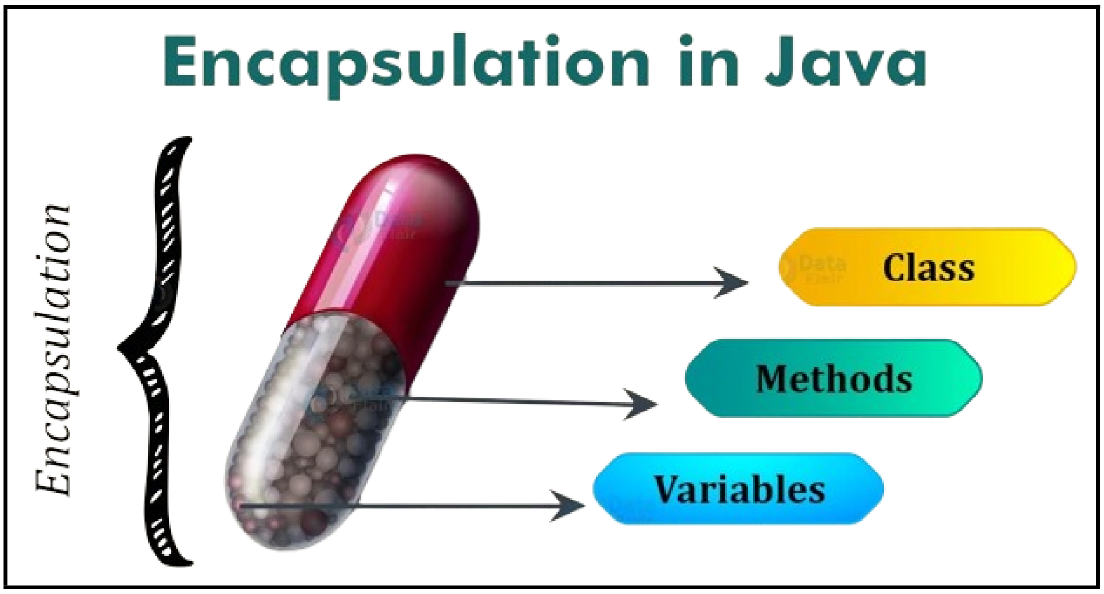
WEEK6

OOP concepts: Encapsulation Concept; What is encapsulation? How to achieve encapsulation in java; Packages; Single Responsibility Principle: Intent; Rules; Benefits; example

Encapsulation concept:

Think of a **capsule** that holds medicine. The capsule protects the medicine inside and ensures it can only be taken the right way (by swallowing it). **Encapsulation in Java** works similarly. It hides the important details (data) inside a "capsule" (class) and allows you to interact with it in a controlled and safe way.



Encapsulation in Java

- Encapsulation is a principle of hiding data (variables) inside a class and allowing controlled access to it using methods.
- It protects the data from being accessed or modified directly by restricting it to the class and exposing it only through specific getter and setter methods.
- **Encapsulation in Java** is a process of wrapping code and data together into a single unit.
- We can create a fully encapsulated class in Java by making all the data members of the class private.
- Now we can use setter and getter methods to set and get the data in it.
- By providing only a setter or getter method, you can make the class **read-only or write-only**.

How to Implement Encapsulation in Java

1. Declare class variables as **private**.
2. Provide public **getter** and **setter** methods to access and update the variables.

Example:

```
class Student
{
      // Private variable (data hiding)
      private String name;

      // Getter method to access the name
      public String getName()
      {
            return name;
      }

      // Setter method to modify the name
      public void setName(String name)
      {
            this.name = name;
      }
}
public class EncapsMain
{
      public static void main(String[] args)
      {
            Student s = new Student();

            // Set the name using the setter method
            s.setName("John");

            // Get the name using the getter method
            System.out.println("Student Name: " + s.getName());
      }
}
```

Example: A Bank Account

Imagine you have a **bank account**. You don’t let anyone see or change the balance directly. Instead, they have to ask the bank to:

- Check the balance (getter).
- Deposit or withdraw money (setter).

```
class BankAccount
{
    // Private variable (hiding the data)
    private int balance;

    // Getter method (to see the balance)
    public int getBalance()
    {
          return balance;
    }
```

```
        // Setter method (to change the balance)
        public void depositMoney(int amount)
        {
                if (amount > 0)
                {
                        balance += amount;
                }
        }
    }

public class BankMain
{
        public static void main(String[] args)
        {
                BankAccount myAccount = new BankAccount();

                // Deposit money
                myAccount.depositMoney(500);

                // Get the balance
                System.out.println("Current Balance: " + myAccount.getBalance());
        }
}
```

Example: Managing student marks

Let’s say we are creating a program to manage a student’s marks. We don’t want others to directly change the marks because they might enter invalid values.

```
public class Student
{
        // Private data (hidden)
        private int marks;

        // Public method to set marks
        public void setMarks(int marks)
        {
                if (marks >= 0 && marks <= 100) // Check for valid marks
                {
                        this.marks = marks;
                }
                else
                {
                        System.out.println("Invalid marks! Please enter a value between 0
                        and 100.");
                }
        }
        // Public method to get marks
        public int getMarks()
        {
                return marks;
        }
}
public class StudentMain
{
        public static void main(String[] args)
        {
                Student student = new Student();

                // Set marks using the setter method
                student.setMarks(85); // Valid
                System.out.println("Marks: " + student.getMarks()); // Output: Marks: 85

                // Try setting invalid marks
                student.setMarks(120); // Output: Invalid marks! Please enter a value
                between 0 and 100.
        }
}
```

Single Responsibility Principle (SRP)

- When we design our classes, we need to ensure that our class is responsible only for 1 task or functionality and when there is a change in that task/functionality, only then, that class should change.
- When our classes do not adhere to this principle, we would be making too many changes to our classes to make our classes adaptable to the new business requirements. This could involve lots of side effects, retesting, and introducing new bugs.

Benefits of Single Responsibility Principle

- When an application has multiple classes, each of them following this principle, then the application becomes more maintainable, easier to understand.
- The code quality of the application is better, thereby having fewer defects.
- On boarding new members are easy, and they can start contributing much faster.
- Testing and writing test cases is much simpler.

Program No1: Write a java program to implement read-only class.

```
public class Employee
{
	private String name = “Ramesh”;
	public String getName()
	{
		return name;
	}
}

public class ReadOnly
{
	public static void main(String[] args)
	{
		Employee e = new Employee();
		System.out.println(e.getName());
	}
}
```

Output:

Program No2: Write a java program to implement read-write class

```
public class Employee
{
        private String name;
        public String getName()
        {
                return name;
        }
        public void setName(String name)
        {
                this.name=name ;
        }
}

public class ReadWrite
{
        public static void main(String[] args)
        {
                Employee e = new Employee();
                e.setName("Charles");
                System.out.println(e.getName());
        }
}
```

Program No3: Write a java program to create student registration form and the registered student age should be at least 16 by using encapsulation class.

```
import java.lang.*;
import java.util.Scanner;

class Student
{
	private int rollno, age, sem;
	private String name, gender, address;

	public int getRollNo()
	{
		return (rollno);
	}
	public void setRollNo(int r)
	{
		rollno = r;
	}

	public int getAge()
	{
		return(age);
	}
	public void setAge(int a)
	{
		if(a < 16)
		{
			System.out.println("Age bellow 16 students are not allowed");
			System.exit(0);
		}
		age = a;
	}

	public int getSem()
	{
		return(sem);
	}
	public void setSem(int s)
	{
		sem = s;
	}
	public String getName()
	{
		return(name);
	}
	public void setName(String n)
	{

		name = n;
	}

	public String getGender()
	{
		return(gender);
	}
	public void setGender(String g)
	{

		gender = g;
	}

	public String getAddress()
	{
		return(address);
	}
	public void setAddress(String a)
	{
		address = a;
	}
}

class StudentReg
{
	public static void main(String args[])
	{
		int r,a,s;
		String n,g,ad;
		Student S = new Student();

		System.out.println("Student Registration Form");
		Scanner input = new Scanner(System.in);

		System.out.println("Name: ");
		n = input.next();
		S.setName(n);
```

```
                System.out.println("Roll No: ");
                r = input.nextInt();
                S.setRollNo(r);

                System.out.println("Gender:");
                g = input.next();
                S.setGender(g);

                System.out.println("Sem: ");
                s = input.nextInt();
                S.setSem(s);

                System.out.println("Age: ");
                a = input.nextInt();
                S.setAge(a);

                System.out.println("Address: ");
                ad = input.next();
                S.setAddress(ad);
                input.close();

                System.out.println("---Entered student details are---");
                System.out.println(S.getName());
                System.out.println(S.getRollNo());
                System.out.println(S.getGender());
                System.out.println(S.getSem());
                System.out.println(S.getAge());
                System.out.println(S.getAddress());
        }
}
```

Program No4: Write a java program to implement a simple calculator.

```
public class Addition
{
        private double no1, no2, result;

        public double getNo1()
        {
                return no1;
        }
        public void setNo1(double n1)
        {
                no1 = n1;
        }

        public double getNo2()
        {
                return no2;
        }
        public void setNo2(double n2)
        {
                no2 = n2;
        }

        public double calculate()
        {
                result = no1 + no2;
                return result;
        }
}
```

```
public class Substraction
{
        private double no1, no2, result;

        public double getNo1()
        {
                return no1;
        }
        public void setNo1(double n1)
        {
                no1 = n1;
        }

        public double getNo2()
        {
                return no2;
        }
        public void setNo2(double n2)
        {
                no2 = n2;
        }

        public double calculate()
        {
                result = no1 - no2;
                return result;
        }
}
```

```
public class Multiplication
{
	private double no1, no2, result;

	public double getNo1()
	{
		return no1;
	}
	public void setNo1(double n1)
	{
		no1 = n1;
	}

	public double getNo2()
	{
		return no2;
	}
	public void setNo2(double n2)
	{
		no2 = n2;
	}

	public double calculate()
	{
		result = no1 * no2;
		return result;
	}
}
```

```
public class Devision
{
        private double no1, no2, result;

        public double getNo1()
        {
                return no1;
        }
        public void setNo1(double n1)
        {
                no1 = n1;
        }

        public double getNo2()
        {
                return no2;
        }
        public void setNo2(double n2)
        {
                no2 = n2;
        }

        public double calculate()
        {
                result = no1 / no2;
                return result;
        }
}
```

```
import java.util.Scanner;
class Calculator
{
        public static void main(String[] args)
        {
                int choice;
                Double number1, number2, result;
                Scanner read = new Scanner(System.in);

                System.out.println("1. Addition");
                System.out.println("2. Substraction");
                System.out.println("3. Multiplication");
                System.out.println("4. Division");

                choice = read.nextInt();

                System.out.println("Enter first number");
                number1 = read.nextDouble();

                System.out.println("Enter second number");
                number2 = read.nextDouble();

                switch (choice)
                {
                        case 1:
                                Addition A = new Addition();
                                A.setNo1(number1);
                                A.setNo2(number2);
                                result = A.calculate();
                                System.out.println(A.getNo1() + " + " + A.getNo2() + " = " +
                                result);
                        break;
                        case 2:
                                Substraction S = new Substraction();
                                S.setNo1(number1);
                                S.setNo2(number2);
                                result = S.calculate();
                                System.out.println(S.getNo1() + " - " + S.getNo2() + " = " +
                                result);
                        break;
                        case 3:
                                Multiplication M = new Multiplication();
                                M.setNo1(number1);
                                M.setNo2(number2);
                                result = M.calculate();
                                System.out.println(M.getNo1() + " * " + M.getNo2() + " = " +
                                result);
                        break;
                        case 4:
                                Devision D = new Devision();
                                D.setNo1(number1);
                                D.setNo2(number2);
                                result = D.calculate();
                                System.out.println(D.getNo1() + " / " + D.getNo2() + " = " +
                                result);
                        break;

                        default:
                                System.out.println("Invalid option!");
                        break;
                }
                input.close();
        }
}
```