1.What is encapsulation in java ? Why it is called data hiding ?
ANSWER:-
Encapsulation is one of the fundamental concepts in object-oriented programming (OOP) that refers to the bundling of data and methods within a class, which provides a level of abstraction and data protection to the data. In Java, encapsulation is implemented using access modifiers such as private, public, and protected.

Private members of a class can only be accessed within the class, whereas public members can be accessed from outside the class. By encapsulating the data within the class, the implementation details are hidden from the outside world, and only the public interface is exposed.

This concept of encapsulating data within a class is also called "data hiding" because the data is hidden from the outside world, and can only be accessed through public methods. This provides an additional layer of security to the data and helps prevent the accidental modification of the data from outside the class. By hiding the implementation details of a class, encapsulation makes it easier to maintain and modify the code without affecting the other parts of the system.

In summary, encapsulation is a mechanism in Java that allows the bundling of data and methods within a class and restricts the access to data from outside the class, which is why it is also known as "data hiding."

2.What are the important features of Encapsulation ?
ANSWER:-
Encapsulation is an essential concept in object-oriented programming that provides a way to bundle data and methods within a class. The key features of encapsulation in Java are:

Data Hiding: Encapsulation helps in hiding the internal details of an object from the outside world. By making the data members private, they are hidden from outside access and can only be accessed through public methods. This ensures the security and integrity of the data and prevents accidental modifications.

Access Control: Encapsulation provides a way to control the access to the data members and methods of a class. By using access modifiers such as public, private, and protected, developers can specify which data members and methods are accessible from outside the class.

Abstraction: Encapsulation helps in achieving abstraction by providing a simplified view of the object. By hiding the internal details, developers can focus on the essential aspects of the object, making it easier to understand and use.

Code Flexibility: Encapsulation helps in making the code more flexible and adaptable. By hiding the implementation details, developers can modify the internal details of the object without affecting the external interface. This makes it easier to maintain and modify the code over time.

Code Reusability: Encapsulation promotes code reusability by providing a modular structure for the code. By bundling data and methods within a class, developers can create reusable code components that can be used in multiple parts of the system.

In summary, Encapsulation provides several important features, such as data hiding, access control, abstraction, code flexibility, and code reusability, that make it an essential concept in object-oriented programming.

3.What are the getter and setter methods in java Explain with an example ?
ANSWER:-

In Java, getter and setter methods are used to access and modify the private data members of a class, respectively. Getter methods are used to retrieve the value of a private data member, and setter methods are used to set the value of a private data member.

Here's an example of getter and setter methods in Java:

```
public class Person {
  private String name;
  private int age;

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

  // Setter method for name
  public void setName(String newName) {
    name = newName;
  }

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

  // Setter method for age
  public void setAge(int newAge) {
    age = newAge;
  }
}
```

In the example above, the Person class has two private data members, name and age. To access these data members, getter and setter methods have been defined for each of them.

The getter methods, getName() and getAge(), are used to retrieve the values of the private data members name and age, respectively.

The setter methods, setName(String newName) and setAge(int newAge), are used to set the values of the private data members name and age, respectively.

For example, if you want to get the name of a Person object, you can call the getName() method like this:

```
Person p = new Person();
String name = p.getName();
```

If you want to set the age of the same Person object, you can call the setAge(int newAge) method like this:

```
p.setAge(30);
```

Getter and setter methods provide a way to access and modify the private data members of a class while maintaining the encapsulation of the data. This helps in maintaining the integrity and security of the data and allows for more flexibility and maintainability in the code.

4.What is the use of the this keyword in java Explain with example ?
ANSWER:-

In Java, the this keyword is a reference variable that refers to the current instance of the class. It can be used to refer to the current object's instance variables or methods, differentiate between the local variables and instance variables, and pass the current object as a parameter to other methods.

Example:-

```
class Student{
    private int age;
    private String name;
    public void setAge(int age){
       this.age=age;
    }
    public int getAge(){
        return age;
    }
    public void setName(String name){
       this.name=name;
    }
    public String getName(){
        return name;
    }
    public void show(){
        System.out.println(age+" "+name);
    }
}
```

```
 class Encapsu {
    public static void main(String[] args) {
        Student obj=new Student();
        Student obj1=new Student();
        obj.setAge(18);
        obj1.setAge(26);
        obj.setName("raju");
        obj1.setName("raja");
        int stud1Age = obj.getAge();
        String stud1Name = obj.getName();
        System.out.println(stud1Age);
        System.out.println(stud1Name);
        int stud2Age = obj1.getAge();
        String stud2Name = obj1.getName();
        System.out.println(stud2Age);
        System.out.println(stud2Name);
    }
}
```

Output:-
18
Raju
26
raja

5.What is the advantage of Encapsulation ?
ANSWER:-
Encapsulation is one of the core concepts of object-oriented programming and provides several advantages, including:

1.Data Hiding: Encapsulation provides data hiding by hiding the implementation details of a class from other classes. This means that the private data members of a class cannot be accessed directly by other classes, which prevents unauthorized access to the data and ensures data integrity.

2.Modularity: Encapsulation promotes modularity by allowing a class to be developed and tested independently of other classes. This means that a change in one class does not affect other classes, which makes the code more manageable and easier to maintain.

3.Flexibility: Encapsulation provides flexibility by allowing the internal representation of an object to be changed without affecting other parts of the code. This means that changes can be made to the implementation of a class without affecting the overall behavior of the program.

4.Reusability: Encapsulation promotes reusability by making it easier to use existing code in new projects. This means that classes can be used in different contexts without modification, which saves time and reduces the risk of introducing errors.

5.Polymorphism: Encapsulation provides polymorphism by allowing different classes to implement the same interface or inherit from the same superclass. This means that objects of different classes can be treated as if they were of the same type, which increases the flexibility and extensibility of the code.

6.How to achieve encapsulation in java ?Give an example ?
ANSWER:-
Encapsulation in Java can be achieved through the use of access modifiers such as private, protected, and public. The private access modifier makes a member accessible only within the class, while the protected access modifier makes a member accessible within the class and its subclasses. The public access modifier makes a member accessible from any class.

Here's an example of encapsulation in Java:

```
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) {
        this.age = age;
    }
}
```

In the above example, the Person class has two private instance variables name and age. These variables are not directly accessible from other classes, which ensures data hiding and prevents unauthorized access to the data.

To access these variables, public getter and setter methods are provided. The getter methods getName() and getAge() return the values of the corresponding private variables, while the setter methods setName(String name) and setAge(int age) set the values of the corresponding private variables.

By providing public access to these private variables through the getter and setter methods, encapsulation is achieved. The internal implementation details of the Person class are hidden from other classes, and the integrity of the data is ensured.