

ใบงานการทดลองที่ 11

เรื่อง การใช้งาน Abstract และ Interface

1. จุดประสงค์ทั่วไป

- 1.1. รู้และเข้าใจการกำหนดวัตถุ การใช้วัตถุ การซ่อนวัตถุ และการสืบทอดประเภทของวัตถุ
- 1.2. รู้และเข้าใจโครงสร้างของโปรแกรมเชิงวัตถุ

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

- 3.1. Abstract Class คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ
คลาสที่ซ่อนรายละเอียดไว้เพื่อแสดงผล บางส่วน

```
// Abstract class
abstract class Animal {
    // Abstract method (does not have a body)
    public abstract void animalSound();
    // Regular method
    public void sleep() {
        System.out.println("Zzz");
    }
}
```

- 3.2. Interfaces คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ
คือ abstract class ที่สมบูรณ์แบบ เมื่อ method class ที่มีใน interface ไม่มีรายละเอียด

```
// interface
interface Animal {
    public void animalSound(); // interface method (does not have a body)
    public void run(); // interface method (does not have a body)
}
```

- 3.3. คำสั่ง extends และ implements มีการใช้งานที่ต่างกันอย่างไร?

Extends ใช้สำหรับสืบทอด class

Implement ใช้สำหรับสืบทอด interface ไปใช้

- 3.4. ภายใน Abstract Class มี Constructor หรือไม่? เพราะเหตุใด?

ไม่มี เพราะเป็นการสร้างที่อยู่ตัวแปรเท่านั้น ส่วนกำหนดค่าจะไปทำในส่วนของ class ลูกที่สืบทอด

- 3.5. ภายใน Interface มี Constructor หรือไม่? เพราะเหตุใด?

ไม่มี เพราะ คล้ายๆกับ abstract class แต่จะสมบูรณ์มากกว่าเลยจะไม่มีกำหนดค่าภายในเลย

4. ลำดับขั้นตอนการปฏิบัติการ

- 4.1. ให้ผู้เรียนสร้าง Abstract Class ของรถถัง(ClassicTank) โดยจะต้องมีรายละเอียดดังต่อไปนี้

- 4.1.1. Properties : HP เพื่อกำหนดค่าพลังให้กับรถถัง
- 4.1.2. Properties : Str เพื่อกำหนดค่าความแรงในการยิงของรถถัง
- 4.1.3. Properties : Vit เพื่อกำหนดค่าพลังป้องกันของรถถัง
- 4.1.4. Properties : BaseDamage เพื่อกำหนดค่าพลังการโจมตีพื้นฐาน
- 4.1.5. Method : SetHP() ; เพื่อทำการกำหนดค่าพลังเริ่มต้น
- 4.1.6. Method : GetHP() ; เพื่อตรวจสอบค่าพลัง ณ เวลาปัจจุบัน

- 4.1.7. Method : Attack(Tank Enemy) ; เพื่อทำการยิงปืนใหญ่โจมตีศัตรู โดยการโจมตี จะเป็นการลดค่าพลังของรถถังฝั่งตรงกันข้าม (Enemy คือรถถังของศัตรู, Points คือค่าพลังโจมตีของเรา)
- 4.2. ให้ผู้เรียนสร้างคลาส NormalTank เพื่อสืบทอด ClassicTank เพื่อเขียนรายละเอียดของ Method ทั้งหมดอันได้แก่ SetHP() , GetHP() , Attack(Tank Enemy)
- 4.3. ในคลาสหลัก ให้สร้าง Instance จาก NormalTank อยู่จำนวน 2 คัน เพื่อทำการต่อสู้กัน โดยควรต้องมีบทบาทดังนี้
- 4.3.1. สร้างรถถัง A และ B ให้มีค่าพลังเบื้องต้นดังต่อไปนี้

ค่าสถานะ	รถถัง A	รถถัง B
HP	200	250
Str	12	8
Vit	9	10
BaseDamage	11	10

- 4.3.2. รถถังทั้ง A และ B ผลัดกันโจมตีซึ่งกันและกัน เพื่อมุ่งหวังให้ค่าพลังของฝั่งตรงกันข้ามลดลงจนค่า HP = 0
- 4.3.3. รายละเอียดของพลังการโจมตีสามารถคำนวณได้ตามสมการดังต่อไปนี้

$$\text{DamagePoint} = \text{MyTank_BaseDamage} * \text{Floor}(\text{MyTank_Str} / \text{Enemy_Vit}) * \text{Random}(0.7, 0.9)$$
- 4.3.4. แสดงผลการทำงานผ่าน Console เพื่อให้เห็นรายละเอียดค่าพลังปัจจุบันของรถถังแต่ละคัน พลังการโจมตี ณ ขณะนั้น จนกว่าจะมีรถถังคันใดคันหนึ่งมีค่า HP = 0

โค้ดโปรแกรมภายใน Abstract Class

```

2  abstract class ClassicTank{
3      int  Str, Vit, BaseDamage;
4      double HP , point_A , point_B  ;
5      public abstract void setHP();
6      public abstract void getHP();
7      public abstract void attank();
8  }
9

```

โค้ดโปรแกรมภายใน NormalTank

```
class normalTankA extends ClassicTank{
    public void setHP() {
        HP = 200;
        Str = 12;
        Vit = 9;
        BaseDamage = 11;
    }
    public void getHP() {
        System.out.println("|-- Tank A |--");
        System.out.println(" HP : " + HP);
        System.out.println(" Str : " + Str);
        System.out.println(" Vit : " + Vit);
        System.out.println(" BaseDamage : "+ BaseDamage);
    }
    public void attank() {
        double min = 0.7 ;
        double max = 0.9 ;
        double number = (double)(Math.random()*(max-min+1)+min);
        double DamagePoint = BaseDamage *( 1.3 ) * number;
        System.out.println(" DamagePoint_tankA : "+DamagePoint);
        point_A = DamagePoint;
    }
}
```

โค้ดโปรแกรมภายในฟังก์ชันการทำงานหลัก

```

61     normalTankA tankA = new normalTankA();
62     normalTankB tankB = new normalTankB();
63
64     tankA.setHP();
65     tankA.getHP();
66     System.out.println("-----");
67     tankB.setHP();
68     tankB.getHP();
69     System.out.println("----- ROUND 1 -----");
70     tankA.attank();
71     tankB.attank();
72     tankA.getHP();
73     tankB.getHP();
74     System.out.println("----- ROUND 2 -----");
75     tankA.attank();
76     tankB.attank();
77     tankA.getHP();
78     tankB.getHP();

```

ผลลัพธ์การทำงานของโปรแกรม

```
|-- Tank A --|
HP : 200.0
Str : 12
Vit : 9
BaseDamage : 11
-----
|-- Tank B --|
HP : 250.0
Str : 8
Vit : 10
BaseDamage : 10
----- ROUND 1 -----
DamagePoint_tankA : 20.58983828454404
DamagePoint_tankB : 7.28513765335083
|-- Tank A --|
HP : 200.0
Str : 12
Vit : 9
BaseDamage : 11
|-- Tank B --|
HP : 250.0
Str : 8
Vit : 10
BaseDamage : 10
```

4.4. เปลี่ยน Abstract Class ให้กลายเป็น Interfaces และเปรียบเทียบผลลัพธ์การทำงานของโปรแกรม

หลังจากเปลี่ยน Abstract Class เป็น Interface แล้ว เกิดอะไรขึ้นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบให้ชัดเจน

```

1 package lap11_tank;
2 interface ClassicTank{
3     public abstract void getHP();
4     public abstract void attank();
5 }
6
7 public class main {
8     public static void main(String[] args) {
9         class normalTankA implements ClassicTank{
10             int HP = 200;
11             int Str = 12;
12             int Vit = 9;
13             int BaseDamage = 11;
14
15             public void getHP() {
16                 System.out.println("|-- Tank A |--");
17                 System.out.println(" HP : " + HP);
18                 System.out.println(" Str : " + Str);
19                 System.out.println(" Vit : " + Vit);
20                 System.out.println(" BaseDamage : "+ BaseDamage);
21             }
22             public void attank() {
23                 double min = 0.7 ;
24                 double max = 0.9 ;
25                 double number = (double)(Math.random()*(max-min+1)+min);
26                 double DamagePoint = BaseDamage * ( 1.3 ) * number;
27                 System.out.println(" DamagePoint_tankA : "+DamagePoint);
28             }
29         }
30     }

```

```

31     class normalTankB implements ClassicTank{
32         int HP = 250;
33         int Str = 8;
34         int Vit = 10;
35         int BaseDamage = 10;
36     public void getHP() {
37         System.out.println("|-- Tank B |--");
38         System.out.println(" HP : "+HP);
39         System.out.println(" Str : "+Str);
40         System.out.println(" Vit : "+Vit);
41         System.out.println(" BaseDamage : "+ BaseDamage);
42     }
43     public void attank() {
44         float min = (float) 0.7 ;
45         float max = (float) 0.9 ;
46         float number = (float)(Math.random()*(max-min+0.1)+min);
47         double DamagePoint = BaseDamage *( 0.8 ) * number;
48         System.out.println(" DamagePoint_tankB : "+ DamagePoint);
49     }
50 }
51 }
52 }
53
54     normalTankA tankA = new normalTankA();
55     normalTankB tankB = new normalTankB();
56 }
57
58 }
59
60     normalTankA tankA = new normalTankA();
61     normalTankB tankB = new normalTankB();
62
63     tankA.getHP();
64     System.out.println("-----");
65
66     tankB.getHP();
67     System.out.println("----- ROUND 1 -----");
68     tankA.attank();
69     tankB.attank();
70     tankA.getHP();
71     tankB.getHP();
72     System.out.println("----- ROUND 2 -----");
73     tankA.attank();
74     tankB.attank();
75     tankA.getHP();
76     tankB.getHP();
77 }

```



5. สรุปผลการปฏิบัติการ

จากการทดลองพบว่าเมื่อเราเปลี่ยนมาเป็นแบบใช้ interface จะยุ่งยากและสับสน

6. คำถามท้ายการทดลอง

6.1. เมื่อใดจึงควรเลือกใช้งาน Abstract Class

เมื่อเวลาที่เราต้องการจะใช้ตัวแปรนั้นหลายๆรอบแต่ค่าจะไม่เหมือนกัน

6.2. เมื่อใดจึงควรเลือกใช้งาน Interface

เมื่อเวลาที่เราต้องการจะใช้ตัวแปรนั้นหลายๆรอบแต่ค่าจะไม่เหมือนกัน และ เราต้องมากำหนดค่าอีก
รอบหนึ่ง