1. What is Encapsulation in Java? Why it is 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 prevent them from being accessed by other classes, thereby we can keep variables and ~~members~~ methods safe from outside interference and misuse. if a field (variable) is declared private in the class then it cannot be accessed by anyone outside the class and hides the fields within a class. Therefore, encapsulation is also called data hiding.

2. What are the important features of Encapsulation?

Ans Encapsulation is a key concept in OOPs that involves combining data and methods within a single unit known as a class.

1. Data Hiding :- way of restricting the access of our data members by hiding the implementation details. we can use access modifiers to achieve data hiding.
2. Abstraction :- is a process of hiding the implementation details from the user and showing only the functionality to the user. It can be achieved by using

abstract classes, methods, and interfaces.

3 Access control :- encapsulation allow us to control the access to the data and the 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.

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.

```
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(){
    this.age = 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.

```
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 current object as
// a parameter to another method
```

```
        otherMethod(this);
    }

    private void otherMethod(Person p){
        // do something wit the person object
           passed as a parameter
    }
}
```

5. What is the advantage of encapsulation?

Ans
- encapsulated code is more flexible and easy to change with new requirements.
- it prevents the other classes from accessing the private field.
- it keeps the data and code safe from external inheritance.
- improves the maintability of the application

6. How to achieve encapsulation in java ?

Ans encapsulation is achieved in java by declaring the instance variables of a class as private, and public getter and setter methods to access and modify the values of these variables.

```
public class BankAccount {

    private String accountNumber;
    private double balance;

    pri
    public BankAccount(String accountNumber,
                       double balance)
    {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public String getAccountNumber(){
        return this.accountNumber;
    }
```

```
public void setAccountNumber(String accountNumber)
{
        this.accountNumber = accountNumber;
}

pubiic   double   getBalance(){

        return balance;
    }

pubiic  void  deposit(double amount){

        balance += amount;
    }

public void withdraw(double amount){
    if(amount <= balance){

        balance -= amount;
    }else{

        System.out.println("insufficient fund");
    }
}
}
```