Part A:

Q.1 Explain single, multilevel, and hierarchical inheritance.

Inheritance is an OOP (Object-Oriented Programming) concept where one class acquires the properties and methods of another class**.**
It helps in code reuse, maintainability, and extensibility**.**

Single Inheritance:

When one child class inherits from only one parent class, it is called Single Inheritance.

```
class Animal {
   void eat() {
      System.out.println("Eating...");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Barking...");
   }
}
```

Multilevel Inheritance:

When a class is derived from another derived class, it is called Multilevel Inheritance**.**

```
class Animal {
   void eat() {
      System.out.println("Eating...");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Barking...");
   }
}

class Puppy extends Dog {
   void play() {
      System.out.println("Playing...");
   }
}
```

Hierarchical Inheritance:

When multiple child classes inherit from a single parent class, it is called Hierarchical Inheritance.

```
class Animal {
   void eat() {
      System.out.println("Eating...");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Barking...");
   }
}

class Cat extends Animal {
   void meow() {
      System.out.println("Meowing...");
   }
}
```

Java doesn't support multiple inheritance using classes.

We can achieve multiple inheritance using classes.

Q.3 What is polymorphism?

Polymorphism means "many forms".

In Object-Oriented Programming, polymorphism allows one method or object to behave differently in different situations.

Here we can one name many behaviours.

Types of Polymorphism in Java:

1.compile time polymorphism.(method overloading)

2.Run time polymorphism.(method overriding)

Q.3 Difference between compile-time and runtime polymorphism.

Feature	**Compile-Time Polymorphism**	**Run-Time Polymorphism**
Also called	Static polymorphism	Dynamic polymorphism
Achieved by	Method Overloading	Method Overriding
Method binding At	compile time	runtime
Inheritance required	No	Yes

Method signature	Same name, different parameters	Same name, same parameters
Decision made by	Compiler	JVM
Performance	Faster	Slightly slower

Q.4 What is method overloading?

Method overloading:

When multiple methods have the same name but different parameters in the same class.

```
class Calculator {

   int add(int a, int b) {
      return a + b;
   }

   double add(double a, double b) {
      return a + b;
   }
}
```

Q.5 What is method overriding?

Method overriding:

When a child class provides a specific implementation of a method already defined in the parent class.

```
class Animal {
   void sound() {
      System.out.println("Animal makes a sound");
   }
}

class Dog extends Animal {
   void sound() {
      System.out.println("Dog barks");
   }
}

public class Test {
   public static void main(String[] args) {
      Animal a = new Dog();
      a.sound();       }
}
```

Part B:

Q.1 Write a program to find the sum of digits of a number.

```
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 number = sc.nextInt();

        int sum = 0;

        while (number != 0) {
            sum = sum + (number % 10);
            number = number / 10;
        }

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

Q.2 Write a program to check if a number is prime

```
import java.util.Scanner;

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

        Scanner sc = new Scanner(System.in);

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

        boolean isPrime = true;

        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= num / 2; i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
```

```
        }

        if (isPrime)
            System.out.println(num + " is a Prime Number");
        else
            System.out.println(num + " is NOT a Prime Number");
    }
}
```

Q.3 Write a program to print all prime numbers between 1 and 100.

```
public class PrimeNumbers1To100 {
    public static void main(String[] args) {

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

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

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

Q.4 Write a program to print multiplication table of a number.

```
import java.util.Scanner;

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

        Scanner sc = new Scanner(System.in);

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

        for (int i = 1; i <= 10; i++) {
            System.out.println(num + " x " + i + " = " + (num * i));
        }
    }
}
```

Q.5 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 number = sc.nextInt();

        int count = 0;

        if (number == 0) {
            count = 1;
        } else {
            while (number != 0) {
                count++;
                number = number / 10;
            }
        }

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