CS 250 - Fall 2021 - Lab 02: Inheritance and Polymorphism

Available: Sep 1, 2021.
Due date: **Sep 6, 2021, 11:59 PM**

Objectives:

The objectives of this lab are:

- Familiar with the concept of inheritance and superclass-subclasses relationships.
- Familiar with the concept of polymorphism and dynamic binding.

Prerequisite:

- Read Chapter 8 of Java, Java, Java, 3E, and read through this document.
- Read online tutorials of Inheritance and Polymorphism on the javatpoint website.

Part 1: Design the hierarchy of the 2D Shape family classes

1.1 General specifications

In this lab, we will practice programming inheritance and polymorphism on the 2D Shape family classes. Suppose we want to write a Java application that manages shapes that has these requirements:

- Shapes can be circles, rectangles, or others (triangles, polygons, etc.)

- Shapes exist in the 2D space

- Shapes can be colored

- The areas and perimeters can be computed in the applications

1.2 Design UML diagram

We often use UML diagrams to show the relationships between classes that are related by inheritance. In UML notations, the classes are described as follows:

- A class is just a rectangular box with the class name in it. If this name is in italics, it means that the class is an **abstract** class. This rectangular box is divided into three sub boxes: the top one displays the class name, the middle one presents the list of attributes, and the bottom one displays the list of actions.

- The inheritance relationship is shown by a directed solid arrow from the subclass (derived class) to the superclass (parent class)
- An association is when one class uses/owns another class as part of its data-member declaration. These are shown by a solid arrow – from the user/owner class to the used/owned class (the type used for the data member)

Part 2: Implement the abstract class Shape2D

Step 1. Run the Eclipse IDE program

NOTE: Remember how to find this program because you're going to need it every week.

Step 2. Create the Lab02 project.

- Click on the "Create a java project" link in the Package Explorer window or right-click on that window then select NEW then Java project. Enter the name of your project, i.e. "Lab02". The "Location" text box should indicate that your project will be stored on the desktop. In the project JRE setting, make sure at least one Java SE version is selected, default to JavaSE-1.8 if you followed instructions on the Prerequisites section. Click on Finish to create the project.

- In the Package Explorer window, right-click on the "src" item, then select "New" and then Class. Next, create the Shape 2D family Java classes following your UML design.

Step 3. Implement the abstract class Shape2D

This class is the parent class for all the shapes we want to define in this family class. In this class, you should define common attributes and abstract methods. Below is the list of attributes and methods that should be defined:

- A variable to keep track of the color of the shape
- Two variables to store the center location of the shape in 2D coordinates (x,y)
- get() and set() methods for each variable
- A default constructor method for this class
- A constructor method that set up color and center location properties
- Two abstract methods calcArea() and calcPerimeter() to calculate area and perimeter
- toString() method to return the string representation of the shape object

Part 3: Implement the Circle class

The Circle class is a subclass extending from the Shape2D class. To define the Circle class, we need an additional **radius** property.

The following steps guide you through implementing the code for this part:

1. Create a new class named Circle that extends Shape2D

2. Declare a new double variable named radius

3. Define getRadius() and setRadius() methods for the radius property

4. Implement the abstract method calcArea() of the Circle class. The area of the circle is defined as $Area = \pi r^2$

5. Similarly, implement the abstract method calcPerimeter() of the Circle class. The perimeter of the Circle is defined as $Perimeter = 2 \pi r$

6. Override the toString() method to display the information of the Circle instance. You should return a string object with this format: "Circle: radius = r, center = (x, y)" (replace r with the actual radius; x,y with the center location.

7. Create a Circle object in the main() method to test its methods

Part 4: Implement the Rectangle class

The Rectangle class is another subclass extending from the Shape2D class. To define the Rectangle class, we need additional length and width properties.

The following steps guide you through implementing the code for this part:

1. Create a new class named Circle that extends Shape2D

2. Declare two new double variables named length and width

3. Define getLength(), getWidth(), setLength(), and setWidth() methods for the length and width properties

4. Implement the abstract method calcArea() of the Rectangle class. The area of the rectangle is defined as $Area = l \times w$

5. Similarly, implement the abstract method calcPerimeter() of the Rectangle class. The perimeter of the Rectangle is defined as $Perimeter = 2 (l + w)$

6. Override the toString() method to display the information of the Rectangle instance. You should return a string object with this format: “Rectangle: length = l, width = w, center = (x, y)” (replace l, w with the actual length and width; x,y with the center location.

7. Create a Rectangle object in the main() method to test its methods

Part 5: Polymorphism methods

In this part, we will practice the polymorphism principle in Java.

1. Implement a new method in the class:

```
/**
 * This method performs reporting on a Shape2D object
 * @param shape: the shape object
 * @return: None
 */
public static void report(Shape2D obj) {
    //TODO: add code below
}
```

You implement this method with the following steps:

- Print the class name of the Shape2D object. Hint: using instanceof
- Print the object information by calling toString() method
- Print the area and perimeter of the object by calling calcArea() and calcPerimeter() accordingly

2. Add the code of the **main**() method to test the report() method as follows:

- Create an instance of a Circle class with random attributes (color, center location, radius)
- Create another instance of the Rectangle class with random attributes (color, center location, length, and width)
- Test the report() method with the above instance of Circle
- Test the report() method with the above instance of Rectangle

Lab Assignment

Similar to part 4, you are required to implement a new class Eclipse that also extends from Shape2D. The Eclipse class is similar to the Circle. However, to form the Eclipse shape, you need to define the major axis and minor axis. Read more here on how to define Eclipse and the equations for area and perimeter.

Below is the requirements of the Eclipse class extending from Shape2D:

- Two variables a and b for major and minor axis
- Define the get() and set() methods for two new properties
- Implement calcArea(), calcPerimeter() for the Eclipse class - check the above link for equations
- Implement toString() method to return a string representation of this class with the format as “Eclipse: major axis = a, minor axis = b, center = (x, y)” (replace a, b with the actual length of major and minor axis; x,y with the center location).

Lab Result Submission:

You need to submit the results of the following sections to the D2L assignment item of Lab#2:

- Complete the required Lab Assignment above

- Compress the whole Eclipse source code of your Lab02 project in zip format using the Export function