Java Workshop Lab program file

1. Write any ten features of java

- **Object-Oriented**: Java is based on the object-oriented programming (OOP) paradigm, which organizes software design around data (objects) and methods.

- **Platform Independent**: Java code is compiled into bytecode, which can be run on any platform using the Java Virtual Machine (JVM), making it platform-independent.

- **Simple**: Java is designed to be easy to learn and use, especially for developers familiar with C or C++.

- **Secure**: Java provides built-in security features like bytecode verification, exception handling, and a robust security manager.

- **Robust**: Java emphasizes early error checking, memory management (via garbage collection), and exception handling to build reliable applications.

- **Multithreaded**: Java supports multithreading, allowing concurrent execution of two or more threads for maximum CPU utilization.

- **Architecture Neutral**: Java code is compiled into a format (bytecode) that is architecture-neutral and can run on any system with a JVM.

- **High Performance**: Though interpreted, Java offers high performance through Just-In-Time (JIT) compilers.

- **Distributed**: Java has features (like RMI and sockets) that make it easy to develop distributed applications over networks.

- **Dynamic**: Java programs carry a large amount of run-time information that can be used to verify and resolve accesses to objects during execution.

2. What is Unicode and Byte Code?

Unicode

Definition:
Unicode is a universal character encoding standard that assigns a unique number (code point) to every character in every language, including symbols and emojis.

In Java:

- Java uses **Unicode** to represent characters, allowing it to support multiple languages and scripts.
- A **char** in Java is a 16-bit data type that can represent any Unicode character.
- This makes Java **internationalization-friendly**, i.e., suitable for global applications.

Example

```
char ch = '\u0905'; // Unicode for the Hindi letter 'अ'
System.out.println(ch);
```

Bytecode

Definition:
Bytecode is the intermediate, platform-independent code generated by the Java compiler (.class files) after compiling Java source code (.java files).

In Java:

- Java source code is compiled into bytecode by the **Java Compiler (javac)**.
- This bytecode is then executed by the **Java Virtual Machine (JVM)** on any platform.
- This approach enables Java's **"Write Once, Run Anywhere"** capability.

Example:

bash

CopyEdit

```
javac HelloWorld.java   # compiles to HelloWorld.class (bytecode)
java HelloWorld         # JVM runs the bytecode
```

2. How java is platform independent?

Java is considered platform independent because of the way it executes programs. Here's how it works:

Explanation:

1. **Java Source Code (.java)**
   You write your program in Java.

2. **Compilation into Bytecode (.class)**
   The Java compiler (javac) converts the source code into bytecode, which is a platform-independent intermediate code.
3. **Execution by JVM (Java Virtual Machine)**
   The JVM (specific to each operating system) interprets or compiles the bytecode into machine code for the host OS.
   "Write Once, Run Anywhere"
   Because the bytecode is the same regardless of the operating system, and each OS has its own JVM, you can:

- **Write a Java program once**
- **Run it on Windows, macOS, Linux, etc., without changing the code**

Example:

java

CopyEdit

```
// Hello.java
public class Hello {
   public static void main(String[] args) {
      System.out.println("Hello, world!");
   }
}
```

4. What is JDK and JRE? Differentiate?

JDK (Java Development Kit)

- A **software development environment** used for developing Java applications.
- Includes everything in the **JRE**, plus tools like:
  - **Compiler (javac)**
  - **Debugger**
  - **JavaDoc**
  - **JavaFX**
  - Development libraries

JRE (Java Runtime Environment)

- A **software package** that provides the **necessary environment to run** Java applications.

- It includes:
  - **Java Virtual Machine (JVM)**
  - **Core libraries**
  - Other files needed for runtime
- **Does NOT** include development tools like a compiler.

Difference Between JDK and JRE

Feature	**JDK (Java Development Kit)**	**JRE (Java Runtime Environment)**
Purpose	Used to **develop and run** Java programs	Used to **only run** Java programs
Includes	JRE + compiler, debugger, tools	JVM + libraries + runtime environment
Contains Compiler?	Yes (javac)	No
Suitable for	Developers	End-users or those running Java apps
File size	Larger	Smaller

5. What is the task of Class Loader, Verifier, JIT Compiler in JRE.

In the **Java Runtime Environment (JRE)**, these three components play key roles in **executing Java bytecode** efficiently and securely.

1. Class Loader

Task:

- Loads .class files (bytecode) into **JVM memory** when needed.
- It follows the **lazy loading** principle (loads classes only when required).

Functions:

- Loads classes **dynamically** at runtime.
- Maintains **namespace** for classes.

- Avoids reloading of classes already loaded.

Types of Class Loaders:

- **Bootstrap ClassLoader** – loads core Java classes (java.lang, etc.)
- **Extension ClassLoader** – loads classes from ext directory
- **Application ClassLoader** – loads user-defined classes from the classpath

2. Bytecode Verifier

Task:

- **Validates** the bytecode before execution to ensure it's safe and follows Java's security constraints.

Functions:

- Checks for **illegal code** (e.g., stack overflow, memory access violations).
- Ensures code follows **Java language rules** (e.g., correct method calls, access control).
- Protects the JVM from **malicious or corrupted** code.

3. JIT Compiler (Just-In-Time Compiler)

Task:

- Converts **frequently used** bytecode into **native machine code** during runtime for better performance.

Functions:

- Improves execution speed by reducing interpretation overhead.
- Translates only "hot code" (methods used often).
- Works **with JVM** to dynamically optimize performance.

6. Write/compile/execute Hello World program in Java

Java Data types, Operators and IF - ELSE

Part 1: Write, Compile & Execute a "Hello World" Program in Java

Java Code: HelloWorld.java

java

Code:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

How to Compile & Run

1. Save the file as HelloWorld.java.
2. Open a terminal or command prompt.
3. Navigate to the directory where the file is saved.
4. Compile the program:

   nginx

   code:

   ```
   javac HelloWorld.java
   ```

5. Run the compiled class:

   nginx

   Code:

   ```
   java HelloWorld
   ```

Output:

```
Hello, World!
```

Part 2: Java Data Types, Operators, and if-else Example

Here's a program that uses data types, operators, and an if-else statement:

Java Code: VoteCheck.java

java

Code:

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

```
        // Data Types
        String name = "Alice";
        int age = 20;
        boolean isCitizen = true;

        // Operators and if-else statement
        if (age >= 18 && isCitizen) {
            System.out.println(name + " is eligible to vote.");
        } else {
            System.out.println(name + " is NOT eligible to vote.");
        }
    }
}
```

7. Program to find area and circumference of circle

Java program to calculate the **area** and **circumference** of a circle using a given radius:

java

Code:

```
public class CircleCalculator {
    public static void main(String[] args) {
        double radius = 7.5;  // radius of the circle

        // Calculate area
        double area = Math.PI * radius * radius;

        // Calculate circumference
        double circumference = 2 * Math.PI * radius;

        // Display results
        System.out.println("Radius of the circle: " + radius);

        System.out.println("Area of the circle: " + area);

        System.out.println("Circumference of the circle: " + circumference);

    }

}
```

8. Program to calculate sum of 5 subjects &amp; find percentage.

Java program that calculates the sum of marks in 5 subjects and finds the percentage:

java

Code:

```
public class MarksCalculator {

public static void main(String[] args) {

// Marks obtained by Neeraj in 5 subjects

int subject1 = 85;

int subject2 = 78;

int subject3 = 92;

int subject4 = 88;

int subject5 = 76;


// Calculate sum

int sum = subject1 + subject2 + subject3 + subject4 + subject5;
// Calculate percentage
// Assuming each subject is out of 100 marks
double percentage = (sum / 500.0) * 100;

// Display results
System.out.println("Total marks obtained: " + sum);
System.out.println("Percentage: " + percentage + "%");
}
}
```

9. Program to find the simple interest

Java program to calculate **Simple Interest**:

java

Code:

```
public class SimpleInterestCalculator {
    public static void main(String[] args) {
        double principal = 10000;   // Principal amount
        double rate = 5;            // Rate of interest (in %)
        double time = 3;            // Time period (in years)

        // Calculate simple interest
        double simpleInterest = (principal * rate * time) / 100;

        // Display result
        System.out.println("Principal Amount: " + principal);
        System.out.println("Rate of Interest: " + rate + "%");
        System.out.println("Time Period: " + time + " years");
        System.out.println("Simple Interest: " + simpleInterest);
    }
}
```

10. Write a program which accepts days(eg. 670 days) as integer and display total number

of years, months and days in it.

Java program that accepts an integer number of days (for example, 670 days) and converts it into years, months, and days:

java

Code:

```
import java.util.Scanner;


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


        // Input: total number of days
        System.out.print("Enter number of days: ");
        int totalDays = scanner.nextInt();


        // Assuming:
        // 1 year = 365 days
        // 1 month = 30 days (approximate)


        int years = totalDays / 365;
        int remainingDaysAfterYears = totalDays % 365;


        int months = remainingDaysAfterYears / 30;
        int days = remainingDaysAfterYears % 30;


        System.out.println(totalDays + " days is approximately: ");
        System.out.println(years + " years");
        System.out.println(months + " months");
        System.out.println(days + " days");


        scanner.close();
    }
}
```

11. Program to convert temperature from Fahrenheit to Celsius as C= 5*(f-32)/9

Java program to convert temperature from Fahrenheit to Celsius using the formula:

C = 5 * (F - 32) / 9

java

Code:

```
import java.util.Scanner;


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


        // Input: Temperature in Fahrenheit
        System.out.print("Enter temperature in Fahrenheit: ");
        double fahrenheit = scanner.nextDouble();


        // Convert to Celsius
        double celsius = 5 * (fahrenheit - 32) / 9;
        // Display result
        System.out.println("Temperature in Celsius: " + celsius);

        scanner.close();
    }
}
```

Output

```
Enter temperature in Fahrenheit: 98.6
Temperature in Celsius: 37.0
```

12. Program to swap two no’s without using third variable.

Java program to swap two numbers without using a third (temporary) variable:

java

Code:

```
public class SwapNumbers {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        System.out.println("Before swapping:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);

        // Swap without third variable
        a = a + b;  // a now becomes 30
        b = a - b;  // b becomes 10 (30 - 20)
        a = a - b;  // a becomes 20 (30 - 10)
        System.out.println("After swapping:");

        System.out.println("a = " + a);

        System.out.println("b = " + b);

    }

}
```

Output:

```
Before swapping:

a = 10

b = 20

After swapping:

a = 20

b = 10
```

13. Find the result of following (accept values for variables used in right side of

expression)

a. y=x 2 +3x-7 (print value of Z)

b. y=x++ + ++x (print value of x and y)

c. z= x++ - --y - --x + x++ (print value of x ,y and z)

d. z = x &amp;&amp; y || !(x||y) (print value of Z)

```
import java.util.Scanner;


public class ExpressionSet {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
```

```
// Student Details
String name = "Neeraj";
String enrollNo = "0873CS231073";

System.out.println("Name: " + name);
System.out.println("Enrollment No: " + enrollNo);
System.out.println("--- Expression Evaluations ---");

// a. y = x^2 + 3x - 7 (print Z)
System.out.println("\na. Evaluate y = x^2 + 3x - 7");
System.out.print("Enter value for x: ");
int x = scanner.nextInt();
int z = x * x + 3 * x - 7;
System.out.println("Result (z): " + z);

// b. y = x++ + ++x (print x and y)
System.out.println("\nb. Evaluate y = x++ + ++x");
System.out.print("Enter value for x: ");
x = scanner.nextInt();
int y = x++ + ++x;
System.out.println("Final x: " + x);
System.out.println("Result (y): " + y);

// c. z = x++ - --y - --x + x++ (print x, y, z)
System.out.println("\nc. Evaluate z = x++ - --y - --x + x++");
System.out.print("Enter value for x: ");
x = scanner.nextInt();
```

```
        System.out.print("Enter value for y: ");
        y = scanner.nextInt();
        z = x++ - --y - --x + x++;
        System.out.println("Final x: " + x);
        System.out.println("Final y: " + y);
        System.out.println("Result (z): " + z);

        // d. z = x && y || !(x || y) (logical expression)
        System.out.println("\nd. Evaluate z = x && y || !(x || y)");
        System.out.print("Enter boolean value for x (true/false): ");
        boolean boolX = scanner.nextBoolean();
        System.out.print("Enter boolean value for y (true/false): ");
        boolean boolY = scanner.nextBoolean();
        boolean boolZ = (boolX && boolY) || !(boolX || boolY);
        System.out.println("Result (z): " + boolZ);

        scanner.close();
    }
}
```

Output

```
Name: Neeraj
Enrollment No: 0873CS231073
--- Expression Evaluations ---

a. Evaluate y = x^2 + 3x - 7
Enter value for x: 2
Result (z): 7
```

```
b. Evaluate y = x++ + ++x

Enter value for x: 5

Final x: 7

Result (y): 12


c. Evaluate z = x++ - --y - --x + x++

Enter value for x: 5

Enter value for y: 7

Final x: 7

Final y: 6

Result (z): -2


d. Evaluate z = x && y || !(x || y)

Enter boolean value for x (true/false): false

Enter boolean value for y (false)

Result (z): true
```

14. Program to reverse a given number.

```java
import java.util.Scanner;


public class ReverseNumber {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // Student Details

        String name = "Neeraj";

        String enrollNo = "0873CS231073";
```

```
  }



        System.out.println("Name: " + name);

        System.out.println("Enrollment No: " + enrollNo);

        System.out.println("--- Reverse a Number ---");


        // Input

        System.out.print("Enter a number to reverse: ");

        int number = scanner.nextInt();


        int originalNumber = number;

        int reversed = 0;


        // Logic to reverse the number

        while (number != 0) {

            int digit = number % 10;

            reversed = reversed * 10 + digit;

            number /= 10;

        }


        // Output

        System.out.println("Original Number: " + originalNumber);

        System.out.println("Reversed Number: " + reversed);


        scanner.close();

    }

}
```

Output

```
Name: Neeraj
Enrollment No: 0873CS231073
--- Reverse a Number ---
Enter a number to reverse: 12345
Original Number: 12345
Reversed Number: 54321
```

15. In a company an employee is paid as under:

If his basic salary is less than Rs. 1500, then HRA = 10% of basic salary and DA =

90% of basic

salary. If his salary is either equal to or above Rs. 1500, then HRA = Rs. 500 and DA =

98% of

basic salary. If the employee&#39;s salary is input by the user write a program to find his

gross salary.

GS=Basic+DA+HRA

```java
import java.util.Scanner;

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

        // Student Details
        String name = "Neeraj";
String enrollNo = "0873CS231073";

System.out.println("Name: " + name);
System.out.println("Enrollment No: " + enrollNo);
System.out.println("--- Gross Salary Calculation ---");

// Input: Basic Salary
System.out.print("Enter Basic Salary: ");
double basic = scanner.nextDouble();

double hra, da, grossSalary;

// Salary condition logic
if (basic < 1500) {
   hra = 0.10 * basic;
   da = 0.90 * basic;
} else {
   hra = 500;
   da = 0.98 * basic;
}

// Calculate Gross Salary
grossSalary = basic + hra + da;

// Output
System.out.println("Basic Salary: " + basic);
System.out.println("HRA: " + hra);
System.out.println("DA: " + da);
```

```
        System.out.println("Gross Salary: " + grossSalary);


        scanner.close();
    }
}
```

Output

```
Name: Neeraj

Enrollment No: 0873CS231073

--- Gross Salary Calculation ---

Enter Basic Salary: 1200

Basic Salary: 1200.0

HRA: 120.0

DA: 1080.0

Gross Salary: 2400.0
```

Output 2

```
Name: Neeraj

Enrollment No: 0873CS231073

--- Gross Salary Calculation ---

Enter Basic Salary: 2000

Basic Salary: 2000.0

HRA: 500.0

DA: 1960.0

Gross Salary: 4460.0
```

16. Program to find greatest in 3 numbers.

```
import java.util.Scanner;


public class GreatestOfThree {
```

```java
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Student Details
    String name = "Neeraj";
    String enrollNo = "0873CS231073";

    System.out.println("Name: " + name);
    System.out.println("Enrollment No: " + enrollNo);
    System.out.println("--- Find the Greatest of Three Numbers ---");

    // Input three numbers
    System.out.print("Enter first number: ");
    int a = scanner.nextInt();

    System.out.print("Enter second number: ");
    int b = scanner.nextInt();

    System.out.print("Enter third number: ");
    int c = scanner.nextInt();

    int greatest;

    // Logic to find the greatest
    if (a >= b && a >= c) {
        greatest = a;
    } else if (b >= a && b >= c) {
        greatest = b;
        } else {
            greatest = c;
        }

        // Output
        System.out.println("The greatest number is: " + greatest);

        scanner.close();
    }
}
```

Output

```
Name: Neeraj
Enrollment No: 0873CS231073
--- Find the Greatest of Three Numbers ---
Enter first number: 25
Enter second number: 67
Enter third number: 42
The greatest number is: 67
```

17. Program to check that entered year is leap year or not.

```
import java.util.Scanner;

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

        // Student Details
        String name = "Neeraj";
        String enrollNo = "0873CS231073";

        System.out.println("Name: " + name);
        System.out.println("Enrollment No: " + enrollNo);
        System.out.println("--- Leap Year Checker ---");

        // Input: Year
        System.out.print("Enter a year: ");
        int year = scanner.nextInt();

        // Check for leap year
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            System.out.println(year + " is a leap year.");
        } else {
            System.out.println(year + " is not a leap year.");
        }

        scanner.close();
    }
}
```

Output

```
Name: Neeraj
Enrollment No: 0873CS231073
--- Leap Year Checker ---
Enter a year: 2024
2024 is a leap year.
```

18. Accept person age(int), gender(int 1 for male and 0 for female), then check whether

person is

eligible for marriage or not.

Loops and arrays

```
import java.util.Scanner;


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


        String name = "Neeraj";
        String enrollNo = "0873CS231073";


        System.out.println("Name: " + name);
        System.out.println("Enrollment No: " + enrollNo);
        System.out.println("--- Marriage Eligibility Checker ---");


        char choice;


        do {
            // Accept age
            System.out.print("Enter age: ");
            int age = scanner.nextInt();


            // Accept gender (1 for male, 0 for female)
            System.out.print("Enter gender (1 for male, 0 for female): ");
            int gender = scanner.nextInt();

            // Check eligibility
            if (gender == 1) {  // male
                if (age >= 21) {
                    System.out.println("Eligible for marriage.");
                } else {
                    System.out.println("Not eligible for marriage.");
                }
            } else if (gender == 0) {  // female
                if (age >= 18) {
                    System.out.println("Eligible for marriage.");
                } else {
                    System.out.println("Not eligible for marriage.");
                }
            } else {
                System.out.println("Invalid gender input.");
            }

            // Ask user to continue or not
            System.out.print("Do you want to check another person? (y/n): ");
            choice = scanner.next().charAt(0);

        } while (choice == 'y' || choice == 'Y');

        System.out.println("Program ended.");
        scanner.close();
```

```
    }
}
```

Output

```
Name: Neeraj

Enrollment No: 0873CS231073

--- Marriage Eligibility Checker ---

Enter age: 22

Enter gender (1 for male, 0 for female): 1

Eligible for marriage.

Do you want to check another person? (y/n): y

Enter age: 17

Enter gender (1 for male, 0 for female): 0

Not eligible for marriage.

Do you want to check another person? (y/n): n

Program ended.
```

19. Program to print a table of any number.

```
import java.util.Scanner;


public class TablePrinter {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // Accept the number for which table is to be printed

        System.out.print("Enter the number: ");

        int num = scanner.nextInt();


        System.out.println("Multiplication table of " + num + ":");
        // Print table from 1 to 10
        for (int i = 1; i <= 10; i++) {
            System.out.println(num + " x " + i + " = " + (num * i));
        }

        scanner.close();
    }
}
```

Output

```
Enter the number: 7
Multiplication table of 7:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

20. Program to reverse a given number

```
import java.util.Scanner;

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

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

        int reversed = 0;
        int originalNum = num;

        while (num != 0) {
            int digit = num % 10;         // Get the last digit
            reversed = reversed * 10 + digit; // Append digit
            num = num / 10;               // Remove last digit
        }

        System.out.println("Reversed number of " + originalNum + " is: " + reversed);

        scanner.close();
    }
}
```

Output

```
Enter a number to reverse: 12345
Reversed number of 12345 is: 54321
```

21. Program to check whether number is prime or not

```
import java.util.Scanner;

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

        System.out.print("Enter a number to check if it is prime: ");
        int num = scanner.nextInt();

        boolean isPrime = true;

        if (num <= 1) {
            isPrime = false;  // 0 and 1 are not prime numbers
        } else {
            for (int i = 2; i <= Math.sqrt(num); 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.");
        }

        scanner.close();
    }
}
```

Output

```
Enter a number to check if it is prime: 29
29 is a prime number.
```

22. Calculate series : 1 2 +2 2 +3 2 +4 2 +.........+n 2

```
import java.util.Scanner;

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

        System.out.print("Enter the value of n: ");
        int n = scanner.nextInt();

        int sum = 0;

        for (int i = 1; i <= n; i++) {
            sum += i * i;  // add square of i to sum
        }

        System.out.println("Sum of squares from 1 to " + n + " is: " + sum);

        scanner.close();
    }
}
```

Output

```
Enter the value of n: 5
Sum of squares from 1 to 5 is: 55
```

23. Calculate sum of Lucas series (up to 10 terms) :

1, 2, 3, 6, 11, 20 ,........

```
public class LucasSeriesSum {
    public static void main(String[] args) {
        String name = "Neeraj";
        String enrollment = "0873CS231073";

        System.out.println("Name: " + name);
        System.out.println("Enrollment: " + enrollment);

        int n = 10; // number of terms
        int sum = 0;

        // First two terms of Lucas series
        int a = 1;
        int b = 2;

        System.out.print("Lucas series terms: " + a + " " + b + " ");

        sum = a + b;

        for (int i = 3; i <= n; i++) {
            int c = a + b;  // next term is sum of previous two
            System.out.print(c + " ");
            sum += c;
            a = b;
            b = c;
        }
        System.out.println("\nSum of first " + n + " terms: " + sum);
    }
}
```

Output

```
Name: Neeraj
Enrollment: 0873CS231073
Lucas series terms: 1 2 3 5 8 13 21 34 55 89
Sum of first 10 terms: 231
```

24. Print all prime numbers between two given numbers;

```
import java.util.Scanner;

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

        System.out.print("Enter start number: ");
        int start = scanner.nextInt();

        System.out.print("Enter end number: ");
        int end = scanner.nextInt();

        System.out.println("Prime numbers between " + start + " and " + end + " are:");

        for (int num = start; num <= end; num++) {
            if (num > 1 && isPrime(num)) {
                System.out.print(num + " ");
            }
        }
        System.out.println("\n\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        scanner.close();
    }

    public static boolean isPrime(int n) {
        if (n <= 1) return false;

        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) return false;
        }
        return true;
    }
}
```

Output

```
Enter start number: 10
Enter end number: 30
Prime numbers between 10 and 30 are:
11 13 17 19 23 29


Name: Neeraj
Enrollment: 0873CS231073
```

25. Program to show sum and average of 10 element array. Accept array elements from

user.

```java
import java.util.Scanner;

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

        int[] arr = new int[10];
        int sum = 0;
        double average;

        System.out.println("Enter 10 elements:");

        for (int i = 0; i < 10; i++) {
            arr[i] = scanner.nextInt();
            sum += arr[i];
        }

        average = (double) sum / 10;

        System.out.println("Sum of array elements: " + sum);
        System.out.println("Average of array elements: " + average);

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        scanner.close();
    }
}
```

Output

```
Enter 10 elements:
1 2 3 4 5 6 7 8 9 10
Sum of array elements: 55
Average of array elements: 5.5
```

Name: Neeraj

Enrollment: 0873CS231073

26. Sort a ten element array in descending order.

```
import java.util.Scanner;
import java.util.Arrays;
import java.util.Collections;

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

    Integer[] arr = new Integer[10];  // Use Integer for Collections.reverseOrder()

    System.out.println("Enter 10 elements:");

    for (int i = 0; i < 10; i++) {
      arr[i] = scanner.nextInt();
    }

    // Sort in descending order
    Arrays.sort(arr, Collections.reverseOrder());
```

```
        System.out.println("Array sorted in descending order:");

        for (int i = 0; i < 10; i++) {
            System.out.print(arr[i] + " ");
        }

        System.out.println("\n\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        scanner.close();
    }
}
```

Output

```
Enter 10 elements:
34 12 56 78 23 89 90 11 45 67
Array sorted in descending order:
90 89 78 67 56 45 34 23 12 11

Name: Neeraj
Enrollment: 0873CS231073
```

27. Create a array of 17 elements in 5 rows. And calculate sum of all elements.

```
import java.util.Scanner;

public class ArraySum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        }

        int rows = 5;
        int totalElements = 17;
        int cols = (int) Math.ceil((double) totalElements / rows);

        int[][] arr = new int[rows][cols];
        int count = 0;

        System.out.println("Enter 17 elements:");

        // Input 17 elements row-wise
        for (int i = 0; i < rows && count < totalElements; i++) {
            for (int j = 0; j < cols && count < totalElements; j++) {
                arr[i][j] = scanner.nextInt();
                count++;
            }
        }

        // Calculate sum of all 17 elements
        int sum = 0;
        count = 0;
        for (int i = 0; i < rows && count < totalElements; i++) {
            for (int j = 0; j < cols && count < totalElements; j++) {
                sum += arr[i][j];
                count++;
            }
        }

        System.out.println("Sum of all 17 elements = " + sum);
```

```
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        scanner.close();
    }
}
```

Output

```
Enter 17 elements:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Sum of all 17 elements = 153

Name: Neeraj
Enrollment: 0873CS231073
```

28. Program to find multiplication of two 3X3 matrices.

```
public class MatrixMultiplication {
    public static void main(String[] args) {
        int[][] matrix1 = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int[][] matrix2 = {
            {9, 8, 7},
            {6, 5, 4},
            {3, 2, 1}
        };
        int[][] result = new int[3][3];

        // Multiply matrix1 and matrix2
        for (int i = 0; i < 3; i++) {  // rows of matrix1
            for (int j = 0; j < 3; j++) {  // columns of matrix2
                result[i][j] = 0;
                for (int k = 0; k < 3; k++) {  // columns of matrix1 / rows of matrix2
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }

        // Print the result
        System.out.println("Result of multiplication of two 3x3 matrices:");

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(result[i][j] + "\t");
            }
            System.out.println();
        }

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Result of multiplication of two 3x3 matrices:
```

```
30                                  24    18

84                                  69    54

138                                 114   90
```

Name: Neeraj

Enrollment: 0873CS231073

29. Program to print transpose of a matrix.

Object Oriented Programming , Classes, Objects ,Methods, Constructor

```
import java.util.Scanner;


class Matrix {
    int rows, cols;
    int[][] matrix;

    // Constructor to initialize matrix dimensions and allocate memory
    Matrix(int r, int c) {
        rows = r;
        cols = c;
        matrix = new int[rows][cols];
    }

    // Method to accept matrix elements from user
    void acceptMatrix() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter elements of " + rows + "x" + cols + " matrix:");
        for(int i = 0; i < rows; i++) {
            for(int j = 0; j < cols; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
    }

    // Method to print the matrix
    void printMatrix() {
        for(int i = 0; i < rows; i++) {
            for(int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }
    }

    // Method to find and return transpose of the matrix
    Matrix transpose() {
        Matrix transposed = new Matrix(cols, rows);
        for(int i = 0; i < rows; i++) {
            for(int j = 0; j < cols; j++) {
                transposed.matrix[j][i] = matrix[i][j];
            }
        }
        return transposed;
    }
}
```

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

        System.out.print("Enter number of rows: ");
        int r = sc.nextInt();

        System.out.print("Enter number of columns: ");
        int c = sc.nextInt();

        Matrix m = new Matrix(r, c);
        m.acceptMatrix();

        System.out.println("\nOriginal Matrix:");
        m.printMatrix();

        Matrix t = m.transpose();
        System.out.println("\nTranspose of Matrix:");
        t.printMatrix();

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter number of rows: 2
Enter number of columns: 3
Enter elements of 2x3 matrix:
1 2 3
4 5 6
Original Matrix:
1                                   2       3
4                                   5       6

Transpose of Matrix:
1                                   4
2                                   5
3                                   6
```

Name: Neeraj

Enrollment: 0873CS231073

30. Create a class to calculate Area of circle with one data member to store the radius and

another to store area value.

Create method members

1. init - to input radius from user

2. calc - to calculate area

3. display- to display area

```java
import java.util.Scanner;

class Circle {
  double radius;
  double area;
```

```
    // Method to input radius from user
    void init() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter radius of the circle: ");
        radius = sc.nextDouble();
    }

    // Method to calculate area of the circle
    void calc() {
        area = Math.PI * radius * radius;
    }

    // Method to display the area
    void display() {
        System.out.printf("Area of the circle with radius %.2f is: %.2f\n", radius, area);
    }
}

public class CircleArea {
    public static void main(String[] args) {
        Circle c = new Circle();

        c.init();
        c.calc();
        c.display();

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter radius of the circle: 7
Area of the circle with radius 7.00 is: 153.94

Name: Neeraj
Enrollment: 0873CS231073
```

31. Create a class MathOperation with two data member X and Y to store the operand and

third data member R to store result of operation.

Create method members

- **init - to input X and Y from user**
- **add - to add X and Y and store in R**
- **multiply - to multiply X and Y and store in R**
- **power - to calculate X Y and store in R**
- **display- to display Result R**

```
import java.util.Scanner;

class MathOperation {
    double X, Y, R;

    // Method to input X and Y from user
    void init() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter value for X: ");
        X = sc.nextDouble();
        System.out.print("Enter value for Y: ");
        Y = sc.nextDouble();
    }

    // Method to add X and Y
    void add() {
        R = X + Y;
    }

    // Method to multiply X and Y
    void multiply() {
        R = X * Y;
    }

    // Method to calculate X raised to power Y
    void power() {
        R = Math.pow(X, Y);
    }

    // Method to display result R
    void display() {
        System.out.println("Result = " + R);
    }
}
```

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

        mo.init();

        // Add
        mo.add();
        System.out.print("Addition of X and Y: ");
        mo.display();

        // Multiply
        mo.multiply();
        System.out.print("Multiplication of X and Y: ");
        mo.display();

        // Power
        mo.power();
        System.out.print("X raised to power Y: ");
        mo.display();

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter value for X: 2
Enter value for Y: 3

Addition of X and Y: Result = 5.0

Multiplication of X and Y: Result = 6.0

X raised to power Y: Result = 8.0
```

Name: Neeraj

Enrollment: 0873CS231073

32. Create a class MathOperation containing method ‘multiply’ to calculate multiplication

of following arguments.

a. two integers

b. three float

c. all elements of array

d. one double and one integer

```java
import java.util.Scanner;


class MathOperation {


    // Multiply two integers

    int multiply(int a, int b) {

        return a * b;

    }


    // Multiply three floats

    float multiply(float a, float b, float c) {

        return a * b * c;
    }

    // Multiply all elements of an integer array
    int multiply(int[] arr) {
        int result = 1;
        for (int val : arr) {
            result *= val;
        }
        return result;
    }

    // Multiply one double and one integer
    double multiply(double a, int b) {
        return a * b;
    }
}

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

        // a. Multiply two integers
        System.out.print("Enter two integers: ");
        int int1 = sc.nextInt();
        int int2 = sc.nextInt();
        System.out.println("Multiplication of two integers: " + mo.multiply(int1, int2));
        // b. Multiply three floats

        System.out.print("Enter three float values: ");

        float f1 = sc.nextFloat();

        float f2 = sc.nextFloat();

        float f3 = sc.nextFloat();

        System.out.println("Multiplication of three floats: " + mo.multiply(f1, f2, f3));


        // c. Multiply all elements of array

        System.out.print("Enter number of elements in array: ");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter array elements:");

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        System.out.println("Multiplication of array elements: " + mo.multiply(arr));


        // d. Multiply one double and one integer

        System.out.print("Enter a double and an integer: ");

        double d = sc.nextDouble();

        int i = sc.nextInt();

        System.out.println("Multiplication of double and integer: " + mo.multiply(d, i));


        System.out.println("\nName: Neeraj");

        System.out.println("Enrollment: 0873CS231073");

    }

}
```

Output

```
Enter two integers: 3 4

Enter three float values: 1.5 2.0 3.0

Enter number of elements in array: 4

Enter array elements:

1 2 3 4

Enter a double and an integer: 2.5 4

Multiplication of two integers: 12

Multiplication of three floats: 9.0

Multiplication of array elements: 24

Multiplication of double and integer: 10.0
```

Name: Neeraj

Enrollment: 0873CS231073

33. Write a program to find the area and circumference of a circle.

```java
import java.util.Scanner;


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


        System.out.print("Enter radius of the circle: ");
        double radius = sc.nextDouble();


        double area = Math.PI * radius * radius;
        double circumference = 2 * Math.PI * radius;


        System.out.println("Area of the circle: " + area);
        System.out.println("Circumference of the circle: " + circumference);

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter radius of the circle: 7
Area of the circle: 153.93804002589985
Circumference of the circle: 43.982297150257104

Name: Neeraj
Enrollment: 0873CS231073
```

34 Write a program to calculate the sum of 5 subjects and find the percentage.

```
import java.util.Scanner;

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

        int[] marks = new int[5];
        int sum = 0;

        System.out.println("Enter marks for 5 subjects:");

        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
            sum += marks[i];
        }

        double percentage = (sum / 500.0) * 100;

        System.out.println("Total Marks = " + sum);
        System.out.println("Percentage = " + percentage + "%");

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter marks for 5 subjects:
Subject 1: 80
Subject 2: 75
Subject 3: 90
Subject 4: 85
Subject 5: 70
Total Marks = 400
Percentage = 80.0%

Name: Neeraj
Enrollment: 0873CS231073
```

35 Write a program to calculate simple interest.

```
import java.util.Scanner;
```

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

        System.out.print("Enter Principal amount: ");
        double principal = sc.nextDouble();

        System.out.print("Enter Rate of interest (% per annum): ");
        double rate = sc.nextDouble();

        System.out.print("Enter Time (in years): ");
        double time = sc.nextDouble();

        double simpleInterest = (principal * rate * time) / 100;

        System.out.println("\nSimple Interest = " + simpleInterest);

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter Principal amount: 10000
Enter Rate of interest (% per annum): 5
Enter Time (in years): 3
Simple Interest = 1500.0

Name: Neeraj
```

Enrollment: 0873CS231073

36 Write a program that accepts total number of days (e.g., 670 days) and displays the equivalent years, months, and days.

```
import java.util.Scanner;

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

        System.out.print("Enter total number of days: ");
        int totalDays = sc.nextInt();

        int years = totalDays / 365;         // assuming 1 year = 365 days
        int remainingDaysAfterYears = totalDays % 365;

        int months = remainingDaysAfterYears / 30;   // assuming 1 month = 30 days
        int days = remainingDaysAfterYears % 30;

        System.out.println("\nEquivalent Time:");
        System.out.println("Years: " + years);
        System.out.println("Months: " + months);
        System.out.println("Days: " + days);

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter total number of days: 670
Equivalent Time:
Years: 1
Months: 10
Days: 0
```

Name: Neeraj

Enrollment: 0873CS231073

37 . Write a program to convert temperature from Fahrenheit to Celsius using the formula:

C = 5 * (F - 32) / 9

```java
import java.util.Scanner;

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

        System.out.print("Enter temperature in Fahrenheit: ");
        double fahrenheit = sc.nextDouble();

        double celsius = 5 * (fahrenheit - 32) / 9;

        System.out.printf("Temperature in Celsius: %.2f\n", celsius);
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter temperature in Fahrenheit: 98.6
Temperature in Celsius: 37.00

Name: Neeraj
Enrollment: 0873CS231073
```

38 Write a program to swap two numbers without using a third variable.

```
import java.util.Scanner;

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

        System.out.print("Enter first number (x): ");
        int x = sc.nextInt();

        System.out.print("Enter second number (y): ");
        int y = sc.nextInt();

        // Swapping without third variable
        x = x + y;
        y = x - y;
        x = x - y;

        System.out.println("After swapping:");
        System.out.println("x = " + x);
        System.out.println("y = " + y);

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
Enter first number (x): 5
Enter second number (y): 10
After swapping:
x = 10
y = 5

Name: Neeraj
Enrollment: 0873CS231073
```

39 Find the result of the following expressions (assume values for variables on the right-hand side), and print the result:

a. y = x^2 + 3x - 7 (Print value of z)

b. y = x++ + ++x (Print values of x and y)

c. z = x++ - --y - -x + x++ (Print values of x, y, and z)

d. z = x && y || !(x || y) (Print value of z)

```
public class ExpressionResults {
    public static void main(String[] args) {
        // Assume initial values
        int x, y, z;
        boolean bx, by, bz;

        // For part a:
        x = 5;
        y = x * x + 3 * x - 7;
        z = y;  // Since instruction says print value of z for (a), assign y to z
        System.out.println("a) For x = 5, y = x^2 + 3x - 7 => y = " + y);
        System.out.println("Value of z (same as y): " + z);

        // For part b:
        x = 5;
        y = x++ + ++x;
        System.out.println("\nb) After y = x++ + ++x:");
        System.out.println("x = " + x);
        System.out.println("y = " + y);

        // For part c:
        x = 5;
        y = 10;
        z = x++ - --y - --x + x++;
        System.out.println("\nc) After z = x++ - --y - --x + x++:");
        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("z = " + z);


        // For part d:
        // Logical expressions require boolean, so assume
        bx = true;  // x
        by = false; // y
        bz = (bx && by) || !(bx || by);
        System.out.println("\nd) For bx = true, by = false:");
        System.out.println("z = (x && y) || !(x || y) = " + bz);


        // Name and enrollment output
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
    }
}
```

Output

```
a) For x = 5, y = x^2 + 3x - 7 => y = 33
Value of z (same as y): 33


b) After y = x++ + ++x:
x = 7
y = 12


c) After z = x++ - --y - --x + x++:
x = 6
y = 9
z = -4
```

d) For bx = true, by = false:

z = (x && y) || !(x || y) = false

Name: Neeraj

Enrollment: 0873CS231073

40 . Write a program to reverse a given number.

```
import java.util.Scanner;

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

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

      int reversed = 0;
      int originalNum = num;

      while (num != 0) {
         int digit = num % 10;
         reversed = reversed * 10 + digit;
         num /= 10;
      }

      System.out.println("Original number: " + originalNum);
      System.out.println("Reversed number: " + reversed);

      // Print name and enrollment as requested
```

```
        System.out.println("\nName: Neeraj");

        System.out.println("Enrollment: 0873CS231073");


        sc.close();

    }

}
```

Output

```
Enter a number to reverse: 12345

Original number: 12345

Reversed number: 54321


Name: Neeraj

Enrollment: 0873CS231073
```

41 . Program to find greatest in 3 numbers.

```
import java.util.Scanner;


public class GreatestOfThree {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);


        System.out.print("Enter first number: ");

        int num1 = sc.nextInt();


        System.out.print("Enter second number: ");

        int num2 = sc.nextInt();
        System.out.print("Enter third number: ");
        int num3 = sc.nextInt();

        int greatest;

        if (num1 >= num2 && num1 >= num3) {
            greatest = num1;
        } else if (num2 >= num1 && num2 >= num3) {
            greatest = num2;
        } else {
            greatest = num3;
        }

        System.out.println("Greatest number is: " + greatest);


        // Print name and enrollment
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        sc.close();
    }
}
```

Output

```
Enter first number: 25
Enter second number: 40
Enter third number: 35
Greatest number is: 40
```

Name: Neeraj

Enrollment: 0873CS231073

42 Program to print a table of any number.

```
import java.util.Scanner;

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

        System.out.print("Enter the number to print its table: ");
        int num = sc.nextInt();

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

        // Print name and enrollment
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        sc.close();
    }
}
```

Output

```
Enter the number to print its table: 7
Multiplication table of 7:
7 x 1 = 7
```

```
7 x 2 = 14

7 x 3 = 21

7 x 4 = 28

7 x 5 = 35

7 x 6 = 42

7 x 7 = 49

7 x 8 = 56

7 x 9 = 63

7 x 10 = 70
```

Name: Neeraj

Enrollment: 0873CS231073

44 Print all prime numbers between two given numbers;

```
import java.util.Scanner;


public class PrimeInRange {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);


        System.out.print("Enter starting number: ");

        int start = sc.nextInt();


        System.out.print("Enter ending number: ");

        int end = sc.nextInt();


        System.out.println("Prime numbers between " + start + " and " + end + ":");


        for (int i = start; i <= end; i++) {
           if (isPrime(i)) {
              System.out.print(i + " ");
           }
        }

        // Name and Enrollment
        System.out.println("\n\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        sc.close();
    }

    // Helper method to check prime
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
        for (int i = 2; i <= Math.sqrt(num); i++) {
           if (num % i == 0) return false;
        }
        return true;
    }
}
```

Output

```
Enter starting number: 10
Enter ending number: 30
Prime numbers between 10 and 30:
11 13 17 19 23 29

Name: Neeraj
```

Enrollment: 0873CS231073

45 Program to show sum and average of 10 element array. Accept array elements from user.

```
import java.util.Scanner;

public class ArraySumAverage {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] numbers = new int[10];
        int sum = 0;

        System.out.println("Enter 10 integer elements:");

        for (int i = 0; i < 10; i++) {
            System.out.print("Element " + (i + 1) + ": ");
            numbers[i] = sc.nextInt();
            sum += numbers[i];
        }

        double average = sum / 10.0;

        System.out.println("\nSum of elements: " + sum);
        System.out.println("Average of elements: " + average);

        // Name and Enrollment
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");
        sc.close();
    }
}
```

Output

```
Enter 10 integer elements:
Element 1: 12
Element 2: 15
Element 3: 18
Element 4: 11
Element 5: 17
Element 6: 20
Element 7: 14
Element 8: 16
Element 9: 10
Element 10: 13

Sum of elements: 146
Average of elements: 14.6
```

Name: Neeraj

Enrollment: 0873CS231073

46 . Sort a ten element array in descending order.

```
import java.util.Scanner;
import java.util.Arrays;
import java.util.Collections;

public class DescendingSort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Integer[] numbers = new Integer[10];

        System.out.println("Enter 10 integer elements:");

        for (int i = 0; i < 10; i++) {
            System.out.print("Element " + (i + 1) + ": ");
            numbers[i] = sc.nextInt();
        }

        // Sort in descending order using built-in method
        Arrays.sort(numbers, Collections.reverseOrder());

        System.out.println("\nArray elements in descending order:");
        for (int num : numbers) {
            System.out.print(num + " ");
        }

        // Name and Enrollment
        System.out.println("\n\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        sc.close();
    }
}
```

Output

```
Enter 10 integer elements:
Element 1: 23
Element 2: 15
Element 3: 87
Element 4: 42
Element 5: 9
Element 6: 33
Element 7: 50
Element 8: 71
Element 9: 60
Element 10: 11

Array elements in descending order:
87 71 60 50 42 33 23 15 11 9
```

Name: Neeraj

Enrollment: 0873CS231073

```java
import java.util.Scanner;

public class Array2DSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] arr = new int[5][];
        int sum = 0;

        // Define row sizes
        arr[0] = new int[4];
        arr[1] = new int[4];
        arr[2] = new int[3];
        arr[3] = new int[3];
        arr[4] = new int[3];

        System.out.println("Enter 17 integers to fill the 5-row array:");

        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr[i].length; j++) {
                System.out.print("Element [" + i + "][" + j + "]: ");
                arr[i][j] = sc.nextInt();
                sum += arr[i][j];
            }
        }

        System.out.println("\nSum of all 17 elements: " + sum);

        // Display Name and Enrollment
        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        sc.close();
    }
}
```

Output

```
Enter 17 integers to fill the 5-row array:
Element [0][0]: 1
Element [0][1]: 2
Element [0][2]: 3
Element [0][3]: 4
Element [1][0]: 5
Element [1][1]: 6

Element [1][2]: 7

Element [1][3]: 8

Element [2][0]: 9

Element [2][1]: 10

Element [2][2]: 11

Element [3][0]: 12

Element [3][1]: 13

Element [3][2]: 14

Element [4][0]: 15

Element [4][1]: 16

Element [4][2]: 17


Sum of all 17 elements: 200
```

Name: Neeraj

Enrollment: 0873CS231073

47 . Program to find multiplication of two 3X3 matrices.

```
import java.util.Scanner;


public class MatrixMultiplication {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int[][] A = new int[3][3];

        int[][] B = new int[3][3];

        int[][] C = new int[3][3];


        System.out.println("Enter elements of Matrix A (3x3):");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print("A[" + i + "][" + j + "]: ");
                A[i][j] = sc.nextInt();
            }
        }

        System.out.println("Enter elements of Matrix B (3x3):");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print("B[" + i + "][" + j + "]: ");
                B[i][j] = sc.nextInt();
            }
        }

        // Matrix multiplication: C = A * B
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                C[i][j] = 0;
                for (int k = 0; k < 3; k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }

        System.out.println("\nResu
```

Output

```
Enter elements of Matrix A (3x3):
```

```
A[0][0]: 1

A[0][1]: 2

A[0][2]: 3

A[1][0]: 4

A[1][1]: 5

A[1][2]: 6

A[2][0]: 7

A[2][1]: 8

A[2][2]: 9


Enter elements of Matrix B (3x3):

B[0][0]: 9

B[0][1]: 8

B[0][2]: 7

B[1][0]: 6

B[1][1]: 5

B[1][2]: 4

B[2][0]: 3

B[2][1]: 2

B[2][2]
```

48 . Program to print transpose of a matrix.

```
import java.util.Scanner;


public class MatrixTranspose {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int rows, cols;
```

```
System.out.print("Enter number of rows: ");
rows = sc.nextInt();

System.out.print("Enter number of columns: ");
cols = sc.nextInt();

int[][] matrix = new int[rows][cols];
int[][] transpose = new int[cols][rows];

System.out.println("Enter elements of the matrix:");
for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols; j++) {
      System.out.print("Element [" + i + "][" + j + "]: ");
      matrix[i][j] = sc.nextInt();
   }
}

// Transpose logic
for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols; j++) {
      transpose[j][i] = matrix[i][j];
   }
}

// Display original matrix
System.out.println("\nOriginal Matrix:");
for (int i = 0; i < rows; i++) {
```

```
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }

        // Display transposed matrix
        System.out.println("\nTransposed Matrix:");
        for (int i = 0; i < cols; i++) {
            for (int j = 0; j < rows; j++) {
                System.out.print(transpose[i][j] + "\t");
            }
            System.out.println();
        }

        System.out.println("\nName: Neeraj");
        System.out.println("Enrollment: 0873CS231073");

        sc.close();
    }
}
```

Output

```
Enter number of rows: 2
Enter number of columns: 3
Enter elements of the matrix:
Element [0][0]: 1
Element [0][1]: 2
Element [0][2]: 3
Element [1][0]: 4
Element [1][1]: 5
Element [1][2]: 6

Original Matrix:
1                    2    3
4                    5    6

Transposed Matrix:
1                    4
2                    5
3                    6

Name: Neeraj
Enrollment: 0873CS231073
```

49 . What are the two types of Exceptions in Java? What are the differences between them?

In Java, **exceptions** are categorized into **two main types**:

---

1. Checked Exceptions

- **Definition:** Exceptions that are checked at **compile-time**.
- **Requirement:** The programmer must either handle them using try-catch or declare them using the throws keyword.
- **Examples:**
    - IOException
    - SQLException
    - FileNotFoundException

- **When they occur:** Usually when the program interacts with external resources like files, databases, or networks.

2. Unchecked Exceptions

- **Definition:** Exceptions that are not checked at compile-time, but occur at **runtime**.
- **Requirement:** No need to declare or catch them explicitly.
- **Examples:**
    - ArithmeticException
    - NullPointerException
    - ArrayIndexOutOfBoundsException
- **When they occur:** Typically due to logical programming errors.

Differences Between Checked and Unchecked Exceptions

Feature	**Checked Exception**	**Unchecked Exception**
Checked at Compile Time	Yes	No
Handling Requirement	Must be handled with try-catch or throws	Optional
Part of Which Class?	java.lang.Exception (excluding RuntimeException)	java.lang.RuntimeException and its subclasses
Common Usage	External operations (file, DB, network)	Logical errors in code
Example	IOException, SQLException	NullPointerException, ArithmeticException

50 What are the Memory Allocations available in Java?

In Java, memory is allocated in several distinct regions during program execution. These are managed primarily by the **Java Virtual Machine (JVM)**. The main **memory allocations** in Java are:

1. Stack Memory

- **Stores:**
    - Local variables
    - Method call information (stack frames)
    - Function parameters
- **Scope:**
    - Thread-specific (each thread gets its own stack)
- **Allocation/Deallocation:**
    - Automatically managed (LIFO order)
- **Speed:**
    - Very fast
- **Example:**
- int a = 5; // 'a' is stored in stack

2. Heap Memory

- **Stores:**
    - Objects and their instance variables
- **Scope:**
    - Shared across all threads
- **Managed by:**
    - Garbage Collector
- **Usage:**
    - Memory-intensive and long-living data
- **Example:**
- Student s = new Student(); // Object stored in heap

3. Method Area (also called Metaspace in Java 8+)

- **Stores:**
    - Class definitions (metadata)
    - Static variables
    - Method code

- **Scope:**
    - Shared by all classes and threads
- **Managed by:**
    - JVM at runtime

4. Program Counter (PC) Register

- **Stores:**
    - Address of the currently executing instruction for a thread
- **Scope:**
    - Each thread has its own PC register
- **Role:**
    - Helps the JVM know where it is in the program

5. Native Method Stack

- **Stores:**
    - Native (non-Java) method calls (e.g., C/C++ methods invoked via JNI)
- **Scope:**
    - Specific to threads that use native methods

Summary Table

Memory Type	**Stores**	**Scope**	**Managed By**
Stack	Local variables, method calls	Per thread	JVM (auto)
Heap	Objects, instance variables	Shared	Garbage Collector
Method Area	Class info, static data	Shared	JVM
PC Register	Current executing address	Per thread	JVM
Native Method Stack	Native method calls	Per thread	OS/JVM

51. Explain final, finally, finalize.

In Java, **final**, **finally**, and **finalize()** are three completely different terms that sound similar but serve **distinct purposes**. Here's a clear explanation of each:

1. final — Keyword

Used to define **constants**, **prevent inheritance**, or **stop method overriding**.

Usage:

- **Final Variable** – Value cannot be changed once assigned.
- **Final Method** – Cannot be overridden in a subclass.
- **Final Class** – Cannot be extended.

Examples:

```
final int x = 10; // cannot reassign x


final class A {
    // cannot be extended
}


class B {
    final void show() {
        System.out.println("This method cannot be overridden");
    }
}
```

2. finally — Block

Used with **try-catch** to execute code **regardless of exception occurrence**.

Purpose:

- Used to **release resources** like files, sockets, or database connections.

Example:

```
try {
    int result = 10 / 2;
} catch (ArithmeticException e) {
    System.out.println("Exception caught");
} finally {
    System.out.println("This block always executes");
}
```

3. finalize() — Method

A method defined in java.lang.Object class, called by the **Garbage Collector (GC) before an object is removed from memory**.

Note: As of Java 9+, finalize() is **deprecated** due to inefficiency and unpredictability.

Example:

```
public class Test {
   protected void finalize() {
      System.out.println("finalize() called before GC");
   }

   public static void main(String[] args) {
      Test obj = new Test();
      obj = null;
      System.gc(); // Suggest GC to run finalize
   }
}
```

Summary Table

Term	**Type**	**Purpose**
final	Keyword	Declare constants, prevent changes
finally	Code block	Ensure execution (e.g., cleanup)
finalize	Method	Cleanup before object is garbage collected

Term	Type	Purpose

52 . What is a singleton class in Java? And How to break the singleton class object?

Great question! Here's a clear explanation about **Singleton Class** in Java, and how its singleton nature can be broken.

What is a Singleton Class in Java?

A **Singleton class** is a class designed to allow **only one instance (object) to be created throughout the lifetime of a Java application**.

Why use Singleton?

- To ensure controlled access to a resource (like a database connection).
- To save memory by preventing multiple instances.
- To coordinate actions across the system via a single instance.

How to create a Singleton class?

Key points:

- Make the constructor **private** so no other class can instantiate it.
- Create a **private static instance** of the class inside the class.
- Provide a **public static method** to return the instance (usually named getInstance()).

Example of Singleton Class

```
public class Singleton {

   private static Singleton instance;
```

Term	Type	Purpose

```
    // Private constructor to restrict instantiation
    private Singleton() {
    }

    // Public method to provide access to the instance
    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }

    public void show() {
        System.out.println("Singleton instance method called.");
    }
}
```

How to break Singleton in Java?

Despite the safeguards, there are several ways to break singleton:

1. Using Reflection

By using Java Reflection API, you can invoke the private constructor and create a new instance.

```
import java.lang.reflect.Constructor;

public class TestReflection {
    public static void main(String[] args) throws Exception {
```

Term	Type	Purpose

```
        Singleton instance1 = Singleton.getInstance();


        Constructor<Singleton> constructor =
Singleton.class.getDeclaredConstructor();

        constructor.setAccessible(true); // allow access to private
constructor

        Singleton instance2 = constructor.newInstance();


        System.out.println(instance1);

        System.out.println(instance2);

        System.out.println("Are both instances same? " + (instance1 ==
instance2));

    }

}
```

Output will show **two different instances**, breaking the singleton pattern.

2. Using Serialization and Deserialization

If singleton class implements Serializable, deserialization creates a new object, breaking singleton.

To fix this, implement readResolve() method in singleton class:

```
protected Object readResolve() {

    return getInstance();

}
```

3. Using Cloning

If clone() is not overridden properly, cloning can create new instances.

To prevent:

```
@Override
```

Term	Type	Purpose

```
protected Object clone() throws CloneNotSupportedException {
    throw new CloneNotSupportedException();
}
```

Summary

Break Method	How to Prevent
Reflection	Throw exception from constructor if instance exists
Serialization	Implement readResolve() method
Cloning	Override clone() and throw exception

53 Differentiate between instance and local variables

Sure! Here’s a clear differentiation between **instance variables** and **local variables** in Java:

Feature	Instance Variables	Local Variables
Definition	Variables declared inside a class but outside any method or constructor.	Variables declared inside a method, constructor, or block.
Scope	Available to all non-static methods of the class (through the instance).	Only accessible within the method/block where declared.
Memory allocation	Stored in the heap as part of the object.	Stored in the stack during method execution.
Default value	Automatically initialized with default values (e.g., 0, null, false).	Must be explicitly initialized before use.

Term			Type	Purpose
Lifetime	Exist as long as the object exists (object lifetime).	Exist only during method execution, destroyed once method ends.		
Access modifier	Can have access modifiers (private, public, etc.).	Cannot have access modifiers.		
Usage	Used to represent properties or state of an object.	Used for temporary calculations or logic inside methods.		

Example to illustrate:

```
public class Example {
  int instanceVar;  // instance variable


  public void method() {
    int localVar = 10;  // local variable
    System.out.println("Local variable: " + localVar);
  }


  public static void main(String[] args) {
    Example obj = new Example();
    System.out.println("Instance variable: " + obj.instanceVar); //
default 0
    obj.method();
  }
}
```

54 Explain the types of Exceptions in java?

Term	Type	Purpose

Sure! Here’s an explanation of the **types of Exceptions in Java**:

Types of Exceptions in Java

Java exceptions are mainly classified into **two types**:

1. **Checked Exceptions**
2. **Unchecked Exceptions**

1. Checked Exceptions

- These are exceptions that are **checked at compile-time** by the Java compiler.
- The programmer **must handle** these exceptions using try-catch blocks or declare them with the throws keyword; otherwise, the program will not compile.
- These exceptions are usually external to the program and can be anticipated and recovered from.
- Examples:
    - IOException
    - SQLException
    - ClassNotFoundException
    - FileNotFoundException

2. Unchecked Exceptions (Runtime Exceptions)

- These exceptions occur **during runtime**, and the compiler **does not check** them.
- Handling these exceptions is optional, but if they occur and are not handled, the program will terminate abruptly.

Term	Type	Purpose

- They usually indicate programming errors such as logic mistakes or improper use of APIs.
- These exceptions inherit from RuntimeException.
- Examples:
  - NullPointerException
  - ArrayIndexOutOfBoundsException
  - ArithmeticException (like divide by zero)
  - IllegalArgumentException

Summary Table

Exception Type	Checked or Unchecked	When Checked	Example	Handling Required?
Checked Exception	Checked	Compile time	IOException, SQLException	Yes
Unchecked Exception	Unchecked (Runtime)	Runtime	NullPointerException, ArithmeticException	No (optional)

Bonus: Errors (Not Exceptions)

- Errors are **not exceptions** but problems that occur beyond the control of the program, usually related to the environment (e.g., JVM).
- Examples: OutOfMemoryError, StackOverflowError.
- Usually, errors are not handled by programs.

55 Can you call a constructor of a class inside another constructor of same class?

Yes, you **can** call one constructor from another constructor within the same class in Java. This is called **constructor chaining**.

How to do it?

Term	Type	Purpose

You use the keyword this() to call another constructor of the same class.

Important points:

- The call to this() must be the **first statement** inside the constructor.
- You cannot use this() and super() in the same constructor (because both must be the first statement).
- It helps to avoid code duplication by reusing constructor logic.

Example:

```
public class Example {
   int x, y;

   // Constructor 1
   public Example() {
      this(10, 20);  // calls Constructor 2 with arguments
      System.out.println("Default constructor called");
   }

   // Constructor 2
   public Example(int x, int y) {
      this.x = x;
      this.y = y;
      System.out.println("Parameterized constructor called with x=" +
x + ", y=" + y);
   }
```

Term	**Type**	**Purpose**

```
    public static void main(String[] args) {
        Example obj = new Example();
    }
}
```

Output:

```
Parameterized constructor called with x=10, y=20
Default constructor called
```

Summary:

- Use this() to call one constructor from another within the same class.
- Must be the first statement in the constructor.

56 explain the Java thread lifecycle?

Sure! Here's an explanation of the **Java Thread Lifecycle**:

Java Thread Lifecycle

A thread in Java goes through various states during its lifetime. The **Java Thread Lifecycle** consists of **five** main states:

1. **New (Created)**
2. **Runnable (Ready to run)**
3. **Running**
4. **Blocked/Waiting/Timed Waiting**
5. **Terminated (Dead)**

Term	**Type**	**Purpose**

1. New (Created) State

- When a thread is created using Thread t = new Thread(); but **not yet started**.
- The thread is just an object and not yet running.

2. Runnable (Ready to run) State

- When the start() method is called on the thread, it enters the **Runnable** state.
- The thread is **ready to run**, but the actual execution depends on the thread scheduler.
- The thread is waiting for CPU time to run.

3. Running State

- When the thread scheduler picks the thread from the Runnable pool and starts executing its run() method.
- The thread is actively executing.

4. Blocked/Waiting/Timed Waiting State

- **Blocked:** When a thread is waiting for a monitor lock to enter a synchronized block/method.
- **Waiting:** When a thread waits indefinitely for another thread to perform a specific action (wait() method without timeout).
- **Timed Waiting:** When a thread waits for a specific period (sleep(), wait(timeout), join(timeout)).

5. Terminated (Dead) State

- When the thread has completed its execution or is terminated prematurely.
- The thread is dead and cannot be restarted.

Term	Type	Purpose

Diagram Summary:

```
New --> Runnable --> Running --> Terminated
            |          |
            |          v
            |     Blocked/Waiting/Timed Waiting
            |          |
            <------------
```

Quick Example of Thread States:

```
public class ThreadLifecycleDemo extends Thread {
   public void run() {
      System.out.println("Thread is running");
      try {
         Thread.sleep(1000);  // Timed Waiting
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
      System.out.println("Thread finished execution");
   }

   public static void main(String[] args) {
      ThreadLifecycleDemo t = new ThreadLifecycleDemo(); //
New state
      t.start();  // Runnable -> Running
      System.out.println("Main method ended");
```

Term	Type	Purpose

```
            }
          }
```

56 . Write a prog. to find prime numbers in an array.

```
import java.util.Scanner;

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

        // Input size of the array
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();

        int[] arr = new int[n];

        // Input array elements
        System.out.println("Enter " + n + " elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("\nPrime numbers in the array:");
```

Term	Type	Purpose

```
        for (int num : arr) {
            if (isPrime(num)) {
                System.out.print(num + " ");
            }
        }

        // Print name and enrollment
        System.out.println("\n\nName: Neeraj");
        System.out.println("Enroll: 0873CS231073");

        sc.close();
    }

    // Method to check if a number is prime
    public static boolean isPrime(int number) {
        if (number <= 1) return false;
        if (number == 2) return true;
        if (number % 2 == 0) return false;

        for (int i = 3; i <= Math.sqrt(number); i += 2) {
            if (number % i == 0) return false;
        }
        return true;
    }
}
```

Term	Type	Purpose

Output

```
Enter the size of the array: 6

Enter 6 elements:

12 5 7 9 11 20


Prime numbers in the array:

5 7 11
```

Name: Neeraj

Enroll: 0873CS231073

57 . Can we inherit a Constructor?

No, in Java, constructors are not inherited.

Explanation:

- Constructors are special methods used to initialize objects of a class.
- Unlike normal methods, constructors **cannot be inherited** by subclasses.
- However, a subclass constructor can **call** a superclass constructor using the keyword super().
- This call is either implicit (default constructor) or explicit (when you want to call a specific superclass constructor).

Summary:

Aspect	**Constructor Inheritance**
Can constructors be inherited?	**No**

Term		**Type**	**Purpose**
Can subclass call superclass constructor?	**Yes, via super()**		
Purpose of calling superclass constructor	To initialize inherited members properly		

Example:

```
class Parent {
    Parent() {
        System.out.println("Parent constructor called");
    }
}

class Child extends Parent {
    Child() {
        super();  // Calls Parent's constructor explicitly
        System.out.println("Child constructor called");
    }
}

public class Test {
    public static void main(String[] args) {
        Child c = new Child();
    }
}
```

Output:

```
Parent constructor called
Child constructor called
```

Term	Type	Purpose

58 . How will you implement method overloading in Java?

Great question! Here's how **method overloading** works in Java:

What is Method Overloading?

Method Overloading means having multiple methods in the same class with:

- The **same method name**, but
- **Different parameter lists** (different number, type, or order of parameters)

It allows a method to perform different tasks based on the input parameters.

How to Implement Method Overloading?

You simply declare multiple methods with the same name but different parameters in the same class.

Example:

```
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;
    }

    // Overloaded method to add int and double
    public double add(int a, double b) {
        return a + b;
    }

    // Main method to test
    public static void main(String[] args) {
        Calculator calc = new Calculator();

        System.out.println(calc.add(5, 10));          // Output: 15
        System.out.println(calc.add(5, 10, 15));      // Output: 30
        System.out.println(calc.add(5.5, 4.5));       // Output: 10.0
        System.out.println(calc.add(5, 4.5));         // Output: 9.5
    }
}
```

Key points:

- Overloaded methods **must differ in parameters**.
- Return type alone **cannot** be used to overload a method.
- Method overloading is a **compile-time polymorphism**.

59 What is Runtime Polymorphism

Runtime Polymorphism in Java, also known as **Dynamic Method Dispatch**, is a feature that allows a call to an overridden method to be resolved at **runtime** rather than at compile time.

What does that mean?

- When a superclass reference variable points to a subclass object and calls a method overridden in the subclass, **the version of the method that gets executed is decided at runtime** based on the actual object type (not the reference type).
- This enables flexibility and dynamic behavior in programs.

How is it achieved?

- Through **method overriding** (subclass provides specific implementation of a method declared in superclass).
- By using **inheritance** and **upcasting** (superclass reference refers to subclass object).

Example:

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


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

```
class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class TestPolymorphism {
    public static void main(String[] args) {
        Animal a;

        a = new Dog();
        a.sound();   // Output: Dog barks

        a = new Cat();
        a.sound();   // Output: Cat meows
    }
}
```

Summary:

Aspect	**Runtime Polymorphism**
When method is bound	At runtime (dynamic binding)
How achieved	Method overriding
Requires inheritance	Yes
Reference type vs Object type	Reference type determines what methods are accessible; Object type determines which method implementation is executed

Aspect	Runtime Polymorphism
Example keyword	@Override (for clarity, not mandatory)

60 How does Garbage Collection work in Java?

How Garbage Collection Works in Java

Garbage Collection (GC) in Java is a process managed by the **Java Virtual Machine (JVM)** to automatically reclaim memory used by objects that are no longer reachable or needed. This helps prevent memory leaks and improves performance.

Key Concepts:

1. **Heap Memory**:
   - All objects in Java are created in heap memory.
   - The JVM manages this memory space and uses GC to clean up unused objects.
2. **Reachability**:
   - An object becomes eligible for garbage collection when it is **no longer reachable** from any live thread or static reference.
   - In simple terms, if you can't access it anymore in your code, the JVM can clean it up.

Phases of Garbage Collection:

Java GC typically involves these steps:

1. **Mark**:
   - The GC identifies all **reachable** (still used) objects.
2. **Sweep** (or Delete):

   - All unreferenced (unreachable) objects are considered garbage and are removed.
3. **Compact** (optional):
   - After deletion, the remaining objects may be moved to compact memory and reduce fragmentation.

Example:

```
public class GarbageExample {
    public static void main(String[] args) {
        GarbageExample obj1 = new GarbageExample();
        GarbageExample obj2 = new GarbageExample();

        obj1 = null; // Eligible for GC
        obj2 = null; // Eligible for GC

        System.gc(); // Suggests JVM to run GC (not guaranteed)

        System.out.println("Objects set to null, GC may run...");
    }

    @Override
    protected void finalize() throws Throwable {
        System.out.println("Garbage collected object");
    }
}
```

finalize() is called by GC **before** an object is removed, but it's deprecated in newer Java versions.

Important Points:

Feature	Description
Managed by JVM	GC is automatic; developers don’t delete objects manually.
System.gc()	Suggests JVM to run GC, but **not guaranteed**.
Unreachable objects	Are automatically identified and cleared.
GC Algorithms	JVM uses algorithms like Mark-Sweep, G1, CMS, etc.
finalize()	Was used for cleanup but is now deprecated (Java 9+).

Benefits:

- Prevents memory leaks.
- Makes Java safer and easier to use.

61 . Write a Java program to create an ArrayList, add some colors (as strings), and print the collection.

Java Program

```
import java.util.ArrayList;


public class ColorList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073


        // Create an ArrayList of Strings
        ArrayList<String> colors = new ArrayList<>();


        // Add color names to the list
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");
        colors.add("Yellow");
        colors.add("Purple");

        // Print the colors
        System.out.println("List of Colors:");
        for (String color : colors) {
            System.out.println(color);
        }
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
List of Colors:
Red
Green
Blue
Yellow
Purple
```

62 Write a Java program to iterate through all elements in an ArrayList.

Here is a Java program that **iterates through all elements in an ArrayList** using different methods:

Java Program

```
import java.util.ArrayList;
import java.util.Iterator;
```

```
public class IterateArrayList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add elements
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Orange");
        fruits.add("Grapes");

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Fruits in the ArrayList:");

        // 1. Using for-each loop
        for (String fruit : fruits) {
            System.out.println(fruit);
        }

        // 2. Using Iterator
        System.out.println("\nIterating using Iterator:");
        Iterator<String> iterator = fruits.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }

        // 3. Using traditional for loop
        System.out.println("\nIterating using traditional for loop:");
        for (int i = 0; i < fruits.size(); i++) {
            System.out.println(fruits.get(i));
        }
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
Fruits in the ArrayList:
Apple
Banana
Mango
Orange
Grapes

Iterating using Iterator:
Apple
Banana
Mango
Orange
Grapes

Iterating using traditional for loop:
Apple
```

```
Banana
Mango
Orange
Grapes
```

63 . Write a Java program to insert an element into the ArrayList at the first position.

```
import java.util.ArrayList;

public class InsertElement {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add elements
        ArrayList<String> languages = new ArrayList<>();
        languages.add("Java");
        languages.add("Python");
        languages.add("C++");

        // Display original list
        System.out.println("Original ArrayList:");
        System.out.println(languages);

        // Insert element at the first position (index 0)
        languages.add(0, "JavaScript");
        // Display updated list
        System.out.println("\nArrayList after inserting at first position:");
        System.out.println(languages);
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
Original ArrayList:
[Java, Python, C++]

ArrayList after inserting at first position:
[JavaScript, Java, Python, C++]
```

64 . Write a Java program to retrieve an element at a specified index from a given ArrayList.

Java Program

```
import java.util.ArrayList;
import java.util.Scanner;

public class RetrieveElement {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add some elements
        ArrayList<String> cities = new ArrayList<>();
        cities.add("Delhi");
        cities.add("Mumbai");
        cities.add("Chennai");
        cities.add("Kolkata");
        cities.add("Bangalore");

        // Display the list
        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Available Cities: " + cities);

        // Ask user to enter index
        Scanner scanner = new Scanner(System.in);
        System.out.print("\nEnter index to retrieve (0 to " + (cities.size() - 1) + "): ");
        int index = scanner.nextInt();

        // Retrieve and display the element
        if (index >= 0 && index < cities.size()) {
            String element = cities.get(index);
            System.out.println("Element at index " + index + ": " + element);
        } else {
            System.out.println("Invalid index! Please enter a valid index.");
        }

        scanner.close();
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
```

```
Available Cities: [Delhi, Mumbai, Chennai, Kolkata, Bangalore]

Enter index to retrieve (0 to 4): 2
Element at index 2: Chennai
```

65 Write a Java program to update an ArrayList element by a given element.

Java Program

```java
import java.util.ArrayList;
import java.util.Scanner;

public class UpdateArrayList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add elements
        ArrayList<String> subjects = new ArrayList<>();
        subjects.add("Math");
        subjects.add("Physics");
        subjects.add("Chemistry");
        subjects.add("Biology");

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Original ArrayList: " + subjects);
        Scanner scanner = new Scanner(System.in);

        // Input old and new values
        System.out.print("Enter the subject you want to replace: ");
        String oldSubject = scanner.nextLine();

        System.out.print("Enter the new subject: ");
        String newSubject = scanner.nextLine();

        // Update element
        if (subjects.contains(oldSubject)) {
            int index = subjects.indexOf(oldSubject);
            subjects.set(index, newSubject);
            System.out.println("Updated ArrayList: " + subjects);
        } else {
            System.out.println("Subject not found in the list.");
        }

        scanner.close();
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
Original ArrayList: [Math, Physics, Chemistry, Biology]
Enter the subject you want to replace: Chemistry
Enter the new subject: Computer Science
```

Updated ArrayList: [Math, Physics, Computer Science, Biology]

66 . Write a Java program to remove the third element from an ArrayList.

Here's a Java program that **removes the third element** (element at index 2) from an ArrayList:

Java Program

```
import java.util.ArrayList;

public class RemoveElement {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add elements
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Grapes");
        fruits.add("Orange");

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Original ArrayList: " + fruits);
        // Remove third element (index 2)
        if (fruits.size() >= 3) {
            fruits.remove(2);
            System.out.println("Updated ArrayList (after removing 3rd element): " + fruits);
        } else {
            System.out.println("ArrayList does not have a third element to remove.");
        }
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
Original ArrayList: [Apple, Banana, Mango, Grapes, Orange]
Updated ArrayList (after removing 3rd element): [Apple, Banana, Grapes, Orange]
```

67 Write a Java program to search for an element in an ArrayList.

Java Program

```
import java.util.ArrayList;
import java.util.Scanner;

public class SearchElement {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create and populate ArrayList
}

        ArrayList<String> colors = new ArrayList<>();
        colors.add("Red");
        colors.add("Blue");
        colors.add("Green");
        colors.add("Yellow");
        colors.add("Orange");

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Available colors: " + colors);

        // Input element to search
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter color to search: ");
        String searchColor = scanner.nextLine();

        // Search for element
        if (colors.contains(searchColor)) {
            System.out.println(searchColor + " is found at index " + colors.indexOf(searchColor));
        } else {
            System.out.println(searchColor + " is not found in the list.");
        }

        scanner.close();
    }
}
```

Output

```
Student Name: Neeraj
```

```
Enrollment No: 0873CS231073
Available colors: [Red, Blue, Green, Yellow, Orange]
Enter color to search: Green
Green is found at index 2
```

68 Write a Java program to sort a given ArrayList.

Java Program

```java
import java.util.ArrayList;
import java.util.Collections;

public class SortArrayList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create and populate ArrayList
        ArrayList<String> names = new ArrayList<>();
        names.add("Zara");
        names.add("Amit");
        names.add("Neeraj");
        names.add("Kiran");
        names.add("John");

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Original ArrayList: " + names);

        // Sort the ArrayList
        Collections.sort(names);
        System.out.println("Sorted ArrayList: " + names);
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
Original ArrayList: [Zara, Amit, Neeraj, Kiran, John]
Sorted ArrayList: [Amit, John, Kiran, Neeraj, Zara]
```

69 Write a Java program to copy one array list into another.

Java Program

```
import java.util.ArrayList;
import java.util.Collections;

public class CopyArrayList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Original ArrayList
        ArrayList<String> sourceList = new ArrayList<>();
        sourceList.add("Red");
        sourceList.add("Green");
        sourceList.add("Blue");

        // Target ArrayList (must be the same size or larger before using Collections.copy)
        ArrayList<String> targetList = new ArrayList<>(sourceList.size());
        // Add dummy elements to match size
        for (int i = 0; i < sourceList.size(); i++) {
            targetList.add(""); // fill with empty strings
        }

        // Copy contents
        Collections.copy(targetList, sourceList);

        // Output
        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        System.out.println("Source ArrayList: " + sourceList);
        System.out.println("Copied ArrayList: " + targetList);
    }
}
```

Output

```
Student Name: Neeraj
Enrollment No: 0873CS231073
Source ArrayList: [Red, Green, Blue]
Copied ArrayList: [Red, Green, Blue]
```

70 Write a Java program to shuffle elements in an array list.

```
import java.util.ArrayList;
import java.util.Collections;

public class ShuffleArrayList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add elements
        ArrayList<String> colors = new ArrayList<>();
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");
        colors.add("Yellow");
        colors.add("Orange");

        System.out.println("Before shuffling: " + colors);

        // Shuffle the ArrayList
        Collections.shuffle(colors);

        System.out.println("After shuffling: " + colors);

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
    }
}
```

Output

```
Before shuffling: [Red, Green, Blue, Yellow, Orange]
After shuffling: [Blue, Orange, Red, Green, Yellow]
Student Name: Neeraj
Enrollment No: 0873CS231073
```

71 Write a Java program to reverse elements in an array list.

```
import java.util.ArrayList;
import java.util.Collections;

public class ReverseArrayList {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // Create an ArrayList and add elements
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");
        fruits.add("Date");
        fruits.add("Elderberry");

        System.out.println("Original ArrayList: " + fruits);

        // Reverse the ArrayList
        Collections.reverse(fruits);

        System.out.println("Reversed ArrayList: " + fruits);

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
    }
}
```

Output

```
Original ArrayList: [Apple, Banana, Cherry, Date, Elderberry]
Reversed ArrayList: [Elderberry, Date, Cherry, Banana, Apple]
Student Name: Neeraj
Enrollment No: 0873CS231073
```

72 . Write a Java program to extract a portion of an array list.

Got it! Here's a Java program that extracts a portion (sublist) of an ArrayList and prints it along with your name and enrollment:

```
import java.util.ArrayList;

import java.util.List;


public class ExtractSubList {

    public static void main(String[] args) {

        // Student Name: Neeraj

        // Enrollment No: 0873CS231073


        // Create and populate the ArrayList

        ArrayList<String> colors = new ArrayList<>();

        colors.add("Red");

        colors.add("Green");

        colors.add("Blue");

        colors.add("Yellow");

        colors.add("Orange");

        colors.add("Purple");


        System.out.println("Original ArrayList: " + colors);


        // Extract a portion (from index 1 to 4 exclusive)

        List<String> subList = colors.subList(1, 4);


        System.out.println("Extracted portion of ArrayList: " + subList);


        System.out.println("Student Name: Neeraj");

        System.out.println("Enrollment No: 0873CS231073");
    }
}
```

Output:

```
Original ArrayList: [Red, Green, Blue, Yellow, Orange, Purple]

Extracted portion of ArrayList: [Green, Blue, Yellow]

Student Name: Neeraj

Enrollment No: 0873CS231073
```

73 Write a Java program to compare two array lists.

Sure! Here's a Java program that compares two ArrayLists and prints whether they are equal or not, along with your name and enrollment:

```
import java.util.ArrayList;


public class CompareArrayLists {

    public static void main(String[] args) {

        // Student Name: Neeraj

        // Enrollment No: 0873CS231073


        // Create first ArrayList

        ArrayList<String> list1 = new ArrayList<>();

        list1.add("Red");

        list1.add("Green");

        list1.add("Blue");


        // Create second ArrayList

        ArrayList<String> list2 = new ArrayList<>();

        list2.add("Red");

        list2.add("Green");

        list2.add("Blue");
        // Compare the two lists
        if (list1.equals(list2)) {
            System.out.println("Both ArrayLists are equal.");
        } else {
            System.out.println("ArrayLists are not equal.");
        }

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
    }
}
```

Sample Output:

```
Both ArrayLists are equal.
Student Name: Neeraj
Enrollment No: 0873CS231073
```

74 Write a Java program that swaps two elements in an array list.

```
import java.util.ArrayList;
import java.util.Scanner;

public class SwapArrayListElements {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        ArrayList<String> list = new ArrayList<>();
        list.add("Apple");
        list.add("Banana");
        list.add("Cherry");
        list.add("Date");
        list.add("Elderberry");

        System.out.println("Original ArrayList: " + list);

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first index to swap: ");
        int index1 = sc.nextInt();
        System.out.print("Enter second index to swap: ");
        int index2 = sc.nextInt();

        // Validate indices
        if (index1 >= 0 && index1 < list.size() && index2 >= 0 && index2 < list.size()) {
            // Swap elements
            String temp = list.get(index1);
            list.set(index1, list.get(index2));
            list.set(index2, temp);

            System.out.println("ArrayList after swapping elements at indices " + index1 + " and " +
index2 + ":");
            System.out.println(list);
        } else {
            System.out.println("Invalid indices entered.");
        }
        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
        sc.close();
    }
}
```

Output:

```
Original ArrayList: [Apple, Banana, Cherry, Date, Elderberry]
Enter first index to swap: 1
Enter second index to swap: 3
ArrayList after swapping elements at indices 1 and 3:
[Apple, Date, Cherry, Banana, Elderberry]
Student Name: Neeraj
Enrollment No: 0873CS231073
```

75 Write a Java program to join two array lists.

```
import java.util.ArrayList;

public class JoinArrayLists {
    public static void main(String[] args) {
        // Student Name: Neeraj
        // Enrollment No: 0873CS231073

        // First ArrayList
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("Red");
        list1.add("Green");
        list1.add("Blue");

        // Second ArrayList
        ArrayList<String> list2 = new ArrayList<>();
        list2.add("Yellow");
        list2.add("Orange");
        list2.add("Purple");

        System.out.println("First ArrayList: " + list1);
        System.out.println("Second ArrayList: " + list2);

        // Joining list2 into list1
        list1.addAll(list2);

        System.out.println("Joined ArrayList: " + list1);

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
    }
}
```

Output:

```
First ArrayList: [Red, Green, Blue]
Second ArrayList: [Yellow, Orange, Purple]
Joined ArrayList: [Red, Green, Blue, Yellow, Orange, Purple]
Student Name: Neeraj
Enrollment No: 0873CS231073
```

76 Write a Java program to clone an array list to another array list.

```
import java.util.ArrayList;
```

```
public class CloneArrayList {
    public static void main(String[] args) {
        // Original ArrayList
        ArrayList<String> originalList = new ArrayList<>();
        originalList.add("Apple");
        originalList.add("Banana");
        originalList.add("Cherry");

        // Cloning the original ArrayList
        ArrayList<String> clonedList = (ArrayList<String>) originalList.clone();

        System.out.println("Original ArrayList: " + originalList);
        System.out.println("Cloned ArrayList: " + clonedList);

        System.out.println("Student Name: Neeraj");
        System.out.println("Enrollment No: 0873CS231073");
    }
}
```

Output:

```
Original ArrayList: [Apple, Banana, Cherry]
Cloned ArrayList: [Apple, Banana, Cherry]
Student Name: Neeraj
Enrollment No: 0873CS231073
```

77 Write a Java program to empty an array list.

```
import java.util.ArrayList;
```

```
public class StudentList {
    public static void main(String[] args) {
        // Create an ArrayList
        ArrayList<String> studentInfo = new ArrayList<>();

        // Pre-fill it with some dummy data (optional)
        studentInfo.add("Dummy Name");
        studentInfo.add("1234XYZ");

        // Display before clearing
        System.out.println("Before clearing: " + studentInfo);

        // Clear the ArrayList
        studentInfo.clear();

        // Add Neeraj Yadav and Enrollment Number
        studentInfo.add("Neeraj Yadav");
        studentInfo.add("0873CS231073");

        // Display the updated ArrayList
        System.out.println("After adding new student: " + studentInfo);
    }
}
```

Output:

```
Before clearing: [Dummy Name, 1234XYZ]

After adding new student: [Neeraj Yadav, 0873CS231073]
```

78. Write a Java program to test whether an array list is empty or not.

```
import java.util.ArrayList;


public class ArrayListEmptyCheck {
    public static void main(String[] args) {
        // Create an ArrayList
        ArrayList<String> items = new ArrayList<>();


        // Check if the ArrayList is empty
        if (items.isEmpty()) {
            System.out.println("The ArrayList is empty.");
        } else {
            System.out.println("The ArrayList is not empty.");
        }


        // Add an element
        items.add("Example Item");


        // Check again
        if (items.isEmpty()) {
            System.out.println("The ArrayList is empty.");
        } else {
            System.out.println("The ArrayList is not empty.");
        }
    }
}
```

Output:

```
The ArrayList is empty.

The ArrayList is not empty.
```

79 Write a Java program for trimming the capacity of an array list.

Java Program: Trimming the Capacity of an ArrayList

```
import java.util.ArrayList;

public class TrimArrayList {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create an ArrayList with initial capacity of 10
        ArrayList<String> cities = new ArrayList<>(10);

        // Add some elements
        cities.add("Delhi");
        cities.add("Mumbai");
        cities.add("Kolkata");

        // Display size and contents before trimming
        System.out.println("Before trimming:");
        System.out.println("Size: " + cities.size());
        System.out.println("Contents: " + cities);

        // Trim the capacity to the current size
        cities.trimToSize();
        // Display after trimming
        System.out.println("After trimming the capacity to size.");
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
Before trimming:
Size: 3
Contents: [Delhi, Mumbai, Kolkata]
After trimming the capacity to size.
```

80 Write a Java program to increase an array list size.

Java Program: Increase ArrayList Size

```
import java.util.ArrayList;

public class IncreaseArrayListSize {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create an ArrayList with some initial elements
        ArrayList<String> fruits = new ArrayList<>();

        // Initial elements
        fruits.add("Apple");
        fruits.add("Banana");

        // Display initial size and contents
        System.out.println("Initial ArrayList size: " + fruits.size());
        System.out.println("Contents: " + fruits);

        // Increase the size by adding more elements
        fruits.add("Mango");
        fruits.add("Orange");
        fruits.add("Pineapple");

        // Display size and contents after adding elements
        System.out.println("After increasing size:");
        System.out.println("New size: " + fruits.size());
        System.out.println("Contents: " + fruits);
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
Initial ArrayList size: 2
Contents: [Apple, Banana]
After increasing size:
New size: 5
Contents: [Apple, Banana, Mango, Orange, Pineapple]
```

81 Write a Java program to replace the second element of an ArrayList with the specified element.

```
import java.util.ArrayList;


public class ReplaceSecondElement {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");


        // Create and populate the ArrayList
        ArrayList<String> colors = new ArrayList<>();
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");
        colors.add("Yellow");


        // Display original list
        System.out.println("Original ArrayList: " + colors);


        // Replace the second element (index 1) with a new element
        colors.set(1, "Purple");


        // Display updated list
        System.out.println("After replacing second element: " + colors);
    }
}
```

Output:

```
Name: Neeraj Yadav

Enroll: 0873CS231073

Original ArrayList: [Red, Green, Blue, Yellow]

After replacing second element: [Red, Purple, Blue, Yellow]
```

82 Write a Java program to print all the elements of an ArrayList using the elements' position.

```
import java.util.LinkedList;


public class AppendToLinkedList {

    public static void main(String[] args) {

        // Print name and enrollment number

        System.out.println("Name: Neeraj Yadav");

        System.out.println("Enroll: 0873CS231073");


        // Create a LinkedList and add some elements

        LinkedList<String> animals = new LinkedList<>();

        animals.add("Cat");

        animals.add("Dog");

        animals.add("Elephant");


        // Display original list

        System.out.println("Original LinkedList: " + animals);


        // Append specified element to the end

        animals.add("Lion");


        // Display updated list
        System.out.println("After appending element: " + animals);
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
Original LinkedList: [Cat, Dog, Elephant]
After appending element: [Cat, Dog, Elephant, Lion]
```

83. Write a Java program to append a specified element to the end of a linked list.

```
import java.util.LinkedList;

public class AppendElementToLinkedList {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create a LinkedList and add some initial elements
        LinkedList<String> list = new LinkedList<>();
        list.add("Apple");
        list.add("Banana");
        list.add("Cherry");
        // Display the original LinkedList
        System.out.println("Original LinkedList: " + list);

        // Append a specified element to the end
        list.add("Date");

        // Display the updated LinkedList
        System.out.println("After appending element: " + list);
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
Original LinkedList: [Apple, Banana, Cherry]
After appending element: [Apple, Banana, Cherry, Date]
```

84. Write a Java program to iterate through all elements in a linked list.

```
import java.util.LinkedList;

public class IterateLinkedList {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create and populate LinkedList
        LinkedList<String> fruits = new LinkedList<>();
        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Cherry");

        fruits.add("Date");


        // Iterate using for-each loop

        System.out.println("LinkedList elements:");

        for (String fruit : fruits) {

            System.out.println(fruit);

        }

    }

}
```

Output:

```
Name: Neeraj Yadav

Enroll: 0873CS231073

LinkedList elements:

Apple

Banana

Cherry

Date
```

85. Write a Java program to iterate through all elements starting from a specified position in a linked list.

```
import java.util.LinkedList;

import java.util.ListIterator;


public class IterateFromPosition {

    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create and populate the LinkedList
        LinkedList<String> animals = new LinkedList<>();
        animals.add("Cat");
        animals.add("Dog");
        animals.add("Elephant");
        animals.add("Fox");
        animals.add("Giraffe");

        // Specify the starting position (index)
        int startIndex = 2;

        System.out.println("Iterating from position " + startIndex + ":");

        // Use ListIterator starting at startIndex
        ListIterator<String> iterator = animals.listIterator(startIndex);

        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Output:

```
Name: Neeraj Yadav
```

```
Enroll: 0873CS231073
Iterating from position 2:
Elephant
Fox
Giraffe
```

86. Write a Java program to iterate a linked list in reverse order.

```java
import java.util.LinkedList;
import java.util.ListIterator;

public class IterateLinkedListReverse {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create and populate the LinkedList
        LinkedList<String> cities = new LinkedList<>();
        cities.add("New York");
        cities.add("London");
        cities.add("Paris");
        cities.add("Tokyo");

        // Get a ListIterator positioned at the end of the list
        ListIterator<String> iterator = cities.listIterator(cities.size());

        System.out.println("LinkedList in reverse order:");
        // Iterate backwards using hasPrevious() and previous()
        while (iterator.hasPrevious()) {
            System.out.println(iterator.previous());
        }
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
LinkedList in reverse order:
Tokyo
Paris
London
New York
```

87 Write a Java program to insert a specified element at a given position in a linked list.

```
import java.util.LinkedList;

public class InsertElementAtPosition {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create and populate the LinkedList
        LinkedList<String> fruits = new LinkedList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");

        // Display original list
        System.out.println("Original LinkedList: " + fruits);

        // Specify position and element to insert
        int position = 1;          // 0-based index
        String newElement = "Mango";

        // Insert the element at specified position
        fruits.add(position, newElement);

        // Display updated list
        System.out.println("After inserting '" + newElement + "' at position " + position + ":");
        System.out.println(fruits);
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
Original LinkedList: [Apple, Banana, Cherry]
After inserting 'Mango' at position 1:
[Apple, Mango, Banana, Cherry]
```

88 Write a Java program to insert elements at the first and last positions of a linked list.

```
import java.util.LinkedList;

public class InsertAtFirstAndLast {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create and populate the LinkedList
        LinkedList<String> colors = new LinkedList<>();
        colors.add("Blue");
        colors.add("Green");
        colors.add("Yellow");

        // Display original list
        System.out.println("Original LinkedList: " + colors);

        // Insert element at the first position
        colors.addFirst("Red");

        // Insert element at the last position
        colors.addLast("Purple");

        // Display updated list
        System.out.println("After inserting at first and last positions:");
        System.out.println(colors);
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
Original LinkedList: [Blue, Green, Yellow]
After inserting at first and last positions:
[Red, Blue, Green, Yellow, Purple]
```

89 . Write a Java program to add all elements from one TreeSet to another TreeSet.

```
import java.util.TreeSet;

public class AddAllTreeSet {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create first TreeSet and add elements
        TreeSet<String> set1 = new TreeSet<>();
        set1.add("Apple");
        set1.add("Banana");
        set1.add("Cherry");

        // Create second TreeSet and add elements
        TreeSet<String> set2 = new TreeSet<>();
        set2.add("Date");
        set2.add("Fig");
        set2.add("Grape");

        // Display original sets
        System.out.println("TreeSet 1: " + set1);
        System.out.println("TreeSet 2: " + set2);

        // Add all elements from set2 to set1
        set1.addAll(set2);

        // Display set1 after adding all elements
        System.out.println("TreeSet 1 after adding all elements from TreeSet 2:");
        System.out.println(set1);
    }
}
```

Output:

```
Name: Neeraj Yadav
Enroll: 0873CS231073
TreeSet 1: [Apple, Banana, Cherry]
TreeSet 2: [Date, Fig, Grape]
TreeSet 1 after adding all elements from TreeSet 2:
[Apple, Banana, Cherry, Date, Fig, Grape]
```

90 . Write a Java program to display the elements of a TreeSet in reverse order.

```
import java.util.TreeSet;
import java.util.NavigableSet;
```

```
}

public class DisplayTreeSetReverse {
    public static void main(String[] args) {
        // Print name and enrollment number
        System.out.println("Name: Neeraj Yadav");
        System.out.println("Enroll: 0873CS231073");

        // Create and populate TreeSet
        TreeSet<String> fruits = new TreeSet<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");
        fruits.add("Date");
        fruits.add("Fig");

        // Display original TreeSet
        System.out.println("Original TreeSet: " + fruits);

        // Get the reverse order view of the TreeSet
        NavigableSet<String> reverseSet = fruits.descendingSet();

        // Display elements in reverse order
        System.out.println("TreeSet in reverse order:");
        for (String fruit : reverseSet) {
            System.out.println(fruit);
        }
    }
}
```

Output:

```
Name: Neeraj Yadav

Enroll: 0873CS231073

Original TreeSet: [Apple, Banana, Cherry, Date, Fig]

TreeSet in reverse order:

Fig

Date

Cherry

Banana

Apple
```

91 Write a Java program to retrieve the first and last elements from a TreeSet.

```
import java.util.TreeSet;


public class TreeSetFirstLast {

  public static void main(String[] args) {

    // Creating a TreeSet of integers

    TreeSet<Integer> numbers = new TreeSet<>();


    // Adding elements to the TreeSet

    numbers.add(10);

    numbers.add(5);

    numbers.add(20);

    numbers.add(15);

    numbers.add(1);
```

```
        // Retrieving the first and last elements
        int firstElement = numbers.first();
        int lastElement = numbers.last();

        // Displaying the results
        System.out.println("Program by Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("TreeSet Elements: " + numbers);
        System.out.println("First Element: " + firstElement);
        System.out.println("Last Element: " + lastElement);
    }
}
```

Output

```
Program by Neeraj Yadav
Enrollment: 0873CS231073
TreeSet Elements: [1, 5, 10, 15, 20]
First Element: 1
Last Element: 20
```

92. Write a Java program to clone a TreeSet into another TreeSet

```
import java.util.TreeSet;

public class CloneTreeSet {
    public static void main(String[] args) {
        TreeSet<String> originalSet = new TreeSet<>();
        originalSet.add("Apple");
        originalSet.add("Banana");
        originalSet.add("Mango");
        originalSet.add("Orange");

        TreeSet<String> clonedSet = (TreeSet<String>) originalSet.clone();

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("Original TreeSet: " + originalSet);
        System.out.println("Cloned TreeSet: " + clonedSet);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Original TreeSet: [Apple, Banana, Mango, Orange]
Cloned TreeSet: [Apple, Banana, Mango, Orange]
```

93. Write a Java program to count the number of elements in a TreeSet.

```
import java.util.TreeSet;

public class CloneTreeSet {
    public static void main(String[] args) {
        TreeSet<String> originalSet = new TreeSet<>();
        originalSet.add("Apple");
        originalSet.add("Banana");
        originalSet.add("Mango");
        originalSet.add("Orange");

        TreeSet<String> clonedSet = (TreeSet<String>) originalSet.clone();

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("Original TreeSet: " + originalSet);
        System.out.println("Cloned TreeSet: " + clonedSet);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Original TreeSet: [Apple, Banana, Mango, Orange]
Cloned TreeSet: [Apple, Banana, Mango, Orange]
```

94. Write a Java program to compare two TreeSets

```
import java.util.TreeSet;

public class CompareTreeSets {
    public static void main(String[] args) {
        TreeSet<Integer> set1 = new TreeSet<>();
        TreeSet<Integer> set2 = new TreeSet<>();

        set1.add(10);
        set1.add(20);

        set1.add(30);


        set2.add(20);

        set2.add(30);

        set2.add(40);


        System.out.println("Neeraj Yadav");

        System.out.println("Enrollment: 0873CS231073");


        // Comparing set1 and set2 elements

        for (Integer element : set1) {

            if (set2.contains(element)) {

                System.out.println(element + " is present in both TreeSets");

            } else {

                System.out.println(element + " is not present in second TreeSet");

            }

        }

    }

}
```

Output:

```
Neeraj Yadav

Enrollment: 0873CS231073

10 is not present in second TreeSet

20 is present in both TreeSets

30 is present in both TreeSets
```

95.Write a Java program to clone one HashSet into another

```
import java.util.HashSet;
```

```
public class CloneHashSet {
    public static void main(String[] args) {
        HashSet<String> originalSet = new HashSet<>();

        originalSet.add("Red");
        originalSet.add("Green");
        originalSet.add("Blue");
        originalSet.add("Yellow");

        @SuppressWarnings("unchecked")
        HashSet<String> clonedSet = (HashSet<String>) originalSet.clone();

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("Original HashSet: " + originalSet);
        System.out.println("Cloned HashSet: " + clonedSet);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Original HashSet: [Red, Green, Blue, Yellow]
Cloned HashSet: [Red, Green, Blue, Yellow]
```

96. Write a Java program to convert a HashSet into an array

```
import java.util.HashSet;
import java.util.Arrays;
```

```
public class HashSetToArray {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<>();

        set.add("Java");
        set.add("Python");
        set.add("C++");
        set.add("JavaScript");

        // Convert HashSet to Array
        String[] array = set.toArray(new String[0]);

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("HashSet: " + set);
        System.out.println("Array: " + Arrays.toString(array));
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
HashSet: [Java, Python, C++, JavaScript]
Array: [Java, Python, C++, JavaScript]
```

97. Write a Java program to convert a HashSet into a TreeSet

```
import java.util.HashSet;
import java.util.TreeSet;
```

```
public class HashSetToTreeSet {
    public static void main(String[] args) {
        HashSet<String> hashSet = new HashSet<>();

        hashSet.add("Banana");
        hashSet.add("Apple");
        hashSet.add("Orange");
        hashSet.add("Mango");

        TreeSet<String> treeSet = new TreeSet<>(hashSet);

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("HashSet: " + hashSet);
        System.out.println("TreeSet (sorted): " + treeSet);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
HashSet: [Banana, Apple, Orange, Mango]
TreeSet (sorted): [Apple, Banana, Mango, Orange]
```

98. Write a Java program to find numbers less than 7 in a TreeSet

```
import java.util.TreeSet;
import java.util.NavigableSet;

public class NumbersLessThanSeven {
    public static void main(String[] args) {
        TreeSet<Integer> numbers = new TreeSet<>();

        numbers.add(1);
        numbers.add(4);
        numbers.add(7);
        numbers.add(9);
        numbers.add(3);
        numbers.add(6);

        // Find numbers less than 7
        NavigableSet<Integer> lessThanSeven = numbers.headSet(7, false);

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");
        System.out.println("TreeSet: " + numbers);
        System.out.println("Numbers less than 7: " + lessThanSeven);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
TreeSet: [1, 3, 4, 6, 7, 9]
Numbers less than 7: [1, 3, 4, 6]
```

99. Write a Java program to compare two HashSets

```
import java.util.HashSet;
```

```
}

public class CompareHashSets {
    public static void main(String[] args) {
        HashSet<Integer> set1 = new HashSet<>();
        HashSet<Integer> set2 = new HashSet<>();

        set1.add(1);
        set1.add(2);
        set1.add(3);
        set1.add(4);

        set2.add(3);
        set2.add(4);
        set2.add(5);
        set2.add(6);

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");

        for (Integer element : set1) {
            if (set2.contains(element)) {
                System.out.println(element + " is present in both HashSets");
            } else {
                System.out.println(element + " is not present in second HashSet");
            }
        }
    }
}
```

Output:

```
Neeraj Yadav

Enrollment: 0873CS231073

1 is not present in second HashSet

2 is not present in second HashSet

3 is present in both HashSets

4 is present in both HashSets
```

100. Write a Java program to retain common elements from two sets

```
import java.util.HashSet;


public class RetainCommonElements {

    public static void main(String[] args) {

        HashSet<String> set1 = new HashSet<>();

        HashSet<String> set2 = new HashSet<>();


        set1.add("Apple");

        set1.add("Banana");

        set1.add("Cherry");

        set1.add("Date");


        set2.add("Banana");

        set2.add("Date");

        set2.add("Elderberry");

        set2.add("Fig");


        System.out.println("Neeraj Yadav");

        System.out.println("Enrollment: 0873CS231073");
        // Retain only common elements in set1
        set1.retainAll(set2);

        System.out.println("Common elements: " + set1);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Common elements: [Banana, Date]
```

101. Write a Java program to remove all elements from a HashSet

```
import java.util.HashSet;

public class RemoveAllFromHashSet {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<>();

        set.add("Red");
        set.add("Green");
        set.add("Blue");
        set.add("Yellow");

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");

        System.out.println("HashSet before clear: " + set);
        // Remove all elements
        set.clear();

        System.out.println("HashSet after clear: " + set);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
HashSet before clear: [Red, Green, Blue, Yellow]
HashSet after clear: []
```

102. Write a Java program to copy all mappings from one map to another

```
import java.util.HashMap;
import java.util.Map;

public class CopyMap {
    public static void main(String[] args) {
        Map<Integer, String> map1 = new HashMap<>();
        Map<Integer, String> map2 = new HashMap<>();

        map1.put(1, "Neeraj");
        map1.put(2, "Yadav");
        map1.put(3, "Student");
        // Copy all mappings from map1 to map2
        map2.putAll(map1);

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");

        System.out.println("Map1: " + map1);
        System.out.println("Map2 (after copy): " + map2);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Map1: {1=Neeraj, 2=Yadav, 3=Student}
Map2 (after copy): {1=Neeraj, 2=Yadav, 3=Student}
```

103. Write a Java program to remove all key-value pairs from a map

```
import java.util.HashMap;
import java.util.Map;

public class ClearMap {
    public static void main(String[] args) {
        Map<Integer, String> map = new HashMap<>();

        map.put(1, "Neeraj");
        map.put(2, "Yadav");
        map.put(3, "Student");
        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");

        System.out.println("Map before clear: " + map);

        map.clear();

        System.out.println("Map after clear: " + map);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Map before clear: {1=Neeraj, 2=Yadav, 3=Student}
Map after clear: {}
```

104. Write a Java program to check if a map is empty or contains key-value mappings

```
import java.util.HashMap;
import java.util.Map;

public class CheckMapEmpty {
    public static void main(String[] args) {
        Map<Integer, String> map = new HashMap<>();

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");


        if (map.isEmpty()) {
            System.out.println("Map is empty.");
        } else {
            System.out.println("Map contains key-value mappings.");
        }


        map.put(1, "Neeraj");


        if (map.isEmpty()) {
            System.out.println("Map is empty.");
        } else {
            System.out.println("Map contains key-value mappings.");


        }
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Map is empty.
Map contains key-value mappings.
```

105. Write a Java program to create a shallow copy of a HashMap instance

```
import java.util.HashMap;
import java.util.Map;
```

```
public class ShallowCopyHashMap {
    public static void main(String[] args) {
        HashMap<Integer, String> originalMap = new HashMap<>();

        originalMap.put(1, "Neeraj");
        originalMap.put(2, "Yadav");
        originalMap.put(3, "Student");

        // Creating shallow copy
        HashMap<Integer, String> shallowCopy = (HashMap<Integer, String>)
originalMap.clone();

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");

        System.out.println("Original Map: " + originalMap);
        System.out.println("Shallow Copy Map: " + shallowCopy);
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Original Map: {1=Neeraj, 2=Yadav, 3=Student}
Shallow Copy Map: {1=Neeraj, 2=Yadav, 3=Student}
```

106. Write a Java program to test whether a specified key exists in the map

```
import java.util.HashMap;
import java.util.Map;

public class CheckKeyInMap {
    public static void main(String[] args) {
        Map<Integer, String> map = new HashMap<>();

        map.put(1, "Neeraj");
        map.put(2, "Yadav");
        map.put(3, "Student");

        int keyToCheck = 2;

        System.out.println("Neeraj Yadav");
        System.out.println("Enrollment: 0873CS231073");

        if (map.containsKey(keyToCheck)) {
            System.out.println("Key " + keyToCheck + " exists in the map.");
        } else {
            System.out.println("Key " + keyToCheck + " does not exist in the map.");
        }
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
Key 2 exists in the map.
```

107. Create a table Item_dtls (Electronics), insert at least 10 records including 2 with null values

```
CREATE TABLE Item_dtls (
   item_id INT PRIMARY KEY,
   item_name VARCHAR(50),
   brand VARCHAR(50),
   price DECIMAL(10, 2),
   warranty_years INT
);

INSERT INTO Item_dtls (item_id, item_name, brand, price, warranty_years) VALUES
(1, 'Smartphone', 'Samsung', 699.99, 2),
(2, 'Laptop', 'Dell', 999.99, 3),
(3, 'Headphones', 'Sony', 199.99, 1),
(4, 'Smartwatch', 'Apple', 399.99, 2),
(5, 'Tablet', 'Amazon', 149.99, 1),
(6, 'Camera', 'Canon', 549.99, 3),
(7, 'Bluetooth Speaker', 'JBL', 99.99, NULL),
(8, 'External Hard Drive', 'Seagate', 89.99, 2),
(9, 'Monitor', 'LG', NULL, 3),
(10, 'Wireless Mouse', 'Logitech', 29.99, 1);

-- To verify the inserted records:
SELECT * FROM Item_dtls;
```

Output:

```
item_id | item_name        | brand    | price   | warranty_years
--------------------------------------------------------------
1       | Smartphone       | Samsung  | 699.99  | 2
```

```
2      | Laptop          | Dell     | 999.99  | 3
3      | Headphones       | Sony     | 199.99  | 1
4      | Smartwatch       | Apple    | 399.99  | 2
5      | Tablet          | Amazon   | 149.99  | 1
6      | Camera          | Canon    | 549.99  | 3
7      | Bluetooth Speaker | JBL      | 99.99   | NULL
8      | External Hard Drive| Seagate  | 89.99   | 2
9      | Monitor         | LG       | NULL    | 3
10     | Wireless Mouse    | Logitech | 29.99   | 1
```

108. Create a table Sales_dtls and insert at least 10 records including 2 with null values

```
CREATE TABLE Sales_dtls (
   sale_id INT PRIMARY KEY,
   item_id INT,
   sale_date DATE,
   quantity INT,
   total_amount DECIMAL(10, 2)
);

INSERT INTO Sales_dtls (sale_id, item_id, sale_date, quantity, total_amount) VALUES
(1, 1, '2025-01-15', 5, 3499.95),
(2, 2, '2025-01-16', 3, 2999.97),
(3, 3, '2025-01-17', 10, 1999.90),
(4, 4, '2025-01-18', 2, 799.98),
(5, 5, '2025-01-19', 6, 899.94),
(6, 6, '2025-01-20', 1, 549.99),
(7, 7, '2025-01-21', NULL, 499.95),
(8, 8, '2025-01-22', 4, NULL),
(9, 9, '2025-01-23', 3, 1799.97),
(10, 10, '2025-01-24', 7, 209.93);

-- To verify the inserted records:
SELECT * FROM Sales_dtls;
```

Output:

```
sale_id | item_id | sale_date  | quantity | total_amount
--------------------------------------------------------
1      | 1      | 2025-01-15 | 5       | 3499.95
2      | 2      | 2025-01-16 | 3       | 2999.97
3      | 3      | 2025-01-17 | 10      | 1999.90
4      | 4      | 2025-01-18 | 2       | 799.98
5      | 5      | 2025-01-19 | 6       | 899.94
6      | 6      | 2025-01-20 | 1       | 549.99
7      | 7      | 2025-01-21 | NULL    | 499.95
8      | 8      | 2025-01-22 | 4       | NULL
9      | 9      | 2025-01-23 | 3       | 1799.97
10     | 10     | 2025-01-24 | 7       | 209.93
```

109. Create a table manufacturers and insert at least 10 records including 2 with null values

```
CREATE TABLE manufacturers (
    manufacturer_id INT PRIMARY KEY,
    manufacturer_name VARCHAR(100),
    country VARCHAR(50),
    contact_email VARCHAR(100)
);
```

```
INSERT INTO manufacturers (manufacturer_id, manufacturer_name, country, contact_email)
VALUES
(1, 'Samsung Electronics', 'South Korea', 'contact@samsung.com'),
(2, 'Apple Inc.', 'USA', 'support@apple.com'),
(3, 'Sony Corporation', 'Japan', 'info@sony.com'),
(4, 'Dell Technologies', 'USA', NULL),
(5, 'Canon Inc.', 'Japan', 'sales@canon.com'),
(6, 'JBL', 'USA', 'contact@jbl.com'),
(7, 'Seagate Technology', 'USA', NULL),
(8, 'LG Electronics', 'South Korea', 'support@lg.com'),
(9, 'Logitech', 'Switzerland', 'info@logitech.com'),
(10, 'Amazon', 'USA', 'service@amazon.com');

-- To verify the inserted records:
SELECT * FROM manufacturers;
```

Output:

```
manufacturer_id | manufacturer_name    | country      | contact_email
------------------------------------------------------------------------
1               | Samsung Electronics  | South Korea  | contact@samsung.com
2               | Apple Inc.           | USA          | support@apple.com
3               | Sony Corporation     | Japan        | info@sony.com
4               | Dell Technologies    | USA          | NULL
5               | Canon Inc.           | Japan        | sales@canon.com
6               | JBL                  | USA          | contact@jbl.com
7               | Seagate Technology   | USA          | NULL
8               | LG Electronics       | South Korea  | support@lg.com
9               | Logitech             | Switzerland  | info@logitech.com
10              | Amazon               | USA          | service@amazon.com
```

110. Fetch all clerks information

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;


public class FetchClerks {

  public static void main(String[] args) {

    String url = "jdbc:mysql://localhost:3306/your_database";  // replace with your DB
details

    String user = "your_username";

    String password = "your_password";


    String query = "SELECT * FROM EMP WHERE job = 'CLERK'";


    System.out.println("Neeraj Yadav");

    System.out.println("Enrollment: 0873CS231073");


    try (Connection con = DriverManager.getConnection(url, user, password);

       Statement stmt = con.createStatement();

       ResultSet rs = stmt.executeQuery(query)) {


      System.out.printf("%-10s %-15s %-10s %-10s %-15s %-10s %-10s%n",

          "emp_id", "emp_name", "job", "mgr", "hire_date", "salary", "dept_id");


      while (rs.next()) {

        System.out.printf("%-10d %-15s %-10s %-10d %-15s %-10.2f %-10d%n",
                    rs.getInt("emp_id"),
                    rs.getString("emp_name"),
                    rs.getString("job"),
                    rs.getInt("mgr"),
                    rs.getDate("hire_date").toString(),
                    rs.getDouble("salary"),
                    rs.getInt("dept_id"));
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Neeraj Yadav
Enrollment: 0873CS231073
emp_id   emp_name    job    mgr    hire_date    salary   dept_id
101     John Doe    CLERK   100    2023-05-12   2500.00   10
107     Jane Smith   CLERK   103    2024-02-20   2700.00   20
```

112. Display product details manufactured in the current year only

```
-- Assuming PROD_DTLS has a column 'manufacture_date' of type DATE

SELECT * FROM PROD_DTLS
WHERE YEAR(manufacture_date) = YEAR(CURDATE());
```

Output:

prod_id	prod_name	prod_category	price	manufacture_date
101	Smartphone	Electronics	699.99	2025-03-15
103	Wireless Mouse	Accessories	29.99	2025-06-10

Neeraj Yadav
Enrollment: 0873CS231073

113. Get the details of customers' accounts who opened the accounts before this year

```
SELECT *

FROM CUST_Act_DTLS

WHERE opened_date < DATE_FORMAT(CURDATE(), '%Y-01-01');
```

Output:

act_id	cust_id	act_type_id	balance	opened_date
201	101	1	5000.00	2023-11-15
202	102	2	15000.50	2024-05-10

Neeraj Yadav
Enrollment: 0873CS231073

114. Get all SALARY account details

```
SELECT cad.*

FROM CUST_Act_DTLS cad

JOIN ACT_TYPES_INFO ati ON cad.act_type_id = ati.act_type_id

WHERE ati.act_type_name = 'SALARY';
```

Output:

act_id	cust_id	act_type_id	balance	opened_date
301	105	3	25000.00	2024-01-10
305	108	3	18000.50	2023-08-05

Neeraj Yadav
Enrollment: 0873CS231073

115. Display customer names and mobile numbers from the city 'Texas'

```
SELECT cust_name, cust_phone

FROM CUST_DTLS

WHERE cust_address LIKE '%Texas%';
```

Output:

cust_name	cust_phone
John Carter	9876543210
Emily Watson	9123456789

Neeraj Yadav
Enrollment: 0873CS231073

116. Get the information of Trading account

```
SELECT cad.*

FROM CUST_Act_DTLS cad

JOIN ACT_TYPES_INFO ati ON cad.act_type_id = ati.act_type_id

WHERE ati.act_type_name = 'TRADING';
```

Output:

act_id	cust_id	act_type_id	balance	opened_date
401	110	4	32000.00	2023-09-18
405	115	4	15000.75	2024-02-12

Neeraj Yadav
Enrollment: 0873CS231073

117. Display only Expired product details

```
SELECT *

FROM PROD_DTLS

WHERE exp < SYSDATE;
```

Output:

prod_id	prod_name	prod_category	price	manufacture_date	exp
204	Earphones	Electronics	49.99	2022-03-10	2024-12-31
208	USB Drive	Accessories	15.00	2021-07-05	2023-09-30

Neeraj Yadav
Enrollment: 0873CS231073

CORE JAVA Concepts:

118. Precedence

Operator Precedence in Java determines the order in which operators are evaluated in an expression.

Example:

```
int result = 10 + 2 * 5;

System.out.println(result);  // Output: 20
```

🔷 Multiplication (*) has higher precedence than addition (+), so 2 * 5 is evaluated first.

Precedence Order (Top Few):

1. () – Parentheses
2. ++, -- – Unary
3. *, /, %
4. +, -
5. Relational: <, >, <=, >=
6. Equality: ==, !=
7. Logical AND: &&
8. Logical OR: ||
9. Assignment: =, +=, -=, etc.

119. Data types

Java is a **strongly typed language**, and every variable must have a declared data type.

Primitive Data Types:

- Integer types: byte, short, int, long
- Floating-point: float, double
- Character: char
- Boolean: boolean

Example:

```
int age = 25;

double salary = 45000.50;

char grade = 'A';

boolean isActive = true;
```

Non-Primitive (Reference) Data Types:

- String, Arrays, Classes, Interfaces, etc.

120. Operators

Java provides several categories of operators:

- **Arithmetic**: +, -, *, /, %
- **Relational**: ==, !=, >, <, >=, <=
- **Logical**: &&, ||, !
- **Assignment**: =, +=, -=, *=, etc.
- **Unary**: ++, --, +, -
- **Bitwise**: &, |, ^, ~, <<, >>
- **Ternary**: condition ? true : false

Example:

```
int a = 10, b = 20;

System.out.println(a > b ? a : b);  // Output: 20
```

121. Class

A **class** in Java is a blueprint for objects. It contains variables (fields) and methods.

Example:

```
public class Student {
    int id;
    String name;

    void display() {
        System.out.println(id + " " + name);
    }

    public static void main(String[] args) {
        Student s1 = new Student();
        s1.id = 101;
        s1.name = "Neeraj";
        s1.display();  // Output: 101 Neeraj
    }
}
```

122. Pattern Printing

Often used in logic-building interviews or assignments.

Example: Print a pyramid pattern

```
public class Pattern {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
*
* *
* * *
* * * *
* * * * *
```

123. static keyword

The static keyword means the member belongs to the class, not instances.

Usage:

- Static variable (shared by all objects)
- Static method (can be called without object)
- Static block (runs once when class is loaded)

Example:

```
public class Example {
    static int count = 0;

    static void show() {
        System.out.println("Static method");
    }

    public static void main(String[] args) {
        Example.show();  // No object needed
        System.out.println("Count: " + count);
    }
}
```

124. Constructor

A **constructor** is a special method used to initialize objects. It has the same name as the class and no return type.

Example:

```
public class Car {
  String model;

  Car() {
    model = "Default Model";
  }

  void display() {
    System.out.println("Model: " + model);
  }

  public static void main(String[] args) {
    Car c = new Car();
    c.display();  // Output: Model: Default Model
  }
}
```