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.”

**// Encapsulated class**

**class Employee {**

**private String name;**

**private int age;**

**private double salary;**

**// Constructor**

**public Employee(String name, int age, double salary) {**

**this.name = name;**

**this.age = age;**

**this.salary = salary;**

**}**

**// Getter for name**

**public String getName() {**

**return name;**

**}**

**// Setter for name**

**public void setName(String name) {**

**this.name = name;**

**}**

**// Getter for age**

**public int getAge() {**

**return age;**

**}**

**// Setter for age**

**public void setAge(int age) {**

**this.age = age;**

**}**

**// Getter for salary**

**public double getSalary() {**

**return salary;**

**}**

**// Setter for salary**

**public void setSalary(double salary) {**

**this.salary = salary;**

**}**

**}**

**// Test class**

**public class TestEmployee {**

**public static void main(String[] args) {**

**// Creating an employee object**

**Employee emp = new Employee("Nimal", 25, 70000.0);**

**// Getting and printing employee information using getters**

**System.out.println("Employee Name: " + emp.getName());**

**System.out.println("Employee Age: " + emp.getAge());**

**System.out.println("Employee Salary: " + emp.getSalary());**

**// Updating employee information using setters**

**emp.setName("Kamal");**

**emp.setAge(30);**

**emp.setSalary(48000.0);**

**// Getting and printing updated employee information**

**System.out.println("\nUpdated Employee Name: " + emp.getName());**

**System.out.println("Updated Employee Age: " + emp.getAge());**

**System.out.println("Updated Employee Salary: " + emp.getSalary());**

**}**

**}**

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

**// Encapsulated class**

**class Employee {**

**private String name;**

**private int age;**

**private double salary;**

**// Constructor**

**public Employee(String name, int age, double salary) {**

**this.name = name;**

**this.age = age;**

**this.salary = salary;**

**}**

**// Getter for name**

**public String getName() {**

**return name;**

**}**

**// Getter for age**

**public int getAge() {**

**return age;**

**}**

**// Getter for salary**

**public double getSalary() {**

**return salary;**

**}**

**}**

**// Test class**

**public class TestEmployee {**

**public static void main(String[] args) {**

**// Creating an employee object using constructor**

**Employee emp = new Employee("John Doe", 30, 50000.0);**

**// Getting and printing employee information using getters**

**System.out.println("Employee Name: " + emp.getName());**

**System.out.println("Employee Age: " + emp.getAge());**

**System.out.println("Employee Salary: " + emp.getSalary());**

**}**

**}**

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

**class Employee {**

**private String name;**

**private double basicSalary;**

**private double bonus;**

**public Employee(String name, double basicSalary, double bonus) {**

**this.name = name;**

**this.basicSalary = basicSalary;**

**this.bonus = bonus;**

**}**

**public String getName() {**

**return name;**

**}**

**public double getBasicSalary() {**

**return basicSalary;**

**}**

**public void setBasicSalary(double basicSalary) {**

**this.basicSalary = basicSalary;**

**}**

**public double getBonus() {**

**return bonus;**

**}**

**public void setBonus(double bonus) {**

**this.bonus = bonus;**

**}**

**public double calculateBonusAmount() {**

**return basicSalary + bonus;**

**}**

**}**

**public class TestEmployee {**

**public static void main(String[] args) {**

**Employee emp = new Employee("Bogdan", 50000, 10000);**

**System.out.println("Employee Name: " + emp.getName());**

**System.out.println("Basic Salary: " + emp.getBasicSalary());**

**System.out.println("Bonus: " + emp.getBonus());**

**System.out.println("Bonus Amount: " + emp.calculateBonusAmount());**

**}**

**}**