National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 212: Object Oriented Programming

Class: BSCS-6ABC

Lab 05: Inheritance

Date: March 27, 2017

Instructors:

Dr. Mian M. Hamayun & Dr. Anis ur Rahman

Learning Objectives

The learning objective of this lab is to understand and practice the concept of inheritance, a very powerful feature of OOP which helps in code reusability.

Activity #1.

What is the output of running the class ActivityOne? Explain the output.

Hint. Remember only super-class data members (instance variables and instance methods) are inherited by the sub-class, and these does not include constructors.

```
class A
{
	public A()
	{
		System.out.println( "A's no-arg constructor is invoked");
	}
}

class B extends A {}

public class ActivityOne
{
	public static void main( String[] args)
	{
		B b = new B();
	}
}
```

Activity #2.

The Java program compiles correctly. Show its output, and explain what is happening when an object of Class B is created?

```
class A
{
	public A()
	{
		System.out.println( "A's constructor is invoked");
	}
}
```

```
class B extends A
{
	public B(int t)
	{
		System.out.println( "B's constructor is invoked");
	}
}

public class ActivityTwo
{
	public static void main( String[] args)
	{
		B b = new B(3);
	}
}
```

Activity #3.

The program fails to compile. Identify the problem and propose a correction? Also explain the reason.

Hint. If any constructor does not explicitly call a super or this constructor as its first statement, a call to super() is automatically added.

```
class A
{
	public A( int x) {}
}

class B extends A
{
	public B() {}
}

public class ActivityThree
{
	public static void main( String[] args)
	{
		B b = new B();
	}
}
```

Activity #4.

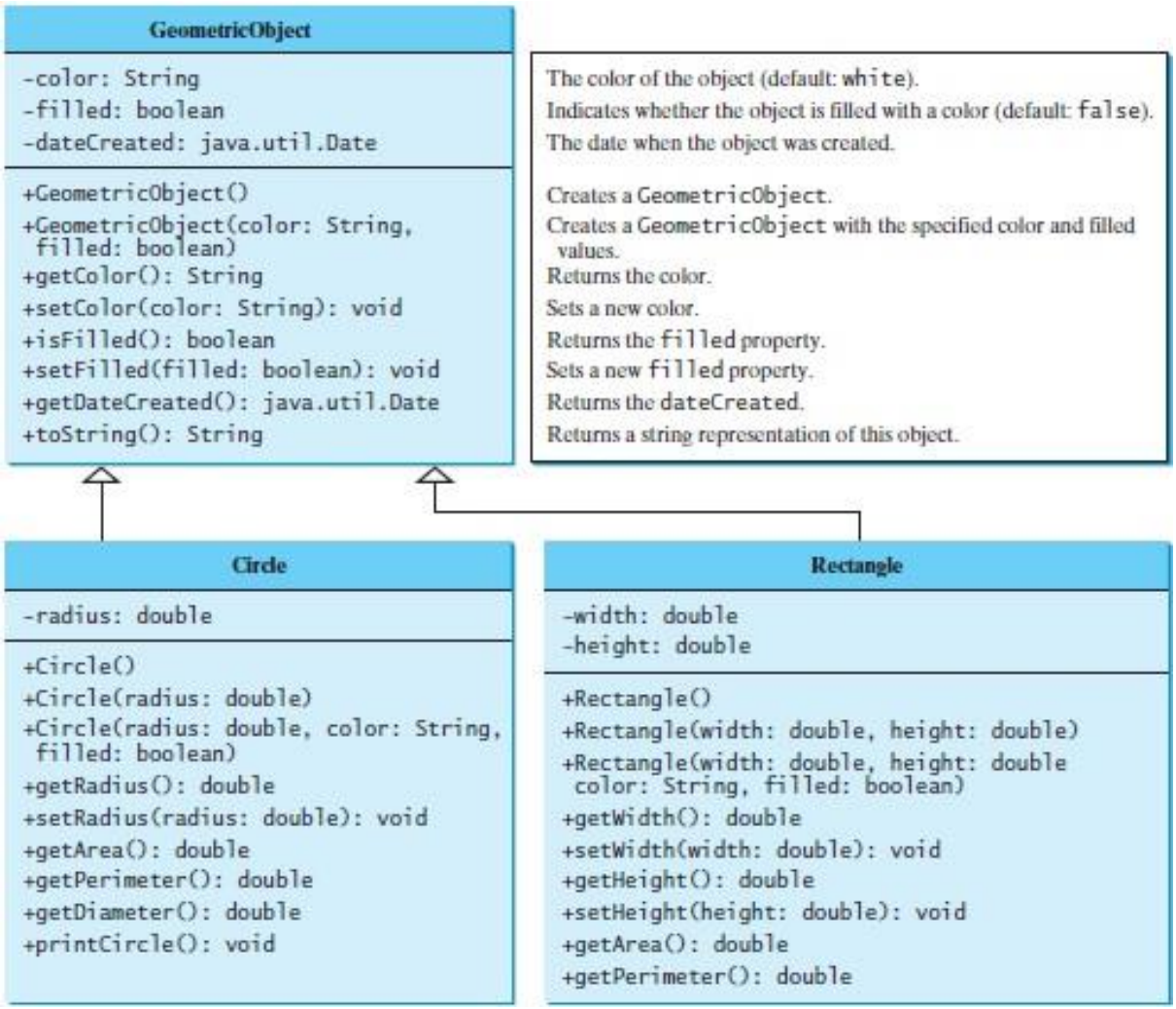
Identify and correct the problems in the following program.

```
class Circle
{
    private double radius;

    public Circle( double radius)
    {
        radius = radius;
    }

    public double getRadius()
    {
        return radius;
    }

    public double getArea()
    {
        return radius * radius * Math.PI;
    }
}

class B extends Circle
{
    private double length;

    B( double radius, double length)
    {
        Circle( radius);
        length = length;
    }

    /** Override getArea() */
    public double getArea()
    {
        return getArea() * length;
    }
}
```

```
public class ActivityFour
{
      public static void main( String[] args)
      {
         B b = new B( 5, 10);
         System.out.println( "Area = " + b.getArea());
      }
}
```

Task #1:

The following UML class diagram illustrates an inheritance relationship, wherein the classes `Circle` and `Rectangle` have been extended from the class `GeometricObject`.



You're required to implement the classes `GeometricObject` and `Rectangle.`

The `Rectangle` class contains:

- Two `double` data fields named `width` and `height` that specify the width and height of the rectangle. The default values are `1.0` for both `width` and `height`.
- A no-arg constructor that creates a default rectangle.
- A constructor that creates a rectangle with the specified `width` and `height`.
- A method named `getArea()` that returns the area of this rectangle.
- A method named `getPerimeter()` that returns the perimeter.
- A method named `toString()` that returns a string description for the rectangle.

The `toString()` method is implemented as follows:

```
return "Rectangle: width = " + width + " height = " + height;
```

Write a test program that prompts the user to enter `width` and `height` of the rectangle, a color, and a Boolean value to indicate whether the rectangle is filled. The program should create a `Rectangle` object and set the `color` and `filled` properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

Task #2:

Design a class named `Person` and its two subclasses named `Student` and `Employee`. Make `Faculty` and `Staff` subclasses of `Employee`.

A person has a name, address, phone number, and email address. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and date hired. A faculty member has office hours and a rank. A staff member has a title. Override the `toString`() method in each class to display the class name and the person's name.

Draw the UML diagram for the classes and implement them. Write a test program that creates a `Person`, `Student`, `Employee`, `Faculty`, and `Staff`, and invokes their `toString()` methods.

Hand in

Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single compressed/archived file named Lab_5_<Your CMS_ID. Your_NAME >.zip (without angle brackets) that contains ONLY the following files.

1) All completed java source files representing the work accomplished for this lab: ActivityOne.java; ActivityTwo.java; ActivityThree.java; ActivityFour.java; Task1.java, Task2.java. The files should contain author in the comments at the top.

2) A plain text file named **README.TXT** that includes a) author information at the beginning, b) a brief explanation of the lab, and c) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.

2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.

3. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the programming/other assignments.