# Department of Computing

# CS 212: Object Oriented Programming

# Class: BESE-11AB

# Lab 07: Fundamentals of OOP

# Date: 21/04/2021

# 

# Instructor: Ms. Hania Aslam

**Learning Objectives**

The learning objectives of this lab is to understand and practice the fundamental concepts 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 previous lectures related to classes and objects. Please go through the complete activity and make the necessary modifications at each step wherever required.

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;  }  } |

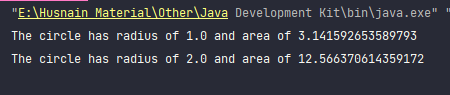
Compile "Circle.java". Can you run the Circle class? Why or why not?

|  |
| --- |
| Answer:  No, I cannot run the Circle class because we have not defined a main method for the circle class as it necessary to run the code. Hence, the compiler gives the error showing that the main method is not found. |

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

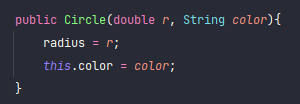
|  |
| --- |
| public class **TestCircle** {  // Save as "TestCircle.java"  public static void main(String[] args) {  // Declare an instance of Circle class called c1.  // Construct the instance c1 by invoking the "default" constructor  // which sets its radius and color to their default value.  Circle 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 observe the results.



##### **As a next step, incorporate the following changes into your code:**

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=9.0)? Try it out and explain the error messages.

|  |
| --- |
| No, we cannot access the instance variable ‘radius’ directly or assign a value to it because it is a ‘private’ variable meaning that it has private access and cannot be called directly. We can get its value or modify it through getters and setters. |

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  setter methods for changing the radius and color fields of a Circle instance .

public void setRadius(double *r*){  
 radius = *r*;  
}  
public void setColor(String *c*){  
 color = *c*;  
}

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 method for variable color and one for radius for retrieving the color and radius of this instance.

public double getRadius() {  
 return radius;  
}  
public String getColor(){  
 return color;  
}

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

Circle c4 = new Circle();  
c4.setRadius(3.0);  
c4.setColor("green");  
System.*out*.println("The circle has radius of "  
 + c4.getRadius() + " and area of " + c4.getArea() + " and color is " + c4.getColor());

**Static Variables Vs. Instance Variables**

In the class we have discussed about the static and instance variables. Modify the Circle class created above by adding a static variable NoOfObjects to calculate the total objects instantiated by the class. Create 5 objects of the circle class and display the resultant value of the static variable NoOfObjects.

public void getNo\_of\_obj(){  
 *no\_of\_obj* += 1;  
}

System.*out*.println("The number of objects created are " + Circle.*no\_of\_obj*);



**Activity #2.**

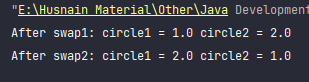
Analyze and briefly explain the output of the following program:

|  |
| --- |
| public class Test  {  public static void main(String[] args)  {  Circle circle1 = new Circle(1);  Circle circle2 = new Circle(2);  swap1(circle1, circle2);  System.out.println("After swap1: circle1 = " +  circle1.radius + " circle2 = " + circle2.radius);  swap2(circle1, circle2);  System.out.println("After swap2: circle1 = " +  circle1.radius + " circle2 = " + circle2.radius);  }  public static void swap1(Circle x, Circle y)  {  Circle temp = x;  x = y;  y = temp;  }  public static void swap2(Circle x, Circle y)  {  double temp = x.radius;  x.radius = y.radius;  y.radius = temp;  }  }  class Circle  {  double radius;  Circle(double newRadius)  {  radius = newRadius;  }  } |

You may like to have a look at: http://www.geeksforgeeks.org/swap-exchange-objects-java/

Answer:

When we call the method ‘swap1’ we took the objects of class Circle as parameters and swapped them but we didn’t get the required output. This is because the memory addresses of the objects are being swapped and not the values. But when we called the method ‘swap2’ we got the required output. This is because we used dot operator to get the actual value of the objects and hence we got the correct output.



**Task #1:**

Create a class SavingsAccount . Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit. Provide method calculateMonthlyInterest to calculate the monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12—this interest should be added to savingsBalance.

Provide a static method modifyInterestRate that sets the annualInterestRate to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with initial balance of $3200.00 and $4500.00, respectively. Set annualInterestRate to 5%, then calculate the monthly interest for each of 12 months and print the new balances for both savers. Next, set the annualInterestRate to 7%, calculate the next month’s interest and print the new balances for both savers.

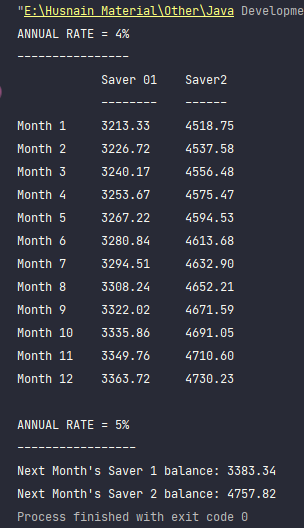
**Code:**

package com.company;  
  
*//Creating our own class SavingAccount*public class SavingsAccount {  
 *//Declaring Fields* static double *annualInterestRate*;  
 private double savingsBalance;  
  
 *//Method for setting balance* public void setSavingsBalance(double *balance*){  
 savingsBalance = *balance*;  
 }  
 *//Method for getting the balance* public double getSavingBalance(){  
 return savingsBalance;  
 }  
 *//Method for modifying the interest rate* public static void modifyInterestRate(double *annual\_rate*){  
 *annualInterestRate* = *annual\_rate*;  
 }  
 *//Method for calculating monthly interest* public double calculateMonthlyInterest(){  
 return (savingsBalance \* (*annualInterestRate* / 100)) / 12;  
 }  
 *//Method for calculating total balance* public double calculateTotalBalance(double *monthly\_interest*, double *balance*){  
 return *monthly\_interest* + *balance*;  
 }  
  
}

**Test Program:**

package com.company;  
  
*//Creating the test class*public class SavingsAccountTest {  
  
 *//Main method* public static void main(String[] *args*) {  
 *//Creating two objects of SavingAccounts class* SavingsAccount saver1 = new SavingsAccount();  
 SavingsAccount saver2 = new SavingsAccount();  
 *//Setting the balance for both objects* saver1.setSavingsBalance(3200);  
 saver2.setSavingsBalance(4500);  
 System.*out*.println("ANNUAL RATE = 4%");  
 System.*out*.println("----------------");  
 *//Taking annual rate 4%* SavingsAccount.*modifyInterestRate*(5);  
 System.*out*.println("\t\t\tSaver 01\tSaver2");  
 System.*out*.println("\t\t\t--------\t------");  
 *//Loop for printing our desired output* for (int i = 0; i < 12; i++){  
 saver1.setSavingsBalance(saver1.calculateTotalBalance(saver1.calculateMonthlyInterest(), saver1.getSavingBalance()));  
 saver2.setSavingsBalance(saver2.calculateTotalBalance(saver2.calculateMonthlyInterest(), saver2.getSavingBalance()));  
 *//Printing saving balance for all 12 months for both objects* if (i < 9) {  
 System.*out*.printf("Month %d\t\t%.2f\t\t%.2f\n", (i + 1), saver1.getSavingBalance(), saver2.getSavingBalance());  
 }else{  
 System.*out*.printf("Month %d\t%.2f\t\t%.2f\n", (i + 1), saver1.getSavingBalance(), saver2.getSavingBalance());  
 }  
 }  
 *//Changing the annual rate to 5%* System.*out*.println("\nANNUAL RATE = 5%");  
 System.*out*.println("-----------------");  
 SavingsAccount.*modifyInterestRate*(7);  
 *//Printing next month's balance for both objects* System.*out*.printf("Next Month's Saver 1 balance: %.2f\n", saver1.calculateTotalBalance(saver1.calculateMonthlyInterest(), saver1.getSavingBalance()));  
 System.*out*.printf("Next Month's Saver 2 balance: %.2f",saver2.calculateTotalBalance(saver2.calculateMonthlyInterest(), saver2.getSavingBalance()));  
  
 }  
}

**Output Screenshot:**



**Task #2:**

As we have discussed earlier, in Java, arrays are capable of storing objects as well. Create a class Student that contains two private fields: student\_id & marks(0-100). Provide appropriate constructors/methods for this class. Next, create an array named studentArray that is capable of storing record for seven students. Initialize all array objects with values of your choice( Remember, Student is a reference type i.e: each array element stores a reference to the Student object ). Finally, iterate over all elements of the studentArray in order to calculate and display the average marks.

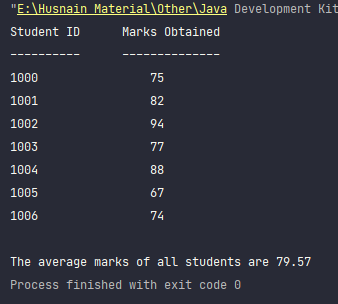
**Code:**

package com.company;  
  
*//Creating our own class Student*public class Student {  
 *//Declaring fields* private int id;  
 private int marks;  
  
 *//Method for setting student Id* public void setId(int *number*) {  
 id = *number*;  
 }  
 *//Method for getting student Id* public int getId() {  
 return id;  
 }  
 *//Method for setting student marks* public void setMarks(int *score*) {  
 marks = *score*;  
 }  
 *//Method for getting student marks* public int getMarks() {  
 return marks;  
 }  
}

**Test Program:**

package com.company;  
  
*//Creating Test class*public class StudentTest {  
  
 *//Main method* public static void main(String[] *args*) {  
  
 *//Creating 7 objects of Student class i.e., 7 students and setting their information* Student ali = new Student();  
 ali.setId(1000);  
 ali.setMarks(75);  
  
 Student husnain = new Student();  
 husnain.setId(1001);  
 husnain.setMarks(82);  
  
 Student abdullah = new Student();  
 abdullah.setId(1002);  
 abdullah.setMarks(94);  
  
 Student haider = new Student();  
 haider.setId(1003);  
 haider.setMarks(77);  
  
 Student muzammil = new Student();  
 muzammil.setId(1004);  
 muzammil.setMarks(88);  
  
 Student ammar = new Student();  
 ammar.setId(1005);  
 ammar.setMarks(67);  
  
 Student kamran = new Student();  
 kamran.setId(1006);  
 kamran.setMarks(74);  
  
 *//Creating an array of objects* Student [] studentArray = {ali, husnain, abdullah, haider, muzammil, ammar, kamran};  
  
 System.*out*.println("Student ID\t\tMarks Obtained");  
 System.*out*.println("----------\t\t--------------");  
 *//Loop for printing each students information and also calculating total marks* int total\_marks = 0;  
 for (Student i : studentArray){  
 System.*out*.printf("%d\t\t\t\t%d\n", i.getId(), i.getMarks());  
 total\_marks += i.getMarks();  
 }  
 *//Printing the average of all marks of students* System.*out*.println();  
 System.*out*.printf("The average marks of all students are %.2f", total\_marks / 7.0);  
  
 }  
}

**Output Screenshot:**

****

**Hand in**

Hand in the source code from this lab at the appropriate location on the LMS system.

1. All completed java source code neatly placed into labeled text boxes representing the work accomplished for this lab. The code should contain necessary comments.

**To Receive Credit**

1. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
2. The lab time is not intended as free time for working on your programming/other assignments.