SAVEETHA SCHOOL OF ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CSA09 –JAVA PROGRAMMING medium programs

NAME :-KOVURU MAHESH NAIDU
REG NO:- 192224085

1. Write a program to count all the prime and composite numbers entered by the user.
    Sample Output:
        Composite number:3
        Prime number:5

```
import java.util.Scanner;

public class PrimeCompositeCounter {
    public static boolean isPrime(int num) {
        if (num < 2) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static void classifyNumbers(int[] numbers) {
        int primeCount = 0;
        int compositeCount = 0;

        for (int num : numbers) {
            if (num < 0) {
                continue;  // Ignore negative numbers
            } else if (num == 0 || num == 1) {
                compositeCount++;
            } else if (isPrime(num)) {
                primeCount++;
            } else {
                compositeCount++;
            }
        }

        System.out.println("Prime number: " + primeCount);
        System.out.println("Composite number: " + compositeCount);
    }

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

        try {
            System.out.print("Enter the numbers separated by spaces: ");
            String[] input = scanner.nextLine().split("\\s+");
            int[] numbers = new int[input.length];

            for (int i = 0; i < input.length; i++) {
                numbers[i] = Integer.parseInt(input[i]);
            }

            classifyNumbers(numbers);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please enter valid integers.");
        } finally {
            scanner.close();
        }
    }
}
```

2. Find the M^th^ maximum number and N^th^ minimum number in an array and then find the sum of it and difference of it.

    Sample Input:
    Array of elements = {14, 16, 87, 36, 25, 89, 34}
    M = 1
    N = 3

    Sample Output:
    1st Maximum Number = 89
    3rd Minimum Number = 25
    Sum = 114
    Difference = 64

```
import java.util.Arrays;

public class MaxMinSumDifference {
    public static void findMaxMinSumDifference(int[] array, int M, int N) {
        Arrays.sort(array);

        int arrayLength = array.length;

        if (M > 0 && M <= arrayLength && N > 0 && N <= arrayLength) {
            int mthMax = array[arrayLength - M];
            int nthMin = array[N - 1];

            System.out.println(M + "th Maximum Number = " + mthMax);
            System.out.println(N + "th Minimum Number = " + nthMin);

            int sum = mthMax + nthMin;
            int difference = Math.abs(mthMax - nthMin);

            System.out.println("Sum = " + sum);
            System.out.println("Difference = " + difference);
        } else {
            System.out.println("Invalid values for M or N. Please provide valid values within
the array length.");
        }
    }

    public static void main(String[] args) {
        int[] array1 = {14, 16, 87, 36, 25, 89, 34};
        int M1 = 1;
        int N1 = 3;

        int[] array2 = {16, 16, 16, 16, 16};
        int M2 = 0;
        int N2 = 1;

        int[] array3 = {0, 0, 0, 0};
        int M3 = 1;
        int N3 = 2;

        int[] array4 = {-12, -78, -35, -42, -85};
        int M4 = 3;
        int N4 = 3;

        int[] array5 = {15, 19, 34, 56, 12};
        int M5 = 6;
        int N5 = 3;

        int[] array6 = {85, 45, 65, 75, 95};
        int M6 = 5;
        int N6 = 7;

        findMaxMinSumDifference(array1, M1, N1);
        findMaxMinSumDifference(array2, M2, N2);
        findMaxMinSumDifference(array3, M3, N3);
        findMaxMinSumDifference(array4, M4, N4);
        findMaxMinSumDifference(array5, M5, N5);
        findMaxMinSumDifference(array6, M6, N6);
    }
}
```

3. Write a program to print the total amount available in the ATM machine with the conditions applied.
   Total denominations are 2000, 500, 200, 100, get the denomination priority from the user and the total number of notes from the user to display the total available balance to the user
   Sample Input:

Enter the 1st Denomination: 500
Enter the 1st Denomination number of notes: 4
Enter the 2nd Denomination: 100
Enter the 2nd Denomination number of notes: 20
Enter the 3rd Denomination: 200
Enter the 3rd Denomination number of notes: 32
Enter the 4th Denomination: 2000
Enter the 4th Denomination number of notes: 1
Sample Output:
Total Available Balance in ATM: 12400

```
import java.util.Scanner;

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

        int[] denominations = {2000, 500, 200, 100};
        int totalAvailableBalance = 0;

        for (int i = 0; i < denominations.length; i++) {
            System.out.print("Enter the " + (i + 1) + " Denomination: ");
            int denomination = scanner.nextInt();

            if (denomination != denominations[i]) {
                System.out.println("Invalid Denomination. Please enter the correct
denomination.");
                return;
            }

            System.out.print("Enter the " + (i + 1) + " Denomination number of notes: ");
            int numOfNotes = scanner.nextInt();

            totalAvailableBalance += denomination * numOfNotes;
        }

        System.out.println("Total Available Balance in ATM: " + totalAvailableBalance);

        scanner.close();
    }
}
```

4. Write a program using choice to check
    Case 1: Given string is palindrome or not
    Case 2: Given number is palindrome or not
    Sample Input:
    Case = 1
    String = MADAM
    Sample Output:

Palindrome

```
import java.util.Scanner;

public class PalindromeChecker {
    public static boolean isPalindrome(String str) {
        str = str.toLowerCase().replaceAll("[^a-zA-Z0-9]", "");
        int left = 0;
        int right = str.length() - 1;

        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }

        return true;
    }

    public static boolean isPalindrome(int num) {
        int originalNum = num;
        int reversedNum = 0;

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

        return originalNum == reversedNum;
    }

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

        System.out.println("Choose an option:");
        System.out.println("Case 1: Check if a given string is palindrome or not");
        System.out.println("Case 2: Check if a given number is palindrome or not");

        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                System.out.print("Enter the string: ");
                String inputString = scanner.next();
                if (isPalindrome(inputString)) {
                    System.out.println("Palindrome");
                } else {
                    System.out.println("Not a Palindrome");
                }
                break;

            case 2:
                System.out.print("Enter the number: ");
                int inputNumber = scanner.nextInt();
                if (isPalindrome(inputNumber)) {
                    System.out.println("Palindrome");
                } else {
                    System.out.println("Not a Palindrome");
                }
                break;

            default:
                System.out.println("Invalid choice. Please choose either 1 or 2.");
                break;
        }

        scanner.close();
    }
}
```

5. Write a program to convert Decimal number equivalent to Binary number and octal numbers?
   Sample Input:
   Decimal Number: 15
   Sample Output:
   Binary Number = 1111
   Octal = 17

```
import java.util.Scanner;

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

        System.out.print("Enter the Decimal Number: ");
        if (scanner.hasNextInt()) {
            int decimalNumber = scanner.nextInt();

            if (decimalNumber < 0) {
                System.out.println("Please enter a non-negative integer.");
            } else {
                String binaryNumber = Integer.toBinaryString(decimalNumber);
                String octalNumber = Integer.toOctalString(decimalNumber);

                System.out.println("Binary Number = " + binaryNumber);
                System.out.println("Octal Number = " + octalNumber);
            }
        } else {
            System.out.println("Invalid input. Please enter a valid integer.");
        }

        scanner.close();
    }
}
```

6. In an organization they decide to give bonus to all the employees on New Year. A 5% bonus on salary is given to the grade A workers and 10% bonus on salary to the grade B workers. Write a program to enter the salary and grade of the employee. If the salary of the employee is less than $10,000 then the employee gets an extra 2% bonus on salary Calculate the bonus that has to be given to the employee and print the salary that the employee will get.

```
Sample Input & Output:
Enter the grade of the employee: B
Enter the employee salary: 50000
Salary=50000
Bonus=5000.0
Total to be paid:55000.0
```

```
import java.util.Scanner;

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

        System.out.print("Enter the grade of the employee: ");
        char grade = scanner.next().toUpperCase().charAt(0);

        System.out.print("Enter the employee salary: ");
        double salary = scanner.nextDouble();

        double bonusPercentage;
        double bonus;

        switch (grade) {
            case 'A':
                bonusPercentage = (salary < 10000) ? 7.0 : 5.0;
                break;
            case 'B':
                bonusPercentage = (salary < 10000) ? 12.0 : 10.0;
                break;
            default:
                System.out.println("Invalid grade. Please enter A or B.");
                return;
        }

        bonus = (bonusPercentage / 100) * salary;
        double totalSalary = salary + bonus;

        System.out.println("Salary = " + salary);
        System.out.println("Bonus = " + bonus);
        System.out.println("Total to be paid: " + totalSalary);
    scanner.close();
  }
}
```

7. Write a program to print the first n perfect numbers. (Hint Perfect number means **a positive integer that is equal to the sum of its proper divisors)**
   Sample Input:
   N = 3
   Sample Output:
   First 3 perfect numbers are: 6 , 28 , 496

```
import java.util.Scanner;

public class PerfectNumberGenerator {
    public static boolean isPerfectNumber(int number) {
        if (number <= 1) {
            return false;
        }

        int sum = 1;
        for (int i = 2; i * i <= number; i++) {
            if (number % i == 0) {
                sum += i;
                if (i != number / i) {
                    sum += number / i;
                }
            }
        }

        return sum == number;
    }

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

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

        if (n <= 0) {
            System.out.println("Please enter a positive integer for N.");
            return;
        }

        System.out.print("First " + n + " perfect numbers are: ");
        int count = 0;
        for (int i = 1; count < n; i++) {
            if (isPerfectNumber(i)) {
                System.out.print(i + " ");
                count++;
            }
        }

        scanner.close();
    }
}
```

8. Write a program to enter the marks of a student in four subjects. Then calculate the total and aggregate, display the grade obtained by the student. If the student scores an aggregate greater than 75%, then the grade is Distinction. If aggregate is 60>= and <75, then the grade is First Division. If aggregate is 50 >= and <60, then the grade is Second Division. If aggregate is 40>= and <50, then the grade is Third Division. Else the grade is Fail.

   Sample Input & Output:

   ```
   Enter the marks in python: 90
   Enter the marks in c programming: 91
   Enter the marks in Mathematics: 92
   Enter the marks in Physics: 93
   Total= 366
   Aggregate = 91.5
   DISTINCTION
   ```

```
import java.util.Scanner;

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

        System.out.print("Enter the marks in Python: ");
        double pythonMarks = scanner.nextDouble();

        System.out.print("Enter the marks in C Programming: ");
        double cProgrammingMarks = scanner.nextDouble();

        System.out.print("Enter the marks in Mathematics: ");
        double mathematicsMarks = scanner.nextDouble();

        System.out.print("Enter the marks in Physics: ");
        double physicsMarks = scanner.nextDouble();

        if (pythonMarks < 0 || cProgrammingMarks < 0 || mathematicsMarks < 0 ||
physicsMarks < 0 ||
            pythonMarks > 100 || cProgrammingMarks > 100 || mathematicsMarks > 100 ||
physicsMarks > 100) {
            System.out.println("Invalid marks. Please enter valid marks between 0 and
100.");
            return;
        }
```

```
        double total = pythonMarks + cProgrammingMarks + mathematicsMarks +
physicsMarks;
        double aggregate = total / 4.0;

        System.out.println("Total= " + total);
        System.out.println("Aggregate = " + aggregate);

        if (aggregate > 75) {
            System.out.println("DISTINCTION");
        } else if (aggregate >= 60 && aggregate < 75) {
            System.out.println("FIRST DIVISION");
        } else if (aggregate >= 50 && aggregate < 60) {
            System.out.println("SECOND DIVISION");
        } else if (aggregate >= 40 && aggregate < 50) {
            System.out.println("THIRD DIVISION");
        } else {
            System.out.println("FAIL");
        }

        scanner.close();
    }
}
```

9. Write a program to read the numbers until -1 is encountered. Find the average of positive numbers and negative numbers entered by user.

   Sample Input:

```
Enter -1 to exit…
Enter the number: 7
Enter the number: -2
Enter the number: 9
Enter the number: -8
Enter the number: -6
Enter the number: -4
Enter the number: 10
Enter the number:  -1
```

   Sample Output:

```
The average of negative numbers is: -5.0
The average of positive numbers is : 8.66666667
```

```
import java.util.Scanner;

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

        int positiveSum = 0;
        int positiveCount = 0;
        int negativeSum = 0;
        int negativeCount = 0;

        System.out.println("Enter -1 to exit...");
        while (true) {
            System.out.print("Enter the number: ");
            int number = scanner.nextInt();

            if (number == -1) {
                break;
            }

            if (number >= 0) {
                positiveSum += number;
                positiveCount++;
            } else {
                negativeSum += number;
                negativeCount++;
            }
        }

        if (positiveCount > 0) {
            double positiveAverage = (double) positiveSum / positiveCount;
            System.out.println("The average of positive numbers is: " + positiveAverage);
        } else {
            System.out.println("No positive numbers entered.");
        }

        if (negativeCount > 0) {
            double negativeAverage = (double) negativeSum / negativeCount;
            System.out.println("The average of negative numbers is: " +
negativeAverage);
        } else {
            System.out.println("No negative numbers entered.");
        }

        scanner.close();
    }
}
```

10. Write a program to read a character until a ***** is encountered. Also count the number of uppercase, lowercase, and numbers entered by the users.

```
Sample Input:
Enter * to exit…
Enter any character: W
Enter any character: d
Enter any character: A
Enter any character: G
Enter any character: g
Enter any character: H
Enter any character: *
Sample Output:
Total count of lower case:2
Total count of upper case:4
```

Total count of numbers =0

```
import java.util.Scanner;

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

        int lowercaseCount = 0;
        int uppercaseCount = 0;
        int numberCount = 0;

        System.out.println("Enter * to exit...");
        while (true) {
            System.out.print("Enter any character: ");
            char character = scanner.next().charAt(0);

            if (character == '*') {
                break;
            }

            if (Character.isLowerCase(character)) {
                lowercaseCount++;
            } else if (Character.isUpperCase(character)) {
                uppercaseCount++;
            } else if (Character.isDigit(character)) {
                numberCount++;
            }
        }

        System.out.println("Total count of lowercase: " + lowercaseCount);
        System.out.println("Total count of uppercase: " + uppercaseCount);
        System.out.println("Total count of numbers: " + numberCount);

        scanner.close();
    }
}
```

11. Write a program to calculate the factorial of number using recursive function.

Sample Input & Output:

```
Enter the value of n: 6
```

Sample Input & Output:

```
The factorial of 6 is: 720
```

```
import java.util.Scanner;

public class FactorialCalculator {
    public static long calculateFactorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        } else {
                return n * calculateFactorial(n - 1);
            }
        }

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

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

            if (n < 0) {
                System.out.println("Please enter a non-negative integer.");
            } else {
                long factorial = calculateFactorial(n);
                System.out.println("The factorial of " + n + " is: " + factorial);
            }

            scanner.close();
        }
}
```

12. Write a Program to Find the Nth Largest Number in a array.

Sample Input:

List : {14, 67, 48, 23, 5, 62}

N = 4

Sample Output:

4th Largest number: 23

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

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

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

        if (size <= 0) {
            System.out.println("Please enter a valid array size.");
            return;
        }

        int[] array = new int[size];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < size; i++) {
            System.out.print("Element " + (i + 1) + ": ");
            array[i] = scanner.nextInt();
        }

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

        if (N <= 0 || N > size) {
            System.out.println("Invalid value of N. N should be between 1 and the array size.");
            return;
        }

        Arrays.sort(array);

        int nthLargest = array[size - N];
        System.out.println(N + "th Largest number: " + nthLargest);

        scanner.close();
    }
}
```

13. Write a program to convert the Binary to Decimal, Octal

Sample Input:

Given Number: 1101

Sample Output:

Decimal Number: 13

Octal:15

```
import java.util.Scanner;

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

        System.out.print("Enter the binary number: ");
        String binaryNumber = scanner.nextLine();

        if (isValidBinary(binaryNumber)) {
            int decimal = convertBinaryToDecimal(binaryNumber);
            String octal = convertDecimalToOctal(decimal);

            System.out.println("Decimal Number: " + decimal);
            System.out.println("Octal: " + octal);
        } else {
            System.out.println("Invalid binary number. Please enter a valid binary number.");
        }

        scanner.close();
    }

    private static boolean isValidBinary(String binaryNumber) {
        return binaryNumber.matches("[01]+");
    }
    private static int convertBinaryToDecimal(String binaryNumber) {
        return Integer.parseInt(binaryNumber, 2);
    }

    private static String convertDecimalToOctal(int decimalNumber) {
        return Integer.toOctalString(decimalNumber);
    }
}
```

14. Write a program to find the number of special characters in the given statement
    Sample Input:
    Given statement: Modi Birthday @ September 17, #&$% is the wishes code for him.
    Sample Output:
    Number of special Characters: 5

```
import java.util.Scanner;

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

        System.out.print("Enter the statement: ");
        String statement = scanner.nextLine();

        int specialCharacterCount = countSpecialCharacters(statement);

        System.out.println("Number of special characters: " + specialCharacterCount);

        scanner.close();
    }

    private static int countSpecialCharacters(String statement) {
        int count = 0;

        for (char ch : statement.toCharArray()) {
            if (!Character.isLetterOrDigit(ch) && !Character.isWhitespace(ch)) {
                count++;
            }
        }

        return count;
    }
}
```

15. Write a Program to Remove the Duplicate Items from a array.
    Sample Input:
        Enter the number of elements in array:7
        Enter element1:10
        Enter element2:20
        Enter element3:20

```
        Enter element4:30
        Enter element5:40
        Enter element6:40
        Enter element7:50
Sample Output:
        Non-duplicate items:
        [10, 20, 30, 40, 50]
```

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

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

        System.out.print("Enter the number of elements in the array: ");
        int size = scanner.nextInt();

        if (size <= 0) {
            System.out.println("Please enter a valid array size.");
            return;
        }

        int[] array = new int[size];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < size; i++) {
            System.out.print("Enter element" + (i + 1) + ": ");
            array[i] = scanner.nextInt();
        }

        int[] nonDuplicateArray = removeDuplicates(array);

        System.out.println("Non-duplicate items:");
        System.out.println(Arrays.toString(nonDuplicateArray));

        scanner.close();
    }

    private static int[] removeDuplicates(int[] array) {
        Set<Integer> uniqueSet = new HashSet<>();
        for (int num : array) {
            uniqueSet.add(num);
        }

        int[] nonDuplicateArray = new int[uniqueSet.size()];
        int index = 0;
        for (int num : uniqueSet) {
            nonDuplicateArray[index++] = num;
        }

        return nonDuplicateArray;
    }
}
```

16. Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest. Write a Java program for above scenario.
    Sample Input SBI, 8.4
    Sample Output
    Test case
    1. SBI, 8.3
    2. ICICI, 7.3
    3. AXIS, 9.7
    4. SBI, 8.6
    5. AXIX, 7.6

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

class Bank {
    private static final Map<String, Double> interestRates = new HashMap<>();

    static {
        interestRates.put("SBI", 8.4);
        interestRates.put("ICICI", 7.3);
        interestRates.put("AXIS", 9.7);
    }

    public static double getInterestRate(String bankName) {
        return interestRates.getOrDefault(bankName, 0.0);
    }
}

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

        System.out.println("Enter the bank name and get the interest rate:");
        System.out.print("Enter bank name: ");
        String bankName = scanner.next();

        double interestRate = Bank.getInterestRate(bankName);

        if (interestRate != 0.0) {
            System.out.println("Interest rate for " + bankName + ": " + interestRate);
        } else {
            System.out.println("Invalid bank name. Please enter a valid bank name.");
        }

        scanner.close();
    }
}
```

17. Bring out the situation in which member names of a subclass hide members by the same name in the super class. How it can be resolved? Write Suitable code in Java and Implement above scenario with the Parametrized Constructor (accept int type parameter) of the Super Class can be called from Sub Class Using super () and display the input values provided.

    Sample Input : 100, 200
    Sample Output : 100, 200
    Test Cases
    1. 10, 20
    2. -20, -30
    3. 0, 0
    4. EIGHT FIVE
    5. 10.57, 12.58

```
class SuperClass {
    int value;

    public SuperClass(int value) {
        this.value = value;
    }

    void display() {
        System.out.println("SuperClass value: " + value);
    }
}

class SubClass extends SuperClass {
    int value; // Hides the value in SuperClass

    public SubClass(int superValue, int subValue) {
        super(superValue); // Call the parameterized constructor of SuperClass
        this.value = subValue;
    }

    void display() {
        super.display(); // Call the display method of SuperClass
        System.out.println("SubClass value: " + value);
    }
}

public class VariableHidingExample {
    public static void main(String[] args) {
        SubClass subObj = new SubClass(100, 200);
        subObj.display();
        // Additional Test Cases
        SubClass test1 = new SubClass(10, 20);
        test1.display();

        SubClass test2 = new SubClass(-20, -30);
        test2.display();

        SubClass test3 = new SubClass(0, 0);
        test3.display();

        // Uncommenting the following line will result in a compilation error for invalid
    input
        // SubClass test4 = new SubClass("EIGHT", "FIVE");

        SubClass test5 = new SubClass(10, 20);
        test5.display();
      }
}
```

18. Display Multiplication table for 5 and 10 using various stages of life cycle of the thread by generating a suitable code in Java.

    Sample Input 5, 10
    5 X 1 = 5
    5 X 2 =10
     ….
    10 X 1 =10
    10 X 2 = 20
    ….

```
class MultiplicationTableThread extends Thread {
  private int number;

  public MultiplicationTableThread(int number) {
    this.number = number;
  }

  @Override
  public void run() {
    System.out.println("Multiplication table for " + number + ":");
    for (int i = 1; i <= 10; i++) {
      System.out.println(number + " X " + i + " = " + (number * i));
      try {
        Thread.sleep(500); // Simulating some processing time
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}
```

```
public class ThreadLifeCycleExample {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 10;

        // Creating threads for displaying multiplication tables
        MultiplicationTableThread thread1 = new MultiplicationTableThread(num1);
        MultiplicationTableThread thread2 = new MultiplicationTableThread(num2);

        // Starting the threads
        thread1.start();
        thread2.start();

        // Main thread continues execution
        for (int i = 1; i <= 5; i++) {
            System.out.println("Main thread is running: " + i);
            try {
                Thread.sleep(1000); // Simulating some processing time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        // Waiting for the threads to finish
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Main thread finished.");
    }
}
```

19. Using the concepts of thread with implementing Runnable interface in Java to generate Fibonacci series.

    Sample Input : 5

    Sample Output : 0 1 1 2 3 …..

    Test Cases

    1. 7
    2. -10
    3. 0
    4. EIGHT FIVE
    5. 12.65

```
class FibonacciGenerator implements Runnable {
    private int count;

    public FibonacciGenerator(int count) {
        this.count = count;
    }

    @Override
    public void run() {
        int first = 0, second = 1;
        System.out.println("Fibonacci series:");

        for (int i = 0; i < count; i++) {
            System.out.print(first + " ");

            int temp = first + second;
            first = second;
            second = temp;

            try {
                Thread.sleep(500); // Simulating some processing time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class FibonacciThreadExample {
    public static void main(String[] args) {
        int n = 5; // Default value

        try {
            if (args.length > 0) {
                n = Integer.parseInt(args[0]);
                if (n < 0) {
                    throw new IllegalArgumentException("Please enter a non-negative
integer.");
                }
            }
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please enter a valid integer.");
            return;
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
            return;
        }

        Thread fibonacciThread = new Thread(new FibonacciGenerator(n));
        fibonacciThread.start();

        // Main thread continues execution
        for (int i = 1; i <= 5; i++) {
            System.out.println("Main thread is running: " + i);
            try {
                Thread.sleep(1000); // Simulating some processing time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        // Waiting for the Fibonacci thread to finish
        try {
            fibonacciThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Main thread finished.");
    }
}
```

20. Generate a Java code to find the sum of N numbers using array and throw ArrayIndexOutOfBoundsException when the loop variable beyond the size N.

    Sample Input : 5

    1 2 3 4 5

    Sample Output : 15

```
import java.util.Scanner;

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

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

            if (N <= 0) {
                throw new IllegalArgumentException("N should be a positive integer.");
            }

            int[] numbers = new int[N];
            System.out.println("Enter " + N + " numbers:");

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

            int sum = calculateSum(numbers);
            System.out.println("Sum of the numbers: " + sum);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException: Loop variable
beyond the size N.");
```

```
        } catch (Exception e) {
            System.out.println("Invalid input. Please enter a valid integer.");
        } finally {
            scanner.close();
        }
    }

    private static int calculateSum(int[] numbers) {
        int sum = 0;

        for (int i = 0; i <= numbers.length; i++) {
            sum += numbers[i];
        }

        return sum;
    }
}
```

21. Using the concepts of thread with implementing Runnable interface in Java to find whether a given number is prime or not.

    Sample Input : 5

    Sample Output : 5 is Prime

    Sample Output : 15

```
class PrimeChecker implements Runnable {
    private int number;

    public PrimeChecker(int number) {
        this.number = number;
    }

    @Override
    public void run() {
        if (isPrime(number)) {
            System.out.println(number + " is Prime.");
        } else {
            System.out.println(number + " is not Prime.");
        }
    }

    private 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;
    }
}

public class PrimeThreadExample {
    public static void main(String[] args) {
        int n = 5; // Default value

        try {
            if (args.length > 0) {
                n = Integer.parseInt(args[0]);
                if (n < 0) {
                    throw new IllegalArgumentException("Please enter a non-negative
integer.");
                }
            }
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please enter a valid integer.");
            return;
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
            return;
        }

        Thread primeThread = new Thread(new PrimeChecker(n));
        primeThread.start();

        // Main thread continues execution
        for (int i = 1; i <= 5; i++) {
            System.out.println("Main thread is running: " + i);
            try {
                Thread.sleep(1000); // Simulating some processing time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        // Waiting for the Prime thread to finish
        try {
            primeThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Main thread finished.");
    }
}
```

22. Generate a Java code to handle Exceptions such as Arithmetic Exception, ArrayIndexOutOfBoundsException, NullPointerException using Multi-Catch Statements.

```
public class MultiCatchExample {
    public static void main(String[] args) {
        try {
            // Arithmetic Exception
            int result = divideByZero(10, 0);
            System.out.println("Result: " + result);

            // ArrayIndexOutOfBoundsException
            int[] numbers = { 1, 2, 3 };
            accessArrayElement(numbers, 5);

            // NullPointerException
            String str = null;
            printStringLength(str);
        } catch (ArithmeticException | ArrayIndexOutOfBoundsException |
NullPointerException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    private static int divideByZero(int numerator, int denominator) {
        return numerator / denominator;
    }

    private static void accessArrayElement(int[] arr, int index) {
        System.out.println("Array element at index " + index + ": " + arr[index]);
    }

    private static void printStringLength(String str) {
        System.out.println("Length of the string: " + str.length());
    }
}
```

23. Generate a Java Code to Write and Read the string “Computer Science and Engineering” using FileWriter and FileReader Class.

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class FileReadWriteExample {
    public static void main(String[] args) {
        String textToWrite = "Computer Science and Engineering";
        String fileName = "sample.txt";

        // Writing to file using FileWriter
        writeToFile(fileName, textToWrite);

        // Reading from file using FileReader
        String readText = readFromFile(fileName);

        System.out.println("Original Text: " + textToWrite);
        System.out.println("Text Read from File: " + readText);
    }
    private static void writeToFile(String fileName, String textToWrite) {
        try (FileWriter fileWriter = new FileWriter(fileName)) {
            fileWriter.write(textToWrite);
            System.out.println("Text written to file successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file: " +
e.getMessage());
        }
    }

    private static String readFromFile(String fileName) {
        StringBuilder content = new StringBuilder();

        try (FileReader fileReader = new FileReader(fileName)) {
            int character;
            while ((character = fileReader.read()) != -1) {
                content.append((char) character);
            }
            System.out.println("Text read from file successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while reading from the file: " +
e.getMessage());
        }

        return content.toString();
    }
}
```

24. Create a java program to construct the volume of Box using default constructor method.

```
import java.util.Scanner;

class Box {
    double length;
    double width;
    double height;

    // Default constructor
    public Box() {
        length = 1.0;
        width = 1.0;
        height = 1.0;
    }

    // Method to calculate volume
    double volume() {
        return length * width * height;
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a Box object using default constructor
        Box myBox = new Box();

        // Calculate and print volume
        System.out.println("Volume of the box: " + myBox.volume());
    }
}
```

25. Accept the string “Welcome to Saveetha university” from the user and perform the following operations by writing a suitable Java code.
    i) Replace any word in the given String
    ii) Find the length
    iii) Uppercase Conversion

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();

        // Replace any word in the given string
        String replacedString = input.replace("Saveetha", "Example");
        System.out.println("Replaced string: " + replacedString);

        // Find the length of the string
        int length = input.length();
        System.out.println("Length of the string: " + length);

        // Uppercase conversion
        String uppercaseString = input.toUpperCase();
        System.out.println("Uppercase conversion: " + uppercaseString);
    }
}
```

26. Create a HashTable to maintain a bank detail which includes Account number and Customer name. Let Account number be the key in the HashTable. Write a Java program to implement the following operations in the HashTable
    i) Add 3 records
    ii) Display the size of HashTable

iii) Clear the HashTable

```
import java.util.Hashtable;
import java.util.Map;

public class BankDetails {
    public static void main(String[] args) {
        // Create a HashTable to maintain bank details
        Hashtable<Integer, String> bankDetails = new Hashtable<>();

        // Add 3 records to the HashTable
        addRecords(bankDetails);

        // Display the size of the HashTable
        displaySize(bankDetails);

        // Clear the HashTable
        clearHashTable(bankDetails);
    }

    private static void addRecords(Hashtable<Integer, String> bankDetails) {
        bankDetails.put(123456, "John Doe");
        bankDetails.put(789012, "Alice Smith");
        bankDetails.put(345678, "Bob Johnson");

        System.out.println("Records added to the HashTable.");
    }

    private static void displaySize(Hashtable<Integer, String> bankDetails) {
        System.out.println("Size of the HashTable: " + bankDetails.size());
    }

    private static void clearHashTable(Hashtable<Integer, String> bankDetails) {
        bankDetails.clear();
        System.out.println("HashTable cleared. Size after clearing: " +
bankDetails.size());
    }
}
```

27. Create a employee record using map interface and do the following operations.

i.	Add object	iii. Remove specified object
ii.	isEmpty or not	iv. Clear

```
import java.util.HashMap;

import java.util.Map;


public class EmployeeRecord {
    public static void main(String[] args) {
        // Create a Map to store employee records (Employee ID as key, Employee
Name as value)
        Map<Integer, String> employeeMap = new HashMap<>();

        // i. Add objects to the map
        addObjects(employeeMap);

        // ii. Check if the map is empty or not
        checkIfEmpty(employeeMap);

        // iii. Remove a specified object
        removeObject(employeeMap, 102);

        // iv. Clear the map
        clearMap(employeeMap);
    }

    private static void addObjects(Map<Integer, String> employeeMap) {
        // Add employee records to the map
        employeeMap.put(101, "John Doe");
        employeeMap.put(102, "Alice Smith");
        employeeMap.put(103, "Bob Johnson");

        System.out.println("Employee records added to the map.");
        System.out.println("Employee Map: " + employeeMap);
    }

    private static void checkIfEmpty(Map<Integer, String> employeeMap) {
        // Check if the map is empty
        boolean isEmpty = employeeMap.isEmpty();
        System.out.println("Is Employee Map empty? " + isEmpty);
    }
    private static void removeObject(Map<Integer, String> employeeMap, int
employeeIdToRemove) {
        // iii. Remove a specified object
        if (employeeMap.containsKey(employeeIdToRemove)) {
            employeeMap.remove(employeeIdToRemove);
            System.out.println("Employee with ID " + employeeIdToRemove + "
removed from the map.");
            System.out.println("Employee Map after removal: " + employeeMap);
        } else {
            System.out.println("Employee with ID " + employeeIdToRemove + " not
found in the map.");
        }
    }

    private static void clearMap(Map<Integer, String> employeeMap) {
        // iv. Clear the map
        employeeMap.clear();
        System.out.println("Employee Map cleared. Size after clearing: " +
employeeMap.size());
    }
}
```

28. Create a simple generics class with type parameters for sorting values of different types.

```
public class GenericSort<T extends Comparable<T>> {
    private T[] array;

    public GenericSort(T[] array) {
        this.array = array;
    }

    public void sort() {
        if (array == null || array.length == 0) {
            System.out.println("Array is empty or null. Nothing to sort.");
            return;
        }

        // Using Bubble Sort for simplicity. You can replace it with any other sorting
algorithm.
        for (int i = 0; i < array.length - 1; i++) {
            for (int j = 0; j < array.length - 1 - i; j++) {
                if (array[j].compareTo(array[j + 1]) > 0) {
                    // Swap elements if they are in the wrong order
                    T temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }

    public void display() {
        System.out.print("Sorted Array: [");
        for (T element : array) {
            System.out.print(element + " ");
        }
        System.out.println("]");
    }

    public static void main(String[] args) {
        // Example of using GenericSort with Integer array
        Integer[] intArray = {5, 2, 8, 1, 7};
        GenericSort<Integer> intSorter = new GenericSort<>(intArray);
        intSorter.sort();
        intSorter.display();

        // Example of using GenericSort with String array
        String[] strArray = {"banana", "apple", "orange", "grape", "kiwi"};
        GenericSort<String> strSorter = new GenericSort<>(strArray);
        strSorter.sort();
        strSorter.display();
    }
}
```

29. Develop a Java code to insert the following elements, using ListIterator to append + symbol in each element and print them in reverse order. {C, A, E, B, D, F}.

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorExample {
    public static void main(String[] args) {
        // Create a list with elements {C, A, E, B, D, F}
        List<String> elements = new ArrayList<>();
        elements.add("C");
        elements.add("A");
        elements.add("E");
        elements.add("B");
        elements.add("D");
        elements.add("F");

        // Use ListIterator to append "+" symbol to each element
        ListIterator<String> iterator = elements.listIterator();
        while (iterator.hasNext()) {
            String originalElement = iterator.next();
            iterator.set(originalElement + "+");
        }

        // Print the elements in reverse order
        System.out.println("Elements after appending \"+\" symbol and printing in reverse
order:");
        while (iterator.hasPrevious()) {
            System.out.print(iterator.previous() + " ");
        }
    }
}
```

30. Generate a Java code to perform simple arithmetic operations and to throw Arithmetic Exception for Division-by-Zero.

```
import java.util.Scanner;

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

        try {
            // Get input numbers from the user
            System.out.print("Enter the first number: ");
            int num1 = scanner.nextInt();

            System.out.print("Enter the second number: ");
            int num2 = scanner.nextInt();

            // Perform arithmetic operations
            int sum = num1 + num2;
            int difference = num1 - num2;
            int product = num1 * num2;

            // Check for Division-by-Zero
            if (num2 == 0) {
                throw new ArithmeticException("Division by zero is not allowed.");
            }

            int quotient = num1 / num2;
            // Display the results
            System.out.println("Sum: " + sum);
            System.out.println("Difference: " + difference);
            System.out.println("Product: " + product);
            System.out.println("Quotient: " + quotient);

        } catch (ArithmeticException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: Invalid input. Please enter valid numbers.");
        } finally {
            scanner.close();
        }
    }
}
```

31. Write a Java program to create three threads in parallel and display the natural numbers in orders using sleep() method.

```
class NumberPrinter implements Runnable {
    private int start;
    private int max;
    private int step;

    public NumberPrinter(int start, int max, int step) {
        this.start = start;
        this.max = max;
        this.step = step;
    }

    @Override
    public void run() {
        for (int i = start; i <= max; i += step) {
            System.out.println(Thread.currentThread().getName() + ": " + i);

            try {
                // Introduce a sleep to simulate some work and allow other threads to run
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ParallelNumberPrinting {
    public static void main(String[] args) {
        // Create three threads for printing numbers
        Thread thread1 = new Thread(new NumberPrinter(1, 10, 3), "Thread 1");
        Thread thread2 = new Thread(new NumberPrinter(2, 10, 3), "Thread 2");
        Thread thread3 = new Thread(new NumberPrinter(3, 10, 3), "Thread 3");

        // Start the threads
        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

32. If n = 8, then array ‘a’ will have 7 elements in the range from 1 to 8. For example {1, 4, 5, 3, 7, 8, 6}. One number will be missing in ‘a’ (2 in this case). Write a source code to find out that missing number

```
import java.util.Arrays;

public class MissingNumberFinder {
    public static void main(String[] args) {
        int n = 8;

        // Create an array with n-1 elements in the range from 1 to n
        int[] a = {1, 4, 5, 3, 7, 8, 6};

        // Find the missing number
        int missingNumber = findMissingNumber(n, a);

        System.out.println("The missing number is: " + missingNumber);
    }

    static int findMissingNumber(int n, int[] a) {
        // Calculate the sum of numbers from 1 to n using the formula n * (n + 1) / 2
        int expectedSum = n * (n + 1) / 2;

        // Calculate the sum of elements in the array
        int actualSum = Arrays.stream(a).sum();

        // The missing number is the difference between the expected sum and the actual
sum
        return expectedSum - actualSum;
    }
}
```

33. Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call
1 - method of parent class by object of parent class

2 - method of child class by object of child class
3 - method of parent class by object of child class

```
class ParentClass {
    void printMessage() {
        System.out.println("This is parent class");
    }
}

class ChildClass extends ParentClass {
    void printMessage() {
        System.out.println("This is child class");
    }
}

public class Main {
    public static void main(String[] args) {
        // 1. Method of parent class by object of parent class
        ParentClass parentObj = new ParentClass();
        parentObj.printMessage();

        // 2. Method of child class by object of child class
        ChildClass childObj = new ChildClass();
        childObj.printMessage();

        // 3. Method of parent class by object of child class
        ParentClass parentObjFromChild = new ChildClass();
        parentObjFromChild.printMessage();
    }
}
```

34. Write a Java program to create a class Student and create constructor which assigns the values for the student details such as student name, register number, and five subject marks. Calculate the total and average of five subject marks and display the marks and average.

```
import java.util.Scanner;

class Student {
    private String studentName;
    private int registerNumber;
    private double[] subjectMarks;
    private double totalMarks;
    private double averageMarks;

    // Constructor to initialize student details
    public Student(String studentName, int registerNumber, double[] subjectMarks) {
        this.studentName = studentName;
        this.registerNumber = registerNumber;
        this.subjectMarks = subjectMarks;
        calculateTotalAndAverage();
    }

    // Method to calculate total and average marks
    private void calculateTotalAndAverage() {
        totalMarks = 0;

        for (double mark : subjectMarks) {
            totalMarks += mark;
        }

        averageMarks = totalMarks / subjectMarks.length;
    }

    // Method to display student details, marks, total, and average
    public void displayStudentDetails() {
        System.out.println("Student Name: " + studentName);
        System.out.println("Register Number: " + registerNumber);

        System.out.print("Subject Marks: ");
        for (double mark : subjectMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();

        System.out.println("Total Marks: " + totalMarks);
        System.out.println("Average Marks: " + averageMarks);
    }
}

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

        // Get student details and marks from the user
        System.out.print("Enter Student Name: ");
        String studentName = scanner.nextLine();

        System.out.print("Enter Register Number: ");
        int registerNumber = scanner.nextInt();

        double[] subjectMarks = new double[5];
        System.out.println("Enter Marks for 5 Subjects:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + ": ");
```

```
                subjectMarks[i] = scanner.nextDouble();
            }

            // Create a Student object
            Student student = new Student(studentName, registerNumber, subjectMarks);

            // Display student details, marks, total, and average
            student.displayStudentDetails();

            scanner.close();
        }
    }
```

35. Generate a code to Count the Number of Words, Character and Lines from the File using Stream I/O in Java.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class FileStatsCounter {
    public static void main(String[] args) {
        String filePath = "path/to/your/file.txt"; // Replace with the actual file path

        try {
            // Read the file using BufferedReader and FileReader
            BufferedReader reader = new BufferedReader(new FileReader(filePath));

            // Use Java Streams to calculate counts
            long wordCount = reader.lines()
                    .flatMap(line -> java.util.Arrays.stream(line.split("\\s+")))
                    .count();

            long characterCount = reader.lines()
                    .flatMapToInt(CharSequence::chars)
                    .count();

            long lineCount = reader.lines().count();

            // Display the counts
            System.out.println("Number of Words: " + wordCount);
            System.out.println("Number of Characters: " + characterCount);
            System.out.println("Number of Lines: " + lineCount);

            // Close the reader
            reader.close();
        } catch (IOException e) {
            System.err.println("Error reading the file: " + e.getMessage());
        }
    }
}
```

36. Generate a code to non-negative integer’s num1 and num2 represented as strings; return the product of num1 and num2, also represented as a string.

```
public class MultiplyStrings {
    public static String multiply(String num1, String num2) {
        int m = num1.length();
        int n = num2.length();
        int[] result = new int[m + n];

        // Multiply each digit and add to the result array
        for (int i = m - 1; i >= 0; i--) {
            for (int j = n - 1; j >= 0; j--) {
                int mul = (num1.charAt(i) - '0') * (num2.charAt(j) - '0');
                int sum = mul + result[i + j + 1];

                result[i + j] += sum / 10;   // Carry
                result[i + j + 1] = sum % 10;
            }
        }

        // Convert result array to a string
        StringBuilder sb = new StringBuilder();
        for (int digit : result) {
            if (!(sb.length() == 0 && digit == 0)) {
                sb.append(digit);
            }
        }

        return sb.length() == 0 ? "0" : sb.toString();
    }

    public static void main(String[] args) {
        String num1 = "123";
        String num2 = "456";

        String product = multiply(num1, num2);
        System.out.println("Product: " + product);
    }
}
```

37. Implement pow(x, n), which calculates x raised to the power n

Input: x = 2.00000, n = 10

Output: 1024.00000

```
public class PowerCalculator {
    public static double myPow(double x, int n) {
        if (n == 0) {
            return 1;
        }

        if (n < 0) {
            x = 1 / x;
            n = -n;
        }

        double result = 1;
        double currentProduct = x;

        for (int i = n; i > 0; i /= 2) {
            if (i % 2 == 1) {
                result *= currentProduct;
            }
            currentProduct *= currentProduct;
        }

        return result;
    }

    public static void main(String[] args) {
        double x = 2.00000;
        int n = 10;

        double result = myPow(x, n);

        System.out.println("Output: " + result);
    }
}
```

38. Given an integer array nums, find the subarray with the largest sum, and return its sum.

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]

Output: 6

Explanation: The subarray [4,-1, 2, 1] has the largest sum 6.

```
public class MaximumSubarraySum {
    public static int maxSubArray(int[] nums) {
        int currentSum = nums[0];
        int maxSum = nums[0];
        for (int i = 1; i < nums.length; i++) {
            currentSum = Math.max(nums[i], currentSum + nums[i]);
            maxSum = Math.max(maxSum, currentSum);
        }

        return maxSum;
    }

    public static void main(String[] args) {
        int[] nums = {-2, 1, -3, 4, -1, 2, 1, -5, 4};

        int result = maxSubArray(nums);

        System.out.println("Output: " + result);
    }
}
```

39. Write a Java program which creates only one object. If user attempts to create second object, he should not be able to create it. (Using Exception Handling).

```
public class SingletonObject {
    private static SingletonObject instance;

    private SingletonObject() {
        // Private constructor to prevent instantiation
    }

    public static SingletonObject getInstance() {
        if (instance == null) {
            instance = new SingletonObject();
            return instance;
        } else {
            throw new IllegalStateException("Only one instance allowed.");
        }
    }

    public static void main(String[] args) {
        try {
            // Creating the first object
            SingletonObject obj1 = SingletonObject.getInstance();
            System.out.println("Object 1 created successfully.");

            // Attempting to create the second object
            SingletonObject obj2 = SingletonObject.getInstance(); // This will throw an
exception
```

```
            System.out.println("Object 2 created successfully.");
        } catch (IllegalStateException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

40. There is an exam room with n seats in a single row labeled from 0 to n - 1.When a student enters the room, they must sit in the seat that maximizes the distance to the closest person. If there are multiple such seats, they sit in the seat with the lowest number. If no one is in the room, then the student sits at seat number 0.Design a class that simulates the mentioned exam room. Implement the ExamRoom class: ExamRoom (int n) Initializes the object of the exam room with the number of the seats n. int seat () Returns the label of the seat at which the next student will set. Void leave (int p) indicates that the student sitting at seat p will leave the room. It is guaranteed that there will be a student sitting at seat p.
    Input["ExamRoom", "seat", "seat", "seat", "seat", "leave", "seat"]
    [[10], [], [], [], [], [4], []]
    Output
    [null, 0, 9, 4, 2, null, 5]

```
import java.util.TreeSet;

public class ExamRoom {
    private int n;
    private TreeSet<Integer> occupiedSeats;

    public ExamRoom(int n) {
        this.n = n;
        this.occupiedSeats = new TreeSet<>();
    }

    public int seat() {
        if (occupiedSeats.isEmpty()) {
            // If no one is in the room, sit at seat 0
            occupiedSeats.add(0);
            return 0;
        }

        int maxDistance = occupiedSeats.first(); // Distance from the start
        int bestSeat = 0;

        Integer prev = null;

        for (int currentSeat : occupiedSeats) {
            if (prev != null) {
                int distance = (currentSeat - prev) / 2;
                if (distance > maxDistance) {
                    maxDistance = distance;
                    bestSeat = prev + distance;
                }
            }
            prev = currentSeat;
        }

        // Check distance from the last occupied seat to the end
        if (n - 1 - occupiedSeats.last() > maxDistance) {
            bestSeat = n - 1;
        }

        occupiedSeats.add(bestSeat);
        return bestSeat;
    }

    public void leave(int p) {
        occupiedSeats.remove(p);
    }

    public static void main(String[] args) {
        ExamRoom examRoom = new ExamRoom(10);

        System.out.println(examRoom.seat());  // Output: 0
        System.out.println(examRoom.seat());  // Output: 9
        System.out.println(examRoom.seat());  // Output: 4
        System.out.println(examRoom.seat());  // Output: 2

        examRoom.leave(4);

        System.out.println(examRoom.seat());  // Output: 5
    }
}
```

41. You have n tiles, where each tile has one letter tiles[i] printed on it. Return the number of possible non-empty sequences of letters you can make using the letters printed on those tiles.

    Input: tiles = "AAB"

    Output: 8

    Explanation: The possible sequences are "A", "B", "AA", "AB", "BA", "AAB", "ABA", "BAA".

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

```
public class LetterTilePermutations {
    public static int numTilePossibilities(String tiles) {
        Set<String> set = new HashSet<>();
        generatePermutations(tiles.toCharArray(), 0, set);
        return set.size();
    }

    private static void generatePermutations(char[] tiles, int index, Set<String> set) {
        if (index == tiles.length) {
            // Convert the char array to a string and add it to the set
            set.add(new String(tiles));
            return;
        }

        for (int i = index; i < tiles.length; i++) {
            swap(tiles, index, i); // Swap characters to generate permutations
            generatePermutations(tiles, index + 1, set);
            swap(tiles, index, i); // Backtrack to the original state
        }
    }

    private static void swap(char[] tiles, int i, int j) {
        char temp = tiles[i];
        tiles[i] = tiles[j];
        tiles[j] = temp;
    }

    public static void main(String[] args) {
        String tiles = "AAB";
        int result = numTilePossibilities(tiles);
        System.out.println("Output: " + result);
    }
}
```

42. Write a program to read the data from the file and copy it on another file

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class FileCopyProgram {
    public static void main(String[] args) {
        String sourceFilePath = "path/to/source/file.txt"; // Replace with the actual
source file path
```

```
        String destinationFilePath = "path/to/destination/copiedFile.txt"; // Replace with
the desired destination file path

        try {
            // Read all bytes from the source file
            byte[] fileContent = Files.readAllBytes(Paths.get(sourceFilePath));

            // Write the bytes to the destination file
            Files.write(Paths.get(destinationFilePath), fileContent);

            System.out.println("File copied successfully!");
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

43. Implementing FileReader and FileWriter class to read and write the data from file

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class FileReaderWriterExample {
    public static void main(String[] args) {
        String sourceFilePath = "path/to/source/file.txt"; // Replace with the actual
source file path
        String destinationFilePath = "path/to/destination/copiedFile.txt"; // Replace with
the desired destination file path

        try (FileReader reader = new FileReader(sourceFilePath);
             FileWriter writer = new FileWriter(destinationFilePath)) {

            // Reading and writing character by character
            int character;
            while ((character = reader.read()) != -1) {
                writer.write(character);
            }

            System.out.println("File copied successfully!");
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

44. Write a program to read the file using BufferedReader class

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class BufferedReaderExample {
   public static void main(String[] args) {
      String filePath = "path/to/your/file.txt"; // Replace with the actual file path

      try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

         String line;
         while ((line = reader.readLine()) != null) {
            System.out.println(line); // Print each line read from the file
         }

      } catch (IOException e) {
         System.out.println("An error occurred: " + e.getMessage());
      }
   }
}
```

45. Write a program for counting number of characters, words and lines in a file

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class FileStatistics {
   public static void main(String[] args) {
      String filePath = "path/to/your/file.txt"; // Replace with the actual file path

      try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

         int characterCount = 0;
         int wordCount = 0;
         int lineCount = 0;

         String line;
         while ((line = reader.readLine()) != null) {
            // Count characters
            characterCount += line.length();

            // Count words
            String[] words = line.split("\\s+");
            wordCount += words.length;
```

```
                // Count lines
                lineCount++;
            }

            System.out.println("Number of characters: " + characterCount);
            System.out.println("Number of words: " + wordCount);
            System.out.println("Number of lines: " + lineCount);

        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

46. Write a program in Java to input an NxN matrix and display it row-wise and column-wise

```
import java.util.Scanner;

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

        // Input size of the matrix
        System.out.print("Enter the size of the matrix (N): ");
        int n = scanner.nextInt();

        // Input matrix elements
        int[][] matrix = new int[n][n];
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print("Enter element at position (" + (i + 1) + "," + (j + 1) + "):
");
                matrix[i][j] = scanner.nextInt();
            }
        }

        // Display matrix row-wise
        System.out.println("Matrix displayed row-wise:");
        displayRowWise(matrix);

        // Display matrix column-wise
        System.out.println("Matrix displayed column-wise:");
        displayColumnWise(matrix);

        scanner.close();
    }

    // Method to display matrix row-wise
    private static void displayRowWise(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }

    // Method to display matrix column-wise
    private static void displayColumnWise(int[][] matrix) {
        for (int j = 0; j < matrix[0].length; j++) {
            for (int i = 0; i < matrix.length; i++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

47. Write a java program which creates an interface IterF1 having 2 methods add () and sub (). Create a class which overloads the given methods for addition and subtraction of two numbers respectively.

```
    // Interface IterF1
    interface IterF1 {
        void add(int a, int b);
        void sub(int a, int b);
    }

    // Class Calculator implementing IterF1
    class Calculator implements IterF1 {
        // Overriding add() for addition
        @Override
        public void add(int a, int b) {
            int result = a + b;
            System.out.println("Addition result: " + result);
        }

        // Overriding sub() for subtraction
        @Override
        public void sub(int a, int b) {
            int result = a - b;
            System.out.println("Subtraction result: " + result);
        }
    }

    public class InterfaceExample {
        public static void main(String[] args) {
            // Creating an object of Calculator
            Calculator calculator = new Calculator();

            // Calling add() and sub() methods
            calculator.add(10, 5);
            calculator.sub(10, 5);
        }
    }
```

48. Write a Java program to calculate the rate of interest of the employee’s Provident Fund use the try and catch and finally block.

```
import java.util.Scanner;

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

        try {
            // Input employee's salary
            System.out.print("Enter the employee's salary: ");
            double salary = scanner.nextDouble();

            // Input employee's years of service
            System.out.print("Enter the employee's years of service: ");
            int yearsOfService = scanner.nextInt();

            // Calculate Provident Fund interest rate
            double interestRate = calculateInterestRate(salary, yearsOfService);

            // Display the calculated interest rate
            System.out.println("The Provident Fund interest rate is: " + interestRate +
"%");
        } catch (Exception e) {
            System.out.println("Error: Invalid input. Please enter valid numeric values.");
        } finally {
            // Close the scanner in the finally block to ensure resources are released
            scanner.close();
        }
    }
    // Method to calculate Provident Fund interest rate
    private static double calculateInterestRate(double salary, int yearsOfService) {
        if (yearsOfService < 5) {
            return 8.0; // 8% interest for less than 5 years of service
        } else {
            return 10.0; // 10% interest for 5 or more years of service
        }
    }
}
```

49. Write a program to create a class MyThread in this class a constructor, call the base class constructor, using super and starts the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently

```
class MyThread extends Thread {
    MyThread(String threadName) {
        // Call the base class (Thread) constructor with the provided threadName
        super(threadName);

        // Start the thread
        start();
    }

    // Override the run method to define the behavior of the thread
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + " Count: " + i);

            try {
                // Sleep for a short duration to simulate concurrent execution
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        // Create an instance of MyThread
        MyThread myThread = new MyThread("ChildThread");
```

```
            // Main thread execution
            for (int i = 0; i < 5; i++) {
                System.out.println(Thread.currentThread().getName() + " Count: " + i);

                try {
                    // Sleep for a short duration to simulate concurrent execution
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        }
    }
```

50. Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". Define the two exception classes.

```
// User-defined exception for age not within range
class AgeNotWithinRangeException extends Exception {
    AgeNotWithinRangeException(String message) {
        super(message);
    }
}

// User-defined exception for name not valid
class NameNotValidException extends Exception {
    NameNotValidException(String message) {
        super(message);
    }
}

// Student class with attributes and exception handling
class Student {
    private int rollNo;
    private String name;
    private int age;
    private String course;

    // Parameterized constructor with exception handling
    Student(int rollNo, String name, int age, String course) throws
AgeNotWithinRangeException, NameNotValidException {
        // Check if age is within the range 15 to 21
        if (age < 15 || age > 21) {
            throw new AgeNotWithinRangeException("Age should be between 15 and 21");
        }

        // Check if name contains numbers or special symbols
        if (!name.matches("[a-zA-Z]+")) {
            throw new NameNotValidException("Name should contain only letters");
        }

        // Initialize attributes
        this.rollNo = rollNo;
        this.name = name;
        this.age = age;
        this.course = course;
    }

    // Getter methods

    public int getRollNo() {
        return rollNo;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getCourse() {
        return course;
    }
}

public class StudentExample {
    public static void main(String[] args) {
        try {
            // Creating a valid student
            Student validStudent = new Student(1, "John", 20, "Computer Science");

            // Displaying details of valid student
            System.out.println("Valid Student Details:");
            System.out.println("Roll No: " + validStudent.getRollNo());
            System.out.println("Name: " + validStudent.getName());
            System.out.println("Age: " + validStudent.getAge());
            System.out.println("Course: " + validStudent.getCourse());

            // Creating an invalid student with age not within range
            Student invalidAgeStudent = new Student(2, "Alice", 25, "Mathematics");

            // Creating an invalid student with name containing a number
            Student invalidNameStudent = new Student(3, "Bob123", 18, "Physics");
        } catch (AgeNotWithinRangeException | NameNotValidException e) {
            // Catching and handling exceptions
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```