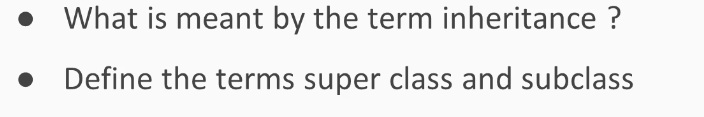
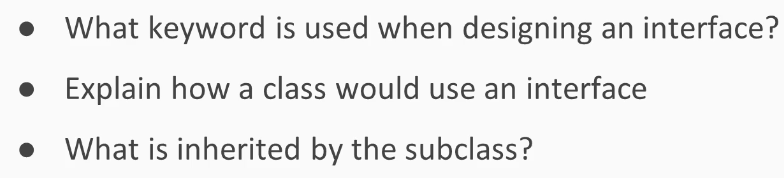
Hands-On Exercises - Java Inheritance

**Setup Instructions:**

1. Use the Word document created previously “AssignmentsDay3.docx” .
2. Answer Assignment 01.
3. Create a new Java Project.
4. Create your classes in the specified package.
5. Define your classes and interfaces as given below for each assignment.

Assignment 01



Assignment 02

Create classes and an interface as below:

TwoDShape

1. Create a public abstract class TwoDShape in a package “shape”.
2. Create 2 private instance variables of type int - xPos and yPos.
3. Create one parametrized constuctor to set the values of the above variables.
4. Implement getters and setters.
5. Create a public abstract method calculateArea( ) with a return type of double. The method body should be empty.
6. Create a public abstract method calculatePerimeter( ) with a return type of double. The method body should be empty.
7. Implement a toString() method for the class.

DrawShape

1. Create a public interface DrawShape in package “shape”.
2. Create 2 public static final variables of type int - ZOOM\_PLUS and ZOOM\_MINUS.
3. Set ZOOM\_PLUS = 1 and ZOOM\_MINUS= -1.
4. Create a public abstract method draw( ) with a return type of void.

Circle

1. Create a public class Circle in package “shape”.
2. Circle extends TwoDShape and implements DrawShape.
3. Create 1 private instance variable of type double called radius.
4. Create one parametrized constuctor passing in 2 ints - x and y.
5. Set the radius equal to 1.0 in the above constructor.
6. Compilation will fail as the code is written now.

*What do you need to do to get the code to compile?*

*Hint: Call the superclass constructor with the appropriate arguments.*

1. Create a parametrized constuctor passing in a double and 2 ints -

- double newValue

- int x

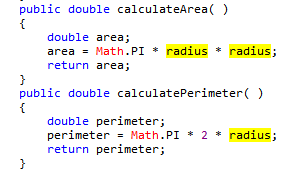
- int y

1. Set the instance variable radius equal to newValue in the above constructor.
2. Compilation will fail as the code is written now.

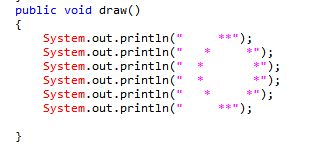
*What do you need to do to get the code to compile?*

*Hint: Call the superclass constructor with the appropriate arguments as in Step 6.*

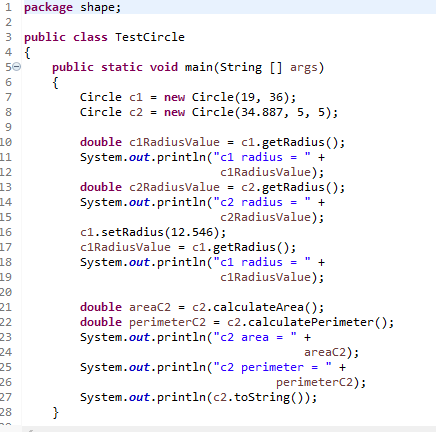
1. Implement a getter and setter for the variable “radius”
2. In the setter, name the parameter as “newRadiusValue” .
3. If newRadiusValue is negative, set radius equal to 1.
4. Else set radius equal to “newRadiusValue” .
5. Implement the following 2 methods:



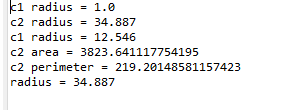
1. Implement a toString() method for the class.
2. Implement the draw() method as below:



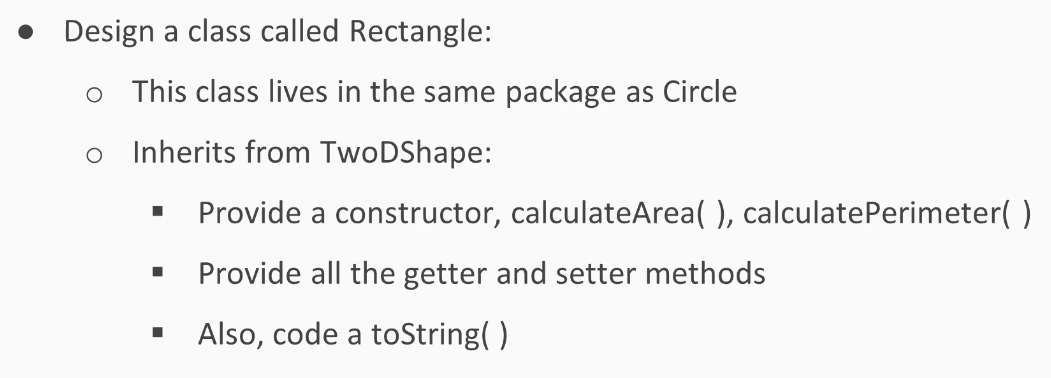
1. Lets’s create a class “TestCircle” in the same package to test the above:



Output:



Assignment 03



1. Create a public class Rectangle in package “shape”.
2. Rectangle extends TwoDShape and implements DrawShape.
3. Create 2 private instance variables of type double called length and width respectively.
4. Create a no-args constructor.
5. Call the superclass constructor with the values (0,0).
6. In the no-args constructor, set length as well as width equal to 1.0.
7. Create a parametrized constuctor passing in 2 doubles and 2 ints :-

- double length

- double width

- int x

- int y

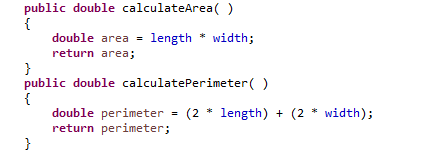
1. Call the superclass constructor with the appropriate arguments.
2. If the passed in values of either length or width is negative,*(use conditional OR)*

set the corresponding instance variables (length and width) equal to 1.0.

else

set the corresponding instance variables (length and width) equal to the values passed in.

1. Implement the following methods in Rectangle class.



1. Implement getters and setters.
2. In the setWidth() method,

if the width passed in is -ve, then set width equal to 1.0.

else

set width equal to whatever value is passed in.

1. In the setLength() method,

if the length passed in is -ve, then set length equal to 1.0.

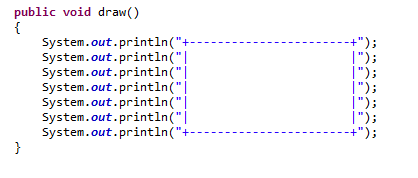
else

set length equal to whatever value is passed in.

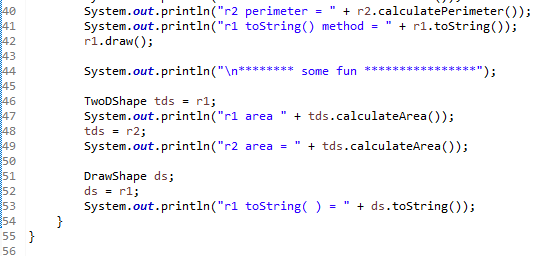
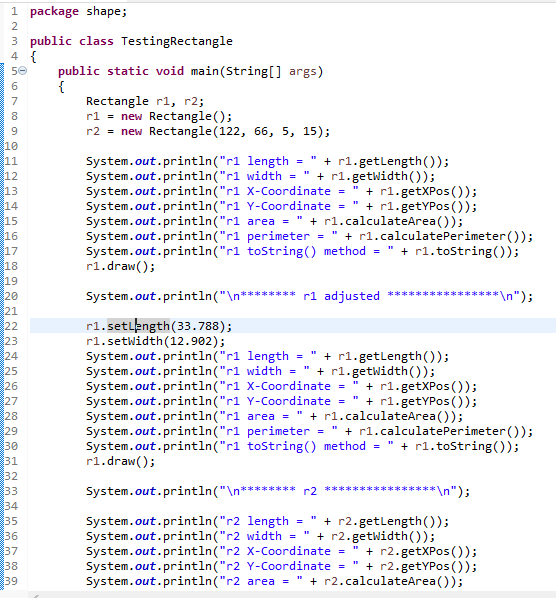
1. Create a method isSquare() with a return type of boolean.

Implement the logic to return true if length is equal to width else return false.

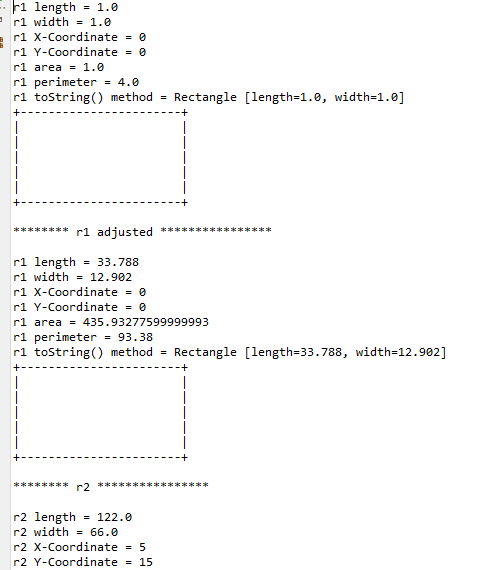
1. Create a toString() method.
2. Implement the following draw() method.

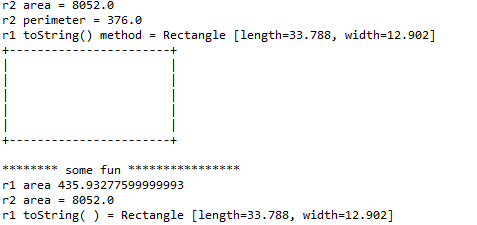


1. Lets’s create a class “TestingRectangle” in the same package to test the above:



Output:

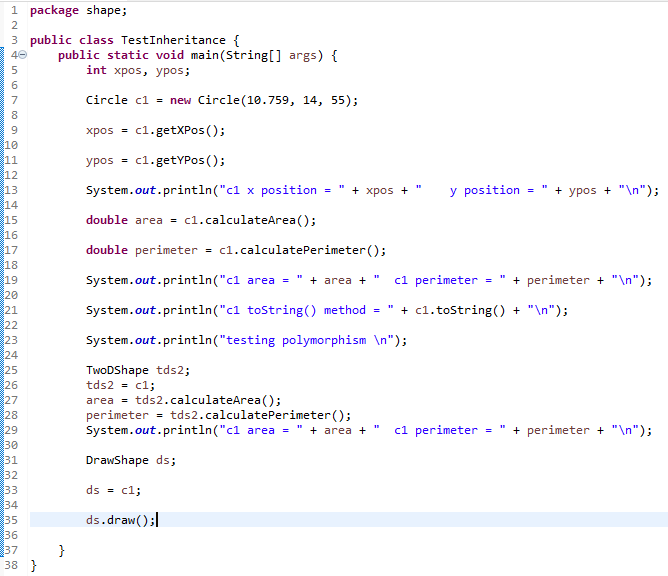




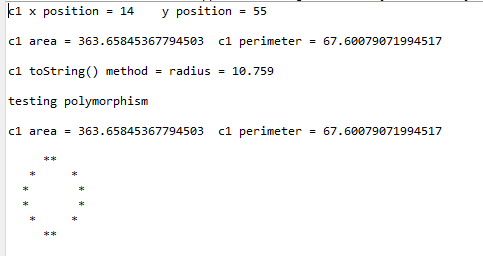
In Day 3, you created a class to test Static Polymorphism (method overloading).

Now let’s create a class to test Dynamic Polymorphism(method overriding).

1. Create the following test class:



Output:



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*