



CSE 215: Programming Language II Lab

Lab – 10

Polymorphism

Objective:

- To understand polymorphism and its usage
- To utilize polymorphism to ensure reusability of existing code

Related Class

```
package Lab10;
import java.util.Date;
public class GeometricObject {
    private String color = "White"; // setting white as default color
    private boolean filled;
    private Date dateCreated;
    public GeometricObject(){
        dateCreated = new Date();
    }
    public GeometricObject(String color, boolean filled) {
        this.color = color; // "this" refers to the current object
        this.filled = filled;
        dateCreated = new Date();
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public boolean getFilled() {
        return filled;
    }
    public void setFilled(boolean filled) {
        this.filled = filled;
    }
    public Date getDateCreated() {
        return dateCreated;
    }
    public String toString(){
        return "Created on: "+dateCreated+" Color: "+color+" Filled: "+filled;
    }
}
```

<pre> Public Circle(String color, boolean filled, double radius){ super(color, filled); this.radius = radius; } </pre>	<pre> public String toString(){ return super.toString()+" radius: "+radius; } </pre>
--	--

Polymorphism

Polymorphism means that a variable of a supertype can refer to a subtype object.

A type defined by a subclass is called a subtype, and a type defined by its superclass is called a supertype. Therefore, you can say that **Circle is a subtype of GeometricObject** and **GeometricObject is a supertype for Circle**.

```

public class PolymorphismDemo {
    public static void main(String[] args) {
        displayObject(new Circle1 (1, "red", false));
    }

    public static void displayObject(GeometricObject object) {
        System.out.println("Created on " + object.getDateCreated() + 14 ". Color is "
+ object.getColor());
    }
}

```

Dynamic Binding

A method can be implemented in several classes along the inheritance chain. The JVM decides which method is invoked at runtime.

For example, the `toString()` method is defined in the `Object` class and overridden in `GeometricObject`.

```

Object o = new GeometricObject();
System.out.println(o.toString());

```

Here `o`'s actual type is `GeometricObject`, because `o` references an object created using `new GeometricObject()`. Which `toString()` method is invoked by `o` is determined by `o`'s actual type. To check if an `Object o` is a subclass/ instance of specified type class or subClass, Java use **instanceof** operator.

```

o instanceof GeometricObject
//Lab task is based on using instanceof operator

```

Lab Task

Design a class named **Triangle** that extends **GeometricObject**. The class contains:

- ☐ Three double data fields named **side1**, **side2**, and **side3** with default values 1.0 to denote three sides of the triangle.
- ☐ A no-arg constructor that creates a default triangle.
- ☐ A constructor that creates a triangle with the specified side1, side2, and side3.
- ☐ The accessor methods for all three data fields.
- ☐ A method named **getArea()** that returns the area of this triangle.
- ☐ A method named **getPerimeter()** that returns the perimeter of this triangle.
- ☐ A method named **toString()** that returns a string description with values of three sides of the triangle.

Design a class name Rectangle that extends **GeometricObject** (Same features as **Triangle** class)

- Define a method name **displayObject(Object object)** that returns the **area**, **perimeter**, **string description (toString())** of a **Rectangle** if object is **Rectangle**, and of a **Triangle** if object is triangle.
- Write a test program that prompts the user to enter three sides of the triangle, a color, and a boolean value to indicate whether the triangle is filled. Create a Triangle object (declared/reference type **Object**) with these inputs.
- Same way create Rectangle object from user input.
- Call **displayObject(Object object)** and print appropriate result.