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

Ans - Encapsulation in Java is one of the four fundamental Object-Oriented Programming (OOP) principles, the others being inheritance, polymorphism, and abstraction. It involves bundling the data (attributes) and methods (functions) that operate on the data into a single unit called a class. It restricts direct access to some of an object's components and exposes only the necessary functionalities. This concept is called "data hiding" because the internal details of how the data is stored and manipulated are hidden from the outside world, which helps in maintaining data integrity and providing a well-defined interface to interact with the object.

2. What are the important features of Encapsulation?

Ans - The important features of encapsulation are:

- Data Hiding: Internal details of the class are hidden from outside access.

- Access Control: Control over which parts of the class are accessible and which are not.

- Modularity: Changes in the internal implementation do not affect the external code using the class.

- Flexibility

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

Ans - Getter and setter methods are used to access and modify the private fields of a class in a controlled manner. Getter methods are used to retrieve the values of private fields, and setter methods are used to set or modify those values.

Example:

public class Person {

private String name; // Private field

public String getName() {

return name; // Getter method.

}

public void setName(String newName) {

name = newName; // Setter method

}

}

public class Main {

public static void main(String[] args) {

Person person = new Person();

person.setName("Tejas"); // Using the setter

System.out.println(person.getName()); // Using the getter

}

}

4. What is the use of the `this` keyword? Explain with an example.

Ans - The `this` keyword in Java is a reference to the current object instance. It is often used inside a class to refer to its own instance variables or methods. This is particularly useful when there might be ambiguity between instance variables and method parameters with the same name.

Example:

public class Student {

private String name;

public void setName(String name) {

this.name = name; // "this.name" refers to the instance variable, "name" refers to the method parameter

}

}

5. What is the advantage of Encapsulation?

Ans - Encapsulation offers several advantages:

- Data Protection

- Flexibility

- Code Organization

- Code Maintenance

- Controlled Access

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

Ans - Encapsulation is achieved by declaring the class's attributes as private and providing public methods (getter and setter) to access and modify those attributes. Here's a simple example:

Example

public class BankAccount {

private double balance;

public double getBalance() {

return balance; // Getter method

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

}

}

}

public class Main {

public static void main(String[] args) {

BankAccount account = new BankAccount();

account.deposit(1000);

System.out.println("Balance: " + account.getBalance());

account.withdraw(500);

System.out.println("Balance after withdrawal: " + account.getBalance());

}

}

In this example, encapsulation is achieved by keeping the `balance` attribute private and providing controlled access through `getBalance()`, `deposit()`, and `withdraw()` methods.