1. What is Encapsulation in Java? Why is it called Data Hiding?

Encapsulation is one of the **four pillars of Object-Oriented Programming (OOP)** that restricts **direct access** to the data members of a class and allows controlled access through methods.

It is called **data hiding** because:

- The data (variables) are **private** and cannot be accessed directly.
- External access is only through **getter and setter** methods.

Example:

java
Copy code
```
class Person {
    private String name;  // Private variable

    public void setName(String name) {  // Setter method
        this.name = name;
    }

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

public class EncapsulationExample {
    public static void main(String[] args) {
        Person obj = new Person();
        obj.setName("John");
        System.out.println("Name: " + obj.getName());
    }
}
```

Output:

makefile
Copy code
```
Name: John
```

2. What are the Important Features of Encapsulation?

1. **Data Hiding** – Protects data by making variables private.
2. **Data Access Control** – Provides controlled access using getter and setter methods.
3. **Improves Code Maintainability** – Changes to the implementation do not affect other parts of the program.
4. **Prevents Accidental Modification** – Encapsulated data cannot be changed unexpectedly.
5. **Enhances Security** – Restricts access to sensitive data.

---

3. What are Getter and Setter Methods in Java? Explain with an Example.

- **Getter Method** – Used to retrieve the value of a private variable.
- **Setter Method** – Used to set or modify the value of a private variable.

java
Copy code
```java
class Car {
    private String model;  // Private variable

    // Setter method
    public void setModel(String model) {
        this.model = model;
    }

    // Getter method
    public String getModel() {
        return model;
    }
}

public class GetterSetterExample {
    public static void main(String[] args) {
        Car car = new Car();
        car.setModel("Tesla Model X");
        System.out.println("Car Model: " + car.getModel());
    }
}
```

Output:

yaml
Copy code
```
Car Model: Tesla Model X
```

---

4. What is the Use of `this` Keyword? Explain with an Example.

The `this` keyword refers to the **current instance** of the class.

- It is used to differentiate **instance variables** from **local variables** when they have the same name.
- It can be used to call **constructors and methods** of the current class.

java
Copy code
```
class Student {
    private String name;

    // Constructor using 'this' keyword
    public Student(String name) {
        this.name = name;
    }

    public void display() {
        System.out.println("Student Name: " + this.name);
    }
}

public class ThisKeywordExample {
    public static void main(String[] args) {
        Student s = new Student("Alice");
        s.display();
    }
}
```

Output:

yaml
Copy code
```
Student Name: Alice
```

---

5. What is the Advantage of Encapsulation?

1. **Security** – Prevents unauthorized access to data.
2. **Code Reusability** – Easily reusable and maintainable.
3. **Flexibility** – Allows changes without affecting other parts of the program.
4. **Increases Readability** – Makes the code cleaner and more organized.

---

6. How to Achieve Encapsulation in Java? Give an Example.

Encapsulation is achieved using:

1. **Private Variables** – Declare class variables as `private`.
2. **Public Getter & Setter Methods** – Provide controlled access.

Example:
java
Copy code
```
class Employee {
    private int empID;
    private String empName;

    // Setter methods
    public void setEmpID(int empID) {
        this.empID = empID;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    // Getter methods
    public int getEmpID() {
        return empID;
    }
    public String getEmpName() {
        return empName;
    }
}

public class EncapsulationDemo {
    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.setEmpID(101);
        emp.setEmpName("John Doe");

        System.out.println("Employee ID: " + emp.getEmpID());
        System.out.println("Employee Name: " + emp.getEmpName());
    }
}
```

Output:

yaml
Copy code

```
Employee ID: 101
Employee Name: John Doe
```