

## 1. WAP to find factorial of a number

```
import java.util.*;
class Factorial {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();
        int f = 1;
        for(int i = 1; i <= n; i++){
            f = f * i;
        }
        System.out.println("Factorial = " + f);
    }
}
```



## 2. Write a program to find the sequence of Fibonacci series upto n terms

```
import java.util.*;
class Fibonacci {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of terms: ");
        int n = sc.nextInt();
        int a = 0, b = 1;
        System.out.print("Fibonacci Series: ");
        for(int i = 1; i <= n; i++){
```

```

        System.out.print(a + " ");
        int c = a + b;
        a = b;
        b = c;
    }
}
}

```



### 3. Write a program to check whether given number is palindrome or not

```

import java.util.Scanner;
public class PalindromeNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int temp = n, rev = 0;
        while (temp > 0) {
            rev = rev * 10 + temp % 10;
            temp /= 10;
        }
        if (rev == n)
            System.out.println("Palindrome");
        else
            System.out.println("Not Palindrome");
    }
}

```

Output    Generated files

```
1221
Palindrome
```

Compiled and executed in 5.859 sec(s)

4. Write a program to find HCF of two numbers.

```
import java.util.Scanner;

public class HCF {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first number: ");
        int a = sc.nextInt();
        System.out.print("Enter second number: ");
        int b = sc.nextInt();
        int hcf = 1;
        for (int i = 1; i <= a && i <= b; i++) {
            if (a % i == 0 && b % i == 0)
                hcf = i;
        }

        System.out.println("HCF of " + a + " and " + b + " is: " + hcf);
    }
}
```

Output    Generated files

```
Enter first number: 20
Enter second number: 24
HCF of 20 and 24 is: 4
|
```

Compiled and executed in 13.954 sec(s)

**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 sc = new Scanner(System.in);

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

        double sum = 0;

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

        System.out.println("Sum of the series is: " + sum);
    }
}
```



The screenshot shows a Java IDE interface. At the top, there are two tabs: 'Output' and 'Generated files'. The 'Output' tab is active, displaying the following text: 'Enter the value of n: 10' followed by 'Sum of the series is: 2.9289682539682538'. Below the output, there is a status bar that says 'Compiled and executed in 9.797 sec(s)'.

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

```
import java.util.Scanner;

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

        System.out.print("Enter rows of first matrix: ");

        int r1 = sc.nextInt();
```

```
System.out.print("Enter columns of first matrix: ");
int c1 = sc.nextInt();
System.out.print("Enter rows of second matrix: ");
int r2 = sc.nextInt();
System.out.print("Enter columns of second matrix: ");
int c2 = sc.nextInt();
if (c1 != r2) {
    System.out.println("Matrix multiplication not possible");
    return;
}
int[][] a = new int[r1][c1];
int[][] b = new int[r2][c2];
int[][] product = new int[r1][c2];
System.out.println("Enter elements of first matrix:");
for (int i = 0; i < r1; i++) {
    for (int j = 0; j < c1; j++) {
        a[i][j] = sc.nextInt();
    }
}
System.out.println("Enter elements of second matrix:");
for (int i = 0; i < r2; i++) {
    for (int j = 0; j < c2; j++) {
        b[i][j] = sc.nextInt();
    }
}
for (int i = 0; i < r1; i++) {
    for (int j = 0; j < c2; j++) {
        for (int k = 0; k < c1; k++) {
            product[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

```

        }
    }
}
System.out.println("Product of the matrices:");
for (int i = 0; i < r1; i++) {
    for (int j = 0; j < c2; j++) {
        System.out.print(product[i][j] + " ");
    }
    System.out.println();
}
}
}

```

Output

Generated files

```

Enter rows of first matrix: 2
Enter columns of first matrix: 3
Enter rows of second matrix: 3
Enter columns of second matrix: 3
Enter elements of first matrix:
2 3 4 5 6 7
Enter elements of second matrix:
1 2 3 4 5 6 7 8 9
Product of the matrices:
42 51 60
78 96 114
|

```

*i* Compiled and executed in 72.87 sec(s)

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

```

import java.util.Scanner;

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

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows: ");
        int rows = sc.nextInt();
        System.out.print("Enter number of columns: ");
    }
}

```

```

int cols = sc.nextInt();

int[][] matrix1 = new int[rows][cols];
int[][] matrix2 = new int[rows][cols];
int[][] sum = new int[rows][cols];
int[][] difference = new int[rows][cols];

System.out.println("Enter elements of first matrix:");
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        matrix1[i][j] = sc.nextInt();
    }
}

System.out.println("Enter elements of second matrix:");
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        matrix2[i][j] = sc.nextInt();
    }
}

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

System.out.println("Sum of matrices:");

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        System.out.print(sum[i][j] + " ");
    }
    System.out.println();
}

System.out.println("Difference of matrices:");

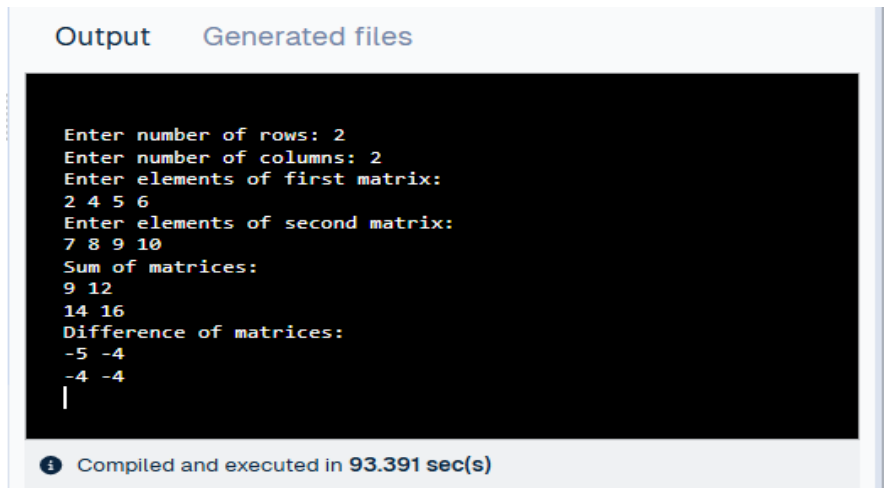
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        System.out.print(difference[i][j] + " ");
    }
}

```

```

        System.out.println();
    }
}
}

```



The screenshot shows a Java IDE with two tabs: 'Output' and 'Generated files'. The 'Output' tab is active, displaying the following text:

```

Enter number of rows: 2
Enter number of columns: 2
Enter elements of first matrix:
2 4 5 6
Enter elements of second matrix:
7 8 9 10
Sum of matrices:
9 12
14 16
Difference of matrices:
-5 -4
-4 -4
|

```

At the bottom of the IDE, a status bar indicates: "Compiled and executed in 93.391 sec(s)".

# 8. Write a Java Program to sort the list in ascending Order.

```

import java.util.Scanner;

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

        System.out.print("Enter number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];

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

        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}

```

```

    }

    System.out.println("Array in ascending order:");
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
}

```

The screenshot shows an IDE output window with two tabs: "Output" and "Generated files". The "Output" tab is active, displaying the following text:

```

Enter number of elements: 5
Enter elements:
20 5 10 6 8
Array in ascending order:
5 6 8 10 20 |

```

Below the output, a status bar indicates: "Compiled and executed in 24.793 sec(s)".

## 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 sc = new Scanner(System.in);

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

        int num = decimal;
        while (num > 0) {
            int remainder = num % 2;
            binary = remainder + binary;
            num = num / 2;
        }

        if (decimal == 0) {
            binary = "0";
        }
    }
}

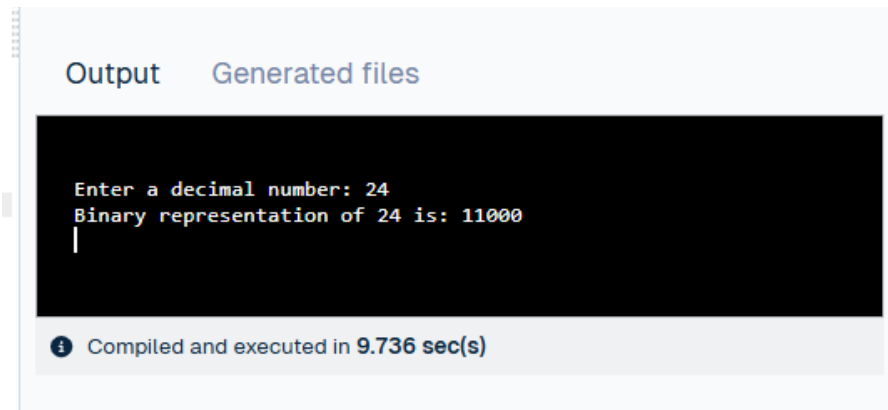
```

```

    }

    System.out.println("Binary representation of " + decimal + " is: " + binary);
}
}

```



#### 10. Write a Java Program to find largest and smallest of n numbers

```

import java.util.Scanner;

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

        System.out.print("Enter number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];

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

```

```
int largest = arr[0];
int smallest = arr[0];

for (int i = 1; i < n; i++) {
    if (arr[i] > largest) {
        largest = arr[i];
    }
    if (arr[i] < smallest) {
        smallest = arr[i];
    }
}

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

Output      Generated files

```
Enter number of elements: 5
Enter the numbers:
56 76 45 40 21
Largest number is: 76
Smallest number is: 21
```

 Compiled and executed in 24.761 sec(s)

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

```
import java.util.Scanner;

class Student {
    String name;
    int age;

    Student(String n, int a) {
        name = n;
        age = a;
    }

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

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

        System.out.print("Enter student name: ");
        String name = sc.nextLine();

        System.out.print("Enter student age: ");
        int age = sc.nextInt();
```

```

        Student s1 = new Student(name, age);
        s1.display();
    }
}

```

Output

Generated files

```

Enter student name: john
Enter student age: 21
Student Name: john
Student Age: 21

```

*i* Compiled and executed in 22.446 sec(s)

## 12. Write a Java program to find the electricity bill using inheritance

```

import java.util.Scanner;

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 bill = 0;
        int u = units;

        if (u <= 100) {

```

```

        bill = u * 1.5;
    } else if (u <= 200) {
        bill = 100 * 1.5 + (u - 100) * 2.5;
    } else if (u <= 300) {
        bill = 100 * 1.5 + 100 * 2.5 + (u - 200) * 4;
    } else {
        bill = 100 * 1.5 + 100 * 2.5 + 100 * 4 + (u - 300) * 6;
    }

    return bill;
}

void displayBill() {
    System.out.println("Customer Name: " + name);
    System.out.println("Units Consumed: " + units);
    System.out.println("Electricity Bill: Rs. " + calculateBill());
}
}

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

        System.out.print("Enter customer name: ");
        String name = sc.nextLine();

        System.out.print("Enter units consumed: ");
        int units = sc.nextInt();

        ElectricityBill bill = new ElectricityBill(name, units);
        bill.displayBill();
    }
}


```

## Output      Generated files

```

Enter customer name: xyz
Enter units consumed: 200
Customer Name: xyz
Units Consumed: 200
Electricity Bill: Rs. 400.0
|

```

 Compiled and executed in 36.263 sec(s)

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.

```
import java.util.Scanner;

class Vehicle {
    String vehicleType;

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

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

class Car extends Vehicle {
    String modelType;
    String companyName;

    Car(String vehicleType, String modelType, String companyName) {
        super(vehicleType);
        this.modelType = modelType;
        this.companyName = companyName;
    }

    void display() {
        super.display();
        System.out.println("Car Model: " + modelType);
        System.out.println("Company Name: " + companyName);
    }
}

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

        System.out.print("Enter vehicle type: ");
        String vehicleType = sc.nextLine();

        System.out.print("Enter car model: ");
        String modelType = sc.nextLine();
    }
}
```

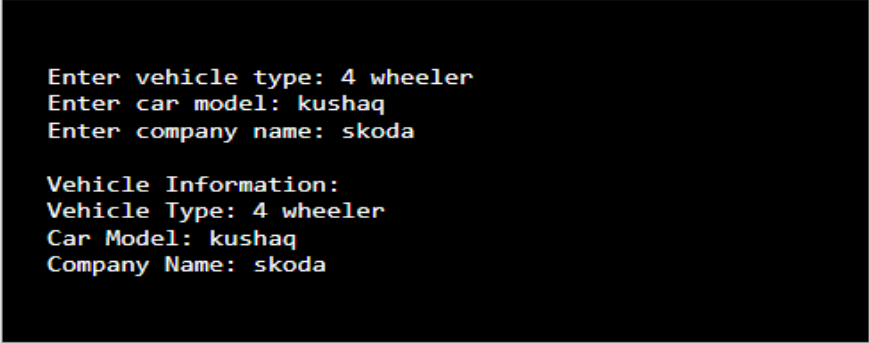
```

        System.out.print("Enter company name: ");
        String companyName = sc.nextLine();

        Car myCar = new Car(vehicleType, modelType, companyName);

        System.out.println("\nVehicle Information:");
        myCar.display();
    }
}

```



The screenshot shows an IDE with two tabs: "Output" and "Generated files". The "Output" tab is active, displaying the following text:

```

Enter vehicle type: 4 wheeler
Enter car model: kushaq
Enter company name: skoda

Vehicle Information:
Vehicle Type: 4 wheeler
Car Model: kushaq
Company Name: skoda

```

Below the output, a status bar indicates: "Compiled and executed in 38.762 sec(s)".

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 instance variable named interest Rate, and the "Current" class should contain instance variable called overdraft Limit. Define appropriate methods for all the classes to enable functionalities to check balance, deposit, and withdraw amount in Savings and Current account

```

import java.util.Scanner;

class Account {
    protected int accountNo;
    protected double balance;

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

    void checkBalance() {

```

```

        System.out.println("Account No: " + accountNo);
        System.out.println("Balance: " + balance);
    }

    void deposit(double amount) {
        balance += amount;
        System.out.println(amount + " deposited successfully.");
    }

    boolean withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println(amount + " withdrawn successfully.");
            return true;
        } else {
            System.out.println("Insufficient balance.");
            return false;
        }
    }
}

class Savings extends Account {
    private 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 of " + interest + " added. New Balance: " + balance);
    }
}

class Current extends Account {
    private double overdraftLimit;

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

```

```

@Override
boolean withdraw(double amount) {
    if (balance + overdraftLimit >= amount) {
        balance -= amount;
        System.out.println(amount + " withdrawn successfully.");
        return true;
    } else {
        System.out.println("Exceeded overdraft limit.");
        return false;
    }
}

}

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

        // Savings account example
        System.out.println("=== Savings Account ===");
        System.out.print("Enter account number: ");
        int savAccNo = sc.nextInt();
        System.out.print("Enter initial balance: ");
        double savBalance = sc.nextDouble();
        System.out.print("Enter interest rate (%): ");
        double rate = sc.nextDouble();

        Savings sav = new Savings(savAccNo, savBalance, rate);
        sav.checkBalance();
        sav.deposit(2000);
        sav.withdraw(1500);
        sav.addInterest();
        sav.checkBalance();

        // Current account example
        System.out.println("\n=== Current Account ===");
        System.out.print("Enter account number: ");
        int curAccNo = sc.nextInt();
        System.out.print("Enter initial balance: ");
        double curBalance = sc.nextDouble();
        System.out.print("Enter overdraft limit: ");
        double overdraft = sc.nextDouble();

        Current cur = new Current(curAccNo, curBalance, overdraft);
        cur.checkBalance();
    }
}

```

```

        cur.deposit(5000);
        cur.withdraw(8000);
        cur.checkBalance();
    }
}

```

```

=== Savings Account ===
Enter account number: 1213425176
Enter initial balance: 5000
Enter interest rate (%): 1
Account No: 1213425176
Balance: 5000.0
2000.0 deposited successfully.
1500.0 withdrawn successfully.
Interest of 55.0 added. New Balance: 5555.0
Account No: 1213425176
Balance: 5555.0

=== Current Account ===
Enter account number: 1452167561
Enter initial balance: 10000
Enter overdraft limit: 2000
Account No: 1452167561
Balance: 10000.0
5000.0 deposited successfully.
8000.0 withdrawn successfully.
Account No: 1452167561
Balance: 7000.0

i Compiled and executed in 72.295 sec(s)

```

**15. Write a program to demonstrate the use of 'this' and 'static' keyword.**

```

class Demo {
    private int value;
    private static int count = 0;

    Demo(int value) {
        this.value = value; // 'this' refers to the current object's instance variable
        count++; // static variable shared among all objects
    }

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

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

```

```

public class ThisStaticExample {
    public static void main(String[] args) {
        Demo obj1 = new Demo(10);
        Demo obj2 = new Demo(20);
        Demo obj3 = new Demo(30);

        obj1.display();
        obj2.display();
        obj3.display();

        Demo.showCount(); // static method called using class name
    }
}

```



The screenshot shows an IDE interface with two tabs: 'Output' and 'Generated files'. The 'Output' tab is active, displaying the following text in a monospaced font:

```

Value: 10
Value: 20
Value: 30
Total objects created: 3

```

Below the output, a status bar indicates: 'Compiled and executed in 1.85 sec(s)'.

- 16. Describe 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**

```

import java.util.Scanner;
abstract class Shape {
    abstract void area();
}

class Triangle extends Shape {
    private double base, height;

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

    @Override
    void area() {

```

```

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

```

```

class Rectangle extends Shape {
    private double length, width;

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

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

```

```

class Circle extends Shape {
    private double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    @Override
    void area() {
        double area = 3.14159 * radius * radius;
        System.out.println("Area of Circle: " + area);
    }
}

```

```

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

        System.out.println("=== Triangle ===");
        System.out.print("Enter base: ");
        double base = sc.nextDouble();
        System.out.print("Enter height: ");
        double height = sc.nextDouble();
        Shape t = new Triangle(base, height);
    }
}

```

```

        t.area();

        System.out.println("\n=== Rectangle ===");
        System.out.print("Enter length: ");
        double length = sc.nextDouble();
        System.out.print("Enter width: ");
        double width = sc.nextDouble();
        Shape r = new Rectangle(length, width);
        r.area();

        System.out.println("\n=== Circle ===");
        System.out.print("Enter radius: ");
        double radius = sc.nextDouble();
        Shape c = new Circle(radius);
        c.area();
    }
}

```

Output

Generated files


```

=== Triangle ===
Enter base: 14
Enter height: 7
Area of Triangle: 49.0

=== Rectangle ===
Enter length: 12
Enter width: 16
Area of Rectangle: 192.0

=== Circle ===
Enter radius: 5
Area of Circle: 78.53975

```

 Compiled and executed in 48.85 sec(s)

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

```

import java.util.Scanner;
// Interface for student details
interface Student {
    void getDetails();
    void displayDetails();
}

// Interface for marks
interface Marks {

```

```
void getMarks();  
void displayMarks();  
}
```

```
// Class implementing both interfaces  
class Result implements Student, Marks {  
    private String name;  
    private int rollNo;  
    private int marks1, marks2, marks3;  
    private int total;  
    private double percentage;
```

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

```
    public void getDetails() {  
        System.out.print("Enter student name: ");  
        name = sc.nextLine();  
        System.out.print("Enter roll number: ");  
        rollNo = sc.nextInt();  
    }
```

```
    public void displayDetails() {  
        System.out.println("\n--- Student Details ---");  
        System.out.println("Name: " + name);  
        System.out.println("Roll Number: " + rollNo);  
    }
```

```
    public void getMarks() {  
        System.out.print("Enter marks for Subject 1: ");  
        marks1 = sc.nextInt();  
        System.out.print("Enter marks for Subject 2: ");  
        marks2 = sc.nextInt();  
        System.out.print("Enter marks for Subject 3: ");  
        marks3 = sc.nextInt();  
  
        total = marks1 + marks2 + marks3;  
        percentage = total / 3.0;  
    }
```

```
    public void displayMarks() {
```

```

        System.out.println("\n--- Marks & Result ---");
        System.out.println("Subject 1: " + marks1);
        System.out.println("Subject 2: " + marks2);
        System.out.println("Subject 3: " + marks3);
        System.out.println("Total: " + total);
        System.out.println("Percentage: " + percentage + "%");

        if (percentage >= 60)
            System.out.println("Grade: A");
        else if (percentage >= 50)
            System.out.println("Grade: B");
        else if (percentage >= 35)
            System.out.println("Grade: C");
        else
            System.out.println("Grade: F (Fail)");
    }
}

public class StudentResultSheet {
    public static void main(String[] args) {
        Result student = new Result();

        student.getDetails();
        student.getMarks();
        student.displayDetails();
        student.displayMarks();
    }
}

```

## Output      Generated files

```

Enter student name: alice
Enter roll number: 25
Enter marks for Subject 1: 90
Enter marks for Subject 2: 86
Enter marks for Subject 3: 89

--- Student Details ---
Name: alice
Roll Number: 25

--- Marks & Result ---
Subject 1: 90
Subject 2: 86
Subject 3: 89
Total: 265
Percentage: 88.33333333333333%
Grade: A
|

```

 Compiled and executed in 58.316 sec(s)

**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.**

Create Student class in student package

// File: Student.java

package student;

```
public class Student {  
    protected String name;  
    protected int rollNo;
```

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

```
    public void displayStudentInfo() {  
        System.out.println("Student Name: " + name);  
        System.out.println("Roll Number: " + rollNo);  
    }  
}
```

Create Result class in exam package

// File: Result.java

package exam;

import student.Student;

```
public class Result extends Student {  
    private int marks1, marks2, marks3;  
    private int total;  
    private double percentage;
```

```
    public Result(String name, int rollNo, int marks1, int marks2, int marks3) {  
        super(name, rollNo); // call Student constructor  
        this.marks1 = marks1;  
        this.marks2 = marks2;  
        this.marks3 = marks3;  
        calculateResult();  
    }
```

```
    private void calculateResult() {  
        total = marks1 + marks2 + marks3;  
        percentage = total / 3.0;
```

```

    }

    public void displayResult() {
        displayStudentInfo();
        System.out.println("Marks:");
        System.out.println("Subject 1: " + marks1);
        System.out.println("Subject 2: " + marks2);
        System.out.println("Subject 3: " + marks3);
        System.out.println("Total: " + total);
        System.out.println("Percentage: " + percentage + "%");

        if (percentage >= 60)
            System.out.println("Grade: A");
        else if (percentage >= 50)
            System.out.println("Grade: B");
        else if (percentage >= 35)
            System.out.println("Grade: C");
        else
            System.out.println("Grade: F (Fail)");
    }
}

```

#### Main program to generate mark sheet

```

// File: MarkSheet.java
import java.util.Scanner;
import exam.Result;

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

        System.out.print("Enter student name: ");
        String name = sc.nextLine();

        System.out.print("Enter roll number: ");
        int rollNo = sc.nextInt();

        System.out.print("Enter marks for Subject 1: ");
        int marks1 = sc.nextInt();
        System.out.print("Enter marks for Subject 2: ");
        int marks2 = sc.nextInt();
        System.out.print("Enter marks for Subject 3: ");
        int marks3 = sc.nextInt();
    }
}

```

```

        Result studentResult = new Result(name, rollNo, marks1, marks2, marks3);
        System.out.println("\n--- Student Mark Sheet ---");
        studentResult.displayResult();
    }
}

```

```

Enter student name: John
Enter roll number: 101
Enter marks for Subject 1: 75
Enter marks for Subject 2: 68
Enter marks for Subject 3: 82

--- Student Mark Sheet ---
Student Name: John
Roll Number: 101
Marks:
Subject 1: 75
Subject 2: 68
Subject 3: 82
Total: 225
Percentage: 75.0%
Grade: A

```

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

```

class MyThread implements Runnable {
    private String threadName;

    MyThread(String name) {
        this.threadName = name;
    }

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(threadName + " is running: " + i);
            try {
                Thread.sleep(500); // pause for 0.5 seconds
            } catch (InterruptedException e) {
                System.out.println(threadName + " interrupted.");
            }
        }
        System.out.println(threadName + " finished execution.");
    }
}

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

```


```

MyThread task1 = new MyThread("Thread-1");
MyThread task2 = new MyThread("Thread-2");

Thread t1 = new Thread(task1);
Thread t2 = new Thread(task2);

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

```

Output	Generated files
<pre> Thread-1 is running: 1 Thread-2 is running: 1 Thread-2 is running: 2 Thread-1 is running: 2 Thread-2 is running: 3 Thread-1 is running: 3 Thread-2 is running: 4 Thread-1 is running: 4 Thread-1 is running: 5 Thread-2 is running: 5 Thread-1 finished execution. Thread-2 finished execution.   </pre>	
<p> Compiled and executed in 4.217 sec(s)</p>	

**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 ThreadOne implements Runnable {
    @Override
    public void run() {
        try {
            while (true) {
                System.out.println("Thread1");
                Thread.sleep(2000); // 2000 milliseconds = 2 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("Thread1 interrupted.");
        }
    }
}

```

```

class ThreadTwo implements Runnable {
    @Override
    public void run() {

```

```

    try {
        while (true) {
            System.out.println("Thread2");
            Thread.sleep(4000); // 4000 milliseconds = 4 seconds
        }
    } catch (InterruptedException e) {
        System.out.println("Thread2 interrupted.");
    }
}
}

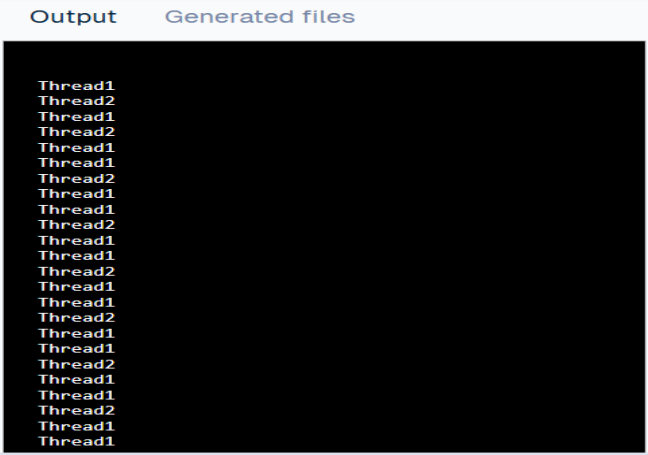
```

```

public class TwoThreadsExample {
    public static void main(String[] args) {
        Thread t1 = new Thread(new ThreadOne());
        Thread t2 = new Thread(new ThreadTwo());

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

```



The screenshot shows a Java IDE with two tabs: 'Output' and 'Generated files'. The 'Output' tab is active and displays the following text:

```

Thread1
Thread2
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1
Thread2
Thread1
Thread1

```

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

```

import java.util.Scanner;

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

        try {
            System.out.print("Enter numerator: ");
            int numerator = sc.nextInt();

```

```

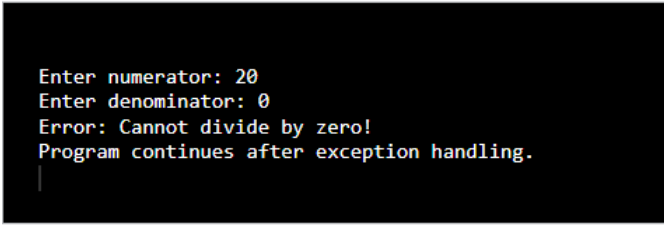
        System.out.print("Enter denominator: ");
        int denominator = sc.nextInt();

        int result = numerator / denominator; // may cause ArithmeticException
        System.out.println("Result: " + result);

    } catch (ArithmeticException e) {
        System.out.println("Error: Cannot divide by zero!");
    } catch (Exception e) {
        System.out.println("Error: Invalid input!");
    }

    System.out.println("Program continues after exception handling.");
}
}

```



```

Output    Generated files

Enter numerator: 20
Enter denominator: 0
Error: Cannot divide by zero!
Program continues after exception handling.

```

### 23. Write a java program which use multiple catch blocks

```

import java.util.Scanner;

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

        try {
            System.out.print("Enter an integer: ");
            int num1 = sc.nextInt();

            System.out.print("Enter another integer: ");
            int num2 = sc.nextInt();

            int result = num1 / num2; // may cause ArithmeticException
            System.out.println("Division Result: " + result);
        }
    }
}

```

```

        int[] arr = new int[3];
        arr[5] = 10; // will cause ArrayIndexOutOfBoundsException

    } catch (ArithmeticException e) {
        System.out.println("Error: Division by zero is not allowed!");
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Error: Array index out of bounds!");
    } catch (Exception e) {
        System.out.println("Error: Some other exception occurred.");
    }

    System.out.println("Program continues after handling exceptions.");
}
}

```

```

Output    Generated files

Enter an integer: 10
Enter another integer: 2
Division Result: 5
Error: Array index out of bounds!
Program continues after handling exceptions.

```

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

```

import java.util.Scanner;

// Custom Exception Class
class AgeException extends Exception {
    public AgeException(String message) {
        super(message);
    }
}

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

        System.out.print("Enter your age: ");
        int age = sc.nextInt();

        try {
            if (age < 18) {

```

```

        throw new AgeException("Age must be 18 or older to register.");
    } else {
        System.out.println("Registration successful!");
    }
} catch (AgeException e) {
    System.out.println("Custom Exception Caught: " + e.getMessage());
}

System.out.println("Program continues after exception handling.");
}
}

```

Output
Generated files

```

Enter your age: 15
Custom Exception Caught: Age must be 18 or older to register
Program continues after exception handling.
|

```

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

```

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

public class AWTButtonLabelExample implements ActionListener {
    Frame frame;
    Label label;
    Button button1, button2;

    AWTButtonLabelExample() {
        // Create frame
        frame = new Frame("AWT Button and Label Example");
        frame.setSize(400, 200);
        frame.setLayout(new FlowLayout());

        // Create label
        label = new Label("Press a button");
        frame.add(label);

        // Create buttons
        button1 = new Button("Button 1");
        button2 = new Button("Button 2");
    }
}

```

```

frame.add(button1);
frame.add(button2);

// Add action listeners
button1.addActionListener(this);
button2.addActionListener(this);

// Close window
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        frame.dispose();
    }
});

frame.setVisible(true);
}

// Handle button click events
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button1) {
        label.setText("Button 1 clicked");
    } else if (e.getSource() == button2) {
        label.setText("Button 2 clicked");
    }
}

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

```

## 26. Write a program to handle Check Boxes using AWT Controls

```

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

public class AWTCheckboxExample implements ItemListener {
    Frame frame;
    Label label;
    Checkbox checkbox1, checkbox2, checkbox3;

    AWTCheckboxExample() {
        // Create frame

```

```

frame = new Frame("AWT Checkbox Example");
frame.setSize(400, 200);
frame.setLayout(new FlowLayout());

// Create label
label = new Label("Select your hobbies:");
frame.add(label);

// Create checkboxes
checkbox1 = new Checkbox("Reading");
checkbox2 = new Checkbox("Traveling");
checkbox3 = new Checkbox("Music");

frame.add(checkbox1);
frame.add(checkbox2);
frame.add(checkbox3);

// Add ItemListener
checkbox1.addItemListener(this);
checkbox2.addItemListener(this);
checkbox3.addItemListener(this);

// Close window
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        frame.dispose();
    }
});

frame.setVisible(true);
}

@Override
public void itemStateChanged(ItemEvent e) {
    StringBuilder selected = new StringBuilder("Selected hobbies: ");
    if (checkbox1.getState()) selected.append("Reading ");
    if (checkbox2.getState()) selected.append("Traveling ");
    if (checkbox3.getState()) selected.append("Music ");
    label.setText(selected.toString());
}

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

```

## 27. Write a program to handle Lists and Scroll Bars using AWT Controls.

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

public class AWTListScrollbarExample implements ActionListener {
    Frame frame;
    List list;
    Scrollbar scrollbar;
    Label label;
    Button showSelectionButton;

    AWTListScrollbarExample() {
        // Create frame
        frame = new Frame("AWT List and Scrollbar Example");
        frame.setSize(400, 300);
        frame.setLayout(new FlowLayout());

        // Create a label
        label = new Label("Select an item and adjust the scrollbar:");
        frame.add(label);

        // Create List with multiple items and visible rows
        list = new List(5, false); // 5 visible rows, single selection
        list.add("Apple");
        list.add("Banana");
        list.add("Cherry");
        list.add("Mango");
        list.add("Orange");
        list.add("Grapes");
        list.add("Pineapple");
        frame.add(list);

        // Button to show selected item
        showSelectionButton = new Button("Show Selection");
        showSelectionButton.addActionListener(this);
        frame.add(showSelectionButton);

        // Create Scrollbar (vertical)
        scrollbar = new Scrollbar(Scrollbar.VERTICAL, 50, 10, 0, 100);
        scrollbar.addAdjustmentListener(new AdjustmentListener() {
            @Override
            public void adjustmentValueChanged(AdjustmentEvent e) {
                label.setText("Scrollbar Value: " + scrollbar.getValue());
            }
        });
    }
}
```

```

        }
    });
    frame.add(scrollbar);

    // Close window
    frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            frame.dispose();
        }
    });

    frame.setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    String selectedItem = list.getSelectedItemAt();
    if (selectedItem != null) {
        label.setText("You selected: " + selectedItem);
    } else {
        label.setText("No item selected.");
    }
}

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

```