**Online Lab 4**

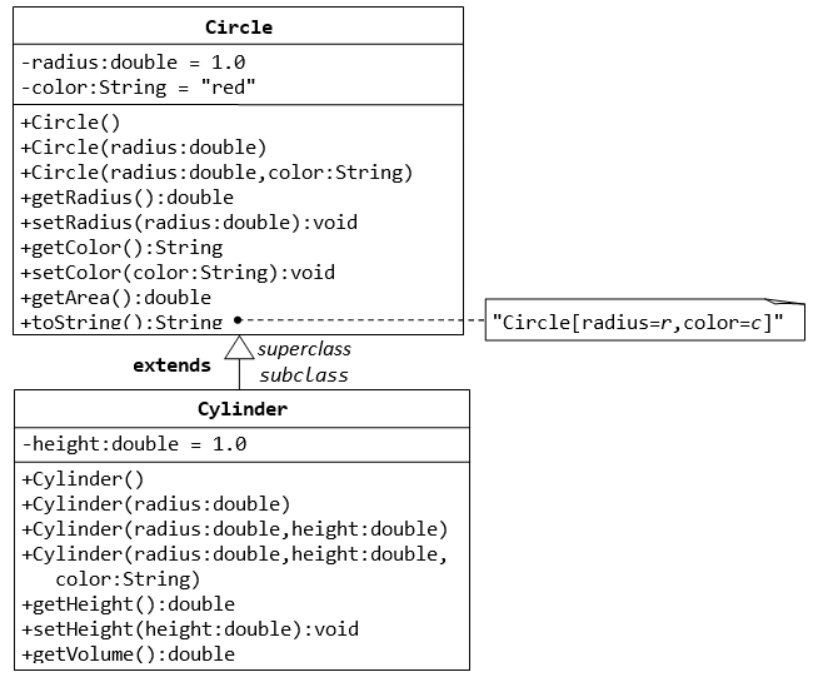
**Object Oriented Programming**

**Learning Objectives**

1. The students will be able to

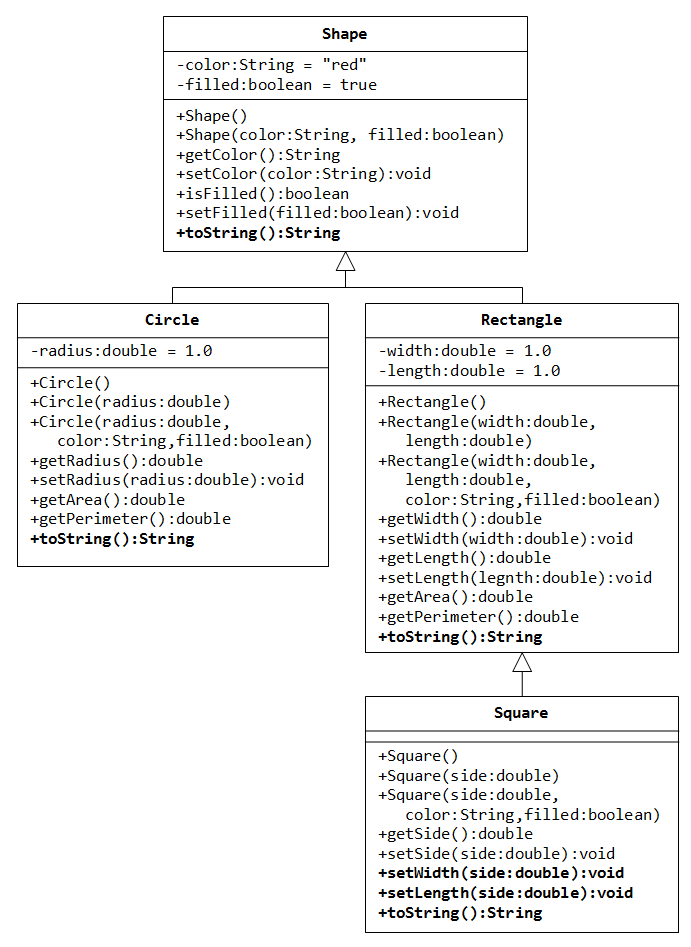
* Implement the inheritance relationship between two classes having an Is-A relationship
* Override and reuse base class behavior in the derived class
* Override the common object class methods such as toString.
* **Polymorphism**
* **Abstract Methods and Abstract Classes**

|  |
| --- |
| **Lab Walkthrough/Demo** |



* **Modify test class to show a parent class variable can refer to a child class object.**
* **Add a constructor to the Cylinder class that take object as parameter to emphasize the affectivity of the rule.**
* **Modify test class to demonstrate dynamic method dispatching.**

|  |
| --- |
| **Home Tasks/Lab Assignment** |



1. Write a **superclass** called **Shape** (as shown in the class diagram), which contains:

* Two instance variables color (String) and filled (boolean).
* Two constructors: **a no-arg (no-argument) constructor** that initializes the color to "green" and filled to true, and a two-argument constructor that initializes the color and filled to the parameters values.
* Getter and setter for all the instance variables. **By convention, the getter for a boolean variable filled is called isFilled()** (instead of getFilled() for all the other types).
* A toString() method that returns "A **<< filled/Not filled >>** Shape with **<< green >>** color.".
* Methods getArea() and getPerimeter() both reurn 0. // We can not write the implementation of getArea() and getPerimeter() until we know the specific shape.

1. There are two subclasses of Shape called Circle and Rectangle, as shown in the class diagram. **Ignore the Circle class.**

The **Rectangle** class contains:

* Two instance variables width (double) and length (double).
* Three constructors as shown. The **no-arg constructor** initializes the width and length to 1.0. All the constructors in **Rectangle** class should call the appropriate super-class (**shape**) constructors.
* Getter and setter for all the instance variables.
* Override methods getArea() -> length\*width and getPerimeter() -> 2\*length + 2\*width.
* Override the toString() method inherited, to return "A Rectangle with width=**<< xxx >>** and length= **<< zzz >>**, which is a subclass of **<< yyy >>**", where yyy is the output of the toString() method from the superclass.

1. Write a class called **Square**, as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

**Note** that a square is a rectangle with same width and height, therefore, whenever the length is changed, width should also be changed to the same value for maintaining the square geometry and vice versa.

* Provide the appropriate constructors (as shown in the class diagram). All the constructors in the **Square** class should call the appropriate super-class (Rectangle) constructors. Use super

public Square(double side) {

super(side, side); // Call superclass Rectangle(double, double)

}

* Override the toString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.
* Do you need to override the getArea() and getPerimeter()? Try them out.
* Override the setLength() and setWidth() to change both the width and length, so as to maintain the square geometry.

The Test Class:

1. Create 6 objects of rectangle and square (3 each) using each type of constructors in these classes using an array of Shape reference variables.
2. Observe the order of constructor calling according to messages printed by your constructors.
3. Display the state of each object via implicit call to toString. Use enhanced for to display the state of objects along with corresponding area and perimeter.

Observation Experiments:

1. Remove the no-argument constructor of shape class. Rerun your test class. Is there any error? Why?
2. Remove all the constructors from each class, except the two-argument constructor from the shape class and no-argument constructors from the Rectangle and Square classes. Is there any error? Why?

**Write your observations as comments at the end of your test class of your program.**