Chapter TWO

Inheritance and Polymorphism

Inheritance keyword “extends”

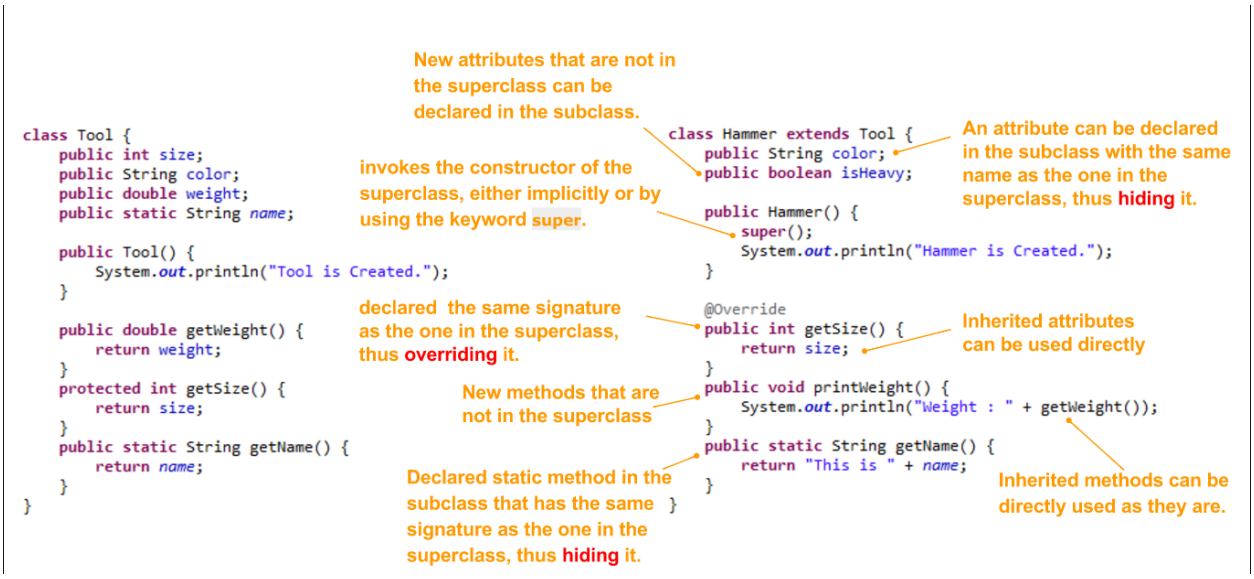
Java อนุญาติให้สืบทอดแบบ single superclass เท่านั้น (singular inheritance)

java.lang.Object ไม่มี superclass

java.lang.Object เป็น superclass ของทุกๆ class

มีความสัมพันธ์ระหว่าง subclass กับ superclass แบบ IS-A เช่น hammer IS A tools

the things you can do in a subclass



Overloading and Overriding

“method signature” - the method's name and the parameter types. Not include return type.

Overloading ชื่อเหมือนเดิม parameter เปลี่ยนได้ เพิ่ม-ลดได้

กฎของ Overload

“Java will look for the **CLOSEST** match **FIRST** (this means a larger type, a superclass, an autoboxed type, or the **MORE** particular type).**”**

```
class Print {
    static void printType(short param) {
        System.out.println("short");
    }
    static void printType(long param) {
        System.out.println("long");
    }
    static void printType(Integer param) {
        System.out.println("Integer");
    }
    static void printType(CharSequence param) {
       System.out.println("CharSequence");
    }

    public static void main(String[] args) {
        byte b = 1;
        int i = 1;
        Integer integer = 1;
        String s = "1";

        printType(b);
        printType(i);
        printType(integer);
        printType(s);
   }
}
```

The output is:

```
short
long
Integer
CharSequence
```

```
public class Question_2_3 {
    public static void print(Integer i) {
        System.out.println("Integer");
    }
    public static void print(Object o) {
        System.out.println("Object");
    }
    public static void main(String[] args) {
        print(null);
    }
}
```

What is the result?

A. `Integer`
B. `Object`
C. Compilation fails
D. An exception occurs at runtime

ตอบ A. ตามกฎ the **MORE** particular type ค่า null จะเข้า method ที่รับ parameter Integer เพราะเป็น type ที่เฉพาะเจาะจงกว่า Object และรับค่า null

```
// Can't find a match
double d = 1.0;
printType(d);
```

Overriding

กฎของการ overriding

1. Access Modifier ต้องเหมือนเดิมหรือสูงกว่าเดิม
2. Return Type ต้องเหมือนเดิมหรือเป็น subtype จากของเดิม
3. ชื่อ method ต้องเหมือนเดิม
4. Argument list type ต้องเหมือนเดิม
5. Exceptions ใน throws ต้องเหมือนเดิม ไม่ใส่เลยก็ได้ หรือต้องเป็น subclass จากของเดิม หรือ อาจจะ throws uncheck exception เลยก็ได้

Object class methods ที่ถูกนำมา override บ่อย ๆ

In Java, all objects inherit from java.lang.Object.

This class has the following methods that can be overridden (redefined):

protected Object clone() throws CloneNotSupportedException

protected void finalize() throws Throwable

public int hashCode()

public boolean equals(Object obj)

public String toString()

public int hashCode() มีข้อกำหนดไว้ ดังนี้

1. เมื่อเรียกใช้ hashCode() ของ object ตัวหนึ่งมากกว่า 1 ครั้ง ค่าที่ได้จะต้องเป็นค่าเดิมเสมอ กำหนดให้ค่า instance variable ภายใน object นั้นไม่มีการเปลี่ยนแปลง
2. เมื่อเปรียบเทียบ object สองตัวด้วย equals() แล้วปรากฏว่าเท่ากัน ค่า hash code ที่ได้จากการเรียก hashCode() ของแต่ละ object จะต้องเป็นค่าเดียวกัน
3. ถ้าเปรียบเทียบ object สองตัวด้วย equals() แล้วปรากฏว่าไม่เท่ากัน ค่า hash code ที่ได้จากการเรียก hashCode() ของแต่ละ object ไม่จำเป็นต้องเท่ากันก็ได้

public boolean equals(Object obj)

ถ้า override method equals มาจำเป็นจะต้อง override hashCode ด้วยเสมอ มีข้อกำหนดไว้ ดังนี้

1. reflexive : x.equals(x) ต้องเป็น true
2. symmetric : ถ้า x.equals(y) เป็น true y.equals(x) เป็น true เช่นกัน
3. transitive : ถ้า x.equals(y) เป็น true และ
   y.equals(z) เป็น true แล้ว
   x.equals(z) จะเป็น true ด้วย
4. consistence : ถ้า object ไม่เปลี่ยนแปลงเรียก x.equals(y) กี่ครั้ง ก็ต้องได้ผลเหมือนเดิม

5. ถ้า x ไม่ใช่ค่า null x.equals(null): ต้องเป็น false

public String toString()

It returns a string representation of the object. The toString method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character '@', and the unsigned hexadecimal representation of the hash code of the object.

Polymorphism

คือการที่ออบเจ็คสามารถมีได้หลายรูปแบบ ซึ่งเกิดจากการสืบทอดจาก super class และมันยังคงรักษาสภาพและคุณสมบัติของ superclass ไว้

```
class HumanBeing {
    public void dress() {
        System.out.println("Dressing a human being");
    }
}
class Man extends HumanBeing {
    public void dress() {
        System.out.println("Put on a shirt");
        System.out.println("Put on some jeans");
    }
}
class Woman extends HumanBeing {
    public void dress() {
        System.out.println("Put on a dress");
    }
}
class Baby extends HumanBeing {
    public void dress() {
        System.out.println(
            "I don't know how to dress!");
    }
}
```

```
HumanBeing[] someHumans = new HumanBeing[3];
someHumans[0] = new Man();
someHumans[1] = new Woman();
someHumans[2] = new Baby();
for(int i = 0; i < someHumans.length; i++) {
    someHumans[i].dress();
    System.out.println();
}
```

The output:

```
Put on a shirt
Put on some jeans
Put on a dress
I don't know how to dress!
```

JVM จะทำการตัดสินตอน Runtime ว่า method ที่เรียกมาจากการ new Object ใด ขั้นตอนนี้เรียกว่า *virtual method invocation (เป็นอีกชื่อเรียกของ overriding)*

Overriding is also known as *dynamic polymorphism* because the type of the object is decided at **RUN** time.
In contrast, overloading is also called *static polymorphism* because it's resolved at **COMPILE** time.

Abstract classes and methods

1. Abstract classes **CANNOT** be instantiated, only subclassed. They are declared with the `abstract` keyword
2. Abstract methods are declared **WITHOUT** an implementation (body)

Rules

1. keyword `abstract` ใช้ได้กับ classes หรือ non-static methods

```
abstract class AClass {
    // Compile-time error
    public static abstract void AMethod();
}
```

2. `abstract` class ไม่จำเป็นต้องประกาศ abstract method ก็ได้
3. class ที่มี abstract method อยู่จะต้องประกาศเป็น abstract class เท่านั้น
4. ถ้า subclass ของ abstract class ไม่ได้ implement method มาทั้งหมด จะต้องประกาศ subclass นั้นเป็น abstract class
5. abstract class สามารถ implements interfaceได้ เพราะ method ใน interface เป็น abstract method