Assignment No. 2	
Assignment Name :	**Classes and Objects**

Class

Classes are the fundamental building blocks of any object-oriented language.

When you define a class, you declare its exact form and nature. You do this by specifying the data that it contains and the code that operates on that data. While very simple classes may contain only code or only data, most real-world classes contain both. A class is declared by use of the class keyword. A simplified general form of a class definition is shown here:

```
accessspecifier class classname
{
    DataType instancevariable1;
    DataType instancevariable 2;
    DataType instancevariable n;
    returntype methodname1 (parameter list)
    {
            Body of the method;
    }
    returntype methodname2 (parameter list)
    {
            Body of the method;
    }
}
```

The data, or variables, defined within a class are called **instance variables**. The code is contained within methods. Collectively, the methods and variables defined within a class are called members of the class. Variables defined within a class are called instance variables because each instance of the class contains its own copy of these variables. Thus, the data for one object is separate and unique from the data for another.

Objects

When you create a class, you are creating a new data type. You can use this type to declare objects of that type. However, obtaining objects of a class is a two-step process.

First, you must declare a variable of the class type. This variable does not define an object. Instead, it is simply a variable that can refer to an object. Second, you must acquire an actual, physical copy of the object and assign it to that variable. You can do this using the new operator. The new operator dynamically allocates memory for an object and returns a reference to it.

This reference is, essentially, the address in memory of the object allocated by new. This reference is then stored in the variable. Thus, in Java, all class objects must be dynamically allocated.

Syntax for declaring Object:

```
Student stud= new Student();
```

This statement combines the two steps just described.

It can be rewritten like this to show each step more clearly:

Student stud;// declare reference to object

Stud = new Student (); // allocate a object

The first line declares stud as a reference to an object of type Student. At this point, stud does not yet refer to an actual object. The next line allocates an object and assigns a reference to it to stud. After the second line executes, you can use stud as if it were a Student object. But in reality, stud simply holds, in essence, the memory address of the actual Student object.

Lab Assignment

SET A

1. Write a program to print the area of a rectangle by creating a class named 'Area' having two methods. First method named as 'setDim' takes length and breadth of rectangle as parameters and the second method named as 'getArea' returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

2. Create a class named 'Student' with String variable 'name' and integer variable 'roll_no'. Assign the value of roll_no as '2' and that of name as "John" by creating an object of the class Student.

SET B

1. Print the average of three numbers entered by user by creating a class named 'Average' having a method to calculate and print the average.

Signature of the instructor: _____ Date: _____

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]