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

Ans-Binding of data and corresponding methods in a single unit is Called encapsulation. The key feature of encapsulation is **access control**, which allows you to decide which parts of an object are accessible from the outside world.

It is called data hiding because it hides the data from the outer world, that is, its data is not accessible to the outer world with the help of an access modifier called:**private.**

2.What are the important features of Encapsulation

Ans-1. **Data Hiding**

- Encapsulation restricts direct access to the internal data of a class.
- It hides implementation details from the outside world, exposing only what is necessary.
- This improves data security by preventing unauthorized or accidental modification of data.

2. Controlled Access

- Encapsulation provides controlled access to class data through **getters** and **setters** (public methods).
- This allows validation and logic to be applied before accessing or modifying the data.

Abstraction

- Encapsulation complements abstraction by exposing only the relevant details and hiding unnecessary ones.
- It allows users to interact with an object at a higher level, focusing on what it does rather than how it works

3.What are getter and setter methods in Java Explain with an example

Ans:Getter Method:

- A public method used to retrieve the value of a private field.
- Typically prefixed with `get`.
- Returns the value of the field.

2. **Setter Method:**
   - A public method used to set or update the value of a private field.
   - Typically prefixed with `set`.
   - Accepts a parameter to assign a value to the field.
   - Often includes validation logic to ensure valid data.

Example: Getter and Setter Methods

public class Person {

```
// Private fields (data hiding)
private String name;
private int age;

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

// Setter for 'name'
public void setName(String name) {
    this.name = name; // Assigning value to the field
}

// Getter for 'age'
public int getAge() {
    return age;
}

// Setter for 'age' with validation
public void setAge(int age) {
    if (age > 0) { // Ensuring valid age
        this.age = age;
    } else {
        System.out.println("Age must be positive.");
```

```
        }
    }
}
```

4.What is the use of this keyword explain with an example

Ans:”this” keyword is used to solve the problem of data shadow in instance member function,this problem occurs when local and instance variable have same name and if we want to pass an argument to the function this creates confusion to JVM to determine which one is local and which one is instance variable,so to solve this problem we use this keyword that help JVM to identify which one is caller object.

Example:

```
public class Employee {

    private String name;



    public void Emp(String name) {
        this.name = name; // 'this.name' refers to the instance variable
    }

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

    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.Emp("Shubham");
        emp.display(); // Output: Employee Name: Shubham
    }

}
```

5.What is the advantage of Encapsulation

Ans:-Encapsulation provides several advantages in software development, making it a cornerstone of object-oriented programming (OOP). Here are the key benefits:

Advantages of Encapsulation

1. Data Security and Protection (Data Hiding):

- Encapsulation restricts direct access to the internal state of an object by making fields `private`.
- It prevents unauthorized or unintended modification of the data.
- Access is provided through controlled getter and setter methods, which can include validation logic.

2. Improved Maintainability:

- Encapsulation allows changes to the internal implementation of a class without affecting external code.
- For example, you can modify a method or change how a field is calculated without altering the public interface.

Scenario:

- If you change how `balance` is calculated in the `Account` class, external code using the `getBalance()` method remains unaffected.

3. Reduces Complexity:

- Encapsulation hides the implementation details, exposing only the essential features through a simplified interface.
- This abstraction reduces complexity for the user of the class.

Example: When using a bank's API to check your balance, you don't need to know how the database query works; you just call a `getBalance()` method.

4. Better Control Over Data:

- You can control how data is accessed or modified. For example:
    - Make fields **read-only** by providing a getter but no setter.
    - Make fields **write-only** by providing a setter but no getter.

6.How to achieve encapsulation in Java? Give an example.

Ans:Steps to Achieve Encapsulation

1. **Declare fields (variables) of a class as `private`.**

- This ensures that the fields are not directly accessible from outside the class**.**

2. **Provide `public` getter and setter methods to access and update the private fields.**
   - Getter methods are used to retrieve the values of the fields.
   - Setter methods are used to modify the values of the fields.
   - Validation logic can be added in the setter methods to control data changes.
3. **Use the class's methods to interact with its fields instead of directly accessing them**

Example:

```
public class Student {

    // Step 1: Declare fields as private

    private String name;

    private int age;


    // Step 2: Provide public getter and setter methods

    // 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' with validation
    public void setAge(int age) {
        if (age > 0) { // Age must be positive
            this.age = age;
        } else {
            System.out.println("Age must be a positive number.");
        }
    }
}
```