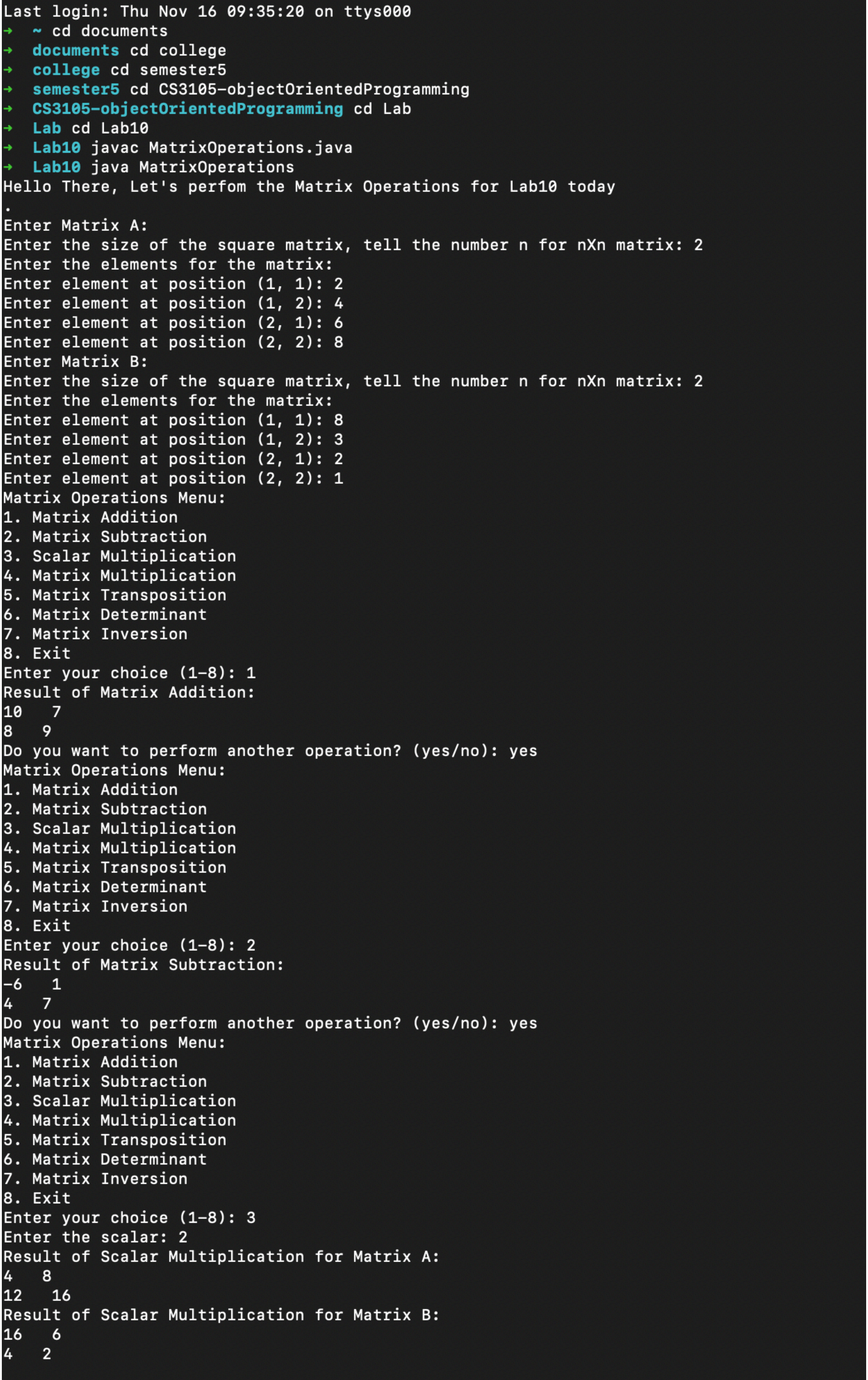
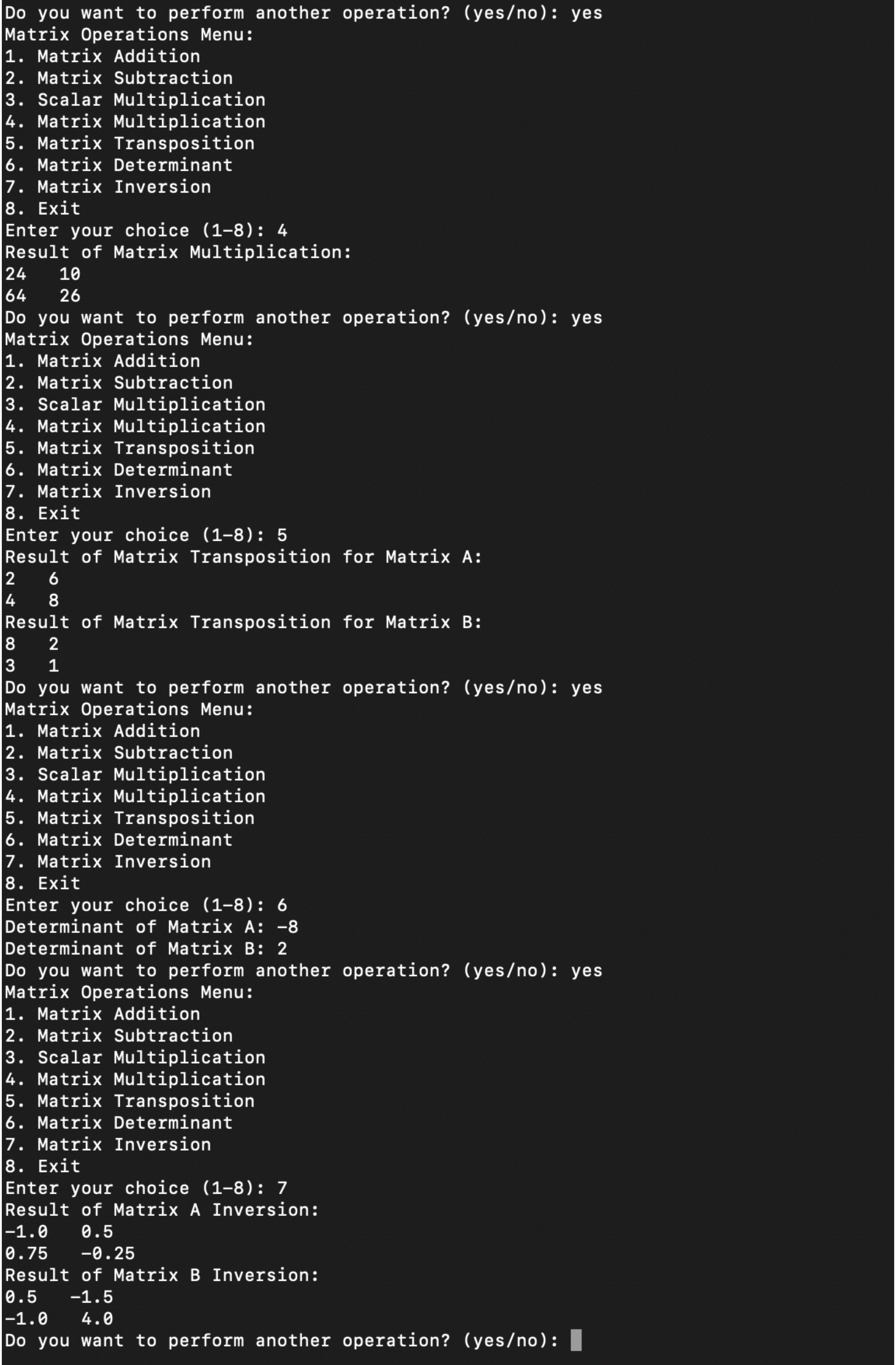


```
Last login: Thu Nov 16 09:35:20 on ttys000
➜  ~ cd documents
➜  documents cd college
➜  college cd semester5
➜  semester5 cd CS3105-objectOrientedProgramming
➜  CS3105-objectOrientedProgramming cd Lab
➜  Lab cd Lab10
➜  Lab10 javac MatrixOperations.java
➜  Lab10 java MatrixOperations
Hello There, Let's perfom the Matrix Operations for Lab10 today
.
Enter Matrix A:
Enter the size of the square matrix, tell the number n for nXn matrix: 2
Enter the elements for the matrix:
Enter element at position (1, 1): 2
Enter element at position (1, 2): 4
Enter element at position (2, 1): 6
Enter element at position (2, 2): 8
Enter Matrix B:
Enter the size of the square matrix, tell the number n for nXn matrix: 2
Enter the elements for the matrix:
Enter element at position (1, 1): 8
Enter element at position (1, 2): 3
Enter element at position (2, 1): 2
Enter element at position (2, 2): 1
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 1
Result of Matrix Addition:
10   7
8   9
Do you want to perform another operation? (yes/no): yes
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 2
Result of Matrix Subtraction:
-6   1
4   7
Do you want to perform another operation? (yes/no): yes
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 3
Enter the scalar: 2
Result of Scalar Multiplication for Matrix A:
4   8
12   16
Result of Scalar Multiplication for Matrix B:
16   6
4   2
```



```
Do you want to perform another operation? (yes/no): yes
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 4
Result of Matrix Multiplication:
24   10
64   26
Do you want to perform another operation? (yes/no): yes
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 5
Result of Matrix Transposition for Matrix A:
2   6
4   8
Result of Matrix Transposition for Matrix B:
8   2
3   1
Do you want to perform another operation? (yes/no): yes
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 6
Determinant of Matrix A: -8
Determinant of Matrix B: 2
Do you want to perform another operation? (yes/no): yes
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Scalar Multiplication
4. Matrix Multiplication
5. Matrix Transposition
6. Matrix Determinant
7. Matrix Inversion
8. Exit
Enter your choice (1-8): 7
Result of Matrix A Inversion:
-1.0   0.5
0.75   -0.25
Result of Matrix B Inversion:
0.5   -1.5
-1.0   4.0
Do you want to perform another operation? (yes/no):
```

```
/*  Bhuvana Kanakam
    SE21UCSE035 - Lab10
*/

import java.util.Scanner;
public class MatrixOperations {

    private static Scanner scanner = new Scanner(System.in);

    private static int[][] matrixA;
    private static int[][] matrixB;

    public static void main(String[] args) {
        boolean exit = false;
        System.out.println("Hello There, Let's perfom the Matrix 
Operations for Lab10 today\n.");

        System.out.println("Enter Matrix A:");
        matrixA = getUserMatrix();
        System.out.println("Enter Matrix B:");
        matrixB = getUserMatrix();

        while (!exit) {
            System.out.println("Matrix Operations Menu:");
            System.out.println("1. Matrix Addition");
            System.out.println("2. Matrix Subtraction");
            System.out.println("3. Scalar Multiplication");
            System.out.println("4. Matrix Multiplication");
            System.out.println("5. Matrix Transposition");
            System.out.println("6. Matrix Determinant");
            System.out.println("7. Matrix Inversion");
            System.out.println("8. Exit");

            System.out.print("Enter your choice (1-8): ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    performMatrixAddition();
                    break;
                case 2:
                    performMatrixSubtraction();
                    break;
                case 3:
                    performScalarMultiplication();
                    break;
                case 4:
                    performMatrixMultiplication();
                    break;
                case 5:
                    performMatrixTransposition();
                    break;
                case 6:
                    performMatrixDeterminant();
                    break;
                case 7:
                    performMatrixInversion();
                    break;
                case 8:
                    exit = true;
                    System.out.println("Exiting program. Goodbye!");
                    break;
                default:
                    System.out.println("Invalid choice. Please try
again.");
            }

            if (!exit) {
                System.out.print("Do you want to perform another
operation? (yes/no): ");
                String continueOption =
scanner.next().toLowerCase();

                if (!continueOption.equals("yes")) {
                    exit = true;
                    System.out.println("Exiting program. Goodbye!");
                }
            }
        }
    }
    private static int[][] getUserMatrix() {
        System.out.print("Enter the size of the square matrix, tell
the number n for nXn matrix: ");
        int size = scanner.nextInt();
        System.out.println("Enter the elements for the matrix:");

        int[][] matrix = new int[size][size];

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                System.out.print("Enter element at position (" + (i
+ 1) + ", " + (j + 1) + "): ");
                matrix[i][j] = scanner.nextInt();
            }
        }

        return matrix;
    }

    private static void performMatrixAddition() {
        int[][] result = addMatrices(matrixA, matrixB);
        System.out.println("Result of Matrix Addition:");
        printMatrix(result);
    }

    private static void performMatrixSubtraction() {
        int[][] result = subtractMatrices(matrixA, matrixB);
        System.out.println("Result of Matrix Subtraction:");
        printMatrix(result);
    }

    private static void performScalarMultiplication() {
        System.out.print("Enter the scalar: ");
        int scalar = scanner.nextInt();

        int[][] resultA = scalarMultiply(matrixA, scalar);
        int[][] resultB = scalarMultiply(matrixB, scalar);

        System.out.println("Result of Scalar Multiplication for
Matrix A:");
        printMatrix(resultA);

        System.out.println("Result of Scalar Multiplication for
Matrix B:");
        printMatrix(resultB);
    }

    private static void performMatrixMultiplication() {
        int[][] result = multiplyMatrices(matrixA, matrixB);
        System.out.println("Result of Matrix Multiplication:");
        printMatrix(result);
    }

    private static void performMatrixTransposition() {
    int[][] resultA = transposeMatrix(matrixA);
    int[][] resultB = transposeMatrix(matrixB);

    System.out.println("Result of Matrix Transposition for Matrix
A:");
    printMatrix(resultA);

    System.out.println("Result of Matrix Transposition for Matrix
B:");
    printMatrix(resultB);
    }

    private static void performMatrixDeterminant() {
        int determinantA = determinant(matrixA);
        int determinantB = determinant(matrixB);

        System.out.println("Determinant of Matrix A: " +
determinantA);
        System.out.println("Determinant of Matrix B: " +
determinantB);
    }

    private static void performMatrixInversion() {
        double[][] invertedMatrixA = invertMatrix(matrixA);
        double[][] invertedMatrixB = invertMatrix(matrixB);

        if (invertedMatrixA != null && invertedMatrixB != null) {
            System.out.println("Result of Matrix A Inversion:");
            printMatrix(invertedMatrixA);

            System.out.println("Result of Matrix B Inversion:");
            printMatrix(invertedMatrixB);
        } else {
        System.out.println("One or both matrices are not
invertible.");
        }
    }



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

    private static void printMatrix(double[][] matrix) {
        int n = matrix.length;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(   matrix[i][j] + "   ");
            }
            System.out.println();
        }
    }

    public static int[][] addMatrices(int[][] A, int[][] B) {
        int n = A.length;
        int[][] result = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                result[i][j] = A[i][j] + B[i][j];
            }
        }
        return result;
    }

    public static int[][] subtractMatrices(int[][] A, int[][] B) {
        int n = A.length;
        int[][] result = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                result[i][j] = A[i][j] - B[i][j];
            }
        }
        return result;
    }

    public static int[][] scalarMultiply(int[][] matrix, int scalar)
{
        int n = matrix.length;
        int[][] result = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                result[i][j] = matrix[i][j] * scalar;
            }
        }
        return result;
    }

    public static int[][] multiplyMatrices(int[][] A, int[][] B) {
        int n = A.length;
        int[][] result = new int[n][n];

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

    public static int[][] transposeMatrix(int[][] matrix) {
        int n = matrix.length;
        int[][] result = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                result[j][i] = matrix[i][j];
            }
        }
        return result;
    }

    public static int determinant(int[][] matrix) {
        int n = matrix.length;
        if (n == 1) {
            return matrix[0][0];
        }
        if (n == 2) {
            return matrix[0][0] * matrix[1][1] - matrix[0][1] *
matrix[1][0];
        }
        int det = 0;

        for (int i = 0; i < n; i++) {
            det += matrix[0][i] * cofactor(matrix, 0, i);
        }
        return det;
    }

    private static int cofactor(int[][] matrix, int row, int col) {
        return (int) Math.pow(-1, row + col) *
determinant(minor(matrix, row, col));
    }

    private static int[][] minor(int[][] matrix, int row, int col) {
        int n = matrix.length;
        int[][] minorMatrix = new int[n - 1][n - 1];

        for (int i = 0, k = 0; i < n; i++) {
            if (i == row) {
                continue;
            }
            for (int j = 0, l = 0; j < n; j++) {
                if (j == col) {
                    continue;
                }
                minorMatrix[k][l] = matrix[i][j];
                l++;
            }
            k++;
        }

        return minorMatrix;
    }

    public static double[][] invertMatrix(int[][] matrix) {
        int n = matrix.length;
        double[][] augmentedMatrix = new double[n][2 * n];

        // Create an augmented matrix [A | I], where I is the
identity matrix
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                augmentedMatrix[i][j] = matrix[i][j];
                augmentedMatrix[i][j + n] = (i == j) ? 1 : 0; //
Identity matrix
            }
        }

        // Apply Gauss-Jordan elimination to obtain the inverse
        for (int i = 0; i < n; i++) {
            double pivot = augmentedMatrix[i][i];

            if (pivot == 0) {
                return null; // Matrix is not invertible
            }

            // Scale the row to make the pivot equal to 1
            for (int j = 0; j < 2 * n; j++) {
                augmentedMatrix[i][j] /= pivot;
            }

            // Eliminate other rows
            for (int k = 0; k < n; k++) {
                if (k != i) {
                    double factor = augmentedMatrix[k][i];
                    for (int j = 0; j < 2 * n; j++) {
                        augmentedMatrix[k][j] -= factor *
augmentedMatrix[i][j];
                    }
                }
            }
        }

        // Extract the inverted matrix from the augmented matrix
        double[][] invertedMatrix = new double[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                invertedMatrix[i][j] = augmentedMatrix[i][j + n];
            }
        }

        return invertedMatrix;
    }

}
```