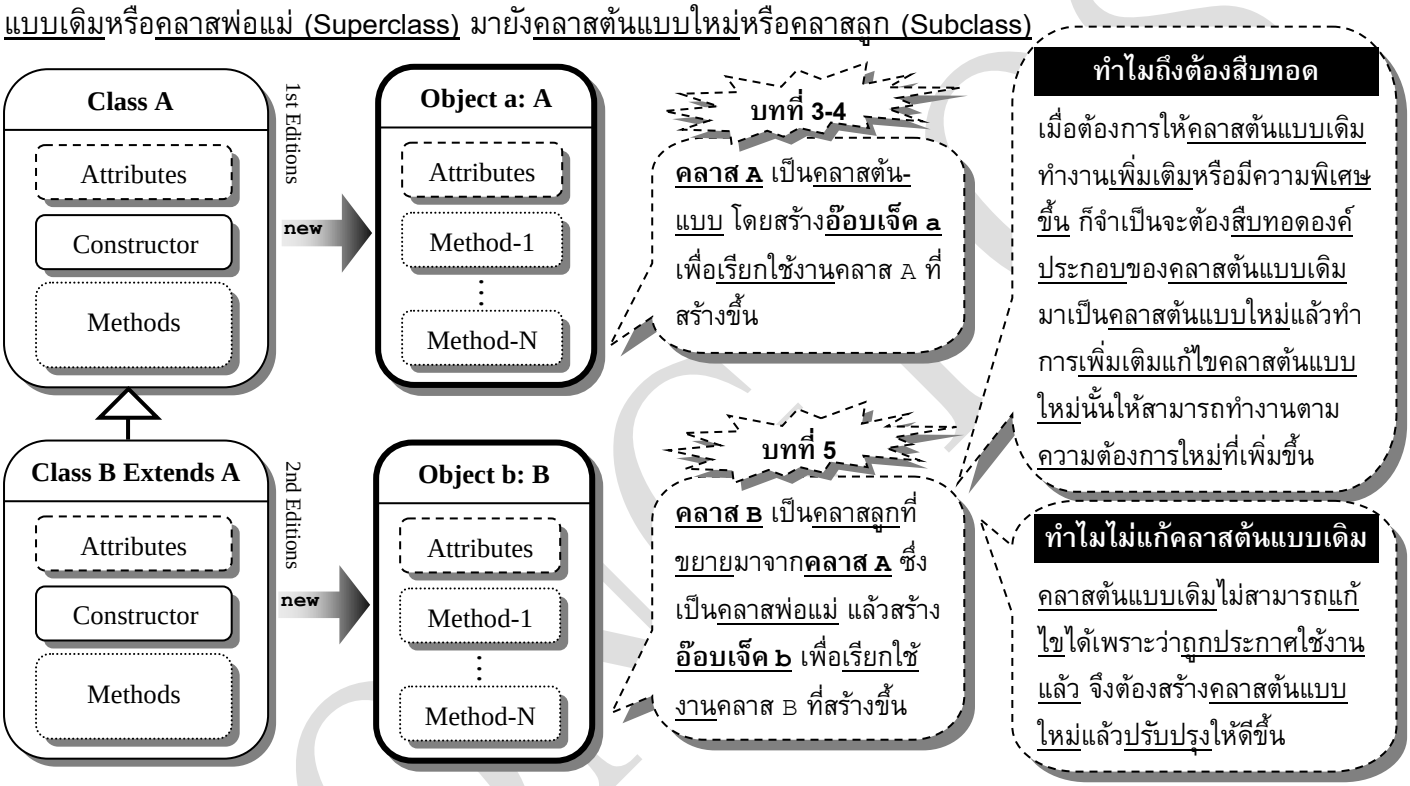
CHAPTER 5 การสืบทอดและอินเทอร์เฟซ (Inheritances and Interfaces)

1 การสืบทอด (Inheritances)

1. แนวคิดของการสืบทอด (Inheritance Concepts)

การสืบทอด คือ การขยายหรือการคัดลอกองค์ประกอบต่างๆ ไม่ว่าจะเป็นแอตทริบิว ตัวสร้าง และเมท็อด ของคลาสต้นแบบเดิมหรือคลาสพ่อแม่ (Superclass) มายังคลาสต้นแบบใหม่หรือคลาสลูก (Subclass)

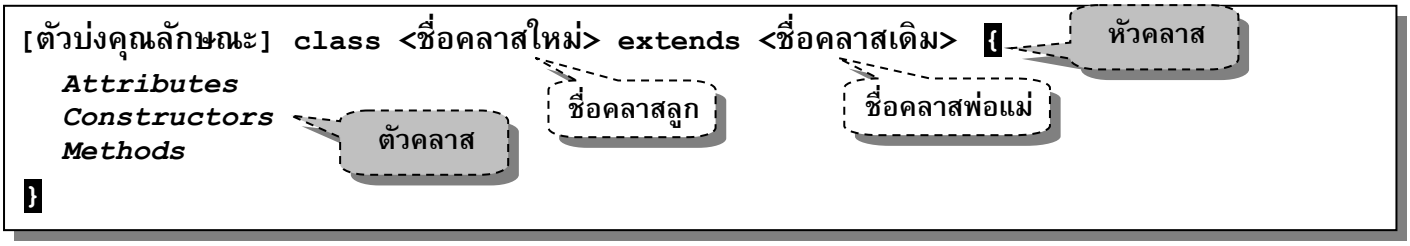


ลักษณะของคลาสต้นแบบใหม่ (คลาสลูก) ที่สืบทอดมาจากคลาสต้นแบบเดิม (คลาสพ่อแม่)

1) คลาสต้นแบบใหม่จะเสมือนกับมีองค์ประกอบทุกประการที่คลาสต้นแบบเดิมมี (เสมือนกับมีแอตทริบิว ตัวสร้าง และเมท็อดเช่นเดียวกับคลาสต้นแบบเดิม)
   - องค์ประกอบใดของคลาสต้นแบบเดิมที่**ดีอยู่แล้ว** คลาสต้นแบบใหม่จะใช้ตามนั้นไม่มีการแก้ไข
   - องค์ประกอบใดของคลาสต้นแบบเดิมที่**ยังไม่ดี** คลาสต้นแบบใหม่จะต้องปรับปรุงแก้ไขให้ดีขึ้น
   - องค์ประกอบใดของคลาสต้นแบบเดิมที่**ยังไม่มี** คลาสต้นแบบใหม่จะต้องเพิ่มเติมเข้าไป
2) การแก้ไขหรือปรับปรุงลักษณะเดิมที่ได้สืบทอดมาจากคลาสต้นแบบเดิม (ทั้งแอตทริบิวและเมท็อด) ให้มีคุณสมบัติเพิ่มเติมหรือเปลี่ยนแปลงไปจากเดิมจะเรียกกระบวนการนั้นว่า**การคร่อมทับ (Overriding)**

2. การประกาศและสร้างคลาสต้นแบบใหม่ด้วยการสืบทอด

คลาสต้นแบบใหม่ (คลาสลูก) ที่เกิดจากการสืบทอด จะมีองค์ประกอบหลัก 2 ส่วนเช่นเดียวกับคลาสต้นแบบเดิม (คลาสพ่อแม่) ได้แก่ หัวคลาสและตัวคลาส และภายในจะมีองค์ประกอบย่อย 3 ส่วนได้แก่ แอตทริบิว ตัวสร้าง และเมท็อด ซึ่งมีรายละเอียดในการประกาศและสร้างคลาสต้นแบบใหม่ดังต่อไปนี้



1) หัวคลาสต้นแบบใหม่ จะต้องเขียนเริ่มด้วยชื่อคลาสใหม่ (คลาสลูก) ตามด้วยคีย์เวิร์ดคำว่า `extends` และตามด้วยชื่อคลาสเดิม (คลาสพ่อแม่) โดยห้ามสลับตำแหน่งกันเด็ดขาด เช่น `GreanStudent extends Student` เป็นต้น
2) คีย์เวิร์ดคำว่า `extends` จะต้องเติม `s` เสมอ ห้ามลืมเด็ดขาด

โจทย์ข้อที่ 1 [ระดับง่าย] จงระบุส่วนประกอบต่าง ๆ ของคลาสจากโปรแกรมต่อไปนี้ *(15 คะแนน)*

```
1  public class Data {
2      protected int x;
3      public Data() {
4          this.x = 0;
5      }
6      public Data(int x) {
7          this.x = x;
8      }
9      public Data(Data d) {
10         this.x = d.x;
11     }
12     public String toString() {
13         return this.x + "";
14     }
15 }
```

(2) (3) (4) (1)

ให้แอตทริบิวและเมท็อดในคลาสแม่เป็น protected เพื่อใช้ในการสืบทอด

```
1  public class Data2 extends Data {
2      protected double y;
3      public Data2() {
4          this(0, 0.0);
5      }
6      public Data2(int x, double y) {
7          super(x);  this.y = y;
8      }
9      public Data2(Data2 d) {
10         this(d.x, d.y);
11     }
12     public String toString() {
13         return super.toString()+this.y;
14     }
15     protected void setY(double y) {
16         this.y = y;
17     }
18 }
```

(6) (7) (5) (8) (9)

ผลลัพธ์ที่ได้

```
1  public class RunData {
2      public static void main(String[] ar){
3          Data a = new Data(4);
4          String s = a.toString();
5          System.out.println(s);
6          Data2 b = new Data2(4, 1.5);
7          System.out.println(b.toString());
8          b.setY(23.7);
9          System.out.println(b.toString());
10     }
11 }
```

(11) (12) (10)

3. นิยามและการใช้คีย์เวิร์ด this และ super

1) คีย์เวิร์ด this ใช้อ้างอิงองค์ประกอบต่างๆ (แอตทริบิว คอนสตรักเตอร์ และเมท็อด) ที่อยู่ในคลาสต้นแบบใหม่ (คลาสลูก) หรือคลาสต้นแบบปัจจุบัน ซึ่งมีรายละเอียดดังนี้

วิธีที่	การอ้างอิง / เรียกใช้งาน	รูปแบบคำสั่ง
1	การอ้างอิงแอตทริบิวที่คลาสต้นแบบใหม่	`this.`<ชื่อแอตทริบิว>
2	การอ้างอิงคอนสตรักเตอร์ที่คลาสต้นแบบใหม่	`this(...)`
3	การอ้างอิงเมท็อดที่คลาสต้นแบบใหม่	`this.`<ชื่อเมท็อด>`(...)`

2) คีย์เวิร์ด super ใช้อ้างอิงองค์ประกอบต่างๆ ที่อยู่ในคลาสต้นแบบเดิม (คลาสพ่อแม่) ซึ่งมีรายละเอียดดังนี้

วิธีที่	การอ้างอิง / เรียกใช้งาน	รูปแบบคำสั่ง
1	การอ้างอิงแอตทริบิวที่คลาสต้นแบบเดิม	`super.`<ชื่อแอตทริบิว>
2	การอ้างอิงคอนสตรักเตอร์ที่คลาสต้นแบบเดิม	`super(...)`
3	การอ้างอิงเมท็อดที่คลาสต้นแบบเดิม	`super.`<ชื่อเมท็อด>`(...)`

3) ขอบเขตข้อมูลของการเรียกใช้คีย์เวิร์ด this และ super

```
1  public class Bag extends B {
2      protected int x;
3      public Bag() {
4          this(0);
5      }
6      public Bag(int x) {
7          this.x = x;
8      }
9      protected void show() {
10         System.out.println(x);
11     }
12 }
```

```
1  public class BigBag extends Bag {
2      protected long y;
3      public BigBag() {
4          super.x = 0;  this.y = 0L;
5      }
6      public BigBag(int x, long y) {
7          super(x);  this.y = y;
8      }
9      public String toString() {
10         return x + "," + y;
11     }
12 }
```

(1) องค์ประกอบใดๆ ก็ตาม ถ้าเรียกใช้โดยไม่ได้ระบุคีย์เวิร์ด `this` และ `super` จะหมายถึง `this`

(2) องค์ประกอบใดที่ใช้คีย์เวิร์ด `this` จะอ้างอิงองค์ประกอบนั้นภายในคลาสต้นแบบใหม่ (คลาสลูก) ก่อน ถ้าไม่พบองค์ประกอบดังกล่าว ก็จะกระโดดขึ้นไปเรียกใช้องค์ประกอบนั้นที่คลาสต้นแบบเดิม (คลาสพ่อแม่)

(3) องค์ประกอบใดที่ใช้คีย์เวิร์ด `super` จะเรียกใช้องค์ประกอบนั้นที่คลาสต้นแบบเดิม (คลาสพ่อแม่) เลยทันที โดยจะไม่เรียกใช้องค์ประกอบนั้นที่คลาสต้นแบบใหม่ (คลาสลูก) แต่อย่างใด

(4) ไม่ว่าจะใช้คีย์เวิร์ด `this` หรือ คีย์เวิร์ด `super` ถ้ายังไม่พบองค์ประกอบที่ต้องการ ก็จะมีการกระโดดขึ้นไปยังคลาสต้นแบบที่อยู่เหนือจากคลาสต้นแบบที่อ้างอิงขึ้นไปเรื่อยๆ เช่น คลาสปู่ย่า คลาสทวด เป็นต้น และจะกระโดดแบบนี้ไปจนถึงคลาสบรรพบุรุษ (Ancestor Class)

(5) ถ้าพบองค์ประกอบที่ต้องการเรียกใช้ในหลายคลาสต้นแบบที่ระดับชั้นแตกต่างกัน ก็จะเรียกใช้องค์ประกอบนั้นที่คลาสต้นแบบที่มีระดับชั้นต่ำสุด (ระดับล่าสุดที่เจอ) เพียงระดับเดียวเท่านั้น

4) ข้อตกลงในการใช้คีย์เวิร์ด this และ super

(1) ถ้าคอนสตรักเตอร์มีการใช้คีย์เวิร์ด `super` จะต้องเรียกใช้ `super` เป็นคำสั่งแรกก่อนเรียก `this`

(2) ไม่สามารถใช้คีย์เวิร์ด `this` และ `super` กับแอตทริบิวและเมท็อดที่มี `static`

โจทย์ข้อที่ 2 [ระดับง่าย] จงประกาศและสร้างแอตทริบิวและตัวสร้างของคลาส `Book` (หนังสือหนึ่งเล่ม) ที่สืบทอดมาจากคลาส `Paper` (กระดาษหนึ่งแผ่น) โดยมีรายละเอียดดังต่อไปนี้ *(15 คะแนน)*

```
1  public class Paper {
2      protected String title;
3      protected String author;
4      public Paper() {
5          this("", "");
6      }
7      public Paper(String t, String a) {
8          this.title = t;  this.author = a;
9      }
10     public Paper(Paper p) {
11         this(p.title, p.author);
12     }
13     ...
14 }
```

แอตทริบิว `title` ใช้แทนชื่อหัวข้อเรื่องของกระดาษ

แอตทริบิว `author` ใช้แทนชื่อคนเขียนกระดาษ

ในการเขียนตัวสร้างของคลาส `Book` นี้ ให้เลียนแบบรูปแบบของการเขียนตัวสร้างจากคลาส `Paper`

คลาส `Book`

1) แอตทริบิวประจำอ๊อบเจ็คแบบ `protected` ชื่อ `price` เพื่อแทนราคาของหนังสือ *(1.5 คะแนน)*
2) แอตทริบิวประจำอ๊อบเจ็คแบบ `protected` ชื่อ `page` เพื่อแทนจำนวนหน้าของหนังสือ *(1.5 คะแนน)*
3) ตัวสร้างแบบ **Default Constructor** เพื่อกำหนดค่าให้กับแอตทริบิวทุกตัว *(4 คะแนน)*
4) ตัวสร้างแบบ **Detail Constructor** เพื่อกำหนดค่าให้กับแอตทริบิวทุกตัว *(4 คะแนน)*
5) ตัวสร้างแบบ **Copy Constructor** เพื่อกำหนดค่าให้กับแอตทริบิวทุกตัว *(4 คะแนน)*

.....

.....

.....

.....

โจทย์ข้อที่ 3 [ระดับปานกลาง] จงเขียนคลาส `ExtendedCourse` ที่สืบทอดมาจากคลาส `Course` ให้สมบูรณ์ พร้อมทั้งเรียกใช้งานที่คลาส `DemoCourse` โดยมีรายละเอียดดังต่อไปนี้ *(25 คะแนน)*

```
1  public class Course {
2      protected int id;
3      protected String title;
4      protected double credit;
5      public Course(int id, String title, double credit) {
6          this.id = id;  this.title = title;  this.credit = credit;
7      }
8      public Course(Course c) { this(c.id, c.title, c.credit); }
9      protected void setID(int id) { ... }
10     protected void setTitle(String title) { ... }
11     protected void setCredit(double credit) { ... }
12     protected String getLevel() { ... }
13     protected int getFaculty() { ... }
14     protected int getDepartment() { ... }
15     public String toString() {
16         return id + " " + title + " (" + credit + ")";
17     }
18 }
```

แอตทริบิว `id` ใช้แทนรหัสวิชา, แอตทริบิว `title` ใช้แทนชื่อวิชา, แอตทริบิว `credit` ใช้แทนจำนวนหน่วยกิต

เมท็อดเหล่านี้ละเอาไว้ในฐานที่เข้าใจแล้ว ซึ่งมีรายละเอียดอยู่ในบทที่ 6

คลาส ExtendedCourse

1) แอตทริบิวประจำอ๊อบเจ็คที่เป็น*อาเรย์*ชื่อ `teacher` เพื่อใช้แทนรายชื่อของอาจารย์ผู้สอน *(1 คะแนน)*
2) ตัวสร้างแบบ **Detail Constructor** และ **Copy Constructor** เพื่อกำหนดค่าให้กับแอตทริบิวทุกตัว *(5 คะแนน)*
3) เมท็อดประจำอ๊อบเจ็คชื่อ `getTeacher(…)` เพื่อคืนรายชื่อของอาจารย์ผู้ที่สอนทุกคน *(3 คะแนน)*
4) เมท็อดประจำอ๊อบเจ็คชื่อ `getTeacher(…)` เพื่อคืนชื่อของอาจารย์ผู้ที่สอนตามหมายเลขลำดับที่ระบุ เช่น อาจารย์ผู้สอนคนที่ 1 คนที่ 2 เป็นต้น *(3 คะแนน)*
5) เมท็อดประจำอ๊อบเจ็คชื่อ `toString(…)` ที่คร่อมทับ (Override) เมท็อด `toString(…)` ในคลาสต้นแบบเดิมเพื่อคืนข้อมูลของรายวิชาในรูปของสตริง *(5 คะแนน)*
   ตัวอย่างเช่น `2110111 Grean Java Programming (9.75): Wongyos Somboon Thitima`

คลาส DemoCourse

สร้างคลาส `DemoCourse` เพื่อเรียกใช้งานเมท็อดในคลาส `ExtendedCourse` มีรายละเอียดดังนี้ *(5 คะแนน)*

1) สร้างวิชาที่มีข้อมูลดังนี้ `2140101 Java Programming (3.0): Gift, Anne, Bird, Tarn`
2) แสดงรายชื่ออาจารย์ผู้สอนคนที่ 2 และข้อมูลทุกอย่างของรายวิชาขึ้นบนจอภาพ

……

……

……

……

โจทย์ข้อที่ 4 [ระดับปานกลาง] จงเขียนคลาส `EditNumber` ที่สืบทอดมาจากคลาส `Number` ให้สมบูรณ์ โดยมีรายละเอียดดังต่อไปนี้ *(20 คะแนน)*

```
1   public class Number {
2       protected double n;
3       public Number() { this(0.0); }
4       public Number(double n) { this.n = n; }
5       public Number(Number num) { this(num.n); }
6       ...
7       protected double pow(int k) { ... }
8       protected double div(double k) { return this.n / k; }
9       public String toString() { return this.n + ""; }
10  }
```

คลาส EditNumber

1) ตัวสร้างแบบ **Default Constructor, Detail Constructor** และ **Copy Constructor** เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัว *(6 คะแนน)*
2) เมท็อดประจำอ๊อบเจ็คชื่อ `pow(…)` ที่คร่อมทับ (Override) เมท็อด `pow(…)` ในคลาสต้นแบบเดิม เพื่อคำนวณหาค่าเลขยกกำลัง โดยจะต้องตรวจสอบว่าเลขฐานและเลขชี้กำลังเป็น 0 ทั้งคู่หรือไม่ ถ้าใช่ให้แสดงข้อความว่า “NaN” แล้วคืนค่า 0 แต่ถ้าไม่เช่นนั้นก็ให้คืนค่าคำตอบที่ประมวลผลได้ *(5 คะแนน)*

3) เมท็อดประจำอ๊อบเจ็คชื่อ `div(…)` ที่คร่อมทับ (Override) เมท็อด `div(…)` ในคลาสต้นแบบเดิมเพื่อใช้คำนวณหาค่าผลหาร โดยจะต้องตรวจสอบว่าตัวหารเป็น 0 หรือไม่ ถ้าใช่ให้แสดงข้อความว่า "Infinity" แล้วคืนค่า 0 ถ้าไม่เช่นนั้นก็ให้คืนค่าคำตอบที่ประมวลผลได้ *(5 คะแนน)*
4) เมท็อดประจำอ๊อบเจ็คชื่อ `toString(…)` ที่คร่อมทับ (Override) เมท็อด `toString(…)` ในคลาสต้นแบบเดิม เพื่อคืนค่าข้อมูลในรูปของสตริงดังตัวอย่าง `"n = 13.0"` เป็นต้น *(4 คะแนน)*

……

……

……

……

……

โจทย์ข้อที่ 5 [ระดับยาก] จงพิจารณาคลาส `Point` ที่กำหนดให้ต่อไปนี้เพื่อใช้ในการสร้างคลาส `Point3D` คลาส `Circle` และ คลาส `Line` ตามลำดับ โดยในแต่ละคลาสมีรายละเอียดดังนี้ *(50 คะแนน)*

```
1   public class Point {
2       protected int x, y;
3       public Point() { this(0, 0); }
4       public Point(int x, int y) { this.x = x;  this.y = y; }
5       public Point(Point p) { this(p.x, p.y); }
6
7       protected boolean equals(Point p) {
8           if (p.x == this.x && p.y == this.y) {
9               return true;
10          } else {
11              return false;
12          }
13      }
14
15      public String toString() {
16          return "(" + this.x + "," + this.y + ")";
17      }
18
19      protected double distanceTo(Point p) {
20          double dX = Math.abs(this.x - p.x);
21          double dY = Math.abs(this.y - p.y);
22          return Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));
23      }
24  }
```

คลาส `Point3D` ที่เป็นจุดแบบ 3 มิติ ซึ่งสืบทอดมาจากคลาส `Point` โดยมีรายละเอียดดังนี้ *(15 คะแนน)*

1) แอตทริบิวประจำอ๊อบเจ็คชื่อ `z` ชนิดจำนวนเต็มซึ่งเป็นพิกัดของมิติที่ 3 ของจุดแบบ 3 มิติ
2) ตัวสร้างแบบ Detail Constructor และ Copy Constructor เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัว
3) เมท็อดประจำอ๊อบเจ็คชื่อ `toString(…)` ที่โอเวอไรด์เมท็อด `toString(…)` ของคลาส `Point` โดยจะคืนค่าผลลัพธ์ที่เป็นข้อความที่แสดงรายละเอียดของพิกัด (x, y, z) ของคลาส `Point3D`
4) เมท็อดประจำอ๊อบเจ็คชื่อ `equals(…)` ที่โอเวอไรด์เมท็อด `equals(…)` ของคลาส `Point` โดยจะคืนค่าผลการเปรียบเทียบของจุดแบบ `Point3D` สองจุดใด ๆ เท่ากันหรือไม่

5) เมท็อดประจำอ๊อบเจ็คชื่อ `distanceTo(…)` ที่โอเวอไรด์เมท็อด `distanceTo(…)` ของคลาส `Point` โดยจะคืนค่าระยะห่างระหว่างจุดของคลาส `Point3D` สองจุดใด ๆ

คลาส `Circle` ที่เป็นวงกลมหนึ่งวง ซึ่งสืบทอดมาจากคลาส `Point` โดยมีรายละเอียดดังนี้ *(15 คะแนน)*

1) แอตทริบิวประจำอ๊อบเจ็คชื่อ `r` ชนิดจำนวนจริงซึ่งเป็นรัศมีของวงกลม
2) ตัวสร้างแบบ **Detail Constructor** เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัว
3) ตัวสร้างแบบ 2 พารามิเตอร์ซึ่งได้แก่ อ๊อบเจ็คชนิด `Point` เพื่อใช้กำหนดค่าเริ่มต้นให้กับจุดศูนย์กลางของวงกลม และจำนวนจริง เพื่อกำหนดค่าเริ่มต้นให้กับรัศมีของวงกลม
4) เมท็อดประจำอ๊อบเจ็คชื่อ `isInCircle(…)` ที่รับจุด (ชนิด `Point`) เข้ามาทางพารามิเตอร์เพื่อตรวจสอบว่าจุดดังกล่าวอยู่ในวงกลมนี้หรือไม่
5) เมท็อดประจำอ๊อบเจ็คชื่อ `isInCircle(…)` ที่รับวงกลม (ชนิด `Circle`) เข้ามาทางพารามิเตอร์เพื่อตรวจสอบว่าวงกลมดังกล่าวอยู่ในวงกลมนี้หรือไม่

คลาส `Line` ที่เป็นเส้นตรงหนึ่งเส้น ซึ่งสืบทอดมาจากคลาส `Point` โดยมีรายละเอียดดังนี้ *(15 คะแนน)*

1) แอตทริบิวประจำอ๊อบเจ็คชื่อ `a` และ `b` ชนิดจำนวนเต็มซึ่งเป็นพิกัดของจุดปลายของเส้นตรง (โดยมีอีกหนึ่งจุดอยู่ที่คลาส `Point`)
2) ตัวสร้างแบบ **Detail Constructor** เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัว
3) ตัวสร้างแบบ 2 พารามิเตอร์ที่เป็นอ๊อบเจ็คชนิด `Point` ชื่อ `p1` และ `p2` เพื่อใช้กำหนดค่าเริ่มต้นให้กับจุดปลายทั้งสองของเส้นตรง
4) เมท็อดประจำอ๊อบเจ็คชื่อ `getLength(…)` เพื่อคำนวณหาความยาวของเส้นตรง
5) เมท็อดประจำอ๊อบเจ็คชื่อ `equals(…)` ที่โอเวอไรด์เมท็อด `equals(…)` ของคลาส `Point` เพื่อเปรียบเทียบว่าเส้นตรงที่รับเข้ามาทางพารามิเตอร์ (ชนิด `Line`) กับเส้นตรงในคลาสนี้เป็นเส้นตรงเดียวกันหรือไม่

คลาส `Test` เพื่อสร้างอ๊อบเจ็คจากคลาส `Point3D` คลาส `Circle` และคลาส `Line` ตามลำดับ (ในการสร้างอ๊อบเจ็คจากคลาส `Circle` และ `Line` ห้ามใช้ **Detail Constructor**) *(5 คะแนน)*

โจทย์ข้อที่ 6 [ระดับยาก] จงเขียนโปรแกรมภาษาจาวาให้สมบูรณ์เพื่อสร้างคลาสต้นแบบและคลาสประมวลผลดังนี้คือ `SSRUStudent`, `SciSSRUStudent` และ `TestStudent` โดยมีรายละเอียดดังนี้ *(30 คะแนน)*

คลาส `Date` กำหนดให้คลาส `Date` ที่ใช้แทนวันเดือนปีใด ๆ

```
1  public class Date {
2      public int day, month, year;
3      public Date(int d, int m, int y) {
4          this.day = d;  this.month = m;  this.year = y;
5      }
6      public Date(Date date) {
7          this(date.day, date.month, date.year);
8      }
9      public String toString() {
10         return this.day + "/" + this.month + "/" + this.year;
11     }
12 }
```

```
1  public class Student {
2      protected int id;
3      protected String name;
4      protected Date date;
5      public Student() { this(0, "", new Date(0, 0, 0)); }
6      public Student(int id, String name, Date date) {
7          this.id = id;  this.name = name;  this.date = new Date(date);
8      }
9      public Student(Student std) { this(std.id, std.name, std.date); }
10     public int getID() {
11         return this.id;
12     }
13     public String getName() { return this.name; }
14     public Date getDate() { return this.date; }
15     public void show() {
16         System.out.println(this.id);
17         System.out.println(this.name);
18         System.out.println(this.date.toString());
19     }
20 }
```

คลาส `SSRUStudent` เป็นคลาสที่ใช้แทนนักศึกษาสวนสุนันทาหนึ่งคน ซึ่งสืบทอดมาจากคลาส `Student` โดยมีรายละเอียดดังนี้ *(10 คะแนน)*

1) **แอตทริบิวประจำอ๊อบเจ็คแบบ `protected` ชื่อ `grade` เพื่อเก็บผลการเรียนเฉลี่ยของนักศึกษา**
2) **ตัวสร้างแบบ Default Constructor, Detail Constructor และ Copy Constructor เพื่อกำหนดค่าเริ่ม ต้นให้กับแอตทริบิวทุกตัว (ในที่นี้จะหมายถึงแอตทริวบิวที่อยู่ทั้งในคลาสต้นแบบใหม่นั่นคือ `grade` และแอตทริบิวในคลาสต้นแบบเดิมนั่นคือ `id`, `name` และ `date`)**
3) **เมท็อดประจำอ๊อบเจ็คชื่อ `getGrade(…)` เพื่อคืนค่าผลการเรียนเฉลี่ยของนักศึกษา**
4) **เมท็อดประจำอ๊อบเจ็คชื่อ `show(…)` ที่โอเวอร์ไรด์เมท็อด `show(…)` ของคลาส `Student` เพื่อแสดงแอตทริบิวทุกตัวออกทางจอภาพทีละบรรทัด**

คลาส `SciSSRUStudent` เป็นคลาสที่ใช้แทนนักศึกษาวิทย์คอม สวนสุนันทาฯ หนึ่งคน ซึ่งสืบทอดมาจากคลาส `SSRUStudent` โดยมีรายละเอียดดังนี้ *(10 คะแนน)*

1) **แอตทริบิวประจำอ๊อบเจ็คแบบ `protected` ชื่อ `noob` เพื่อเก็บระดับความเกรียนของนักศึกษา**

2) ตัวสร้างแบบ 2 พารามิเตอร์ที่ประกอบไปด้วยอ๊อบเจ็คของคลาส `SSRUStudent` และจำนวนเต็มที่เป็นระดับความเกรียน เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัว (ทุกคลาสต้นแบบ)
3) เมท็อดประจำอ๊อบเจ็คชื่อ `getNoobLevel(…)` เพื่อคืนค่าระดับความเกรียนของนักศึกษา
4) เมท็อดประจำอ๊อบเจ็คชื่อ `show(…)` ที่โอเวอร์ไรด์เมท็อด `show(…)` ของคลาส `SSRUStudent` เพื่อแสดงแอตทริบิวทุกตัวออกทางจอภาพทีละบรรทัด ซึ่งการแสดงระดับความเกรียนให้พิจารณาดังต่อไปนี้ ถ้าเป็นค่า 0 ให้แสดงคำว่า "Common" ถ้าเป็นค่า 1 ให้แสดงคำว่า "Noob" ถ้าเป็นค่า 2 ให้แสดงคำว่า "Fa-ther Noob" ถ้าเป็นค่า 3 ให้แสดงคำว่า "God Noob" ถ้าเป็นค่า 3 ให้แสดงคำว่า "God Father Noob" ถ้าไม่เช่นนั้น แสดงคำว่า "Unknown"

คลาส `TestStudent` ซึ่งเป็นคลาสที่มีเมท็อด `main(…)` เพื่อเรียกใช้งานคลาสและประมวลผลเมท็อดของคลาส `Student`, `SSRUStudent` และ `SciSSRUStudent` ดังรายละเอียดต่อไปนี้ *(10 คะแนน)*

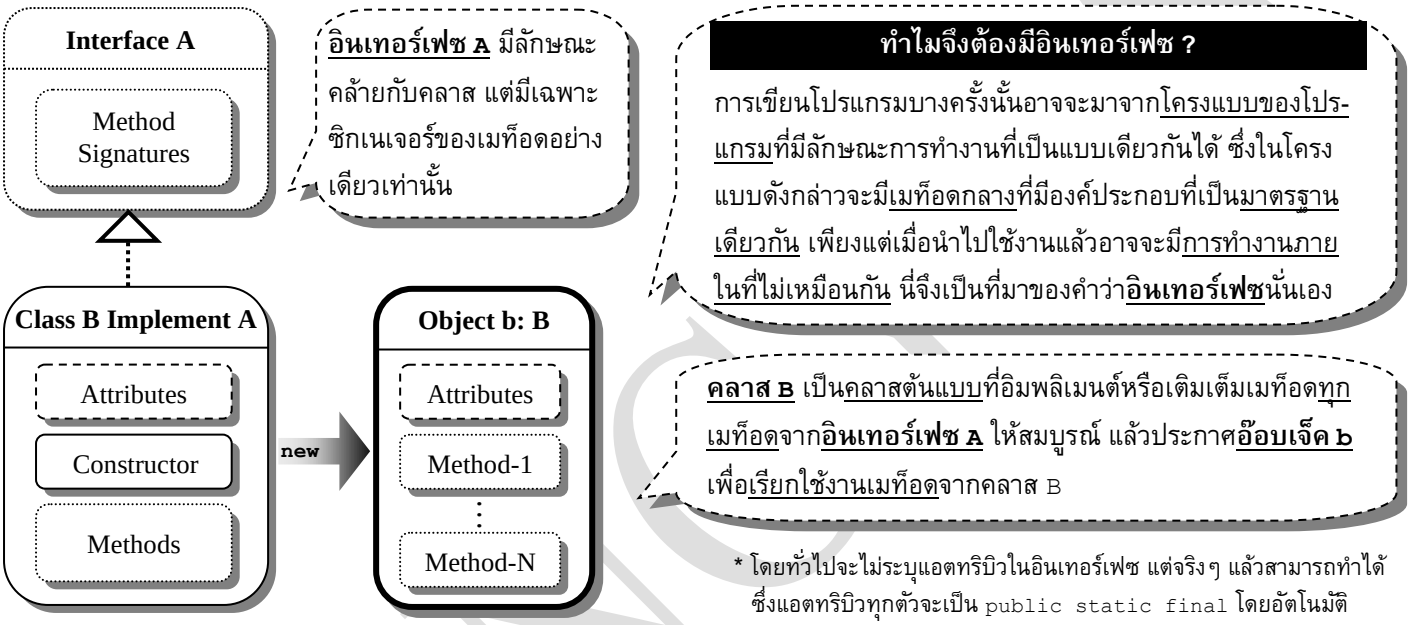
1) สร้างอ๊อบเจ็คชื่อ `std` มีชนิดเป็น `Student` เพื่อเป็นตัวแทนของนักศึกษาใด ๆ โดยมีเลขประจำตัวเป็น 553001 ชื่อ Sudkhet และวันเดือนปีเกิดคือ 18/6/1995 พร้อมทั้งเรียกใช้เมท็อด `show(…)`
2) สร้างอ๊อบเจ็คชื่อ `ctd` มีชนิดเป็น `SSRUStudent` เพื่อเป็นตัวแทนของนักศึกษาจุฬาฯ โดยมีเลขประจำตัวเป็น 553002 ชื่อ Sudyod วันเดือนปีเกิดคือ 9/3/1996 และมีผลการเรียนเฉลี่ยเป็น 4.00 พร้อมทั้งเรียกใช้เมท็อด `show(…)`
3) สร้างอ๊อบเจ็คชื่อ `etd` มีชนิดเป็น `SciSSRUStudent` เพื่อเป็นตัวแทนของนักศึกษาวิทย์คอม สวนสุนันทาฯ โดยมีเลขประจำตัวเป็น 553003 ชื่อ Sudteen วันเดือนปีเกิดคือ 12/12/2012 ผลการเรียนเฉลี่ย 4.01 และมีระดับความเกรียนเป็น 4 พร้อมทั้งเรียกใช้เมท็อด `show(…)`

2 อินเทอร์เฟส (Interfaces)

1. นิยามของอินเทอร์เฟซ (Definition of Interfaces)

อินเทอร์เฟซ คือ โครงแบบของโปรแกรมที่มีลักษณะคล้ายกับคลาส (Class) แต่มีความแตกต่างจากคลาสดังต่อไปนี้

1) อินเทอร์เฟซจะมีเพียงเมท็อด (Method) เท่านั้นไม่มีแอตทริบิว (Attribute)* และคอนสตรักเตอร์ (Constructor)
2) เมท็อดภายในอินเทอร์เฟซจะเป็นเพียงเมท็อดคัดย่อ (Abstract Method) เท่านั้น นั่นคือจะมีเพียงส่วนของหัวเมท็อดหรือซิกเนเจอร์ (Signature) อย่างเดียวโดยไม่มีตัวเมท็อดหรืออิมพลิเมนเทชั่น (Implementation)
3) เมท็อดภายในอินเทอร์เฟซจะประกอบไปด้วยประเภทข้อมูลที่ส่งกลับ (Return Type) ชื่อเมท็อด (Method Name) และพารามิเตอร์ (Parameters) โดยไม่ต้องมีม็อดดิฟายเออร์ (Modifier) แต่เสมือนว่าเมท็อดทุกตัวเป็น `public`

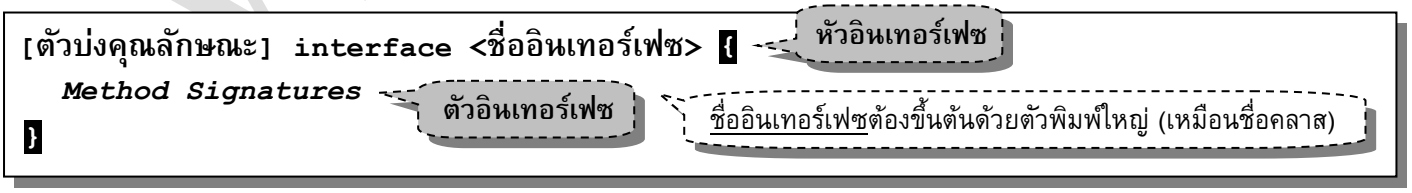


* โดยทั่วไปจะไม่ระบุแอตทริบิวในอินเทอร์เฟซ แต่จริง ๆ แล้วสามารถทำได้ ซึ่งแอตทริบิวทุกตัวจะเป็น `public static final` โดยอัตโนมัติ

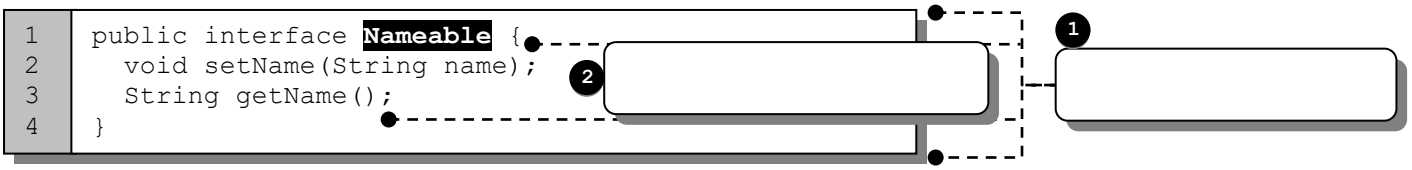
2. ความแตกต่างระหว่างอินเทอร์เฟซ (Interfaces) และการสืบทอด (Inheritance)

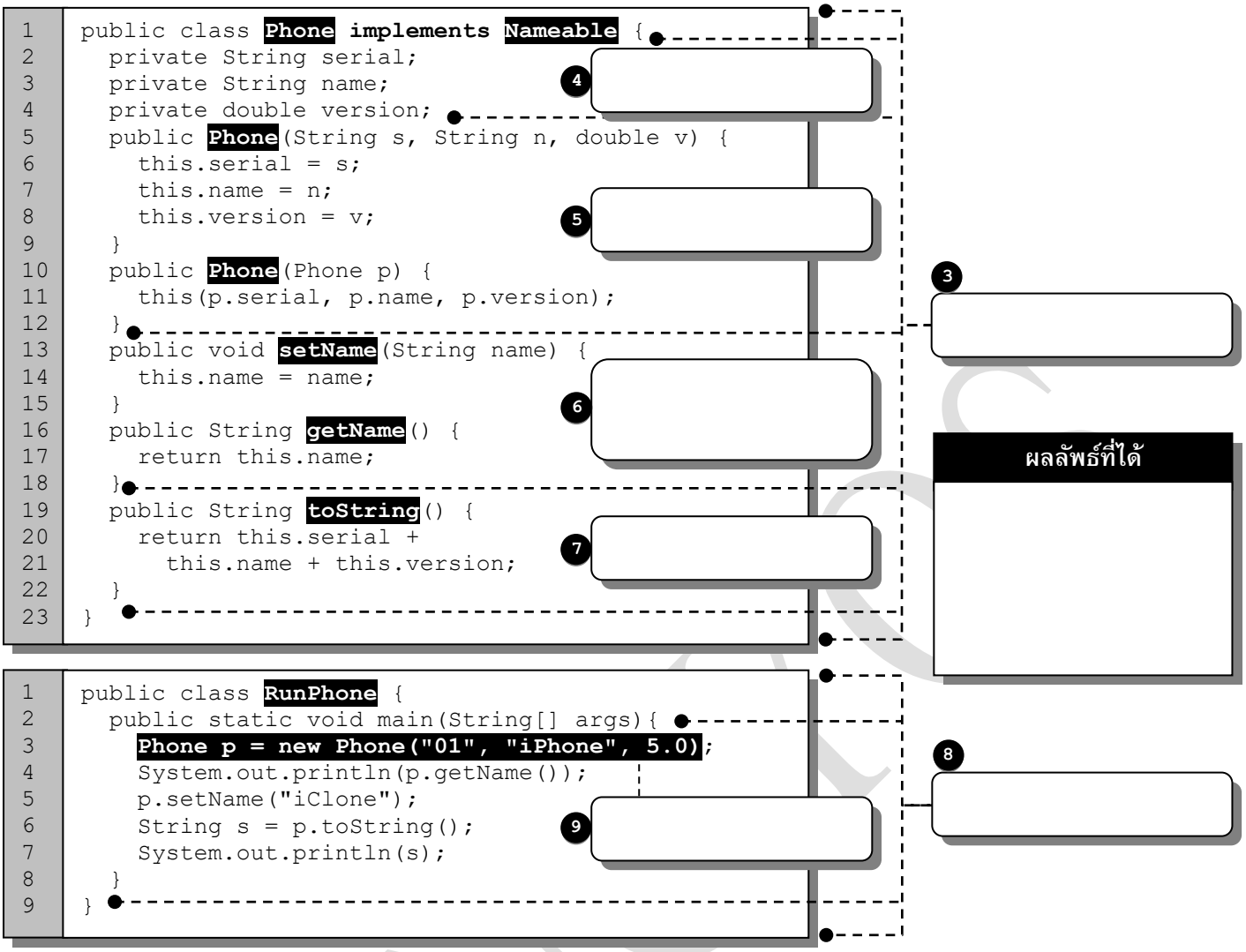
1) คลาสจะใช้คีย์เวิร์ด `implements` ในการใช้งานอินเทอร์เฟซ แต่จะใช้คีย์เวิร์ด `extends` ในการสืบทอดคลาส
2) จุดเริ่มต้นของอินเทอร์เฟซคือตัวอินเทอร์เฟซที่จะถูกอิมพลิเมนต์ให้เป็นคลาสต้นแบบใหม่ แต่จุดเริ่มต้นของการสืบทอดคือตัวคลาสต้นแบบพ่อแม่ที่จะถูกขยายหรือสืบทอดให้เป็นคลาสต้นแบบใหม่
3) คลาสต้นแบบสามารถอิมพลิเมนต์อินเทอร์เฟซได้หลายตัว แต่สามารถสืบทอดคลาสได้เพียงตัวเดียวเท่านั้น

3. การประกาศและสร้างอินเทอร์เฟซ



โจทย์ข้อที่ 7 [ระดับง่าย] จงระบุส่วนประกอบต่าง ๆ ของอินเทอร์เฟซและคลาสต่อไปนี้ให้ถูกต้อง *(15 คะแนน)*





โจทย์ข้อที่ 8 [ระดับง่าย] จงพิจารณาการประกาศและสร้างองค์ประกอบของอินเทอร์เฟซและคลาสต่อไปนี้ผิด (☒) หรือถูก (☑) ตามหลักไวยากรณ์พร้อมทั้งบอกเหตุผลกำกับ (โจทย์แต่ละข้อจะต่อเนื่องกัน) *(15 คะแนน)*

1) ☐
```
public interface Addable {
  double zero = 0.0;
  public double add(double n);
}
```

2) ☐
```
public interface Subable {
  protected double sub(double n);
}
```

3) ☐
```
public class Multiplication {
  protected double x;
  public Multiplication(double n) {
    this.x = n;
  }
  public double mul(double n);  //return this.x * n;
  public void setX(double n) { this.x = n; }
  public double getX() { return this.x; }
  public String toString() { return x + ""; }
}
```

4) ☐
```
public class Number1 extends Addable {
  private double x;
  public Number1(double n) { this.x = n; }
  public double add(double n) { return this.x + n; }
}
```

5) ☐
```
public class Number2 implement Addable {
  private double x;
  public Number2(double n) {
    this.x = n;
  }
  public double add(double n) { return this.x + n; }
}
```

6) ☐
```
public class Number3 implements Addable {
  private double x;
  public Number3(double n) {
    this.x = n;
  }
  double add(double n) { return this.x + n; }
}
```

7) ☐
```
public class Number4 implements Addable {
  private double x;
  public Number4(double n) {
    this.x = n;
  }
  public double add(double m, double n) { return m + n; }
}
```

8) ☐
```
public class Number5 implements Addable, Subable {
  private double x;
  public Number5(double n) { this.x = n; }
  public double add(double n) { return this.x + n; }
  public double sub(double n) { return this.x - n; }
}
```

9) ☐
```
public class TheNumber extends Multiplication implements Addable, Subable {
  public TheNumber(double n) { super(n); }
  public double add(double n) { return super.x + n; }
  public double sub(double n) { return super.x - n; }
  public double div(double n) { return super.x / n; }
}
```

10) ☐
```
public class Test {
  public static void operate1() {
    Addable a = new Addable(10);
    System.out.println(a.add(7));
  }
```

11) ☐
```
  public static void operate2() {
    Addable a = new TheNumber(10);
    System.out.println(a.add(7));
  }
```

12) ☐
```
  public static void operate3() {
    Addable a = new TheNumber(20);
    System.out.println(a.toString());
  }
```

13) ☐
```
  public static void operate4(Addable a) {
    double num = a.add(9);
    System.out.println(num);
  }
```

14) ☐
```
  public static Addable operate5(TheNumber t) {
    Addable a = (Addable) t;
    return a;
  }
```

15) ☐
```
  public static void main(String[] args) {
    TheNumber t = new TheNumber(10);
    operate4(t);
  }
} //End of class Test
```

โจทย์ข้อที่ 9 [ระดับง่าย] จงเขียนคลาส Student (นักเรียน) คลาส Room (ห้องเรียน) และคลาส Course (วิชาเรียน) ที่อิมพลิเมนต์อินเทอร์เฟซ Identity (เอกลักษณ์) โดยมีรายละเอียดดังต่อไปนี้ *(24 คะแนน)*

```
1  public interface Identity {
2    String getIdentity();
3    void setIdentity(String iden);
4  }
```

อินเทอร์เฟซ Identity แสดงถึงเอกลักษณ์ของวัตถุใด ๆ

คลาส Student

1) กำหนดให้มีแอตทริบิวและคอนสตรักเตอร์ดังต่อไปนี้

```
private String id;
private String name;
public Student(String i, String n) { this.id = i;  this.name = n }
```

2) เมท็อด getIdentity(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Identity *(4 คะแนน)*
3) เมท็อด setIdentity(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Identity *(4 คะแนน)*

คลาส Room

1) กำหนดให้มีแอตทริบิวและคอนสตรักเตอร์ดังต่อไปนี้

```
private String name;
private int size;
public Room(String n, int s) { this.name = n;  this.size = s; }
```

2) เมท็อด getIdentity(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Identity *(4 คะแนน)*
3) เมท็อด setIdentity(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Identity *(4 คะแนน)*

คลาส Courses

1) กำหนดให้มีแอตทริบิวและคอนสตรักเตอร์ดังต่อไปนี้

```
private int code;
private String name;
private double credit;
public Courses(int c, String n, double x) {
  this.code = c;  this.name = n;  this.credit = x;
}
```

2) เมท็อด getIdentity(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Identity *(4 คะแนน)*
3) เมท็อด setIdentity(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Identity *(4 คะแนน)*

โจทย์ข้อที่ 10 [ระดับปานกลาง] จงสร้างอินเทอร์เฟซ Vehicle (ยานพาหนะ) แล้วสร้างคลาส Motorcycle (รถจักรยานยนต์) ที่อิมพลิเมนต์อินเทอร์เฟซ Vehicle และคลาส Car (รถยนต์) ที่สืบทอดจากคลาส Motorcycle พร้อมทั้งเรียกใช้งานคลาสทั้งสองที่คลาส DemoVehicle โดยมีรายละเอียดต่าง ๆ ดังนี้ *(90 คะแนน)*

อินเทอร์เฟซ Vehicle

1) เมท็อดซิกเนเจอร์ชื่อ getWheel(…) เพื่อคืนค่าจำนวนล้อของยานพาหนะ *(2 คะแนน)*
2) เมท็อดซิกเนเจอร์ชื่อ getSeat(…) เพื่อคืนค่าจำนวนที่นั่งของยานพาหนะ *(2 คะแนน)*
3) เมท็อดซิกเนเจอร์ชื่อ getEngine(…) เพื่อคืนค่าประเภทเครื่องยนต์ของยานพาหนะ *(2 คะแนน)* กำหนดให้ประเภทเครื่องยนต์มี 3 ประเภทได้แก่ Diesel, Benzine และ Gasohol
4) เมท็อดซิกเนเจอร์ชื่อ getFuel(…) เพื่อคืนค่าปริมาณน้ำมันที่เหลือในยานพาหนะ *(2 คะแนน)*
5) เมท็อดซิกเนเจอร์ชื่อ addFuel(…) เพื่อเติมน้ำมันให้กับยานพาหนะ *(2 คะแนน)*
6) เมท็อดซิกเนเจอร์ชื่อ getSpeed(…) เพื่อคืนค่าความเร็วปัจจุบัน ณ ขณะที่ขับยานพาหนะ *(2 คะแนน)*
7) เมท็อดซิกเนเจอร์ชื่อ upSpeed(…) เพื่อเพิ่มความเร็วของยานพาหนะตาม Scale ที่ระบุ *(2 คะแนน)*
8) เมท็อดซิกเนเจอร์ชื่อ downSpeed(…) เพื่อลดความเร็วของยานพาหนะตาม Scale ที่ระบุ *(2 คะแนน)*

คลาส Motorcycle ที่พัฒนามาจากอินเทอร์เฟซ Vehicle

1) แอตทริบิวแบบ protected ชื่อ wheel เพื่อเก็บจำนวนล้อของรถจักรยานยนต์ *(1 คะแนน)*
2) แอตทริบิวแบบ protected ชื่อ seat เพื่อเก็บจำนวนที่นั่งของรถจักรยานยนต์ *(1 คะแนน)*
3) แอตทริบิวแบบ protected ชื่อ engine เพื่อเก็บประเภทเครื่องยนต์ของรถจักรยานยนต์ *(1 คะแนน)*
4) แอตทริบิวแบบ protected ชื่อ fuel เพื่อเก็บปริมาณน้ำมันของรถจักรยานยนต์ *(1 คะแนน)*
5) แอตทริบิวแบบ protected ชื่อ speed เพื่อเก็บความเร็วของรถจักรยานยนต์ *(1 คะแนน)*
6) คอนสตรักเตอร์ที่เป็นแบบ Detail Constructor และ Copy Constructor เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัวของคลาส *(5 คะแนน)*
7) เมท็อด getWheel(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle *(2 คะแนน)*
8) เมท็อด getSeat(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle *(2 คะแนน)*
9) เมท็อด getEngine(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle *(2 คะแนน)*
10) เมท็อด getFuel(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle *(2 คะแนน)*
11) เมท็อด addFuel(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle โดยปริมาณน้ำมันที่ใส่ในถังได้สูงสุดคือ 50.0 ลิตร *(4 คะแนน)*
12) เมท็อด getSpeed(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle *(2 คะแนน)*
13) เมท็อด upSpeed(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle โดยจะเพิ่มความเร็วรถขึ้นครั้งละ x กิโลเมตรต่อชั่วโมง โดยที่ x เป็นช่วง Scale ที่ระบุทางพารามิเตอร์ ซึ่งจะเพิ่มความเร็วสูงสุดได้ไม่เกิน 140.0 กิโลเมตรต่อชั่วโมง *(4 คะแนน)*
14) เมท็อด downSpeed(…) ที่ได้จากการอิมพลิเมนต์เมท็อดของอินเทอร์เฟซ Vehicle โดยจะลดความเร็วรถลงครั้งละ x กิโลเมตรต่อชั่วโมง ซึ่งความเร็วต่ำสุดคือ 0.0 กิโลเมตรต่อชั่วโมง *(4 คะแนน)*

15) เมท็อดแบบ `public` ชื่อ `break(…)` เพื่อลดความเร็วของรถลงครั้งละ `x` กิโลเมตรต่อชั่วโมงไปเรื่อย ๆ จนกว่ารถจะจอดสนิท (ให้เรียกใช้เมท็อด `downSpeed(…)`) *(4 คะแนน)*

คลาส `Car` ที่สืบทอดมาจากคลาส `Motorcycle`

1) แอตทริบิวแบบ `protected` ชื่อ `distance` เพื่อเก็บระยะทางในการเคลื่อนที่ของรถยนต์ *(1 คะแนน)*
2) คอนสตรักเตอร์ที่เป็นแบบ **Detail Constructor** และ **Copy Constructor** เพื่อกำหนดค่าเริ่มต้นให้กับแอตทริบิวทุกตัวของคลาส *(5 คะแนน)*
3) เมท็อดแบบ `public` ชื่อ `getDistance(…)` เพื่อคืนค่าระยะทางทั้งหมดของรถยนต์ *(2 คะแนน)*
4) เมท็อดชื่อ `upSpeed(…)` ที่โอเวอร์ไรด์เมท็อด `upSpeed(…)` จากคลาส `Motorcycle` ซึ่งจะเพิ่มความ- เร็วรถขึ้นครั้งละ `x` กิโลเมตรต่อชั่วโมง และให้ความเร็วสูงสุดไม่เกิน **260.0** กิโลเมตรต่อชั่วโมง โดยขณะที่เพิ่มความเร็วจะต้องคำนวณระยะทางที่เกิดขึ้น ณ ขณะนั้นด้วย ซึ่งกำหนดให้การเพิ่มความเร็วในแต่ละครั้ง (1 วินาที) จะทำให้มีระยะทางเพิ่มขึ้นตามสูตร $distance_t = (speed + 0.5x) / 3600$ *(4 คะแนน)*
5) เมท็อดชื่อ `downSpeed(…)` ที่โอเวอร์ไรด์เมท็อด `downSpeed(…)` จากคลาส `Motorcycle` ซึ่งจะลดความเร็วรถลงครั้งละ `x` กิโลเมตรต่อชั่วโมง โดยขณะที่ลดความเร็วจะต้องคำนวณระยะทางที่เกิดขึ้น ณ ขณะนั้นด้วย ซึ่งกำหนดให้การลดความเร็วในแต่ละครั้ง (1 วินาที) จะทำให้มีระยะทางเพิ่มขึ้นตามสูตรที่กำหนดให้ $distance_t = (speed - 0.5x) / 3600$ *(4 คะแนน)*
6) เมท็อดชื่อ `break(…)` ที่โอเวอร์ไรด์เมท็อด `break(…)` จากคลาส `Motorcycle` ซึ่งจะลดความเร็วของรถลงครั้งละ `x` กิโลเมตรต่อชั่วโมงไปจนกว่ารถจะจอดสนิท โดยขณะที่ลดความเร็วจะต้องคำนวณระยะ ทางที่เกิดขึ้น ณ ขณะนั้นด้วย (ให้เรียกใช้เมท็อด `downSpeed(…)` ของคลาส `Car`) *(4 คะแนน)*

คลาส `DemoVehicle` เป็นคลาสประมวลผลที่มีเมท็อด `main(…)` ซึ่งมีรายละเอียดดังนี้ *(10 คะแนน)*

1) สร้างรถจักรยานยนต์แบบ **2** ที่นั่ง เครื่องยนต์แบบ **Diesel** โดยมีน้ำมันอยู่ในถัง **10** ลิตร *(2 คะแนน)*
2) สร้างรถยนต์แบบ **6** ที่นั่ง เครื่องยนต์แบบ **Benzine** โดยมีน้ำมันอยู่ในถัง **30** ลิตร *(2 คะแนน)*
3) แสดงรายละเอียดของความเร็วในทุก ๆ ขั้นตอนของการขับรถจักรยานยนต์จากหยุดนิ่งโดยเพิ่มความเร็ววินาทีละ **2** กิโลเมตรต่อชั่วโมง นาน **50** วินาที แล้วจึงค่อยเหยียบเบรกทีละ **1** *(4 คะแนน)*
4) แสดงระยะทางทั้งหมดของการขับรถยนต์จากหยุดนิ่งโดยเพิ่มความเร็ววินาทีละ **3** กิโลเมตรต่อชั่วโมง นาน **43** วินาที หลังจากนั้นค่อย ๆ ลดความเร็วลงวินาทีละ **1.5** กิโลเมตรต่อชั่วโมง นาน **10** วินาที แล้วจึงค่อยเหยียบเบรกทีละ **1** *(4 คะแนน)*

……

……

……

……

……

……

……

โจทย์ข้อที่ 11 [ระดับปานกลาง] จงสร้างอินเทอร์เฟซ `StudentModel` และ `EmployeeModel` ที่เป็นโมเดลของนักเรียนและพนักงานตามลำดับ และสร้างคลาส `Student` ที่อิมพลิเมนต์อินเทอร์เฟซ `StudentModel` และคลาส `Employee` ที่อิมพลิเมนต์อินเทอร์เฟซ `EmployeeModel` พร้อมทั้งสร้างคลาส `StudentEmployee` ที่อิมพลิเมนต์อินเทอร์เฟซ `StudentModel` และ `EmployeeModel` และประมวลผลลัพธ์ต่าง ๆ ที่คลาส `Run` โดยมีรายละเอียดดังต่อไปนี้ *(60 คะแนน)*

กำหนดให้อินเทอร์เฟซ `StudentModel` ที่ประกอบไปด้วยเมท็อดซิกเนเจอร์ `setGPA(…)` และ `getGPA(…)` ซึ่งใช้ในการกำหนดค่าและคืนค่าผลการเรียนเฉลี่ยของนักเรียน *(5 คะแนน)*

```
1  public interface StudentModel {
2    void setGPA(double gpa);
3    double getGPA();
4  }
```

อินเทอร์เฟซ `EmployeeModel` ที่ประกอบไปด้วยเมท็อดซิกเนเจอร์ `setSalary(…)` และ `getSalary(…)` ซึ่งใช้ในการกำหนดค่าและคืนค่าเงินเดือนของพนักงาน *(5 คะแนน)*

คลาส `Student` ที่อิมพลิเมนต์อินเทอร์เฟซ `StudentModel` โดยประกอบไปด้วยแอตทริบิว `id`, `name` และ `gpa` เพื่อเก็บรหัสนักเรียน ชื่อนักเรียน และผลการเรียนเฉลี่ย ตามลำดับ มี Detail Constructor และ Copy Constructor และมีเมท็อดดังนี้ `getID(…)`, `getName(…)`, `getGPA(…)` และ `setGPA(…)` *(15 คะแนน)*

คลาส `Employee` ที่อิมพลิเมนต์อินเทอร์เฟซ `EmployeeModel` โดยประกอบไปด้วยแอตทริบิว `id`, `name` และ `salary` เพื่อเก็บรหัสพนักงาน ชื่อพนักงาน และเงินเดือน ตามลำดับ มี Detail Constructor และ Copy Constructor และมีเมท็อดดังนี้ `getID(…)`, `getName(…)`, `getSalary(…)` และ `setSalary(…)` *(15 คะแนน)*

คลาส `StudentEmployee` ที่อิมพลิเมนต์อินเทอร์เฟซ `StudentModel` และ `EmployeeModel` โดยประกอบไปด้วยแอตทริบิว `student` ที่เป็นชนิด `Student` และแอตทริบิว `employee` ที่เป็นชนิด `Employee` เพื่อเป็นข้อมูลของคนที่เป็นทั้งนักเรียนและพนักงานในเวลาเดียวกัน โดยมี Detail Constructor และ Copy Constructor และมีเมท็อด `setGPA(…)` และ `getGPA(…)` เพื่อกำหนดและคืนค่าผลการเรียนเฉลี่ยของแอตทริบิว `student` ตามลำดับและเมท็อด `setSalary(…)` และ `getSalary(…)` เพื่อกำหนดและคืนค่าเงินเดือนของแอตทริบิว `employee` ตามลำดับ *(15 คะแนน)*

คลาส `Run` ที่มีเมท็อด `main(…)` ที่มีการประมวลผลดังต่อไปนี้ *(10 คะแนน)*

1) **สร้างนักเรียนหนึ่งคนโดยมีข้อมูลเป็น 59001, Best, 3.51 ตามลำดับ และนักเรียนคนดังกล่าวก็ยังทำงานเป็นพนักงานในบริษัทหนึ่งซึ่งมีข้อมูลเป็น EM001, Best, 29500.0 ตามลำดับ**
2) **กำหนดให้นักเรียนคนดังกล่าวมีผลการเรียนเฉลี่ยเพิ่มขึ้นเป็น 3.63 พร้อมแสดงข้อมูลที่ปรับปรุงดังกล่าวขึ้นบนจอภาพ**
3) **กำหนดให้นักเรียนคนดังกล่าวและเป็นพนักงานของบริษัทได้ขึ้นเงินเดือนเป็น 31000.0 พร้อมแสดงข้อมูลที่ปรับปรุงดังกล่าวขึ้นบนจอภาพ**