Experiment 1

Student Name: Diksha
UID: 23BCS10994
Branch: CSE
Section/Group: KRG_2B
Semester: 5th
Date of Performance: 12/08/25
Subject Name: PBLJ
Subject Code: 23CSH-304

1. Aim:

To design and implement Java programs for analyzing strings, performing matrix operations, and simulating a basic banking system using object-oriented concepts.

A)– Easy Level:

- To analyze a user-input string and count vowels, consonants, digits, and special characters.

B)– Medium Level:

- To implement matrix operations (addition, subtraction, multiplication) with validation of dimensions.

C) – Hard Level:

- To create a basic banking system with account creation, deposit, and withdrawal functionalities ensuring no overdraft.

2. Objective:

- To understand string manipulation and character classification in Java.
- To apply conditional statements for analyzing vowels, consonants, digits, and special characters.
- To implement multidimensional arrays and validate dimensions for performing matrix operations.
- To apply nested loops for addition, subtraction, and multiplication of matrices.
- To design and implement a basic banking system using object-oriented programming concepts.

3. JAVA script and output:

EASY-LEVEL PROBLEM

```
import java.util.Scanner;

public class StringAnalysis {     public
static void main(String[] args)
{         Scanner sc = new
Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();

        int vowels = 0, consonants = 0, digits = 0, special = 0;
str = str.toLowerCase();

        for (int i = 0; i < str.length(); i++)
{             char ch = str.charAt(i);
if ("aeiou".indexOf(ch) != -1)
{                 vowels++;
            } else if (ch >= 'a' && ch <= 'z')
{                 consonants++;
            } else if (ch >= '0' && ch <= '9') {
               digits++;
            } else if (ch != ' ')
{                 special++;
            }
        }

        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
        System.out.println("Digits: " + digits);
        System.out.println("Special Characters: " + special);
    }
}
```

Output:

```
Enter a string: Hey 123!
Vowels: 1
Consonants: 2
Digits: 3
Special Characters: 1
BUILD SUCCESSFUL (total time: 21 seconds)
```

MEDIUM LEVEL PROBLEM:

```
import java.util.Scanner;

public class MatrixOperations {     public
static void main(String[] args)
{       Scanner sc = new
Scanner(System.in);
      System.out.print("Enter rows and columns for Matrix A: ");
int r1 = sc.nextInt(), c1 = sc.nextInt();
      System.out.print("Enter rows and columns for Matrix B: ");
      int r2 = sc.nextInt(), c2 = sc.nextInt();

      int[][] A = new int[r1][c1];
      int[][] B = new int[r2][c2];

      System.out.println("Enter elements of Matrix A:");
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 Matrix B:");
for (int i = 0; i < r2; i++)          for (int j = 0; j < c2; j++)
            B[i][j] = sc.nextInt();
      if (r1 == r2 && c1 == c2)
{          System.out.println("Addition:");
for (int i = 0; i < r1; i++) {
            for (int j = 0; j < c1; j++)
               System.out.print((A[i][j] + B[i][j]) + " ");
            System.out.println();
         }
```

```
        System.out.println("Subtraction:");
        for (int i = 0; i < r1; i++)
{           for (int j = 0; j < c1; j++)
             System.out.print((A[i][j] - B[i][j]) + " ");
          System.out.println();
        }
      } else {
        System.out.println("Addition and Subtraction not possible");
      }
      if (c1 == r2) {
        System.out.println("Multiplication:");
int[][] result = new int[r1][c2];          for (int
i = 0; i < r1; i++) {             for (int j = 0; j <
c2; j++) {               for (int k = 0; k < c1;
k++)                  result[i][j] += A[i][k] *
B[k][j];
System.out.print(result[i][j] + " ");
          }
          System.out.println();
        }
      } else {
        System.out.println("Multiplication not possible");
      }
   }
}
```

Output:

```
run:
Enter rows and columns for Matrix A: 2 2
Enter rows and columns for Matrix B: 2 2
Enter elements of Matrix A:
4 0 1 2
Enter elements of Matrix B:
1 25 6 1
Addition:
5 25
7 3
Subtraction:
3 -25
-5 1
Multiplication:
4 100
13 27
BUILD SUCCESSFUL (total time: 13 seconds)
```

HARD LEVEL PROBLEM

```
import java.util.Scanner;

class BankAccount
{    private String name;
private String accNumber;
   private double balance;

   public BankAccount(String name, String accNumber, double balance)
{       this.name = name;        this.accNumber = accNumber;
      this.balance = balance;
   }

   public void deposit(double amount) {
      balance += amount;
      System.out.println("Deposit successful! Current Balance: " + balance);    }

   public void withdraw(double amount) {
      if (amount <= balance)
{          balance -= amount;
         System.out.println("Withdrawal successful! Current Balance: " + balance);
      } else {
         System.out.println("Error: Insufficient funds. Current Balance: " + balance);
      }
   }
}

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

      System.out.println("Create Account:");
      System.out.print("Name: ");
      String name = sc.nextLine();
      System.out.print("Account Number: ");
```

```
        String accNo = sc.nextLine();
System.out.print("Initial Balance: ");
        double bal = sc.nextDouble();

        BankAccount account = new BankAccount(name, accNo, bal);

        System.out.print("Enter amount to Deposit: ");
double dep = sc.nextDouble();        account.deposit(dep);

        System.out.print("Enter amount to Withdraw: ");
double wd = sc.nextDouble();
        account.withdraw(wd);
    }
}
```

Output:

```
Create Account:
Name: Akshara
Account Number: 1234500
Initial Balance: 50000
Enter amount to Deposit: 34000
Deposit successful! Current Balance: 84000.0
Enter amount to Withdraw: 10000
Withdrawal successful! Current Balance: 74000.0
BUILD SUCCESSFUL (total time: 33 seconds)
```