SWEN20003
Object Oriented Software Development

Inheritance and Polymorphism Lecture 2 - Questions

Bach Le
bach.le@unimelb.edu.au

University of Melbourne

The Road So Far

- Subject Introduction
- Java Introduction
- Classes and Objects
- Arrays and Strings
- Input and Output
- Software Tools

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

Inheritance and Polymorphism Lecture 2 - Questions

Topics covered in the last lecture:

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

Topics for this lecture:

- Inheritance and Information Hiding
- The Object Class
- Abstract Classes

Answer true of false:

Private methods of a superclass can be overridden in a subclass.

false

A public method in a superclass can be made protected when overriding it in the subclass

false

A protected method in a superclass can be made public when overriding it in the subclass

true

An abstract class can be final

false

What is the problem with the following class definitions. How can you fix it?

```java
public class Shape {
    private double centreX;
    private double centreY;
    public void move(double x, double y) {
    }
}
class Circle extends Shape {
    private double radius;
    public void move(double x, double y) {
        super.centreX = x;
        super.centreY = y;
    }
    public static void main(String[] args) {
        Shape c = new Circle();
        c.move(1.3, 2.5);
    }
}
```

Answer:

Attempting to access private attributes in the superclass. Access them using public setter in the superclass.

How would you restrict a subclass extending superclass methods?

Answer:
By defining them as final.

What is the correct signature for the `equals` method?

Answer:

```
public boolean equals(Object o) {}
```

The following is an example of ....

```
Shape s = new Circle(1.4, 3.5);
```

Answer: Upcasting

Which ones of the following Java definitions are valid?

1. `public class B extends A{....}` - Single Inheritance
2. `public class A extends B,C{....}` - Multiple Inheritance
3. `public class B extends A{....}`
   `public class C extends A{....}` - Hierarchical Inheritance
4. `public class B extends A{....}`
   `public class C extends B{....}` - Multi-level Inheritance
5. `public class B extends A{....}`
   `public class D extends B,C{....}` - Hybrid Inheritance
6. `public class B extends A{....}`
   `public class C extends A{....}`
   `public class D extends B,C{....}` - Multi-path Inheritance

Answer: 1, 3, 4

What are the different ways in which you can check the class of an Object?

Answer:

Keyword

getClass: Returns an object of type `Class` that represents the details of the *calling object's class*.

Keyword

instanceof: An *operator* that gives `true` if an object A is an instance of the same class as object B, or a class that inherits from B.

What is the output of the following program?

```
class SuperClass
{
    public int i = 6;
}
```

```
class SubClass extends Superclass
{
    public int i = 4;
}
```

```
public class MainClass
{
    public static void main(String[] args)
    {
        SuperClass s = new Subclass();
        System.out.println(s.i);
    }
}
```

Answer:

6

Which line of the following code has an error and how would you fix the error?

```java
1   public abstract class Shape {
2       private double centreX;
3       private double centreY;
4       public abstract void move(double x, double y){};
5   }
```

Answer: line 4

```java
1   public abstract class Shape {
2       private double centreX;
3       private double centreY;
4       public abstract void move(double x, double y);
5   }
```

Polymorphism

What is polymorphism?

Keyword

Polymorphism:
The ability to use objects or methods in many different ways; roughly means "multiple forms".

Overloading	same method with various forms depending on **signature** (Ad Hoc polymorphism)
Overriding	same method with various forms depending on **class** (Subtype polymorphism)
Substitution	using subclasses in place of superclasses (Subtype polymorphism)
Generics	defining parametrised methods/classes (Parametric polymorphism, *coming soon*)

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