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.”

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

```
public class Employee
{
    private String Name;
    private int Age;
    private double Salary;

    // Getter & Setter Methods

    public Employee(String Name,int Age,double Salary)
    {
        this.Name=Name;
        this.Age=Age;
        this.Salary=Salary;
    }
    public String getName()
    {
        return Name;
    }
    public int getAge()
    {
        return Age;
    }
    public double getSalary()
    {
        return  Salary;
    }

}
```

```
public class Test {
    public static void main(String[] args)
    {
        Employee e1=new Employee("kavinda",21,250000.00);
        System.out.println("Hi "+e1.getName()+" You are "+e1.getAge()+" years
old & your salary is "+e1.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

```
public class Employee
{
    private String Name;
    private double Salary;
    private double Bonus;

    // Getter & Setter Methods

    public Employee(String Name,double Salary,double Bonus)
    {
        this.Name=Name;
        this.Salary=Salary;
        this.Bonus=Bonus;

    }
    public String getName()
    {
        return Name;
    }
    public double getSalary()
    {
        return  Salary;
    }
    public double getBonus()
    {
        return Bonus;
    }
    public double getBonus_Amount() {
        return Salary+Bonus;
    }
}
```

Practical 03 - Encapsulation

```
public class Test {
    public static void main(String[] args)
    {
        Employee emp=new Employee("Kavinda",250000,10000);
        System.out.println("Employee Name: "+emp.getName());
        System.out.println("Employee Basic Salary: "+emp.getSalary());
        System.out.println("Employee Bonus: "+emp.getBonus());
        System.out.println("Employee Bonus Amount: "+emp.getBonus_Amount());
    }
}
```