

12.3 สามารถอธิบายการทำงานของเมธอดสำหรับการวาดกราฟิกในจาวาได้

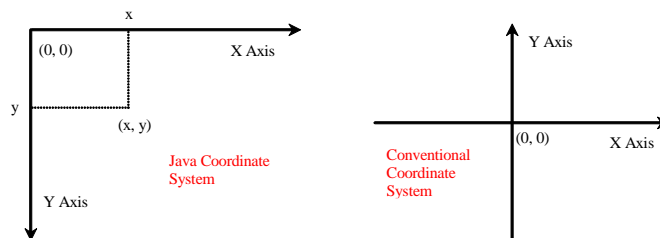
12.1 ความนำ

การพัฒนาโปรแกรมเชิงวัตถุโดยเฉพาะภาษาจาวาสามารถแสดงในรูปของกราฟิกได้เพื่อให้โปรแกรมน่าสนใจ เช่น การวาดภาพเรขาคณิตต่างๆ การทำให้ภาพเคลื่อนไหว เราสามารถเขียนข้อความ วาดเส้นตรง วาดสี่เหลี่ยม วาดสามเหลี่ยม โพลีกอน โดยใช้เมธอดต่างๆ ที่อยู่ภายในคลาส **Graphics** รายละเอียดของเมธอดต่าง ๆ จะได้อธิบายในลำดับถัดไป

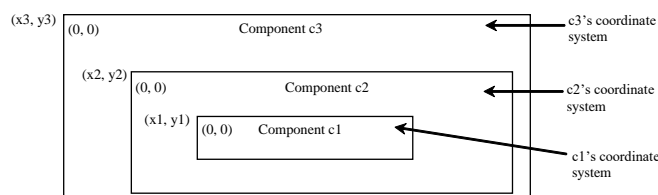
12.2 ระบบพิกัด (coordinate) ของจาวา

การเขียนกราฟโดยทั่วไปจะมีการกำหนดแกน x y และมีค่าบวก ลบ และจุด $0\ 0$ จะอยู่ตรงกลางกราฟที่จุดตัด x y แต่ในระบบพิกัดจุดในหน้าจอคอมพิวเตอร์จะไม่มีค่าที่เป็นลบ มีเฉพาะค่าบวก ซึ่งจุด $0\ 0$ จะอยู่ที่มุมบนซ้าย โดยมีการเพิ่มค่า x จากซ้ายไปขวา ส่วนการเพิ่มค่าของ y จะเพิ่มจากบนลงล่าง

ระบบพิกัดหรือ **Coordinate** ที่ใช้ในการอ้างอิงตำแหน่งของวัตถุที่แสดงในหน้าต่างหนึ่งๆ สามารถแสดงได้ดังรูปต่อไปนี้



คอมโพเนนต์แต่ละตัวจะมีระบบพิกัดอ้างอิงกับตัวคอมโพเนนต์เองและจะมีระบบพิกัดที่อ้างอิงกับตัวคอมโพเนนต์อื่นดังแสดงตัวอย่างด้านล่าง



12.3 คลาส Graphics

เราสามารถเขียนข้อความ วาดเส้นตรง วาดสี่เหลี่ยม วาดสามเหลี่ยม โพลีกอนโดยการใช้อนุเมธอดต่าง ๆ ที่อยู่ภายใน คลาส **Graphics** ตัวอย่างเมธอดที่ใช้ในการวาดรูปกราฟิกต่าง ๆ มีดังนี้

เมธอด	รายละเอียด
<code>public void drawLine(int x1, int y1, int x2, int y2)</code>	วาดเส้นจากจุด (x1, y1) ไปยังจุด (x2, y2)
<code>public void drawRect(int x, int y, int width, int height)</code>	วาดสี่เหลี่ยม จุด (x,y) คือจุดมุมบนซ้ายของสี่เหลี่ยม
<code>public void fillRect(int x, int y, int width, int height)</code>	วาดสี่เหลี่ยมที่มีการระบายสีภายในสี่เหลี่ยม
<code>public void clearRect(int x, int y, int width, int height)</code>	วาดสี่เหลี่ยมและระบายสีภายในด้วยสี background
<code>public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	วาดสี่เหลี่ยมมุมมน
<code>public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	วาดสี่เหลี่ยมมุมมนและระบายสีภายในด้วย
<code>public void draw3DRect(int x, int y, int width, int height, boolean b)</code>	วาดสี่เหลี่ยมในระบบ 3 มิติ หาก b มีค่าเป็น true สี่เหลี่ยมจะนูนขึ้น แต่หาก b มีค่าเป็น false สี่เหลี่ยมจะบุ๋มลง
<code>public void fill3DRect(int x, int y, int width, int height,boolean b)</code>	วาดสี่เหลี่ยมในระบบ 3 มิติ และระบายสีภายในด้วย
<code>public void drawOval (int x, int y, int width, int height)</code>	วาดวงกลมหรือวงรีภายในโครงสี่เหลี่ยมที่กำหนดใน parameter ทั้งสี่ โดยเส้นรอบวงจะสัมผัสกับจุดกึ่งกลางด้านแต่ละด้านของสี่เหลี่ยม
<code>public void fillOval(int x, int y, int width, int height)</code>	วาดวงกลมหรือวงรี และระบายสีภายในด้วย
<code>public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)</code>	วาดส่วนโค้งซึ่งอยู่ภายใต้กรอบสี่เหลี่ยม เช่นเดียวกับการวาดวงกลม/วงรี แต่ส่วนโค้งที่วาดจะเริ่มต้นวาดที่มุมที่ระบุใน startAngle และจากจุดนี้กวาดไปเป็นมุม arcAngle องศา
<code>public void fillArc (int x, int y, int width, int height, int startAngle, int arcAngle)</code>	วาดส่วนโค้งและระบายสีภายในส่วนโค้งนั้นๆ ซึ่งขอบเขตจะเริ่มจากศูนย์กลางของกรอบสี่เหลี่ยม ไปยังขอบของส่วนโค้งนั้นๆ ในการกวาดมุมของส่วนโค้งที่วาด หากค่ามุมมีค่าเป็นบวก การกวาดมุมจะกวาดในทิศทางทวนเข็มนาฬิกา แต่หากค่ามุมมีค่าเป็นลบ การกวาดมุมจะกวาดในทิศทางตามเข็มนาฬิกา
<code>public void drawPolygon(int xPoints[], int yPoints[], int points)</code>	xPoints และ yPoints เป็น array ที่เก็บค่า coordinate ของแต่ละจุดที่จะทำ การวาด polygon method นี้จะวาด polygon ปิด แม้ว่าจุดสุดท้ายกับจุดแรกจะไม่ใช้จุดเดียวกัน argument ตัวสุดท้าย คือ จำนวนจุด
<code>public void drawPolyline(int xPoints[], int yPoints[], int points)</code>	เป็นการวาดเส้นเชื่อมจุดแต่ละจุดที่ระบุใน xPoints และ yPoints หากจุดสุดท้ายกับจุดแรกไม่ใช่จุดเดียวกัน ก็จะไม่มีการลากจุดเชื่อมระหว่างจุดแรกกับจุดสุดท้าย
<code>public void drawPolygon (Polygon p)</code>	วาดรูป polygon จาก object p

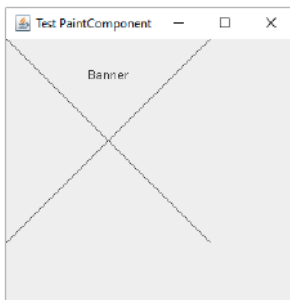
public void fillPolygon(int xPoints[], int yPoints[],int points);	วาดรูป polygon และระบายสีภายในรูปด้วย
public Polygon()	เป็นการสร้าง Polygon object แต่ยังไม่มีการระบุจุดต่างๆ ของ polygon
public Polygon(int xValues[], int yValues[], int numberOfPoints)	เป็นการสร้าง Polygon object พร้อมระบุจุดต่างๆ ของ polygon ด้วย numberOfPoints เป็นจำนวนด้านของ polygon นั้น

เมื่อต้องการวาดรูปวาดต่าง ๆ บนคอมพิวเตอร์ จำเป็นต้องกำหนดให้คลาสที่จะใช้วาดภาพสืบทอดคลาส JPanel และโอเวอร์ไรด์เมธอด paintComponent เพื่อแสดงรายละเอียดถึงสิ่งที่จำเป็นต้องการวาด ตัวอย่างการโอเวอร์ไรด์เมธอด paintComponent สามารถแสดงได้ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 12.1 ตัวอย่างการโอเวอร์ไรด์เมธอด paintComponent

```
import javax.swing.*;
import java.awt.Graphics;
public class Ex12_1 extends JFrame {
    public Ex12_1() {
        add(new JPanel());
    }
    public static void main(String[] args) {
        Ex12_1 frame = new Ex12_1();
        frame.setTitle("Test PaintComponent");
        frame.setSize(300, 300);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
class JPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawLine(0, 0, 200, 200);
        g.drawLine(0, 200, 200, 0);
        g.drawString("Banner", 80, 40);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



12.4 การวาดภาพกราฟิกเบื้องต้น

12.4.1 การวาดวงกลม สีเหลี่ยม วงรี

ตัวอย่างเมธอดที่ใช้ในการวาดเส้น สีเหลี่ยม และวงกลม มีดังนี้

เมธอด	รายละเอียด
<code>public void drawLine(int x1, int y1, int x2, int y2)</code>	วาดเส้นจากจุด (x1, y1) ไปยังจุด (x2, y2)
<code>public void drawRect(int x, int y, int width, int height)</code>	วาดสี่เหลี่ยม จุด (x,y) คือจุดมุมบนซ้ายของสี่เหลี่ยม
<code>public void fillRect(int x, int y, int width, int height)</code>	วาดสี่เหลี่ยมที่มีการระบายสีภายในสี่เหลี่ยม
<code>public void clearRect(int x, int y, int width, int height)</code>	วาด สีเหลี่ยมและระบายสีภายในด้วยสี background
<code>public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	วาดสี่เหลี่ยมมุมมน
<code>public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	วาดสี่เหลี่ยมมุมมนและระบายสีภายในด้วย
<code>public void draw3DRect(int x, int y, int width, int height, boolean b)</code>	วาดสี่เหลี่ยมในระบบ 3 มิติ หาก b มีค่าเป็น true สี่เหลี่ยมจะนูนขึ้น แต่หาก b มีค่าเป็น false สี่เหลี่ยมจะบุ๋มลง
<code>public void fill3Drect(int x, int y, int width, int height,boolean b)</code>	วาดสี่เหลี่ยมในระบบ 3 มิติ และระบายสีภายในด้วย
<code>public void drawOval (int x, int y, int width, int height)</code>	วาดวงกลมหรือวงรีภายในโครงสร้างสี่เหลี่ยมที่กำหนดใน parameter ทั้งสี่ โดยเส้นรอบวงจะสัมผัสกับจุดกึ่งกลางด้านแต่ละด้านของสี่เหลี่ยม
<code>public void fillOval(int x, int y, int width, int height)</code>	วาดวงกลมหรือวงรี และระบายสีภายในด้วย

ตัวอย่างที่ 12.2 ตัวอย่างเมธอดที่ใช้ในการวาดเส้น สีเหลี่ยม และวงกลม

```
import java.awt.*;
import javax.swing.JPanel;
import javax.swing.*;
public class TestFigurePanel extends JFrame {
    public TestFigurePanel() {
        setLayout(new GridLayout(2, 3, 5, 5));
        add(new FigurePanel(FigurePanel.LINE));
        add(new FigurePanel(FigurePanel.RECTANGLE));
        add(new FigurePanel(FigurePanel.ROUND_RECTANGLE));
        add(new FigurePanel(FigurePanel.OVAL));
        add(new FigurePanel(FigurePanel.RECTANGLE, true));
        add(new FigurePanel(FigurePanel.ROUND_RECTANGLE, true));
    }
    public static void main(String[] args) {
        TestFigurePanel frame = new TestFigurePanel();
        frame.setSize(400, 200);
        frame.setTitle("TestFigurePanel");
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

```

}

class FigurePanel extends JPanel {
    public static final int LINE = 1;
    public static final int RECTANGLE = 2;
    public static final int ROUND_RECTANGLE = 3;
    public static final int OVAL = 4;
    private int type = 1;
    private boolean filled = false;
    public FigurePanel() {
    }
    public FigurePanel(int type) {
        this.type = type;
    }
    public FigurePanel(int type, boolean filled) {
        this.type = type;
        this.filled = filled;
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int width = getWidth();
        int height = getHeight();
        switch (type) {
            case LINE:
                g.setColor(Color.BLACK);
                g.drawLine(10, 10, width - 10, height - 10);
                g.drawLine(width - 10, 10, 10, height - 10);
                break;
            case RECTANGLE:
                g.setColor(Color.BLUE);
                if (filled)
                    g.fillRect((int)(0.1 * width), (int)(0.1 * height),
                        (int)(0.8 * width), (int)(0.8 * height));
                else
                    g.drawRect((int)(0.1 * width), (int)(0.1 * height),
                        (int)(0.8 * width), (int)(0.8 * height));
                break;
            case ROUND_RECTANGLE:
                g.setColor(Color.RED);
                if (filled)
                    g.fillRoundRect((int)(0.1 * width), (int)(0.1 * height),
                        (int)(0.8 * width), (int)(0.8 * height), 20, 20);
                else
                    g.drawRoundRect((int)(0.1 * width), (int)(0.1 * height),
                        (int)(0.8 * width), (int)(0.8 * height), 20, 20);
                break;
            case OVAL:
                g.setColor(Color.BLACK);
                if (filled)
                    g.fillOval((int)(0.1 * width), (int)(0.1 * height),
                        (int)(0.8 * width), (int)(0.8 * height));
                else
                    g.drawOval((int)(0.1 * width), (int)(0.1 * height),
                        (int)(0.8 * width), (int)(0.8 * height));
        }
    }
    public void setType(int type) {
        this.type = type;
        repaint();
    }
    public int getType() {
        return type;
    }
    public void setFilled(boolean filled) {
        this.filled = filled;
    }
}

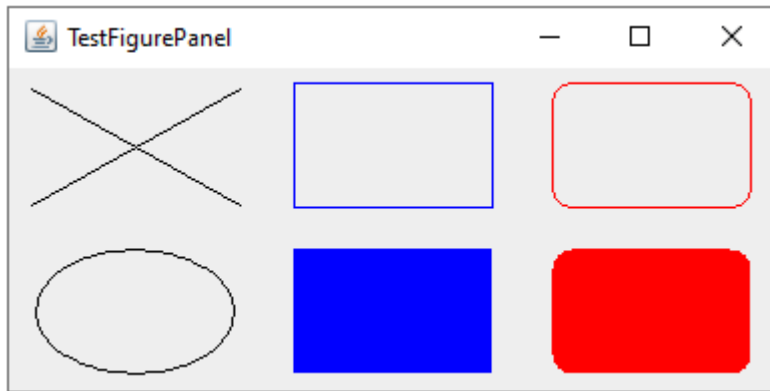
```

```

repaint();
}
public boolean isFilled() {
    return filled;
}
@Override
public Dimension getPreferredSize() {
    return new Dimension(80, 80);
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



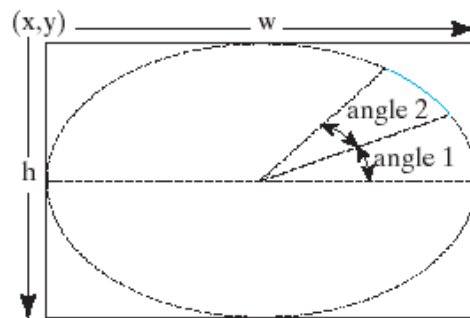
12.4.2 การวาดส่วนโค้ง สามารถทำได้โดยการใช้เมธอดต่อไปนี้

เมธอด

`public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)`

รายละเอียด

วาดส่วนโค้งซึ่งอยู่ภายใต้กรอบสี่เหลี่ยม เช่นเดียวกับการวาดวงกลม/วงรี แต่ส่วนโค้งที่วาดจะเริ่มต้นวาดที่มุมที่ระบุใน **startAngle** และจากจุดนี้กวาดไปเป็นมุม **arcAngle** องศา



`public void fillArc (int x, int y, int width, int height, int startAngle, int arcAngle)`

วาดส่วนโค้งและระบายสีภายในส่วนโค้งนั้นๆ ซึ่งขอบเขตจะเริ่มจากศูนย์กลางของกรอบสี่เหลี่ยม ไปยังขอบของส่วนโค้งนั้นๆ ในการกวาดมุมของส่วนโค้งที่วาด หากค่ามุมมีค่าเป็นบวก การกวาดมุมจะกวาดในทิศทางทวนเข็มนาฬิกา แต่หากค่ามุมมีค่าเป็นลบ การกวาดมุมจะกวาดในทิศทางตามเข็มนาฬิกา

ตัวอย่างที่ 12.3 ตัวอย่างเมธอดที่ใช้ในการวาดเส้น สีเหลี่ยม และวงกลม

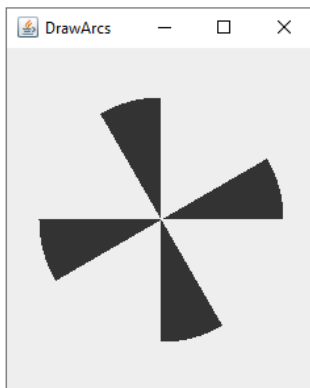
```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;
public class DrawArcs extends JFrame {
    public DrawArcs() {
        setTitle("DrawArcs");
        add(new ArcsPanel());
    }
    public static void main(String[] args) {
        DrawArcs frame = new DrawArcs();
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(250, 300);
        frame.setVisible(true);
    }
}
class ArcsPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;
        int radius = (int) (Math.min(getWidth(), getHeight()) * 0.4);

        int x = xCenter - radius;
        int y = yCenter - radius;

        g.fillArc(x, y, 2 * radius, 2 * radius, 0, 30);
        g.fillArc(x, y, 2 * radius, 2 * radius, 90, 30);
        g.fillArc(x, y, 2 * radius, 2 * radius, 180, 30);
        g.fillArc(x, y, 2 * radius, 2 * radius, 270, 30);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

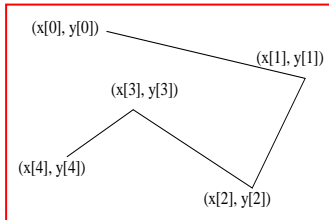


12.4.3 การวาด Polygons และ Polylines

Polygons คือ รูปหลายเหลี่ยม ส่วน polylines คือ การวาดเส้นหลายเส้นเชื่อมต่อกัน เช่นต้องการวาดรูปที่มีพิกัดดังนี้

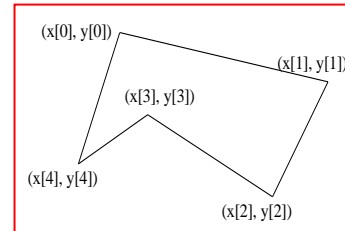
```
int[] x = {40, 70, 60, 45, 20};
```

```
int[] y = {20, 40, 80, 45, 60};
```



เราสามารถใช้อเมธอดต่อไปนี้สำหรับวาดภาพโพลีไลน์

```
g.drawPolyline(x, y, x.length);
```



เราสามารถใช้อเมธอดต่อไปนี้สำหรับวาดภาพโพลีกอน

```
g.drawPolygon(x, y, x.length);
```

การวาด polygons และ polylines สามารถทำได้โดยการใช้เมธอดต่อไปนี้

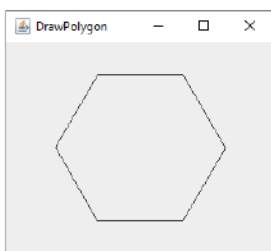
เมธอด	รายละเอียด
<code>public void drawPolygon(int xPoints[], int yPoints[], int points)</code>	xPoints และ yPoints เป็น array ที่เก็บค่า coordinate ของแต่ละจุดที่จะทำ การวาด polygon method นี้จะวาด polygon ปิด แม้ว่าจุดสุดท้ายกับจุดแรกจะไม่ใช่ว่าจุดเดียวกัน argument ตัวสุดท้าย คือ จำนวนจุด
<code>public void drawPolygon(int xPoints[], int yPoints[], int points)</code>	เป็นการวาดเส้นเชื่อมจุดแต่ละจุดที่ระบุใน xPoints และ yPoints หากจุดสุดท้ายกับจุดแรกไม่ใช่จุดเดียวกัน ก็จะไม่มีการลากจุดเชื่อมระหว่างจุดแรกกับจุดสุดท้าย
<code>public void drawPolygon (Polygon p)</code>	วาดรูป polygon จาก object p
<code>public void fillPolygon(int xPoints[], int yPoints[], int points)</code>	วาดรูป polygon และระบายสีภายในรูปด้วย
<code>public Polygon()</code>	เป็นการสร้าง Polygon object แต่ยังไม่มีการระบุจุดต่างๆ ของ polygon
<code>public Polygon(int xValues[], int yValues[], int numberOfPoints)</code>	เป็นการสร้าง Polygon object พร้อมระบุจุดต่างๆ ของ polygon ด้วย numberOfPoints เป็นจำนวนด้านของ polygon นั้น

ตัวอย่างที่ 12.4 ตัวอย่างเมธอดที่ใช้ในการวาดเส้น สีเหลี่ยม และวงกลม

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Polygon;
public class DrawPolygon extends JFrame {
    public DrawPolygon() {
        setTitle("DrawPolygon");
        add(new PolygonsPanel());
    }
    public static void main(String[] args) {
        DrawPolygon frame = new DrawPolygon();
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 250);
        frame.setVisible(true);
    }
}
class PolygonsPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

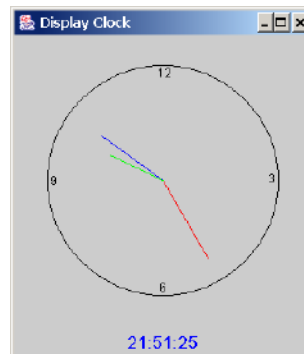
        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;
        int radius = (int) (Math.min(getWidth(), getHeight()) * 0.4);
        Polygon polygon = new Polygon();
        polygon.addPoint(xCenter + radius, yCenter);
        polygon.addPoint((int) (xCenter + radius *
            Math.cos(2 * Math.PI / 6)), (int) (yCenter - radius *
            Math.sin(2 * Math.PI / 6)));
        polygon.addPoint((int) (xCenter + radius *
            Math.cos(2 * 2 * Math.PI / 6)), (int) (yCenter - radius *
            Math.sin(2 * 2 * Math.PI / 6)));
        polygon.addPoint((int) (xCenter + radius *
            Math.cos(3 * 2 * Math.PI / 6)), (int) (yCenter - radius *
            Math.sin(3 * 2 * Math.PI / 6)));
        polygon.addPoint((int) (xCenter + radius *
            Math.cos(4 * 2 * Math.PI / 6)), (int) (yCenter - radius *
            Math.sin(4 * 2 * Math.PI / 6)));
        polygon.addPoint((int) (xCenter + radius *
            Math.cos(5 * 2 * Math.PI / 6)), (int) (yCenter - radius *
            Math.sin(5 * 2 * Math.PI / 6)));
        g.drawPolygon(polygon);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

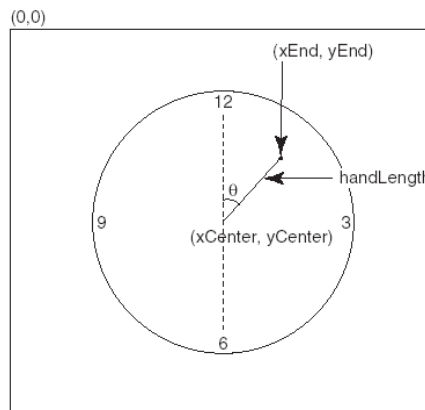


12.5 ตัวอย่างการประยุกต์การวาดภาพกราฟิกด้วยภาษาจาวา

ต้องการวาดนาฬิกาที่มีการแสดงเวลาดังรูป



จากรูปนาฬิกาจะประกอบด้วยเข็มนาฬิกาจำนวน 3 เข็มคือ ชั่วโมง นาที วินาที โดยความยาวของเข็มแต่ละอันจะ 3 ไม่เท่ากัน เข็มทั้งสามจะมีจุดเริ่มต้นที่ตำแหน่งเดียวกันคือจุดศูนย์กลางของวงกลมดังนั้นในการวาดเข็มนาฬิกาแต่ละเข็มจะต้องเริ่มต้นที่ตำแหน่งเดียวกัน หลังจากนั้นจุดที่เป็นจุดสิ้นสุดของเข็มแต่ละอันจะได้รับการคำนวณจากเวลาปัจจุบันโดยสูตรในการคำนวณตำแหน่งสุดท้ายของเข็มแต่ละเข็มสามารถคำนวณได้ดังนี้ จากรูป



เข็มวินาที

$$xEnd = xCenter + handLength \times \sin(\theta)$$

$$yEnd = yCenter - handLength \times \cos(\theta)$$

เนื่องจากในหนึ่งนาทึมี 60วินาทีดังนั้นมุมของเข็มในแต่ละวินาทีคำนวณได้จาก $second \times (2\pi/60)$

เข็มนาฬิกา

$$xEnd = xCenter + handLength \times \sin(\theta)$$

$$yEnd = yCenter - handLength \times \cos(\theta)$$

ตำแหน่งของเข็มนาฬิกาจะคำนวณได้จากการพิจารณาเข็มของนาฬิกาและวินาทีโดย ต้องนำนาฬิกาที่รวมกับ

วินาที/60 ดังนั้นมุมของเข็มในแต่ละวินาทีคำนวณได้จาก $(\text{minute} + \text{second}/60) \times (2\pi/60)$

เข็มชั่วโมง

$$x_{\text{End}} = x_{\text{Center}} + \text{handLength} \times \sin(\theta)$$

$$y_{\text{End}} = y_{\text{Center}} - \text{handLength} \times \cos(\theta)$$

เนื่องจากในหนึ่งวงกลมมี ชั่วโมง ดังนั้นมุมของเข็มในแต่ละชั่วโมงจะคำนวณได้จากการพิจารณาเข็มของ 12 ชั่วโมง นาทีและวินาทีโดย ต้องนำชั่วโมง รวมกับนาที/รวมกับ วินาที 60/60 ดังนั้นมุมของเข็มในแต่ละชั่วโมงคำนวณได้จาก $(\text{hour} + \text{minute}/60 + \text{second}/(60 \times 60)) \times (2\pi/12)$

ตัวอย่างที่ 12.5 ตัวอย่างการประยุกต์การวาดภาพกราฟิกด้วยภาษาจาวา

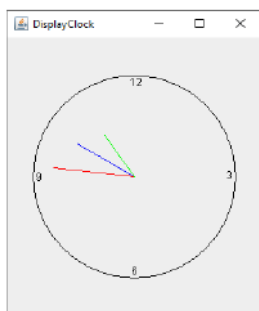
```
import java.awt.*;
import javax.swing.*;
import java.util.*;
public class StillClock extends JPanel {
    private int hour;
    private int minute;
    private int second;
    public StillClock() {
        setCurrentTime();
    }
    public StillClock(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }
    public int getHour() {
        return hour;
    }
    public void setHour(int hour) {
        this.hour = hour;
        repaint();
    }
    public int getMinute() {
        return minute;
    }
    public void setMinute(int minute) {
        this.minute = minute;
        repaint();
    }
    public int getSecond() {
        return second;
    }
    public void setSecond(int second) {
        this.second = second;
        repaint();
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        int clockRadius =
            (int) (Math.min(getWidth(), getHeight()) * 0.8 * 0.5);
        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;
        g.setColor(Color.black);
        g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
            2 * clockRadius, 2 * clockRadius);
    }
}
```

```

g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);
int sLength = (int) (clockRadius * 0.8);
int xSecond = (int) (xCenter + sLength *
    Math.sin(second * (2 * Math.PI / 60)));
int ySecond = (int) (yCenter - sLength *
    Math.cos(second * (2 * Math.PI / 60)));
g.setColor(Color.red);
g.drawLine(xCenter, yCenter, xSecond, ySecond);
int mLength = (int) (clockRadius * 0.65);
int xMinute = (int) (xCenter + mLength *
    Math.sin(minute * (2 * Math.PI / 60)));
int yMinute = (int) (yCenter - mLength *
    Math.cos(minute * (2 * Math.PI / 60)));
g.setColor(Color.blue);
g.drawLine(xCenter, yCenter, xMinute, yMinute);
int hLength = (int) (clockRadius * 0.5);
int xHour = (int) (xCenter + hLength *
    Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
int yHour = (int) (yCenter - hLength *
    Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
g.setColor(Color.green);
g.drawLine(xCenter, yCenter, xHour, yHour);
}
public void setCurrentTime() {
    Calendar calendar = new GregorianCalendar();
    this.hour = calendar.get(Calendar.HOUR_OF_DAY);
    this.minute = calendar.get(Calendar.MINUTE);
    this.second = calendar.get(Calendar.SECOND);
}
@Override
public Dimension getPreferredSize() {
    return new Dimension(200, 200);
}
}
class DisplayClock extends JFrame {
    public DisplayClock() {
        StillClock clock = new StillClock();
        JPanel p1 = new JPanel(new GridLayout(1,3));
        p1.add(clock);
        add(p1);
    }
    public static void main(String[] args) {
        DisplayClock frame = new DisplayClock();
        frame.setTitle("DisplayClock");
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 350);
        frame.setVisible(true);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



12.6 บทสรุป

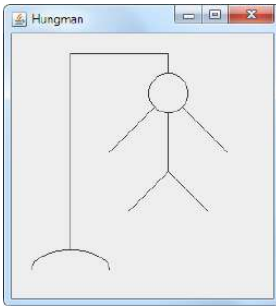
ระบบ coordinate ของจาวา ระบบ Coordinate ที่ใช้ในการอ้างอิงตำแหน่งของ object ที่แสดงใน window หนึ่งๆ คอมพิวเตอร์แต่ละตัวจะมีระบบพิกัดอ้างอิงกับตัวคอมพิวเตอร์เองและจะมีระบบพิกัดที่อ้างอิงกับตัวคอมพิวเตอร์อื่น

คลาส Graphics เราสามารถเขียนข้อความ วาดเส้นตรง วาดสี่เหลี่ยม วาดสามเหลี่ยม โพลีกอนโดยใช้เมธอดต่าง ๆ ที่อยู่ภายในคลาส Graphics เมธอดต่าง ๆ ที่จะสามารถเลือกใช้ได้

เมธอด paintComponent เมื่อต้องการวาดรูปวาดต่าง ๆ บนคอมพิวเตอร์ จำเป็นต้องกำหนดให้คลาสที่จะใช้วาดภาพสืบทอดคลาส JPanel และโอเวอร์ไรด์เมธอด paintComponent เพื่อแสดงรายละเอียดถึงสิ่งที่จำเป็นต้องการวาด

12.7 แบบฝึกหัดปฏิบัติการ

1. ให้สร้าง GUI ของเกมส์ Hangman ต่อไปนี้



- 2 จงเขียน class ที่วาดกราฟจากฟังก์ชันต่อไปนี้

```
abstract class AbstractDrawFunction extends JPanel
{
    /**Polygon to hold the points*/
    private Polygon p = new Polygon();
    /**Default constructor*/
    protected AbstractDrawFunction ()
    {
        drawFunction();
        setBackground(Color.white);
    }
    /**Draw the function*/
    public abstract double f(double x);
    /**Obtain points for x coordinates 100, 101, ..., 300*/
    public void drawFunction()
    {
        for (int x = -100; x <= 100; x++)
        {
            p.addPoint(x+200, 200-(int)f(x));
        }
    }
    /**Paint the function diagram*/
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        // Draw x axis
        g.drawLine(10, 200, 390, 200);
        // Draw y axis
        g.drawLine(200,30, 200, 390);
        // Draw arrows on x axis
        g.drawLine(390, 200, 370, 190);
        g.drawLine(390, 200, 370, 210);
        // Draw arrows on y axis
        g.drawLine(200, 30, 190, 50);
        g.drawLine(200, 30, 210, 50);
        // Draw x, y
        g.drawString("X", 370, 170);
        g.drawString("Y", 220, 40);
        // Draw a polygon line by connecting the points in the polygon
        g.drawPolyline(p.xpoints, p.ypoints, p.npoints);
    }
}
```

ทดสอบคลาสโดยใช้ฟังก์ชันต่อไปนี้

- a. $f(x) = x^2$;
- b. $f(x) = \sin(x)$;
- c. $f(x) = \cos(x)$;
- d. $f(x) = \tan(x)$;
- e. $f(x) = \cos(x) + 5\sin(x)$;
- f. $f(x) = 5\cos(x) + \sin(x)$;
- g. $f(x) = \log(x) + x^2$;

สำหรับฟังก์ชันแต่ละฟังก์ชันให้สร้างคลาสที่ extends คลาส AbstractDrawFunction และ implements เมธอด `f(double x)`;

กำหนดให้คลาสทดสอบคือ

```
public class Test extends JFrame
{
    public Test()
    {
        getContentPane().setLayout(new GridLayout(1, 2, 5, 5));
        getContentPane().add(new DrawSine());
    }
    public static void main(String[] args)
    {
        Test frame = new Test();
        frame.setSize(400, 400);
        frame.setTitle("Exercise 10.10");
        frame.setVisible(true);
    }
}
```

บทที่ 13 การจัดการเหตุการณ์และการจัดการข้อผิดพลาด

วัตถุประสงค์

- 13.1 สามารถอธิบายการเขียนโปรแกรมในรูปแบบ *Procedural* และ *Event-Driven*
- 13.2 สามารถอธิบายการทำงานของอินเตอร์เฟสประเภท *Listener*
- 13.3 สามารถอธิบายการจัดการกับเหตุการณ์ การควบคุมเหตุการณ์ที่เกิดขึ้นกับวินโดว์ เม้าส์ คีย์บอร์ด
- 13.4 สามารถอธิบายการทำงานของคลาส *Timer*
- 13.5 สามารถอธิบายการทำงานของจัดการข้อผิดพลาด
- 13.6 สามารถเขียนโปรแกรมเพื่อจัดการเหตุการณ์ประเภทต่างๆ ได้

13.1 ความนำ

การเขียนโปรแกรมในรูปแบบการเขียนโปรแกรมเชิงโครงสร้าง (**Procedural programming**) จะประมวลผลโปรแกรมตามลำดับของการเขียนที่เรียกว่า **procedural order** ส่วนการเขียนโปรแกรมในรูปแบบ **Event-driven programming** จะประมวลผลเมื่อเกิดเหตุการณ์เกิดขึ้น ผู้พัฒนาโปรแกรมสามารถจัดการการทำงานของโปรแกรมตามที่ต้องการเมื่อมีเหตุการณ์เกิดขึ้น เช่น เมื่อคลิกปุ่มแล้วโปรแกรมมีการคำนวณต่าง ๆ หรือเมื่อลากเมาส์แล้วโปรแกรมมีการวาดรูปบนหน้าจอตามที่ผู้ใช้งานกำหนดหรือแม้กระทั่งเมื่อผู้ใช้พิมพ์ข้อความผ่านคีย์บอร์ดแล้วโปรแกรมมีการแสดงข้อมูลบนหน้าจอ จากตัวอย่างดังกล่าวจะเป็นการเขียนโปรแกรมที่ประมวลผลเมื่อเหตุการณ์เกิดขึ้น โดยในบทนี้จะอธิบายรายละเอียดของการเขียนโปรแกรมในลักษณะ **Event-driven programming**

13.2 การเขียนโปรแกรมในรูปแบบ Procedural และ Event-Driven

เหตุการณ์ (**Event**) เป็นสถานการณ์ที่เกิดขึ้นในขณะรันโปรแกรม เช่น การใช้เมาส์หรือคีย์บอร์ดติดต่อกับโปรแกรม **GUI** การเกิดเหตุการณ์ในโปรแกรมภาษาจาวาจะเป็นการสร้างวัตถุของคลาสประเภท **Event** ชนิดต่าง ๆ ขึ้นมาตามประเภทของเหตุการณ์ เช่น

- เมื่อเลื่อนเมาส์ในเฟรมจะเกิดวัตถุของคลาส **MouseEvent** ขึ้นมา
- เมื่อปิดเฟรมจะเกิดวัตถุของคลาส **WindowEvent** ขึ้นมา
- เมื่อกดปุ่มที่อยู่ในเฟรมจะเกิดวัตถุของคลาส **ActionEvent** ขึ้นมา
- เมื่อพิมพ์ข้อความใน **TextField** จะเกิดวัตถุของคลาส **KeyEvent** ขึ้นมา
- หรือเหตุการณ์ที่เกิดกับ **OS** เช่น เหตุการณ์ของ **timer**

องค์ประกอบของเหตุการณ์ประกอบด้วย **event** คือวัตถุที่เกิดขึ้นตามประเภทของเหตุการณ์ **Event Source** คือส่วนที่ทำให้เกิดเหตุการณ์ และ **Event Handler** คือวัตถุที่ทำหน้าที่จัดการกับเหตุการณ์ที่เกิดขึ้นโดยมีเมธอดที่จะรับวัตถุชนิด **Event** ดังกล่าวและมีคำสั่งในการจัดการกับเหตุการณ์เพื่อโต้ตอบกับผู้ใช้

13.3 อินเทอร์เน็ตประเภท Listener

ภาษาจาวาจะจัดการกับเหตุการณ์โดยการสร้างวัตถุที่สามารถรับฟังเหตุการณ์จากคลาสที่อิมพลีเมนต์อินเทอร์เน็ตประเภท **Listener** ที่สอดคล้องกัน วัตถุนี้จะทำหน้าที่เป็น **Event Handler** เช่น วัตถุที่จะจัดการกับเหตุการณ์ประเภท **ActionEvent** จะต้องอิมพลีเมนต์อินเทอร์เน็ตประเภท **ActionListener**

อินเทอร์เน็ตประเภท **Listener** มีดังนี้

- ▶ **ActionListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **ActionEvent**
- ▶ **AdjustmentListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **AdjustmentEvent**
- ▶ **ComponentListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **ComponentEvent**
- ▶ **ContainerListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **ContainerEvent**
- ▶ **FocusListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **FocusEvent**
- ▶ **ItemListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **ItemEvent**
- ▶ **KeyListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **KeyEvent**
- ▶ **MouseListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **MouseEvent**
- ▶ **MouseMotionListener** เป็นอินเทอร์เน็ตสำหรับออบเจกต์ของคลาส **MouseEvent**

การที่จะให้วัตถุแต่ละชนิด เช่น การป้อนข้อมูลจากคีย์บอร์ด การกดปุ่ม การเปิดปิดหน้าต่าง การเลื่อนเมาส์ให้สามารถทำงานได้ วัตถุต่างๆ เหล่านี้จำเป็นต้องมีการลงทะเบียนกับอินเทอร์เน็ตที่เกี่ยวข้องเพื่อจะทำได้ ทำให้สามารถรับฟังเหตุการณ์ที่จะเกิดขึ้นกับวัตถุแต่ละประเภทได้ โดยเมธอดที่ใช้ในการลงทะเบียนอินเทอร์เน็ตเพื่อรับฟังเหตุการณ์ที่จะเกิดขึ้นกับวัตถุแต่ละประเภทมีดังนี้

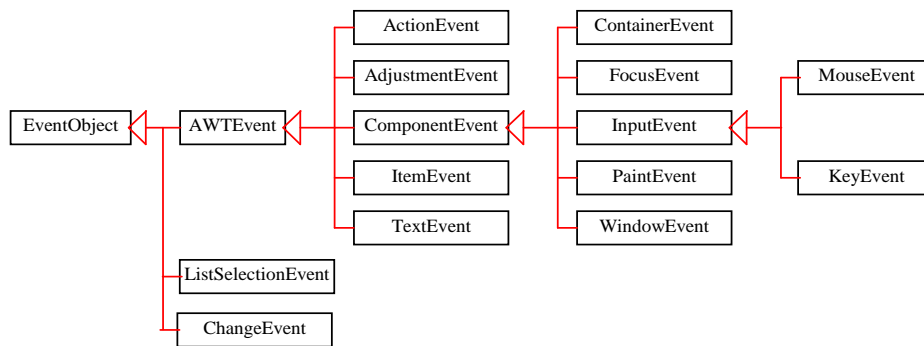
อินเทอร์เน็ต	เมธอดที่ใช้รับฟังเหตุการณ์
ActionListener	addActionListener()
ItemListener	addItemListener()
KeyListener	addKeyListener()
MouseListener	addMouseListener()
TextListener	addTextListener()
FocusListener	addFocusListener()
AdjustmentListener	addAdjustmentListener()
ComponentListener	addComponentListener()
ContainerListener	addContainerListener()
WindowListener	addWindowListener()

เนื่องจาก Listener เป็นอินเทอร์เฟซที่จะใช้ในการจัดการกับเหตุการณ์ที่เกิดขึ้นและภายในอินเทอร์เฟซแต่ละอินเทอร์เฟซจะมีเมธอดที่ยังไม่สมบูรณ์ ดังนั้นหากผู้ใช้ต้องการที่จะเขียนโปรแกรมเพื่อจัดการกับเหตุการณ์ ต้องมีการเขียนเมธอดดังต่อไปนี้เมื่อต้องการจัดการกับเหตุการณ์

อินเทอร์เฟซ	เมธอดที่ต้อง implements
ActionListener	actionPerformed(ActionEvent)
ItemListener	itemStateChanged(ItemEvent)
KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
MouseListener	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
TextListener	TextValueChanged(TextEvent)
FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener	componentMoved(ComponentEvent) componentHidden(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)
ContainerListener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
WindowListener	windowClosing(WindowEvent) windowOpened(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent)

13.4 การจัดการกับเหตุการณ์

คลาส AWTEvent เป็น superclass ของคลาสประเภท Event สำหรับเหตุการณ์ทางด้านกราฟิก ซึ่งจะมีอยู่ทั้งหมด 11 คลาส คือ ActionEvent, AdjustmentEvent, ComponentEvent, ItemEvent, TextEvent, FocusEvent, WindowEvent, InputEvent, ContainerEvent, KeyEvent และ MouseEvent



คลาส AWTEvent และคลาส EventObject มีเมธอดที่สำคัญดังนี้

Object getSource() เพื่อเรียกดูออบเจ็กต์ประเภท Event Source

int getID() เพื่อระบุชนิดของเหตุการณ์

13.4.1 รายละเอียดของเหตุการณ์ (Event)

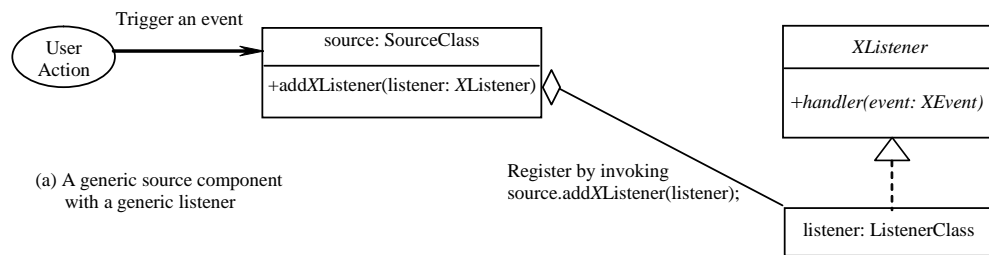
วัตถุนิต event จะประกอบด้วยคุณสมบัติต่าง ๆ ที่ผูกติดกับเหตุการณ์ที่เกิดขึ้น เราสามารถระบุวัตถุที่เกิดเหตุการณ์ได้โดยการใช้ getSource() ซึ่งเป็น instance method ในคลาส EventObject โดยที่ subclass ของ EventObject จะเกี่ยวข้องกับเหตุการณ์ชนิดต่าง ๆ เช่น เหตุการณ์ที่เกิดขึ้นจากการทำงานกับปุ่ม เหตุการณ์ที่เกิดขึ้นกับ window เหตุการณ์ที่เกิดขึ้นกับ component เหตุการณ์ที่เกิดขึ้นกับ mouse เหตุการณ์ที่เกิดขึ้นกับการกด key

ตารางต่อไปนี้แสดง การทำงานของผู้ใช้ (User Action) แหล่งที่เกิดเหตุการณ์ (Source Event Type) และวัตถุของเหตุการณ์ที่สร้างขึ้น (Object Generated)

User Action	Source Event Type	Object Generated
Click a button	JButton	ActionEvent
Click a check box	JCheckBox	ItemEvent, ActionEvent
Click a radio button	JRadioButton	ItemEvent, ActionEvent
Press return on a text field	JTextField	ActionEvent
Select a new item	JComboBox	ItemEvent, ActionEvent
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released, etc.	Component	MouseEvent
Key released, pressed, etc.	Component	KeyEvent

13.4.2 การจัดการกับเหตุการณ์

ภาษาจาวาจะมีวิธีการจัดการกับเหตุการณ์ที่เรียกว่า **Delegation Model** โดยจะมีหลักการดังนี้



1. ผู้ใช้เป็นผู้กระตุ้นให้เกิดเหตุการณ์กับวัตถุประเภทแหล่งกำเนิดเหตุการณ์ (Event Source) เช่น ปุ่ม (Button) ช่องกรอกข้อมูล (Text Field)

2. วัตถุของส่วนประกอบกราฟิกใด ๆ สามารถเป็นวัตถุประเภทของแหล่งกำเนิดเหตุการณ์ได้ เช่น ออบเจ็กต์ของคลาส **Button** สามารถเป็นแหล่งกำเนิดเหตุการณ์ของเหตุการณ์ประเภท **ActionEvent** ได้

3. คลาสใด ๆ ที่จะใช้ในการรับฟังเหตุการณ์ใด ๆ คลาสนั้นต้องอิมพลีเมนต์อินเทอร์เฟซประเภท **Listener** ที่สอดคล้องกัน เช่น คลาสที่ต้องการรับฟังเหตุการณ์ **ActionEvent** จะต้องอิมพลีเมนต์อินเทอร์เฟซที่ชื่อ **ActionListener** ซึ่งภายในอินเทอร์เฟซดังกล่าวจะมีเมธอดที่ยังไม่สมบูรณ์ที่ผู้เขียนโปรแกรมต้องทำการโอเวอร์ไรด์เพื่อให้โปรแกรมทำงานได้เมื่อมีเหตุการณ์เกิดขึ้น

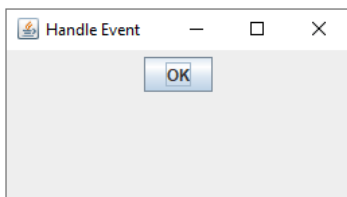
4. ออบเจ็กต์ประเภทเหตุการณ์ (**Event**) ที่เกิดจากแหล่งกำเนิดเหตุการณ์จะถูกส่งไปยังออบเจ็กต์ของคลาสที่สามารถรับฟังเหตุการณ์ประเภทนั้น โดยตัววัตถุของแหล่งกำเนิดเหตุการณ์ต้องทำการลงทะเบียนเพื่อให้สามารถรับฟังเหตุการณ์ที่จะเกิดขึ้นได้ โดยตัวออบเจ็กต์ประเภทเหตุการณ์จะถูกส่งไปยังคลาสที่ทำหน้าที่รับฟังเหตุการณ์ดังกล่าวและจะทำงานเมื่อมีเหตุการณ์เกิดขึ้น

การเขียนชุดคำสั่งเพื่อให้แหล่งกำเนิดเหตุการณ์สามารถทำงานเมื่อมีเหตุการณ์เกิดขึ้นทำได้โดย การลงทะเบียนตัวรับฟังเหตุการณ์(**Listener**) ที่เกี่ยวข้องโดยใช้เมธอด **addXListener(XListener listener)** เช่น หากแหล่งกำเนิดเหตุการณ์คือ ปุ่ม(**Button**) จะต้องลงทะเบียนโดยใช้เมธอด **addActionListener** และต้องสร้างตัวรับฟังเหตุการณ์ผูกติดกับปุ่มดังกล่าว และทำการโอเวอร์ไรด์เมธอดที่ใช้ในการจัดการเหตุการณ์ ซึ่งกรณีที่เป็นปุ่ม เมธอดที่จะใช้ในการจัดการเหตุการณ์คือ **actionPerformed()** ตัวอย่างของการจัดการเหตุการณ์โดยใช้ **Delegation Model** มีดังนี้

ตัวอย่างที่ 13.1 การเขียนชุดคำสั่งเพื่อให้แหล่งกำเนิดเหตุการณ์สามารถทำงานเมื่อมีเหตุการณ์เกิดขึ้น

```
import javax.swing.*;
import java.awt.event.*;
public class Ex13_1 extends JFrame {
    public Ex13_1() {
        JButton jbtOK = new JButton("OK");
        JPanel panel = new JPanel();
        panel.add(jbtOK);
        add(panel);
        OKListenerClass listener1 = new OKListenerClass();
        jbtOK.addActionListener(listener1);
    }
    public static void main(String[] args) {
        JFrame frame = new Ex13_1();
        frame.setTitle("Handle Event");
        frame.setSize(200, 150);
        frame.setLocation(200, 100);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
class OKListenerClass implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("OK button clicked");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



OK button clicked

13.4.3 ออบเจ็กต์ของคลาสประเภท Event Handler

แหล่งกำเนิดเหตุการณ์(Event Source) ใดต้องการที่จะจัดการเหตุการณ์ใดต้องลงทะเบียนเพื่อรับฟังเหตุการณ์ โดยมีรูปแบบดังนี้

```
eventSource.addXxxListener (Listener)
```

โดยวัตถุที่ทำหน้าที่จัดการกับเหตุการณ์ที่เกิดขึ้น(Event Handler) ต้องโอเวอร์ไรต์เมธอดดังนี้

Event Class	Listener Interface	Listener Methods (Handlers)
ActionEvent	ActionListener	actionPerformed(ActionEvent)
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
WindowEvent	WindowListener	windowClosing(WindowEvent) windowOpened(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent)

ContainerEvent	ContainerListener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
MouseEvent	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseClicked(MouseEvent) mouseExited(MouseEvent) mouseEntered(MouseEvent)
KeyEvent	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)

ตัวอย่างที่ 13.2 การควบคุมการย่อขยายของวงกลมโดยไม่มี listener

```
import javax.swing.*.*;
import java.awt.*.*;
public class Ex13_2 extends JFrame {
    private JButton jbtEnlarge = new JButton("Enlarge");
    private JButton jbtShrink = new JButton("Shrink");
    private CirclePanel canvas = new CirclePanel();
    public Ex13_2() {
        JPanel panel = new JPanel();
        panel.add(jbtEnlarge);
        panel.add(jbtShrink);
        this.add(canvas, BorderLayout.CENTER);
        this.add(panel, BorderLayout.SOUTH);
    }
    public static void main(String[] args) {
        JFrame frame = new Ex13_2();
        frame.setTitle("Control Circle Without Event Handling");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 200);
        frame.setVisible(true);
    }
}
class CirclePanel extends JPanel {
    private int radius = 5;
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawOval(getWidth() / 2 - radius, getHeight() / 2 - radius,
            2 * radius, 2 * radius);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



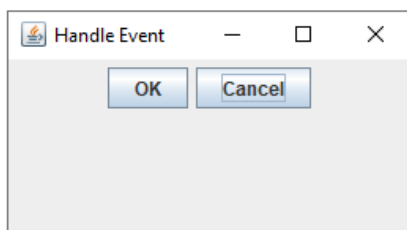
เราสามารถสร้างออบเจ็กต์ของคลาสประเภท Event Handler หรือ Listener ได้หลายรูปแบบดังนี้

1. กำหนดคลาสประเภท Listener เป็นคลาสภายนอกที่ใช้ในการจัดการเหตุการณ์ โดยคลาสประเภท Listener แบบคลาสภายนอกดังกล่าวมีการอิมพลิเมนต์หรือสืบทอดอินเทอร์เฟซ Listener ที่เกี่ยวข้องกับเหตุการณ์ที่ต้องการจัดการ เช่น ตัวอย่างที่ 13.3 คลาส OKListenerClass เป็นคลาสภายนอกที่สืบทอดอินเทอร์เฟซ ActionListener ที่จะทำให้หน้าที่เป็น Event Handler จัดการเหตุการณ์ที่เกี่ยวข้อง

ตัวอย่างที่ 13.3 กำหนดคลาสประเภท Listener เป็นคลาสภายนอกที่ใช้ในการจัดการเหตุการณ์

```
import javax.swing.*.*;
import java.awt.event.*;
public class Ex13_3 extends JFrame {
    public Ex13_3() {
        JButton jbtOK = new JButton("OK");
        JButton jbtCancel = new JButton("Cancel");
        JPanel panel = new JPanel();
        panel.add(jbtOK);
        panel.add(jbtCancel);
        add(panel);
        OKListenerClass listener1 = new OKListenerClass();
        CancelListenerClass listener2 = new CancelListenerClass();
        jbtOK.addActionListener(listener1);
        jbtCancel.addActionListener(listener2);
    }
    public static void main(String[] args) {
        JFrame frame = new Ex13_3();
        frame.setTitle("Handle Event");
        frame.setSize(200, 150);
        frame.setLocation(200, 100);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
class OKListenerClass implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("OK button clicked");
    }
}
class CancelListenerClass implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Cancel button clicked");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



OK button clicked
Cancel button clicked

2. กำหนดคลาสประเภท Listener เป็นคลาสภายใน (inner class) อยู่ในคลาสที่ใช้ในการจัดการเหตุการณ์

inner class หมายถึงคลาสที่เป็นสมาชิกของคลาสอื่น ข้อดีคือ สามารถใช้ **inner class** เพื่อให้โปรแกรมทำงานได้ง่ายขึ้น ตัว **inner class** สามารถอ้างถึงแอตทริบิวต์และเมธอดที่ประกาศไว้ใน **outer class** หรือคลาสภายนอกได้ ดังนั้นจึงไม่จำเป็นต้องส่งค่าอ้างอิงของ **outer class** มายังคอนสตรัคเตอร์ของ **inner class**

การคอมไพล์ของ **inner class** จะถูกคอมไพล์ไปในรูปแบบ **OuterClassName\$InnerClassName.class** เช่น **InnerClass** ที่อยู่ใน **OuterClass** จะถูกคอมไพล์ไปเป็น **OuterClass\$InnerClass.class** เราสามารถประกาศ **inner class** ในรูป **public, protected,** หรือ **private** นอกจากนี้ยังสามารถประกาศด้วยคีย์เวิร์ด **static** โดยที่ **static inner class** สามารถเข้าถึงได้โดยการใช้ชื่อของ **outer class**

การประกาศให้คลาสเป็น **static inner class** จะไม่สามารถเข้าถึงสมาชิกที่ไม่เป็น **static** ของ **outer class**

ตัวอย่างที่ 13.4

```
public class Ex13_4 {
    private int x;
    public void g() {
    }
    public Ex13_4() {
    }
    public static void main(String[] args) {
    }
    class InnerClass {
        public void f() {
            x++;
            g();
        }
    }
}
```

inner class ถูกออกแบบมาเพื่อสร้างวัตถุของ **listener** คลาส ไว้ภายใน GUI component เช่น **button** โดยตรง จะไม่ถูกใช้ร่วมกับ **application** ตัวอื่น ๆ

ตัวอย่างที่ 13.5 กำหนดคลาสที่เป็นคลาสภายในอยู่ในคลาสที่ใช้ในการจัดการเหตุการณ์ การควบคุมการย่อขยายของวงกลม โดยมี **listeners** เป็นตัวควบคุมการเกิดเหตุการณ์

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Ex13_5 extends JFrame {
    private JButton jbtEnlarge = new JButton("Enlarge");
    private JButton jbtShrink = new JButton("Shrink");
    private CirclePanel canvas = new CirclePanel();
    public Ex13_5() {
        JPanel panel = new JPanel();
        panel.add(jbtEnlarge);
        panel.add(jbtShrink);

        this.add(canvas, BorderLayout.CENTER);
        this.add(panel, BorderLayout.SOUTH);

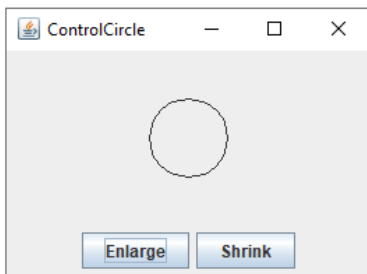
        jbtEnlarge.addActionListener(new EnlargeListener());
    }
    public static void main(String[] args) {
        JFrame frame = new Ex13_5();
        frame.setTitle("ControlCircle");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```

frame.setSize(200, 200);
frame.setVisible(true);
}
class EnlargeListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        canvas.enlarge();
    }
}
class CirclePanel extends JPanel {
    private int radius = 5;
    public void enlarge() {
        radius++;
        repaint();
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawOval(getWidth() / 2 - radius, getHeight() / 2 - radius,
            2 * radius, 2 * radius);
    }
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



ตัวอย่างที่ 13.6 กำหนดให้คลาสที่ใช้ในการจัดการเหตุการณ์อิมพลีเมนต์อินเทอร์เฟซประเภท **listeners** ที่สอดคล้องกัน และสร้างออบเจกต์ของคลาสดังกล่าวไว้ในคลาสเอง

```

import javax.swing.*;
import java.awt.event.*;
public class Ex13_6 extends JFrame {
    private JButton jbtNew = new JButton("New");
    private JButton jbtOpen = new JButton("Open");
    private JButton jbtSave = new JButton("Save");
    private JButton jbtPrint = new JButton("Print");
    public Ex13_6() {
        JPanel panel = new JPanel();
        panel.add(jbtNew);
        panel.add(jbtOpen);
        panel.add(jbtSave);
        panel.add(jbtPrint);
        add(panel);
        ButtonListener listener = new ButtonListener();
        jbtNew.addActionListener(listener);
        jbtOpen.addActionListener(listener);
        jbtSave.addActionListener(listener);
        jbtPrint.addActionListener(listener);
    }
    class ButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {

```



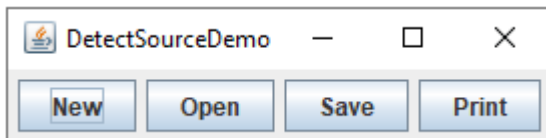
```

        if (e.getSource() == jbtNew)
            System.out.println("Process New");
        else if (e.getSource() == jbtOpen)
            System.out.println("Process Open");
        else if (e.getSource() == jbtSave)
            System.out.println("Process Save");
        else if (e.getSource() == jbtPrint)
            System.out.println("Process Print");
    }
}

public static void main(String[] args) {
    JFrame frame = new Ex13_6();
    frame.setTitle("DetectSourceDemo");
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



```

Process New
Process Open
Process Save
Process Print

```

3. กำหนดให้คลาสที่ใช้ในการจัดการเหตุการณ์อิมพลีเมนต์อินเทอร์เฟซประเภท Listener ที่สอดคล้องกันและสร้างออบเจ็กต์ของคลาสดังกล่าวภายในคลาสเอง

ตัวอย่างการเขียนโปรแกรมเพื่อสร้าง Listener Class อีกวิธีหนึ่ง โดยสร้าง listener แค่ 1 คลาสแล้วลงทะเบียนให้กับปุ่มหลังจากนั้นที่ตัว listener ให้ทำการเช็คค่าเหตุการณ์ที่เกิดขึ้นกับปุ่มใดโดยใช้ `e.getSource()` ซึ่งเมื่อได้ปุ่มที่เกิดเหตุการณ์แล้วก็ทำงานตามการทำงานของแต่ละปุ่ม ดังตัวอย่างด้านล่าง

ตัวอย่างที่ 13.7 กำหนดให้คลาสที่ใช้ในการจัดการเหตุการณ์อิมพลีเมนต์อินเทอร์เฟซประเภท Listener ที่สอดคล้องกันและสร้างออบเจ็กต์ของคลาสดังกล่าวภายในคลาสเอง

```

import javax.swing.*.*;
import java.awt.event.*;

public class Ex13_7 extends JFrame implements ActionListener {
    private JButton jbtNew = new JButton("New");
    private JButton jbtOpen = new JButton("Open");
    private JButton jbtSave = new JButton("Save");
    private JButton jbtPrint = new JButton("Print");
    public Ex13_7() {
        JPanel panel = new JPanel();
        panel.add(jbtNew);
        panel.add(jbtOpen);
        panel.add(jbtSave);
        panel.add(jbtPrint);
        add(panel);
        jbtNew.addActionListener(this);
        jbtOpen.addActionListener(this);
        jbtSave.addActionListener(this);
        jbtPrint.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {

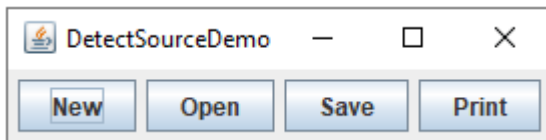
```

```

    if (e.getSource() == jbtNew)
        System.out.println("Process New");
    else if (e.getSource() == jbtOpen)
        System.out.println("Process Open");
    else if (e.getSource() == jbtSave)
        System.out.println("Process Save");
    else if (e.getSource() == jbtPrint)
        System.out.println("Process Print");
}
public static void main(String[] args) {
    JFrame frame = new Ex13_7();
    frame.setTitle("FrameAsListenerDemo");
    frame.setLocationRelativeTo(null);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



Process New
Process Open
Process Save
Process Print

จากตัวอย่างด้านบนจะพบว่าเมื่อเขียน **code** ในลักษณะดังกล่าวต้องทำการเขียนหรือลงทะเบียน listener class ให้ตรวจจับเหตุการณ์ที่เกิดขึ้นกับเฟรมโดยการอิมพลีเมนต์ **ActionListener** ที่เฟรมโดยตรง

4. กำหนดให้คลาสที่ใช้ในการจัดการเหตุการณ์เป็นคลาสภายในเมธอด(คลาสประเภท anonymous)

Anonymous inner class จะต้องสืบทอดข้อมูลจาก **superclass** หรืออิมพลีเมนต์ **interface** แต่ไม่สามารถมีการเขียนประโยค **extends** หรือ **implements** ได้อย่างแน่ชัด แต่ **anonymous inner class** จะต้องอิมพลีเมนต์ **abstract methods** ทั้งหมดใน **superclass** หรือใน **interface**

Anonymous inner class จะต้องใช้ **constructor** แบบไม่มีพารามิเตอร์จาก **superclass** สำหรับสร้าง **instance** ถ้า **anonymous inner class** ทำการอิมพลีเมนต์อินเตอร์เฟซคอนสตรักเตอร์ที่ใช้คือ **Object()**.

การใช้ **inner class listeners** สามารถที่จะทำให้สั้นลงได้ด้วยการใช้ **anonymous inner class** ซึ่งก็คือ **inner class** ที่ไม่มีชื่อโดยที่ภายในตัวคลาสนี้จะประกาศ **inner class** และสร้าง **instance** ของคลาสใน 1 ขั้นตอน โดยถูกประกาศในรูปแบบดังนี้

```

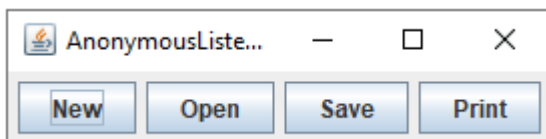
new SuperClassName/InterfaceName() {
    // Implement or override methods in superclass or interface
    // Other methods if necessary
}

```

ตัวอย่างที่ 13.8 กำหนดคลาสภายในเมธอด(คลาสประเภท anonymous)

```
import javax.swing.*;
import java.awt.event.*;
public class Ex13_8 extends JFrame {
    public Ex13_8() {
        JButton jbtNew = new JButton("New");
        JButton jbtOpen = new JButton("Open");
        JButton jbtSave = new JButton("Save");
        JButton jbtPrint = new JButton("Print");
        JPanel panel = new JPanel();
        panel.add(jbtNew);
        panel.add(jbtOpen);
        panel.add(jbtSave);
        panel.add(jbtPrint);
        add(panel);
        jbtNew.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Process New");
            }
        });
        jbtOpen.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Process Open");
            }
        });
        jbtSave.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Process Save");
            }
        });
        jbtPrint.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Process Print");
            }
        });
    }
    public static void main(String[] args) {
        JFrame frame = new Ex13_8();
        frame.setTitle("AnonymousListenerDemo");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



```
Process New
Process Open
Process Save
Process Print
```

13.5 การควบคุมเหตุการณ์ที่เกิดขึ้นกับคอมโพเนนต์ประเภทต่างๆ

13.5.1 การควบคุมเหตุการณ์เมื่อเกิด ActionEvent

ActionEvent อยู่ในแพ็คเกจ `java.awt.event.ActionEvent` จะถูกสร้างขึ้นในกรณีที่มีเหตุการณ์เกิดขึ้นในโปรแกรม GUI ดังนี้)เมื่อมีการคลิกเมาส์บนปุ่ม-Button) เมื่อมีการป้อนคีย์ Enter ใน TextField เมื่อมีการเลือกเมนูใน MenuItem เมื่อมีการกด double click ใน List

เมธอดที่สำคัญของคลาส ActionEvent มีดังนี้

`String getActionCommand()` จะส่งชื่อคำสั่งที่เกิดขึ้นจาก ActionEvent

`int getModifier()` จะส่งสถานะของคีย์)Modifier) คีย์ alt, ctrl, Meta, และ shift ที่เกิดขึ้นจากออบเจ็กต์ของคลาส ActionEvent

13.5.2 การควบคุมเหตุการณ์ที่เกิดขึ้นกับ Window

WindowEvent จะถูกสร้างขึ้นในกรณีที่มีเหตุการณ์เกิดขึ้นเป็นออบเจ็กต์ของคลาสประเภท Window โดยมีรูปแบบต่าง ๆ ดังนี้ Opened, closed, closing, iconified, deiconified, activated, deactivated ตัวอย่างการเขียนโปรแกรมเพื่อจัดการกับเหตุการณ์ที่เกิดขึ้นกับ Window คือ window opened, closing, closed, activated, deactivated, iconified, and deiconified โดยในโปรแกรมจะสร้าง frame และคอยฟังเหตุการณ์ที่เกิดขึ้นกับ window และแสดงข้อความบ่งบอกถึงเหตุการณ์ที่เกิดขึ้น แสดงดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 13.9

```
import java.awt.event.*;
import javax.swing.JFrame;
public class Ex13_9 extends JFrame {
    public static void main(String[] args) {
        Ex13_9 frame = new Ex13_9();
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("TestWindowEvent");
        frame.setSize(220, 80);
        frame.setVisible(true);
    }
    public Ex13_9() {
        addWindowListener(new WindowListener() {
            public void windowDeiconified(WindowEvent event) {
                System.out.println("Window deiconified");
            }
            public void windowIconified(WindowEvent event) {
                System.out.println("Window iconified");
            }
            public void windowActivated(WindowEvent event) {
                System.out.println("Window activated");
            }
            public void windowDeactivated(WindowEvent event) {
                System.out.println("Window deactivated");
            }
            public void windowOpened(WindowEvent event) {
                System.out.println("Window opened");
            }
            public void windowClosing(WindowEvent event) {
                System.out.println("Window closing");
            }
            public void windowClosed(WindowEvent event) {
                System.out.println("Window closed");
            }
        });
    }
}
```

```
}
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



Window activated
 Window opened
 Window deactivated
 Window activated
 Window closing

13.5.3 การควบคุมเหตุการณ์ที่เกิดขึ้นกับเมาส์

การควบคุมเหตุการณ์ที่เกิดขึ้นกับเมาส์โดยการใช้คลาส `MouseEvent` จะถูกสร้างขึ้นในกรณีที่ใช้เมาส์เพื่อติดต่อกับผู้ใช้โดยมีเหตุการณ์ที่เกิดขึ้นได้ดังนี้ dragged moved clicked entered exited pressed released

เมธอดที่สำคัญของคลาส `MouseEvent` มีดังนี้

เมธอด	รายละเอียด
<code>int getX()</code>	จะส่งตำแหน่งพิกัดของเมาส์แกน x ที่มีชนิดข้อมูลเป็น int คืนมา
<code>int getY()</code>	จะส่งตำแหน่งพิกัดของเมาส์แกน y ที่มีชนิดข้อมูลเป็น int คืนมา
<code>Point getPoint()</code>) จะส่งตำแหน่งพิกัด x, y ของเมาส์คืนมา โดยมีชนิดข้อมูลเป็นออบเจ็กต์ของคลาส (Point)
<code>int getClickCount()</code>	จะส่งจำนวนครั้งของการคลิกเมาส์คืนมา

การควบคุมเหตุการณ์ที่เกิดขึ้นกับเมาส์ โดยการใช้คลาส `Handling Mouse Events`

การตรวจจับเหตุการณ์ (Event) ต่างๆ ที่เกิดจากการใช้เมาส์สามารถทำได้โดยใช้เมธอดของอินเทอร์เฟซที่ชื่อ `MouseListener` และ `MouseMotionListener`

`MouseListener` จะคอยรับฟังเหตุการณ์เมื่อเมาส์ถูกกด (**pressed**) เมาส์ถูกปล่อย (**released**) เมาส์เคลื่อนไปยังคอมโพเนนต์ (**entered**) เมาส์ออกจากคอมโพเนนต์ (**exited**) เมาส์คลิกที่คอมโพเนนต์ (**clicked**)

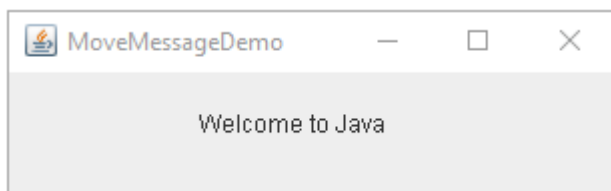
`MouseMotionListener` จะคอยรับฟังเหตุการณ์เมื่อเมาส์มีการ **dragging** หรือการเคลื่อนเมาส์

ตัวอย่างที่ 13.10 การควบคุมเหตุการณ์ที่เกิดขึ้นกับเมาส์

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Ex13_10 extends JFrame {
    public Ex13_10() {
        MovableMessagePanel p = new MovableMessagePanel("Welcome to Java");
        add(p);
    }
    public static void main(String[] args) {
        Ex13_10 frame = new Ex13_10();
        frame.setTitle("MoveMessageDemo");
        frame.setSize(200, 100);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    static class MovableMessagePanel extends JPanel {
        private String message = "Welcome to Java";
        private int x = 20;
        private int y = 20;
        public MovableMessagePanel(String s) {
            message = s;
            addMouseListener(new MouseMotionListener() {
                @Override
                public void mouseDragged(MouseEvent e) {
                    x = e.getX();
                    y = e.getY();
                    repaint();
                }

                @Override
                public void mouseMoved(MouseEvent e) {
                }
            });
        }
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawString(message, x, y);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ



13.5.4 การควบคุมเหตุการณ์ที่เกิดขึ้นกับคีย์บอร์ด

การดักจับเหตุการณ์ที่เกิดขึ้นจากการใช้คีย์บอร์ด (Keyboard) จะใช้ `KeyListener` เป็นตัวดักจับเหตุการณ์ที่เกิดขึ้นซึ่งคลาสใดที่มีการอิมพลิเมนต์คลาส `KeyListener` จะต้องกำหนดขั้นตอนการประมวลผลให้กับเมธอดต่อไปนี้

เมธอด	รายละเอียด
<code>keyPressed()</code> <code>KeyEvent e</code>	เป็นการตรวจจับการกดคีย์ใดๆ
<code>keyTyped()</code> <code>KeyEvent e</code>	เป็นการตรวจจับการกดคีย์ที่ไม่ใช่ function key เช่น arrow key, Home, End, Page Up, Page Down, Num Lock, Print Screen, Scroll Lock, Caps Lock และ Pause
<code>keyReleased()</code> <code>KeyEvent e</code>	เป็นการตรวจจับการปล่อยคีย์
<code>getKeyChar()</code> <code>getKeyCode()</code>	จะคืนค่าคีย์ออกมา

ค่าคีย์ต่าง ๆ มีดังนี้

คีย์	สัญลักษณ์ที่ใช้
Home	<code>VK_HOME</code>
End	<code>VK_END</code>
Page Up	<code>VK_PGUP</code>
Page Down	<code>VK_PGDN</code>

ตัวอย่างที่ 13.11 การควบคุมเหตุการณ์ที่เกิดขึ้นกับคีย์บอร์ด

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Ex13_11 extends JFrame {
    private KeyboardPanel keyboardPanel = new KeyboardPanel();
    public Ex13_11() {
        add(keyboardPanel);
        keyboardPanel.setFocusable(true);
    }
    public static void main(String[] args) {
        Ex13_11 frame = new Ex13_11();
        frame.setTitle("KeyEventDemo");
        frame.setSize(300, 300);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    static class KeyboardPanel extends JPanel {
        private int x = 100;
        private int y = 100;
        private char keyChar = 'A'; // Default key

        public KeyboardPanel() {
            addKeyListener(new KeyAdapter() {
                @Override
                public void keyPressed(KeyEvent e) {
                    switch (e.getKeyCode()) {
                        case KeyEvent.VK_DOWN: y += 10; break;
                        case KeyEvent.VK_UP: y -= 10; break;
                        case KeyEvent.VK_LEFT: x -= 10; break;
                        case KeyEvent.VK_RIGHT: x += 10; break;
                        default: keyChar = e.getKeyChar();
                    }
                }
            });
        }
    }
}
```

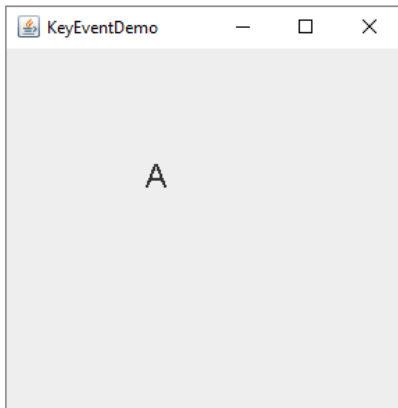
```

        repaint();
    }
});
}
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    g.setFont(new Font("TimesRoman", Font.PLAIN, 24));
    g.drawString(String.valueOf(keyChar), x, y);
}
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



13.5.5 การควบคุมเหตุการณ์ที่เกิดขึ้นกับ Timer

คอมพิวเตอร์บางตัวที่ไม่ใช่คอมพิวเตอร์ของส่วนประกอบกราฟิก(GUI) เช่น **Timer** หรือคลาสที่เกี่ยวข้องเวลาสามารถที่จะกระตุ้นให้เกิดเหตุการณ์ได้ คลาสที่สำคัญคือ **javax.swing.Timer** ซึ่งเป็นแหล่งของการเกิดเหตุการณ์ของคลาส **ActionEvent** เราสามารถใช้คลาส **Timer** สำหรับควบคุมการเกิดภาพเคลื่อนไหวได้ เช่น ต้องการควบคุมการเคลื่อนที่ของ **message** ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 13.12 การควบคุมเหตุการณ์ที่เกิดขึ้นกับ Timer

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Ex13_12 extends JFrame {
    public Ex13_12() {
        this.setLayout(new GridLayout(2, 1));
        add(new MovingMessagePanel("message 1 moving?"));
        add(new MovingMessagePanel("message 2 moving?"));
    }
    public static void main(String[] args) {
        Ex13_12 frame = new Ex13_12();
        frame.setTitle("AnimationDemo");
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(280, 100);
        frame.setVisible(true);
    }
    static class MovingMessagePanel extends JPanel {
        private String message = "Welcome to Java";
        private int xCoordinate = 0;
    }
}

```



```

private int yCoordinate = 20;
private Timer timer = new Timer(1000, new TimerListener());

public MovingMessagePanel(String message) {
    this.message = message;
    timer.start();
    this.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            int delay = timer.getDelay();
            if (e.getButton() == MouseEvent.BUTTON1)
                timer.setDelay(delay > 10 ? delay - 10 : 0);
            else if (e.getButton() == MouseEvent.BUTTON3)
                timer.setDelay(delay < 50000 ? delay + 10 : 50000);
        }
    });
}

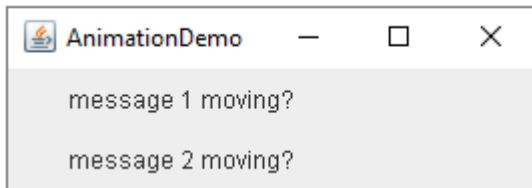
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    if (xCoordinate > getWidth()) {
        xCoordinate = -20;
    }
    xCoordinate += 5;
    g.drawString(message, xCoordinate, yCoordinate);
}

class TimerListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        repaint();
    }
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



13.6 ตัวอย่างการประยุกต์การควบคุมเหตุการณ์โดยใช้คลาส Timer

การใช้คลาส **StillClock** สำหรับแสดงเวลาปัจจุบัน นาฬิกาดังกล่าวยังไม่สามารถเคลื่อนที่ได้ ถ้าเราต้องการให้คลาสสามารถแสดงเวลาปัจจุบัน การแสดงดังกล่าวสามารถทำได้โดยใช้ **timer** สำหรับควบคุมการ **repaint** นาฬิกา โดยขั้นตอนของการเพิ่มเหตุการณ์ที่ทำให้นาฬิกาแต่ละเรือนเดินได้ประกอบด้วยขั้นตอนหลัก ๆ ดังนี้

1. ทำการดึงเวลาจริงจากเครื่อง
2. นำเวลาที่ได้ทำการคำนวณตำแหน่งจริงบนหน้าจอ
3. สร้างวัตถุจากคลาส **Timer** และคอยตรวจจับการเกิดเหตุการณ์กับวัตถุ **timer** โดยการลงทะเบียนการดักจับ

เหตุการณ์ให้กับ **timer** โดยใช้คำสั่ง

```
Timer timer = new Timer(1000, new TimerListener());
timer.start();
```

4. วัตถุที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์และคอยตอบสนองเหตุการณ์ที่เกิดขึ้นคือวัตถุที่สร้างจากคลาสที่ทำการอิมพลีเมนต์อินเตอร์เฟซ **ActionListener** โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการโอเวอร์ไรด์เมธอด **actionPerformed()** ดังนี้

```
private class TimerListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        clock.setCurrentTime();
        clock.repaint();
    }
}
```

5. เมื่อเวลาผ่านไปทุก ๆ 1000 ms ให้ **repaint** ตำแหน่งของเข็มใหม่โดยใช้คำสั่ง **repaint()** ซึ่งคำสั่งดังกล่าวจะทำการเรียกคำสั่ง **paintComponent()** สำหรับการวาดรูปใหม่

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    int clockRadius =
        (int) (Math.min(getWidth(), getHeight()) * 0.8 * 0.5);
    int xCenter = getWidth() / 2;
    int yCenter = getHeight() / 2;
    g.setColor(Color.black);
    g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
        2 * clockRadius, 2 * clockRadius);
    g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
    g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
    g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
    g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);
    int sLength = (int) (clockRadius * 0.8);
    int xSecond = (int) (xCenter + sLength *
        Math.sin(second * (2 * Math.PI / 60)));
    int ySecond = (int) (yCenter - sLength *
        Math.cos(second * (2 * Math.PI / 60)));
    g.setColor(Color.red);
    g.drawLine(xCenter, yCenter, xSecond, ySecond);
    int mLength = (int) (clockRadius * 0.65);
    int xMinute = (int) (xCenter + mLength *
        Math.sin(minute * (2 * Math.PI / 60)));
    int yMinute = (int) (yCenter - mLength *
        Math.cos(minute * (2 * Math.PI / 60)));
    g.setColor(Color.blue);
    g.drawLine(xCenter, yCenter, xMinute, yMinute);
    int hLength = (int) (clockRadius * 0.5);
    int xHour = (int) (xCenter + hLength *
        Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
    int yHour = (int) (yCenter - hLength *
        Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
}
```

```

        Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
g.setColor(Color.green);
g.drawLine(xCenter, yCenter, xHour, yHour);
}

```

ตัวอย่างที่ 13.13 ตัวอย่างการประยุกต์การควบคุมเหตุการณ์โดยใช้คลาส Timer

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
public class Ex13_13 extends JFrame {
    private StillClock clock = new StillClock();
    public Ex13_13() {
        add(clock);
        javax.swing.Timer timer = new javax.swing.Timer(1000, new TimerListener());
        timer.start();
    }
    private class TimerListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            clock.setCurrentTime();
            clock.repaint();
        }
    }
    public static void main(String[] args) {
        JFrame frame = new Ex13_13();
        frame.setTitle("Clock Animation");
        frame.setSize(200, 200);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
class StillClock extends JPanel {
    private int hour;
    private int minute;
    private int second;
    public StillClock() {
        setCurrentTime();
    }
    public StillClock(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }
    public int getHour() {
        return hour;
    }
    public void setHour(int hour) {
        this.hour = hour;
        repaint();
    }
    public int getMinute() {
        return minute;
    }
    public void setMinute(int minute) {
        this.minute = minute;
        repaint();
    }
    public int getSecond() {
        return second;
    }
    public void setSecond(int second) {
        this.second = second;
        repaint();
    }
}

```

```

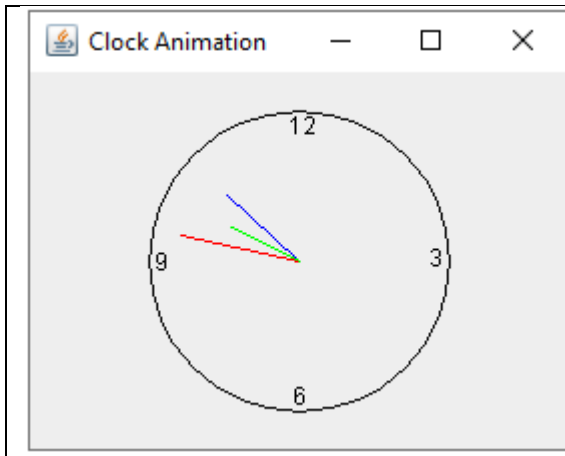
}
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    int clockRadius =
        (int) (Math.min(getWidth(), getHeight()) * 0.8 * 0.5);
    int xCenter = getWidth() / 2;
    int yCenter = getHeight() / 2;
    g.setColor(Color.black);
    g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
        2 * clockRadius, 2 * clockRadius);
    g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
    g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
    g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
    g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);
    int sLength = (int) (clockRadius * 0.8);
    int xSecond = (int) (xCenter + sLength *
        Math.sin(second * (2 * Math.PI / 60)));
    int ySecond = (int) (yCenter - sLength *
        Math.cos(second * (2 * Math.PI / 60)));
    g.setColor(Color.red);
    g.drawLine(xCenter, yCenter, xSecond, ySecond);
    int mLength = (int) (clockRadius * 0.65);
    int xMinute = (int) (xCenter + mLength *
        Math.sin(minute * (2 * Math.PI / 60)));
    int yMinute = (int) (yCenter - mLength *
        Math.cos(minute * (2 * Math.PI / 60)));
    g.setColor(Color.blue);
    g.drawLine(xCenter, yCenter, xMinute, yMinute);
    int hLength = (int) (clockRadius * 0.5);
    int xHour = (int) (xCenter + hLength *
        Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
    int yHour = (int) (yCenter - hLength *
        Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
    g.setColor(Color.green);
    g.drawLine(xCenter, yCenter, xHour, yHour);
}

public void setCurrentTime() {
    Calendar calendar = new GregorianCalendar();
    this.hour = calendar.get(Calendar.HOUR_OF_DAY);
    this.minute = calendar.get(Calendar.MINUTE);
    this.second = calendar.get(Calendar.SECOND);
}

@Override
public Dimension getPreferredSize() {
    return new Dimension(200, 200);
}
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



13.7 การจัดการกับข้อผิดพลาด (Exception Handling)

โปรแกรมภาษาจาวาแบ่งข้อผิดพลาดที่อาจเกิดขึ้นขณะรันโปรแกรมเป็นสองประเภทคือ

1. Error เป็นข้อผิดพลาดที่ไม่สามารถแก้ไขและจัดการได้ เช่น `OutOfMemoryError`
2. Exception เป็นข้อผิดพลาดที่สามารถแก้ไขหรือจัดการได้ เช่น `ArrayIndexOutOfBoundsException`, `FileNotFoundException`, `ArithmeticException`

ข้อผิดพลาดในภาษาจาวาจะกำหนดเป็นอ็อปเจ็คของคลาสต่างๆ โดยมีคลาส `Throwable` เป็นคลาสราก

13.7.1 คลาสประเภท Exception

Exception เป็นข้อผิดพลาดที่เกิดในขณะรันโปรแกรมภาษาจาวาโดยคลาสประเภท `Exception` แบ่งออกเป็นสองประเภท คือ

1. `RuntimeException` เป็นข้อผิดพลาดที่อาจหลีกเลี่ยงได้หากมีการเขียนโปรแกรมที่ถูกต้อง
2. `IOException` เป็นข้อผิดพลาดที่ภาษาจาวากำหนดให้ต้องมีการจัดการหากมีการเรียกใช้เมธอดที่อาจเกิดข้อผิดพลาดประเภทนี้ได้

ตัวอย่างของคลาสประเภท `Exception` มีตัวอย่างดังนี้ `ArithmeticException`

`ArrayIndexOutOfBoundsException` `EOFException` `FileNotFoundException` `InterruptedException`
`IOException` `NullPointerException` `NumberFormatException`

ตัวอย่างที่ 13.14 ตัวอย่างการเขียนโปรแกรมที่ไม่มีการจัดการกับข้อผิดพลาด (Exception Handling)

```
public class Ex13_14 {
    public static void main(String args[]) {
        System.out.println(args[1]);
        System.out.println("Test Exception");
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
    at ExceptionDemo.main(ExceptionDemo.java:3)
```

13.7.2 คำสั่ง try...catch

ภาษาจาวามีคีย์เวิร์ด **try** ที่เป็นคำสั่งที่ใช้ในการจัดการกับเมธอดหรือคำสั่งที่อาจเกิดข้อผิดพลาดซึ่งจะส่งวัตถุประเภท **Exception** ในขณะรันโปรแกรม โดยมีรูปแบบในการเขียนคำสั่ง **try...catch** ดังนี้

```
try {
    [statements]
}
catch(ExceptionType argumentName){
    [statements]
}
```

โปรแกรมภาษาจาวาจะส่งงานชุดคำสั่งที่อยู่ในบล็อกทีละคำสั่ง และหากเกิดข้อผิดพลาดขึ้นในคำสั่งประเภทใดก็จะมีการส่งออปเจกของข้อผิดพลาดประเภท **Exception** นั้นขึ้นมา ในกรณีที่ต้องการจัดการกับข้อผิดพลาดที่เกิดขึ้น โปรแกรมจะต้องมีชุดคำสั่งอยู่ในบล็อกของคีย์เวิร์ด **catch** ที่จะระบุชนิดของออปเจกในคลาสประเภท **Exception** ที่ต้องการจัดการ

ตัวอย่างที่ 13.15 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
public class Ex13_15 {
    public static void main(String args[]) {
        try {
            System.out.println(args[1]);
        }
        catch (ArrayIndexOutOfBoundsException ex) {
            System.out.println("Invalid Arguments");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ
Invalid Arguments

ตัวอย่างที่ 13.16 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
class Ex13_16 {
    public static void main (String args[]) {
        char str = '\0';
        System.out.println ("Enter 1 charater:");
        try {
            str = (char) System.in.read();
        }
        catch (Exception e) {
            System.out.println ("Error:"+e.toString());
        }
        System.out.println("Your input data is "+str);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ
Enter 1 charater:
A
Your input data is A

การจัดการกับข้อผิดพลาดหลายๆประเภท

โปรแกรมภาษาจาวาสามารถจะมีชุดคำสั่งของบล็อก catch ได้มากกว่าหนึ่งชุดสำหรับในแต่ละบล็อกคำสั่ง try ชนิดของออปเจ็คประเภท Exception ที่อยู่ในชุดคำสั่งของบล็อก catch จะต้องเรียงตามลำดับการสืบทอดในกรณีที่มีข้อผิดพลาดเกิดขึ้น ภาษาจาวาจะพิจารณาว่าเป็นข้อผิดพลาดชนิดใด ซึ่งการที่จะจัดการกับออปเจ็คประเภท Exception นั้นจะพิจารณาจากคลาสที่มีการสืบทอดตามลำดับชั้น

ตัวอย่างที่ 13.17 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
public class Ex13_17 {
    public static void main(String args[]) {
        try {
            int i = 0;
            System.out.println(4 / i);
        } catch (ArithmeticException ex) {
            System.out.println(ex.toString());
        } catch (NumberFormatException ex) {
            System.out.println("Invalid numeric format");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
java.lang.ArithmeticException: / by zero
```

บล็อก finally

ภาษาจาวามีคีย์เวิร์ด finally ที่จะมีชุดคำสั่งอยู่ในบล็อกเพื่อระบุให้โปรแกรมทำชุดคำสั่งดังกล่าวหลังจากสิ้นสุดการทำงานของชุดคำสั่งในบล็อก try หรือ catch ภาษาจาวาจะทำชุดคำสั่งในบล็อก finally เสมอ แม้ว่าจะมีคำสั่ง return ในบล็อก try หรือ catch ก่อนก็ตาม กรณีเดียวที่จะไม่ทำชุดคำสั่งในบล็อก finally คือมีคำสั่ง System.exit();

ตัวอย่างที่ 13.18 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
public class Ex13_18 {
    public static void main(String args[]) {
        try {
            System.out.println(args[1]);
            System.out.println("Hello");
        } catch (ArrayIndexOutOfBoundsException ex) {
            System.out.println("No arguments");
        } finally {
            System.out.println("finally");
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
No arguments
finally
```

13.7.3 การจัดการกับเมธอดที่ส่งออปป็เจ็คประเภท Exception

เราสามารถจะจัดการกับออปป็เจ็คของ Exception โดยใช้คลาสที่เป็น superclass ของ Exception นั้นได้ อาทิ เช่น เราสามารถจัดการกับ FileNotFoundException โดยใช้ IOException หรือ Exception แทนได้ การจัดการกับ Exception มีสองแบบ คือ

1. ใช้คำสั่ง try/catch

2. ใช้คำสั่ง throws ในการประกาศเมธอดที่จะมีการเรียกใช้เมธอดใดๆที่อาจส่งออปป็เจ็คประเภท Exception

คำสั่ง throws

รูปแบบการใช้คำสั่ง throws มีดังนี้

```
[modifier] return_type methodName([arguments]) throws ExceptionType[,ExceptionType2]
{
}
```

ตัวอย่างที่ 13.19 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
public void openfile(String s) throws FileNotFoundException {
    ...
}
```

เมธอดใดๆสามารถที่จะจัดการกับ Exception โดยใช้คำสั่ง throws ได้มากกว่าหนึ่งประเภท

ตัวอย่างที่ 13.20 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
public void openFile(String s) throws FileNotFoundException,UnknownHostException {
    ...
}
```

กรณีที่มีการใช้คำสั่ง throws ส่งต่อไปเรื่อยๆ แล้วเมธอด main() ซึ่งเรียกใช้เมธอดสุดท้ายที่ใช้คำสั่ง throws ไม่มีการจัดการกับออปป็เจ็คประเภท Exception ดังกล่าว โปรแกรมจะเกิดข้อผิดพลาดในขั้นตอนการรันโปรแกรม เมื่อมีข้อผิดพลาดของออปป็เจ็คประเภท Exception ดังกล่าวเกิดขึ้น

ตัวอย่างที่ 13.21 ตัวอย่างการจัดการกับข้อผิดพลาด (Exception Handling)

```
public class Ex13_18 {
    public static void main(String args[]) {
        Ex13_18 ex1 = new Ex13_18 ();
        ex1.method1();
    }
}
```



```

    }
    public void method1() throws ArithmeticException {
        method2();
    }
    public void method2() throws ArithmeticException {
        System.out.println(2/0);
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

Exception in thread "main" java.lang.ArithmeticException: / by zero
    at ExceptionDemo1.method2 (Ex13_18.java:10)
    at ExceptionDemo1.method1 (Ex13_18.java:7)
    at ExceptionDemo1.main (Ex13_18.java:4)

```

13.7.3 กฎของการกำหนดเมธอดแบบ overridden

เมธอดแบบ **overridden** จะไม่อนุญาตให้มีการจัดการออบเจ็กต์ประเภท **Exception** โดยใช้คำสั่ง **throws** มากกว่าที่เมธอดเดิมจัดการอยู่ได้

ตัวอย่างที่ 13.22 ตัวอย่างโปรแกรมที่มีเมธอดแบบ **overridden** ที่ถูกต้อง

```

import java.io.*;
public class Parent {
    public void myMethods() throws IOException { }
}
class OverrideException extends Parent{
    public void myMethods() throws IOException {
        new FileInputStream("temp.txt");
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

ตัวอย่างที่ 13.23 ตัวอย่างโปรแกรมที่มีเมธอดแบบ **overridden** ที่ไม่ถูกต้อง

```

import java.io.*;
public class Parent {
    public void myMethods() throws FileNotFoundException { }
}
class OverrideException extends Parent{
    public void myMethods() throws FileNotFoundException, IOException {
        new FileInputStream("temp.txt");
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

13.7.4 การสร้างคลาสประเภท Exception ขึ้นใหม่

การสร้างคลาสประเภท Exception ขึ้นมาใหม่ สามารถทำได้โดยนิยามคลาสใดๆ ให้สืบทอดมาจากคลาสที่ชื่อ Exception โดยทั่วไปคลาสที่ชื่อ Exception จะมี constructor สองรูปแบบคือ public Exception() และ public Exception(String s) ดังนั้นคลาสที่สืบทอดมาจากคลาสที่ชื่อ Exception ควรจะมี constructor ทั้งสองแบบ โดยรูปแบบหนึ่งจะมี argument ที่มีชนิดข้อมูลเป็น String และมีคำสั่งแรกใน constructor เป็นคำสั่ง super(s);

ตัวอย่างที่ 13.23 ตัวอย่างคลาสประเภท Exception ที่กำหนดขึ้นใหม่

```
public class MyOwnException extends Exception {
    public MyOwnException (String s) {
        super(s);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

13.7.5 การเขียนเมธอดเพื่อส่งออปปเจ็คประเภท Exception

เมธอดที่ต้องการส่งออปปเจ็คประเภท Exception เมื่อเกิดข้อผิดพลาดขึ้นในคำสั่งใด จะต้องเรียกใช้คำสั่งที่ชื่อ throw เพื่อจะสร้างออปปเจ็คของคลาสประเภท Exception ขึ้นมา

รูปแบบ throw new ExceptionType([arguments])

นอกจากนี้คำสั่งประกาศเมธอดนั้นจะต้องมีคำสั่ง throws เพื่อกำหนดให้คำสั่งในเมธอดอื่นๆ ที่เรียกใช้เมธอดนี้ต้องเขียนคำสั่งในการจัดการกับข้อผิดพลาดนี้

ตัวอย่างที่ 13.23 ตัวอย่างคลาสประเภท Exception ที่กำหนดขึ้นใหม่

```
import java.io.*;
class MyOwnException extends Exception {
    public MyOwnException (String s) {
        super(s);
    }
}
class FileHandler {
    public static void openFile(String filename) throws MyOwnException {
        File f = new File(filename);
        if (!f.exists()) {
            throw new MyOwnException("File Not Found");
        }
    }
}
public class FileOpener {
    public static void main (String args[]) {
        try {
            FileHandler.openFile(args[0]);
            System.out.println("Open successful");
        } catch (MyOwnException ex) {
            System.err.println(ex);
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

13.8 บทสรุป

การเขียนโปรแกรมในรูปแบบการเขียนโปรแกรมเชิงโครงสร้าง (Procedural programming) จะประมวลผลโปรแกรมตามลำดับของการเขียนที่เรียกว่า procedural order ส่วนการเขียนโปรแกรมในรูปแบบ Event-driven programming จะประมวลผลเมื่อเกิดเหตุการณ์เกิดขึ้น ผู้พัฒนาโปรแกรมสามารถจัดการการทำงานของโปรแกรมตามที่ต้องการเมื่อมีเหตุการณ์เกิดขึ้น เช่น เมื่อคลิกปุ่มแล้วโปรแกรมมีการคำนวณต่าง ๆ หรือเมื่อลากเมาส์แล้วโปรแกรมมีการวาดรูปบนหน้าจอตามที่ผู้ใช้งานกำหนดหรือแม้กระทั่งเมื่อผู้ใช้พิมพ์ข้อความผ่านคีย์บอร์ดแล้วโปรแกรมมีการแสดงข้อมูลบนหน้าจอ จากตัวอย่างดังกล่าวจะเป็นการเขียนโปรแกรมที่ประมวลผลเมื่อเหตุการณ์เกิดขึ้น โดยเหตุการณ์(Event) เป็นสถานการณ์ที่เกิดขึ้นในขณะรันโปรแกรม เช่น การใช้เมาส์หรือคีย์บอร์ดติดต่อกับโปรแกรม GUI การเกิดเหตุการณ์ในโปรแกรมภาษาจาวาจะเป็นการสร้างวัตถุของคลาสประเภท Event ชนิดต่าง ๆ ขึ้นมาตามประเภทของเหตุการณ์ เช่น เมื่อเลื่อนเมาส์ในเฟรมจะเกิดวัตถุของคลาส MouseEvent ขึ้นมา เมื่อปิดเฟรมจะเกิดวัตถุของคลาส WindowEvent ขึ้นมา เมื่อกดปุ่มที่อยู่ในเฟรมจะเกิดวัตถุของคลาส ActionEvent ขึ้นมา เมื่อพิมพ์ข้อความใน TextField จะเกิดวัตถุของคลาส KeyEvent ขึ้นมา หรือเหตุการณ์ที่เกิดกับ OS เช่น เหตุการณ์ของ timer

องค์ประกอบของเหตุการณ์ประกอบด้วย event คือวัตถุที่เกิดขึ้นตามประเภทของเหตุการณ์ Event Source คือส่วนที่ทำให้เกิดเหตุการณ์ และ Event Handler คือวัตถุที่ทำหน้าที่จัดการกับเหตุการณ์ที่เกิดขึ้นโดยมีเมธอดที่จะรับวัตถุชนิด Event ดังกล่าวและมีคำสั่งในการจัดการกับเหตุการณ์เพื่อโต้ตอบกับผู้ใช้

ภาษาจาวามีวิธีการจัดการกับเหตุการณ์ที่เรียกว่า Delegation Model โดยจะมีหลักการดังนี้

1. ผู้ใช้เป็นผู้กระตุ้นให้เกิดเหตุการณ์กับวัตถุประเภทแหล่งกำเนิดเหตุการณ์ (Event Source) เช่น ปุ่ม (Button) ช่องกรอกข้อมูล (Text Field)
2. วัตถุของส่วนประกอบกราฟิกใด ๆ สามารถเป็นวัตถุประเภทของแหล่งกำเนิดเหตุการณ์ได้ เช่น ออบเจกต์ของคลาส Button สามารถเป็นแหล่งกำเนิดเหตุการณ์ของเหตุการณ์ประเภท ActionEvent ได้
3. คลาสใด ๆ ที่จะใช้ในการรับฟังเหตุการณ์ใด ๆ คลาสนั้นต้องอิมพลีเมนต์อินเตอร์เฟซประเภท Listener ที่สอดคล้องกัน เช่น คลาสที่ต้องการรับฟังเหตุการณ์ ActionEvent จะต้องอิมพลีเมนต์อินเตอร์เฟซที่ชื่อ ActionListener ซึ่งภายในอินเตอร์เฟซดังกล่าวจะมีเมธอดที่ยังไม่สมบูรณ์ที่ผู้เขียนโปรแกรมต้องทำการโอเวอร์ไรด์เพื่อให้โปรแกรมทำงานได้เมื่อมีเหตุการณ์เกิดขึ้น
4. ออบเจกต์ประเภทเหตุการณ์ (Event) ที่เกิดจากแหล่งกำเนิดเหตุการณ์จะถูกส่งไปยังออบเจกต์ของคลาสที่สามารถรับฟังเหตุการณ์ประเภทนั้น โดยตัววัตถุของแหล่งกำเนิดเหตุการณ์ต้องทำการลงทะเบียนเพื่อให้สามารถรับฟังเหตุการณ์ที่จะเกิดขึ้นได้ โดยตัวออบเจกต์ประเภทเหตุการณ์จะถูกส่งไปยังคลาสที่ทำหน้าที่รับฟังเหตุการณ์ดังกล่าวและจะทำงานเมื่อมีเหตุการณ์เกิดขึ้น

การเขียนชุดคำสั่งเพื่อให้แหล่งกำเนิดเหตุการณ์สามารถทำงานเมื่อมีเหตุการณ์เกิดขึ้นทำได้โดย การลงทะเบียนตัวรับฟังเหตุการณ์(Listener) ที่เกี่ยวข้องโดยการใช้เมธอด addXListener(XListener listener) เช่น หากแหล่งกำเนิดเหตุการณ์คือ ปุ่ม(Button) จะต้องลงทะเบียนโดยใช้เมธอด addActionListener และต้องสร้างตัวรับฟังเหตุการณ์ผูกติด

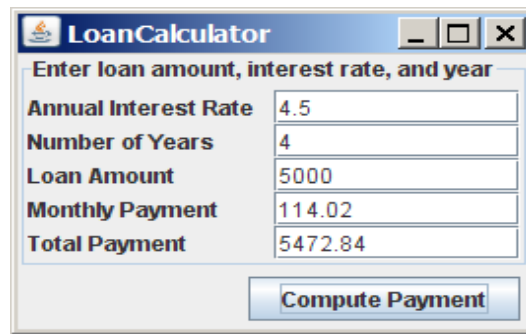
กับปุ่มดังกล่าว และทำการโอเวอร์ไรด์เมธอดที่ใช้ในการจัดการเหตุการณ์ ซึ่งกรณีที่เป็นปุ่ม เมธอดที่จะใช้ในการจัดการเหตุการณ์คือ `actionPerformed()` ตัวอย่างของการจัดการเหตุการณ์โดยการใช้ Delegation Model

การจัดการกับข้อผิดพลาด)Exception Handling)

โปรแกรมภาษาจาวาแบ่งข้อผิดพลาดที่อาจเกิดขึ้นขณะรันโปรแกรมเป็นสองประเภทคือ

1. Error เป็นข้อผิดพลาดที่ไม่สามารถแก้ไขและจัดการได้ เช่น `OutOfMemoryError`
2. Exception เป็นข้อผิดพลาดที่สามารถแก้ไขหรือจัดการได้ เช่น `ArrayIndexOutOfBoundsException`, `FileNotFoundException`, `ArithmeticException`

1. ให้ศึกษาและทดลองพิมพ์ตัวอย่างการสร้าง GUI จากตัวอย่างต่อไปนี้ ให้เขียนอธิบายการทำงานในแต่ละบรรทัด



```

1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4 import javax.swing.border.TitledBorder;
5 public class LoanCalculator extends JFrame {
6     // Create text fields for interest rate, years
7     // loan amount, monthly payment, and total payment
8     private JTextField jtfAnnualInterestRate = new JTextField();
9     private JTextField jtfNumberOfYears = new JTextField();
10    private JTextField jtfLoanAmount = new JTextField();
11    private JTextField jtfMonthlyPayment = new JTextField();
12    private JTextField jtfTotalPayment = new JTextField();
13
14    // Create a Compute Payment button
15    private JButton jbtComputeLoan = new JButton("Compute Payment");
16    public LoanCalculator() {
17        // Panel p1 to hold labels and text fields
18        JPanel p1 = new JPanel(new GridLayout(5, 2));
19        p1.add(new JLabel("Annual Interest Rate"));
20        p1.add(jtfAnnualInterestRate);
21        p1.add(new JLabel("Number of Years"));
22        p1.add(jtfNumberOfYears);
23        p1.add(new JLabel("Loan Amount"));
24        p1.add(jtfLoanAmount);
25        p1.add(new JLabel("Monthly Payment"));
26        p1.add(jtfMonthlyPayment);
27        p1.add(new JLabel("Total Payment"));
28        p1.add(jtfTotalPayment);
29        p1.setBorder(new
30            TitledBorder("Enter loan amount, interest rate, and years"));
31        // Panel p2 to hold the button
32        JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
33        p2.add(jbtComputeLoan);
34        // Add the panels to the frame
35        add(p1, BorderLayout.CENTER);
36        add(p2, BorderLayout.SOUTH);
37
38        // Register listener
39        jbtComputeLoan.addActionListener(new ButtonListener());
40    }
41    /** Handle the Compute Payment button */
42    private class ButtonListener implements ActionListener {
43        @Override

```

```

44     public void actionPerformed(ActionEvent e) {
45         // Get values from text fields
46         double interest =
47             Double.parseDouble(jtfAnnualInterestRate.getText());
48         int year = Integer.parseInt(jtfNumberOfYears.getText());
49         double loanAmount =
50             Double.parseDouble(jtfLoanAmount.getText());
51
52         // Create a loan object
53         Loan loan = new Loan(interest, year, loanAmount);
54
55         // Display monthly payment and total payment
56         jtfMonthlyPayment.setText(String.format("%.2f",
57             loan.getMonthlyPayment()));
58         jtfTotalPayment.setText(String.format("%.2f",
59             loan.getTotalPayment()));
60     }
61 }
62 public static void main(String[] args) {
63     LoanCalculator frame = new LoanCalculator();
64     frame.pack();
65     frame.setTitle("LoanCalculator");
66     frame.setLocationRelativeTo(null); // Center the frame
67     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
68     frame.setVisible(true);
69 }
70 }

1 public class Loan {
2     private double annualInterestRate;
3     private int numberOfYears;
4     private double loanAmount;
5     private java.util.Date loanDate;
6
7     /** Default constructor */
8     public Loan() {
9         this(2.5, 1, 1000);
10    }
11
12    /** Construct a loan with specified annual interest rate,
13        number of years, and loan amount
14        */
15    public Loan(double annualInterestRate, int numberOfYears,
16        double loanAmount) {
17        this.annualInterestRate = annualInterestRate;
18        this.numberOfYears = numberOfYears;
19        this.loanAmount = loanAmount;
20        loanDate = new java.util.Date();
21    }
22
23    /** Return annualInterestRate */
24    public double getAnnualInterestRate() {
25        return annualInterestRate;
26    }
27
28    /** Set a new annualInterestRate */

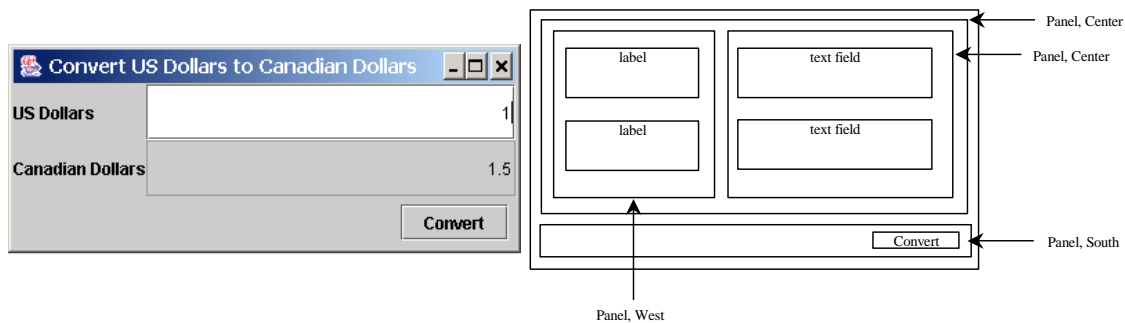
```

```

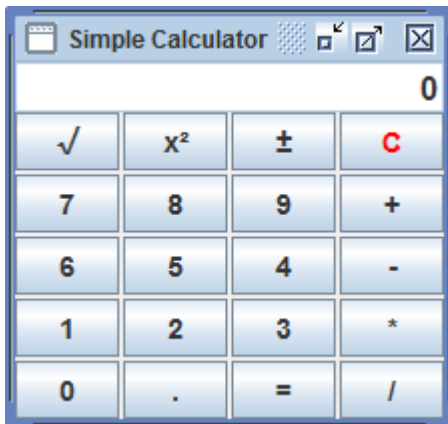
29  public void setAnnualInterestRate(double annualInterestRate) {
30      this.annualInterestRate = annualInterestRate;
31  }
32  /** Return numberOfYears */
33  public int getNumberOfYears() {
34      return numberOfYears;
35  }
36  /** Set a new numberOfYears */
37  public void setNumberOfYears(int numberOfYears) {
38      this.numberOfYears = numberOfYears;
39  }
40  }

```

2. จาก GUI ของโปรแกรมสำหรับการแปลงเงิน US Dollars เป็น Canadian Dollars จงเขียนโปรแกรมสำหรับการแปลงเงิน US Dollars เป็น Canadian Dollars ต่อไปนี้

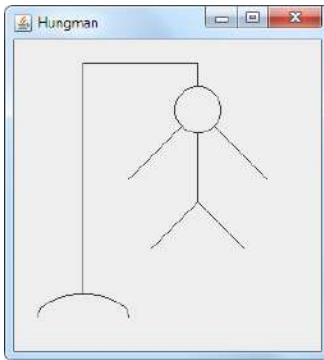


3. จาก GUI ของเครื่องคิดเลข ให้เพิ่มคำสั่งที่ทำให้เครื่องคิดเลขสามารถคำนวณได้



4. จาก GUI ของเกมส์ Hangman อธิบายผลลัพธ์ที่เกิดจากการใช้คำสั่งต่อไปนี้ในเมธอด

`public void paintComponent(Graphics g)` และให้เขียนโปรแกรมที่ทำให้ Hangman สามารถขยับตัวซ้ายขวาได้



```

g.drawArc(20, 220, 80, 40, 0, 180);
g.drawLine(20 + 40, 220, 20 + 40, 20);
g.drawLine(20 + 40, 20, 20 + 40 + 100, 20);
g.drawLine(20 + 40 + 100, 20, 20 + 40 + 100, 40);

int radius = 20;
g.drawOval(20 + 40 + 100 - radius, 40, 2 * radius, 2 *
radius);

g.drawLine(20 + 40 + 100 - (int)(radius *
Math.cos(Math.toRadians(45))),
40 + radius + (int)(radius *
Math.sin(Math.toRadians(45))),
20 + 40 + 100 - 60, 40 + radius + 60);

g.drawLine(20 + 40 + 100 + (int)(radius *
Math.cos(Math.toRadians(45))),
40 + radius + (int)(radius *
Math.sin(Math.toRadians(45))),
20 + 40 + 100 + 60, 40 + radius + 60);

g.drawLine(20 + 40 + 100, 40 + 2* radius,
20 + 40 + 100, 40 + radius + 80);

g.drawLine(20 + 40 + 100, 40 + radius + 80, 20 + 40 + 100 -
40, 40 + radius + 80 + 40);

g.drawLine(20 + 40 + 100, 40 + radius + 80, 20 + 40 + 100 +
40, 40 + radius + 80 + 40);

```

5. [Application] จาก GUI สำหรับระบบเก็บชื่อผู้เปิดบัญชีธนาคารจากคลาส Account ให้เพิ่มคำสั่งที่ทำให้โปรแกรมสามารถบันทึกข้อมูลได้

Show Detail of Account ...

ACCOUNT MONEY

Enter Data Account Money

ID: MONEY: BATH:

ANNUALINTERATE RATE:

DAY OPEN ACCOUNT:

FIRST NAME :

LAST NAME :


BIRTH DAY :

AGE : YEAR

SAVE SHOW

6. ให้ออกแบบคลาส AngryBirds และออกแบบ GUI ของเกมส์ตามหน้าจอดังนี้

SCENE 1: At Tokyo SCORE



Bird Position in y-axis m

Shooting speed m/s

Angle degree

OK

บทที่ 14 เธรด

วัตถุประสงค์

14.1 สามารถอธิบายความหมายของเธรด และการทำงานของเธรด

14.2 สามารถพัฒนาโปรแกรมโดยใช้เธรด

14.1 ความนำ

การเขียนโปรแกรมเชิงวัตถุด้วยภาษาจาวาสามารถสนับสนุนการเขียนโปรแกรมในรูปแบบ **Multithreading** หรือการทำงานที่โปรแกรมสามารถทำงานควบคู่กันไปพร้อมๆ กันได้ในแต่ละเธรดได้ โดยที่ทุกๆ เธรดนั้นมีลำดับความสำคัญของเธรด ในภาษาจาวาจะมี **main thread** เป็นหนึ่งในเธรดที่จะถูกเรียกให้ทำงาน

14.2 ความหมายของเธรด

เธรด(Thread) คือส่วนของการประมวลผลชุดลำดับคำสั่งของโปรแกรมที่เล็กที่สุดที่สามารถจัดการโดยตัวจัดการซึ่งโดยปกติแล้วจะพบในระบบปฏิบัติการ เธรด คือส่วนประกอบย่อยของโปรเซส (Process) โดยปกติโปรเซสที่มี 1 เธรดจะเรียกว่า **Single thread** แต่ถ้า 1 โปรเซสมีหลายเธรดจะเรียกว่า **Multithread** เพราะในโปรเซสหนึ่งอาจมีได้หลายเธรด เช่นในหน้าเว็บ 1 หน้า สามารถดาวน์โหลดข้อมูลพร้อมกับแสดงข้อความพร้อมทั้งแสดงรูปภาพในหน้าเดียวกันได้

เธรดจะแตกต่างจากโปรเซสที่ทำงานภายใต้ระบบปฏิบัติการแบบ **multi-tasking** ตรงที่โปรเซสแต่ละโปรเซสจะมีความเป็นอิสระจากกัน แต่เธรดแต่ละเธรดอาจใช้ข้อมูลร่วมกัน

คลาสแบบเธรดในภาษาจาวาคือคลาสที่สืบทอดมาจากคลาสที่ชื่อ Thread หรือคลาสที่อิมพลีเมนต์อินเตอร์เฟสชื่อ Runnable ภายในคลาสแบบเธรดจะต้องมีเมธอด run() ที่ไม่มีการรับอาร์กิวเมนต์ใด ๆ เข้ามา สถานะของเธรดอาจจะเป็น new runnable running blocked หรือ dead

14.3 การสร้างเธรด

ขั้นตอนในการสร้างเธรดสามารถสร้างได้ด้วยวิธีการดังต่อไปนี้

1. สร้างวัตถุของเธรด เช่น Thread myThread = new Thread();
2. เรียกเมธอด start() ในการสั่งให้เธรดทำงาน เช่น myThread.start();

โดยชุดคำสั่งที่ใช้ในการสร้างเธรดสามารถสร้างได้ด้วยวิธีการดังต่อไปนี้

- 14.3.1 การสร้างเธรดโดยสืบทอดมาจากคลาสเธรดและโอเวอร์ไรด์เมธอด run()

ตัวอย่างที่ 14.1 การสร้างเธรดโดยสืบทอดมาจากคลาสเธรดและโอเวอร์ไรด์เมธอด run()

```
class ThreadNum extends Thread {
    int num;
    public ThreadNum(int n) {
        num = n;
    }
    public void run() {
        for (int k=0; k < 10; k++)
            System.out.print(num);
    }
}
public class Ex14_1 {
    public static void main(String args[]) {
        ThreadNum t1 = new ThreadNum(1);
        ThreadNum t2 = new ThreadNum(2);
        t1.run();
        t2.run();
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

11111111112222222222

14.3.2 การสร้างเธรดโดยวิธีการอิมพลีเมนต์อินเทอร์เฟส Runnable

ตัวอย่างที่ 14.2 การสร้างเธรดโดยวิธีการอิมพลีเมนต์อินเทอร์เฟส Runnable

```
class ThreadNum implements Runnable {
    int num;
    public ThreadNum(int n) {
        num = n;
    }
    public void run() {
        for (int k=0; k < 10; k++)
            System.out.print(num);
    }
}
public class Ex14_2 {
    public static void main(String args[]) {
        Thread number1, number2, number3, number4, number5;
        number1 = new Thread(new ThreadNum(1)); number1.start();
        number2 = new Thread(new ThreadNum(2)); number2.start();
        number3 = new Thread(new ThreadNum(3)); number3.start();
        number4 = new Thread(new ThreadNum(4)); number4.start();
        number5 = new Thread(new ThreadNum(5)); number5.start();
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

11111111112222222222333333333355555555554444444444

15.4 การทำงานของเธรด

14.4.1 Thread Priority เป็นการกำหนดลำดับความสำคัญของเธรดโดยที่เธรดที่มีการกำหนด Priority สูงจะมีโอกาสที่จะได้รับการประมวลผลก่อนและนานกว่าเธรดที่มี Priority ต่ำกว่า n มีค่าระหว่าง 0-10 โดยสามารถใช้คำสั่งต่อไปนี้

```
setPriority(n)
```

ตัวอย่างที่ 14.3 การกำหนดลำดับความสำคัญของเธรด

```
public ThreadNum(int n) {
    num = n;
    setPriority(n);
}
```

14.4.2 Sleep Thread เป็นการให้เธรดหยุดรอในเวลาที่กำหนด โดยที่หน่วยเป็น millisecond เช่น `Thread.sleep(1000);` เป็นการหยุดการทำงาน 1 วินาที

ตัวอย่างที่ 14.3 การให้เธรดหยุดรอในเวลาที่กำหนด

```
class ThreadNum implements Runnable {
    int num;
    public ThreadNum(int n) {
        num = n;
    }
    public void run() {
        for (int k=0; k < 10; k++) {
            try {
                Thread.sleep((long) (Math.random() * 1000));
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
            System.out.print(num);
        }
    }
}

public class Ex14_3 {
    public static void main(String args[]) {
        Thread number1, number2, number3, number4, number5;
        number1 = new Thread(new ThreadNum(1)); number1.start();
        number2 = new Thread(new ThreadNum(2)); number2.start();
        number3 = new Thread(new ThreadNum(3)); number3.start();
        number4 = new Thread(new ThreadNum(4)); number4.start();
        number5 = new Thread(new ThreadNum(5)); number5.start();
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
111111111114444444222333332222222445555555555433333
```

14.4.3 สถานะของเธรด

วงจรชีวิตของแต่ละเธรดเริ่มต้นที่ขั้น `new` เมื่อถูกเริ่มสร้างและจะอยู่ที่ขั้นนี้จนกระทั่งเมธอด `start` ถูกเรียกประมวลผล ซึ่งจะทำให้เธรดนั้นๆ อยู่ที่ขั้น `ready` โดยที่เธรดที่มี `priority` สูงที่สุดจะถูกทำงานก่อนโดยเขาสู่ขั้น `running` แต่ละเธรดจะเข้าสู่ขั้นสุดท้ายของวงจรชีวิตคือ `dead` เมื่อเธรดนั้นทำงานในเมธอด `run` เสร็จเรียบร้อยหรือเมื่อถูกหยุดการทำงานด้วยเหตุผลใดๆ ก็ตาม ออปเจ็คแบบเธรดเมื่อลงทะเบียนไว้กับตัวตารางเวลาแล้ว อาจยังไม่มีกรันโปรแกรมโดยทันที แต่ทั้งนี้จะขึ้นอยู่กับสถานะของออปเจ็ค โดยสามารถแสดงรายละเอียดของสถานะของเธรดดังนี้

14.4.4 `suspend()` และ `resume()`

`suspend()` สั่งให้เธรดหยุดทำงานและเข้าสู่สถานะ `blocked` ส่วน `resume()` ให้เธรดเข้าสู่สถานะ `ready`

ตัวอย่างที่ 14.5 ตัวอย่างการใช้ `suspend()` และ `resume()`

```
class ThreadNum extends Thread {
    int num;
    public ThreadNum(int n) {
        num = n;
    }
    public void run() {
        for (int k=0; k < 20; k++) {
            System.out.print(num);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}

public class Ex14_5 {
    public static void main(String args[]) {
        ThreadNum t1 = new ThreadNum(1);
        ThreadNum t2 = new ThreadNum(2);
        t1.start();
        t2.start();
        try {
            Thread.sleep(200);
            t1.suspend();
            Thread.sleep(1000);
            t1.resume();
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

1221222222222121212212121122112111111111

14.4.5 join() หยุดรอเธรดลูกให้ทำงานจบก่อน ส่วน yied() จะปล่อยให้เธรดอื่นทำงาน

ตัวอย่างที่ 14.6 join() หยุดรอเธรดลูกให้ทำงานจบก่อน ส่วน yied() จะปล่อยให้เธรดอื่นทำงาน

```
public class Ex14_6 {
    public static void main(String args[]) {
        Thread number1, number2;
        number1 = new Thread(new ThreadNum(1)); number1.start();
        number2 = new Thread(new ThreadNum(2)); number2.start();
        try{
            number1.join();
            number2.join();
        } catch (Exception e) {}
        System.out.print("\nMain stop");
    }
}

class ThreadNum implements Runnable {
    int num;
    public ThreadNum(int n) {
        num = n;
    }
    public void run() {
        for (int k=0; k < 10; k++)
            System.out.print(num);
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

11111111112222222222

Main stop

14.4.6 stop() สั่งให้เธรดหยุดทำงาน

ตัวอย่างที่ 14.7 สั่งให้เธรดหยุดทำงาน

```
public class Ex14_7 {
    public static void main(String args[]) {
        MyThreadStop t=new MyThreadStop(1);
        t.start();
        try {
            Thread.sleep(4000);
        } catch (InterruptedException e) { }
        t.stop();
    }
}

class MyThreadStop extends Thread {
    int num;
    public MyThreadStop(int n) {
        num = n;
    }
    public void run(){
        try{
            while(true)
            {
                sleep(1000);
                System.out.println("running");
            }
        } catch (ThreadDeath e){
            System.out.print("Killed");
        }
        catch (InterruptedException ee){}
    }
}
```

ผลลัพธ์ที่เกิดขึ้นคือ

```
running
running
running
Killed
```

14.4.7 wait() notify() notifyAll()

โดยเมธอด `wait()` เป็นการสั่งให้เธรดหยุดทำงานไปอยู่ในสถานะ `block` ส่วนเมธอด `notify()` เป็นการสั่งให้เธรดกลับเข้ามาทำงานอีกครั้ง

ตัวอย่างที่ 14.8 wait() notify() notifyAll()

```
import java.util.List;
import java.util.ArrayList;
class Producer implements Runnable
{
    private final List<Integer> taskQueue;
    private final int MAX_CAPACITY;

    public Producer(List<Integer> sharedQueue, int size)
    {
        this.taskQueue = sharedQueue;
        this.MAX_CAPACITY = size;
    }

    @Override
    public void run()
    {
        int counter = 0;
        while (true)
        {
            try
            {
                produce(counter++);
            }
            catch (InterruptedException ex)
            {
                ex.printStackTrace();
            }
        }
    }

    private void produce(int i) throws InterruptedException
    {
        synchronized (taskQueue)
        {
            while (taskQueue.size() == MAX_CAPACITY)
            {
                System.out.println("Queue is full " + Thread.currentThread().getName() + " is
waiting , size: " + taskQueue.size());
                taskQueue.wait();
            }

            Thread.sleep(1000);
            taskQueue.add(i);
            System.out.println("Produced: " + i);
            taskQueue.notifyAll();
        }
    }
}
class Consumer implements Runnable
{

```

```

private final List<Integer> taskQueue;

public Consumer(List<Integer> sharedQueue)
{
    this.taskQueue = sharedQueue;
}

@Override
public void run()
{
    while (true)
    {
        try
        {
            consume();
        } catch (InterruptedException ex)
        {
            ex.printStackTrace();
        }
    }
}

private void consume() throws InterruptedException
{
    synchronized (taskQueue)
    {
        while (taskQueue.isEmpty())
        {
            System.out.println("Queue is empty " + Thread.currentThread().getName() + " is
waiting , size: " + taskQueue.size());
            taskQueue.wait();
        }
        Thread.sleep(1000);
        int i = (Integer) taskQueue.remove(0);
        System.out.println("Consumed: " + i);
        taskQueue.notifyAll();
    }
}
}

public class Ex14_8
{
    public static void main(String[] args)
    {
        List<Integer> taskQueue = new ArrayList<Integer>();
        int MAX_CAPACITY = 5;
        Thread tProducer = new Thread(new Producer(taskQueue, MAX_CAPACITY), "Producer");
        Thread tConsumer = new Thread(new Consumer(taskQueue), "Consumer");
        tProducer.start();
        tConsumer.start();
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ

```

Produced: 0
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Queue is full Producer is waiting , size: 5
Consumed: 0
Consumed: 1
Consumed: 2
Consumed: 3

```



```

Consumed: 4
Queue is empty Consumer is waiting , size: 0
Produced: 5
Produced: 6
Produced: 7
Produced: 8
Produced: 9
Queue is full Producer is waiting , size: 5
Consumed: 5
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Queue is empty Consumer is waiting , size: 0

```

14.5 ตัวอย่างโปรแกรมโดยใช้เธรด

ตัวอย่างที่ 14.8 ตัวอย่างโปรแกรมโดยใช้เธรด

```

import java.awt.*;
import java.util.Formatter;
import javax.swing.*;
public class BouncingBallSimple extends JPanel {
    private static final int BOX_WIDTH = 640;
    private static final int BOX_HEIGHT = 480;
    private float ballRadius = 200;
    private float ballX = ballRadius + 50;
    private float ballY = ballRadius + 20;
    private float ballSpeedX = 3;
    private float ballSpeedY = 2;
    private static final int UPDATE_RATE = 30;
    public BouncingBallSimple() {
        this.setPreferredSize(new Dimension(BOX_WIDTH, BOX_HEIGHT));
        Thread gameThread = new Thread() {
            public void run() {
                while (true) {
                    ballX += ballSpeedX;
                    ballY += ballSpeedY;
                    if (ballX - ballRadius < 0) {
                        ballSpeedX = -ballSpeedX;
                        ballX = ballRadius;
                    } else if (ballX + ballRadius > BOX_WIDTH) {
                        ballSpeedX = -ballSpeedX;
                        ballX = BOX_WIDTH - ballRadius;
                    }
                    if (ballY - ballRadius < 0) {
                        ballSpeedY = -ballSpeedY;
                        ballY = ballRadius;
                    } else if (ballY + ballRadius > BOX_HEIGHT) {
                        ballSpeedY = -ballSpeedY;
                        ballY = BOX_HEIGHT - ballRadius;
                    }
                    repaint();
                    try {
                        Thread.sleep(1000 / UPDATE_RATE);
                    } catch (InterruptedException ex) { }
                }
            }
        };
    }
};

```

```

        gameThread.start();
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.BLACK);
        g.fillRect(0, 0, BOX_WIDTH, BOX_HEIGHT);
        g.setColor(Color.BLUE);
        g.fillOval((int) (ballX - ballRadius), (int) (ballY - ballRadius),
            (int) (2 * ballRadius), (int) (2 * ballRadius));
        g.setColor(Color.WHITE);
        g.setFont(new Font("Courier New", Font.PLAIN, 12));
        StringBuilder sb = new StringBuilder();
        Formatter formatter = new Formatter(sb);
        formatter.format("Ball @(%3.0f,%3.0f) Speed=(%2.0f,%2.0f)", ballX, ballY,
            ballSpeedX, ballSpeedY);
        g.drawString(sb.toString(), 20, 30);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new JFrame("A Bouncing Ball");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setContentPane(new BouncingBallSimple());
                frame.pack();
                frame.setVisible(true);
            }
        });
    }
}

```

ผลลัพธ์ที่เกิดขึ้นคือ



14.6 บทสรุป

การเขียนโปรแกรมเชิงวัตถุด้วยภาษาจาวาสามารถสนับสนุนการเขียนโปรแกรมในรูปแบบการทำงานที่โปรแกรมสามารถทำงานควบคู่กันไปพร้อมๆ กันได้ในแต่ละเธรดได้ โดยที่ทุกๆ เธรดนั้นมีลำดับความสำคัญของเธรด เธรด (Thread) คือส่วนของการประมวลผลชุดลำดับคำสั่งของโปรแกรมที่เล็กที่สุดที่สามารถจัดการโดยตัวจัดการ ซึ่งโดยปกติแล้วจะพบในระบบปฏิบัติการ เธรด คือส่วนประกอบย่อยของโปรเซส (Process) โดยปกติโปรเซสที่มี 1 เธรดจะเรียกว่า **Single thread** แต่ถ้า 1 โปรเซสมีหลายเธรดจะเรียกว่า **Multithread** เพราะในโปรเซสหนึ่งอาจมีได้หลายเธรด เช่นในหน้าเว็บ 1 หน้า สามารถดาวน์โหลดข้อมูลพร้อมกับแสดงข้อความพร้อมทั้งแสดงรูปภาพในหน้าเดียวกันได้ ขั้นตอนในการสร้างเธรดสามารถสร้างได้ด้วยวิธีการดังต่อไปนี้

1. สร้างวัตถุของเธรด เช่น `Thread myThread = new Thread();`
2. เรียกเมธอด `start()` ในการสั่งให้เธรดทำงาน เช่น `myThread.start();`

โดยชุดคำสั่งที่ใช้ในการสร้างเธรดสามารถสร้างได้ด้วยวิธีการดังต่อไปนี้ การสร้างเธรดโดยสืบทอดมาจากคลาสเธรดและโอเวอร์ไรด์เมธอด `run()` และการสร้างเธรดโดยวิธีการอิมพลีเมนต์อินเทอร์เฟซ `Runnable` วงจรชีวิตของแต่ละเธรดเริ่มต้นที่ขึ้น `new` เมื่อถูกเริ่มสร้างและจะอยู่ที่ขึ้นนี้จนกระทั่งเมธอด `start` ถูกเรียกประมวลผล ซึ่งจะทำให้เธรดนั้นๆ อยู่ที่ขึ้น `ready` โดยที่เธรดที่มี `priority` สูงที่สุดจะถูก ทำงานก่อนโดยเขาสูงขึ้น `running` แต่ละเธรดจะเข้าสู่ขั้นสุดท้ายของวงจรชีวิตคือ `dead` เมื่อเธรดนั้น ทำงานในเมธอด `run` เสร็จเรียบร้อยหรือเมื่อถูกหยุดการทำงานด้วยเหตุผลใดๆ ก็ตาม ออปเจ็คแบบเธรดเมื่อลงทะเลเป็นไว้กับตัวตารางเวลาแล้ว อาจยังไม่มีกรันโปรแกรมโดยทันที แต่ทั้งนี้จะขึ้นอยู่กับสถานะของออปเจ็ค

14.7 แบบฝึกหัดปฏิบัติการ

1. จงอธิบายผลลัพธ์ของการทำงานโดยการใช้ Thread ต่อไปนี้

```
class Thread1 extends Thread{
    Thread1(String name){
        super(name);
    }
    public void run(){
        for(int i=0;i<10;i++){
            System.out.println(getName()+" ");
        }
    }
}
class TestThread1{
    public static void main(String args[]){
        new Thread1("A").start();
        new Thread1("B").start();
    }
}
```

2. ให้ศึกษาตัวอย่างของการใช้ Thread แทน Timer ต่อไปนี้

```
1 import java.awt.*;
2 import java.util.Formatter;
3 import javax.swing.*;
9 public class BouncingBallSimple extends JPanel {
10     // Container box's width and height
11     private static final int BOX_WIDTH = 640;
12     private static final int BOX_HEIGHT = 480;
13
14     // Ball's properties
15     private float ballRadius = 200; // Ball's radius
16     private float ballX = ballRadius + 50; // Ball's center (x, y)
17     private float ballY = ballRadius + 20;
18     private float ballSpeedX = 3; // Ball's speed for x and y
19     private float ballSpeedY = 2;
20
21     private static final int UPDATE_RATE = 30; // Number of refresh per second
22
23     /** Constructor to create the UI components and init game objects. */
24     public BouncingBallSimple() {
25         this.setPreferredSize(new Dimension(BOX_WIDTH, BOX_HEIGHT));
26
27         // Start the ball bouncing (in its own thread)
28         Thread gameThread = new Thread() {
29             public void run() {
30                 while (true) { // Execute one update step
31                     // Calculate the ball's new position
32                     ballX += ballSpeedX;
33                     ballY += ballSpeedY;
34                     // Check if the ball moves over the bounds
35                     // If so, adjust the position and speed.
36                     if (ballX - ballRadius < 0) {
37                         ballSpeedX = -ballSpeedX; // Reflect along normal
38                         ballX = ballRadius; // Re-position the ball at the edge
39                     } else if (ballX + ballRadius > BOX_WIDTH) {
40                         ballSpeedX = -ballSpeedX;
41                         ballX = BOX_WIDTH - ballRadius;
42                     }
43                     // May cross both x and y bounds
44                     if (ballY - ballRadius < 0) {
```

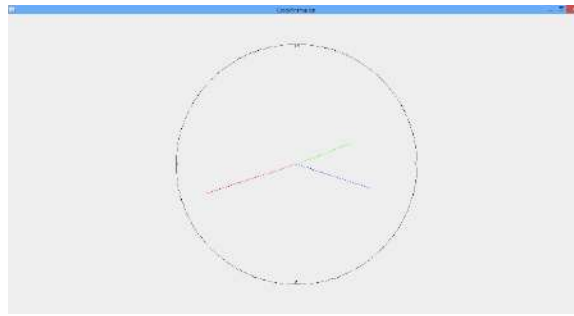
```

45         ballSpeedY = -ballSpeedY;
46         ballY = ballRadius;
47     } else if (ballY + ballRadius > BOX_HEIGHT) {
48         ballSpeedY = -ballSpeedY;
49         ballY = BOX_HEIGHT - ballRadius;
50     }
51     // Refresh the display
52     repaint(); // Callback paintComponent()
53     // Delay for timing control and give other threads a chance
54     try {
55         Thread.sleep(1000 / UPDATE_RATE); // milliseconds
56     } catch (InterruptedException ex) { }
57 }
58 }
59 };
60 gameThread.start(); // Callback run()
61 }
62
63 /** Custom rendering codes for drawing the JPanel */
64 @Override
65 public void paintComponent(Graphics g) {
66     super.paintComponent(g); // Paint background
67
68     // Draw the box
69     g.setColor(Color.BLACK);
70     g.fillRect(0, 0, BOX_WIDTH, BOX_HEIGHT);
71
72     // Draw the ball
73     g.setColor(Color.BLUE);
74     g.fillOval((int) (ballX - ballRadius), (int) (ballY - ballRadius),
75         (int) (2 * ballRadius), (int) (2 * ballRadius));
76
77     // Display the ball's information
78     g.setColor(Color.WHITE);
79     g.setFont(new Font("Courier New", Font.PLAIN, 12));
80     StringBuilder sb = new StringBuilder();
81     Formatter formatter = new Formatter(sb);
82     formatter.format("Ball @(%3.0f,%3.0f) Speed=(%2.0f,%2.0f)", ballX, ballY,
83         ballSpeedX, ballSpeedY);
84     g.drawString(sb.toString(), 20, 30);
85 }
86
87 /** main program (entry point) */
88 public static void main(String[] args) {
89     // Run GUI in the Event Dispatcher Thread (EDT) instead of main thread.
90     javax.swing.SwingUtilities.invokeLater(new Runnable() {
91         public void run() {
92             // Set up main window (using Swing's JFrame)
93             JFrame frame = new JFrame("A Bouncing Ball");
94             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
95             frame.setContentPane(new BouncingBallSimple());
96             frame.pack();
97             frame.setVisible(true);
98         }
99     });
100 }
101 }

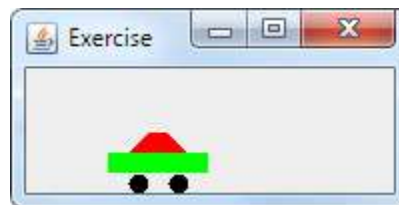
```

จงอธิบายผลลัพธ์ของการทำงานโดยที่มีการใช้ Thread ยกตัวอย่างพร้อมอธิบายบรรทัดที่มีการใช้

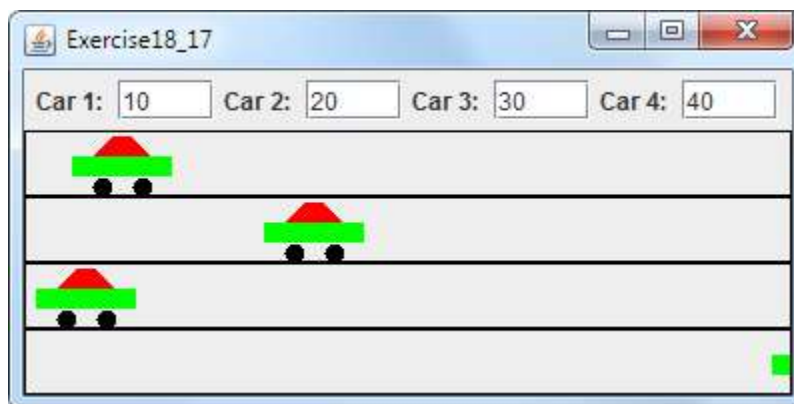
3. ให้เปลี่ยนการทำงานของ Clock Animation จากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



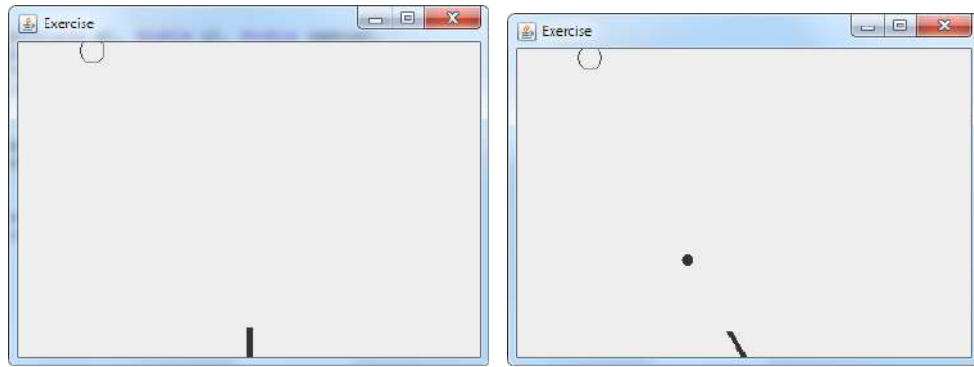
4. ให้เปลี่ยนการทำงานของ Race car จากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



5. จากคลาส Race car ในปฏิบัติการข้อที่ผ่านมาให้เพิ่มจำนวนรถเป็น คันจำลองให้เป็นสนามแข่งรถ และสามารถ 4 กำหนดความเร็วให้รถแต่ละคันได้จากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



6. เขียนโปรแกรมที่แสดงรูปบอลลอนในตำแหน่ง random ใน Panel ให้ใช้ปุ่มลูกศรซ้ายและลูกศรขวาในการเล็งตำแหน่งของปืนให้ตรงกับบอลลูน ใช้ปุ่มลูกศร up-arrow สำหรับยิงลูกกระสุน เมื่อลูกกระสุนถูกบอลลูนให้บอลลูนที่โดนกระสุนหายไป และ random บอลลูนที่ตำแหน่งใหม่ขึ้นมาจากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



บทที่ 15 การพัฒนาเกมส์และแอปพลิเคชันโดยการเขียนโปรแกรมเชิงวัตถุ

วัตถุประสงค์

15.1 สามารถพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุ

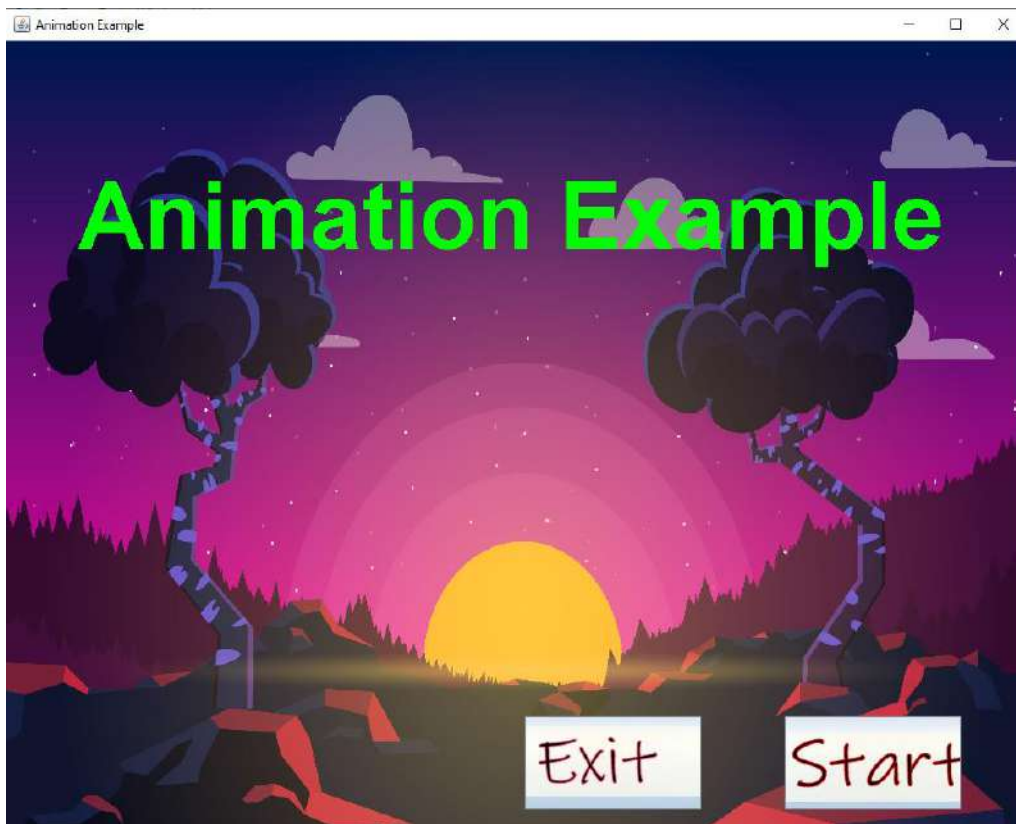
15.2 สามารถพัฒนาแอปพลิเคชันโดยการเขียนโปรแกรมเชิงวัตถุ

15.1 ความนำ

การเขียนโปรแกรมโดยการจำลองแนวคิดเชิงวัตถุเป็นอีกหนึ่งในภาษาโปรแกรมที่มีการพัฒนามาอย่างต่อเนื่องและถูกนำไปใช้ในการพัฒนาในด้านต่างๆ ในปัจจุบัน เช่น การเขียนโปรแกรมผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชันต่าง ๆ หรือแม้กระทั่งการใช้แนวความคิดในการเขียนโปรแกรมเชิงวัตถุมาเพื่อใช้พัฒนาเกมส์ต่างๆ ในบทนี้จะเป็นการยกตัวอย่างการพัฒนาเกมส์อย่างง่ายโดยอาศัยแนวคิดเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุ

15.2 ตัวอย่างการพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุ

ตัวอย่างการพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุในบทนี้จะนำเสนอเกมส์ที่ประกอบด้วยตัวละครผีลอยอยู่บนหน้าจอแล้วตัวละครอัศวินสามารถยิงตัวละครผีดังกล่าว เมื่อยิงโดนจะได้คะแนนเพิ่ม โดยสามารถยิงได้ภายในเวลาที่กำหนด นอกจากนี้ภายในเกมส์จะมีกับดักถ้าหากเรายิงโดนเราจะเสียเลือด ตัวอย่างการพัฒนาเกมส์ในบทนี้จะนำแนวคิดเรื่อง คลาส วัตถุ การสืบทอดคุณสมบัติ การห่อหุ้ม การมีหลายรูป เรต มาใช้ในการพัฒนาเกมส์ โดยตัวอย่างของเกมส์มีดังนี้



การเขียนโปรแกรมภายในคลาสที่เกี่ยวข้องกับการพัฒนาเกมส์ข้างต้นมีดังนี้

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.net.URL;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JPanel;
public class homenames extends JPanel{
    private ImageIcon bg = new ImageIcon(this.getClass().getResource("eezy_61-01.jpg"));
    private ImageIcon exit = new ImageIcon(this.getClass().getResource("exit.png"));
    private ImageIcon starts = new ImageIcon(this.getClass().getResource("start.png"));
    public JButton BStart = new JButton(starts);
    public JButton BExit1 = new JButton(exit);
    homenames() {
        setLayout(null);
        BExit1.setBounds(500, 650, 170,90);
        add(BExit1);
        BStart.setBounds(750,650,170,90);
        add(BStart);
    }
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawImage(bg.getImage(),0,0,1000,800,this);
        g.setColor(Color.GREEN);
        g.setFont(new Font("2005_iannnnnTKO",Font.CENTER_BASELINE,90));
        g.drawString("Animation Example",70,200);
    }
}
```

หมายเหตุ รูปภาพ background จากเว็บ <https://www.vecteezy.com/free-vector/game-background> Game Background Vectors by Vecteezy

คลาส homenames เป็นคลาสที่สืบทอดมาจากคลาส JPanel โดยจะแสดงหน้าแรกของเกมส์โดยประกอบด้วยปุ่ม Exit และปุ่ม Start

การสร้างปุ่มทั้งสองปุ่มดังกล่าวสามารถเพิ่มรูปลงไปในแต่ละปุ่มได้โดยการใช้คำสั่งต่อไปนี้

```
private ImageIcon exit = new ImageIcon(this.getClass().getResource("exit.png"));
private ImageIcon starts = new ImageIcon(this.getClass().getResource("start.png"));
public JButton BStart = new JButton(starts);
public JButton BExit1 = new JButton(exit);
```

เมื่อสร้างปุ่มเรียบร้อยแล้วจะนำปุ่มดังกล่าวไปเป็นส่วนประกอบของหน้าจอแรกของเกมส์โดยการเพิ่มภายในคอนสตรักเตอร์ของคลาสด้วยคำสั่งต่อไปนี้

```
homenames(){
    setLayout(null);
    BExit1.setBounds(500, 650, 170,90);
    add(BExit1);
    BStart.setBounds(750,650,170,90);
    add(BStart);
}
```



```
import java.awt.geom.Rectangle2D;
import java.net.URL;
import javax.swing.ImageIcon;
import javax.swing.JPanel;
public class Fireball extends JPanel{
    public ImageIcon[] imfire = new ImageIcon[5];
    public int y;
    public int x;
    public int count=0;
    Fireball(int x,int y){
        for(int i=0;i<imfire.length;i++){
            String imageLocation = "b"+(i+1)+".png";
            imfire[i] = new ImageIcon(this.getClass().getResource(imageLocation));
        }
        this.x=x;
        this.y=y;
    }
    public void move() {
        this.y-=1;
    }
    public Rectangle2D getbound(){
        return (new Rectangle2D.Double(x,y,25,25));
    }
}
```

ภายในเกมส์จะมีการยิงก้อนหินไฟไปยังวัตถุที่กำลังลอยอยู่ ดังนั้นเราสามารถสร้างคลาสที่เก็บรายละเอียดของก้อนหินไฟนี้ได้โดยการสร้างคลาส Fireball เป็นคลาสที่สืบทอดมาจากคลาส JPanel

ในคลาส Fireball จะประกอบด้วยรูปภาพที่เก็บเป็นอาร์เรย์เพื่อให้สามารถแสดงภาพเคลื่อนไหวที่มีรูปที่ต่างกันได้โดยมีตัวอย่างการเขียนโปรแกรมดังนี้

```
for(int i=0;i<imfire.length;i++){
    String imageLocation = "b"+(i+1)+".png";
    imfire[i] = new ImageIcon(this.getClass().getResource(imageLocation));
}
```

นอกจากนี้จะมีการเก็บพิกัดปัจจุบันของก้อนหินไฟด้วยตัวแปร x y และเพื่อให้สามารถนำก้อนหินไฟไปตรวจสอบการชนกับวัตถุอื่นหรือไม่จะทำได้โดยการสร้างขอบเขตให้กับรูปก้อนหินโดยการสร้างเป็นกรอบสี่เหลี่ยมตามขนาดที่ต้องการดังตัวอย่างเมธอดต่อไปนี้

```
public Rectangle2D getbound(){
    return (new Rectangle2D.Double(x,y,25,25));
}
```



```
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.geom.Rectangle2D;
import java.net.URL;
import javax.swing.ImageIcon;
public class Ghost {
    Image img;
    public int x = 0;
    public int y= (int) ((Math.random()*300)+20);
    public int count = 0;
    Ghost(){
        String imageLocation = "g0.png";
        URL imageURL = this.getClass().getResource(imageLocation);
        img = Toolkit.getDefaultToolkit().getImage(imageURL);
        runner.start();
    }
    Thread runner = new Thread(new Runnable() {
        public void run() {
            while(true){
                x += 2;
                if(x >= 1100){
                    img = null;
                    runner = null;
                    x = -500;
                    y = -500;
                }
                try{
                    runner.sleep(10);
                }catch(InterruptedException e){}
            }
        }
    });

    public Image getImage(){
        return img;
    }

    public int getX(){
        return x;
    }
    public int getY(){
        return y;
    }
    public Rectangle2D getbound(){
        return (new Rectangle2D.Double(x,y,45,45));
    }
}
```

ตัวละครผีที่กำลังลอยอยู่ สามารถสร้างคลาสที่เก็บรายละเอียดของตัวละครผีนี้ได้โดยการสร้างคลาส Ghost ซึ่งเป็นคลาสที่สืบทอดมาจากคลาส JPanel

ในคลาส Ghost จะประกอบด้วยรูปภาพตัวละครผี นอกจากนี้จะมีการเก็บพิกัดปัจจุบันของตัวละครผีด้วยตัวแปร x y และเพื่อให้ตัวละครผีสามารถขยับได้ตลอดเวลาจะใช้เธรดเพื่อให้ตัวละครมีการเคลื่อนที่ด้วยคำสั่งต่อไปนี้

```
Thread runner = new Thread(new Runnable() {
    public void run() {
        while(true){
            x += 2;
            if(x >= 1100){
                img = null;
                runner = null;
                x = -500;
                y = -500;
            }
            try{
                runner.sleep(10);
            }catch(InterruptedException e){}
        }
    }
});
```

การนำตัวละครผีไปตรวจสอบว่ามีกรชนกับวัตถุอื่นหรือไม่จะทำได้โดย การสร้างขอบเขตให้กับตัวละครผีโดยการสร้างเป็นกรอบสี่เหลี่ยมตามขนาดที่ต้องการ ดังตัวอย่างเมธอดต่อไปนี้

```
public Rectangle2D getbound(){
    return (new Rectangle2D.Double(x,y,45,45));
}
```



```
import javax.swing.ImageIcon;
import javax.swing.JPanel;
public class Knight{
    public ImageIcon[] im = new ImageIcon[7];
    public int x;
    public int count = 0;
    Knight (){
        for(int i=0;i<im.length;i++){
            im[i] = new ImageIcon(this.getClass().getResource((i+1)+".png"));
        }
    }
}
```

ตัวละครอัศวินภายในเกมส์ สามารถสร้างคลาสที่เก็บรายละเอียดของตัวละครอัศวินนี้ได้โดย การสร้างคลาส Knight

ในคลาส Knight จะประกอบด้วยรูปภาพตัวละครอัศวินที่เก็บเป็นอาร์เรย์เพื่อให้สามารถแสดงภาพเคลื่อนไหวที่มีรูปที่แตกต่างกันได้ นอกจากนี้จะมีการเก็บพิกัดปัจจุบันของตัวละครอัศวินด้วยตัวแปร x

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.sound.sampled.Clip;
import javax.swing.JFrame;
public class PlayGames extends JFrame implements ActionListener{
    homegames homegames1 = new homegames();
    playstatel statel = new playstatel();
    gameover gover = new gameover();
    public PlayGames() {
        this.setSize(1000,800);
        this.add(homegames1);
        homegames1.BExit1.addActionListener(this);
        homegames1.BStart.addActionListener(this);
        statel.BExithome.addActionListener(this);
        statel.BPause.addActionListener(this);
        statel.Bresum.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == homegames1.BStart){
            this.setLocationRelativeTo(null);
            this.remove(homegames1);
            this.setSize(1000,800);
            this.add(statel);
            statel.requestFocusInWindow();
            statel.timestart = false;
            statel.scor=0;
            statel.HP =3;
            statel.times = 100;
            statel.startball=false;
            statel.timestart=false;
        }else if(e.getSource() == statel.BExithome){
            System.exit(0);
        }else if (e.getSource() == homegames1.BExit1){
            System.exit(0);
        }else if(e.getSource() == statel.BPause){
            this.setLocationRelativeTo(null);
            this.setSize(1000,800);
            this.add(statel);
            statel.requestFocusInWindow();
            statel.t.suspend();
            statel.time.suspend();
            statel.actor.suspend();
            statel.tballs1.suspend();
        }else if(e.getSource() == statel.Bresum){
            this.setLocationRelativeTo(null);
            this.setSize(1000,800);
            this.add(statel);
            statel.requestFocusInWindow();
            statel.t.resume();
            statel.time.resume();
            statel.actor.resume();
            statel.tballs1.resume();
        }
    }
}
```

```

    }
}

public static void main(String[] args) {
    JFrame jf = new PlayGames();
    jf.setSize(1000,800);
    jf.setTitle("Animation Example");
    jf.setDefaultCloseOperation(EXIT_ON_CLOSE);
    jf.setVisible(true);
    jf.setLocationRelativeTo(null);
}
}

```

คลาส PlayGames เป็นคลาสที่สืบทอดมาจากคลาส JFrame และอิมพลีเมนต์อินเทอร์เฟซ ActionListener โดยจะแสดงหน้าแรกของเกมส์และเป็นคลาสที่ใช้สำหรับควบคุมการกดปุ่ม Exit และปุ่ม Start

การควบคุมการกดปุ่มต่างๆ จะทำภายในเมธอด public void actionPerformed(ActionEvent e) ซึ่งเป็นการโอเวอร์ไรด์จากคลาส ActionListener รายละเอียดการทำงานเมื่อกดปุ่มต่าง ๆ มีดังนี้

```

public void actionPerformed(ActionEvent e) {
    if(e.getSource()== homegames1.BStart){
        this.setLocationRelativeTo(null);
        this.remove(homegames1);
        this.setSize(1000,800);
        this.add(statel);
        statel.requestFocusInWindow();
        statel.timestart = false;
        statel.scor=0;
        statel.HP =3;
        statel.times = 100;
        statel.startball=false;
        statel.timestart=false;
    }else if(e.getSource() == statel.BExithome){
        System.exit(0);
    }else if (e.getSource() == homegames1.BExit1){
        System.exit(0);
    }else if(e.getSource() == statel.BPause){
        this.setLocationRelativeTo(null);
        this.setSize(1000,800);
        this.add(statel);
        statel.requestFocusInWindow();
        statel.t.suspend();
        statel.time.suspend();
        statel.actor.suspend();
        statel.tballs1.suspend();
    }else if(e.getSource() == statel.Bresum){
        this.setLocationRelativeTo(null);
        this.setSize(1000,800);
        this.add(statel);
        statel.requestFocusInWindow();
        statel.t.resume();
        statel.time.resume();
        statel.actor.resume();
        statel.tballs1.resume();
    }
}

```



```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Rectangle2D;
import java.net.URL;
import java.util.ArrayList;
import javax.swing.*;

public class playstate1 extends JPanel implements ActionListener{
    private final ImageIcon imgstate1 = new ImageIcon(this.getClass().getResource("swamp-01.jpg"));
    private final ImageIcon imgstate2 = new ImageIcon(this.getClass().getResource("eezy_61-01.jpg"));
    private final ImageIcon imgmeleon = new
ImageIcon(this.getClass().getResource("meleon.png"));
    private final ImageIcon pause = new ImageIcon(this.getClass().getResource("p.png"));
    private final ImageIcon resum = new ImageIcon(this.getClass().getResource("play.png"));
    private final ImageIcon back = new ImageIcon(this.getClass().getResource("back.png"));
    Knight m = new Knight();

    homegames hg = new homegames();
    ImageIcon feildover = new
ImageIcon(this.getClass().getResource("a_cartoon_forest_election_by_88srenaissance88-
d4zim5f.jpg"));
    ImageIcon img_paralyze = new ImageIcon(this.getClass().getResource("7.1.png"));
    ImageIcon exitover = new ImageIcon(this.getClass().getResource("exit.png"));
    ImageIcon restart = new ImageIcon(this.getClass().getResource("start.png"));
    JButton BStartover = new JButton(restart);
```

```

JButton BExitover = new JButton(exitover);

private JLabel score = new JLabel();
public JButton BPause = new JButton(pause);
public JButton BExithome = new JButton(back);
public JButton Bresum = new JButton(resum);

public ArrayList<Fireball> fireball = new ArrayList<Fireball>();
public ArrayList< Ghost > bal = new ArrayList< Ghost >();

public int times ;
public int HP = 3;
public int rs1 = 1;
public int rs2 = 2;
boolean timestart = true;
boolean startball = false;

private gameover gover = new gameover();
public int scor = 0;
boolean paralyze1 = false;
int time_paralyze=5;

Thread time = new Thread(new Runnable(){
    public void run(){
        while(true){
            try{
                Thread.sleep(10);
            }catch(Exception e){ }

            if(timestart == false){
                repaint();
            }
        }
    }
});

Thread actor = new Thread(new Runnable(){
    public void run(){
        while(true){
            try{
                Thread.sleep(1);
            }catch(Exception e){}
            repaint();
        }
    }
});

Thread tballs1 = new Thread(new Runnable(){
    public void run() {
        while(true){
            try{
                if(startball == false){
                    Thread.sleep((long) (Math.random()*10000)+2000);
                }
            }catch(InterruptedException e){
                e.printStackTrace();
            }
            if(startball == false){
                bal.add(new ball1());
            }
        }
    }
});

```



```

Thread paralyze = new Thread(new Runnable() {
    public void run() {
        while (true) {
            if (time_paralyze < 1) {
                paralyze1 = false;
                time_paralyze = 5;
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {e.printStackTrace();}
        }
    }
});
Thread t = new Thread(new Runnable() {
    public void run() {
        while (true) {
            if (timestart == false) {
                times = (times-1);
                if (paralyze1) {
                    time_paralyze--;
                }
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});

playstate1() {
    this.setFocusable(true);
    this.setLayout(null);
    BPause.setBounds(850, 50, 40, 40);
    Bresum.setBounds(900, 50, 40, 40);
    BPause.addActionListener(this);
    BExithome.addActionListener(this);
    Bresum.addActionListener(this);
    BExithome.addActionListener(this);
    this.add(BPause);
    this.add(BExithome);
    this.add(score);
    this.add(Bresum);

    this.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            int a = e.getKeyCode();
            if (!paralyze1) {
                if (a == KeyEvent.VK_A) {
                    m.x -= 10;
                    m.count++;
                }
                else if (a == KeyEvent.VK_D) {
                    m.x += 10;
                    m.count++;
                }
            }
            if (m.count > 3) {
                m.count = 0;
            }
            else if (a == KeyEvent.VK_UP) {
                m.count = 5;
                fireball.add(new Fireball(m.x, 550));
            }
        }
    })
}

```

```

    }
    public void keyReleased(KeyEvent e){
        m.count=0;
    }
});
m.x = 400;
time.start();
actor.start();
t.start();
tballs1.start();
paralyze.start();
}
public void paintComponent(Graphics g){
    super.paintComponent(g);
    if(times <= 0 || HP<=0){
        this.remove(BPause);
        this.remove(Bresum);
        this.remove(BExithome);
        this.setLayout(null);
        g.drawImage(feildover.getImage(),0,0,1000,800,this);
        g.setColor(Color.BLACK);
        g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,40));
        g.drawString("SCORE   "+scor,380,200);
        g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,70));
        g.drawString("GAME OVER",290,150);
        g.drawImage(imgmeleon.getImage(), 580, 360, 400, 400, this);

    }else if(times <= 50){
        g.drawImage(imgstate2.getImage(),0,0,1000,800,this);
        if(paralyze1){
            g.setColor(Color.WHITE);
            g.setFont(new Font("Hobo Std",Font.BOLD,50));
            g.drawImage(img_paralyze.getImage(), m.x, 550,100,150, this);
            g.drawString("AHHHH !!", m.x+100, 560);
        }else{
            g.drawImage(m.im[m.count].getImage(), m.x, 550,110,160, this);
        }
        if(m.x<0){
            m.x=this.getWidth()-10;
        }
        if(m.x>this.getWidth()){
            m.x=20;
        }
        for(int i=0;i<fireball.size();i++){
            Fireball ba = fireball.get(i);
            g.drawImage(ba.imfire[ba.count%5].getImage(), ba.x, ba.y,50,50, null);
            ba.move();
            ba.count++;
            if(ba.y<0){
                fireball.remove(i);
            }
        }
        for(int i=0 ; i<bal.size();i++){
            g.drawImage(bal.get(i).getImage(),
                bal.get(i).getX(),
                bal.get(i).getY(),
                100,100,this);
        }
        for(int i=0 ; i<fireball.size();i++){
            for(int j=0 ; j<bal.size();j++){
                if(Intersect(fireball.get(i).getbound(),bal.get(j).getbound())){
                    bal.remove(j);
                    fireball.remove(i);
                    scor += 10;
                    g.drawString("+10",m.x+100,650);
                }
            }
        }
    }
}

```

```

    }
    g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,30));
    g.setColor(Color.WHITE);
    g.drawString("SCORE = "+scor,50, this.getHeight()-10);
    g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,50));
    g.drawString("Time "+times,this.getWidth()-200,this.getHeight()-50);
    g.setColor(Color.WHITE);
    g.drawString("HP "+HP,50,this.getHeight()-50);
}else if(times <= 0 || HP<=0){
    this.remove(BPause);
    this.remove(Bresum);
    this.remove(BExithome);
    this.setLayout(null);
    g.drawImage(feildover.getImage(),0,0,1000,800,this);
    g.setColor(Color.BLACK);
    g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,40));
    g.drawString("SCORE "+scor,380,200);
    g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,70));
    g.drawString("GAME OVER",290,150);
    g.drawImage(imgmeleon.getImage(), 580, 360, 400, 400, this);
}else{
    g.drawImage(imgstatel.getImage(),0,0,1000,800,this);
    if(paralyzel){
        g.setColor(Color.RED);
        g.setFont(new Font("Hobo Std",Font.BOLD,50));
        g.drawImage(img_paralyze.getImage(), m.x, 550,100,150, this);
        g.drawString("-10",m.x+100,650);
        g.drawString("AHHHH !!!", m.x+100, 560);
    }else{
        g.drawImage(m.im[m.count].getImage(), m.x, 550,110,160, this);
    }
    if(m.x<0){
        m.x=this.getWidth()-10;
    }
    if(m.x>this.getWidth()){
        m.x=20;
    }
    for(int i=0;i<fireball.size();i++){
        Fireball ba = fireball.get(i);
        g.drawImage(ba.imfire[ba.count%5].getImage(),
            ba.x, ba.y,
            50,50, null);

        ba.move();
        ba.count++;
        if(ba.y<0){
            fireball.remove(i);
        }
    }

    for(int i=0 ; i<bal.size();i++){
        g.drawImage(bal.get(i).getImage(),bal.get(i).getX(),
            bal.get(i).getY(),
            100,100,this);
    }
    for(int i=0 ; i<fireball.size();i++){
        for(int j=0 ; j<bal.size();j++){
            if(Intersect(fireball.get(i).getbound(),bal.get(j).getbound())){
                bal.remove(j);
                fireball.remove(i);
                scor += 10;
                g.drawString("+10",m.x+100,650);
            }
        }
    }
    g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,30));
    g.setColor(Color.WHITE);
    g.drawString("SCORE = "+scor,50, this.getHeight()-10);

```

```

        g.setFont(new Font("Hobo Std", Font.HANGING_BASELINE, 50));
        g.drawString("Time "+times, this.getWidth()-200, this.getHeight()-50);
        g.setColor(Color.WHITE);
        g.drawString("HP "+HP, 50, this.getHeight()-50);
    }
}
public boolean Intersect(Rectangle2D a, Rectangle2D b){
    return (a.intersects(b));
}
public void actionPerformed(ActionEvent e) {
    if(e.getSource()== BStartover){
        this.setSize(1000,800);
        this.add(hg);
        this.setLocation(null);
        timestart = true;
        startball = true;
    }else if(e.getSource() == BExitover){
        System.exit(0);
    }
}
}

```

คลาส playstate1 เป็นคลาสที่สืบทอดมาจากคลาส JPanel และอิมพลีเมนต์อินเทอร์เฟซ ActionListener โดยจะแสดงหน้าจอของการเล่นเกมส์ ภายในคลาสประกอบด้วยชุดคำสั่งสำหรับการเพิ่มปุ่มต่างๆ ในการควบคุมเกมส์ดังนี้

```

private final Imgelcon imgstate1 = new Imgelcon(this.getClass().getResource("swamp-01.jpg"));
private final Imgelcon imgstate2 = new Imgelcon(this.getClass().getResource("eezy_61-01.jpg"));
private final Imgelcon imgmeleon = new Imgelcon(this.getClass().getResource("meleon.png"));
private final Imgelcon pause = new Imgelcon(this.getClass().getResource("p.png"));
private final Imgelcon resum = new Imgelcon(this.getClass().getResource("play.png"));
private final Imgelcon back = new Imgelcon(this.getClass().getResource("back.png"));

```

การสร้างตัวละครอัศวินที่หน้าจอนี้ทำได้โดยคำสั่ง

```
Knight m = new Knight();
```

เนื่องจากภายในเกมส์จะมีตัวละครผีและก้อนหินไฟปรากฏตลอดเวลา ดังนั้นจึงต้องมีการเก็บตัวละครผีและก้อนหินไฟในอาร์เรย์ลิสต์ดังชุดคำสั่งต่อไปนี้

```

public ArrayList<Fireball> fireball = new ArrayList<Fireball>();
public ArrayList< Ghost > ba1 = new ArrayList< Ghost >();

```

การควบคุมเชรตหรือการประมวลผลพร้อมกันในเกมส์ทำได้โดยการประกาศตัวแปรดังต่อไปนี้สำหรับการควบคุม

```

public int times ;
public int HP = 3;
public int rs1 = 1;
public int rs2 = 2;
boolean timestart = true;
boolean startball = false;

```

ในการควบคุมการทำงานแบบเชรตของเวลาของการเล่นเกมส์ ตัวละครต่าง ๆ จะมีรูปแบบในการสร้างเชรตดังตัวอย่างต่อไปนี้

```

Thread time = new Thread(new Runnable(){
    public void run(){
        while(true){
            try{
                Thread.sleep(10);
            }catch(Exception e){ }
            if(timestart == false){
                repaint();
            }
        }
    }
});

```

ซึ่งในโปรแกรมจะทำการสร้างเธรดควบคุมตัวละครแต่ละตัวในเกมส์ให้สามารถทำงานพร้อมกันได้

การกำหนดให้ Panel สามารถตรวจสอบการกดปุ่มจากคีย์บอร์ดได้ ทำได้โดยการลงทะเบียนการดักจับเหตุการณ์ผ่านคลาสที่มีลักษณะเป็น anonymous class โดยใช้คำสั่ง addKeyListener() โดยวัตถุที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์การกดปุ่มและคอยตอบสนองเหตุการณ์ที่เกิดขึ้นคือวัตถุที่สร้างจากคลาส KeyAdapter โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด keyPressed () เพื่อตรวจสอบเหตุการณ์ของการกดปุ่ม โดยเหตุการณ์ที่เกิดขึ้นกับปุ่ม โดยกรณีกดปุ่ม A ตัวละครอัศวินจะเดินไปด้านซ้ายครึ่งละ กรณีกดปุ่ม 10D ตัวละครอัศวินจะเดินไปด้านขวาครึ่งละ กดปุ่ม 10UP ตัวละครอัศวินจะยิงก้อนหินไฟ ณ ตำแหน่งที่ตัวละครอยู่ปัจจุบัน เมธอด keyReleased () เพื่อตรวจสอบเหตุการณ์ของการปล่อยปุ่ม ตัวละครอัศวินจะแสดงภาพเริ่มต้น ดังคำสั่งด้านล่าง

```

this.addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        int a = e.getKeyCode();
        if(!paralyze1) {
            if(a==KeyEvent.VK_A) {
                m.x-=10;
                m.count++;
            }
            else if(a == KeyEvent.VK_D) {
                m.x+=10;
                m.count++;
            }
            if(m.count>3) {
                m.count=0;
            }
            else if(a == KeyEvent.VK_UP) {
                m.count=5;
                fireball.add(new Fireball(m.x, 550));
            }
        }
    }
    public void keyReleased(KeyEvent e) {
        m.count=0;
    }
});

```

เมธอด `paintComponent(Graphics g)` เป็นเมธอดที่ทำหน้าที่ในการวาดรูปบน Panel โดยใช้คำสั่ง `super.paintComponent(g);` เป็นคำสั่งสำหรับการเคลียร์หน้าจอเก่า และเตรียมพร้อมสำหรับหน้าจอใหม่และการวาดตัวละครต่างๆ ที่หน้าจอของเกมส์

หากต้องการตรวจสอบว่าตัวละครอัศวินกับก้อนหินไฟชนกันหรือยังจะใช้ส่วนของโปรแกรมดังต่อไปนี้

```
for(int i=0 ; i<fireball.size();i++){
    for(int j=0 ; j<bal.size();j++){
        if(Intersect(fireball.get(i).getbound(),bal.get(j).getbound())){
            bal.remove(j);
            fireball.remove(i);
            scor += 10;
            g.drawString("+10",m.x+100,650);
        }
    }
}
```

15.3 ตัวอย่างการพัฒนาโปรแกรมการแสดงผลแอนิเมชันโดยการเขียนโปรแกรมเชิงวัตถุ

ตัวอย่างการพัฒนาโปรแกรมการแสดงผลแอนิเมชันโดยการเขียนโปรแกรมเชิงวัตถุในส่วนนี้จะนำเสนอตัวอย่างการสร้างรถประกอบด้วยหลังการถ ตัวรถ ล้อ และรถสามารถสามารถเคลื่อนไหวได้ โดยตัวอย่างของหน้าจอของโปรแกรมมีดังนี้



```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Exercise extends JFrame {
    public Exercise() {
        add(new RaceCar());
    }
    public static void main(String[] args) {
        Exercise frame = new Exercise();
        frame.setTitle("Exercise");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 100);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setVisible(true);
    }
}
class RaceCar extends JPanel {
    private int xBase = 0;
    private Timer timer = new Timer(10, new Listener());
    public RaceCar() {
        timer.start();
        this.setFocusable(true);
        this.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if (e.isControlDown() && e.getKeyCode() == 61) {
                    if (timer.getDelay() > 5)
                        timer.setDelay(timer.getDelay() - 5);
                }
                else if (e.isControlDown() && e.getKeyCode() == 45)
                    timer.setDelay(timer.getDelay() + 1);
            }
        });
    }
}
```

```

    }
    });
}
class Listener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        repaint();
    }
}
public void paintComponent(Graphics g) {
    super.paintComponent(g);

    int yBase = getHeight();
    if (xBase > getWidth())
        xBase = -20;
    else
        xBase += 1;
    g.setColor(Color.BLACK);
    g.fillOval(xBase + 10, yBase - 10, 10, 10);
    g.fillOval(xBase + 30, yBase - 10, 10, 10);
    g.setColor(Color.GREEN);
    g.fillRect(xBase, yBase - 20, 50, 10);
    g.setColor(Color.RED);
    Polygon polygon = new Polygon();
    polygon.addPoint(xBase + 10, yBase - 20);
    polygon.addPoint(xBase + 20, yBase - 30);
    polygon.addPoint(xBase + 30, yBase - 30);
    polygon.addPoint(xBase + 40, yBase - 20);
    g.fillPolygon(polygon);
}
}

```

โดยโปรแกรมดังกล่าวมีรายละเอียดการทำงานดังนี้

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

```

ทำการ Import คลาสและเมธอดต่าง ๆ ที่จำเป็นสำหรับการประมวลผล

```

public class Exercise extends JFrame {
    public Exercise() {
        add(new RaceCar());
    }
}

```

สร้างคลาส Exercise ที่มีลักษณะเป็นเฟรมโดยการสืบทอดจากคลาส JFrame กำหนดคอนสตรักเตอร์ของคลาส Exercise ให้สร้างวัตถุใหม่ขึ้นมาจากคลาส RaceCar โดยการเรียกใช้คอนสตรักเตอร์ของ Racecar ด้วยคำสั่ง new RaceCar() จากนั้นให้นำวัตถุที่สร้างจากคลาส RaceCar เพิ่มเข้าไปที่คลาส Exercise ที่มีลักษณะเป็น Frame ด้วยคำสั่ง add(new RaceCar());

```

public static void main(String[] args) {
    Exercise frame = new Exercise();
    frame.setTitle("Exercise");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(200, 100);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setVisible(true);
}
}

```

Exercise frame = new Exercise();ทำการสร้างวัตถุจากคลาส Exercise

frame.setTitle("Exercise");กำหนดข้อความที่ส่วน Title ของเฟรม เป็น "Excercise"

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);กำหนดให้สามารถปิดโปรแกรมโดยการกดปุ่มเครื่องหมายกากบาทบนเฟรมได้

frame.setSize(200, 100); กำหนดขนาดความกว้างของเฟรมให้มีขนาด 200 * 100

frame.setLocationRelativeTo(null); กำหนดให้เมื่อประมวลผลชุดคำสั่งให้แสดงเฟรมที่ตำแหน่งกลางหน้าจอ

frame.setVisible(true); กำหนดให้เฟรมปรากฏที่หน้าจอ

```
class RaceCar extends JPanel {
    private int xBase = 0;
    private Timer timer = new Timer(10, new Listener());
    public RaceCar() {
        timer.start();
        this.setFocusable(true);
        this.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if (e.isControlDown() && e.getKeyCode() == 61) {
                    if (timer.getDelay() > 5)
                        timer.setDelay(timer.getDelay() - 5);
                }
                else if (e.isControlDown() && e.getKeyCode() == 45)
                    timer.setDelay(timer.getDelay() + 1);
            }
        });
    }
}
```

- สร้างคลาส RaceCar ที่มีลักษณะเป็น Panel โดยการสืบทอดจากคลาส JPanel ซึ่งเป็นบริเวณที่จะใช้วาดรถ

- กำหนดตัวแปร xBase มีค่าเป็น 0 ซึ่งเป็นตัวแปรสำหรับเก็บค่าเริ่มต้นของตัวรถ

- สร้างวัตถุจากคลาส Timer และคอยตรวจจับการเกิดเหตุการณ์กับวัตถุ timer โดยการลงทะเบียนการดักจับเหตุการณ์ให้กับ timer โดยใช้คำสั่ง Timer timer = new Timer(10, new Listener());

- วัตถุที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์และคอยตอบสนองเหตุการณ์ที่เกิดขึ้นคือวัตถุที่สร้างจากคลาสที่ทำการ Implements อินเตอร์เฟส ActionListener โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด actionPerformed()

- กำหนดคอนสตรักเตอร์ของคลาส RaceCar โดยเมื่อเริ่มสร้างวัตถุจากคลาส RaceCar โปรแกรมจะสั่งให้เวลาจากวัตถุ timer เริ่มเดินโดยใช้ timer.start(); ซึ่งเมื่อเวลาเริ่มเดินจะทำให้วัตถุที่สร้างจากคลาส Listener คอยตอบสนองเหตุการณ์ที่เกิดขึ้นโดยการทำการวาดรถที่ตำแหน่งต่าง ๆ ตามเวลาที่เปลี่ยนไป

- กำหนดให้ Panel สามารถตรวจสอบการกดปุ่มจากคีย์บอร์ดได้โดยการลงทะเบียนการดักจับเหตุการณ์ผ่านคลาสที่มีลักษณะเป็น anonymous class โดยใช้คำสั่ง addKeyListener() โดยวัตถุที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์การกดปุ่มและคอยตอบสนองเหตุการณ์ที่เกิดขึ้นคือวัตถุที่สร้างจากคลาส KeyAdapter โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด keyPressed () เพื่อตรวจสอบเหตุการณ์ของการกดปุ่ม โดยเหตุการณ์ที่เกิดขึ้นกับปุ่มคือ

1. ปุ่ม Ctrl และ ปุ่ม - ตรวจสอบได้จาก (e.isControlDown() && e.getKeyCode() == 61) จะทำให้การแสดงผลของรถข้างล่าง โดยใช้คำสั่ง timer.setDelay(timer.getDelay() - 5);

2. ปุ่ม Ctrl และ ปุ่ม + ตรวจสอบได้จาก (e.isControlDown() && e.getKeyCode() == 45)

จะทำให้การแสดงผลของรถเร็วขึ้น โดยการใช้คำสั่ง timer.setDelay(timer.getDelay() + 1);

```
class Listener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        repaint();
    }
}
```

คลาส Listener เป็นคลาสที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์และคอยตอบสนองเหตุการณ์ที่เกิดขึ้นโดยเป็นคลาสที่ Implements อินเตอร์เฟส ActionListener โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด actionPerformed() ภายในเมธอด actionPerformed() เมื่อเกิดการเปลี่ยนของเวลาจะทำการเรียกใช้เมธอด repaint() เพื่อทำการวาดตำแหน่งของรถใหม่ โดยการทำงานที่เมธอด paintComponent(Graphics g)

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);

    int yBase = getHeight();
    if (xBase > getWidth())
        xBase = -20;
    else
        xBase += 1;
}
```

เมธอด paintComponent(Graphics g) เป็นเมธอดที่หน้าที่ในการวาดรูปบน Panel โดยใช้คำสั่ง super.paintComponent(g); เป็นคำสั่งสำหรับการเคลียร์หน้าจอเก่า และเตรียมพร้อมสำหรับหน้าจอใหม่ ในชุดคำสั่งแรกจะเป็นการเช็คตำแหน่งเริ่มต้นของตัวรถว่าเกินความกว้างของ Panel หรือไม่หาก ไม่เกินจะทำการเพิ่มตำแหน่งของรถไป 1 ตำแหน่ง ในแกน x

```
g.setColor(Color.BLACK);
g.fillOval(xBase + 10, yBase - 10, 10, 10);
g.fillOval(xBase + 30, yBase - 10, 10, 10);
```

วาดล้อรถที่ 1 ที่ตำแหน่งถัดจากตัวรถในแนวแกน x เป็น xBase + 10 และถัดจากตัวรถในแนวแกน y เป็น yBase - 10 โดยความกว้างและความยาวของวงกลมเป็น 10 วาดล้อรถที่ 2 ที่ตำแหน่งถัดจากตัวรถในแนวแกน x เป็น xBase + 30 และถัดจากตัวรถในแนวแกน y เป็น yBase - 10 โดยความกว้างและความยาวของวงกลมเป็น 10 ล้อทั้งสองจะวาดด้วยสีดำจากการใช้คำสั่ง g.setColor(Color.BLACK);

```
g.setColor(Color.GREEN);
g.fillRect(xBase, yBase - 20, 50, 10);
```

วาดตัวรถสีเขียวที่มีความกว้างเท่ากับ ที่ตำแหน่ง 10 50และสูงเท่ากับxBase, yBase - 20

```
g.setColor(Color.RED);
Polygon polygon = new Polygon();
polygon.addPoint(xBase + 10, yBase - 20);
polygon.addPoint(xBase + 20, yBase - 30);
polygon.addPoint(xBase + 30, yBase - 30);
polygon.addPoint(xBase + 40, yBase - 20);
g.fillPolygon(polygon);
}
```

วาดหลังคารถสีแดงโดยการกำหนดจุดผ่าน Polygon แล้วแสดงผลด้วยการระบายสีบน Polygon ดังกล่าว ผลลัพธ์ที่ได้จากการรันโปรแกรมทั้งหมดคือ



15.4 บทสรุป

การเขียนโปรแกรมโดยการจำลองแนวคิดเชิงวัตถุเป็นอีกหนึ่งในภาษาโปรแกรมที่มีการพัฒนาอย่างต่อเนื่องและถูกนำไปใช้ในการพัฒนาในด้านต่างๆ ในปัจจุบัน เช่น การเขียนโปรแกรมผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชันต่าง ๆ หรือแม้กระทั่งการใช้แนวความคิดในการเขียนโปรแกรมเชิงวัตถุมาเพื่อใช้พัฒนาเกมส์ต่างๆ ในบทนี้จะเป็นการยกตัวอย่างการพัฒนาเกมส์อย่างง่ายโดยอาศัยแนวคิดเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุ

15.5 แบบฝึกหัดปฏิบัติการ

นักศึกษาที่ผ่านการเรียนวิชาการเขียนโปรแกรมเชิงวัตถุเกิดแรงบันดาลใจในการพัฒนาซอฟต์แวร์เกมส์ตัวใหม่หลังการไปดูภาพยนตร์สมเด็จพระนเรศวร จึงพัฒนาเกมส์ใหม่ชื่อว่า เกมส์ยุทธหัตถี (King Naresuan Fighting) โดยมีรูปแบบการเล่น คือ การขยับข้างฝั่งไทยเพื่อวิ่งชนข้างฝั่งพม่าเพื่อเก็บคะแนน

โปรแกรมจะแสดงรูปช้างก้านกล้วย (kankuay.gif) เป็นตัวละครหลักฝั่งไทยในตำแหน่งเริ่มต้นที่พิกัด 350 ของแกน y และตำแหน่ง 200 ของแกน x โดยรูปมีความกว้าง 50 และสูง 50 บน JPanel หน้าจอหลัก (JPanel) ขนาดกว้าง 350 สูง 400 จะประกอบด้วยส่วนของรูปช้างของไทย ส่วนของข้อความบอกคะแนนที่ตำแหน่ง $x = 15$ $y = 340$ และช้างหม่องของพม่า (mong.gif) ที่วิ่งลงมาทุก ๆ 10 ms ในแนวเส้นตรงจากบนลงล่าง

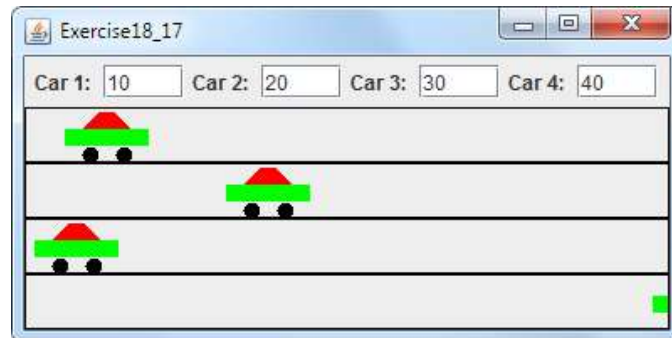
ในการเล่นแต่ละครั้งจะใช้ปุ่มลูกศรซ้าย (VK_UP) และลูกศรขวา (VK_DOWN) ปุ่มลูกศรบน (VK_UP) และลูกศรล่าง (VK_DOWN) ในการขยับตำแหน่งของช้างก้านกล้วย เพื่อที่จะวิ่งไปรบกับช้างหม่องของพม่าที่เคลื่อนที่จากบนลงล่างที่หล่นลงมา ณ ตำแหน่งต่าง ๆ ให้ทัน โดยการขยับตำแหน่งของช้างก้านกล้วย จะขยับได้ครั้งละ 2 pixels ในแนวแกน x หรือในแนวแกน y โดยหากช้างก้านกล้วย สามารถชนโดนช้างหม่องของพม่าได้ให้เพิ่มคะแนน 10 คะแนน และ random ข้างหม่องของพม่าที่ตำแหน่งใหม่ขึ้นมา



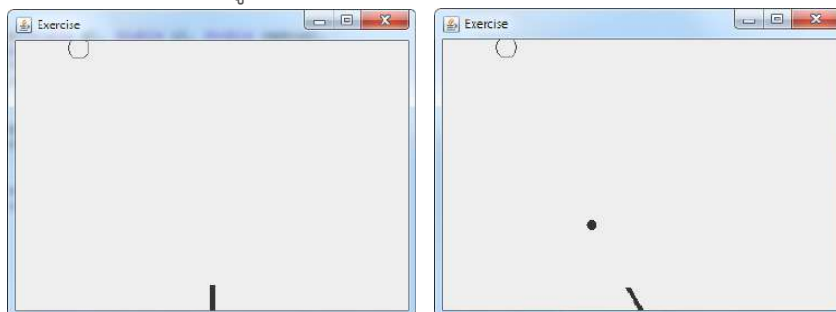
จากเงื่อนไขดังกล่าวจงตอบคำถามต่อไปนี้

- 1 เขียนคำสั่งเพื่ออ่านรูป kankuay.gif จากไดเรกทอรีปัจจุบันที่คลาสอยู่
- 2 เขียนชุดคำสั่งเพื่อกำหนดค่าเริ่มต้นของตัวแปรต่าง ๆ ภายในคลาสที่เป็นหน้าจอหลักของเกมส์ (คลาส GamePanel) โดยกำหนดตัวแปรและ constructor ที่จำเป็นในการแสดงผลหน้าจอดังกล่าว
- 3 เขียนชุดคำสั่งเพื่อกำหนดการดักจับเหตุการณ์ของการเคลื่อนที่ของช้างพม่า (Mong) โดยเขียนในลักษณะของ *anonymous class* และเขียนชุดคำสั่งที่กำหนดการดักจับเหตุการณ์ผ่านคีย์บอร์ดโดยเขียนในลักษณะของ *inner class* เพื่อควบคุมการเคลื่อนที่ของช้างก้านกล้วย
- 4 เขียนเมธอด `public boolean overlaps(double x1, double y1, double radius1, double x2, double y2)` เพื่อตรวจสอบว่าตำแหน่งของช้างก้านกล้วย (`double x1, double y1`) ที่มีรัศมี `radius1` อยู่ในบริเวณตำแหน่งของช้างหม่องของพม่า ซึ่งเป็นพื้นที่สี่เหลี่ยมที่มีจุดศูนย์กลางอยู่ที่ตำแหน่ง (`double x2, double y2`) และมีความกว้างเป็น 30 และสูง 30 หรือไม่ โดยคืนค่า `True` เมื่อช้างก้านกล้วย อยู่ในบริเวณรูปช้างหม่องของพม่า หรือคืนค่า `False` เมื่อช้างก้านกล้วย ไม่อยู่ในบริเวณรูปช้างหม่องของพม่า
- 5 ให้เขียนเมธอดที่ `override` เมธอด `paintComponent(Graphics g)` สำหรับวาด ส่วนของการแสดงคะแนน ส่วนของการแสดงรูปช้างก้านกล้วย และส่วนของการแสดงช้างหม่องของพม่า

2. จากคลาส Race car ในปฏิบัติการข้อที่ผ่านมาให้เพิ่มจำนวนรถเป็น 4 คันจำลองให้เป็นสนามแข่งรถ และสามารถกำหนดความเร็วให้รถแต่ละคันได้



3.เขียนโปรแกรมที่แสดงรูปบอลลอนในตำแหน่ง random ใน Panel ให้ใช้ปุ่มลูกศรซ้ายและลูกศรขวาในการเล็งตำแหน่งของปืนให้ตรงกับบอลลอน ใช้ปุ่มลูกศร up-arrow สำหรับยิงลูกกระสุน เมื่อลูกกระสุนถูกบอลลอนให้บอลลอนที่โดนกระสุนหายไป และ random บอลลอนที่ตำแหน่งใหม่ขึ้นมา



บทที่ 16 การพัฒนาเกมส์และแอปพลิเคชันโดยการเขียนโปรแกรมเชิงวัตถุ

วัตถุประสงค์

16.1 สามารถพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุ

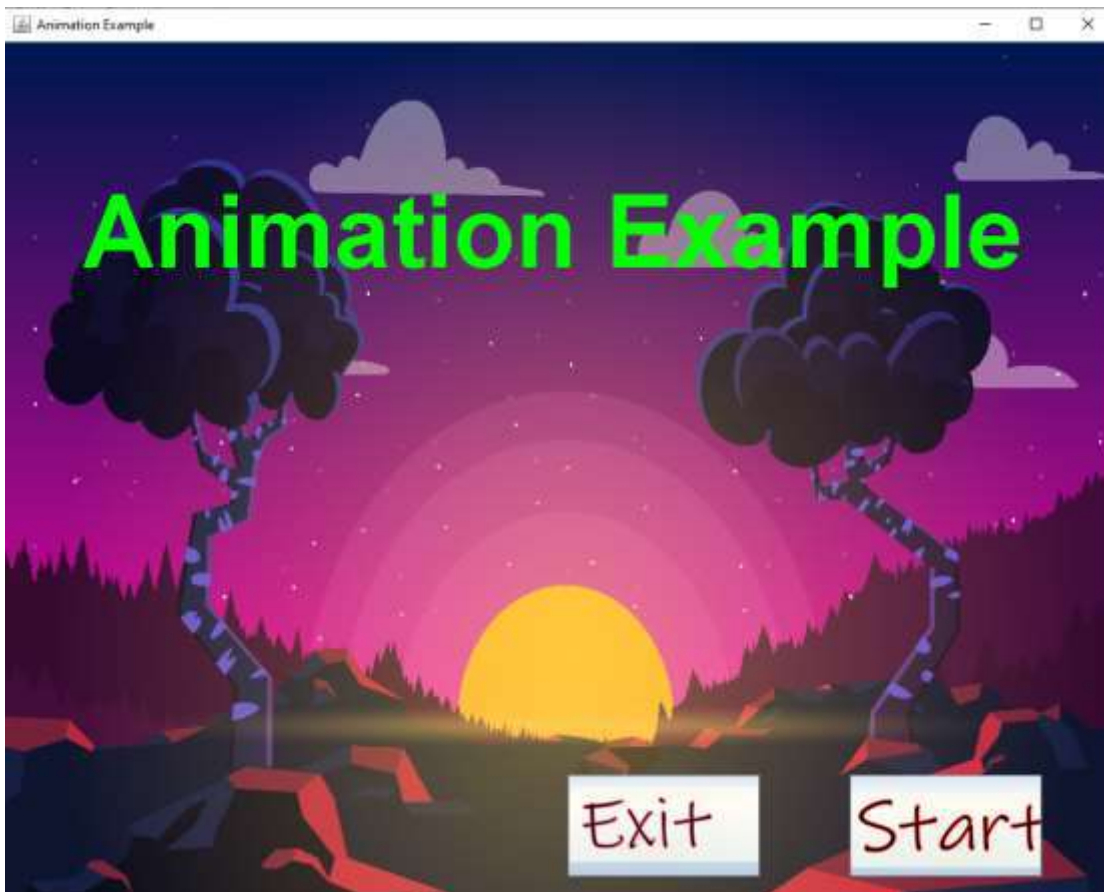
16.2 สามารถพัฒนาแอปพลิเคชันโดยการเขียนโปรแกรมเชิงวัตถุ

16.1 ความนำ

การเขียนโปรแกรมโดยการจำลองแนวคิดเชิงวัตถุเป็นอีกหนึ่งในภาษาโปรแกรมที่มีการพัฒนามาอย่างต่อเนื่องและถูกนำไปใช้ในการพัฒนาในด้านต่างๆ ในปัจจุบัน เช่น การเขียนโปรแกรมผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชันต่าง ๆ หรือแม้กระทั่งการใช้แนวความคิดในการเขียนโปรแกรมเชิงวัตถุมาเพื่อใช้พัฒนาเกมส์ต่างๆ ในบทนี้จะเป็นการยกตัวอย่างการพัฒนาเกมส์อย่างง่ายโดยอาศัยแนวคิดเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุ

16.2 ตัวอย่างการพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุ

ตัวอย่างการพัฒนาเกมส์โดยการเขียนโปรแกรมเชิงวัตถุในบทนี้จะนำเสนอเกมส์ที่ประกอบด้วยวัตถุลอยอยู่บนหน้าจอ แล้วตัวละครสามารถยิงวัตถุดังกล่าวเมื่อยิงโดนจะได้คะแนนเพิ่ม โดยสามารถยิงได้ภายในเวลาที่กำหนด นอกจากนี้ภายในเกมส์จะมีกับดักถ้าหากเรายิงโดนเราจะเสียเลือด โดยตัวอย่างของเกมส์มีดังนี้



```

import java.applet.Applet;
import java.applet.AudioClip;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.net.URL;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JPanel;

public class homegames extends JPanel{
    private ImageIcon feild = new ImageIcon(this.getClass().getResource("eezy_61-01.jpg"));
    private ImageIcon exit = new ImageIcon(this.getClass().getResource("exit.png"));
    private ImageIcon starts = new ImageIcon(this.getClass().getResource("start.png"));
    public JButton BStart = new JButton(starts);
    public JButton BExit1 = new JButton(exit);
    homegames(){
        setLayout(null);
        BExit1.setBounds(500, 650, 170,90);
        add(BExit1);
        add(BStart);
        BStart.setBounds(750,650,170,90);
        add(BStart);
    }
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawImage(feild.getImage(),0,0,1000,800,this);
        g.setColor(Color.GREEN);
        g.setFont(new Font("2005_iannnnnTKO",Font.CENTER_BASELINE,90));
        g.drawString("Animation Example",70,200);
    }
}

```

หมายเหตุ รูปภาพ background จากเว็บ <https://www.vecteezy.com/free-vector/game-background> Game Background Vectors by Vecteezy



```
import java.awt.geom.Rectangle2D;
import java.net.URL;

import javax.swing.ImageIcon;
import javax.swing.JPanel;

public class Fireball extends JPanel{
    public ImageIcon[] imfire = new ImageIcon[5];
    public int y;
    public int x;
    public int count=0;
    Fireball(int x,int y){
        for(int i=0;i<imfire.length;i++){
            String imageLocation = "b"+(i+1)+".png";
            imfire[i] = new ImageIcon(this.getClass().getResource(imageLocation));
        }

        this.x=x;
        this.y=y;
    }

    public void move(){
        this.y-=1;
    }
    public Rectangle2D getbound(){
        return (new Rectangle2D.Double(x,y,25,25));
    }
}
```



```
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.geom.Rectangle2D;
import java.net.URL;
import javax.swing.ImageIcon;
public class ball1 {
    Image img;
    public int x = 0;
    public int y= (int) ((Math.random()*300)+20);
    public int count = 0;
    ball1(){
        String imageLocation = "g0.png";
        URL imageURL = this.getClass().getResource(imageLocation);
        img = Toolkit.getDefaultToolkit().getImage(imageURL);
        runner.start();
    }
    Thread runner = new Thread(new Runnable() {
        public void run() {
            while(true){
                x += 2;
                if(x >= 1100){
                    img = null;
                    runner = null;
                    x = -500;
                    y = -500;
                }
                try{
                    runner.sleep(10);
                }catch(InterruptedException e){}
            }
        }
    });

    public Image getImage(){
        return img;
    }

    public int getX(){
        return x;
    }
    public int getY(){
        return y;
    }
    public Rectangle2D getbound(){
        return (new Rectangle2D.Double(x,y,45,45));
    }
}
```



```
import javax.swing.ImageIcon;
import javax.swing.JPanel;
public class meleon{
    public ImageIcon[] im = new ImageIcon[7];
    public int x;
    public int count = 0;
    meleon(){
        for(int i=0;i<im.length;i++){
            im[i] = new ImageIcon(this.getClass().getResource((i+1)+".png"));
        }
    }
}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.sound.sampled.Clip;
import javax.swing.JFrame;
public class PlayGames extends JFrame implements ActionListener{
    homegames homegames1 = new homegames();
    playstatel statel = new playstatel();
    gameover gover = new gameover();
    public PlayGames(){
        this.setSize(1000,800);
        this.add(homegames1);
        homegames1.BExit1.addActionListener(this);
        homegames1.BStart.addActionListener(this);
        statel.BExithome.addActionListener(this);
        statel.BPause.addActionListener(this);
        statel.Bresum.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()== homegames1.BStart){
            this.setLocationRelativeTo(null);
            this.remove(homegames1);
            this.setSize(1000,800);
            this.add(statel);
            statel.requestFocusInWindow();
            statel.timestart = false;
            statel.scor=0;
            statel.HP =3;
            statel.times = 100;
            statel.startball=false;
            statel.timestart=false;
        }else if(e.getSource() == statel.BExithome){
```

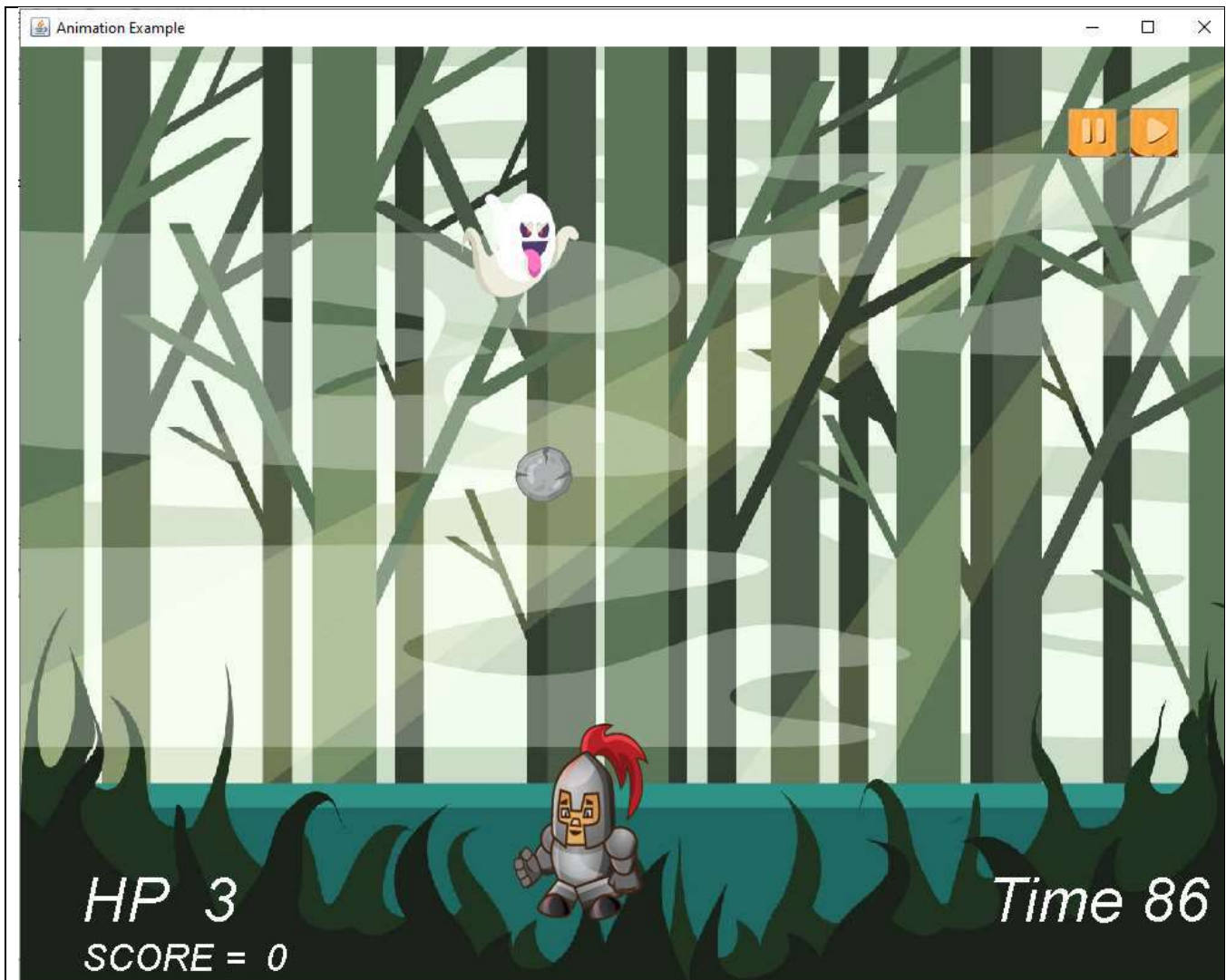


```

        System.exit(0);
    }else if (e.getSource() == homegames1.BExit1){
        System.exit(0);
    }else if(e.getSource() == statel.BPause){
        this.setLocationRelativeTo(null);
        this.setSize(1000,800);
        this.add(statel);
        statel.requestFocusInWindow();
        statel.t.suspend();
        statel.time.suspend();
        statel.actor.suspend();
        statel.tballs1.suspend();
    }else if(e.getSource() == statel.Bresum){
        this.setLocationRelativeTo(null);
        this.setSize(1000,800);
        this.add(statel);
        statel.requestFocusInWindow();
        statel.t.resume();
        statel.time.resume();
        statel.actor.resume();
        statel.tballs1.resume();
    }
    //this.validate();
    // this.repaint();
}

public static void main(String[] args) {
    JFrame jf = new PlayGames();
    jf.setSize(1000,800);
    jf.setTitle("Animation Example");
    jf.setDefaultCloseOperation(EXIT_ON_CLOSE);
    jf.setVisible(true);
    jf.setLocationRelativeTo(null);
}
}

```



```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Rectangle2D;
import java.net.URL;
import java.util.ArrayList;
import javax.swing.*;

public class playstate1 extends JPanel implements ActionListener{
    private final ImageIcon imgstate1 = new
    ImageIcon(this.getClass().getResource("swamp-01.jpg"));
    private final ImageIcon imgstate2 = new
    ImageIcon(this.getClass().getResource("eezy_61-01.jpg"));
    private final ImageIcon imgmeleon = new
    ImageIcon(this.getClass().getResource("meleon.png"));
    private final ImageIcon pause = new
    ImageIcon(this.getClass().getResource("p.png"));
    private final ImageIcon resum = new
    ImageIcon(this.getClass().getResource("play.png"));
    private final ImageIcon back = new
    ImageIcon(this.getClass().getResource("back.png"));
    meleon m = new meleon();

    homegames hg = new homegames();
    ImageIcon feildover = new
```

```

ImageIcon(this.getClass().getResource("a_cartoon_forest_election_by_88srenaissance
88-d4zim5f.jpg"));
ImageIcon img_paralyze = new ImageIcon(this.getClass().getResource("7.1.png"));
ImageIcon exitover = new ImageIcon(this.getClass().getResource("exit.png"));
ImageIcon restart = new ImageIcon(this.getClass().getResource("start.png"));
    JButton BStartover = new JButton(restart);
    JButton BExitover = new JButton(exitover);

private JLabel score = new JLabel();
    public JButton BPause = new JButton(pause);
public JButton BExithome = new JButton(back);
public JButton Bresum = new JButton(resum);

public ArrayList<Fireball> fireball = new ArrayList<Fireball>();
public ArrayList<ball1> bal = new ArrayList<ball1>();

public int times ;
public int HP = 3;
public int rs1 = 1;
public int rs2 = 2;
boolean timestart = true;
boolean startball = false;

private gameover gover = new gameover();
public int scor = 0;
boolean paralyze1 = false;
int time_paralyze=5;

Thread time = new Thread(new Runnable(){
    public void run(){
        while(true){
            try{
                Thread.sleep(10);
            }catch(Exception e){ }

            if(timestart == false){
                repaint();
            }
        }
    }
});

Thread actor = new Thread(new Runnable(){
    public void run(){
        while(true){
            try{
                Thread.sleep(1);
            }catch(Exception e){}
            repaint();
        }
    }
});

Thread tballs1 = new Thread(new Runnable(){
    public void run() {
        while(true){
            try{
                if(startball == false){
                    Thread.sleep((long) (Math.random()*10000)+2000);
                }
            }catch(InterruptedException e){

```

```

        e.printStackTrace();
    }
    if(startball == false){
        bal.add(new ball1());
    }
}
});

Thread paralyze = new Thread(new Runnable() {
    public void run() {
        while (true) {
            if(time_paralyze < 1){
                paralyze1 = false;
                time_paralyze = 5;
            }
            try{
                Thread.sleep(5000);
            }catch (InterruptedException e) {e.printStackTrace();}
        }
    }
});

Thread t = new Thread(new Runnable() {
    public void run() {
        while(true) {
            if(timestart == false){
                times = (times-1) ;
                if(paralyze1){
                    time_paralyze--;
                }
            }
            try{
                Thread.sleep(1000);
            }catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});

playstatel() {
    this.setFocusable(true);
    this.setLayout(null);
    BPause.setBounds(850,50,40,40);
    //BExithome.setBounds(850,30,40,40);
    Bresum.setBounds(900, 50, 40,40);
    BPause.addActionListener(this);
    BExithome.addActionListener(this);
    Bresum.addActionListener(this);
    BExithome.addActionListener(this);
    this.add(BPause);
    this.add(BExithome);
    this.add(score);
    this.add(Bresum);

    this.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            int a = e.getKeyCode();
            if(!paralyze1){

```

```

        if(a==KeyEvent.VK_A){
            m.x-=10;

            m.count++;
        }
        else if(a == KeyEvent.VK_D){
            m.x+=10;
            m.count++;
        }

        if(m.count>3){
            m.count=0;
        }
        else if(a == KeyEvent.VK_UP){
            m.count=5;
            fireball.add(new Fireball(m.x,550));
        }
    }

    public void keyReleased(KeyEvent e){
        m.count=0;
    }
});
m.x = 400;
time.start();
actor.start();
t.start();
tballs1.start();
paralyze.start();
}

public void paintComponent(Graphics g){
    super.paintComponent(g);
    if(times <= 0 || HP<=0){
        this.remove(BPause);
        this.remove(Bresum);
        this.remove(BExithome);
        this.setLayout(null);
        g.drawImage(feildover.getImage(),0,0,1000,800,this);
        g.setColor(Color.BLACK);
        g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,40));
        g.drawString("SCORE "+scor,380,200);
        g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,70));
        g.drawString("GAME OVER",290,150);
        g.drawImage(imgmeleon.getImage(), 580, 360, 400, 400, this);

    }else if(times <= 50){
        g.drawImage(imgstate2.getImage(),0,0,1000,800,this);
        if(paralyze1){
            g.setColor(Color.WHITE);
            g.setFont(new Font("Hobo Std",Font.BOLD,50));
            g.drawImage(img_paralyze.getImage(), m.x, 550,100,150, this);
            g.drawString("AHHHH !!!", m.x+100, 560);
        }else{
            g.drawImage(m.im[m.count].getImage(), m.x, 550,110,160, this);
        }
    }
    if(m.x<0){
        m.x=this.getWidth()-10;
    }
    if(m.x>this.getWidth()){
        m.x=20;
    }
    for(int i=0;i<fireball.size();i++){

```

```

        Fireball ba = fireball.get(i);
        g.drawImage(ba.imfire[ba.count%5].getImage(), ba.x, ba.y, 50, 50,
null);

        ba.move();
        ba.count++;
        if(ba.y<0){
            fireball.remove(i);
        }
    }

    //=====ball1=====
    for(int i=0 ; i<bal.size();i++){
        g.drawImage( bal.get(i).getImage()
, bal.get(i).getX(), bal.get(i).getY(), 100, 100, this);
    }

    for(int i=0 ; i<fireball.size();i++){
        for(int j=0 ; j<bal.size();j++){
            if(Intersect(fireball.get(i).getbound(), bal.get(j).getbound())){
                bal.remove(j);
                fireball.remove(i);
                scor += 10;
                g.drawString("+10", m.x+100, 650);
            }
        }
    }

    g.setFont(new Font("Hobo Std", Font.HANGING_BASELINE, 30));
    g.setColor(Color.WHITE);
    g.drawString("SCORE = "+scor, 50, this.getHeight()-10);
    g.setFont(new Font("Hobo Std", Font.HANGING_BASELINE, 50));
    g.drawString("Time "+times, this.getWidth()-200, this.getHeight()-50);
    g.setColor(Color.WHITE);
    g.drawString("HP "+HP, 50, this.getHeight()-50);
    }else if(times <= 0 || HP<=0){
this.remove(BPause);
this.remove(Bresum);
this.remove(BExithome);
        this.setLayout(null);
        g.drawImage(feildover.getImage(), 0, 0, 1000, 800, this);
        g.setColor(Color.BLACK);
        g.setFont(new Font("Hobo Std", Font.HANGING_BASELINE, 40));
        g.drawString("SCORE "+scor, 380, 200);
        g.setFont(new Font("Hobo Std", Font.HANGING_BASELINE, 70));
        g.drawString("GAME OVER", 290, 150);
        g.drawImage(imgmeleon.getImage(), 580, 360, 400, 400, this);
    }else{
        g.drawImage(imgstatel.getImage(), 0, 0, 1000, 800, this);
        if(paralyzel){
            g.setColor(Color.RED);
            g.setFont(new Font("Hobo Std", Font.BOLD, 50));
            g.drawImage(img_paralyze.getImage(), m.x, 550, 100, 150, this);
            g.drawString("-10", m.x+100, 650);
            g.drawString("AHHHH !!!", m.x+100, 560);
        }else{
            g.drawImage(m.im[m.count].getImage(), m.x, 550, 110, 160, this);
        }
    }
    if(m.x<0){
        m.x=this.getWidth()-10;
    }
    if(m.x>this.getWidth()){
        m.x=20;
    }
}

```

```

        for(int i=0;i<fireball.size();i++){
            Fireball ba = fireball.get(i);
            g.drawImage(ba.imfire[ba.count%5].getImage(), ba.x,
ba.y,50,50, null);
            ba.move();
            ba.count++;
            if(ba.y<0){
                fireball.remove(i);
            }
        }

//=====ball1=====
        for(int i=0 ; i<bal.size();i++){
g.drawImage(bal.get(i).getImage(),bal.get(i).getX(),bal.get(i).getY(),100,100,this
);
        }
        for(int i=0 ; i<fireball.size();i++){
            for(int j=0 ; j<bal.size();j++){
                if(Intersect(fireball.get(i).getbound(),bal.get(j).getbound())){
                    bal.remove(j);
                    fireball.remove(i);
                    scor += 10;
                    g.drawString("+10",m.x+100,650);
                }
            }
        }
        g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,30));
        g.setColor(Color.WHITE);
        g.drawString("SCORE = "+scor,50, this.getHeight()-10);
        g.setFont(new Font("Hobo Std",Font.HANGING_BASELINE,50));
        g.drawString("Time "+times,this.getWidth()-200,this.getHeight()-50);
        g.setColor(Color.WHITE);
        g.drawString("HP "+HP,50,this.getHeight()-50);
    }

    public boolean Intersect(Rectangle2D a, Rectangle2D b){
        return (a.intersects(b));
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getSource()== BStartover){
            this.setSize(1000,800);
            this.add(hg);
            this.setLocation(null);
            timestart = true;
            startball = true;
        }else if(e.getSource() == BExitover){
            System.exit(0);
        }
    }
}

```

16.3 บทสรุป

การเขียนโปรแกรมโดยการจำลองแนวคิดเชิงวัตถุเป็นอีกหนึ่งในภาษาโปรแกรมที่มีการพัฒนาอย่างต่อเนื่องและถูกนำไปใช้ในการพัฒนาในด้านต่างๆ ในปัจจุบัน เช่น การเขียนโปรแกรมผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชันต่าง ๆ หรือแม้กระทั่งการใช้แนวความคิดในการเขียนโปรแกรมเชิงวัตถุมาเพื่อใช้พัฒนาเกมส์ต่างๆ ในบทนี้จะเป็นการยกตัวอย่างการพัฒนาเกมส์อย่างง่ายโดยอาศัยแนวคิดเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุ

16.4 แบบฝึกหัดปฏิบัติการ

นักศึกษาที่ผ่านการเรียนวิชาการเขียนโปรแกรมเชิงวัตถุเกิดแรงบันดาลใจในการพัฒนาซอฟต์แวร์เกมส์ตัวใหม่หลังการไปดูภาพยนตร์สมเด็จพระนเรศวร จึงพัฒนาเกมส์ใหม่ชื่อว่า เกมส์ยุทธหัตถี(King Naresuan Fighting) โดยมีรูปแบบการเล่น คือ การขยับช้างฝังไทยเพื่อวิ่งชนช้างฝังพม่าเพื่อเก็บคะแนน

โปรแกรมจะแสดงรูปช้างก้านกล้วย (kankuay.gif) เป็นตัวละครหลักฝังไทยในตำแหน่งเริ่มต้นที่พิกัด 350 ของแกน y และตำแหน่ง 200 ของแกน x โดยรูปมีความกว้าง 50 และสูง 50 บน JPanel หน้าจอหลัก (JPanel) ขนาดกว้าง 350 สูง 400 จะประกอบด้วยส่วนของรูปช้างของฝังไทย ส่วนของข้อความบอกคะแนนที่ตำแหน่ง $x = 15$ $y = 340$ และช้างหม่องของพม่า (mong.gif) ที่วิ่งลงมาทุก ๆ 10 ms ในแนวเส้นตรงจากบนลงล่าง

ในการเล่นแต่ละครั้งจะใช้ปุ่มลูกศรซ้าย (VK_UP) และลูกศรขวา (VK_DOWN) ปุ่มลูกศรบน (VK_UP) และลูกศรล่าง (VK_DOWN) ในการขยับตำแหน่งของช้างก้านกล้วย เพื่อที่จะวิ่งไปพบกับช้างหม่องของพม่าที่เคลื่อนที่จากบนลงล่างที่หล่นลงมา ณ ตำแหน่งต่าง ๆ ให้ทัน โดยการขยับตำแหน่งของช้างก้านกล้วย จะขยับได้ครั้งละ 2 pixels ในแนวแกน x หรือในแนวแกน y โดยหากช้างก้านกล้วย สามารถชนโดนช้างหม่องของพม่าได้ให้เพิ่มคะแนน 10 คะแนน และ random ช้างหม่องของพม่าที่ตำแหน่งใหม่ขึ้นมา



จากเงื่อนไขดังกล่าวจงตอบคำถามต่อไปนี้

- 1 เขียนคำสั่งเพื่ออ่านรูป kankuay.gif จากไดเรกทอรีปัจจุบันที่คลาสอยู่
- 2 เขียนชุดคำสั่งเพื่อกำหนดค่าเริ่มต้นของตัวแปรต่าง ๆ ภายในคลาสที่เป็นหน้าจอหลักของเกมส์ (คลาส GamePanel) โดยกำหนดตัวแปรและ constructor ที่จำเป็นในการแสดงผลหน้าจอดังกล่าว

3 เขียนชุดคำสั่งเพื่อกำหนดการดักจับเหตุการณ์ของการเคลื่อนที่ของช้างพม่า (Mong) โดยเขียนในลักษณะของ *anonymous class* และเขียนชุดคำสั่งที่กำหนดการดักจับเหตุการณ์ผ่านคีย์บอร์ดโดยเขียนในลักษณะของ *inner class* เพื่อควบคุมการเคลื่อนที่ของช้างก้านกล้วย

4 เขียนเมธอด `public boolean overlaps(double x1, double y1, double radius1, double x2, double y2)` เพื่อตรวจสอบว่าตำแหน่งของช้างก้านกล้วย (`double x1, double y1`) ที่มีรัศมี `radius1` อยู่ในบริเวณตำแหน่งของช้างหม่องของพม่า ซึ่งเป็นพื้นที่สี่เหลี่ยมที่มีจุดศูนย์กลางอยู่ที่ตำแหน่ง (`double x2, double y2`) และมีความกว้างเป็น 30 และสูง 30 หรือไม่ โดยคืนค่า `True` เมื่อช้างก้านกล้วย อยู่ในบริเวณรูปช้างหม่องของพม่า หรือคืนค่า `False` เมื่อช้างก้านกล้วย ไม่อยู่ในบริเวณรูปช้างหม่องของพม่า

5 ให้เขียนเมธอดที่ `override` เมธอด `paintComponent(Graphics g)` สำหรับวาด ส่วนของการแสดงคะแนน ส่วนของการแสดงรูปช้างก้านกล้วย และส่วนของการแสดงช้างหม่องของพม่า

เอกสารอ้างอิง

1. Y. Daniel Liang, Introduction to Java Programming, 8th Edition, Pearson, 2011.
2. USING JAVA WITH 101 EXAMPLES. Atiwong Suchato. 1. Java (Computer program language). 005.133. ISBN 978-616-551-368-5. First Printing: July, 2011.
3. C. Thomas WU, McGRAW-Hill, An Introduction To Object-Oriented Programming with JAVA (4th Edition)
4. Cay Horstmann. Computing Concepts with Java Essentials (3rd Edition)
5. Kindler, E.; Krivy, I. (2011). "Object-Oriented Simulation of systems with sophisticated control". International Journal of General Systems: 313–343.
6. Lewis, John; Loftus, William (2008). Java Software Solutions Foundations of Programming Design 6th ed. Pearson Education Inc. ISBN 978- 0- 321- 53205- 3. , section 1. 6 " Object- Oriented Programming"
7. Deborah J. Armstrong. The Quarks of Object-Oriented Development. A survey of nearly 40 years of computing literature which identified a number of fundamental concepts found in the large majority of definitions of OOP, in descending order of popularity: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, and Abstraction.
8. John C. Mitchell, Concepts in programming languages, Cambridge University Press, 2003, ISBN 0- 521- 78098- 5, p. 278. Lists: Dynamic dispatch, abstraction, subtype polymorphism, and inheritance.
9. Michael Lee Scott, Programming language pragmatics, Edition 2, Morgan Kaufmann, 2006, ISBN 0-12-633951-1, p. 470. Lists encapsulation, inheritance, and dynamic dispatch.
10. Pierce, Benjamin (2002). Types and Programming Languages. MIT Press. ISBN 978-0-262-16209-8., section 18.1 "What is Object-Oriented Programming?" Lists: Dynamic dispatch, encapsulation or multi- methods (multiple dispatch), subtype polymorphism, inheritance or delegation, open recursion ("this"/"self")
11. Jacobsen, Ivar; Magnus Christerson; Patrik Jonsson; Gunnar Overgaard (1992). Object Oriented Software Engineering. Addison-Wesley ACM Press. pp. 43–69. ISBN 978-0-201-54435-0.
12. Dr. Alan Kay on the Meaning of "Object-Oriented Programming". 2003. Retrieved 11 February 2010.
13. Dahl, Ole Johan (2004). "The Birth of Object Orientation: The Simula Languages" (PDF). From Object-Orientation to Formal Methods. Lecture Notes in Computer Science. 2635. pp. 15–25. CiteSeerX 10.1.1.133.6730. doi:10.1007/978-3-540-39993-3_3. ISBN 978-3-540-21366-6. Retrieved 3 March 2018.
14. Bertrand Meyer (2009). Touch of Class: Learning to Program Well with Objects and Contracts. Springer Science & Business Media. p. 329. Bibcode:2009tclp.book.....M. ISBN 9783540921448.
15. อรพิน ประวัติดิวิสุทธิ , “คู่มือการเขียนโปรแกรมด้วยภาษา Java ฉบับสมบูรณ์” , โปรวีชั่น, 2556.
16. สุดา เขียวมนตรี, “คู่มือการเรียนรู้เขียนโปรแกรมภาษา Java ฉบับสมบูรณ์” , ไอทีซีฯ, 2555.

17. ดร.วีรศักดิ์ ชิงถาวร. JAVA Programming Volume I และ JAVA Programming Volume II ซีอีโอเคชั่น, 2545
18. เอกสารประกอบการสอน อ.ธนิศา เครือไวยวรรณ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
19. กิตติ ภัคดีวัฒนะกุล และกิตติพงษ์ กลมกล่อม UML วิเคราะห์และออกแบบเชิงวัตถุ (พิมพ์ครั้งที่ 2) บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด KTP กรุงเทพฯ (2547)

ดัชนี

การโอเวอร์โหลดเมธอด	66
การโปรแกรมแบบเน้นกรรมวิธี	1
การโปรแกรมเชิงวัตถุ	1
การห่อหุ้ม)Encapsulation(2,50,127
การสืบทอด)Inheritance(2,106
การมีได้หลายรูปแบบ)Polymorphism(2
การมองแบบนามธรรม)Abstraction(2
การแปลงข้อมูลที่กว้างขึ้น (Widening conversion)	18
การแปลงข้อมูลที่แคบลง (narrowing conversion)	18
การเข้าถึงข้อมูลของวัตถุ	46
ข้อมูลเบื้องต้น)Primitive data types(8
ชนิดข้อมูลแบบอ้างอิง)reference data type(12
ข้อมูลในระดับความคิด)Abstract data types(8
คุณลักษณะ	2,43
โครงสร้างข้อมูลแบบเชิงเส้น	8
โครงสร้างข้อมูลแบบไม่ใช่เชิงเส้น	8
คำสั่งทดสอบเงื่อนไข การตัดสินใจ	26
คำสั่งการวนซ้ำ	32
คลาส	43
คอนสตรัคเตอร์	48
คลาสนามธรรม(Abstract Class)	136
คลาส Graphics	183
เธรด(Thread)	246
เมธอด	61
โพลีมอร์ฟิซึม(Polymorphism)	136
พฤติกรรม	2,43
ตัวแปรอ้างอิงวัตถุ	45
ตัวดำเนินการทางคณิตศาสตร์	14
ตัวแปร(Variables)	9
ตัวแปรค่าคงที่(Constants)	10
วัตถุ	43
อาร์เรย์	77
อาเรย์ของตัวแปรพื้นฐาน	79
อาเรย์สองมิติ	84
โอเวอร์โหลดดั่ง)Overloading Methods(113
แอปเพล็ต	223

โอเวอร์ไรด์(Overriding Methods)	112
อาร์เรย์ลิสต์	87
อาร์เรย์ของตัวแปรชนิดอ้างอิง	89
แอปพลิเคชัน	43
abstract	61
AWT	161
Constructor Chaining	110
class method	62
Dynamic binding	136
Downcasting	139
Event-Driven	195
encapsulation	128
Event	198
final	61
inner class	203
instance method	62
Listener	196
private	50,61
protected	50,61
public	50,61
Recursive Method	69
super	108
Static binding	138
Swing	161
superclass	111
this	108
Upcasting	139