ใบงานการทดลองที่ 11

เรื่อง การใช้งาน Abstract และ Interface

1. จุดประสงค์ทั่วไป

1.1. รู้และเข้าใจการกำหนดวัตถุ การใช้วัตถุ การซ่อนวัตถุ และการสืบทอดประเภทของวัตถุ

1.2. รู้และเข้าใจโครงสร้างของโปรแกรมเชิงวัตถุ

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

3.1. Abstract Class คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Abstract Class คือคลาสที่ถูกออกแบบมาให้เป็นแม่แบบในการสืบทอดคุณสมบัติให้กับคลาสอื่น โดย Abstract Class ไม่สามารถถูกสร้างเป็น Object ได้แต่สามารถสร้างเป็นคลาสลูก ๆ และใช้เป็นต้นแบบในการสร้าง Object ได้

3.2. Interfaces คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

Interfaces เป็นตัวกลางในการสื่อสารระหว่างคลาสที่ต่างกัน โดย Interface จะกำหนดชุดของ methods ที่ต้องมีในคลาสที่ใช้งาน Interface นั้น ๆ ซึ่งคลาสที่ใช้งาน Interface จะต้องประกาศว่าจะ implementหรือใช้งาน Interface นี้เพื่อให้คลาสดังกล่าวต้องทการสร้าง method ที่ถูกกำหนดไว้ไน Interface และเติม implementation ให้เรียบร้อยตามที่ต้องการ

3.3. คำสั่ง extends และ implements มีการใช้งานที่แตกต่างกันอย่างไร?

คำสั่ง extends และ implements เป็นคำสั่งที่ใช้ในการสืบทอดคุณสมบัติ (inheritance) ของคลาสหรืออินเทอร์เฟซในภาษา Java โดยทั้งสองคำสั่งนี้มีการใช้งานที่แตกต่าง

3.4. ภายใน Abstract Class มี Constructor หรือไม่? เพราะเหตุใด?

ใน Abstract Class สามารถมี Constructor ได้เหมือนกับ Class ทั่วไป แต่จะต้องระบุเป็น protected เพื่อให้ Constructor นั้นสามารถเรียกใช้งานได้เฉพาะจาก Subclass ของ Abstract Class เท่านั้น โดยที่ไม่สามารถเรียกใช้งาน Constructor ของ Abstract Class โดยตรงได้ นอกจากนั้นก็สามารถสร้าง Constructor ที่ไม่มีพารามิเตอร์หรือพารามิเตอร์มีค่าเริ่มต้นเองได้เช่นกัน

3.5. ภายใน Interface มี Constructor หรือไม่? เพราะเหตุใด?

ภายใน Interface ไม่มี Constructor เพราะว่า Interface เป็ นแค่แบบฟอร์มที่บอกว่า Class จะต้องมี methodอะไรบ้าง แต่ไม่ได้ระบุว่าจะให้สร้าง Constructor หรือโครงสร้างอื่นใด ๆ ดังนั้นไม่จำเป็ นต้องมี Constructor ในInterface อย่างเดียวกันกับ Abstract Class โดยทั่วไป Constructor จะถูกนิยามใน Class ที่สืบทอด Abstract Class หรือ implement Interface นั้นเอง

4. ลำดับขั้นการปฏิบัติการ

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 เพื่อการกำหนดค่าพลังการโจมตีพื้นฐาน

4.1.5. Method : SetHP( ) ; เพื่อทำการกำหนดค่าพลังเริ่มต้น

4.1.6. Method : GetHP( ) ; เพื่อตรวจสอบค่าพลัง ณ เวลาปัจจุบัน

4.1.7. Method : Attack(Tank Enemy) ; เพื่อทำการยิงปืนใหญ่โจมตศัตรู โดยการโจมตีจะเป็นการลดค่าพลังของรถถังฝั่ง ตรงกันข้าม (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 DamagePoint = MyTank_BaseDamage * Floor(MyTank_Str / Enermy_Vit) * Random(0.7, 0.9)

4.3.4. แสดงผลการทำงานผ่าน Console เพื่อให้เห็นรายละเอียดค่าพลังปัจจุบันของรถถังแต่ละคัน พลังการโจมต่อ ณ ขณะนั้น จนกว่าจะมีรถถังคันใดคันหนึ่งมีค่า HP = 0

โค้ดโปรแกรมภายใน Abstract Class

```
1  package lab11;
2
3  abstract class ClassicTank{
4      int  Str, Vit, BaseDamage;
5      double HP , point_A , point_B  ;
6      public abstract void setHP();
7      public abstract void getHP();
8      public abstract void attank();
```

โค้ดโปรแกรมภายใน NormalTank

```
public class test {
    public static void main(String[] atgs) {
        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;

            }
        }
        class normalTankB extends ClassicTank{
            public void setHP() {
                HP = 250;
                Str = 8;
                Vit = 10;
                BaseDamage = 10;
            }
            public void getHP() {
                System.out.println("|-- Tank B --|");
                System.out.println(" HP : "+HP);
                System.out.println(" Str : "+Str);
                System.out.println(" Vit : "+Vit);
                System.out.println(" BaseDamage : "+ BaseDamage);
            }
            public void attank() {
                float min = (float) 0.7 ;
                float max = (float) 0.9 ;
                float number = (float)(Math.random()*(max-min+0.1)+min);
                double DamagePoint = BaseDamage *( 0.8 ) * number;
                System.out.println(" DamagePoint_tankB : "+ DamagePoint);
                point_B = DamagePoint;

            }
        }
```

โค้ดโปรแกรมภายในฟังก์ชันการทำงานหลัก

```
    normalTankA tankA = new normalTankA();
    normalTankB tankB = new normalTankB();

    tankA.setHP();
    tankA.getHP();
    System.out.println("-----------------------------------------------------------------");
    tankB.setHP();
    tankB.getHP();
    System.out.println("-------------- ROUND 1 ---------------");
    tankA.attank();
    tankB.attank();
    tankA.getHP();
    tankB.getHP();
    System.out.println("-------------- ROUND 2 ---------------");
    tankA.attank();
    tankB.attank();
    tankA.getHP();
    tankB.getHP();
```

ผลลัพธ์การทำงานของโปรแกรม

```
<terminated> test (1) [Java Application] C:\Users\thany\.p2\pool\plugins\org.eclipse.ju
|-- 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 : 16.957952889807846
 DamagePoint_tankB : 7.9145917892456055
|-- Tank A --|
 HP : 200.0
 Str : 12
 Vit : 9
 BaseDamage : 11
|-- Tank B --|
 HP : 250.0
 Str : 8
 Vit : 10
 BaseDamage : 10
-------------- ROUND 2 ---------------
 DamagePoint_tankA : 19.38857752370959
 DamagePoint_tankB : 5.75675106048584
|-- 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 แลว้ เกิดอะไรขึ้น อย่ ง? อธิบายพร้อมยกตัวอย่างประกอบให้ชัดเจน
*ตัวฟังก์ชั่นใช้งานคล้ายกัน แต่จะเปลี่ยนการกำหนดค่าตัวแปล จาก Class หลักไปเป็น Class ลูกแทน *ตัวผลลัพธ์ ของโปรแกรมเหมือนเดิม แต่อาจเปลี่ยนตัว การทำงานบางอย่าง เช่น การทำ DMG หรือ HP ของ interface อาจหากกัน 0 –9 DMG แต่ผลลัพธ์ของมันคือ WIN เหมือนเดิม

5. สรุปผลการปฏิบัติการ

การใช้Abstract Class กับ Interface มีการใช้งานที่คล้ายๆกัน จะมีส่วนที่ต่างกันตรงที่Properties โดยใน Interface จะไม่สามารถประกาศ Properties ได้แต่ใน Abstract Class ทำได้จากการทดลองที่ให้ทำการสร้างรถถัง2 คันมาสลับกันยิงแบบใช้Abstract กับ Interface หากแก้ให้ตรงตามเงื่อนไขแล้วพบว่า ผลลัพธ์ของทั้ง 2 แบบเหมือนกัน และ ผลลัพธ์ของ DamagePoint ที่คำนวณได้จากสูตร Nt2.BaseDamage * Math.floorDiv(Nt2.Str ,Nt1.Vit) * random(0.7, 0.9) พบว่าได้0 ตลอด เพราะ Math.floorDiv(Nt2.Str , Nt1.Vit) หากแทนค่าจะพบว่า Math.floorDiv(8 , 9) จะได้0 แล้วคูณในสมการก็จะได้ 0 (10 * 0 * random(0.7, 0.9))

6. คำถามท้ายการทดลอง

6.1. เมื่อใดจึงควรเลือกใช้งาน Abstract Class

เมื่อต้องเขียนโปรแกรมที่มีProperties ซ้ำกันเยอะๆ หรือมีPropertiesที่เหมือนกันเยอะ เช่น HP STR DEF AGI เป็นต้น

6.2. เมื่อใดจึงควรเลือกใช้งาน Interface

เมื่อต้องเขียนโปรแกรมที่มีProperties ไม่ซ้ำกัน หรือ มีProperties เฉพาะเยอะ