Inheritance

saacsos

Inheritance การสืบทอด

- เป็นรูปแบบหนึ่งของการนำโค้ดกลับมาใช้
- เป็นการนำคลาสที่เคยประกาศแล้ว มาปรับปรุงแก้ไขให้เป็นคลาสใหม่
- คลาสที่เป็นต้นแบบเรียกว่า คลาสแม่ (superclass)
- คลาสที่ปรับปรุงจากคลาสแม่ เรียกว่า คลาสลูก (subclass)

Subclass สืบทอดอะไรจาก Superclass

- Subclass สืบทอด attributes จาก Superclass
  - ยกเว้น private attributes
- Subclass สืบทอด methods จาก Superclass
  - ยกเว้น private methods
- Subclass เหมาะใช้สร้าง object ที่มีความเจาะจงมากกว่า Superclass

หลักสำคัญในการสร้าง Subclass ข้อ 1

- Subclass ต้องมี attribute(s) เพิ่มเติมจาก Superclass
  - ถ้ามี attribute เหมือน Superclass

    ก็สร้างเป็น object ของ Superclass พอ
- Subclass ต้องมี method(s) เพิ่มเติมจาก Superclass
  - เนื่องจากมีการสร้าง attribute(s) เพิ่ม

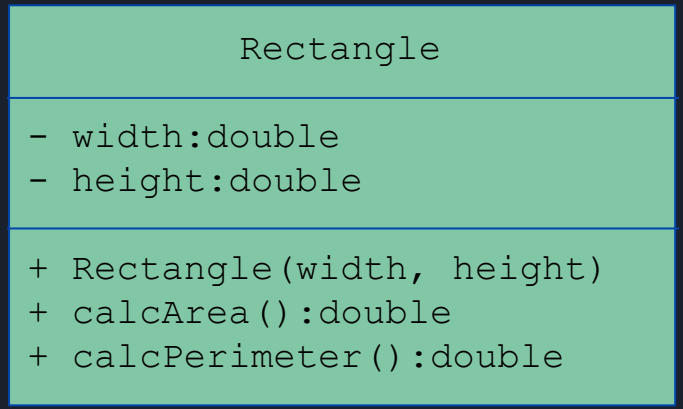
    ย่อมต้องมีการสร้าง method(s) เพื่อจัดการกับ attribute(s) ที่เพิ่มขึ้นด้วย

หลักสำคัญในการสร้าง Subclass ข้อ 2

- Subclass มีความสัมพันธ์แบบ is-a กับ Superclass
  - Subclass is-a Superclass

ตัวอย่าง การสืบทอด - Superclass Rectangle

```
class Rectangle {
   private double width;
   private double height;
   public Rectangle(double width, double height) {
      this.width = width;
      this.height = height;
   }
   public double calcArea() {
      return width * height;
   }
   public double calcPerimeter() {
      return 2 * (width + height);
   }
}
```



ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (1)

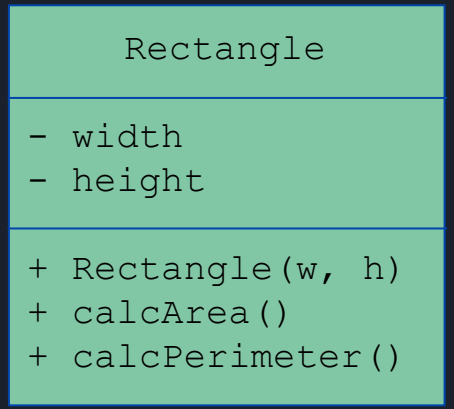
BigRectangle, SmallRectangle

- มี width กับ height เหมือนกับ Rectangle ต่างที่ค่า
- มี method คำนวณ area และ perimeter เหมือนกัน

สรุป ไม่ต้องสร้างเป็น Subclass

(เพราะไม่มี attribute เพิ่ม) สร้าง object ก็พอ

```
Rectangle big = new Rectangle(100, 200);
Rectangle small = new Rectangle(3, 2.4);
```

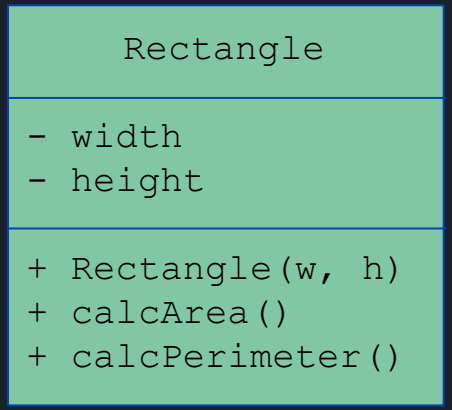


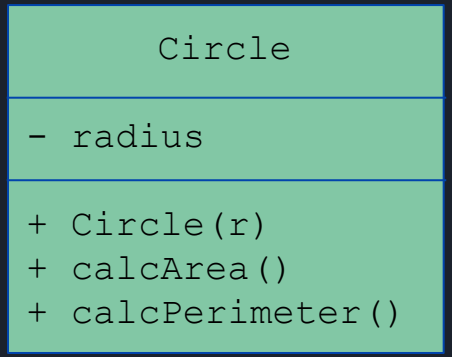
ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (2)

Circle

- ไม่มี width กับ height แบบ Rectangle
  แต่มี radius
- มี method คำนวณ area
  และ perimeter เหมือนกัน

สรุป ไม่ใช่ Subclass (เพราะไม่สืบทอด attribute)
ต้องสร้างคลาสใหม่แยกออกไป

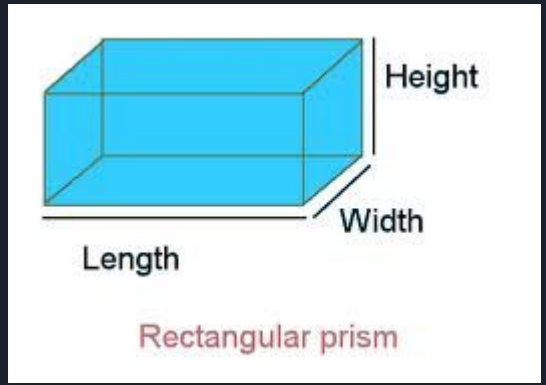




ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (3)

RectangularPrism

- มี width กับ height เหมือนกับ Rectangle แต่มี length เพิ่ม
- มี method คำนวณ area และ perimeter เหมือนกัน
  แต่มี method คำนวณ volumn เพิ่ม



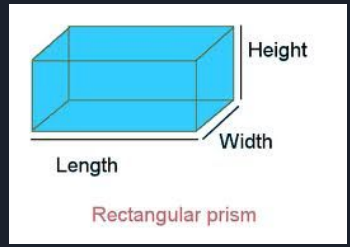
Rectangle
- width - height
+ Rectangle(w, h) + calcArea() + calcPerimeter()

ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (3)

RectangularPrism

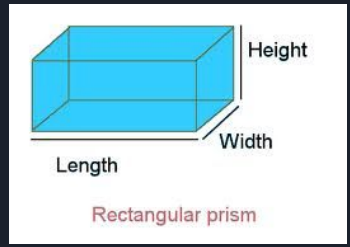
Rectangle
- width - height
+ Rectangle(w, h) + calcArea() + calcPerimeter()

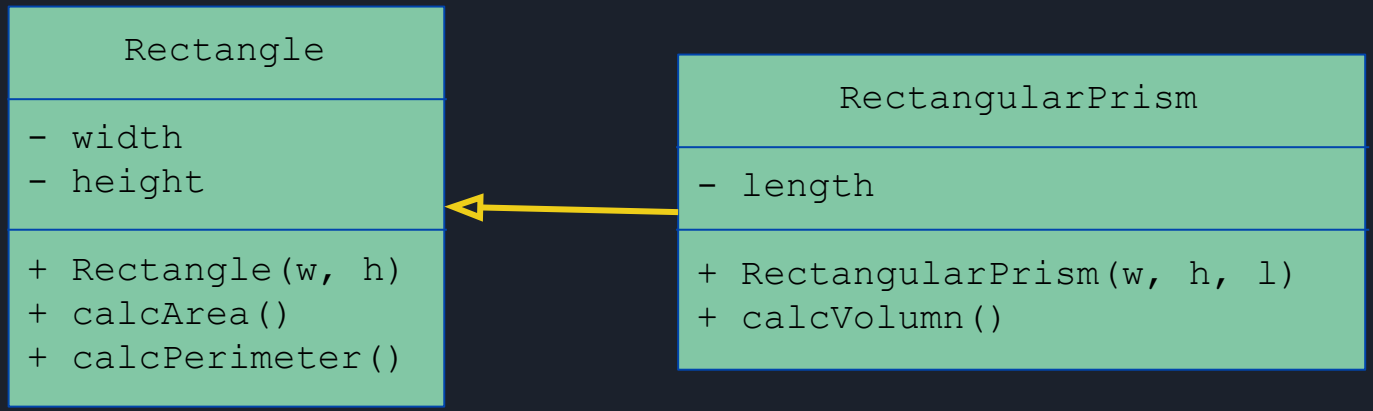
RectangularPrism
- width - height - length
+ RectangularPrism(w, h, l) + calcArea() + calcPerimeter() + calcVolumn()



ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (3)

RectangularPrism

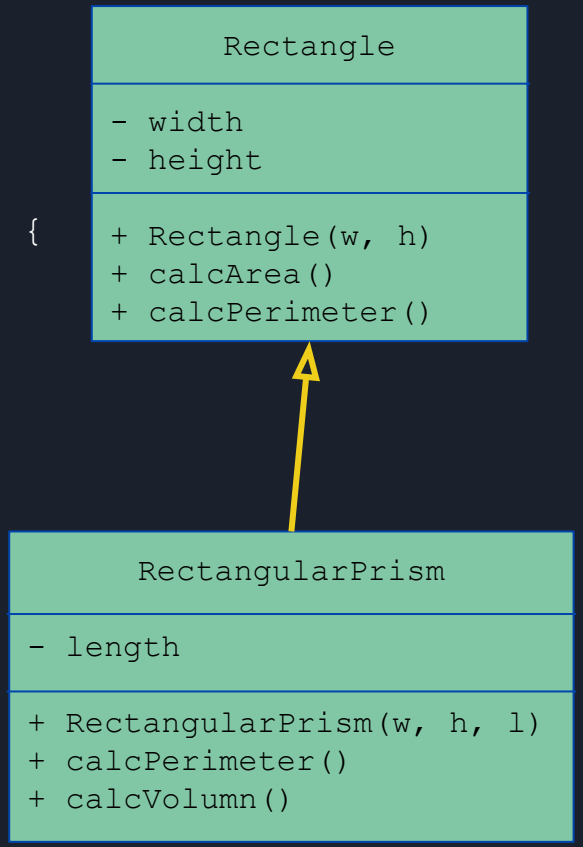




ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (3)

```
class RectangularPrism extends Rectangle {
   private double length;
   public RectangularPrism(double w, double h, double l) {
      super(w, h);
      length = l;
   }
   @Override
   public double calcPerimeter() {
      return super.calcPerimeter() * 2 + 4 * length;
 }

   public double calcVolumn() {
      return calcArea() * length;
   }
}
```



ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (3)

RectangularPrism

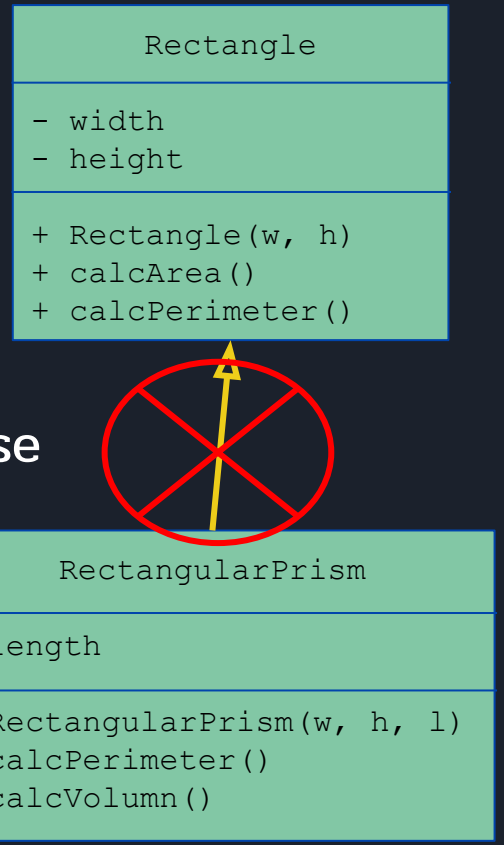
- แต่ไม่ใช่ความสัมพันธ์แบบ is-a (การเป็น)

  เป็นความสัมพันธ์แบบ has-a (การมี)

  RectangularPrism has-a Rectangle as base

สรุปใช้ Composition (over Inheritance)

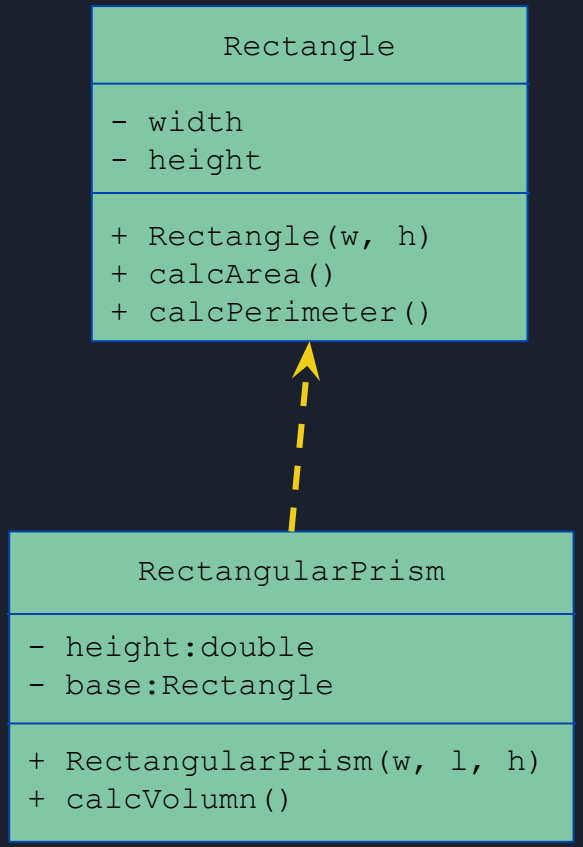
Composition ก็เป็นการนำโค้ดกลับมาใช้อีกแบบ



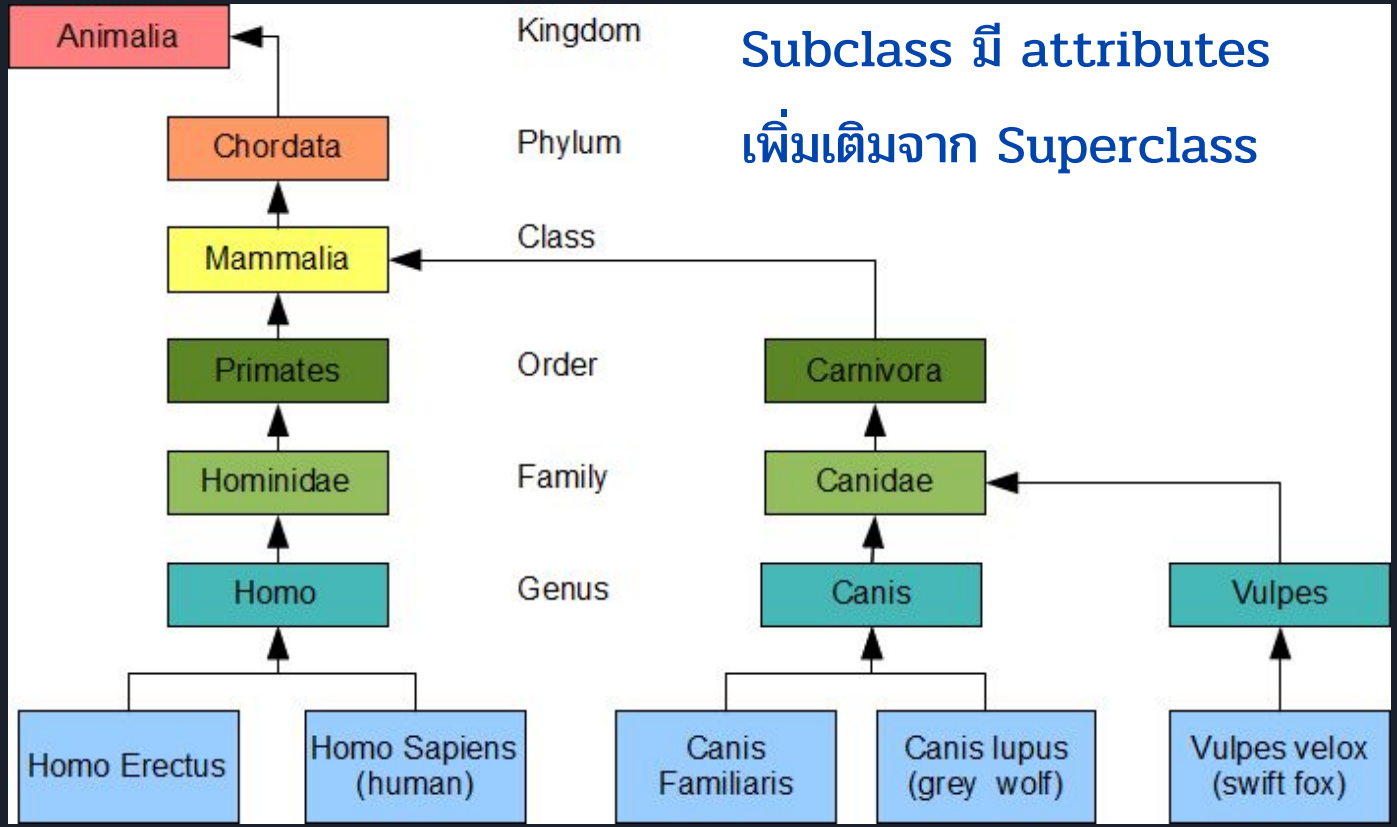
ตัวอย่าง การสืบทอด - Subclass ของ Rectangle (3)

```
class RectangularPrism {
   private Rectangle base;
   private double height;
   public RectangularPrism(double w, double l, double h)
{
      base = new Rectangle(w, l);
      height = h;
   }

   public double calcVolumn() {
      return base.calcArea() * height;
   }
}
```

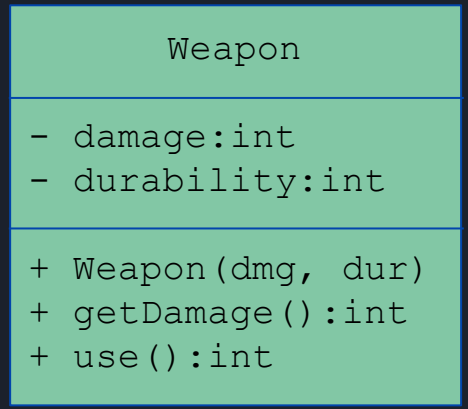


การสืบทอด - IS-A Relationship



ตัวอย่าง การสืบทอด - Superclass Weapon

```
class Weapon {
   private int damage;
   private int durability;
   public Weapon(int dmg, int dur) {
      damage = dmg;
      durability = dur;
   }
   public int getDamage() {
      return damage;
   }
   public void use() {
      if (--durability < 0) durability = 0;
   }
}
```

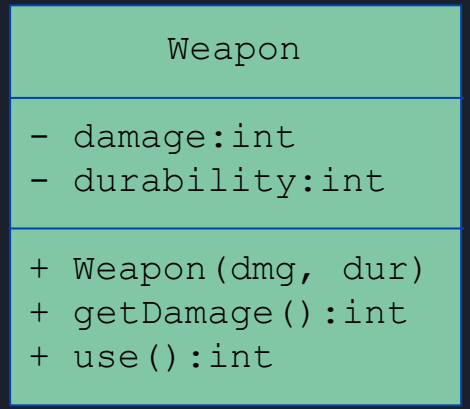


ตัวอย่าง การสืบทอด - Subclass ของ Weapon (1)

Gun

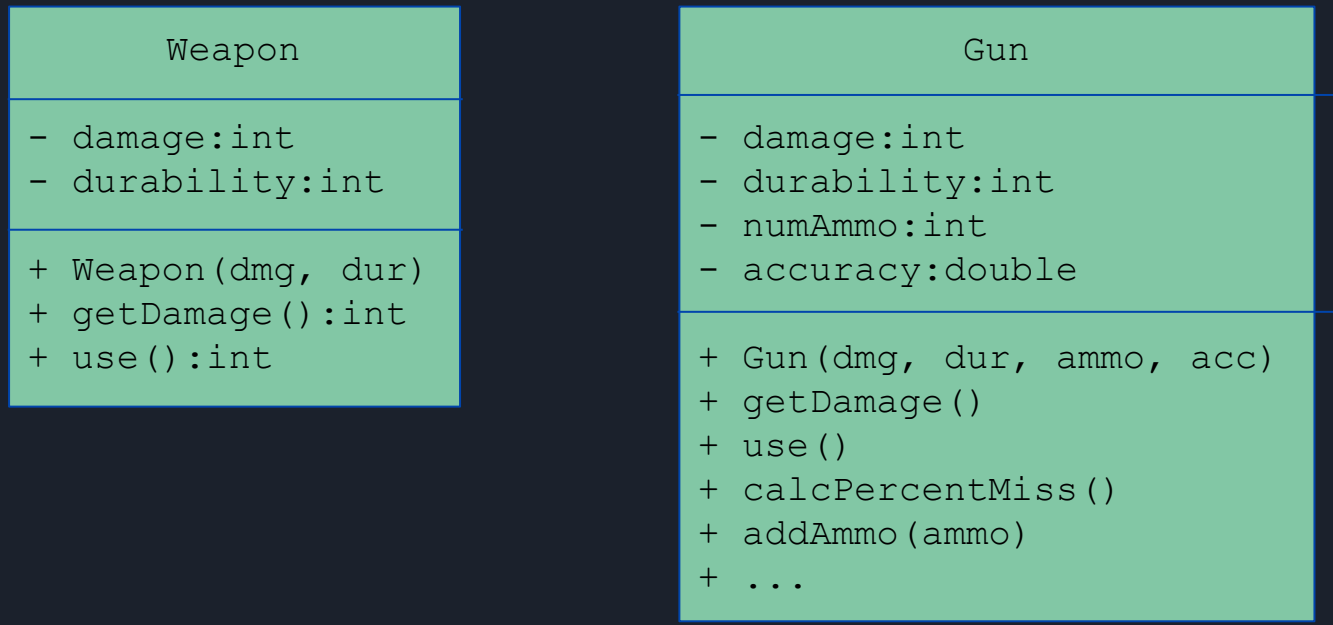
- มี damage กับ durability เหมือนกับ Weapon ต่างที่ค่า
- มีจำนวนกระสุน และความแม่นยำ เพิ่มจาก Weapon
- มี method เพิ่ม เพราะ attribute เพิ่ม
- มีความสัมพันธ์ Gun is-a Weapon

สรุป สร้าง Gun เป็น Subclass ของ Weapon



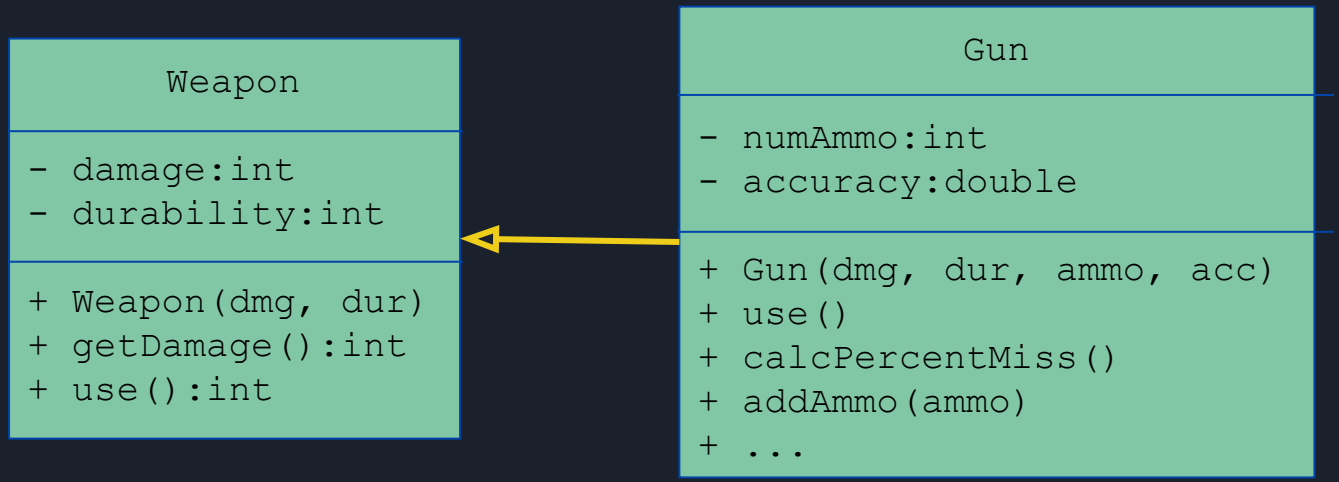
ตัวอย่าง การสืบทอด - Subclass ของ Weapon (1)

Gun



ตัวอย่าง การสืบทอด - Subclass ของ Weapon (1)

Gun



ตัวอย่าง การสืบทอด - Subclass ของ Weapon (1)

```
class Gun extends Weapon {
   private int numAmmo;
   private double accuracy;
   public Gun(int dmg, int dur, int ammo, double acc) {
      super(dmg, dur);
      numAmmo = ammo;
      accuracy = acc;
   }
   @Override
   public void use() {
      super.use();
      if (--numAmmo < 0) numAmmo = 0;
   }
   ...
}
```

```
Gun rpg = new Gun(3000, 1, 1, 90.0);
```

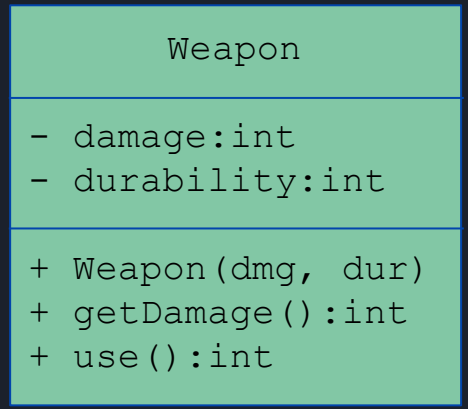
ตัวอย่าง การสืบทอด - Subclass ของ Weapon (2)

Sword

- มี damage กับ durability เหมือนกับ Weapon ต่างที่ค่า
- ไม่มี attribute เพิ่มจาก Weapon
- มีความสัมพันธ์ Sword is-a Weapon

สรุป สร้าง Sword เป็น object ของ Weapon

```
Weapon katana = new Weapon(1500, 200);
```



ตัวอย่าง การสืบทอด - Subclass ของ Weapon (3)

Bow

- มี damage กับ durability เหมือนกับ Weapon ต่างที่ค่า
- มีจำนวนกระสุน และความแม่นยำ เพิ่มจาก Weapon
- มี method เพิ่ม เพราะ attribute เพิ่ม
- มีความสัมพันธ์ Bow is-a Weapon

สรุป สร้าง Bow เป็น object ของ Gun

(เพราะ attribute เหมือน Gun)

```
Gun bow = new Gun(15, 40, 10, 80.0);
```