BEGINNER LEVEL

1) **Write a program to find the factorial of n Number.**

```
public class factorial {
    public static long calculateFactorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("Factorial is not defined for negative numbers.");
        }
        if (n == 0 || n == 1) {
            return 1;
        }
        long factorial = 1;
        for (int i = 2; i <= n; i++) {
            factorial *= i;
        }
        return factorial;
    }

    public static void main(String[] args) {
        int number = 5;
        long result = calculateFactorial(number);
        System.out.println("Factorial of " + number + " is: " + result);
    }
}
```

Output:

```
PS D:\MCA Practicals\java-lab-Rahul-Sehraya\Beginner> cd "d:\MCA Practicals\java-lab-Rahul-Sehraya\Beginner\" ; if ($?) { javac fac
torial.java } ; if ($?) { java factorial }
Factorial of 5 is: 120
PS D:\MCA Practicals\java-lab-Rahul-Sehraya\Beginner>
```

2)**Write a program to find the sequence of Fibonacci series up to n terms**

```
public class fib{
    static void Fibonacci(int N)
    {
        int n1 = 0, n2 = 1;

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

            System.out.print(n1 + " ");
```

```
            int n3 = n2 + n1;
            n1 = n2;
            n2 = n3;
        }
    }
    public static void main(String args[])
    {

        int N = 10;

        Fibonacci(N);
    }
}
```

```
PS D:\MCA Practicals\java-lab-Rahul-Sehraya\Beginner> c
.java } ; if ($?) { java fib }
0 1 1 2 3 5 8 13 21 34
PS D:\MCA Practicals\java-lab-Rahul-Sehraya\Beginner>
```

3) **Write a program to check whether given number is palindrome or not**

```
public class PalindromeCheck {
    public static boolean isNumberPalindrome(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) {
        int num1 = 121;
        int num2 = 12345;
        System.out.println(num1 + " is a palindrome: " + isNumberPalindrome(num1));
        System.out.println(num2 + " is a palindrome: " + isNumberPalindrome(num2));
    }
}
```

```
121 is a palindrome: true
12345 is a palindrome: false
```

4) **Write a program to find the HCF of two numbers**.

```
import java.util.Scanner;

public class HCFCalculator {

    public static int findHCF(int num1, int num2) {
        while (num2 != 0) {
            int temp = num2;
            num2 = num1 % num2;
            num1 = temp;
        }
        return num1;
    }

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

        System.out.print("Enter the first number: ");
        int number1 = scanner.nextInt();

        System.out.print("Enter the second number: ");
        int number2 = scanner.nextInt();

        int hcf = findHCF(number1, number2);
        System.out.println("The HCF of " + number1 + " and " + number2 + " is: " + hcf);

        scanner.close();
    }
}
```

```
Enter the first number: 12
Enter the second number: 88
The HCF of 12 and 88 is: 4
```

5) **Write a Java Program that will display the sum of 1+1/2+1/3…..+1/n.**

```
import java.util.Scanner;

public class HarmonicSum {

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

```
        System.out.print("Enter a positive integer (n) to calculate the sum of harmonic numbers: ");
        int n = scanner.nextInt();

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

        double sum = 0.0;

        for (int i = 1; i <= n; i++) {
            sum += 1.0 / i;
        }

        System.out.println("The sum of the first " + n + " harmonic numbers is: " + sum);

        scanner.close();
    }
}
```

```
Enter a positive integer (n) to calculate the sum of harmonic numbers: 6
The sum of the first 6 harmonic numbers is: 2.4499999999999997
```

6) **Write a Java Program to find product of two matrices.**

```
public class MatrixMultiplication {

    public static void main(String[] args) {
        int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}};
        int[][] matrix2 = {{7, 8}, {9, 10}, {11, 12}};


        if (matrix1[0].length != matrix2.length) {
            System.out.println("Matrix multiplication not possible. Number of columns in first matrix
must equal number of rows in second matrix.");
            return;
        }

        int rows1 = matrix1.length;
        int cols1 = matrix1[0].length;
        int rows2 = matrix2.length;
        int cols2 = matrix2[0].length;
```

```
        int[][] resultMatrix = new int[rows1][cols2];

        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    resultMatrix[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
        System.out.println("Resultant Matrix:");
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                System.out.print(resultMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

```
Resultant Matrix:
58 64
139 154
```

7) **Write a Java Program to find the sum and subtraction of two matrices.**

```
import java.util.Scanner;
public class MatrixOperations {

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

        System.out.print("Enter the number of rows for matrices: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns for matrices: ");
        int cols = scanner.nextInt();

        int[][] matrix1 = new int[rows][cols];
        int[][] matrix2 = new int[rows][cols];
        int[][] sumMatrix = new int[rows][cols];
        int[][] diffMatrix = new int[rows][cols];

        System.out.println("\nEnter elements for the first matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print("Enter element at [" + i + "][" + j + "]: ");
                matrix1[i][j] = scanner.nextInt();
```

```
        }
    }

    System.out.println("\nEnter elements for the second matrix:");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            System.out.print("Enter element at [" + i + "][" + j + "]: ");
            matrix2[i][j] = scanner.nextInt();
        }
    }

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            sumMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
            diffMatrix[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }

    System.out.println("\nFirst Matrix:");
    printMatrix(matrix1);

    System.out.println("\nSecond Matrix:");
    printMatrix(matrix2);

    System.out.println("\nSum of Matrices:");
    printMatrix(sumMatrix);

    System.out.println("\nDifference of Matrices:");
    printMatrix(diffMatrix);

    scanner.close();
}

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

}

```
Enter the number of rows for matrices: 2
Enter the number of columns for matrices: 2

Enter elements for the first matrix:
Enter element at [0][0]: 5
Enter element at [0][1]: 4
Enter element at [1][0]: 1
Enter element at [1][1]: 2

Enter elements for the second matrix:
Enter element at [0][0]: 2
Enter element at [0][1]: 2
Enter element at [1][0]: 4
Enter element at [1][1]: 4

First Matrix:
5       4
1       2

Second Matrix:
2       2
4       4

Sum of Matrices:
7       6
5       6

Difference of Matrices:
3       2
-3      -2
```

8) **Write a Java Program to sort the list in ascending order.**

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

public class SortListAscending {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(5);
        numbers.add(2);
        numbers.add(8);
        numbers.add(1);
        numbers.add(9);
```

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

        Collections.sort(numbers);

        System.out.println("Sorted list (ascending): " + numbers);
    }
}
```

```
Original list: [5, 2, 8, 1, 9]
Sorted list (ascending): [1, 2, 5, 8, 9]
```

9) **Write a Java Program to convert decimal into binary number.**

```
import java.util.Scanner;

public class DecimalToBinary {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a decimal number: ");
        int decimalNumber = scanner.nextInt();
        scanner.close();

        StringBuilder binaryResult = new StringBuilder();

        if (decimalNumber == 0) {
            binaryResult.append(0);
        } else {
            int tempDecimal = decimalNumber;
            while (tempDecimal > 0) {
                int remainder = tempDecimal % 2;
                binaryResult.append(remainder);
                tempDecimal /= 2;
            binaryResult.reverse();
        }

        System.out.println("Decimal: " + decimalNumber);
        System.out.println("Binary: " + binaryResult.toString());
    }
}
}
```

```
Enter a decimal number: 4
Decimal: 4
Binary: 100
```

10) **Write a Java Program to find largest and smallest of n numbers**

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

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

        if (n <= 0) {
            System.out.println("Please enter a positive number of elements.");
            return;
        }

        int largest = Integer.MIN_VALUE;
        int smallest = Integer.MAX_VALUE;

        System.out.println("Enter " + n + " numbers:");

        for (int i = 0; i < n; i++) {
            System.out.print("Enter number " + (i + 1) + ": ");
            int currentNumber = scanner.nextInt();

            if (currentNumber > largest) {
                largest = currentNumber;
            }

            if (currentNumber < smallest) {
                smallest = currentNumber;
            }
        }

        System.out.println("Largest number: " + largest);
        System.out.println("Smallest number: " + smallest);

        scanner.close();
    }
}
```

```
Enter the number of elements (n): 4
Enter 4 numbers:
Enter number 1: 6
Enter number 2: 8
Enter number 3: 4
Enter number 4: 2
Largest number: 8
Smallest number: 2
```

INTERMEDIATE LEVEL

11) **Write a Java program that shows the application of constructors.**

```
class Student {
    String name;
    int age;

    Student() {
        name = "Unknown";
        age = 0;
        System.out.println("Default Constructor Called");
    }

    Student(String n, int a) {
        name = n;
        age = a;
        System.out.println("Parameterized Constructor Called");
    }

    Student(String n) {
        name = n;
        age = 18;
        System.out.println("Overloaded Constructor Called (Name Only)");
    }

    void display() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}

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

        Student s1 = new Student();
        s1.display();

        Student s2 = new Student("Rahul", 21);
        s2.display();

        Student s3 = new Student("Ajay");
        s3.display();
```

```
    }
}
```

```
Default Constructor Called
Name: Unknown, Age: 0
Parameterized Constructor Called
Name: Rahul, Age: 21
Overloaded Constructor Called (Name Only)
Name: Ajay, Age: 18
```

12) **Write a Java program to find the electricity bill using inheritance**

```
class Customer {
    String name;
    int units;

    Customer(String name, int units) {
        this.name = name;
        this.units = units;
    }
}

class ElectricityBill extends Customer {

    ElectricityBill(String name, int units) {
        super(name, units);
    }

    double calculateBill() {
        double amount;

        if (units <= 100)
            amount = units * 5;
        else if (units <= 200)
            amount = 100 * 5 + (units - 100) * 7;
        else
            amount = 100 * 5 + 100 * 7 + (units - 200) * 10;

        return amount;
    }
}

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

```
        ElectricityBill eb = new ElectricityBill("Rahul", 250);

        System.out.println("Customer Name: " + eb.name);
        System.out.println("Units Consumed: " + eb.units);
        System.out.println("Total Bill: Rs." + eb.calculateBill());
    }
}
```

```
Customer Name: Rahul
Units Consumed: 250
Total Bill: Rs.1700.0
```

13) **Create a class "Vehicle" with instance variable vehicle_type. Inherit the class in a class called "Car" with instance model_type, company name etc. Display the information of the vehicle by defining the display() in both super and sub class**.

```
class Vehicle {
    String vehicle_type;

    Vehicle(String vehicle_type) {
        this.vehicle_type = vehicle_type;
    }

    void display() {
        System.out.println("Vehicle Type: " + vehicle_type);
    }
}

class Car extends Vehicle {
    String model_type;
    String company_name;

    Car(String vehicle_type, String model_type, String company_name) {
        super(vehicle_type);
        this.model_type = model_type;
        this.company_name = company_name;
    }

    void display() {
        super.display();
        System.out.println("Company Name: " + company_name);
        System.out.println("Model Type: " + model_type);
    }
```

```
}

public class Vehicles {
    public static void main(String[] args) {
        Car c = new Car("Four Wheeler", "Sedan", "Honda");
        c.display();
    }
}
```

```
Vehicle Type: Four Wheeler
Company Name: Honda
Model Type: Sedan
```

14) **Create a class “Account” containing account No, and balance as an instance variable. Derive the Account class into two classes named “Savings” and “Current”. The “Savings” class should contain an instance variable named interest Rate, and the “Current” class should contain an instance variable called overdraft Limit. Define appropriate methods for all the classes to enable functionalities to check balance, deposit, and withdraw amounts in Savings and Current accounts.**

```
class Account {
    int accountNo;
    double balance;

    Account(int accountNo, double balance) {
        this.accountNo = accountNo;
        this.balance = balance;
    }

    void checkBalance() {
        System.out.println("Account No: " + accountNo);
        System.out.println("Current Balance: Rs." + balance);
    }

    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: Rs." + amount);
        checkBalance();
    }

    void withdraw(double amount) {
        if (amount <= balance) {
```

```
            balance -= amount;
            System.out.println("Withdrawn: Rs." + amount);
        } else {
            System.out.println("Insufficient Balance!");
        }
        checkBalance();
    }
}

class Savings extends Account {
    double interestRate;

    Savings(int accountNo, double balance, double interestRate) {
        super(accountNo, balance);
        this.interestRate = interestRate;
    }

    void addInterest() {
        double interest = balance * (interestRate / 100);
        balance += interest;
        System.out.println("Interest Added: Rs." + interest);
        checkBalance();
    }
}

class Current extends Account {
    double overdraftLimit;

    Current(int accountNo, double balance, double overdraftLimit) {
        super(accountNo, balance);
        this.overdraftLimit = overdraftLimit;
    }

    @Override
    void withdraw(double amount) {
        if (amount <= balance + overdraftLimit) {
            balance -= amount;
            System.out.println("Withdrawn: Rs." + amount);
        } else {
            System.out.println("Overdraft Limit Exceeded!");
        }
        checkBalance();
    }
}
```

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

        System.out.println("---- Savings Account ----");
        Savings s = new Savings(1001, 5000, 5);
        s.checkBalance();
        s.deposit(2000);
        s.withdraw(1000);
        s.addInterest();

        System.out.println("\n---- Current Account ----");
        Current c = new Current(2001, 3000, 2000);
        c.checkBalance();
        c.deposit(1500);
        c.withdraw(6000);
        c.withdraw(2000);
    }
}
```

```
---- Savings Account ----
Account No: 1001
Current Balance: Rs.5000.0
Deposited: Rs.2000.0
Account No: 1001
Current Balance: Rs.7000.0
Withdrawn: Rs.1000.0
Account No: 1001
Current Balance: Rs.6000.0
Interest Added: Rs.300.0
Account No: 1001
Current Balance: Rs.6300.0

---- Current Account ----
Account No: 2001
Current Balance: Rs.3000.0
Deposited: Rs.1500.0
Account No: 2001
Current Balance: Rs.4500.0
Withdrawn: Rs.6000.0
Account No: 2001
Current Balance: Rs.-1500.0
Overdraft Limit Exceeded!
Account No: 2001
Current Balance: Rs.-1500.0
```

15) **Write a program to demonstrate the use of ‘this’ and ‘static’ keywords.**

```
class Demo {

    int number;

    static int count = 0;

    Demo(int number) {
        this.number = number;
        count++;
    }

    void show() {
        System.out.println("Number = " + this.number);
    }

    static void showCount() {
        System.out.println("Total objects created = " + count);
    }
}


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

        Demo d1 = new Demo(10);
        Demo d2 = new Demo(20);
        Demo d3 = new Demo(30);

        d1.show();
        d2.show();
        d3.show();

        Demo.showCount();
    }
}
```

```
Number = 10
Number = 20
Number = 30
Total objects created = 3
```

16) **Describe an abstract class called Shape which has three subclasses say Triangle, Rectangle, and Circle. Define one method area () in the abstract class and override this area () in these three subclasses to calculate for specific object i.e. area () of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.**

```
abstract class Shape {
    abstract void area();
}

class Triangle extends Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
    void area() {
        double result = 0.5 * base * height;
        System.out.println("Area of Triangle: " + result);
    }
}

class Rectangle extends Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    void area() {
        double result = length * width;
        System.out.println("Area of Rectangle: " + result);
    }
}

class Circle extends Shape {
    double radius;

    Circle(double radius) {
```

```
        this.radius = radius;
    }

    @Override
    void area() {
        double result = Math.PI * radius * radius;
        System.out.println("Area of Circle: " + result);
    }
}

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

        Shape t = new Triangle(10, 5);
        Shape r = new Rectangle(4, 6);
        Shape c = new Circle(7);

        t.area();
        r.area();
        c.area();
    }
}
```

```
Area of Triangle: 25.0
Area of Rectangle: 24.0
Area of Circle: 153.93804002589985
```

17) **Write a java program to find the result sheet of a student using Interfaces.**

```
interface StudentDetails {
    void getStudentDetails(String name, int roll);
}

interface Marks {
    void getMarks(int m1, int m2, int m3);
}

class ResultRecord implements StudentDetails, Marks {
    String name;
    int roll;
    int m1, m2, m3;

    @Override
    public void getStudentDetails(String name, int roll) {
```

```
        this.name = name;
        this.roll = roll;
    }

    @Override
    public void getMarks(int m1, int m2, int m3) {
        this.m1 = m1;
        this.m2 = m2;
        this.m3 = m3;
    }

    void displayResult() {
        int total = m1 + m2 + m3;
        double percentage = total / 3.0;

        System.out.println("\n----- Result Sheet -----");
        System.out.println("Name: " + name);
        System.out.println("Roll No: " + roll);
        System.out.println("Marks: " + m1 + ", " + m2 + ", " + m3);
        System.out.println("Total: " + total);
        System.out.println("Percentage: " + percentage + "%");

        if (percentage >= 33)
            System.out.println("Result: PASS");
        else
            System.out.println("Result: FAIL");
    }
}


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

        ResultRecord r = new ResultRecord();

        r.getStudentDetails("Rahul", 81);
        r.getMarks(78, 85, 90);

        r.displayResult();
    }
}
```

```
----- Result Sheet -----
Name: Rahul
Roll No: 81
Marks: 78, 85, 90
Total: 253
Percentage: 84.33333333333333%
Result: PASS
```

18) **Write a java program which shows importing of classes from other packages.**

File 1 :

```
package Importing_packages;
public class Addition {
    public int add(int a, int b) {
        return a + b;
    }
}
```

File 2 :

```
package Importing_packages;

public class importing {
    public static void main(String[] args) {
        Addition obj = new Addition();
        int result = obj.add(10, 20);

        System.out.println("Sum = " + result);
    }
}
```

```
PS D:\MCA Practicals\
Sum = 30
```

19) **Assume that there are two packages, student and exam. A student package contains Student class and the exam package contains Result class. Write a program that generates mark sheet for students.**

File 1 :

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

        Result r = new Result("Rahul Sehraya", 81, 78, 85, 90);
```

```
        r.generateMarksheet();
    }
}
```

File 2 :

```
public class Result extends Student {
    int m1, m2, m3;

    public Result(String name, int roll, int m1, int m2, int m3) {
        super(name, roll);
        this.m1 = m1;
        this.m2 = m2;
        this.m3 = m3;
    }

    public void generateMarksheet() {
        int total = m1 + m2 + m3;
        double percentage = total / 3.0;

        System.out.println("----- MARKSHEET -----");
        displayStudent();
        System.out.println("Marks: " + m1 + ", " + m2 + ", " + m3);
        System.out.println("Total: " + total);
        System.out.println("Percentage: " + percentage + "%");

        if (percentage >= 33)
            System.out.println("Result: PASS");
        else
            System.out.println("Result: FAIL");
    }
}
```

File 3:

```
public class Student {
    public String name;
    public int roll;

    public Student(String name, int roll) {
        this.name = name;
        this.roll = roll;
    }
```

```
    public void displayStudent() {
        System.out.println("Name: " + name);
        System.out.println("Roll No: " + roll);
    }
}
```

```
----- MARKSHEET -----
Name: Rahul Sehraya
Roll No: 81
Marks: 78, 85, 90
Total: 253
Percentage: 84.33333333333333%
Result: PASS
```

20) **Write a program to implement the concept of threading by implementing "Runnable" Interface.**

```
class MyRunnable implements Runnable {
    private String name;

    MyRunnable(String name) {
        this.name = name;
    }

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(name + " - iteration " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println(name + " interrupted");
                Thread.currentThread().interrupt();
                break;
            }
        }
    }
}

public class RunnableDemo {
    public static void main(String[] args) {
        Thread t1 = new Thread(new MyRunnable("RunnableThread1"));
        Thread t2 = new Thread(new MyRunnable("RunnableThread2"));
```

```
        t1.start();
        t2.start();
    }
}
```

```
RunnableThread2 - iteration 1
RunnableThread1 - iteration 1
RunnableThread1 - iteration 2
RunnableThread2 - iteration 2
RunnableThread2 - iteration 3
RunnableThread1 - iteration 3
RunnableThread1 - iteration 4
RunnableThread2 - iteration 4
RunnableThread1 - iteration 5
RunnableThread2 - iteration 5
PS D:\MCA_Practicals\java-lab-Rahul-
```

21) **Write a program that executes two threads. One thread displays “Thread1” every 2,000 milliseconds, and the other displays “Thread2” every 4,000 milliseconds.**

```
class TimedPrinter implements Runnable {
    private final String label;
    private final long intervalMillis;
    private final int repeats;

    TimedPrinter(String label, long intervalMillis, int repeats) {
        this.label = label;
        this.intervalMillis = intervalMillis;
        this.repeats = repeats;
    }

    @Override
    public void run() {
        for (int i = 0; i < repeats; i++) {
            System.out.println(label + " (time: " + System.currentTimeMillis() + ")");
            try {
                Thread.sleep(intervalMillis);
            } catch (InterruptedException e) {
                System.out.println(label + " interrupted");
                Thread.currentThread().interrupt();
                return;
            }
        }
        System.out.println(label + " finished.");
    }
```

```
}

public class TwoThreadsTimed {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new TimedPrinter("Thread1", 2000, 5));
        Thread thread2 = new Thread(new TimedPrinter("Thread2", 4000, 3));

        thread1.start();
        thread2.start();

        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        System.out.println("Both threads completed.");
    }
}
```

```
Thread2 (time: 1763728186499)
Thread1 (time: 1763728186499)
Thread1 (time: 1763728188524)
Thread2 (time: 1763728190517)
Thread1 (time: 1763728190532)
Thread1 (time: 1763728192538)
Thread2 (time: 1763728194525)
Thread1 (time: 1763728194541)
Thread1 finished.
Thread2 finished.
Both threads completed.
```

22) **Write a java program which use try and catch for exception handling.**

```
public class TryCatch {
    public static void main(String[] args) {
        try {
            int a = 10;
            int b = 0;
            System.out.println("Dividing " + a + " by " + b);
            int c = a / b;
            System.out.println("Result = " + c);
        } catch (Exception e) {
```

```
            System.out.println("An exception occurred: " + e);
        }

        System.out.println("Program continues after try-catch.");
    }
}
```

```
Dividing 10 by 0
An exception occurred: java.lang.ArithmeticException: / by zero
Program continues after try-catch.
```

23) **Write a java program which use multiple catch blocks.**

```
public class MultipleCatch {
    public static void main(String[] args) {
        try {
            String s = args.length > 0 ? args[0] : null;
            if (s != null) {
                System.out.println("Length: " + s.length());
            } else {
                throw new NullPointerException("String reference is null");
            }
        } catch (ArithmeticException ae) {
            System.out.println("Arithmetic problem: " + ae.getMessage());
        } catch (NullPointerException ne) {
            System.out.println("Null pointer encountered: " + ne.getMessage());
        } catch (Exception e) {
            System.out.println("Some other exception: " + e);
        }

        System.out.println("End of multiple-catch demo.");
    }
}
```

```
Null pointer encountered: String reference is null
End of multiple-catch demo.
```

24) **Write a java program which shows throwing our own exception.**

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
```

```
}

class Voter {
    static void validateAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age " + age + " is not eligible to vote.");
        } else {
            System.out.println("Age " + age + " is eligible to vote.");
        }
    }
}

class CustomExceptionDemo {
    public static void main(String[] args) {
        int[] testAges = {20, 16};

        for (int age : testAges) {
            try {
                Voter.validateAge(age);
            } catch (InvalidAgeException ex) {
                System.out.println("Caught custom exception: " + ex.getMessage());
            }
        }

        System.out.println("Custom exception demo finished.");
    }
}
```

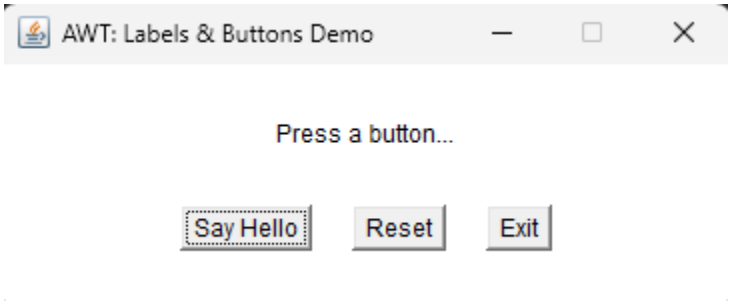
```
Age 20 is eligible to vote.
Caught custom exception: Age 16 is not eligible to vote.
Custom exception demo finished.
```

25) **Write a program to handle Labels and Buttons using AWT Controls**.

```
import java.awt.*;
import java.awt.event.*;

public class AWTControlsDemo extends Frame implements ActionListener {

    Label statusLabel;
    Button btnHello;
    Button btnReset;
    Button btnExit;
```

```
public AWTControlsDemo() {
    super("AWT: Labels & Buttons Demo");

    setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));

    statusLabel = new Label("Press a button...", Label.CENTER);
    statusLabel.setPreferredSize(new Dimension(300, 30));

    btnHello = new Button("Say Hello");
    btnReset = new Button("Reset");
    btnExit  = new Button("Exit");

    btnHello.addActionListener(this);
    btnReset.addActionListener(this);
    btnExit.addActionListener(this);

    add(statusLabel);
    add(btnHello);
    add(btnReset);
    add(btnExit);

    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            dispose();
            System.exit(0);
        }
    });

    setSize(380, 160);
    setResizable(false);
    setLocationRelativeTo(null);
}

@Override
public void actionPerformed(ActionEvent e) {
    Object src = e.getSource();
    if (src == btnHello) {
        statusLabel.setText("Hello! Welcome to AWT controls demo.");
    } else if (src == btnReset) {
        statusLabel.setText("Press a button...");
    } else if (src == btnExit) {
        dispose();
        System.exit(0);
```

```
        }
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            AWTControlsDemo demo = new AWTControlsDemo();
            demo.setVisible(true);
        });
    }
}
```

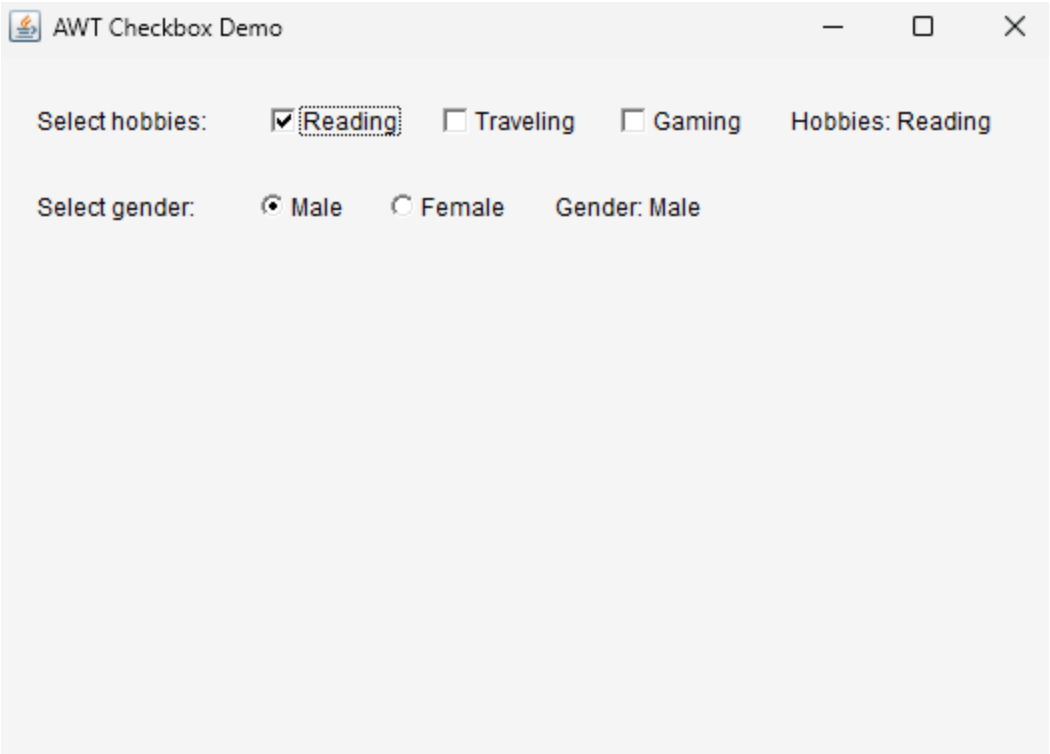


26) **Write a program to handle Check Boxes using AWT Controls**

```
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Label;
import java.awt.Checkbox;
import java.awt.CheckboxGroup;
import java.awt.Color;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class checkbox extends Frame implements ItemListener {

    private final Checkbox reading;
    private final Checkbox traveling;
    private final Checkbox gaming;
    private final CheckboxGroup genderGroup;
    private final Checkbox male;
    private final Checkbox female;
    private final Label hobbyStatus;
    private final Label genderStatus;
```

```
public checkbox() {
    super("AWT Checkbox Demo");
    setSize(400, 250);
    setLayout(new FlowLayout(FlowLayout.LEFT, 20, 20));
    setBackground(new Color(245, 245, 245));

    add(new Label("Select hobbies:"));
    reading = new Checkbox("Reading");
    traveling = new Checkbox("Traveling");
    gaming = new Checkbox("Gaming");
    reading.addItemListener(this);
    traveling.addItemListener(this);
    gaming.addItemListener(this);
    add(reading);
    add(traveling);
    add(gaming);

    hobbyStatus = new Label("No hobbies selected");
    add(hobbyStatus);

    add(new Label("Select gender:"));
    genderGroup = new CheckboxGroup();
    male = new Checkbox("Male", genderGroup, true);
    female = new Checkbox("Female", genderGroup, false);
    male.addItemListener(this);
    female.addItemListener(this);
    add(male);
    add(female);

    genderStatus = new Label("Gender: Male");
    add(genderStatus);

    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            dispose();
        }
    });

    setVisible(true);
}

@Override
public void itemStateChanged(ItemEvent e) {
```

```
        updateHobbyStatus();
        updateGenderStatus();
    }

    private void updateHobbyStatus() {
        StringBuilder sb = new StringBuilder("Hobbies: ");
        boolean hasSelection = false;

        if (reading.getState()) {
            sb.append("Reading ");
            hasSelection = true;
        }
        if (traveling.getState()) {
            sb.append("Traveling ");
            hasSelection = true;
        }
        if (gaming.getState()) {
            sb.append("Gaming ");
            hasSelection = true;
        }

        hobbyStatus.setText(hasSelection ? sb.toString().trim() : "No hobbies selected");
    }

    private void updateGenderStatus() {
        Checkbox selected = genderGroup.getSelectedCheckbox();
        genderStatus.setText("Gender: " + (selected != null ? selected.getLabel() : "Not selected"));
    }

    public static void main(String[] args) {
        new checkbox();
    }
}
```

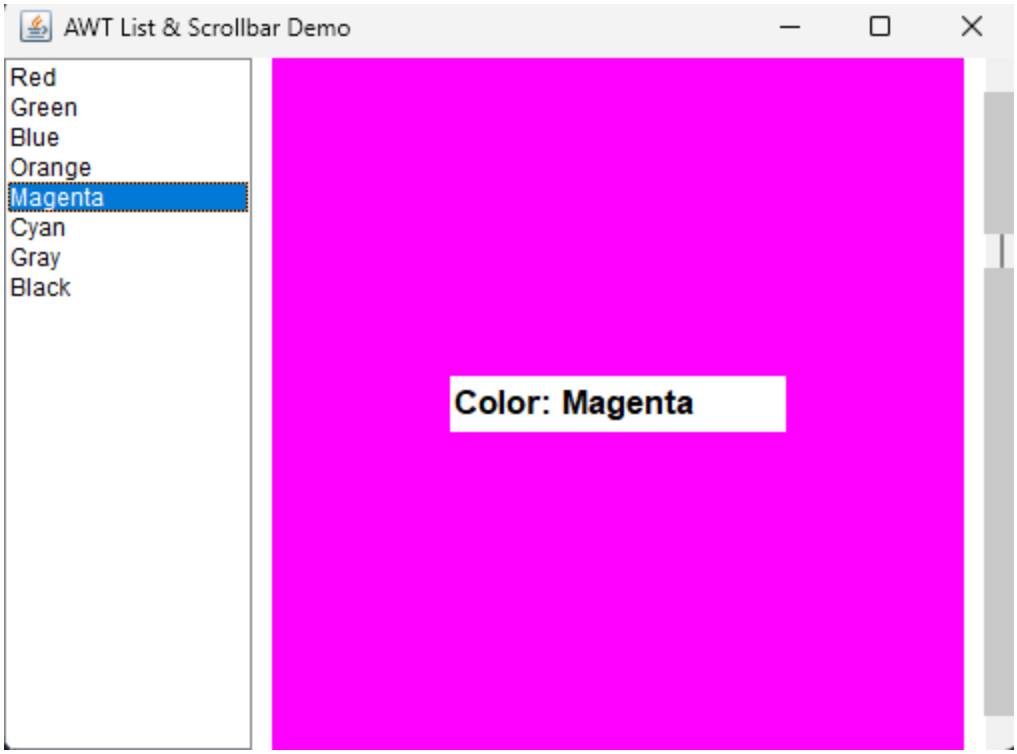


27) **Write a program to handle Lists and Scroll Bars using AWT Controls**

```
import java.awt.*;
import java.awt.event.*;

public class ListScroll extends Frame implements ItemListener, AdjustmentListener,
WindowListener {

    private final List colorList;
    private final Scrollbar sizeScrollbar;
    private final Label previewLabel;
    private final Panel previewPanel;

    public ListScroll() {
        super("AWT List & Scrollbar Demo");
        setSize(400, 300);
        setLayout(new BorderLayout(10, 10));
        addWindowListener(this);

        colorList = new List(6, false);
        String[] colors = {"Red", "Green", "Blue", "Orange", "Magenta", "Cyan", "Gray", "Black"};
        for (String color : colors) {
            colorList.add(color);
```

```
    }
    colorList.addItemListener(this);
    add(colorList, BorderLayout.WEST);

    previewPanel = new Panel(new GridBagLayout());
    previewLabel = new Label("Select a color & size");
    previewLabel.setFont(new Font("SansSerif", Font.BOLD, 16));
    previewPanel.add(previewLabel);
    add(previewPanel, BorderLayout.CENTER);

    sizeScrollbar = new Scrollbar(Scrollbar.VERTICAL, 16, 1, 10, 40);
    sizeScrollbar.addAdjustmentListener(this);
    add(sizeScrollbar, BorderLayout.EAST);
}

@Override
public void itemStateChanged(ItemEvent e) {
    String selected = colorList.getSelectedItem();
    if (selected != null) {
        previewPanel.setBackground(getColorFromName(selected));
        previewLabel.setText("Color: " + selected);
    }
}

@Override
public void adjustmentValueChanged(AdjustmentEvent e) {
    int fontSize = e.getValue();
    previewLabel.setFont(new Font("SansSerif", Font.BOLD, fontSize));
    previewLabel.setText(previewLabel.getText().split("\\|")[0] + " | Size: " + fontSize);
}

private Color getColorFromName(String name) {
    switch (name) {
        case "Red":
            return Color.RED;
        case "Green":
            return Color.GREEN;
        case "Blue":
            return Color.BLUE;
        case "Orange":
            return Color.ORANGE;
        case "Magenta":
            return Color.MAGENTA;
        case "Cyan":
```

```
            return Color.CYAN;
        case "Gray":
            return Color.GRAY;
        case "Black":
            return Color.BLACK;
        default:
            return getBackground();
      }
    }

    @Override
    public void windowOpened(WindowEvent e) {}

    @Override
    public void windowClosing(WindowEvent e) {
      dispose();
    }

    @Override
    public void windowClosed(WindowEvent e) {
      System.exit(0);
    }

    @Override
    public void windowIconified(WindowEvent e) {}

    @Override
    public void windowDeiconified(WindowEvent e) {}

    @Override
    public void windowActivated(WindowEvent e) {}

    @Override
    public void windowDeactivated(WindowEvent e) {}

    public static void main(String[] args) {
      EventQueue.invokeLater(() -> {
        ListScroll demo = new ListScroll();
        demo.setVisible(true);
      });
    }
}
```


AWT List & Scrollbar Demo
Red
Green
Blue
Orange
Magenta
Cyan
Gray
Black
Color: Magenta


AWT List & Scrollbar Demo
Red
Green
Blue
Orange
Magenta
Cyan
Gray
Black
Color: Green