

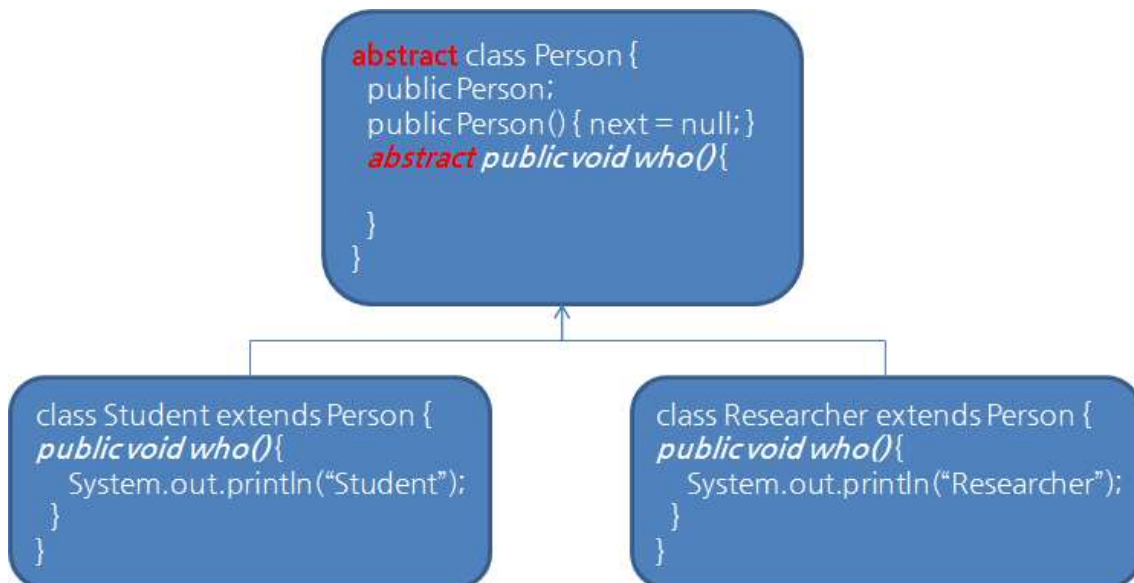
Week 12-1 : Interfaces

Part1. Abstract Class

- Abstract class' inheritance

When subclass is inherited from an abstract class, a subclass become an abstract class.

For preventing a subclass from becoming an abstract class, abstract methods have to be overridden.



Part2. Interfaces

One of the abstract classes, make a multiple inheritance possibly. 'implements' keyword means implementing abstract methods in the class.

- Feature of interface

- ① Only Being composed of abstract methods and constants.
- ② Every methods can omit 'abstract public'.
- ③ Constants can omit 'public static final'.
- ④ Not possibly generating interface's objects.

- ⑤ Possibly inherited from other interface.
- ⑥ Interface can be used as a type of reference variable.

- Example

```
interface CanFight {
    void fight();
}

interface CanSwim {
    void swim();
}

interface CanFly {
    void fly();
}

class ActionCharacter {
    public void fight() {
        System.out.println("fight");
    }
}

class Hero extends ActionCharacter implements CanFight, CanSwim, CanFly {
    public void swim() {
        System.out.println("swim");
    }

    public void fly() {
        System.out.println("fly");
    }
}

public class Adventure {
    public static void t(CanFight x) {
        x.fight();
    }

    public static void u(CanSwim x) {
        x.swim();
    }
}
```

```
    }  
  
    public static void v(CanFly x) {  
        x.fly();  
    }  
  
    public static void w(ActionCharacter x) {  
        x.fight();  
    }  
  
    public static void main(String[] args) {  
        Hero h = new Hero();  
        t(h); // Treat it as a CanFight  
        u(h); // Treat it as a CanSwim  
        v(h); // Treat it as a CanFly  
        w(h); // Treat it as an ActionCharacter  
    }  
}
```

-Example : interface including inheritance

```
interface Monster {  
    void menace();  
}  
  
interface DangerousMonster extends Monster {  
    void destroy();  
}  
  
interface Lethal {  
    void kill();  
}  
  
class DragonZilla implements DangerousMonster {  
    public void menace() {  
        System.out.println("menace1");  
    }  
  
    public void destroy() {  
        System.out.println("destory1");  
    }  
}
```

```
}  
  
interface Vampire extends DangerousMonster, Lethal {  
    void drinkBlood();  
}  
  
class VeryBadVampire implements Vampire {  
    public void menace() {  
        System.out.println("menace2");  
    }  
  
    public void destroy() {  
        System.out.println("destroy2");  
    }  
  
    public void kill() {  
        System.out.println("kill2");  
    }  
  
    public void drinkBlood() {  
        System.out.println("drinkBlood2");  
    }  
}  
  
public class HorrorShow {  
    static void u(Monster b) {  
        b.menace();  
    }  
  
    static void v(DangerousMonster d) {  
        d.menace();  
        d.destroy();  
    }  
  
    static void w(Lethal l) {  
        l.kill();  
    }  
  
    public static void main(String[] args) {  
        DangerousMonster barney = new DragonZilla();  
    }  
}
```

```
System.out.println("1");
u(barney);
System.out.println("2");
v(barney);
Vampire vlad = new VeryBadVampire();
System.out.println("3");
u(vlad);
System.out.println("4");
v(vlad);
System.out.println("5");
w(vlad);
}
}
```

☞ Possibly inheriting a multiple interface, making a interface. You can check a multiple inheritance using interface.

- Interface vs Abstract class

Abstract class	Interface
<ul style="list-style-type: none">- Including not abstract methods.- Constant and variable fields can include.- Abstract class is suitable when every subclasses have a same method.	<ul style="list-style-type: none">- Every methods are abstract methods.- Only constant fields include.- supporting a multiple inheritance.

[Exercise]

Implement a Messenger and Video as interfaces, and Implement a Device class. Inherit a Calculator, Messenger, and Video.

The following box show methods that implemented in a class or interfaces.

```
Messenger : sendMessage, receiveMessage  
Video : play, stop  
Calculator : adder(x,y)  
Device : version
```

```
public class Device .... {  
  
    /* BLANK */  
  
    public static void main (String args[]) {  
        Device d = new Device();  
        d.sendMessage();  
        d.play();  
        d.adder(1, 2);  
        d.version();  
    }  
}
```