1. **Ans.**	**What is Encapsulation in java? Why is it called Data hiding?** Encapsulation is a mechanism that combines data and methods into a single unit called a class. It involves bundling data (variables) and the operations (methods) that can be performed on that data, within a class. The purpose of encapsulation is to hide the internal details of the object and provide a public interface through which other parts of the program can interact with the object. Encapsulation is often referred to as "data hiding" because it allows you to hide the internal state and implementation details of an object from the outside world. This means that the data members of a class are declared as private, preventing direct access from other classes. Access to these private members is provided through public methods, known as getters and setters, which allow controlled access and manipulation of the data.
2. **Ans.**	**What are the important features of Encapsulation?** The important features of encapsulation in Java are: 1. **Data Hiding**: Encapsulation allows you to hide the internal state and implementation details of an object from the outside world 2. **Access Modifiers**: Access modifiers such as private, public, protected, and default are used to control the visibility and accessibility of the class members. Encapsulation typically involves using private access modifiers for data members and public access modifiers for getter and setter methods. 3. **Getter and Setter Methods**: Encapsulation provides public methods (getters and setters) to access and modify the private data members of a class. These methods allow controlled access to the data, ensuring data integrity and providing a level of abstraction between the internal state and the external world. 4. **Data Validation and Manipulation**: With encapsulation, you can add validation and manipulation logic within the setter methods. This allows you to enforce constraints, perform checks, and ensure that the data being set is valid and consistent. 5. **Flexibility and Maintainability**: Encapsulation provides flexibility by allowing you to change the internal implementation of a class without affecting the external code that uses the class, as long as the public interface remains unchanged. This makes your code more maintainable, as you can update or optimize the internal details of a class without impacting the rest of the program.

6. **Code Reusability**: Encapsulation supports code reusability by encapsulating related data and behaviour into a single unit (class). This allows you to create objects based on the class and reuse them in different parts of the program or in different programs altogether.

7. **Security**: Encapsulation helps in enhancing the security of the program. By hiding the internal details of the class, it prevents unauthorized access and manipulation of data, ensuring that only the intended methods can interact with the object.

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

Ans. In Java, getter and setter methods are used to provide controlled access to the private data members of a class. Getters are used to retrieve the values of private variables, while setters are used to set or modify the values of private variables. This allows you to enforce encapsulation and maintain control over how the data is accessed and manipulated.

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

   // Getter for name
   public String getName() {
      return name;
   }
   // Setter for name
   public void setName(String newName) {
      name = newName;
   }
   // Getter for age
   public int getAge() {
      return age;
   }

   // Setter for age
   public void setAge(int newAge) {
      if (newAge >= 0) {
         age = newAge;
      } else {
         System.out.println("Invalid age!");
      }
   }
}
```

4.	**What is the use ‘this’ keywords explain with an example?**
Ans.	In Java, the **‘this’** keyword is a reference to the current object within a method or constructor. It is primarily used to differentiate between class members (variables or methods) and local variables or parameters that have the same name. The **‘this**‘ keyword allows you to access or refer to the instance variables and methods of the current object.

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

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

    public void changeName(String newName) {
        this.name = newName;
    }
}
```

5.	**What is the advantage of Encapsulation?**
Ans.	Encapsulation in Java offers several advantages, including: 1. **Data Protection**: Encapsulation provides a way to protect the internal state of an object by keeping the data private and hidden from external access. This prevents unauthorized modification or access to the object's data, maintaining data integrity and security. 2. **Control over Data Access**: By encapsulating data within a class and providing public methods (getters and setters) to access and modify that data, you can control how the data is accessed and manipulated. This allows for validation, data consistency checks, and additional logic to be implemented within the class, ensuring that the data remains valid and consistent. 3. **Code Flexibility**: Encapsulation promotes modularity and separation of concerns. By hiding the internal implementation details of a class, you can change or update the internal workings without affecting the external code that uses the

class, as long as the public interface remains the same. This makes your code more flexible, maintainable, and less prone to bugs or unintended side effects.

4. **Code Reusability**: Encapsulation supports code reusability by creating self-contained objects that can be easily used in different parts of the program or even in other programs altogether. Encapsulated classes with well-defined interfaces can be used as building blocks in larger systems, reducing code duplication and promoting efficient development.

5. **Improved Debugging and Testing**: Encapsulation helps in isolating the behaviour and data of a class, making it easier to debug and test. Since the internal implementation details are hidden, debugging efforts can focus on the specific class or method in question, without being concerned about the entire system. It also allows for easier unit testing, as the class's behaviour can be tested independently of its dependencies.

6. **Enhanced Code Maintainability**: Encapsulation contributes to code maintainability by encapsulating related data and behaviour into a single unit (class). This makes the code more organized, readable, and easier to understand.

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

Ans. To achieve encapsulation in Java, you can follow these steps:

1. Declare the data members of a class as private: By declaring the instance variables of a class as private, you restrict direct access to those variables from outside the class. This ensures that the internal state of the object is hidden and cannot be modified without going through controlled access methods.
2. Provide public getter and setter methods: To allow controlled access to the private variables, provide public methods known as getter and setter methods. Getter methods retrieve the value of a variable, while setter methods modify the value of a variable. These methods should be public and can be used to interact with the private variables.
3.

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

    public String getName() {
        return name;
    }

    public void setName(String newName) {
        name = newName;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int newAge) {
        age = newAge;
    }
}
```