Practical 02 - Encapsulation

We have already discussed a about encapsulation while discussing OOPs concepts.

The whole idea behind encapsulation is to hide the implementation details from users. If a data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class. However if we setup public getter and setter methods to update (for e.g. void setSSN(int ssn))and read (for e.g. int getSSN()) the private data fields then the outside class can access those private data fields via public methods. This way data can only be accessed by public methods, thus making the private fields and their implementation hidden for outside classes. That's why encapsulation is known as data hiding.

```
public class EncapsulationDemo{
    private String empName;

    //Getter and Setter methods

    public String getEmpName(){
        return empName;
    }

    public void setEmpName(String newValue){
        empName = newValue;
    }

}
public class EncapsTest{
    public static void main(String args[]){
         EncapsulationDemo obj = new EncapsulationDemo();
         obj.setEmpName("Mario");
         System.out.println("Employee Name: " + obj.getEmpName());
    }
}
```

Exercise 3-1: Develop a code for the following scenario.

"An encapsulated class contains three variables to store Name, Age and Salary of the employee. Evelop getters and setters to set and get values. Develop a test class to test your code."

```
public class Test

{
  //data

  private String name;

  private int age;

  private float salary;
```

```
   //methods
   public void setName(String n)
   {
      name=n;
   }
   public String getName()
   {
      return name;
   }
   public void setAge(int a)
   {
      age=a;
   }
   public int getAge()
   {
      return age;
   }
   public void setSalary(float s)
   {
      salary=s;
   }
   public float getSalary()
   {
      return salary;
   }
}

public class EncapTest
```

```
{
   public static void main(String[] args)
   {
      Test t1=new Test();
      t1.setName("Fazla Fiyas");
      t1.setAge(21);
      t1.setSalary(125000.00f);
      System.out.println("Name: "+t1.getName());
      System.out.println("Age: "+t1.getAge());
      System.out.println("Salary: "+t1.getSalary());
   }
}
```

Now modify the same code by trying to replace the setters using a constructor.

```
public class Test
{
   //data
   private String name;
   private int age;
   private float salary;

   //methods
   public void setDetails(String n,int a,float s)
   {
      name=n;
      age=a;
      salary=s;
   }
   public String getName()
   {
      return name;
   }
```

Practical 02 - Encapsulation

```
   public int getAge()
   {
      return age;
   }
   public float getSalary()
   {
      return salary;
   }
}


public class EncapTest
{
   public static void main(String[] args)
   {

      Test t1=new Test();
      t1.setDetails("fazla fiyas", 21, 125000.00f);
      System.out.println("Name: "+t1.getName());
      System.out.println("Age: "+t1.getAge());
      System.out.println("Salary: "+t1.getSalary());
   }
}
```