# Introducing Classes and Objects - II

## Lab Objective

Familiarize students with the implementation of classes and instantiation of objects accordingly.

## Lab Outcomes

After completing this lab successfully, students will be able to:

1. **Understand** and **Apply** the concepts of classes and objects.
2. **Write** the definition of a class as per the specification and **create** objects accordingly using constructors.
3. **Write** various instance methods performing different actions on the objects of a class.

## Psychomotor Learning Levels

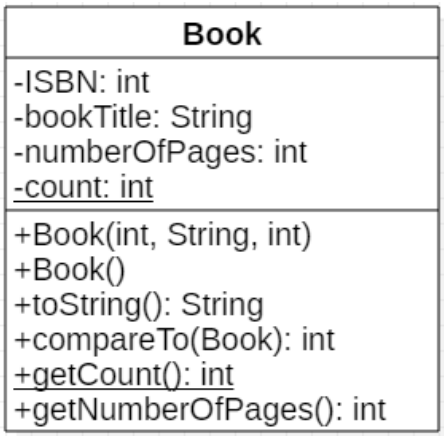
This lab involves activities that encompass the following learning levels in psychomotor domain.

| Level | Category | Meaning | Keywords |
|---|---|---|---|
| P1 | Imitation | Copy action of another; observe and replicate. | Relate, Repeat, Choose, Copy, Follow, Show, Identify, Isolate. |
| P2 | Manipulation | Reproduce activity from instruction or memory | Copy, response, trace, Show, Start, Perform, Execute, Recreate. |

## Lab Activities

### A. Static Variables in a class

- We have only seen instance variables so far. Consider the following class diagram of Book class.




- Here, ISBN, bookTitle and numberOfPages are instance variables. Each object has their own copy of instance variables.
- On the other hand, **objects share only one copy of a static variable.** Like static methods, static variables belong to a class, not to any objects.

- A static variable such as count in the Book class can be used to count the total number of Book type objects. **Each time, a Book type object is created, we need to increment the value of the count variable within the constructor.**
- A method that returns the value of a static variable is called a static method. In this Book class, getCount() is a static method that returns the value of count.
- **Your job is to implement the Book class as per the above-mentioned class diagram.**

### B. Demonstrating Calendar class for manipulating dates

```
public class DateDemo {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Calendar date = Calendar.getInstance();
		System.out.println("The date is " + date.getTime() + " " +
					date.getTimeInMillis());
		//changing the date
		date.set(Calendar.YEAR, 2020);
		date.set(Calendar.MONTH, 0); //January starts from 0
		date.set(Calendar.DAY_OF_MONTH, 1);
		date.set(Calendar.HOUR_OF_DAY, 0);
		date.set(Calendar.MINUTE, 0);
		date.set(Calendar.SECOND, 1);
		System.out.println("The changed date is " + date.getTime() + " " +
					date.getTimeInMillis());
		System.out.println(date.toString());
		System.out.println(System.currentTimeMillis());
	}
}
```

### C. Demonstrating Point2D class for manipulating points in a two-dimensional space

| javafx.geometry.Point2D | |
|---|---|
| +Point2D(x: double, y: double) | Constructs a Point2D object with the specified *x*- and *y*-coordinates. |
| +distance(x: double, y: double): double | Returns the distance between this point and the specified point (*x*, *y*). |
| +distance(p: Point2D): double | Returns the distance between this point and the specified point p. |
| +getX(): double | Returns the *x*-coordinate from this point. |
| +getY(): double | Returns the *y*-coordinate from this point. |
| +toString(): String | Returns a string representation for the point. |

```
import java.util.Scanner;
import javafx.geometry.Point2D;

public class TestPoint2D {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter point1's x-, y-coordinates: ");
    double x1 = input.nextDouble();
    double y1 = input.nextDouble();
    System.out.print("Enter point2's x-, y-coordinates: ");
    double x2 = input.nextDouble();
    double y2 = input.nextDouble();

    Point2D p1 = new Point2D(x1, y1);
    Point2D p2 = new Point2D(x2, y2);
    System.out.println("p1 is " + p1.toString());
    System.out.println("p2 is " + p2.toString());
    System.out.println("The distance between p1 and p2 is " +
      p1.distance(p2));
  }
}
```

## D. Immutable Class and Immutable Objects

- **<u>The contents of immutable objects cannot be changed.</u>**
- The corresponding class needs to be immutable too.
- **<u>A class is immutable if it satisfies the following:</u>**
  - All data fields must be private.
  - There can't be any mutator methods (setters) for data fields.
  - No accessor methods can return a reference to a data field that is mutable.
- **<u>Is the following class immutable?</u>**

```
public class Student {
      private int id;
      private String name;
      private java.util.Date dateCreated;
      public Student(int ssn, String newName) {
            id = ssn;
            name = newName;
            dateCreated = new java.util.Date();
      }
      public int getId() {
            return id;
      }
      public String getName() {
            return name;
      }
      public java.util.Date getDateCreated() {
            return dateCreated;
      }
}
```

*This lab sheet must be preserved till the end of the semester.*

| **Student ID** | | **Solved during the LAB** | |
|---|---|---|---|
| **Student Name** | | | |
| **Section** | | **Solved after the LAB** | |
| **Date** | | | |

## Lab 05: Lab Problems (Mid I Problems)

**Lab05_Problem01:**

**Write** a Java program that prompts user to input the radius, *r*, of a circle. If the radius is positive ($r > 0$), you must display the area and the perimeter of the circle. If *r* is equal to 0 then you must print "It is a point". Otherwise ($r < 0$), print "Invalid radius".
You must take input and print output of the program using dialog boxes (JOptionPane class). Use Math.PI for the value of π. The formulas for calculating area and perimeter of the circle is given below.

area = $\pi r^2$ , perimeter = $2\pi r$

**Lab05_Problem02:**

Consider the following dataset that shows marks of a set of students in different subjects in EWU admission test.

| Name | Subjects | | | | |
|---|---|---|---|---|---|
| | English | Mathematics | Physics | Chemistry | General Knowledge |
| Alice | 15 | 10 | 12 | 8 | 14 |
| Bob | 8 | 18 | 12 | 12 | 20 |
| Charlie | 8 | 15 | 20 | 17 | 9 |
| Danny | 12 | 14 | 11 | 10 | 10 |
| Elton | 9 | 7 | 19 | 15 | 10 |

A student will pass the admission test if he/she satisfies one of the following criteria:

i) Obtained at least 10 in all subjects
ii) Obtained at least 50 in Mathematics, Physics and Chemistry
iii) Obtained at least 70 in all subjects

**Write** a program in Java that uses the above-mentioned dataset and determines who will pass or fail in the admission test. You must use a method named *isPassed* that takes data of a particular student and returns *true* if the student passed the test and *false* otherwise. Finally, print the result accordingly. A sample output is given below.

```
Admission Test Result
Alice   - Failed
Bob     - Passed
Charlie - Passed
Danny   - Passed
Elton   - Failed
```

**Lab05_Problem03:**

**Define** a class *Point* that has two private instance variables: x and y, representing the value of the x-coordinate and y-coordinate, respectively, in a two-dimensional coordinate system.

The class *Point* has the following instance methods.

- *findDistance(Point p)*: the method takes another point *p* as a parameter and returns the Euclidean distance between the caller and callee points.
- *toString()*: the method returns a String containing the values of x-coordinate and y-coordinate of a point. As an example, for a point *p(3, 4)*, the method may return this String - `(3, 4)`

Use appropriate datatype. Your class definition must also include constructors, setters and getters. The formula for finding Euclidean distance between two points *p1(x1, y1)* and *p2(x2, y2)* is given below.

$$\text{Euclidean distance} = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

You may use *Math.pow(a,b)* or *Math.sqrt(n)* to calculate this distance.

Now, **write** a Java program that creates an array of *Point* type objects, as specified in question no. 5. The size of the array is 4. User may input the values of x and y-coordinates of those points using Scanner class.

Your program should also have the following static method.

*furthestPoint(Point[] points, Point p)*

The method takes two parameters – the array of Points and a Point type object.

The method must return the point within the array which is furthest (maximum distance) from the given Point type object. You must use *findDistance()* instance method as defined in question 5 while finding the distance between two points.

Your program must print the furthest point's coordinates. As an example, you may follow the format below.

```
Point (10, 10) is the furthest point from Point (3, 4)
```