Encapsulation

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

Ans: The process of binding data and corresponding methods (behaviour) together into a single unit is called encapsulation in Java. In other words, encapsulation is a programming technique that binds the class members (variables and methods) together and prevents them from being accessed by other classes, thereby we can keep variables and methods safe from outside interference and misuse. If a field is declared private in the class then it cannot be accessed by anyone outside the class and hides the fields within the class. Therefore, Encapsulation is also called data hiding.

2. What are the important features of Encapsulation?

Ans: Encapsulation is a key concept in object-oriented programming (OOP) that involves combining data and methods within a single unit known as a class. Here are some important features of encapsulation:

Data Hiding: Encapsulation hides the internal details of an object from the outside world. This means that the data stored within an object can only be accessed through the methods provided by that object. This helps to prevent accidental modification of data and makes the code more robust.

Abstraction: Encapsulation allows us to focus on the essential features of an object, while hiding the implementation details. This is achieved by defining public methods that provide a simple and consistent interface for interacting with the object.1⁄4

Access Control: Encapsulation allows us to control the access to the data and methods of an object. We can specify which methods and data members are public, private, or protected. Private data and methods can only be accessed from within the class, while protected data and methods can be accessed from within the class and its subclasses.1⁄4

Data Integrity: Encapsulation ensures that the data stored within an object is consistent and valid. This is achieved by enforcing constraints on the data through methods, which prevents the data from being modified in an inconsistent or incorrect manner.1⁄4

Modularity: Encapsulation allows us to create modular code, which makes it easier to manage and maintain complex systems. By encapsulating related data and methods within a single class, we can create a self-contained unit that can be reused and extended in different parts of the code.

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

Ans: Getter and setter methods, also known as accessor and mutator methods, are used in Java to access and modify the private data members of a class, respectively. They provide a layer of abstraction that allows us to control the access to the data and enforce constraints on the data as needed. Here's an example to illustrate how getter and setter methods work:

```
public class Person {

private String name;

private int age;

public String getName() {
```

```
return name;
}
public void setName(String name) {
this.name = name;
}
public int getAge() {
return age;
}
public void setAge(int age) {
if (age > 0 && age < 120) {
this.age = age;
}
else {
System.out.println("Invalid age");
     }
  }
}
```

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

Ans: This keyword refers to the current object or instance of a class. It is used to differentiate between instance variables and parameters or local variables that have the same name, and to pass the current object as a parameter to other methods or constructors. Here's an example to illustrate the use of the this keyword:

```
public class Person {
private String name;
private int age;
public Person(String name, int age) {
this.name = name;
this.age = age;
}
public void printDetails() {
System.out.println("Name: " + this.name);
System.out.println("Age: " + this.age);
}
public void changeName(String name) {
this.name = name;
}
public void changeAge(int age) {
this.age = age;
}
public void callOtherMethod() {
// Pass the current object as a parameter to another method
otherMethod(this);
}
private void otherMethod(Person p) {
// Do something with the Person object passed as a parameter
   }
}
```

5. What is the advantage of Encapsulation?

Ans: There are the following advantages of encapsulation in Java.

They are as follows:

i)The encapsulated code is more flexible and easier to change with new requirements.

ii)It prevents the other classes from accessing the private fields.

iii)Encapsulation allows modifying implemented code without breaking other code who have implemented the code.

iv)It keeps the data and codes safe from external inheritance. Thus, Encapsulation helps to achieve security.

v)It improves the maintainability of the application.

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

Ans: Encapsulation is achieved in Java by declaring the instance variables of a class as private, and providing public getter and setter methods to access and modify the values of these variables. This allows us to control access to the data and ensure that it can only be modified through the methods of the class, rather than directly accessing the instance variables from outside the class.

Here's an example to illustrate how encapsulation can be achieved in Java:

```
public class BankAccount {
private String accountNumber;
private double balance;
public BankAccount(String accountNumber, double balance) {
this.accountNumber = accountNumber;
this.balance = balance;
}
public String getAccountNumber() {
return accountNumber;
}
public void setAccountNumber(String accountNumber) {
this.accountNumber = accountNumber;
}
public double getBalance() {
return balance;
}
public void deposit(double amount) {
balance += amount;
}
public void withdraw(double amount) {
if (amount <= balance) {
balance -= amount;
}
else {
System.out.println("Insufficient funds");
    }
  }
}
```