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

Department of Computing

CS 212: Object Oriented Programming

Class: BESE-7AB

Lab 05: Inheritance

Date: March 31, 2017

Instructor:

Dr. Muhammad Muneeb Ullah

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 C1 (*you may provide the output in commented form at the top of the program's source file*)?

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

class B extends A
{
}

public class C1
{
   public static void main(String[] args)
   {
      B b = new B();
   }
}
```

Activity #2.

What problem arises in compiling the following program? How can you correct that?

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

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

public class C2
```

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

Activity #3.

Identify and correct the problems in the following classes.

```
public 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;
   }
}
```

Activity #4.

Show the output of the following program (*you may provide the output in commented form at the top of the program's source file*). Is the no-arg constructor of class Object invoked when `new A(3)` is invoked?

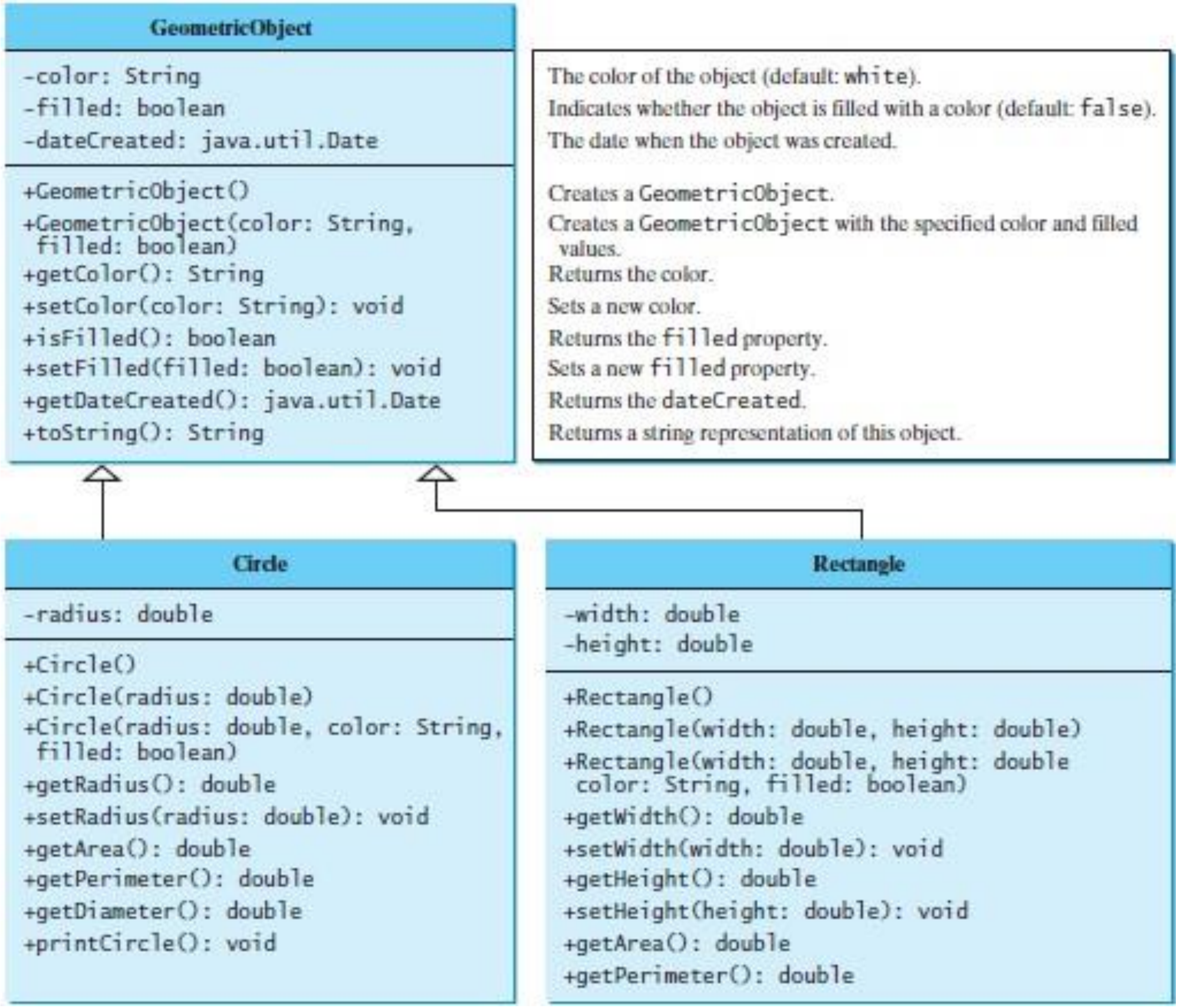
```
public class Test
{
   public static void main(String[] args)
   {
      A a = new A(3);
   }
}

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

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

Task #1:

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


GeometricObject
-color: String
-filled: boolean
-dateCreated: java.util.Date
+GeometricObject()
+GeometricObject(color: String, filled: boolean)
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String
The color of the object (default: white).
Indicates whether the object is filled with a color (default: false).
The date when the object was created.
Creates a GeometricObject.
Creates a GeometricObject with the specified color and filled values.
Returns the color.
Sets a new color.
Returns the filled property.
Sets a new filled property.
Returns the dateCreated.
Returns a string representation of this object.
Circle
-radius: double
+Circle()
+Circle(radius: double)
+Circle(radius: double, color: String, filled: boolean)
+getRadius(): double
+setRadius(radius: double): void
+getArea(): double
+getPerimeter(): double
+getDiameter(): double
+printCircle(): void
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+Rectangle(width: double, height: double color: String, filled: boolean)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void
+getArea(): double
+getPerimeter(): double

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.

Task #3:

Package-delivery services, such as FedEx®, DHL® and UPS®, offer several different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages.

Use class `Package` as the base class of the hierarchy. Then include classes `TwoDayPackage` and `OvernightPackage` that derive from `Package`. Base class `Package` should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. `Package's` constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. `Package` should provide a public member method `calculateCost` that returns a double indicating the cost associated with shipping the package. `Package's` `calculateCost` method should determine the cost by multiplying the weight by the cost per ounce.

Derived class TwoDayPackage should inherit the functionality of the base class Package, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. TwoDayPackage's constructor should receive a value to initialize this data member. TwoDayPackage should redefine member method calculateCost so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by the base class Package's calculateCost method.

Class OvernightPackage should inherit directly from class Package and contains an additional data member representing an additional fee per ounce charged for overnight-delivery service. OvernightPackage should redefine member method calculateCost so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost.

Write a test program that creates objects of each type of Package and tests the member method calculateCost.

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_2_<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; Task3.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.