Experiment No. 3

Title: Create a class called Employee that includes three pieces of information as instance variables- first name, a last name and a monthly salary. Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.

Objectives: 1. To learn creating class and objects.
2. To learn method and constructor..

Theory:

Objects

To work with OOP, you should be able to identify three key characteristics of objects:

- The object's *behavior*—What can you do with this object, or what methods can you apply to it?
- The object's *state*—How does the object react when you apply those methods?
- The object's *identity*—How is the object distinguished from others that may have the same behavior and state?

All objects that are instances of the same class share a family resemblance by supporting the same *behavior*. The behavior of an object is defined by the methods that you can call.

Class fundamentals

Class defines a new data type. Once defined, this new type can be used to create objects of that type. Thus, a class is a *template* for an object, and an object is an *instance* of a class. 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. As you will see, a class' code defines the interface to its data. A class is declared by use of the **class** keyword.

The general form of a class definition is shown here:

class *classname* { *type*
***instance-variable1*;**
type instance-variable2;
// ...
type instance-variableN;
type methodname1(*parameter-list*) {

```
        // body of method
}
type methodname2(parameter-list) {
        // body of method
}
// ...
type methodnameN(parameter-list) {  //
        body of 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.

Methods:

Classes usually consist of two things: instance variables and methods. This is the general form of a method:

```
type name(parameter-list) {  //
        body of method
}
```

Here, *type* specifies the type of data returned by the method. This can be any valid type, including class types that you create Methods that have a return type other than **void** return a value to the calling routine using the following form of the **return** statement: return *value*; Here, *value* is the value returned.

Constructor:

A *constructor* initializes an object immediately upon creation. It has the same name as the class in which it resides and is syntactically similar to a method. Once defined, the constructor is automatically called immediately after the object is created, before the new operator completes. Constructors look a little strange because they have no return type, not even **void**. It is the constructor's job to initialize the internal state of an object so that the code creating an instance will have a fully initialized, usable object immediately.

Constructor Overloading:

In addition to overloading normal methods, you can also overload constructor methods. In fact, for most real-world classes that you create, overloaded constructors will be the norm, not the exception.

Reading input from keyboard

There are various ways to read input from the keyboard; the java.util.Scanner class is one of them. The **Java Scanner** class breaks the input into tokens using a delimiter that is whitespace by default. It provides many methods to read and parse various primitive values. Java Scanner class is widely used to parse text for string and primitive types using regular expression. To use scanner class, you need to import class util. e.g.

```
import java.util.Scanner;  class ScannerTest{
public static void main(String args[]){
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter your rollno");
                int rollno=sc.nextInt();
                System.out.println("Enter your name");
                String name=sc.next();
                System.out.println("Enter your fee");  double
                fee=sc.nextDouble();
                System.out.println("Rollno:"+rollno+" name:"+name+" fee:"+fee);  sc.close();
        }
}
```

Key concepts: class, method, object, constructor.

Algorithm:

1. Create class employee with data member's first name, a last name and a monthly salary.
2. Define a default constructor and initialize the instance variables.
3. Write a method set which accepts values for instance variables from user.
4. If user provides negative value for monthly salary, set it to 0.0(zero).
5. Write a get method to display the instance-variables.
6. Define class EmployeeTest, write a main method in this class.
7. Create two Employee objects and display each object's yearly salary.
8. Raise the salary by 10% and display each object's yearly salary