Program No: 5

Date of Submission: 07-03-2025

Aim:

Write a java program to multiply two given matrices.

Algorithm:

Step 1: Input the dimensions of the matrices

- Read dimensions of matrix `A` → `m x n`
- Read dimensions of matrix `B` → `n2 x p`

Step 2: Check multiplication condition

- If `n != n2`, print **"Matrix multiplication not possible"** and exit.

Step 3: Input the elements of matrix `A`

- For `i = 0` to `m - 1`
    - For `j = 0` to `n - 1`
        - Read element `A[i][j]`

Step 4: Input the elements of matrix `B`

- For `i = 0` to `n2 - 1`
    - For `j = 0` to `p - 1`
        - Read element `B[i][j]`

Step 5: Initialize the result matrix `C` with zeros

- Create matrix `C` of size `m x p` and initialize all elements to `0`

Step 6: Perform matrix multiplication

- For `i = 0` to `m - 1`
    - For `j = 0` to `p - 1`
        - For `k = 0` to `n - 1`
            - `C[i][j] = C[i][j] + (A[i][k] * B[k][j])`

Step 7: Output the result matrix

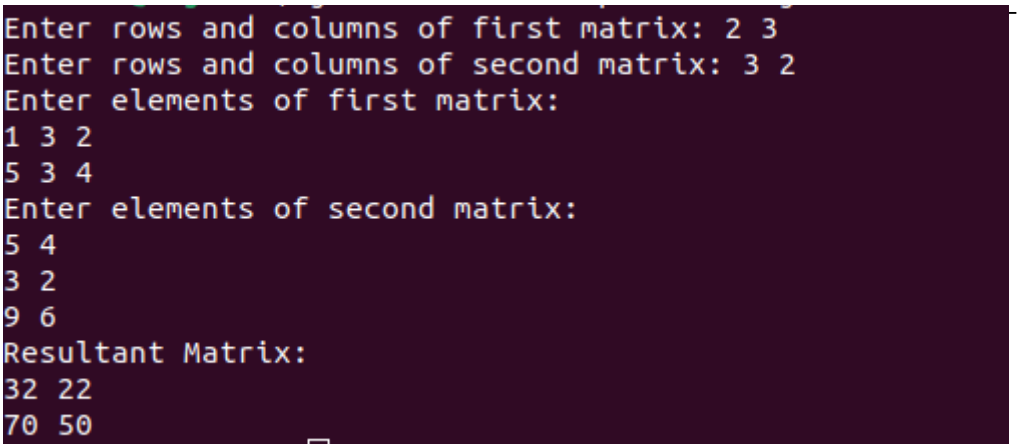
- For `i = 0` to `m - 1`
    - For `j = 0` to `p - 1`
        - Print `C[i][j]`

Documented Program Code:

```
import java.util.Scanner;

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

        // Input dimensions of the first matrix
        System.out.print("Enter rows and columns of first matrix: ");
        int m = sc.nextInt();
        int n = sc.nextInt();

        // Input dimensions of the second matrix
        System.out.print("Enter rows and columns of second matrix: ");
        int n2 = sc.nextInt();
        int p = sc.nextInt();

        // Check if matrices can be multiplied
        if (n != n2) {
            System.out.println("Matrix multiplication is not possible (column of A != row of B)");
            return;
        }

        // Input first matrix
        int[][] A = new int[m][n];
        System.out.println("Enter elements of first matrix:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                A[i][j] = sc.nextInt();
            }
        }

        // Input second matrix
        int[][] B = new int[n][p];
        System.out.println("Enter elements of second matrix:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < p; j++) {
                B[i][j] = sc.nextInt();
            }
        }
```

```
        // Multiply matrices
        int[][] C = new int[m][p];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < p; j++) {
                for (int k = 0; k < n; k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }

        // Print the result matrix
        System.out.println("Resultant Matrix:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < p; j++) {
                System.out.print(C[i][j] + " ");
            }
            System.out.println();
        }

        sc.close();
    }
}
```

Output Screens of the Program:



Program No: 6

Date of Submission: 07-03-2025

Aim:

Create a class 'Account' to represent a bank account. Write a program to deposit and withdraw amounts from the account.

Algorithm:

Step 1: Define an Account class

- Create a class Account with a balance attribute.
    - Initialize balance = 5000 (Initial balance).

Step 2: Create a method to deposit money

- deposit(double amount) method:
    - Add the deposit amount to the balance: balance=balance+amount
    - Display the deposit message.

Step 3: Create a method to withdraw money

- withdraw(double amount) method:
    - If the amount > balance, display "Not enough balance!"
    - Else:
        - Subtract the withdrawal amount from the balance: balance=balance−amount
        - Display the withdrawal message.

Step 4: Create a method to display balance

- display() method:
    - Print the current balance.

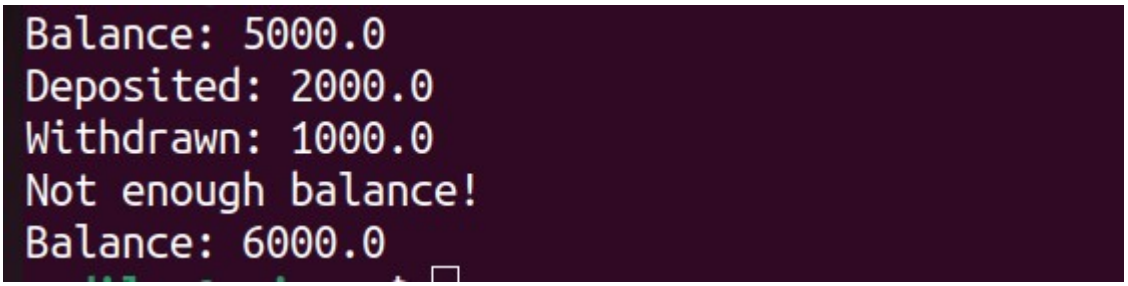
Step 5: Create the main() method

1. Create an Account object.
2. Call display() to print the initial balance.
3. Call deposit(2000) to add money to the account.
4. Call withdraw(1000) to withdraw money from the account.
5. Call withdraw(7000) to test the insufficient funds condition.
6. Call display() to print the final balance.

Documented Program Code:

```
class Account {
   double balance = 5000; // Initial balance

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

   void withdraw(double amount) {
      if (amount > balance) {
         System.out.println("Not enough balance!");
      } else {
         balance -= amount;
         System.out.println("Withdrawn: " + amount);
      }
   }

   void display() {
      System.out.println("Balance: " + balance);
   }

   public static void main(String[] args) {
      Account acc = new Account();
      acc.display();
      acc.deposit(2000);
      acc.withdraw(1000);
      acc.withdraw(7000);
      acc.display();
   }
}
```

Output Screens of the Program:



-

Program No: 7

Date of Submission: 07-03-2025

Aim:

Create a class Time with hh, mm, ss as data members. Write a java program to find the sum of two time intervals (Hint: Use object as parameter to function).

Algorithm:

Step 1: Define a Time class

- Create a class Time with three integer attributes:
  - hh → for hours
  - mm → for minutes
  - ss → for seconds

Step 2: Create a constructor to initialize time

- Input hh, mm, and ss values.
- Call the normalizeTime() method to adjust overflow (seconds ≥ 60, minutes ≥ 60).

Step 3: Define normalizeTime() method to handle overflow

- If ss >= 60, convert excess seconds into minutes:
  - mm = mm + (ss / 60)
  - ss = ss % 60
- If mm >= 60, convert excess minutes into hours:
  - hh = hh + (mm / 60)
  - mm = mm % 60
- Keep hh in 24-hour format:
  - hh = hh % 24

Step 4: Define a method to add two time objects

- Create addTime(Time t) method:
  - Add hh, mm, and ss from both time objects.
  - Call normalizeTime() to adjust overflow.
  - Return the new Time object.

Step 5: Define a method to display time

- Create display() method to print time in HH:MM:SS format using printf.

Step 6: Define a method to take user input

- Create inputTime(Scanner sc) method:

- Take input for hours, minutes, and seconds.
- Create and return a Time object using the constructor.

Step 7: Create main() method

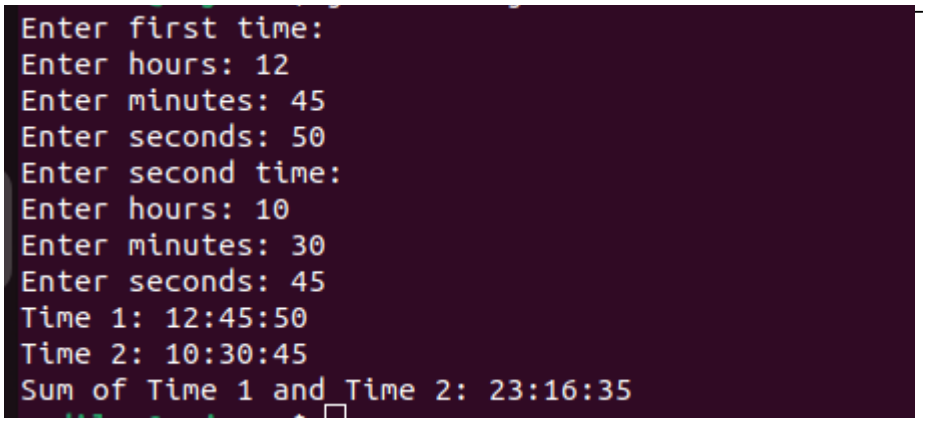
1. Create a Scanner object.
2. Take input for the first time using inputTime().
3. Take input for the second time using inputTime().
4. Display both times using display().
5. Add both times using addTime() and store in a new Time object.
6. Display the result.
7. Close the Scanner object.

Documented Program Code:

```
import java.util.Scanner;

class Time {
   private int hh, mm, ss;

   // Constructor to initialize time
   public Time(int hh, int mm, int ss) {
      this.hh = hh;
      this.mm = mm;
      this.ss = ss;
      normalizeTime();
   }

   // Method to normalize time (handle overflow)
   private void normalizeTime() {
      if (ss >= 60) {
         mm += ss / 60;
         ss = ss % 60;
      }
      if (mm >= 60) {
         hh += mm / 60;
         mm = mm % 60;
      }
      hh = hh % 24; // Keep hours in 24-hour format
   }
```

```
    // Method to add two Time objects
    public Time addTime(Time t) {
        return new Time(this.hh + t.hh, this.mm + t.mm, this.ss + t.ss);
    }

    // Display time in HH:MM:SS format
    public void display() {
        System.out.printf("%02d:%02d:%02d\n", hh, mm, ss);
    }

    // Method to take user input
    public static Time inputTime(Scanner sc) {
        System.out.print("Enter hours: ");
        int h = sc.nextInt();
        System.out.print("Enter minutes: ");
        int m = sc.nextInt();
        System.out.print("Enter seconds: ");
        int s = sc.nextInt();
        return new Time(h, m, s);
    }

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

        System.out.println("Enter first time:");
        Time t1 = inputTime(sc);

        System.out.println("Enter second time:");
        Time t2 = inputTime(sc);

        System.out.print("Time 1: ");
        t1.display();

        System.out.print("Time 2: ");
        t2.display();

        Time t3 = t1.addTime(t2);
        System.out.print("Sum of Time 1 and Time 2: ");
        t3.display();

        sc.close();
    }
}
```

Output Screens of the Program:



Program No: 8

Date of Submission: 07-03-2025

Aim:

Write a program to add two complex numbers using this function.

Algorithm:

Step 1: Define a ComplexNumber class

- Create a class ComplexNumber with two attributes:
  - real → for the real part of the complex number
  - imaginary → for the imaginary part of the complex number

Step 2: Create a constructor to initialize the complex number

- Assign the values of real and imaginary using the constructor.

Step 3: Define a method to add two complex numbers

- Create a static method add(ComplexNumber c1, ComplexNumber c2):
  - Add the real parts: real=c1.real+c2.real
  - Add the imaginary parts: imaginary=c1.imaginary+c2.imaginary
  - Return a new ComplexNumber object with the calculated values.

Step 4: Define a method to display the complex number

- Create display() method to print complex numbers in the format:

real+imaginaryi

Step 5: Create the main() method

1. Create a Scanner object to take user input.
2. Take input for the real and imaginary parts of the first complex number.
3. Take input for the real and imaginary parts of the second complex number.
4. Create two ComplexNumber objects using the constructor.
5. Call the add() method to add the two complex numbers.
6. Display the result using display() method.
7. Close the Scanner object.

Documented Program Code:

```
import java.util.Scanner;

class ComplexNumber {
   double real;
   double imaginary;

   // Constructor
   ComplexNumber(double real, double imaginary) {
      this.real = real;
      this.imaginary = imaginary;
   }

   // Function to add two complex numbers
   static ComplexNumber add(ComplexNumber c1, ComplexNumber c2) {
      return new ComplexNumber(c1.real + c2.real, c1.imaginary + c2.imaginary);
   }

   // Function to display a complex number
   void display() {
      System.out.println(real + " + " + imaginary + "i");
   }

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

      // Taking input for the first complex number
      System.out.print("Enter real part of first complex number: ");
      double real1 = scanner.nextDouble();
      System.out.print("Enter imaginary part of first complex number: ");
      double imag1 = scanner.nextDouble();

      // Taking input for the second complex number
      System.out.print("Enter real part of second complex number: ");
      double real2 = scanner.nextDouble();
      System.out.print("Enter imaginary part of second complex number: ");
      double imag2 = scanner.nextDouble();

      // Creating complex number objects
      ComplexNumber num1 = new ComplexNumber(real1, imag1);
      ComplexNumber num2 = new ComplexNumber(real2, imag2);
```

```
        // Adding two complex numbers
        ComplexNumber result = ComplexNumber.add(num1, num2);

        // Displaying the result
        System.out.print("Sum: ");
        result.display();

        scanner.close();
    }
}
```

Output Screens of the Program: