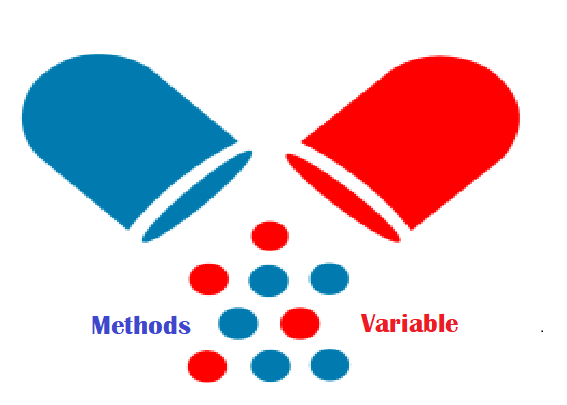
**Encapsulation-**

* The binding of data into single entity called as “Encapsulation.”
* Integrating data (variables) and code (methods) into a single unit.
* A class's variables are hidden from other classes and can only be accessed by the methods of the class in which they are found.



Why Encapsulation?

* In Java, encapsulation helps us to keep related fields and methods together, which makes our code cleaner and easy to read.
* By using getters and setter programming will be effective.
* It provides security as it not allow direct access from external classes.
* It provides flexibility as changes in one part can be successfully implemented in other part of code.
* It helps to control the values of our data fields. For example,

class Person {

private int age;

public void setAge(int age) {

if (age >= 0) {

this.age = age;

}

}

}

Here, we are making the age variable private and applying logic inside the setAge() method. Now, age cannot be negative.

• The getter and setter methods provide read-only or write-only access to our class fields. For example,

getName() // provides read-only access

setName() // provides write-only access

Real Time Example-

Suppose you have an account in the bank. If your balance variable is declared as a public variable in the bank software, your account balance will be known as public, In this case, anyone can know your account balance. So, is it correct approach? answer is No.

So, they declare balance variable as private for making your account safe, so that anyone cannot see your account balance.

The person who has to see his account balance, will have to access only private members through public methods defined inside that class.

Thus, we can achieve security by utilizing the concept of data hiding. This is called Encapsulation in Java.

Why ?

Employee e1= new Employee();

e1.salary=5000; //case 1

Employee e2= new Employee();

e2.salary=-3000; //case 2

In this example, case 1, we are passing the 5000 salary to the employee that is correct but case 2, we are passing the salary -3000 that is the negative salary.

So salary can not be negative.

How we are going to achieve this by using encapsulation-

**public** **class** Employee {

**private** **int** salary;

**public** **void** setSalary(**int** sal) {

**if** (sal > 0) {

salary = sal;

} **else** {

salary = 0;

}

}

**public** **int** getSalary() {

**return** salary;

}

}

In this example, we are checking the whether the salary is greater than zero. Because salary cannot be negative so in this way, we are going to achieve the encapsulation.

Note- Always keeps global variables private and allows others to assign values through public methods.

Program for Encapsulation using hard coded values

**package** com.encapsulation;

**public** **class** EncapsulationTest {

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

Employee employee= **new** Employee ();

employee.setSalary(-5000);

System.***out***.println("salary>>"+employee.getSalary());

}

}

Program for Encapsulation using Dynamic values.

**package** com.encapsulation;

**public** **class** Employee {

**private** **int** employeeId;

**private** String employeeName;

**private** String employeeCity;

**public** **int** getEmployeeId() {

**return** employeeId;

}

**public** **void** setEmployeeId(**int** employeeId) {

**this**.employeeId = employeeId;

}

**public** String getEmployeeName() {

**return** employeeName;

}

**public** **void** setEmployeeName(String employeeName) {

**this**.employeeName = employeeName;

}

**public** String getEmployeeCity() {

**return** employeeCity;

}

**public** **void** setEmployeeCity(String employeeCity) {

**this**.employeeCity = employeeCity;

}

}

**package** com.encapsulation;

**import** java.util.Scanner;

**public** **class** TestMain {

**public** **static** **void** getUserInput() {

System.***out***.println("Enter the ID>>");

Scanner scanner = **new** Scanner(System.***in***);

**int** id = scanner.nextInt();

System.***out***.println("Enter the Name>>");

String name = scanner.next();

System.***out***.println("Enter the City");

String city = scanner.next();

Employee employee = **new** Employee();

employee.setEmployeeId(id);

employee.setEmployeeName(name);

employee.setEmployeeCity(city);

System.***out***.println("Employee Id>>" + employee.getEmployeeId());

System.***out***.println("Employee Name>>" + employee.getEmployeeName());

System.***out***.println("Employee City>>" + employee.getEmployeeCity());

}

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

*getUserInput*();

}

}

Output-

Enter the ID>>

10

Enter the Name>>

ram

Enter the City

pune

Employee Id>>10

Employee Name>>ram

Employee City>>pune