1. **Difference between while and do while loop?**

In Java, a `while` loop evaluates the condition first and then executes the block of code if the condition is true. Conversely, a `do-while` loop executes the block of code first and then evaluates the condition, ensuring that the block of code is executed at least once, regardless of the condition's initial state.

1. **What are the different concepts of OOP?**

1. Encapsulation: Encapsulation is the bundling of data and methods that operate on the data into a single unit. It helps in hiding the internal state of an object and only allows access through controlled interfaces.

2. Inheritance: Inheritance is the mechanism by which a new class is derived from an existing class. It allows a class to inherit properties and behavior (methods) from another class, enabling code reuse and hierarchical classification.

3. Polymorphism: Polymorphism allows objects of different classes to be treated as objects of a common superclass. It enables methods to be invoked on objects without knowing their exact type, facilitating flexibility and extensibility in code design.

4. Abstraction: Abstraction is the process of hiding the implementation details and showing only the essential features of an object. It allows developers to focus on what an object does rather than how it does it, simplifying the complexity of systems and improving readability.

1. **Difference between function overloading vs function overriding?**

Method overloading occurs when a class has multiple methods with the same name but different parameters within the same class. It allows methods to perform similar tasks with different input.

Method overriding happens when a subclass provides a specific implementation of a method that is already provided by one of its parent classes. It enables a subclass to provide a specialized implementation of a method that is already defined in its superclass.

1. **Example of function overloading vs function overriding?**

public class Calculator {

// Method to add two integers

public int add(int a, int b) {

return a + b;

}

// Overloaded method to add three integers

public int add(int a, int b, int c) {

return a + b + c;

}

// Overloaded method to add two doubles

public double add(double a, double b) {

return a + b;

}

}

public class Animal {

public void makeSound() {

System.out.println("Animal makes a sound");

}

}

public class Dog extends Animal {

@Override

public void makeSound() {

System.out.println("Dog barks");

}

}

public class Cat extends Animal {

@Override

public void makeSound() {

System.out.println("Cat meows");

}

}

1. **types of inheritence in java and which inheritance is implemented using interface**

In Java, the types of inheritance include:

1. Single Inheritance: A subclass inherits from only one superclass.

2. Multilevel Inheritance: A subclass inherits from another subclass, creating a hierarchy of classes.

3. Hierarchical Inheritance: Multiple subclasses inherit from a single superclass.

4. Multiple Inheritance (not directly supported in Java): A subclass inherits from more than one superclass.

In Java, the inheritance implemented using interfaces is known as \*\*Interface Inheritance\*\*. In this type of inheritance, a class can implement multiple interfaces, inheriting the method signatures defined in those interfaces. However, it's important to note that interfaces only define method signatures, not method implementations.

**6. What are the access modifiers in Java and explain their visibility levels?**

public: Accessible from any class or package.

protected: Accessible from the same package or subclasses, even if they are in different packages.

default: Accessible only within the same package.

private: Accessible only within the same class.

**7. What is the static keyword in Java? How is it used?**

In Java, the static keyword is used to declare members (variables and methods) that belong to the class itself, rather than to instances of the class. Static members are shared among all instances of the class and can be accessed directly using the class name.

Static variables: Shared among all instances of the class. There is only one copy of a static variable per class.

Static methods: Belong to the class rather than to any instance. They can be called without creating an instance of the class.

public class Example {

static int count = 0; // static variable

static void incrementCount() { // static method

count++;

}

public static void main(String[] args) {

Example.incrementCount();

System.out.println(Example.count); // Output: 1

}

}