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());
    }
}
```

Answer

Let's break down the code step by step:

- We have a class **'EncapsulationDemo'** with a private data member **'empName'**. By making **'empNam'e** private, we are hiding the implementation details of this variable from other classes.
- To access and modify the value of **'empName'**, we provide public getter and setter methods**: 'getEmpName()**' and '**setEmpName(String newValue**)'. These methods allow other classes to read and update the value of **'empName'** indirectly.

- In the '**EncapsTest**' class, we create an object **'obj'** of the **'EncapsulationDemo'** class.

- We use the **'setEmpName("Mario")**' method to set the value of '**empName**' to "Mario".
- Finally, we use the **'getEmpName()**' method to retrieve the value of **'empName'** and print it to the console using **'System.out.println()'**

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."

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

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

Answers

public class Employee { private String name; private int age;
private double basicSalary;

public Employee(String name, int age, double basicSalary) { this.name = name;
this.age = age;
this.basicSalary = basicSalary;

```
}
// Getter and setter methods for name public String getName() {

return name; }

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

}
// Getter and setter methods for age public int getAge() {

return age; }

public void setAge(int age) { this.age = age;

}
// Getter and setter methods for basicSalary public double getBasicSalary() {

return basicSalary; }
```



```
public void setBasicSalary(double basicSalary) { this.basicSalary = basicSalary;

}

// Method to calculate bonus amount
public double calculateBonusAmount(double bonus) {

return basicSalary + bonus; }

}
public class TestEmployee {

public static void main(String[] args) {
// Create an Employee object using the constructor Employee employee = new
Employee("Bogdan", 30, 50000);

// Use setter and getter methods employee.setName("Bogdan"); employee.setAge(30);
employee.setBasicSalary(50000);

// Print employee details using getter methods System.out.println("Employee Name: " +
employee.getName()); System.out.println("Basic Salary: " + employee.getBasicSalary());
```

```
// Pass bonus using the constructor and calculate bonus amount double bonus = 10000;
System.out.println("Bonus: " + bonus);
double bonusAmount = employee.calculateBonusAmount(bonus); System.out.println("Bonus
Amount: " + bonusAmount);

} }
```

OUTPUT

```
Employee Name: Bogdan

Basic Salary: 50000

Bonus: 10000
Bonus Amount: 60000
```