Practical 03 - 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."

```
package com.mycompany.empobj1;

public class employee

{

  private String name;

  private int age;

  private double salary;
```

```
//setter method
public void setname(String n)
{
   name=n;
}
public void setage(int a)
{
   age=a;
}
public void setsalary(double s)
{
   salary=s;
}
//getter method
public String getname()
{
   return name;
}
public int getage()
{
   return age;
}
public double getsalary()
{
        return salary;
    }
}
```

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

```
              package com.mycompany.empobj1;


public class employee
{
    private String name;
    private int age;
    private double salary;


    public employee(String n,int a, double s)
    {
        n=name;
        a=age;
        s=salary;
    }
}
```

Exercise 3-2: Code for the last example that we have discussed during the class. We need the following Output. (Use Netbeans code generation option where necessary)

Employee Name: xxxxx (Use setter to set and getter to retrieve)

Basic Salary: xxxx (Use setter to set and getter to retrieve)

Bonus: xxxx (You may use the constructor to pass this value)

Bonus Amount: xxxxx (Develop a separate method to calculate Bonus amount. Bonus amount is the total of Bonus and Basic Salary)

E.g.

```
Employee Name: Bogdan
Basic Salary: 50000
Bonus: 10000
Bonus Amount: 60000
```

Main class

```
package com.mycompany.empobj1;


public class Empobj1 {


  public static void main(String[] args)

  {

     employee e1= new employee(10000.00f);

           e1.setname("Bogdan");

           e1.setsalary(50000.00f);

           e1.CalcBonus();
```

```
            e1.display();
   }


}
```

<u>Employee class</u>

```
package com.mycompany.empobj1;

public class employee
{
   private String name;
   private double salary;
   private double bonus;
   private double amount;

   public void setname(String n)
   {
      name=n;
   }
   public void setsalary(double s)
   {
       salary=s;
```

```
    }
    // getter method
    public String getname()
    {
       return name;
    }
    public double getsalary()
    {
       return salary;
    }

    public employee(double b)
    {
      bonus= b;
    }
    public double CalcBonus()
    {
       amount=salary+bonus;
       return amount;
    }
    public void display()
    {
       System.out.println("Employee Name: "+name);
         System.out.println("Basic Salary: "+salary);
         System.out.println("Bonus: "+bonus);
```

```
        System.out.println("Bonus Amount: "+amount);
    }
}
```