Explain single, multilevel, and hierarchical inheritance. ?

Single Inheritance :

Single inheritance means one child class inherits from only one parent class.

- Simple and easy to understand
- Code reuse from one parent class
- Most commonly used inheritance type

Multilevel Inheritance

Multilevel inheritance means a class is derived from another derived class.

Grandparent → Parent → Child

- Inheritance occurs in levels
- Child class gets properties of **a**ll its ancestors
- Promotes high code reusability

Hierarchical Inheritance

Hierarchical inheritance means multiple child classes inherit from the same parent class.

Relationship: 1 Parent → Multiple Children

- One parent class shared by many child classes
- Reduces code duplication
- Commonly used in real-world applications

What is polymorphism?

Polymorphism in Java means one name, many forms.

Types of Polymorphism in Java

Compile-time Polymorphism (Method Overloading)

- Same method name
- Different parameters (number or type)
- Compile-time decision
- Inheritance not required

Runtime Polymorphism (Method Overriding)

- Determined at runtime
- Inheritance is must
- Achieved using method overriding
- Same method name in parent and child
- Child overrides parent method
- Decision made at runtime

Difference between compile-time and runtime polymorphism.

Compile Time polymorphism

- Achieved using method overloading
- Same method name, different parameters
- Also called**:** Static Polymorphism
- Decision time**:** Compile-time
- How achieved**:** Method Overloading (or operator overloading)
- Parameters**:** Must differ (number, type, or order)
- Return type: Can be different, but cannot be the only difference

Run Time polymorphism :

- Achieved using method overriding.
- Child class provides specific implementation of parent method
- Also called: Dynamic Polymorphism
- Decision time: Runtime
- How achieved: Method Overriding
- Parameters: Must be the same as parent method
- Return type: Must be same OR covariant (child class type)
- Inheritance required? Yes, parent-child relationship

What is method overloading?

Method overloading is a feature of Java that allows a class to have multiple methods with the same name but different parameter lists. It is an example of compile-time polymorphism. Method name must be same.Parameters must be different (number, type, or order).Return type can be different, but not the only difference. Can occur within the same class. Inheritance is not required.

What is method overriding?

Method overriding occurs when a child (subclass) provides its own implementation of a method that is already defined in its parent (superclass). The method in the child class replaces the parent class method at runtime. Return type must be same or covariant. Must follow IS-A relationship (inheritance required). Access modifier cannot be more restrictive. Static, final, and private methods cannot be overridden.

==

Prime Number

```
import java.util.Scanner;

public class ForPrime{
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);

		int n, cnt=0;
		System.out.println("Enter Value of n");
		n =sc.nextInt();
		for(int i =1; i<=n;i++){
			if(n % i == 0)
				cnt++;
			}
			if(cnt == 2)
				System.out.println("This is Prime ");
			else
```

```
                                        System.out.println("This is not Prime");
            }
}
```

Multiplication Table of Number

```
import java.util.*;
import java.lang.*;

public class Multiplication{
            public static void main(String[] args){
                        Scanner sc = new Scanner(System.in);
                        int a, b;
                        System.out.println("Enter Values of A");
                        a = sc.nextInt();
                        System.out.println("Multiplication Table of "+a+"  Is :");
                        for(int i =1;i<=10;i++)
                        System.out.println(a*i);
            }
}
```

Prime Upto 1 TO 100

```
public class PrimeNumbers {
    public static void main(String[] args) {

        System.out.println("Prime numbers from 1 to 100:");

        for (int num = 2; num <= 100; num++) {
            boolean isPrime = true;

            for (int i = 2; i <= num / 2; i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }

            if (isPrime) {
                System.out.print(num + " ");
            }
        }
    }
}
```

Sum Of Digits

```
import java.util.Scanner;

public class SumOfDigits {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        int sum = 0;
        while (num != 0) {
            int digit = num % 10;
            sum = sum + digit;
            num = num / 10;
        }

        System.out.println("Sum of digits = " + sum);
    }
}
```

Write a program to count the number of digits in a number.

```
import java.util.Scanner;

public class CountDigits {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        int count = 0;

        // Special case when number is 0
        if (num == 0) {
            count = 1;
        } else {
            while (num != 0) {
                count++;
                num = num / 10;
            }
        }

        System.out.println("Number of digits = " + count);
    }
}
```