# Department of Computing

# CS 212: Object Oriented Programming

# Class: BSCS-8AB

# Lab 04: Classes & Objects

# Date: March 4, 2019

# Instructor: Hirra Anwar

**Learning Objectives**

The learning objective of this lab is to understand and practice the concepts of constructors, constructor overloading, get and set methods, this reference and overloaded methods.

**Activity #1**

Complete the following Time Class and provide the missing constructor definitions. For each constructor, you need to use the **setTime** method for setting-up the parameters that are passed to it. In this way, you can ensure that all parameters are valid. Also, see what happens if you pass an invalid parameter to the constructor, while creating an object of Time class (as done for t4 object).

|  |
| --- |
| class Time  {  private int hour; // 0 - 23  private int minute; // 0 - 59  private int second; // 0 - 59  // TODO: Add constructors definitions here  // set a new time value using universal time; throw an  // exception if the hour, minute or second is invalid  public void setTime( int h, int m, int s )  {  // validate hour, minute and second  if ( ( h >= 0 && h < 24 ) && ( m >= 0 && m < 60 ) && ( s >= 0 && s < 60 ) )  {  hour = h;  minute = m;  second = s;  } // end if  else  System.out.println("hour, minute and/or second was out of range" );  } // end method setTime  // convert to String in universal-time format (HH:MM:SS)  public String toUniversalString()  {  return String.format( "%02d:%02d:%02d", hour, minute, second );  } // end method toUniversalString  // convert to String in standard-time format (H:MM:SS AM or PM)  public String toString()  {  return String.format( "%d:%02d:%02d %s",  ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ),  minute, second, ( hour < 12 ? "AM" : "PM" ) );  } // end method toString  } // end class Time  // This class tests the Time Class by creating different objects.  public class TimeTest{  public static void main(String [] args){  Time t0 = new Time(); // Set Time to 00:00:00  Time t1 = new Time(11); // Set Time to 11:00:00  Time t2 = new Time(12, 40); // Set Time to 12:40:00  Time t3 = new Time(23, 40, 55); // Set Time to 23:40:55  Time t4 = new Time(23, 40, 65); // Set Time to 23:40:65  // Print All Times in Universal Format  System.out.println(t0.toUniversalString());  System.out.println(t1.toUniversalString());  System.out.println(t2.toUniversalString());  System.out.println(t3.toUniversalString());  // Print All Times in Standard Format  System.out.println(t0);  System.out.println(t1);  System.out.println(t2);  System.out.println(t3);  }  } |

**Activity #2**

What will be the output of the following program? Execute and confirm your understanding.

|  |
| --- |
| // Demonstrate final constant.  class Test {  int x=1;  final static y=0;  void test() {  System.out.println("No parameters");  }    public class OverloadTest {  public static void main(String args[]) {  Test ol = new Test();  double result;  //TODO: Access x and y in main function.  //TODO: Try making an object as final and see what access permissions does it have    System.out.println("Result of ol.test(123.2): " + result);  }  } |

**Activity #3**

There are some issues with the following definition of the “Overload” Class. Find them out?

|  |
| --- |
| // Demonstrate method overloading.  class Overload {  void Overload(){  System.out.println("Constructor Called");  }  void test() {  System.out.println("No parameters");  }  // Overload test for one integer parameter.  void test(int a) {  System.out.println("a: " + a);  }  // Overload test for one integer parameter.  int test(int a) {  System.out.println("a: " + a);  return 0;  }  // Overload test for two integer parameters.  void test(int a, int b) {  System.out.println("a and b: " + a + " " + b);  }  // Overload test for two integer parameters.  void test(int b, int a) {  System.out.println("a and b: " + a + " " + b);  }  // Overload test for a double parameter  double test(double a) {  System.out.println("double a: " + a);  return a\*a;  }  } |

**Task #1:**

Create 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 balances of $2000.00 and $3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest for each of 12 months and print the new balances for both savers. Next, set the annualInterestRate to 5%, calculate the next month’s interest and print the new balances for both savers.

**Task #2:**

**An Array of a Primitive Type as a Field**

|  |
| --- |
| By the usage of arrays, they are primarily like variables. This means that an array can be declared as a field in a class. To create a field as an array, you can declare it like a normal array in the body of the class. Here is an example:  public class CoordinateSystem {  private int[] Points;  } |

Like any field, when an array has been declared as a member variable, it is made available to all the other members of the same class. You can use this feature to initialize the array in one method and let other methods use the initialized variable. This also means that you do not have to pass the array as argument nor do you have to explicitly return it from a method.

you can allocate memory for an array when declaring it also. You can also allocate memory for an array field in a constructor of the class.

For this task, create a Points class having an array as a data member. The size of the array should be 10. Construct suitable constructors for the class. Create methods of Display(), AddPoints(),which will add two objects of the class by implicitly adding the data member array e.g. A[0] will be added to B[0] and A[1] will be added to B[1] and so on. Similarly display Point object of the class. The arguments for the methods can be added by you based on the requirement.

**Deliverables**

Compile a single word file with all codes and their screenshots and upload on the link provided on LMS.