Experiment-1.1

Student Name: Gagnesh Kakkar
UID: 23BCS11196
Branch: B.E-C.S.E
Section/Group: 23KRG-2B
Semester: 5th
Date of Performance: 18/08/2025
Subject Name: PBLJ
Subject Code: 23CSH-304

Easy Level

1. **Aim:** Create Java programs to manage product details, library systems, and student information using classes, inheritance, and abstraction.

2. **Objective:** To understand string manipulation in Java.

3. **Input/Apparatus Used:** Java basic input and string handling.

4. **Procedure:**
   Step1: Prompt the user to enter a string.
   Step2: Traverse each character in the string.
   Step3: Classify each character using conditions:
   - If the character is a vowel (a, e, i, o, u), increment the vowel count.
   - If it is a consonant (alphabetic and not a vowel), increment the consonant count.
   - If it is a digit (0–9), increment the digit count.
   - If it is none of the above and not a space, it is a special character.
   Step4: Print the counts of vowels, consonants, digits, and special characters.

   Sample Input:
   Enter a string: Hello World 2024!

   Sample Output:
   Vowels: 3
   Consonants: 7
   Digits: 4

Special Characters: 3

5. Code:

```java
package Experiments;
import java.util.Scanner;

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

        System.out.print("Enter a string: ");
        String input = sc.nextLine();

        int vowels = 0, consonants = 0, digits = 0, specialChars = 0;

        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);

            if (Character.isLetter(ch)) {
                char lower = Character.toLowerCase(ch);
                if (lower == 'a' || lower == 'e' || lower == 'i' || lower == 'o' || lower == 'u') {
                    vowels++;
                } else {
                    consonants++;
                }
            } else if (Character.isDigit(ch)) {
                digits++;
            } else if (!Character.isWhitespace(ch)) {
                specialChars++;
            }
        }

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

    }
}
```

6. Output:

```
Enter a string: GAGNESH@015
Vowels: 2
Consonants: 5
Digits: 3
Special Characters: 1

Process finished with exit code 0
```

Medium Level

1. Aim: Write a Java program to perform matrix operations (addition, subtraction, and multiplication) on two matrices provided by the user. The program should: Check the dimensions of the matrices to ensure valid operations.

2. Objective: Understand multidimensional array manipulation and matrix operation validation.

3. Input/Apparatus Used: Java multidimensional arrays and control structures.

4. Procedure:

Step1: Accept input from the user for two matrices (2D arrays).
Step2: Check that the dimensions of matrices are valid for the desired operations:
- For addition/subtraction: dimensions must be equal.
- For multiplication: columns of Matrix A = rows of Matrix B.

Step3: Use nested loops to perform:
- Addition: result[i][j] = matrixA[i][j] + matrixB[i][j]
- Subtraction: result[i][j] = matrixA[i][j] - matrixB[i][j]
- Multiplication: result[i][j] = sum(matrixA[i][k] * matrixB[k][j])

Step4: Display the resulting matrices.

Sample Input:
Matrix 1:
1 2
3 4

Matrix 2:
5 6
7 8

Sample Output:
Addition:
6 8
10 12

Subtraction:
-4 -4
-4 -4

Multiplication:
19 22
43 50

5. Code:

```
                    /*MEDIUM LEVEL*/
class MatrixOperations {

    public static int[][] readMatrix(Scanner sc, int rows, int cols) {  2 usages
        int[][] matrix = new int[rows][cols];
        System.out.println("Enter elements (" + rows + "x" + cols + "):");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
        return matrix;
    }

    public static void displayMatrix(int[][] matrix) {  5 usages
        for (int[] row : matrix) {
            for (int val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }

    public static int[][] add(int[][] A, int[][] B) {  1 usage
        int rows = A.length;
        int cols = A[0].length;
        int[][] result = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = A[i][j] + B[i][j];
            }
        }
        return result;
    }

    public static int[][] subtract(int[][] A, int[][] B) {  1 usage
        int rows = A.length;
        int cols = A[0].length;
        int[][] result = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = A[i][j] - B[i][j];
            }
        }
```

```java
    public static int[][] multiply(int[][] A, int[][] B) {  1 usage
        int rowsA = A.length, colsA = A[0].length, colsB = B[0].length;
        int[][] result = new int[rowsA][colsB];

        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                result[i][j] = 0;
                for (int k = 0; k < colsA; k++) {
                    result[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        return result;
    }

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

        System.out.print("Enter rows and columns of Matrix A: ");
        int rowsA = sc.nextInt(), colsA = sc.nextInt();
        int[][] A = readMatrix(sc, rowsA, colsA);

        System.out.print("Enter rows and columns of Matrix B: ");
        int rowsB = sc.nextInt(), colsB = sc.nextInt();
        int[][] B = readMatrix(sc, rowsB, colsB);

        System.out.println("\nMatrix A:");
        displayMatrix(A);
        System.out.println("Matrix B:");
        displayMatrix(B);

        if (rowsA == rowsB && colsA == colsB) {
            System.out.println("\nAddition:");
            displayMatrix(add(A, B));

            System.out.println("Subtraction:");
            displayMatrix(subtract(A, B));
        } else {
            System.out.println("\nAddition and Subtraction not possible (dimension mismatch).");
        }

        if (colsA == rowsB) {
            System.out.println("\nMultiplication:");
            displayMatrix(multiply(A, B));
        } else {
            System.out.println("\nMultiplication not possible (Matrix A columns != Matrix B rows).");
        }
    }
```

6. Output:

```
Enter rows and columns of Matrix A: 2 2
Enter elements (2x2):
2 3 6 5
Enter rows and columns of Matrix B: 2 2
Enter elements (2x2):
9 6 1 4

Matrix A:
2 3
6 5
Matrix B:
9 6
1 4

Addition:
11 9
7 9
Subtraction:
-7 -3
5 1

Multiplication:
21 24
59 56
```

Hard Level

1. **Aim:** Create a Java program to implement a basic banking system with the following features: Account creation (Name, Account Number). Deposit and withdrawal operations. Prevent overdraft by checking the balance before withdrawal.

2. **Objective:** Apply object-oriented programming concepts in a practical system.

3. **Input/Apparatus Used:** Java classes, objects, and control structures.

4. **Procedure:**
   Step1: Define a `BankAccount` class with fields like name, account number, and balance.
   Step2: Implement methods for:
   - deposit(double amount): Adds amount to balance.
   - withdraw(double amount): Checks balance before subtracting.
   Step3: In the main program, create a new account by taking user input.
   Step4: Allow the user to perform deposit and withdrawal operations.
   Step5: Display appropriate messages and updated balances.

   Sample Input:
   Create Account:
   Name: John Doe
   Account Number: 12345
   Initial Balance: 1000
   Deposit: 500
   Withdraw: 2000

   Sample Output:
   Deposit successful! Current Balance: 1500
   Error: Insufficient funds. Current Balance: 1500

5. Code:

```java
import java.util.Scanner;
                            /*HARD LEVEL*/
class BankAccount {  2 usages
    private String name;  2 usages
    private String accountNumber;  2 usages
    private double balance;  8 usages

    public BankAccount(String name, String accountNumber, double initialBalance) {  1 usage
        this.name = name;
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }

    public void deposit(double amount) {  1 usage
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposit successful! Current Balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

    public void withdraw(double amount) {  1 usage
        if (amount <= 0) {
            System.out.println("Invalid withdrawal amount.");
        } else if (amount > balance) {
            System.out.println("Error: Insufficient funds. Current Balance: " + balance);
        } else {
            balance -= amount;
            System.out.println("Withdrawal successful! Current Balance: " + balance);
        }
    }

    public void displayInfo() {  2 usages
        System.out.println("Account Holder: " + name);
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
    }
}
```

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

        System.out.println("=== Create New Account ===");
        System.out.print("Enter Name: ");
        String name = sc.nextLine();
        System.out.print("Enter Account Number: ");
        String accountNumber = sc.nextLine();
        System.out.print("Enter Initial Balance: ");
        double initialBalance = sc.nextDouble();

        BankAccount account = new BankAccount(name, accountNumber, initialBalance);
        account.displayInfo();

        int choice;
        do {
            System.out.println("\n=== Banking Menu ===");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Display Account Info");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmt = sc.nextDouble();
                    account.deposit(depositAmt);
                    break;
                case 2:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmt = sc.nextDouble();
                    account.withdraw(withdrawAmt);
                    break;
                case 3:
                    account.displayInfo();
                    break;
                case 4:
                    System.out.println("Exiting... Thank you!");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (choice != 4);
    }
}
```

6. Output:

```
=== Create New Account ===
Enter Name: GAGNESH
Enter Account Number: 96520
Enter Initial Balance: 600
Account Holder: GAGNESH
Account Number: 96520
Balance: 600.0
```

```
=== Banking Menu ===
1. Deposit
2. Withdraw
3. Display Account Info
4. Exit
Enter your choice: 1
Enter amount to deposit: 1500
Deposit successful! Current Balance: 2100.0
```

```
=== Banking Menu ===
1. Deposit
2. Withdraw
3. Display Account Info
4. Exit
Enter your choice: 2
Enter amount to withdraw: 650
Withdrawal successful! Current Balance: 1450.0
```

```
=== Banking Menu ===
1. Deposit
2. Withdraw
3. Display Account Info
4. Exit
Enter your choice: 3
Account Holder: GAGNESH
Account Number: 96520
Balance: 1450.0
```

```
=== Banking Menu ===
1. Deposit
2. Withdraw
3. Display Account Info
4. Exit
Enter your choice: 4
Exiting... Thank you!
```