# ใบงานการทดลองที่ 11 เรื่อง การใช้งาน Abstract และ Interface

## จุดประสงค์ทั่วไป

### รู้และเข้าใจการกำหนดวัตถุ การใช้วัตถุ การซ่อนวัตถุ และการสืบทอดประเภทของวัตถุ

### รู้และเข้าใจโครงสร้างของโปรแกรมเชิงวัตถุ

## เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

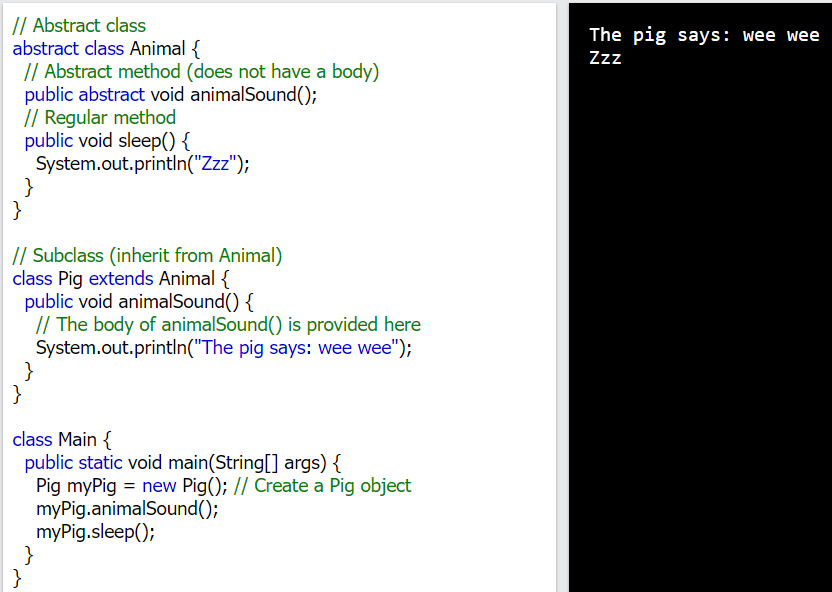
## ทฤษฎีการทดลอง

### Abstract Class คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

#### คือ Class ที่เอาไว้ซ่อนรายละเอียด จุดเด่น คือ จะมี Abstract Method

ซึ่ง Abstract Method จะไม่มีรายละเอียดของ Method อยู่ข้างใน

ถ้าอยากจะใช้งานต้องสืบทอดไปอีกทีหนึ่ง



### Interfaces คืออะไร? มีลักษณะการทำงานอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

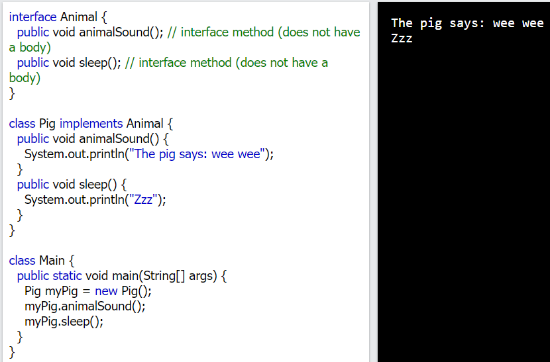
#### คือ Abstract Class ที่สมบูรณ์แบบ

#### สมบูรณ์แบบในที่นี้ คือ ใน Method ของ Interfaces จะไม่มีรายละเอียดอยู่เลย

ถ้าอยากจะใช้งานต้อง implements ไป ถึงจะใช้งานได้

Interfaces ไม่สามารถที่จะสร้าง instance ตรงๆได้ จะต้องสืบทอดไป

แล้วไปสร้าง instance ใน class ลูก อีกที่หนึ่งถึงจะทำได้



### คำสั่ง extends และ implements มีการใช้งานที่แตกต่างกันอย่างไร?

#### Extends ใช้กับ Class และ Abstract Class

Implements ใช้กับ Interfaces

### ภายใน Abstract Class มี Constructor หรือไม่? เพราะเหตุใด?

#### มีได้ เพราะ มีการประกาศ Properties ใน class

### ภายใน Interface มี Constructor หรือไม่? เพราะเหตุใด?

#### ไม่มี เพราะ ใน Interface ไม่มี Properties

## ลำดับขั้นการปฏิบัติการ

### ให้ผู้เรียนสร้าง Abstract Class ของรถถัง(ClassicTank) โดยจะต้องมีรายละเอียดดังต่อไปนี้

#### Properties : HP เพื่อกำหนดค่าพลังให้กับรถถัง

#### Properties : Str เพื่อกำหนดค่าความแรงในการยิงของรถถัง

#### Properties : Vit เพื่อกำหนดค่าพลังป้องกันของรถถัง

#### Properties : BaseDamage เพื่อการกำหนดค่าพลังการโจมตีพื้นฐาน

#### Method : SetHP( ) ; เพื่อทำการกำหนดค่าพลังเริ่มต้น

#### Method : GetHP( ) ; เพื่อตรวจสอบค่าพลัง ณ เวลาปัจจุบัน

#### Method : Attack( Tank Enemy ) ; เพื่อทำการยิงปืนใหญ่โจมตีศัตรู โดยการโจมตี จะเป็นการลดค่าพลังของรถถังฝั่งตรงกันข้าม (Enemy คือรถถังของศัตรู, Points คือค่าพลังโจมตีของเรา)

### ให้ผู้เรียนสร้างคลาส NormalTank เพื่อสืบทอด ClassicTank เพื่อเขียนรายละเอียดของ Method ทั้งหมดอันได้แก่ SetHP() , GetHP() , Attack( Tank Enemy )

### ในคลาสหลัก ให้สร้าง Instance จาก NormalTank อยู่จำนวน 2 คัน เพื่อทำการต่อสู้กัน โดยควรต้องมีบทบาทดังนี้

#### สร้างรถถัง A และ B ให้มีค่าพลังเบื้องต้นดังต่อไปนี้

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

#### รถถังทั้ง A และ B ผลัดกันโจมตีซึ่งกันและกัน เพื่อมุ่งหวังให้ค่าพลังของฝั่งตรงกันข้ามลดลงจนค่า HP = 0

#### รายละเอียดของพลังการโจมตีสามารถคำนวณได้ตามสมการดังต่อไปนี้

DamagePoint = MyTank\_BaseDamage \* Floor( MyTank\_Str / Enermy\_Vit ) \* Random( 0.7, 0.9 )

#### 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  **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 |
|  |

### เปลี่ยน Abstract Class ให้กลายเป็น Interfaces และเปรียบเทียบผลลัพธ์การทำงานของโปรแกรม

|  |
| --- |
| หลังจากเปลี่ยน Abstract Class เป็น Interface แล้ว เกิดอะไรขึ้นอย่าง? 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 |

## 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))

## คำถามท้ายการทดลอง

### เมื่อใดจึงควรเลือกใช้งาน Abstract Class

#### เมื่อต้องเขียนโปรแกรมที่มีProperties ซํ้ากันเยอะๆ หรือมีPropertiesที่เหมือนกันเยอะ เช่น HP STR

#### DEF AGI เป็นต้น

### เมื่อใดจึงควรเลือกใช้งาน Interface

#### เมื่อต้องเขียนโปรแกรมที่มีProperties ไม่ซํ้ากัน หรือ มีProperties เฉพาะเยอะ