SWEN20003
Object Oriented Software Development

Inheritance and Polymorphism Lecture 1 - Questions

Bach Le
bach.le@unimelb.edu.au

University of Melbourne

The Road So Far

- Subject Introduction
- A Quick Tour of Java
- Classes and Objects
- Software Tools and Bagel
- Arrays and Strings
- Input and Output

Learning Outcomes

Upon completion of this topic you will be able to:

- Use **inheritance** to abstract common properties of classes
- Explain the relationship between a **superclass** and a **subclass**
- Make better use of **privacy** and **information hiding**
- Identify errors caused by **shadowing** and **privacy leaks**, and avoid them
- Describe and use method **overriding**
- Describe the **Object** class, and the properties inherited from it
- Describe what **upcasting** and **downcasting** are, and when they would be used
- Explain **polymorphism**, and how it is used in Java
- Describe the purpose and meaning of an **abstract** class

Lecture Summary

- Introduction and Motivation
- Inheriting Attributes
- Inheriting and Overriding methods

Inheritance

Keyword

Inheritance: A form of abstraction that permits "generalisation" of similar attributes/methods of classes; analogous to passing genetics on to your children.

Inheritance

Keyword

Superclass: The "parent" or "base" class in the inheritance relationship; provides general information to its "child" classes.

Keyword

Subclass: The "child" or "derived" class in the inheritance relationship; inherits common attributes and methods from the "parent" class.

True or false?

Methods of the superclass can access public attributes of a subclass.

false

An overridden method in a subclass must have the same signature as that of the superclass.

true

When a method it overridden, the method that will execute will depend on the type of reference used to access the object.

false

What java keyword is used to define an inheritance relationship?

extends

Inheritance indicates a ..... relationship.

is-a

What keyword is used to access the superclass constructor from the subclass constructor?

super

What annotation is used to indicate that a method is an overridden method?

@Override

What is the output of the following program?

```
class A  {
    public void printInfo() {
        System.out.println(1);
    }
}
class B extends A {
    public void printInfo() {
        System.out.println(2);
        super.printInfo();
    }
}
class C extends B {
    public void printInfo() {
        System.out.println(3);
        super.printInfo();
    }
}
public class TestABC {
    public static void main(String[] args) {
        C c = new C();
        c.printInfo();
    }
}
```

Answer:

3

2

1

What is the output of the following program?

```
class A  {
    public void printInfo() {
        System.out.println(1);
    }
}
class B extends A {
    public void printInfo() {
        super.printInfo();
        System.out.println(2);
    }
}
class C extends B   {
    public void printInfo() {
        super.printInfo();
        System.out.println(3);
    }
}
public class TestABC {
    public static void main(String[] args) {
        C c = new C();
        c.printInfo();
    }
}
```

Answer:

1

2

3

```
public class AA extends BB {
}
class BB extends AA {
}
```

Which of the following statements regarding the above class definition are true?

1. Class AA is a subclass of class BB
2. Class BB is a subclass of class AA
3. Class AA is a parent class of class BB
4. The above definition is not valid

What is the output of the following program?

```
class X {
    static void staticMethod() {
        System.out.println("Class X");
    }
}
class Y extends X {
    static void staticMethod() {
        System.out.println("Class Y");
    }
}
public class TestXYZ {
    public static void main(String[] args) {
        Y.staticMethod();
    }
}
```

Answer:

Class Y

What is the problem the following class definitions? Can you fix it?

```
public class XX {
    XX(int x) {
        System.out.println(1);
    }
}
public class YY extends XX {
    YY(int i) {
        System.out.println(2);
        super(i);
    }
}
public TestYY {
    public static void main(String[] args) {
        System.out.println("Hello");
        YY y = new YY(1);
    }
}
```

Answer:

```
public class YY extends XX {
    YY(int i) {
        super(i);
        System.out.println(2);
    }
}
```