Lab04: Classes and Objects

Designing and implementing Java programs that deal with:

1. **A class Definition**
2. **Constroctor**
3. **Class members**
4. **Class methods**
5. **The *new* Operator to create a new object**
6. **The *this* Operator**

1. Creating a Class

Type-in the following class in netbeans project. Compile and run this program.

```
/*
 * Creat the Student Class with attributes, methods and Constructors
 */
public class Student {
     String name = "Tariq";  // Student's name
     int ID = 123;           // unique ID number for the student
     public Student() {}    //default constructor(empty)
     public Student(String theName, int theID){
     name = theName;
      ID  = theID;
     }
     public void printInfo(){
      System.out.println("Student Name : "+name);
     System.out.println("Student ID    : " +ID);
     }
 }  // end of class Student
```

Constructor:

- Constructor is a special class method used to create a properly initialized instance of the class.
- Constructor must have the same name as the class itself, which is defined.
- Constructors also must have no defined return value, or type of void.
- The purpose is to create an instance to initialize the value of its members to default values.

Compile:

Compile the above program and try to Run.

- You will see the following error message.
- <no main class found>
- Now add this code segment into the **Student** class.

```
public static void main(String [] args){
     Student std1 = new Student();
     Student std2 = new Student("Fahim", 456);
     std1.printInfo();
     std2.printInfo();
}
```

***new*:**

> The keyword *new* is used to create the object of a class.

You are expected to:

Create five (5) more objects and pass different values to constructor and display the output.

2. Creating an Application class and a set of objects

Type-in the following class in the same netbeans project. Compile and run this program.

```
/*
 * Creat the MyApplcation Class to create the objects of Student class
 */
public class MyApplication {
    public static void main(String [] args){
     Student std1 = new Student();
     Student std2 = new Student("Fahim", 456);
     std1.printInfo();
     std2.printInfo();
    }
}
```

Delete the main method:

After creating this class delete the main method from **Student** class.

Compiling the Source File:

To compile your source file, choose Build > Build Main Project (F11) from the IDE's main menu.
You can view the output of the build process by choosing Window > Output > Output.
The Output window opens and displays output similar to what you see in the following figure.

You are expected to:

Add the following methods to set (Mutator) and get (Accessor) the values of name and ID attributes in **Student** class.

```
public void setName(String theName) {
      name = theName;
}
public void setID(int theID) {
       ID = theID;
}
public String gettName( ) {
      return name ;
}
public int getID( ) {
      return ID;
}
```

Use Scanner class object to get input from user and then assign these values to Student class attributes in **MyApplication** class.

***this*:**

> **If the attribute in method header has the same name as the class attribute then use *this* keyword with class attribute.**

This example shows the use of keyword *this* with class attribute.

```
public public void setName(String name) {
            this.name = name;
}
```

3. Example

Write a class named **Car** that has the following data members:

- *yearModel* – an int field that hold the car's year model.
- *make* – a String field that holds the make of the car.
- *speed* – an int field that holds the car's current speed.

The class also should have the following constructor and other methods:

- constructor – that accepts the car's year model and make as arguments. These values should be assigned to the object's *yearModel* and *make* fields. The constructor also should assign 0 to the *speed* field.
- Accessors. Appropriate accessor methods should get the values stored in an object's yearModel, make and speed fields.
- *accelerate*. The accelerate method should add 5 to the speed field each time it is called.
- *brake*. The brake method should subtract 5 from the speed field each time it is called.

```
public class Car {
      private int yearModel;
      private String make;
      private int speed;

      public Car(int yearModel, String make){
      this.yearModel = yearModel;
      this.make = make;
      speed =0;
      }

      public int getYearModel(){ return yearModel;}
      public String getMake(){ return make;}
      public int getSpeed(){ return speed;}
      public void accelerate(){ speed+=5;}
      public void brake(){ speed-=5;}

}  // end of class Car
```

Demonstrate the class in a program that creates a **Car** object, and then calls the *accelerate* method five times. After each call to the *accelerate* method, get the current speed of the car and display it. Then call the *brake* method five times. After each call to the *brake* method, get the current speed of the car and display it.

```
public class CarTest {

      public static void main(String [] args){
      Car car = new Car(2005,"Volvo");
      for(int i=1; i<=5;i++){
      car.accelerate();
      System.out.println("The speed of car is " + car.getSpeed()+
      "kms");
      }
      for(int i=1; i<=5;i++){
      car.brake();
      System.out.println("The speed of car is " + car.getSpeed()+
      "kms");
      }
   }// end main
}  // end of class CarTest
```

4. Example

Implement a Student class with the following fields, constructors and methods:
Fields (All should be **private**):

studentName;
totalScore;
numOfQuizzes;

Constructor:

public Student(String name)

Methods:

public String getName()
//The following method should return zero if no quiz has been taken.
public double getTotalScore()
public double getAverage()
public void addQuiz(double score)
// The following method should print the student's name and average score
public String toString()

```
public class Student{
        private String studentName;
        private double totalScore;
        private int numberOfQuizzes;
        public Student(String studentName){
            this.studentName = studentName;
            totalScore = 0;
            numberOfQuizzes = 0;
        }
        public String getName(){  return studentName;}
        public void addQuiz(double score){
              totalScore += score;
              numberOfQuizzes++;
        }
        public double getTotalScore() {return totalScore;}
        public double getAverageScore() {
              return totalScore/numberOfQuizzes;
        }
        public String toString(){
            return ("Student Name :" +getName()+"\nAverage Score :
            " + getAverageScore());
       }
 }
```

Write an application StudentTest that reads a student name and use the Student class to create a Student object. Then read the scores of the student in 3 quizzes and add each to the totalScore of the student using the addQuiz method, then print the student object.

Sample Output

```
Enter Student Name: Ali
Enter Quiz 1 Score for Ali: 10
Enter Quiz 2 Score for Ali: 8
Enter Quiz 3 Score for Ali: 9
Student Name : Ali
Average Score: 9.0
```

```
public class StudentTester {

    public static void main(String[] args){
        Student student= new Student("Ali");
        Scanner input = new Scanner(System.in);

        for(int i=1; i<=3;i++){
          System.out.println("Enter Quiz " + i + " Score
          for " + student.getName()+" :");
          int score = input.nextInt();
          student.addQuiz(score);
        }
        System.out.println(student);

    }
}
```

Exercises

Exercise 1(a) *(Car.java)*

Implement a class *Car*, that has the following characteristics:

a) *brandName*,
b) *priceNew*, which represents the price of the car when it was new,
c) *color*, and
d) *odometer*, which is milo meter shows number of mileage travelled by car

The class should have:

A. A method *getPriceAfterUse()* which should return the price of the car after being used according to the following formula:

$$\text{car price after being used} = priceNew \times (1 - \frac{odemeter}{600{,}000})$$

B. A method *updateMilage(double traveledDistance)* that changes the current state of the car by increasing its mileage, and
C. A method *outputDetails()* that will output to the screen all the information of the car, i.e., brand name, price new, price used, color, and odometer.

Exercise 1(b) *(TestCar.java)*

Write a test class for the *Car* class above. You are required to do the followings:

a. Create an object of type *Car*.
b. Assign any valid values to the instance variables of the object created in '**A**'.
c. Use the method *getPriceAfterUse* on the object created in '**A**' then output the result to the screen.
d. Use the method *updateMilage* on the object created in '**A**' by passing a valid value.
e. Do part '**C**' again.
f. Use the method *outputDetails* on the object created in '**A**'.

Exercise 2(a) (Flight.java)

Design then implement a class to represent a **Flight**. A Flight has a *flight number*, a *source*, a *destination* and a *number of available seats*. The class should have:

a. A **constructor** to initialize the 4 instance variables. You have to shorten the name of the source and the destination to 3 characters only if it is longer than 3 characters by a call to the method in the 'j' part.

b. An **overloaded constructor** to initialize the *flight number* and the *number of available seats* instance variables only.

(**NOTE:** Initialize the *source* and the *destination* instance variables to empty string, i.e." ")

c. An **overloaded constructor** to initialize the *flight number* instance variable only.

(**NOTE:** Initialize the *source* and the *destination* instance variables to empty string; and the *number of available seats* to zero)

d. One **accessor** method for each one of the 4 instance variables.

e. One **mutator** method for each one of the 4 instance variables **except** the *flight number* instance variable.

f. A **method** `public void reserve(int numberOfSeats)` to reserve seats on the flight. (**NOTE:** You have to check that there is enough number of seats to reserve)

g. A **method** `public void cancel(int numberOfSeats)` to cancel one or more reservations

h. A `toString` method to easily return the flight information as follows:

```
Flight No: 1234
From: KAR
To: LAH
```

i. An `equals` method to compare 2 flights.

(**NOTE:** 2 Flights considered being equal if they have the same flight number)

j. The following method:

```
private String shortAndCapital (String name) {
  if (name.length() <= 3) {
    return name.toUpperCase();
  } else {
    return name.substring(0,3).toUpperCase();
  }
}
```

Exercise 2(b) (FlightTest.java)

Write a test class for the *Flight* class you wrote. You should try to use all the methods you wrote.

Exercise 3(a) (Ship.java)

MyJava Coffee Outlet runs a catalog business. It sells only one type of coffee beans. The company sells the coffee in 2-lb bags only and the price of a single 2-lb bag is $5.50. when a customer places an order, the company ships the order in boxes. The boxes come in 3 sizes with 3 different costs:

	Large box	Medium box	Small box
Capacity	20 bags	10 bags	5 bags
Cost	$1.80	$1.00	$0.60

The order is shipped using the least number boxes. For example, the order of 52 bags will be shipped in 2 boxes: 2 large boxes, 1medium and 1 small.

Exercise 3(b) (ShipTest.java)

Develop an application that computes the total cost of an order.

Sample out put:

```
Number of Bags Ordered: 52
The Cost of Order: $ 286.00

Boxes Used:
        2 Large  - $3.60
        1 Medium - $1.00
        1 Small  - $0.60

Your total cost is: $ 291.20
```

Project

- Project proposal template is uploaded on the website under labs section.
- Write down one page problem description of your title and identify the major classes.