Encapsulation

```
diff --git a/Matrix.java b/Matrix.java
index 294d96c..7daf24d 100644
--- a/Matrix.java
+++ b/Matrix.java
@@ -8,17 +8,64 @@ import java.util.Scanner;
  */
 public class Matrix {

-    private static int[][] matrix1;
-
-    private static int[][] matrix2;
+    private int[][] matrix1;
+    private int[][] matrix2;

     int row1,row2,col1,col2;

-    private static final Scanner sc = new Scanner(System.in);
+    private final Scanner sc = new Scanner(System.in);

+    public int[][] getMatrix1() {
+        return matrix1;
+    }
+
+    public void setMatrix1(int[][] matrix1) {
+        this.matrix1 = matrix1;
+    }
+
+
+    public int[][] getMatrix2() {
+        return matrix2;
+    }
+
+    public void setMatrix2(int[][] matrix2) {
+        this.matrix2 = matrix2;
+    }
+
+    public int getRow1() {
+        return row1;
+    }
+
+    public void setRow1(int row1) {
+        this.row1 = row1;
+    }
+
+    public int getRow2() {
+        return row2;
+    }
+
+    public void setRow2(int row2) {
+        this.row2 = row2;
+    }
+
+    public int getCol1() {
+        return col1;
+    }
+
+    public void setCol1(int col1) {
+        this.col1 = col1;
+    }
+
+    public int getCol2() {
+        return col2;
+    }
+
+    public void setCol2(int col2) {
+        this.col2 = col2;
+    }

     //Integer validator
```

```
-    public static int checkInputInt(){
+    public int checkInputInt(){
         while(true){
             try {
                 int result = Integer.parseInt(sc.nextLine().trim());
@@ -29,7 +76,7 @@ public class Matrix {
         }
     }
     //Matrix row and col >0
-    private static int checkArrayElements(){
+    private int checkArrayElements(){
         while(true){
             try {
                 int result = Integer.parseInt(sc.nextLine().trim());
@@ -41,7 +88,7 @@ public class Matrix {
         }
     }

-    private static boolean checkPositive(int result){
+    private boolean checkPositive(int result){
         if (result>0) return true;
         return false;
     }
@@ -221,8 +268,6 @@ public class Matrix {
                 getMatrixForSubtraction();break;
             case 3:
                 getMatrixForMulltiplication();break;
-            case 4:
-               System.exit(0);
             default:
                 System.out.println("Invalid command! Please enter again: ");
         }
diff --git a/MatrixCalculator.java b/MatrixCalculator.java
index 3d9f8b8..4ad6303 100644
--- a/MatrixCalculator.java
+++ b/MatrixCalculator.java
@@ -15,7 +15,7 @@ public class MatrixCalculator {
             System.out.println("2.Subtraction Matrix.");
             System.out.println("3.Multiplication Matrix.");
             System.out.println("4.Exit.");
-            choice = Matrix.checkInputInt();
+            choice = mat.checkInputInt();
             mat.getOperation(choice);
         }
     }
```

```
        modified:   Matrix.java
        modified:   MatrixCalculator.java


U@LAPTOP-627MJLEP MINGW64 ~/git/src/matrixcalculator (master)
$ git commit -a -m "Extract Method"
[master 7c15519] Extract Method
 2 files changed, 124 insertions(+), 117 deletions(-)

U@LAPTOP-627MJLEP MINGW64 ~/git/src/matrixcalculator (master)
$ git show
commit 7c155190a641222b3a778f70b25398048b876403 (HEAD -> master)
Author: Uijin <uijin.park1@ucalgary.ca>
Date:   Fri Oct 6 02:39:47 2023 -0400

    Extract Method

diff --git a/Matrix.java b/Matrix.java
index 7daf24d..dbac54b 100644
--- a/Matrix.java
+++ b/Matrix.java
@@ -63,6 +63,25 @@ public class Matrix {
     public void setCol2(int col2) {
         this.col2 = col2;
     }
+
+    public enum MatrixOperation {
+        ADDITION,
+        SUBTRACTION,
+        MULTIPLICATION,
+        EXIT
+    }
+
+    private int[][] initializeMatrix(int rowCount, int colCount) {
+        int[][] matrix = new int[rowCount][colCount];
+        for (int i = 0; i < rowCount; i++) {
+            for (int j = 0; j < colCount; j++) {
+                System.out.print("Enter Matrix [" + (i + 1) + "][" + (j + 1) + "]:
");
+                matrix[i][j] = checkInputInt();
+            }
+        }
+        return matrix;
+    }
+

     //Integer validator
     public int checkInputInt(){
@@ -93,30 +112,30 @@ public class Matrix {
         return false;
     }

-
-    //Arithmetic operator is sent to String op
-    private void printMatrix(int[][] matrix3, int row, int col, String op){
-        System.out.println("----------------Result-----------------");
-        for (int i =0; i<row1;i++){
-            for (int j = 0; j < col1; j++){
-                System.out.print("["+matrix1[i][j]+"] ");
+    private void printSingleMatrix(int[][] matrix, int rows, int cols) {
+        for (int i = 0; i < rows; i++) {
+            for (int j = 0; j < cols; j++) {
+                System.out.print("[" + matrix[i][j] + "] ");
             }
             System.out.println();
         }
+    }
+
+    private void printMatrixHeader(String header) {
+        System.out.println(header);
+    }
+
+    private void printOperation(String op) {
         System.out.println(op);
-        for (int i =0; i<row2;i++){
-            for (int j = 0; j < col2; j++){
-                System.out.print("["+matrix2[i][j]+"] ");
-            }
-            System.out.println();
-        }
+    }
+
+    private void printMatrix(int[][] resultMatrix, int row, int col, String op) {
+        printMatrixHeader("-----------------Result-----------------");
+        printSingleMatrix(matrix1, row1, col1);
+        printOperation(op);
+        printSingleMatrix(matrix2, row2, col2);
         System.out.println("=");
-        for (int i =0; i<row;i++){
-            for (int j = 0; j < col; j++){
-                System.out.print("["+matrix3[i][j]+"] ");
-            }
-            System.out.println();
-        }
+        printSingleMatrix(resultMatrix, row, col);
     }

     //Multiplication of Matrix
@@ -153,6 +172,30 @@ public class Matrix {
         printMatrix(matrix3, row, col, "-");
     }

+    private int validateMatrixRowsForMultiplication(int col1) {
+        while(true){
+            System.out.print("Enter row matrix 2:");
+            int row2 = checkArrayElements();
+            if (row2 == col1) return row2;
+            System.out.println("Invalid Matrix for multiplication, column of matrix 1 
must equal row of matrix 2.");
+        }
+    }
+
+    private void validateMatrixForAddOrSub(int expectedRows, int expectedCols) {
+        while (true) {
+            System.out.print("Enter row matrix 2:");
+            row2 = checkArrayElements();
+            if (row2 == expectedRows) break;
+            System.out.println("Invalid Matrix, both matrices must have the same 
number of rows.");
+        }
+
+        while (true) {
+            System.out.print("Enter column matrix 2:");
+            col2 = checkArrayElements();
+            if (col2 == expectedCols) break;
+            System.out.println("Invalid Matrix, both matrices must have the same 
number of columns.");
+        }
+    }

     private void getMatrixForMulltiplication(){
         System.out.println("---------Multiplication-----------");
@@ -160,66 +203,24 @@ public class Matrix {
         row1 = checkArrayElements();
         System.out.print("Enter column matrix 1:");
         col1 = checkArrayElements();
-        matrix1 = new int[row1][col1];
-        for (int i =0; i< row1;i++){
-            for (int j=0; j<col1;j++){
-                System.out.print("Enter Matrix 1 [" + (i+1) + "][ "+ (j+1) + "]:");
-                matrix1[i][j]= checkInputInt();
-            }
-        }
-        boolean isMatrixValid=false;
-        while(!isMatrixValid){
-            System.out.print("Enter row matrix 2:");
-            row2 = checkArrayElements();
-            if (row2!=col1) System.out.println("Invalid Matrix for multiplication,
column of matrix 1 must equal row of matrix 2.");
-            else isMatrixValid=true;
-        }
+        matrix1 = initializeMatrix(row1, col1); //Extract Method
+        row2 = validateMatrixRowsForMultiplication(col1); //Extract Method
         System.out.print("Enter column matrix 2:");
         col2 = checkArrayElements();
-        matrix2= new int[row2][col2];
-        for (int i =0; i< row2;i++){
-            for (int j=0; j<col2;j++){
-                System.out.print("Enter Matrix 2 [" + (i+1) + "][ "+ (j+1) + "]:");
-                matrix2[i][j]= checkInputInt();
-            }
-        }
+        matrix2 = initializeMatrix(row2, col2); //Extract Method
         mulMatrix(matrix1, matrix2);
     }

+
     private void getMatrixForAddition(){
         System.out.println("-------------Addition------------");
         System.out.print("Enter row matrix 1:");
         row1 = checkArrayElements();
         System.out.print("Enter column matrix 1:");
         col1 = checkArrayElements();
-        matrix1 = new int[row1][col1];
-        for (int i =0; i< row1;i++){
-            for (int j=0; j<col1;j++){
-                System.out.print("Enter Matrix 1 [" + (i+1) + "][ "+ (j+1) + "]:");
-                matrix1[i][j]= checkInputInt();
-            }
-        }
-        boolean isRowValid=false;
-        boolean isColValid=false;
-        while (!isRowValid){
-            System.out.print("Enter row matrix 2:");
-            row2=checkArrayElements();
-            if (row1==row2) isRowValid=true;
-            else System.out.println("Invalid Matrix for addition, both matrix must
have the same number of row.");
-        }
-        while (!isColValid){
-            System.out.print("Enter column matrix 2:");
-            col2=checkArrayElements();
-            if (col2==col1) isColValid=true;
-            else System.out.println("Invalid Matrix for addition, both matrix must
have the same number of column.");
-        }
-        matrix2=new int[row2][col2];
-        for (int i =0; i< row2;i++){
-            for (int j=0; j<col2;j++){
-                System.out.print("Enter Matrix 2 [" + (i+1) + "][ "+ (j+1) + "]:");
-                matrix2[i][j]= checkInputInt();
-            }
-        }
+        matrix1 = initializeMatrix(row1, col1); //Extract Method
+        validateMatrixForAddOrSub(row1, col1);
+        matrix2 = initializeMatrix(row2, col2); //Extract Method
         addMatrix(matrix1, matrix2, row1, col1);
     }

@@ -229,45 +230,26 @@ public class Matrix {
         row1 = checkArrayElements();
         System.out.print("Enter column matrix 1:");
         col1 = checkArrayElements();
-        matrix1 = new int[row1][col1];
-        for (int i =0; i< row1;i++){
-            for (int j=0; j<col1;j++){
```

```
-                   System.out.print("Enter Matrix 1 [" + (i+1) + "][ "+ (j+1) + "]:");
-                   matrix1[i][j]= checkInputInt();
-               }
-           }
-           boolean isRowValid=false;
-           boolean isColValid=false;
-           while (!isRowValid){
-               System.out.print("Enter row matrix 2:");
-               row2=checkArrayElements();
-               if (row1==row2) isRowValid=true;
-               else System.out.println("Invalid Matrix for subtraction, both matrix must
have the same number of row.");
-           }
-           while (!isColValid){
-               System.out.print("Enter column matrix 2:");
-               col2=checkArrayElements();
-               if (col2==col1) isColValid=true;
-               else System.out.println("Invalid Matrix for subtraction, both matrix must
have the same number of column.");
-           }
-           matrix2=new int[row2][col2];
-           for (int i =0; i< row2;i++){
-               for (int j=0; j<col2;j++){
-                   System.out.print("Enter Matrix 2 [" + (i+1) + "][ "+ (j+1) + "]:");
-                   matrix2[i][j]= checkInputInt();
-               }
-           }
+           matrix1 = initializeMatrix(row1, col1); //Extract Method
+           validateMatrixForAddOrSub(row1, col1);
+           matrix2 = initializeMatrix(row2, col2); //Extract Method
            subMatrix(matrix1, matrix2, row1, col1);
       }

-      public void getOperation(int n){
-          switch(n){
-              case 1:
-                  getMatrixForAddition();break;
-              case 2:
-                  getMatrixForSubtraction();break;
-              case 3:
-                  getMatrixForMulltiplication();break;
+      public void getOperation(MatrixOperation operation){
+          switch(operation){
+              case ADDITION:
+                  getMatrixForAddition();
+                  break;
+              case SUBTRACTION:
+                  getMatrixForSubtraction();
+                  break;
+              case MULTIPLICATION:
+                  getMatrixForMulltiplication();
+                  break;
+              case EXIT:
+                  System.exit(0);
+                  break;
               default:
                   System.out.println("Invalid command! Please enter again: ");
           }
diff --git a/MatrixCalculator.java b/MatrixCalculator.java
index 4ad6303..b98ef30 100644
--- a/MatrixCalculator.java
+++ b/MatrixCalculator.java
@@ -1,24 +1,49 @@
 package matrixcalculator;

+import matrixcalculator.Matrix.MatrixOperation;
+
 /**
  *
  * @author son75
  */
+
 public class MatrixCalculator {

-    public void menu(){
-        int choice =0;
-        Matrix mat = new Matrix();
-        while (choice!=4){
-            System.out.println("===========Calculator Program=============");
-            System.out.println("1.Addition Matrix.");
-            System.out.println("2.Subtraction Matrix.");
-            System.out.println("3.Multiplication Matrix.");
-            System.out.println("4.Exit.");
-            choice = mat.checkInputInt();
-            mat.getOperation(choice);
-        }
-    }
+       public void menu(){
+           MatrixOperation choice = null;
+           Matrix mat = new Matrix();
+           do {
+               System.out.println("===========Calculator Program=============");
+               System.out.println("1. Addition Matrix.");
+               System.out.println("2. Subtraction Matrix.");
+               System.out.println("3. Multiplication Matrix.");
+               System.out.println("4. Exit.");
+
+               int input = mat.checkInputInt();
+
+               switch(input) {
+                   case 1:
+                       choice = MatrixOperation.ADDITION;
+                       break;
+                   case 2:
+                       choice = MatrixOperation.SUBTRACTION;
+                       break;
+                   case 3:
+                       choice = MatrixOperation.MULTIPLICATION;
+                       break;
+                   case 4:
+                       choice = MatrixOperation.EXIT;
+                       break;
+                   default:
+                       System.out.println("Invalid command! Please enter again: ");
+                       continue;
+               }
+
+               mat.getOperation(choice);
+
+           } while (choice != MatrixOperation.EXIT);
+       }
+
     public static void main(String[] args) {
         // TODO code application logic here
         MatrixCalculator mc = new MatrixCalculator();
```

```
commit d22b99102e935994ce62da3353f84944f560b3a3 (HEAD -> master)
Author: Uijin <uijin.park1@ucalgary.ca>
Date:   Fri Oct 6 05:59:35 2023 -0400

    Testing and Extract

diff --git a/Matrix.java b/Matrix.java
index dbac54b..63bff3b 100644
--- a/Matrix.java
+++ b/Matrix.java
@@ -140,36 +140,19 @@ public class Matrix {
 
     //Multiplication of Matrix
     private void mulMatrix(int[][] matrix1, int[][] matrix2){
-        int[][] matrix3 = new int[row1][col2];
-         for (int i =0; i<row1;i++){
-            for (int j = 0; j < col2; j++){
-                for (int k = 0; k<row2;k++){
-                    matrix3[i][j] += matrix1[i][k] * matrix2[k][j];
-                }
-            }
-        }
-        printMatrix(matrix3, row1, col2, "*");
+       int[][] resultMatrix = MatrixOperatorCalculation.multiply(matrix1, matrix2, 
row1, col1, row2, col2);
+        printMatrix(resultMatrix, row1, col2, "*");
     }
     //Addition of Matrix
     private void addMatrix(int[][] matrix1,int[][] matrix2, int row,int col){
-         int[][] matrix3 = new int[row][col];
-         for (int i =0; i<row;i++){
-             for (int j = 0; j < col; j++){
-                 matrix3[i][j]=matrix1[i][j]+matrix2[i][j];
-             }
-         }
-        printMatrix(matrix3, row, col, "+");
+       int[][] resultMatrix = MatrixOperatorCalculation.add(matrix1, matrix2, row1, 
col1);
+        printMatrix(resultMatrix, row, col, "+");
     }
 
     //Subtraction of Matrix
     private void subMatrix(int[][] matrix1,int[][] matrix2, int row,int col){
-         int[][] matrix3 = new int[row][col];
-         for (int i =0; i<row;i++){
-             for (int j = 0; j < col; j++){
-                 matrix3[i][j]=matrix1[i][j]-matrix2[i][j];
-             }
-         }
-        printMatrix(matrix3, row, col, "-");
+       int[][] resultMatrix = MatrixOperatorCalculation.subtract(matrix1, matrix2, 
row1, col1);
+        printMatrix(resultMatrix, row, col, "-");
     }
 
     private int validateMatrixRowsForMultiplication(int col1) {
@@ -199,10 +182,8 @@ public class Matrix {
 
     private void getMatrixForMulltiplication(){
         System.out.println("---------Multiplication-----------");
-        System.out.print("Enter row matrix 1:");
-        row1 = checkArrayElements();
-        System.out.print("Enter column matrix 1:");
-        col1 = checkArrayElements();
```

```
+        row1 = getMatrixDimension("Enter row matrix 1: ");
+        col1 = getMatrixDimension("Enter column matrix 1: ");
         matrix1 = initializeMatrix(row1, col1); //Extract Method
         row2 = validateMatrixRowsForMultiplication(col1); //Extract Method
         System.out.print("Enter column matrix 2:");
@@ -211,13 +192,15 @@ public class Matrix {
         mulMatrix(matrix1, matrix2);
     }

+    private int getMatrixDimension(String message) {
+        System.out.print(message);
+        return checkArrayElements();
+    }

     private void getMatrixForAddition(){
         System.out.println("-------------Addition------------");
-        System.out.print("Enter row matrix 1:");
-        row1 = checkArrayElements();
-        System.out.print("Enter column matrix 1:");
-        col1 = checkArrayElements();
+        row1 = getMatrixDimension("Enter row matrix 1: ");
+        col1 = getMatrixDimension("Enter column matrix 1: ");
         matrix1 = initializeMatrix(row1, col1); //Extract Method
         validateMatrixForAddOrSub(row1, col1);
         matrix2 = initializeMatrix(row2, col2); //Extract Method
@@ -226,10 +209,8 @@ public class Matrix {

     private void getMatrixForSubtraction(){
         System.out.println("-------------Addition------------");
-        System.out.print("Enter row matrix 1:");
-        row1 = checkArrayElements();
-        System.out.print("Enter column matrix 1:");
-        col1 = checkArrayElements();
+        row1 = getMatrixDimension("Enter row matrix 1: ");
+        col1 = getMatrixDimension("Enter column matrix 1: ");
         matrix1 = initializeMatrix(row1, col1); //Extract Method
         validateMatrixForAddOrSub(row1, col1);
         matrix2 = initializeMatrix(row2, col2); //Extract Method
diff --git a/MatrixOperatorCalculation.java b/MatrixOperatorCalculation.java
new file mode 100644
index 0000000..60bf9df
--- /dev/null
+++ b/MatrixOperatorCalculation.java
@@ -0,0 +1,40 @@
+package matrixcalculator;
+
+public class MatrixOperatorCalculation {
+
+       public static int[][] multiply(int[][] matrix1, int[][] matrix2, int row1, int
col1, int row2, int col2) {
+        int[][] result = new int[row1][col2];
+        for (int i = 0; i < row1; i++) {
+            for (int j = 0; j < col2; j++) {
+                for (int k = 0; k < row2; k++) {
+                    result[i][j] += matrix1[i][k] * matrix2[k][j];
+                }
+            }
+        }
+        return result;
+    }
+
+       public static int[][] add(int[][] matrix1, int[][] matrix2, int row, int col)
{
+               int[][] result = new int[row][col];
+        for (int i = 0; i < row; i++) {
+            for (int j = 0; j < col; j++) {
+                result[i][j] = matrix1[i][j] + matrix2[i][j];
+            }
+        }
+        return result;
+       }
+
+       public static int[][] subtract(int[][] matrix1, int[][] matrix2, int row, int
col) {
+        int[][] result = new int[row][col];
+        for (int i = 0; i < row; i++) {
+            for (int j = 0; j < col; j++) {
+                result[i][j] = matrix1[i][j] - matrix2[i][j];
+            }
+        }
+        return result;
+    }
+}
+
+
+
+
diff --git a/MatrixTest.java b/MatrixTest.java
new file mode 100644
index 0000000..fda4825
--- /dev/null
+++ b/MatrixTest.java
@@ -0,0 +1,71 @@
+package matrixcalculator;
+
+import org.junit.Test;
+import static org.junit.Assert.*;
+
+public class MatrixTest {
+
+
+    @Test
+    public void testAddMatrix() {
+        int[][] matrix1 = {
+            {1, 2},
+            {3, 4}
+        };
+
+        int[][] matrix2 = {
+            {2, 3},
+            {4, 5}
+        };
+
+        int[][] expected = {
+            {3, 5},
+            {7, 9}
+        };
+
+        int[][] result = MatrixOperatorCalculation.add(matrix1, matrix2, 2, 2);
+        assertArrayEquals(expected, result);
+    }
+
+    @Test
+    public void testSubtractMatrix() {
+        int[][] matrix1 = {
+            {5, 6},
+            {7, 8}
+        };
+
+        int[][] matrix2 = {
+            {2, 3},
+            {4, 5}
+        };
+
+        int[][] expected = {
+            {3, 3},
+            {3, 3}
+        };
+
+        int[][] result = MatrixOperatorCalculation.subtract(matrix1, matrix2, 2, 2);
+        assertArrayEquals(expected, result);
+    }
+
+    @Test
+    public void testMultiplyMatrix() {
+        int[][] matrix1 = {
+            {1, 2},
+            {3, 4}
+        };
+
+        int[][] matrix2 = {
+            {2, 0},
+            {1, 3}
+        };
+
+        int[][] expected = {
+            {4, 6},
+            {10, 12}
+        };
+
+        int[][] result = MatrixOperatorCalculation.multiply(matrix1, matrix2, 2, 2,
2, 2);
+        assertArrayEquals(expected, result);
+    }
+}
```

    Refactoring

```
diff --git a/Matrix.java b/Matrix.java
index 63bff3b..505b891 100644
--- a/Matrix.java
+++ b/Matrix.java
@@ -1,19 +1,24 @@
 package matrixcalculator;

-import java.util.Scanner;
-
 /**
  *
  * @author son75
  */
 public class Matrix {

+       MatrixHandleInput handleInput;
+
+       public Matrix() {
+           this.handleInput = new MatrixHandleInput();
+       }
+
+       private MatrixPrint handlePrint = new MatrixPrint();
+
     private int[][] matrix1;
     private int[][] matrix2;

     int row1,row2,col1,col2;

-    private final Scanner sc = new Scanner(System.in);

     public int[][] getMatrix1() {
         return matrix1;
@@ -70,166 +75,72 @@ public class Matrix {
         MULTIPLICATION,
         EXIT
     }
-
-    private int[][] initializeMatrix(int rowCount, int colCount) {
-        int[][] matrix = new int[rowCount][colCount];
-        for (int i = 0; i < rowCount; i++) {
-            for (int j = 0; j < colCount; j++) {
-                System.out.print("Enter Matrix [" + (i + 1) + "][" + (j + 1) + "]: 
");
-                matrix[i][j] = checkInputInt();
-            }
-        }
-        return matrix;
-    }
-
-
-    //Integer validator
-    public int checkInputInt(){
-        while(true){
-            try {
-                int result = Integer.parseInt(sc.nextLine().trim());
-                return result;
-            } catch (NumberFormatException e) {
-                System.out.println("Integer number not found");
-            }
-        }
-    }
-    //Matrix row and col >0
-    private int checkArrayElements(){
-        while(true){
-            try {
-                int result = Integer.parseInt(sc.nextLine().trim());
-                if (checkPositive(result)==true) return result;
-                else System.out.println("The number of row/column must be higher than 
0, please enter again: ");
-            } catch (NumberFormatException e) {
-                System.out.println("Integer not found, please enter again: ");
-            }
-        }
-    }
-
-    private boolean checkPositive(int result){
-        if (result>0) return true;
-        return false;
-    }
-
-    private void printSingleMatrix(int[][] matrix, int rows, int cols) {
-        for (int i = 0; i < rows; i++) {
-            for (int j = 0; j < cols; j++) {
-                System.out.print("[" + matrix[i][j] + "] ");
-            }
-            System.out.println();
-        }
-    }
-
-    private void printMatrixHeader(String header) {
-        System.out.println(header);
-    }
-
-    private void printOperation(String op) {
-        System.out.println(op);
-    }
-
-    private void printMatrix(int[][] resultMatrix, int row, int col, String op) {
-        printMatrixHeader("-----------------Result-----------------");
-        printSingleMatrix(matrix1, row1, col1);
-        printOperation(op);
-        printSingleMatrix(matrix2, row2, col2);
-        System.out.println("=");
-        printSingleMatrix(resultMatrix, row, col);
-    }
-
+
     //Multiplication of Matrix
-    private void mulMatrix(int[][] matrix1, int[][] matrix2){
-       int[][] resultMatrix = MatrixOperatorCalculation.multiply(matrix1, matrix2,
row1, col1, row2, col2);
-        printMatrix(resultMatrix, row1, col2, "*");
+    private void multiplyMatrix(int[][] matrix1, int[][] matrix2){
+       int[][] resultMatrix = MatrixOperatorCalculation.multiply(getMatrix1(),
getMatrix2(), getRow1(), getCol1(), getRow2(), getCol2());
+       handlePrint.printMatrix(getMatrix1(), getRow1(), getCol1(), getMatrix2(),
getRow2(), getCol2(), resultMatrix, getRow1(), getCol2(), "*");
     }
     //Addition of Matrix
-    private void addMatrix(int[][] matrix1,int[][] matrix2, int row,int col){
-       int[][] resultMatrix = MatrixOperatorCalculation.add(matrix1, matrix2, row1,
col1);
-        printMatrix(resultMatrix, row, col, "+");
+    private void addMatrix(int[][] matrix1,int[][] matrix2, int resultRow,int
resultCol){
+       int[][] resultMatrix = MatrixOperatorCalculation.add(getMatrix1(),
getMatrix2(), getRow1(), getCol1());
+       handlePrint.printMatrix(getMatrix1(), getRow1(), getCol1(), getMatrix2(),
getRow2(), getCol2(), resultMatrix, resultRow, resultCol, "+");
     }

     //Subtraction of Matrix
-    private void subMatrix(int[][] matrix1,int[][] matrix2, int row,int col){
-       int[][] resultMatrix = MatrixOperatorCalculation.subtract(matrix1, matrix2,
row1, col1);
-        printMatrix(resultMatrix, row, col, "-");
+    private void subtractMatrix(int[][] matrix1,int[][] matrix2, int resultRow,int
resultCol){
+       int[][] resultMatrix = MatrixOperatorCalculation.subtract(getMatrix1(),
getMatrix2(), getRow1(), getCol1());
+       handlePrint.printMatrix(getMatrix1(), getRow1(), getCol1(), getMatrix2(),
getRow2(), getCol2(), resultMatrix, resultRow, resultCol, "-");
     }

-    private int validateMatrixRowsForMultiplication(int col1) {
-        while(true){
-            System.out.print("Enter row matrix 2:");
-            int row2 = checkArrayElements();
-            if (row2 == col1) return row2;
-            System.out.println("Invalid Matrix for multiplication, column of matrix 1
must equal row of matrix 2.");
-        }
-    }

-    private void validateMatrixForAddOrSub(int expectedRows, int expectedCols) {
-        while (true) {
-            System.out.print("Enter row matrix 2:");
-            row2 = checkArrayElements();
-            if (row2 == expectedRows) break;
-            System.out.println("Invalid Matrix, both matrices must have the same
number of rows.");
-        }
-
-        while (true) {
-            System.out.print("Enter column matrix 2:");
-            col2 = checkArrayElements();
-            if (col2 == expectedCols) break;
-            System.out.println("Invalid Matrix, both matrices must have the same
number of columns.");
-        }
+    private int getMatrixDimension(String message) {
+        System.out.print(message);
+        return handleInput.checkArrayElements();
     }
```

```
-    private void getMatrixForMulltiplication(){
+    private void performMatrixForMulltiplication(){
         System.out.println("----------Multiplication-----------");
-        row1 = getMatrixDimension("Enter row matrix 1: ");
-        col1 = getMatrixDimension("Enter column matrix 1: ");
-        matrix1 = initializeMatrix(row1, col1); //Extract Method
-        row2 = validateMatrixRowsForMultiplication(col1); //Extract Method
+        setRow1(getMatrixDimension("Enter row matrix 1: "));
+        setCol1(getMatrixDimension("Enter column matrix 1: "));
+        setMatrix1(handleInput.initializeMatrix(getRow1(), getCol1())); //Extract
Method
+        setRow2(handleInput.validateMatrixRowsForMultiplication(getCol1()));
//Extract Method
         System.out.print("Enter column matrix 2:");
-        col2 = checkArrayElements();
-        matrix2 = initializeMatrix(row2, col2); //Extract Method
-        mulMatrix(matrix1, matrix2);
-    }
-
-    private int getMatrixDimension(String message) {
-        System.out.print(message);
-        return checkArrayElements();
+        setCol2(handleInput.checkArrayElements());
+        setMatrix2(handleInput.initializeMatrix(getRow2(), getCol2())); //Extract
Method
+        multiplyMatrix(getMatrix1(), getMatrix2());
     }

-    private void getMatrixForAddition(){
+    private void performMatrixForAddition(){
         System.out.println("-------------Addition------------");
-        row1 = getMatrixDimension("Enter row matrix 1: ");
-        col1 = getMatrixDimension("Enter column matrix 1: ");
-        matrix1 = initializeMatrix(row1, col1); //Extract Method
-        validateMatrixForAddOrSub(row1, col1);
-        matrix2 = initializeMatrix(row2, col2); //Extract Method
-        addMatrix(matrix1, matrix2, row1, col1);
+        setRow1(getMatrixDimension("Enter row matrix 1: "));
+        setCol1(getMatrixDimension("Enter column matrix 1: "));
+        setMatrix1(handleInput.initializeMatrix(getRow1(), getCol1())); //Extract
Method
+        handleInput.validateMatrixForAddOrSub(getRow1(), getCol1());
+        setMatrix2(handleInput.initializeMatrix(getRow2(), getCol2())); //Extract
Method
+        addMatrix(getMatrix1(), getMatrix2(), getRow1(), getCol1());
     }

-    private void getMatrixForSubtraction(){
-        System.out.println("-------------Addition------------");
-        row1 = getMatrixDimension("Enter row matrix 1: ");
-        col1 = getMatrixDimension("Enter column matrix 1: ");
-        matrix1 = initializeMatrix(row1, col1); //Extract Method
-        validateMatrixForAddOrSub(row1, col1);
-        matrix2 = initializeMatrix(row2, col2); //Extract Method
-        subMatrix(matrix1, matrix2, row1, col1);
+    private void performMatrixForSubtraction(){
+        System.out.println("-------------Subtraction------------");
+        setRow1(getMatrixDimension("Enter row matrix 1: "));
+        setCol1(getMatrixDimension("Enter column matrix 1: "));
+        setMatrix1(handleInput.initializeMatrix(getRow1(), getCol1())); //Extract
Method
+        handleInput.validateMatrixForAddOrSub(getRow1(), getCol1());
+        setMatrix2(handleInput.initializeMatrix(getRow2(), getCol2())); //Extract
Method
+        subtractMatrix(getMatrix1(), getMatrix2(), getRow1(), getCol1());
     }

     public void getOperation(MatrixOperation operation){
         switch(operation){
             case ADDITION:
-                getMatrixForAddition();
+                performMatrixForAddition();
```

```
                 break;
             case SUBTRACTION:
-                getMatrixForSubtraction();
+                performMatrixForSubtraction();
                 break;
             case MULTIPLICATION:
-                getMatrixForMulltiplication();
-                break;
-            case EXIT:
-                System.exit(0);
+                performMatrixForMulltiplication();
                 break;
             default:
                 System.out.println("Invalid command! Please enter again: ");
diff --git a/MatrixTest.java b/MatrixCalculationTest.java
similarity index 97%
rename from MatrixTest.java
rename to MatrixCalculationTest.java
index fda4825..ccbf507 100644
--- a/MatrixTest.java
+++ b/MatrixCalculationTest.java
@@ -3,7 +3,7 @@ package matrixcalculator;
 import org.junit.Test;
 import static org.junit.Assert.*;

-public class MatrixTest {
+public class MatrixCalculationTest {


     @Test
diff --git a/MatrixCalculator.java b/MatrixCalculator.java
index b98ef30..b65df43 100644
--- a/MatrixCalculator.java
+++ b/MatrixCalculator.java
@@ -8,18 +8,21 @@ import matrixcalculator.Matrix.MatrixOperation;
  */

 public class MatrixCalculator {
+
+       public void printChoice() {
+               System.out.println("===========Calculator Program=============");
+        System.out.println("1. Addition Matrix.");
+        System.out.println("2. Subtraction Matrix.");
+        System.out.println("3. Multiplication Matrix.");
+        System.out.println("4. Exit.");
+       }

        public void menu(){
            MatrixOperation choice = null;
            Matrix mat = new Matrix();
            do {
-               System.out.println("===========Calculator Program=============");
-               System.out.println("1. Addition Matrix.");
-               System.out.println("2. Subtraction Matrix.");
-               System.out.println("3. Multiplication Matrix.");
-               System.out.println("4. Exit.");
-
-               int input = mat.checkInputInt();
+               printChoice();
+               int input = mat.handleInput.checkInputInt();

                switch(input) {
                    case 1:
diff --git a/MatrixEncapsulationTest.java b/MatrixEncapsulationTest.java
new file mode 100644
index 0000000..1f6279e
--- /dev/null
+++ b/MatrixEncapsulationTest.java
@@ -0,0 +1,63 @@
+package matrixcalculator;
+
+import org.junit.Before;
```

```
+import org.junit.Test;
+import static org.junit.Assert.*;
+
+public class MatrixEncapsulationTest {
+
+       private Matrix matrix;
+
+       @Before
+    public void setUp() {
+        matrix = new Matrix();
+    }
+
+       @Test
+    public void testSetGetMatrix1() {
+        int[][] TestMatrix = {
+            {14, 25},
+            {63, 64}
+        };
+        matrix.setMatrix1(TestMatrix);
+        assertArrayEquals(TestMatrix, matrix.getMatrix1());
+    }
+
+    @Test
+    public void testSetGetMatrix2() {
+        int[][] TestMatrix = {
+            {52, 16},
+            {72, 85}
+        };
+        matrix.setMatrix2(TestMatrix);
+        assertArrayEquals(TestMatrix, matrix.getMatrix2());
+    }
+
+    @Test
+    public void testSetGetRow1() {
+        int TestValue = 576;
+        matrix.setRow1(TestValue);
+        assertEquals(TestValue, matrix.getRow1());
+    }
+
+    @Test
+    public void testSetGetRow2() {
+        int TestValue = 635;
+        matrix.setRow2(TestValue);
+        assertEquals(TestValue, matrix.getRow2());
+    }
+
+    @Test
+    public void testSetGetCol1() {
+        int TestValue = 157;
+        matrix.setCol1(TestValue);
+        assertEquals(TestValue, matrix.getCol1());
+    }
+
+    @Test
+    public void testSetGetCol2() {
+        int TestValue = 827;
+        matrix.setCol2(TestValue);
+        assertEquals(TestValue, matrix.getCol2());
+    }
+}
\ No newline at end of file
diff --git a/MatrixHandleInput.java b/MatrixHandleInput.java
new file mode 100644
index 0000000..96a7bdc
--- /dev/null
+++ b/MatrixHandleInput.java
@@ -0,0 +1,74 @@
+package matrixcalculator;
+
+import java.util.Scanner;
+
+public class MatrixHandleInput {
```

```
+
+    private final Scanner sc = new Scanner(System.in);
+
+       public int[][] initializeMatrix(int rowCount, int colCount) {
+        int[][] matrix = new int[rowCount][colCount];
+        for (int i = 0; i < rowCount; i++) {
+            for (int j = 0; j < colCount; j++) {
+                System.out.print("Enter Matrix [" + (i + 1) + "][" + (j + 1) + "]:
");
+                matrix[i][j] = checkInputInt();
+            }
+        }
+        return matrix;
+    }
+
+       //Integer validator
+    public int checkInputInt(){
+        while(true){
+            try {
+                int result = Integer.parseInt(sc.nextLine().trim());
+                return result;
+            } catch (NumberFormatException e) {
+                System.out.println("Integer number not found");
+            }
+        }
+    }
+  //Matrix row and col >0
+    public int checkArrayElements(){
+        while(true){
+            try {
+                int result = Integer.parseInt(sc.nextLine().trim());
+                if (isPositive(result)==true) return result;
+                else System.out.println("The number of row/column must be higher than
0, please enter again: ");
+            } catch (NumberFormatException e) {
+                System.out.println("Integer not found, please enter again: ");
+            }
+        }
+    }
+
+    private boolean isPositive(int result){
+       return result > 0;
+    }
+
+    public int validateMatrixRowsForMultiplication(int col1) {
+        while(true){
+            System.out.print("Enter row matrix 2:");
+            int row2 = checkArrayElements();
+            if (row2 == col1) return row2;
+            System.out.println("Invalid Matrix for multiplication, column of matrix 1
must equal row of matrix 2.");
+        }
+    }
+
+    public void validateMatrixForAddOrSub(int expectedRows, int expectedCols) {
+       int row2, col2;
+        while (true) {
+            System.out.print("Enter row matrix 2:");
+            row2 = checkArrayElements();
+            if (row2 == expectedRows) break;
+            System.out.println("Invalid Matrix, both matrices must have the same
number of rows.");
+        }
+
+        while (true) {
+            System.out.print("Enter column matrix 2:");
+            col2 = checkArrayElements();
+            if (col2 == expectedCols) break;
+            System.out.println("Invalid Matrix, both matrices must have the same
number of columns.");
+        }
+    }
```

```
+
+}
diff --git a/MatrixPrint.java b/MatrixPrint.java
new file mode 100644
index 0000000..c75c4da
--- /dev/null
+++ b/MatrixPrint.java
@@ -0,0 +1,31 @@
+package matrixcalculator;
+
+public class MatrixPrint {
+       public void printSingleMatrix(int[][] matrix, int rows, int cols) {
+        for (int i = 0; i < rows; i++) {
+            for (int j = 0; j < cols; j++) {
+                System.out.print("[" + matrix[i][j] + "] ");
+            }
+            System.out.println();
+        }
+    }
+
+       public void printMatrixHeader(String header) {
+        System.out.println(header);
+    }
+
+    public void printOperation(String op) {
+        System.out.println(op);
+    }
+
+    public void printMatrix(int[][] matrix1, int rows1, int cols1,
+                                       int[][] matrix2, int rows2, int cols2,
+                                       int[][] resultMatrix, int resultRows, int 
resultCols, String op) {
+        printMatrixHeader("----------------Result-----------------");
+        printSingleMatrix(matrix1, rows1, cols1);
+        printOperation(op);
+        printSingleMatrix(matrix2, rows2, cols2);
+        System.out.println("=");
+        printSingleMatrix(resultMatrix, resultRows, resultCols);
+    }
+}
```