**Centennial College**

**COMP 228: Java Programming**

**LAB #1 – Java Class**

**FULL MARKS: 100**

**Student:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Due Date: Week 3

References: Learning materials for week 1, 2, textbook, and other references (if any)

Purpose: The purpose of this Lab assignment is to:

* Practice the use of Java classes, Java methods, and other concepts taught.

This material provides the necessary information you need to complete the exercises.

Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students.

**THERE ARE 2 DELIVERABLES FOR THIS LAB AS EXPLAINED BELOW. PLEASE NOTE: IF YOU SUBMIT ONE OF THEM AND MISS THE OTHER, THE SUBMISSION WOULD BE CONSIDERED AS INCOMPLETE. IN THAT CASE, YOU WOULD BE GRADED ON THE REMAINING 50% OF THR TOTAL MARKS.**

**YOU NEED TO SUBMIT THE FOLLOWING 2 DOCUMENTS IN THE DROPBOX TITLED LAB1:**

1. **THE FIRST DELIVERABLE:** A WORD DOCUMENT. USE THIS DOCUMENT AND ADD SCREEN SHOTS OF THE RUNNING STATE OF EACH EXERCISE (If there are more than 1 exercise). DO NOT DELETE THE QUESTIONS. THE SCREEN SHOTS SHOULD FOLLOW EACH QUESTION AND COVER ALL THE ASPECTS/FUNCTIONALITIES OF EACH EXERCISE. AFTER THE SCREEN SHOTS PLEASE COPY THE CODE FROM THE CODE WINDOW AND PASTE THE COMPLETE CODE. DO NOT GIVE ME SCREEN SHOTS OF THE CODE. DO NOT ZIP THIS FILE AND KEEP IT SEPARATE FROM YOUR ZIPPED PROGAM FILE.
2. **THE SECOND DELIVERABLE:** THE ZIPPED PROJECT FOLDER THAT CONTAINS ALL THE EXERCISES (if the lab contains more than 1 exercise) INTO THE SAME DROP BOX.

You must name your Eclipse/IntelliJ project according to the following rule:

**YourFullName\_COMP228Labnumber**

Example: **JohSmith\_COMP228Lab1**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228Labnumber.zip**

Example: **JohSmith\_COMP228Lab1.zip**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

**Exercise 1:**

Write a Java application that creates a **Java console application** to keep records of singers and displays their records. Follow the following instructions to develop the application:

Create a class named Singers with the following specifications:

* 5 instance variables that would store the following singer data (Use recommended variable naming rules and conventions. Use also appropriate data type for each instance variable):
  + Singer’s id
  + Singer’s name
  + Singer’s address
  + Singer’s Date of birth
  + Singer’s Number of albums published.
* Several constructors that would allow you to construct Singer object with no arguments, 1 argument, 2 arguments, 3 arguments, 4 arguments, and 5 arguments.
* Create Setters and getters for all the instance variables of class Singer. Make sure to have 5 setters that would allow you to set the values of each instance variable of a singer object individually. Also create one setter that would allow you to set all the values of the instance variables of a singer object at once. Create 5 getters that would allow you to get the current values of each instance variable of the Singer object individually.
* Create a driver class that would create 1 Singer (singer1) object with the help of the no argument constructor. Display the default values of the instance variables of this object singer1.
* Set the values of each instance variable one by one of singer1 with the help of individual setters. Display the value each time you set 1 value.
* Create another singer object singer2 with the 5-argument constructor.
* Display the all the values of the object singer2 with the help of the overridden toString() method. Please note: you should override the toString() method in the appropriate place in your code in order to accomplish the desired display.

1. Put all the screenshots in the space below. Use your judgement to provide enough screen shots to fairly represent your application. Label and number each screen shot in a very ORGANIZED way.

**Screenshot 1 : Creating all the instance variables and starting to create the constructors**

A screenshot of a computer

Description automatically generated

**Screenshot 2 and 3 : all the constructors**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Screenshot 4: setters**

A screenshot of a computer

Description automatically generated

**Screenshot 5: all the getter and the last setter with the five values**

A screenshot of a computer

Description automatically generated

**Screenshot 6: toString method**

A screenshot of a computer

Description automatically generated

**Screenshot 7: main class**

A screenshot of a computer

Description automatically generated

**Screenshot 8 and 9: output**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **COPY** the entire code that you developed (All the classes if you have more than 1 class) from the code window and **PASTE** below. Number the classes and label them. **DO NOT** provide screen shots of the code.

**Main class**

**package** exercise1;

**import** java.time.LocalDate;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// no argument object

Singers singer1 = **new** Singers();

System.***out***.print("Singer 1 ");

System.***out***.print(singer1);

// setting values for all the individual instance variables

System.***out***.print("\nSinger1");

singer1.setId(301275893);

System.***out***.print(singer1);

System.***out***.print("\nSinger1");

singer1.setName("Vansh");

System.***out***.print(singer1);

System.***out***.print("\nSinger1");

singer1.setAddress("Scarborough");

System.***out***.print(singer1);

System.***out***.print("\nSinger1");

singer1.setDateOfBirth(LocalDate.*of*(2003, 3, 10));

System.***out***.print(singer1);

System.***out***.print("\nSinger1");

singer1.setnoOfAlbums(10);

System.***out***.print(singer1);

// five argument object

Singers singer2 = **new** Singers(22222,"Vansh","12 blue eagle trail",LocalDate.*of*(2023, 05, 25), 22);

System.***out***.print("\nSinger 2");

System.***out***.print(singer2);

}

}

**Singers class**

**package** exercise1;

**import** java.time.LocalDate;

**public** **class** Singers {

**private** **int** id ;

**private** String name ;

**private** String address;

**private** LocalDate dateOfBirth;

**private** **int** noOfAlbums;

// constructors

// constructor with no argument

**public** Singers() {

}

// constructor with one argument

**public** Singers(**int** id ) {

**this**.id = id ;

}

// constructor with two arguments

**public** Singers(**int** id , String name) {

**this**.id = id ;

**this**.name= name ;

}

// constructor with three arguments

**public** Singers(**int** id , String name , String address) {

**this**.id = id ;

**this**.name = name ;

**this**.address = address;

}

// constructor with four arguments

**public** Singers(**int** id, String name, String address,LocalDate dateOfBirth ) {

**this**.id = id ;

**this**.name =name ;

**this**.address = address;

**this**.dateOfBirth = dateOfBirth;

}

// constructor with five arguments

**public** Singers (**int** id, String name, String address,LocalDate dateOfBirt,**int** noOfAlbums ) {

**this**.id = id ;

**this**.name =name ;

**this**.address = address;

**this**.dateOfBirth = dateOfBirth;

**this**.noOfAlbums = noOfAlbums;

}

// setters

// creating all the individual setters

**public** **void** setId(**int** id) {

**this**.id=id;

}

**public** **void** setName(String name) {

**this**.name=name ;

}

**public** **void** setAddress(String address) {

**this**.address=address;

}

**public** **void** setDateOfBirth(LocalDate DateOfBirth) {

**this**.dateOfBirth=dateOfBirth;

}

**public** **void** setnoOfAlbums(**int** noOfAlbums) {

**this**.noOfAlbums=noOfAlbums;

}

// setter for setting the values of the instance variables at once

**public** **void** setAll(**int** id, String name, String address, LocalDate dateOfBirth, **int** noOfAlbums) {

**this**.id = id;

**this**.name = name;

**this**.address = address;

**this**.dateOfBirth = dateOfBirth;

**this**.noOfAlbums = noOfAlbums;

}

// getters

**public** **int** id () {

**return** id;

}

**public** String name() {

**return** name;

}

**public** String address() {

**return** address;

}

**public** LocalDate DateOfBirth() {

**return** dateOfBirth;

}

**public** **int** noOfAlbums() {

**return** noOfAlbums;

}

**public** String toString()

{

**return** ( " \nid: " + id + "\nname: " + name + "\naddress: " + address+ "\nDate Of Birth: " +

dateOfBirth + "\nNumber Of Albums: " + noOfAlbums + "\n");

}

}

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of classes (instance variable declarations, constructors, getter and setter methods, etc.) | 45% |
| Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results) | 45% |
| **Organization of the submitted assignment** |  |
| Use of comments in the code, proper  indents etc, that enhance the readability. | 10% |
| **Total** | 100% |