

Inheritance and Polymorphism

Super classes and Subclasses

Introduction

Suppose you want to define classes to model circles, rectangles, and triangles. These classes have many common features. What is the best way to design these classes so to avoid redundancy? The answer is to use inheritance.

Super classes and Subclasses

Inheritance enables you to define a general class (a superclass) and later extend it to more specialized classes (subclasses).

Different classes may have some common properties and behaviors, which can be generalized in a class that can be shared by other classes. You can define a specialized class that extends the generalized class. The specialized classes inherit the properties (data fields) and methods from the general class.

Super classes and Subclasses

In Java terminology, a class **C1** extended from another class **C2**, is called a *subclass*, and **C2** is called a *superclass*.

A superclass is also referred to as a *parent class* or a *base class*, and a subclass as a *child class*, an *extended class*, or a *derived class*.

A subclass inherits accessible data fields and methods from its superclass and may also add new data fields and methods.

Super classes and Subclasses

Consider geometric objects. Suppose you want to design the classes to model geometric objects such as circles and rectangles. Geometric objects have many common properties and behaviors. They can be drawn in a certain color and be filled or unfilled.

Thus a general class **GeometricObject** (superclass) can be used to model all geometric objects. This class contains the properties **color** and **filled** and their appropriate **get** and **set** methods.

Assume that this class also contains a **toString()** method that returns a string representation of the object.

Super classes and Subclasses

Since a circle is a special type of geometric object, it shares common properties and methods with other geometric objects.

Thus it makes sense to define the **Circle** as a subclass of **GeometricObject**. This means that **Circle** class inherits from (extends) the **GeometricObject** class. Likewise, **Rectangle** can also be defined as a subclass of **GeometricObject**.

The UML diagrams in the next slide show the relationship among these classes. A triangular arrow pointing to the superclass is used to denote the inheritance relationship between the two classes involved.

Super classes and Subclasses

GeometricObject	
-color: String -filled: boolean	The color of the object (default: white). Indicates whether the object is filled with a color (default: false).
+GeometricObject() +GeometricObject(color: String, filled: boolean) +getColor(): String +setColor(color: String): void +isFilled(): boolean +setFilled(filled: boolean): void +toString(): String	Creates a GeometricObject. Creates a GeometricObject with the specified color and filled values. Returns the color. Sets a new color. Returns the filled property. Sets a new filled property. Returns a string representation of this object.



Circle
-radius: double
+Circle(radius: double) +Circle(radius: double, color: String, filled: boolean) +getRadius(): double +setRadius(radius: double): void +area(): double +perimeter(): double + toString(): String

Rectangle
-width: double -height: double
+Rectangle(width: double, height: double) +Rectangle(width: double, height: double, color: String, filled: boolean) +getWidth(): double +setWidth(width: double): void +getHeight(): double +setHeight(height: double): void +area(): double +perimeter(): double +toString(): String

You can download and run the java project [TestingInheritance](#).

Class GeometricObject

```
public class GeometricObject {  
  
    private String color ;  
  
    private boolean filled;  
  
    public GeometricObject() { // No arg constructor  
  
        color = "white";  
  
        filled = false;  
  
    }  
  
    public GeometricObject(String c, boolean f) {  
  
        color = c;  
  
        filled = f;  
  
    }  
  
    public String getColor() {  
  
        return color;  
  
    }  
  
    public void setColor(String c) {  
  
        color = c;  
  
    }  
  
    public boolean isFilled() {  
  
        return filled;  
  
    }  
  
    public void setFilled(boolean f) {  
  
        filled = f;  
  
    }  
}
```



```
    public String toString() {  
        String s = "The color is " + color;  
        if (filled)  
            s+= ".\nThe geometric object is filled";  
        else  
            s+= ".\nThe geometric object is not filled";  
        return s;  
    }  
} // end class GeometricObject
```

Class Circle

```
public class Circle extends GeometricObject {

    private double radius;

    public Circle(String c, boolean f, double r) {

        setColor(c);

        setFilled(f);

        radius = r;
    }

    public double getRadius() { return radius; }

    public void setRadius(double radius) {

        this.radius = radius;

    }

    public double area() {

        return radius * radius * Math.PI;

    }

    public double perimeter() {

        return 2 * radius * Math.PI;

    }

    public String toString() {

        String s = "The color is " + getColor();

        if (isFilled())

            s+= ".\nThe circle is filled";

        else

            s+= ".\nThe circle is not filled";

        s+= ".\nThe radius is " + radius;

        return s;
    }

} //end class Circle
```

Class Rectangle

```
public class Rectangle extends GeometricObject {  
  
    private double width;  
  
    private double height;  
  
    public Rectangle(String c, boolean f, double width, double height) {  
  
        setColor(c);  
  
        setFilled(f);  
  
        this.width = width;  
  
        this.height = height;  
  
    }  
  
    public double getWidth() {  
  
        return width;  
  
    }  
  
    public void setWidth(double width) {  
  
        this.width = width;  
  
    }  
  
    public double getHeight() {  
  
        return height;  
  
    }  
  
    public void setHeight(double height) {  
  
        this.height = height;  
  
    }  
  
    public double area() {  
  
        return width * height;  
  
    }  
}
```



```
    public double perimeter() {  
        return 2 * (width + height);  
    }  
  
    public String toString() {  
        String s = "The color is " +  
                    getColor();  
        if (isFilled())  
            s += ".\nThe rectangle is  
                filled";  
        else  
            s += ".\nThe rectangle is  
                not filled";  
        s += ".\nThe height is " + height;  
        s += ".\nThe width is " + width;  
        return s;  
    }  
} // end class Rectangle
```

Application

```
public class TestingInheritance {

    public static void main(String[] args) {

        Circle c1 = new Circle("red", true, 2);

        System.out.println(c1.getColor());

        System.out.println(c1.getRadius());

        System.out.println(c1);

        GeometricObject g1 = new GeometricObject("black", false);

        System.out.println(g1.getColor());

        // getRadius() will give an error since class GeometricObject does not
        // have a method called getRadius()

        // System.out.println(g1.getRadius());

        System.out.println(g1);

        Rectangle r1 = new Rectangle("green", true, 3,4);

        System.out.println(r1.getColor());

        System.out.println(r1.getHeight());

        System.out.println(r1);

    }

}
```

Super classes and Subclasses

The **Circle** class inherits all accessible data fields and methods from the **GeometricObject** class. In addition, it has a new data field, **radius**, and its associated **get** and **set** methods. The **Circle** class also contains the **area()**, **perimeter()**, and **toString()** methods for returning the area, perimeter, and a string representation of the circle.

The **Rectangle** class inherits all accessible data fields and methods from the **GeometricObject** class. In addition, it has the data fields **width** and **height** and their associated **get** and **set** methods. It also contains the **area()**, **perimeter()**, and **toString()** methods for returning the area, perimeter, and a string representation of the rectangle.

Super classes and Subclasses

As explained in the previous slide a subclass inherits (extends) data fields and methods from the superclass.

In the subclass you can also:

- ➡ Add new data fields
- ➡ Add new methods
- ➡ Override the methods of the superclass (e.g. The toString() method in class **Circle** is a modified version of the toString() method in class **GeometricObject**). Details about overriding will be provided later.