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

3.1.1. คือ Class ที่เอาไว้ซ่อนรายละเอียด จุดเด่น คือ จะมี Abstract Method
ซึ่ง Abstract Method จะไม่มีรายละเอียดของ Method อยู่ข้างใน
ถ้าอยากจะใช้งานต้องสืบทอดไปอีกทีหนึ่ง

```
// 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");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

```
The pig says: wee wee
Zzz
```

3.2. Interfaces คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

3.2.1. คือ Abstract Class ที่สมบูรณ์แบบ
สมบูรณ์แบบในที่นี้ คือ ใน Method ของ Interfaces จะไม่มีรายละเอียดอยู่เลย
ถ้าอยากจะใช้งานต้อง implements ไป ถึงจะใช้งานได้
Interfaces ไม่สามารถที่จะสร้าง instance ตรงๆได้ จะต้องสืบทอดไป
แล้วไปสร้าง instance ใน class ลูก อีกที่หนึ่งถึงจะทำได้

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

class Pig implements Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    System.out.println("Zzz");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();
    myPig.animalSound();
    myPig.sleep();
  }
}
```

```
The pig says: wee wee
Zzz
```

3.3. คำสั่ง extends และ implements มีการใช้งานที่แตกต่างกันอย่างไร?

3.3.1. Extends ใช้กับ Class และ Abstract Class

Implements ใช้กับ Interfaces

3.4. ภายใน Abstract Class มี Constructor หรือไม่? เพราะเหตุใด?

3.4.1. มีได้ เพราะ มีการประกาศ Properties ใน class

3.5. ภายใน Interface มี Constructor หรือไม่? เพราะเหตุใด?

3.5.1. ไม่มี เพราะ ใน Interface ไม่มี Properties

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) ; 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

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

```
package Lab11Tank;

abstract class ClassicTank1 {

	public int HP;
	public int Str;
	public int Vit;
	public int BaseDamage;

	public abstract void SetHP(int x);
	public abstract void GetHP();
	public abstract void Attack(int x);

}//end abstract class
```

โค้ดโปรแกรมภายใน NormalTank

```
package Lab11Tank;

public class NmTank1 extends ClassicTank1{

	@Override
	public void SetHP(int x) {
		HP = x;
	}

	@Override
	public void GetHP() {
		System.out.println("Tank 1 Have HP " + HP);
	}

	@Override
	public void Attack(int x) {
		System.out.println("Tank 1 Take DMG " + x);
		HP = HP - x;
	}

}//end class
```

```java
package Lab11Tank;

public class NmTank2 extends ClassicTank1{

	@Override
	public void SetHP(int x) {
		HP = x;
	}

	@Override
	public void GetHP() {
		System.out.println("Tank 2 Have HP " + HP);
	}

	@Override
	public void Attack(int x) {
		System.out.println("Tank 2 Take DMG " + x);
		HP = HP - x;
	}

}//end class
```

โค้ดโปรแกรมภายในฟังก์ชันการทำงานหลัก

```
package Lab11Tank;

import java.lang.Math;

public class main {
	public static float random(double d, double e) {
		return (float) ( (float)(Math.random() * (e - d )) +d);
    }
	public static void main(String[] args) {

		int turn = 1;
		int DamagePoint = 0;

		NmTank1 Nt1 = new NmTank1();
		NmTank2 Nt2 = new NmTank2();

		//Tank A
		((NmTank1) Nt1).SetHP(200);
		Nt1.Str = 12;
		Nt1.Vit = 9;
		Nt1.BaseDamage = 11;

		//Tank B
		Nt2.SetHP(250);
		Nt2.Str = 8;
		Nt2.Vit = 10;
		Nt2.BaseDamage = 10;

		System.out.println("////////////////////////////");
		Nt1.GetHP();
		Nt2.GetHP();

		System.out.println("////////////////////////////");
		System.out.println("Start");

		do {
			System.out.println("////////////////////////////");
			System.out.println("Round = "+ turn);
			if(turn%2 == 0) {
				//Tank1 ATTACK
				DamagePoint = (int) (Nt1.BaseDamage * Math.floorDiv(
Nt1.Str , Nt2.Vit ) * random(0.7, 0.9) ) ;
				Nt2.Attack(DamagePoint);
				Nt2.GetHP();
			} else {
				//Tank2 ATTACK
				//floor(8/9) = 0 Tank2 DamagePoint == 0 ?
					DamagePoint = (int) (Nt2.BaseDamage *
						Math.floorDiv( Nt2.Str ,
Nt1.Vit ) * random(0.7, 0.9) ) ;
				Nt1.Attack(DamagePoint);
				Nt1.GetHP();
			}
			turn++;
			if( Nt1.HP <= 0 || Nt2.HP <= 0) {
```

```
                        break;
                    }
                    System.out.println("////////////////////////////");
            }while(turn !=0 );
            System.out.println("////////////////////////////");
            if(Nt1.HP <= 0 ) {
                    System.out.println("Tank 2 WIN!!!!!!");
            }else if(Nt2.HP <= 0) {
                    System.out.println("Tank 1 WIN!!!!!!");
            }
            System.out.println("////////////////////////////");
      }
}//end class
```

ผลลัพธ์การทำงานของโปรแกรม

```
////////////////////////////
Tank 1 Have HP 200
Tank 2 Have HP 250
////////////////////////////
Start
////////////////////////////
Round = 1
Tank 1 Take DMG 0
Tank 1 Have HP 200
////////////////////////////
////////////////////////////
Round = 2
Tank 2 Take DMG 9
Tank 2 Have HP 241
////////////////////////////
////////////////////////////
Round = 3
Tank 1 Take DMG 0
Tank 1 Have HP 200
////////////////////////////
////////////////////////////
Round = 4
Tank 2 Take DMG 9
Tank 2 Have HP 232
////////////////////////////
////////////////////////////
Round = 5
Tank 1 Take DMG 0
Tank 1 Have HP 200
////////////////////////////
////////////////////////////
Round = 6
Tank 2 Take DMG 7
Tank 2 Have HP 225
////////////////////////////
////////////////////////////
Round = 7
Tank 1 Take DMG 0
Tank 1 Have HP 200
////////////////////////////
```

4.4. เปลี่ยน Abstract Class ให้กลายเป็น Interfaces และเปรียบเทียบผลลัพธ์การทำงานของโปรแกรม

หลังจากเปลี่ยน **Abstract Class** เป็น **Interface** แล้ว เกิดอะไรขึ้นอย่าง? อธิบายพร้อมยกตัวอย่างประกอบให้ชัดเจน

```
abstract class ClassicTank1 {

    public int HP;
    public int Str;
    public int Vit;
    public int BaseDamage;


public class NmTank1 extends ClassicTank1{
```

```
interface ClassicTank1 {



public class NmTank1 implements ClassicTank1{

    public int HP;
    public int Str;
    public int Vit;
    public int BaseDamage;
```

---

*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 แบบ เหมือนกัน 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