# Department of Computing

# CS 212: Object Oriented Programming

# Class: BSCS 8A

# Lab 04: Fundamentals of OOP

# Date: 11th February 2019

# Instructor: Hirra Anwar

**Learning Objectives**

The learning objectives of this lab is to understand and practice the fundamentals of object-oriented programming, such as classes and objects, access specifiers, constructors, setters/getters, static members.

**Activity #1**

This first exercise shall lead you through all the basic concepts discussed in lectures related to classes and objects. Go through the complete activity and make the necessary modifications at each step.

A class called **circle** is designed which contains:

* Two private instance variables: radius (of the type double) and color (of the type String), with default value of 1.0 and "red", respectively.
* Two overloaded constructors - a default constructor with no argument, and a constructor which takes a double argument for radius.
* Two public methods: getRadius() and getArea(), which return the radius and area of this instance, respectively.

|  |
| --- |
| /\*  \* The Circle class models a circle with a radius and color.  \*/  public class Circle {  // Save as "Circle.java"  // private instance variable, not accessible from outside this class  private double radius;  private String color;    // The default constructor with no argument.  // It sets the radius and color to their default value.  public Circle() {  radius = 1.0;  color = "red";  }    // 2nd constructor with given radius, but color default  public Circle(double r) {  radius = r;  color = "red";  }    // A public method for retrieving the radius  public double getRadius() {  return radius;  }    // A public method for computing the area of circle  public double getArea() {  return radius\*radius\*Math.PI;  }  } |

Compile "Circle.java". Can you run the Circle class? Why or why not?

|  |
| --- |
| Answer: |

Let us write a test program called TestCircle (in another source file called TestCircle.java) which uses the Circle class, as follows:

|  |
| --- |
| public class **TestCircle** {  // Save as "TestCircle.java"  public static void main(String[] args) {  // Declare an instance of Circle class called c1.  // Construct the instance c1 by invoking the "default" constructor  // which sets its radius and color to their default value.  Circle c1 = new Circle();  // Invoke public methods on instance c1, via dot operator.  System.out.println("The circle has radius of "  + c1.getRadius() + " and area of " + c1.getArea());    // Declare an instance of class circle called c2.  // Construct the instance c2 by invoking the second constructor  // with the given radius and default color.  Circle c2 = new Circle(2.0);  // Invoke public methods on instance c2, via dot operator.  System.out.println("The circle has radius of "  + c2.getRadius() + " and area of " + c2.getArea());}} |

Now, run the TestCircle and study the results.

##### As a next step, execute the following

1. Constructor: Modify the class Circle to include a third constructor for constructing a Circle instance with two arguments - a double for radius and a String for color.  
   Modify the test program TestCircle to construct an instance of Circle using this constructor.

**Test Public/Private Access Modifiers**

1. **public vs. private:** In TestCircle, can you access the instance variable radius directly (e.g., System.out.println(c1.radius)); or assign a new value to radius (e.g., c1.radius=5.0)? Try it out and explain the error messages.

|  |
| --- |
|  |

1. Setter: Is there a need to change the values of radius and color of a Circle instance after it is constructed? If yes then in that case we cannot directly change the private data members and we are in need of public functions that can help us achieve that purpose. Add two public methods (or just one) called setters for changing the radius and color of a Circle instance as follows:

// Setter for instance variable radius

public void setRadius(double newRadius) {

radius = newRadius;

}

// Setter for instance variable color

public void setColor(String newColor) { ...... }

1. Getter:  If we want to directly access the values of the data members then we can do that with the help of getter functions. Add a getter for variable color and one for radius for retrieving the color and radius of this instance.
2. // Getter for instance variable color

public String getColor() { ...... }

Modify the test program to test the getter and setter methods.

**Static Variables/Methods Vs. Instance Variables/Methods**

In the class we have discussed about the static methods and instance methods. First of all create static method in your class definition. Invoke that method in the main function and check the differences between static and instance methods. Modify the Circle class created above. Add a static variable and static function NoOfObjects to calculate the total objects instantiated by the class.

**Activity #2.**

Carefully analyze the following code.

Can you invoke an instance method or reference an instance variable from a static method? Can you invoke a static method or reference a static variable from an instance method? What is wrong in the following code? Please correct it.

|  |
| --- |
| public class Example  {  public static void main(String[] args)  {  method1();  }  public void method1()  {  method2();  }  public static void method2()  {  System.out.println("What is radius " + c.getRadius());  }  Circle c = new Circle();  } |

**Task #1:**

Create a class called Employee that includes three instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Provide a constructor that initializes the three instance variables. Provide a *set* and a *get* method for each instance variable. If the monthly salary is not positive, it should be set to 0.

Write a test application named EmployeeTest that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10% raise and display each Employee’s yearly salary again.

**Task #3:**

Create a class called Date that includes three instance variables—a month (type int), a day (type int) and a year (type int). Provide a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes ( / ). Write a test application named DateTest that demonstrates class Date’s capabilities.

**Deliverables**

Compile a single word file with code and screenshot of output on LMS.