矩阵大作业说明

物联网 181

刘恒星

185626

目录

一、类的设计

1. 类的说明

1.1 Matrix 类：普通矩阵

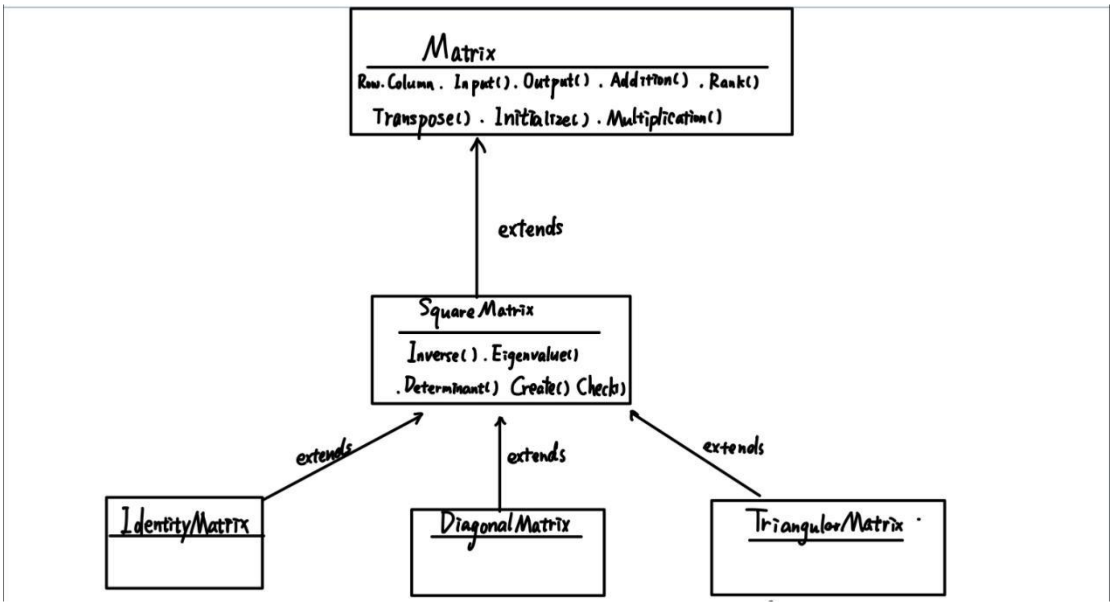
1.2 SquareMatrix 类：方阵

1.3 IdentityMatrix 类：单位矩阵

1.4 DiagonalMatrix 类：对角矩阵

1.5 TriangularMatrix 类：上/下三角矩阵

2. 类之间的关系


Matrix
Row. Column . Input() . Output() . Addition() . Rank()
Transpose() . Initialize() . Multiplication()
extends
SquareMatrix
Inverse() . Eigenvalue()
. Determinant() Create() Check()
extends
extends
extends
IdentityMatrix
DiagonalMatrix
TriangularMatrix

二、类的成员函数以及属性说明

1. 类的属性说明

1.1 Matrix 类

类名	Matrix	
属性名	属性类型	属性含义
Row	int	矩阵的行数
Column	int	矩阵的列数
matrix	double[][]	矩阵的数据

2. 类的成员函数说明

2.1 Matrix 类

类名	Matrix		
成员函数名	返回值	参数	作用
Matrix	null	null	无参数构造函数
Matrix	null	int Row, int Column	带参构造函数，初始化矩阵行列
Matrix	null	Matrix p	带参构造函数，根据参数初始化矩阵所有数据
Input	void	null	输入矩阵数据
Output	void	null	输出矩阵数据
Transpose	Matrix	null	返回当前矩阵的转置矩阵
Addition	Matrix	Matrix p	与参数矩阵进行矩阵加法，并把结果以矩阵的形式返回
Multiplication	Matrix	Matrix p	与参数矩阵进行矩阵乘法，并把结果以矩阵的形式返回
Inverse	Matrix	null	若该矩阵是方阵，则创建方阵的对象，调用方阵中的此函数，反之则输出错误信息并返回 null
Determeinant	double	null	若该矩阵是方阵，则创建方阵的对象，调用方阵中的此函数，反之则输出错误信息并返回 null
Rank	int	null	求该矩阵的秩
Solve	double[]	null	求用该矩阵表达的线性方程组的解

2.2 SquareMatrix 类

类名	SquareMatrix		
成员函数名	返回值	参数	作用
SquareMatrix	null	null	无参数构造函数
SquareMatrix	null	int Row	带参构造函数，初始化方阵行列
SquareMatrix	null	Matrix a	带参构造函数，根据参数初始化方阵所有数据
CheckDiagonalMatrix	boolean	null	检测此矩阵是否是对角矩阵
CheckTriangleMatrix	int	null	检测此矩阵是否是三角矩阵
Create	SquareMatrix	int i, int j, SquareMatrix M	对于矩阵 M 返回一个以 M 为基础的第 i 行和第 j 列的余子式
Determeinant	double	null	计算方阵的行列式的值
Inverse	Matrix	null	计算方阵的逆矩阵

2.3 IdentityMatrix 类

类名	IdentityMatrix		
成员函数名	返回值	参数	作用
IdentityMatrix	null	null	无参数构造函数
IdentityMatrix	null	int Row	带参构造函数，初始化矩阵行列
IdentityMatrix	null	Matrix a	带参构造函数，根据参数初始化矩阵所有数据
Determeinant	double	null	计算方阵的行列式的值

2.4 DiagonalMatrix 类

类名	DiagonalMatrix		
成员函数名	返回值	参数	作用
DiagonalMatrix	null	null	无参数构造函数
DiagonalMatrix	null	int Row	带参构造函数，初始化矩阵行列
DiagonalMatrix	null	Matrix a	带参构造函数，根据参数初始化矩阵所有数据
Determeinant	double	null	计算方阵的行列式的值
CheckDiagonalMatrix	boolean	null	检测矩阵时候是对角矩阵

2.5 TriangularMatrix 类

类名	TriangularMatrix		
成员函数名	返回值	参数	作用
TriangularMatrix	null	null	无参数构造函数
TriangularMatrix	null	int Row	带参构造函数，初始化矩阵行列
TriangularMatrix	null	Matrix a	带参构造函数，根据参数初始化矩阵所有数据
Determeinant	double	null	计算方阵的行列式的值

三、接口说明

1. 接口名 Calculate

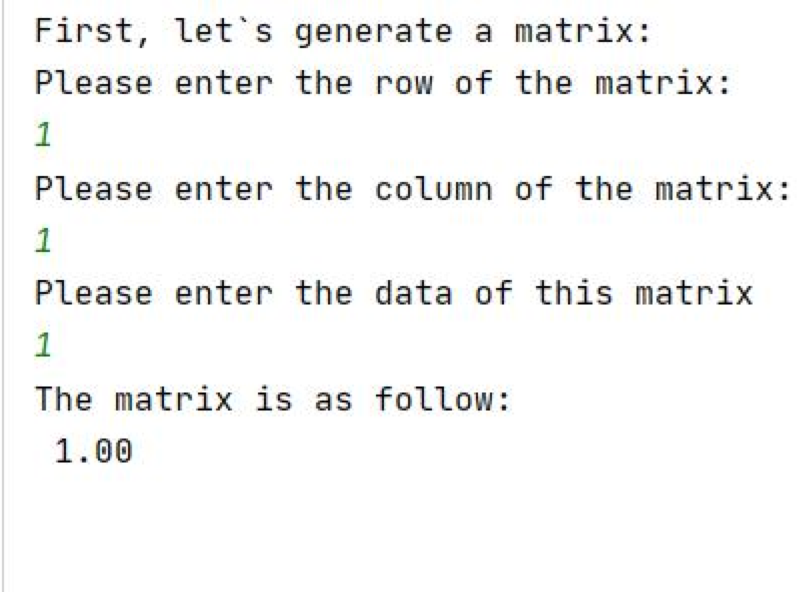
2. 接口详情

接口名	Calculate		
函数名	返回值	参数	作用
Addition	Matrix	Matrix p	矩阵加法的接口
Multiplication	Matrix	Matrix p	矩阵乘法的接口
Inverse	Matrix	null	求矩阵的接口
Determeinant	double	null	求行列式的接口

四、功能说明

1. 初始化矩阵

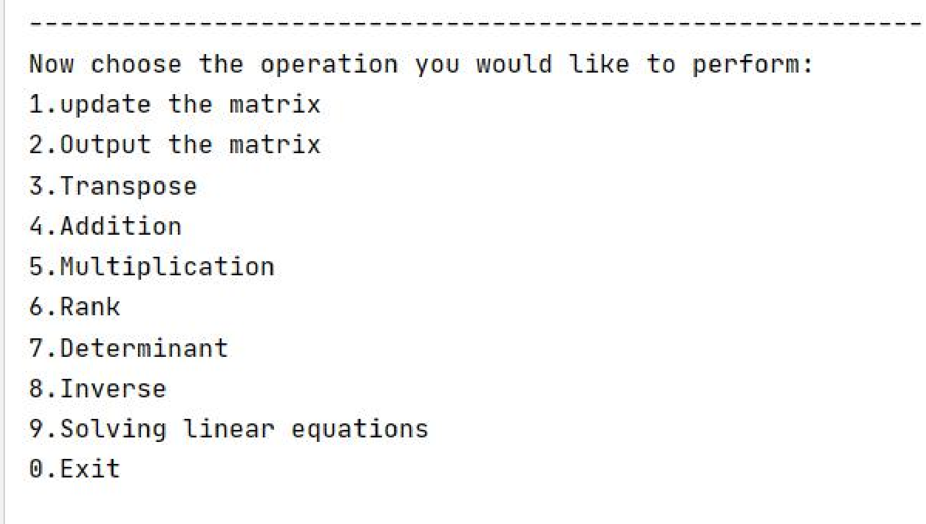
按照显示的信息初始化一个主矩阵，此后的计算大多围绕这个主矩阵进行



```
First, let`s generate a matrix:
Please enter the row of the matrix:
1
Please enter the column of the matrix:
1
Please enter the data of this matrix
1
The matrix is as follow:
 1.00
```

2. 菜单界面

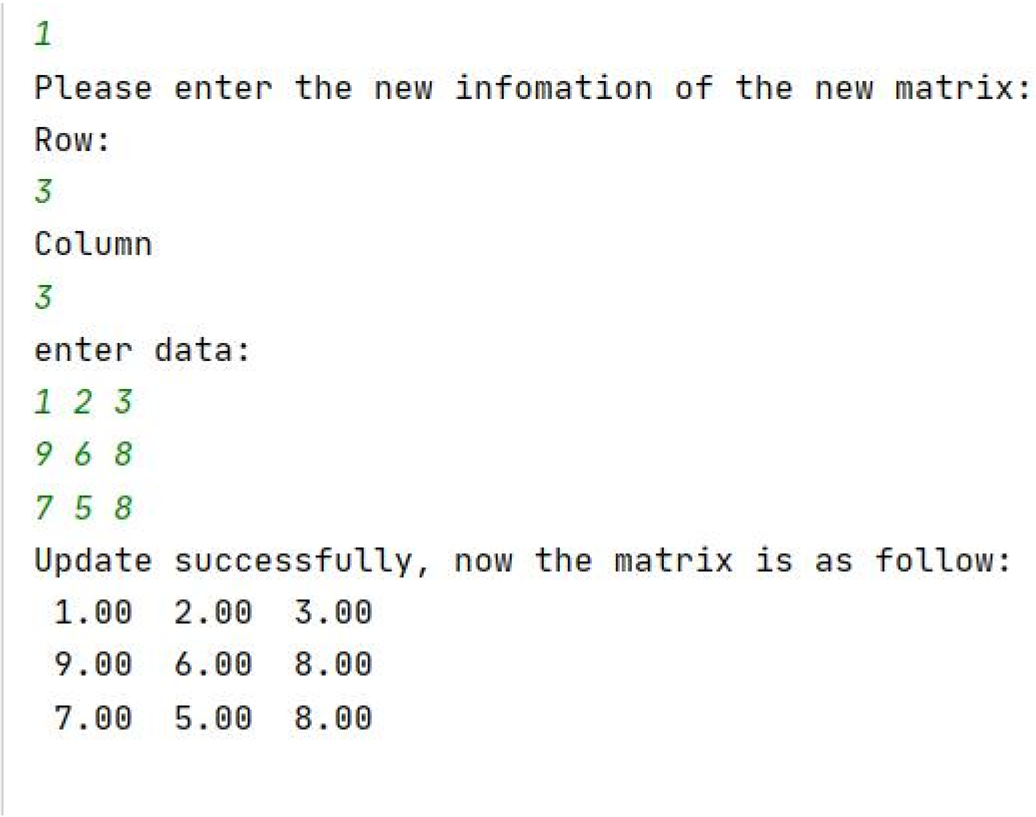
显示功能菜单



```
----------------------------------------------------------
Now choose the operation you would like to perform:
1.update the matrix
2.Output the matrix
3.Transpose
4.Addition
5.Multiplication
6.Rank
7.Determinant
8.Inverse
9.Solving linear equations
0.Exit
```

3. 更新矩阵

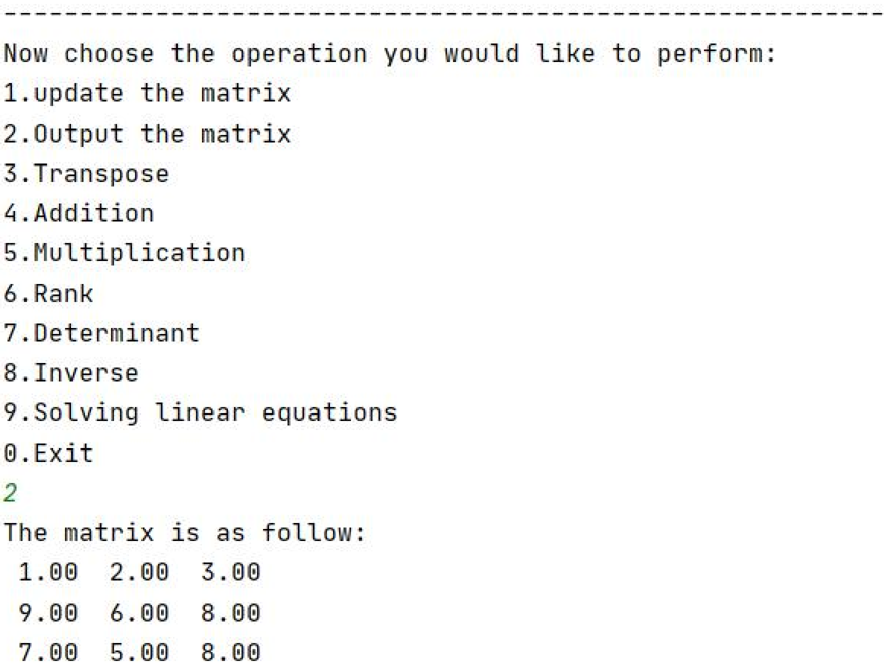
对主矩阵信息进行更新修改并且显示更新后的矩阵



```
1
Please enter the new infomation of the new matrix:
Row:
3
Column
3
enter data:
1 2 3
9 6 8
7 5 8
Update successfully, now the matrix is as follow:
 1.00  2.00  3.00
 9.00  6.00  8.00
 7.00  5.00  8.00
```

4. 输出矩阵

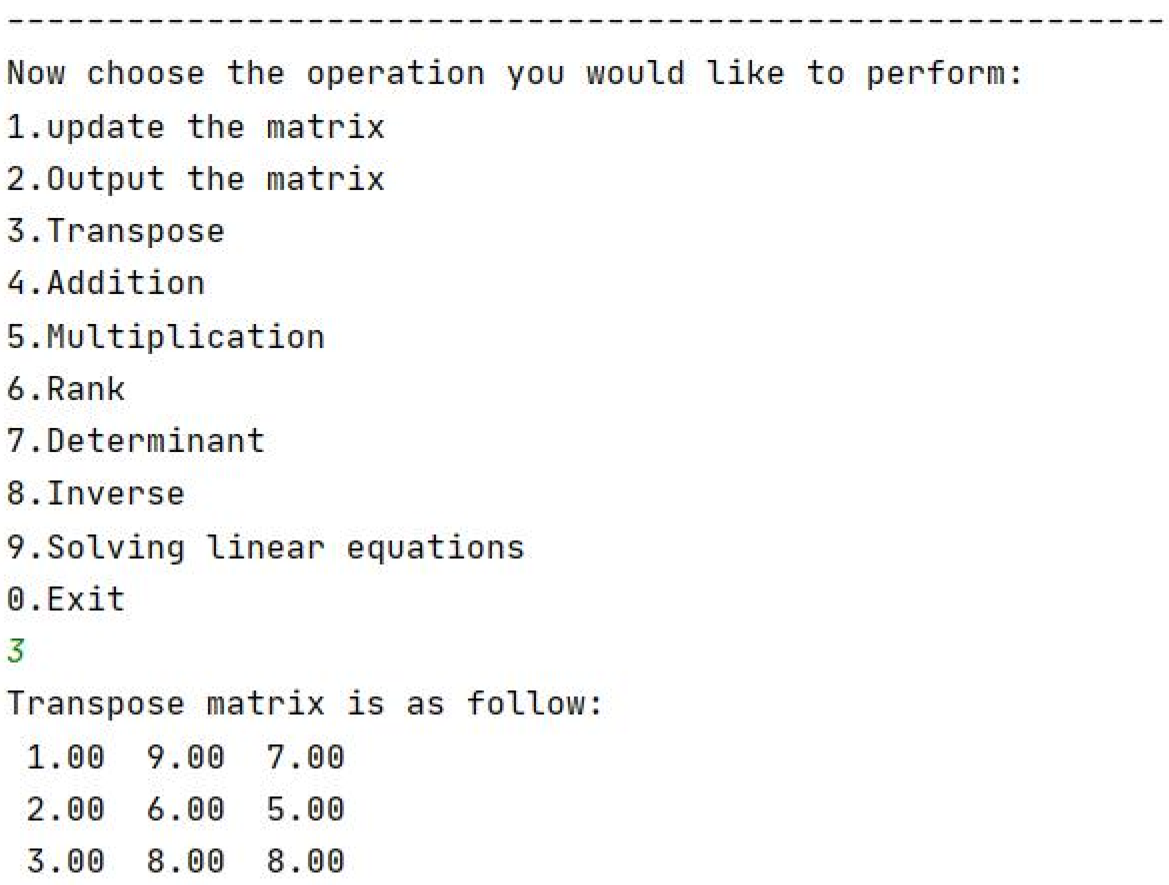
输出当前的矩阵



```
----------------------------------------------------------
Now choose the operation you would like to perform:
1.update the matrix
2.Output the matrix
3.Transpose
4.Addition
5.Multiplication
6.Rank
7.Determinant
8.Inverse
9.Solving linear equations
0.Exit
2
The matrix is as follow:
 1.00  2.00  3.00
 9.00  6.00  8.00
 7.00  5.00  8.00
```

5. 矩阵转置

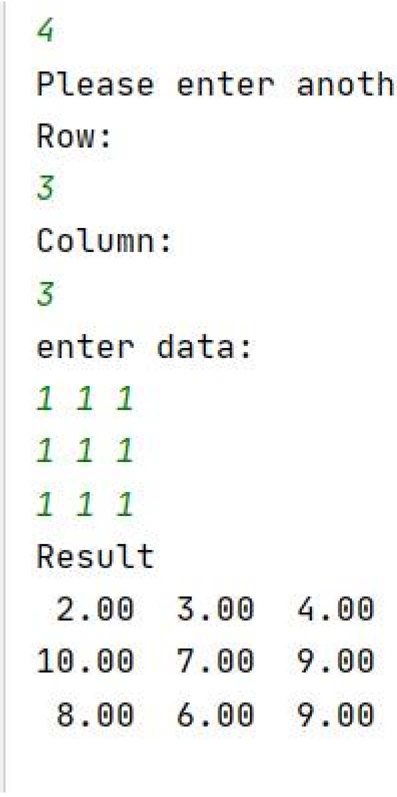
对矩阵做转置操作，并显示转置矩阵



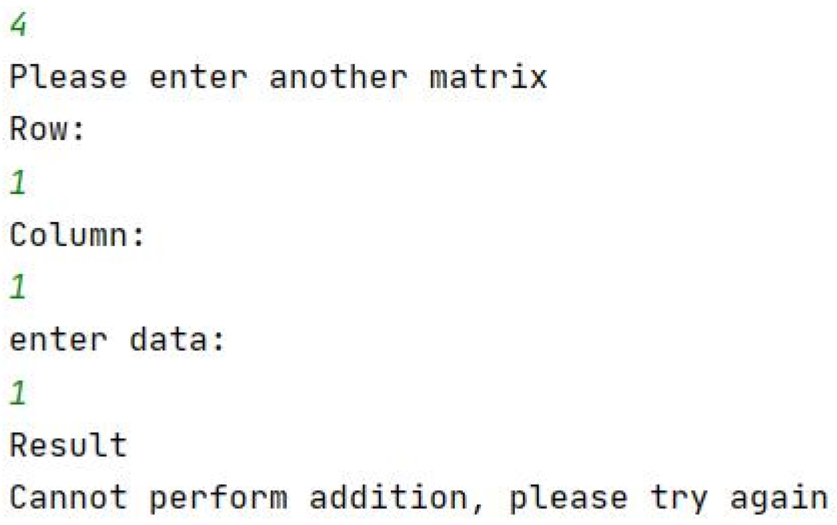
```
-----------------------------------------------------------
Now choose the operation you would like to perform:
1.update the matrix
2.Output the matrix
3.Transpose
4.Addition
5.Multiplication
6.Rank
7.Determinant
8.Inverse
9.Solving linear equations
0.Exit
3
Transpose matrix is as follow:
 1.00   9.00   7.00
 2.00   6.00   5.00
 3.00   8.00   8.00
```

6. 矩阵加法

进行矩阵的加法，如果数据合法就输出结果，不合法就输出错误信息



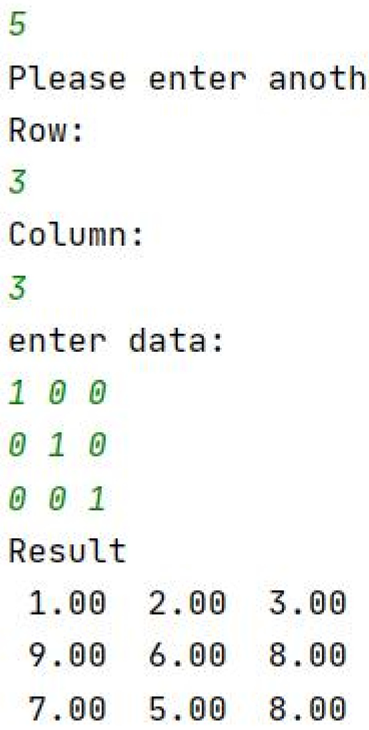
```
4
Please enter another matrix
Row:
3
Column:
3
enter data:
1 1 1
1 1 1
1 1 1
Result
 2.00   3.00   4.00
10.00   7.00   9.00
 8.00   6.00   9.00
```



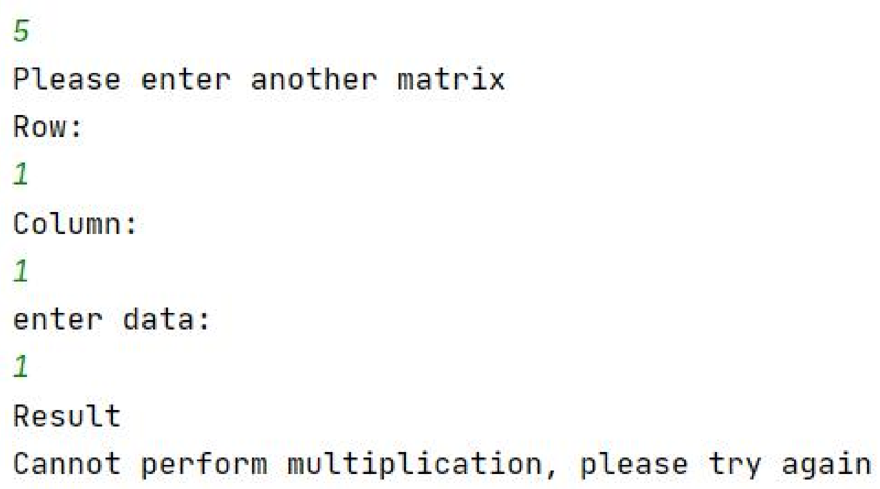
```
4
Please enter another matrix
Row:
1
Column:
1
enter data:
1
Result
Cannot perform addition, please try again
```

7. 矩阵乘法

进行矩阵的乘法，如果数据合法就输出结果，不合法就输出错误信息



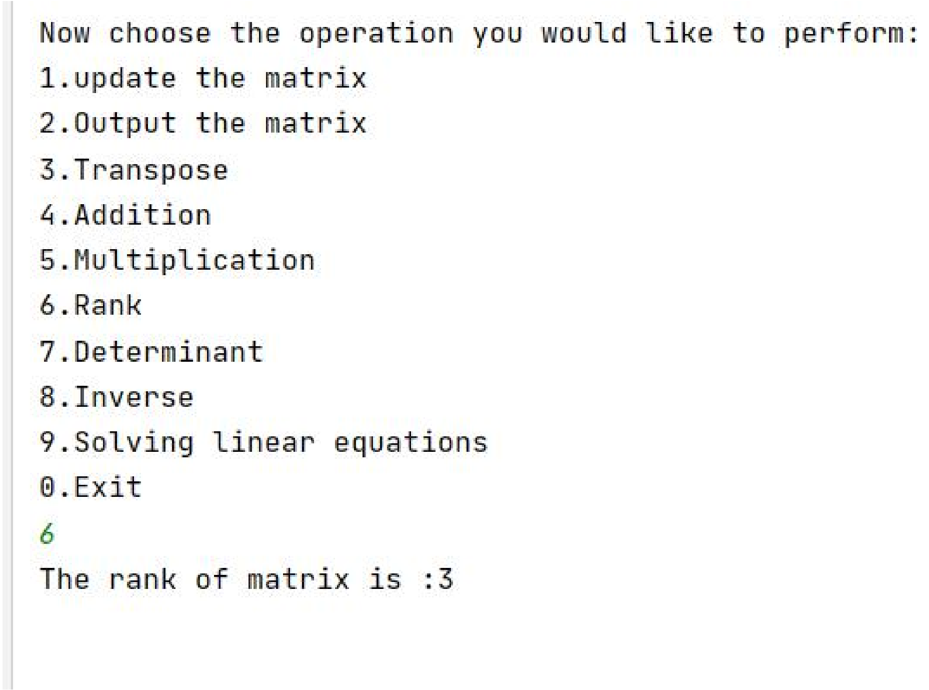
```
5
Please enter another matrix
Row:
3
Column:
3
enter data:
1 0 0
0 1 0
0 0 1
Result
 1.00  2.00  3.00
 9.00  6.00  8.00
 7.00  5.00  8.00
```



```
5
Please enter another matrix
Row:
1
Column:
1
enter data:
1
Result
Cannot perform multiplication, please try again
```

8. 矩阵的秩

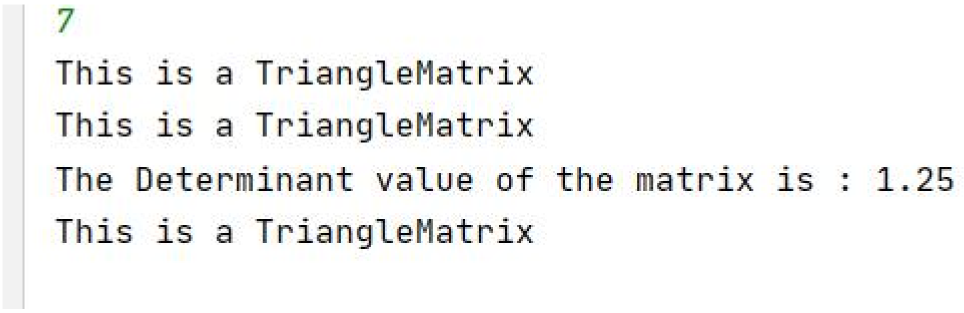
计算矩阵的秩并输出



```
Now choose the operation you would like to perform:
1.update the matrix
2.Output the matrix
3.Transpose
4.Addition
5.Multiplication
6.Rank
7.Determinant
8.Inverse
9.Solving linear equations
0.Exit
6
The rank of matrix is :3
```

9. 矩阵的行列式的值

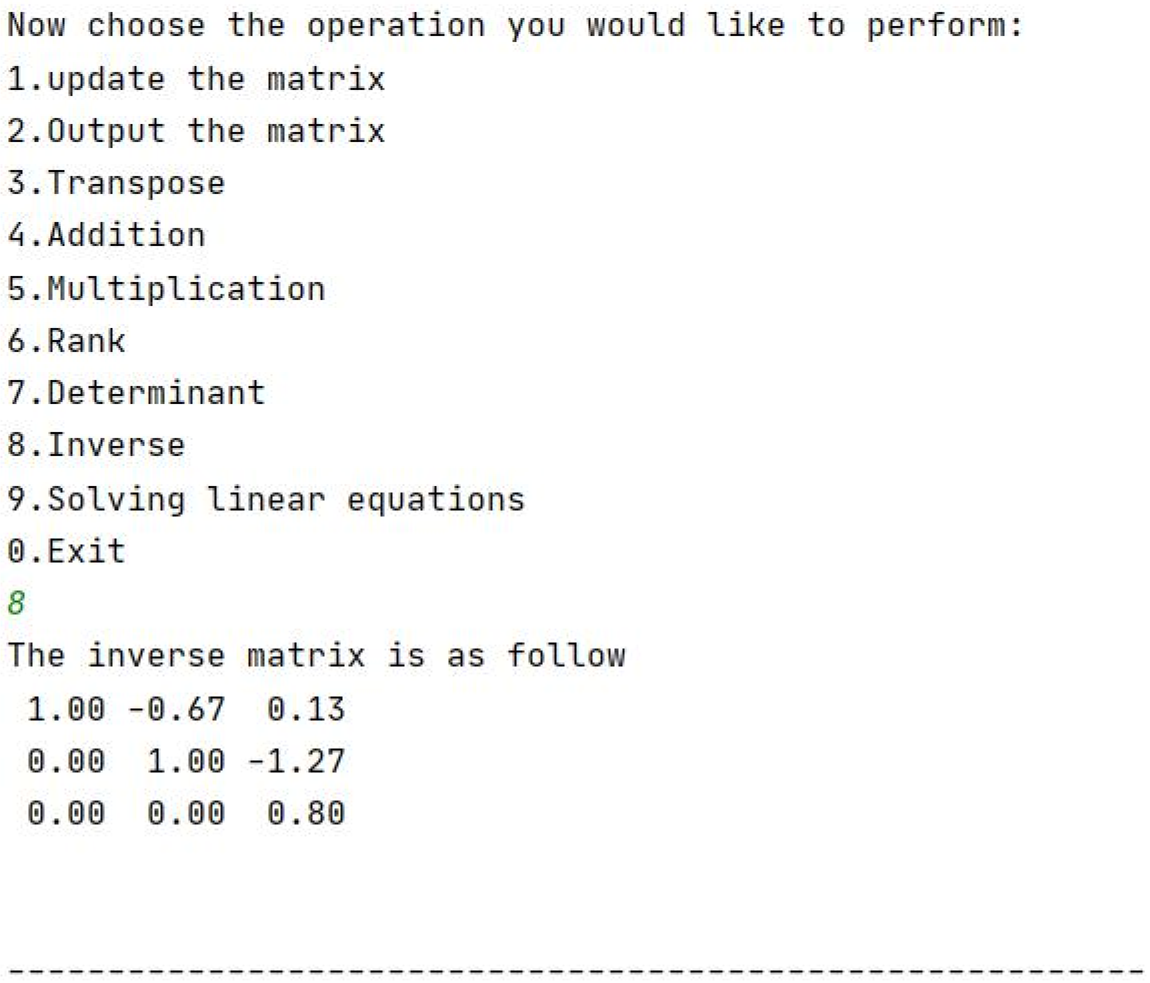
计算行列式的值并输出，中间还会输出余子式是否是特殊矩阵



```
7
This is a TriangleMatrix
This is a TriangleMatrix
The Determinant value of the matrix is : 1.25
This is a TriangleMatrix
```

10. 矩阵的逆

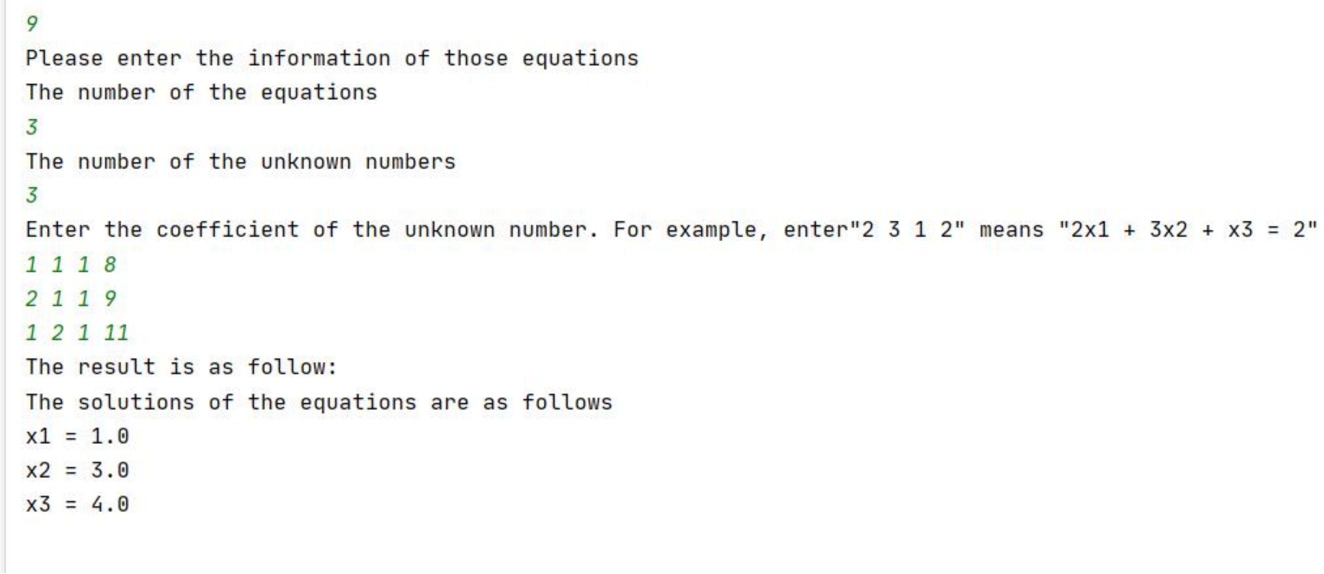
计算矩阵的逆并输出



```
Now choose the operation you would like to perform:
1.update the matrix
2.Output the matrix
3.Transpose
4.Addition
5.Multiplication
6.Rank
7.Determinant
8.Inverse
9.Solving linear equations
0.Exit
8
The inverse matrix is as follow
 1.00 -0.67  0.13
 0.00  1.00 -1.27
 0.00  0.00  0.80


----------------------------------------------------------
```

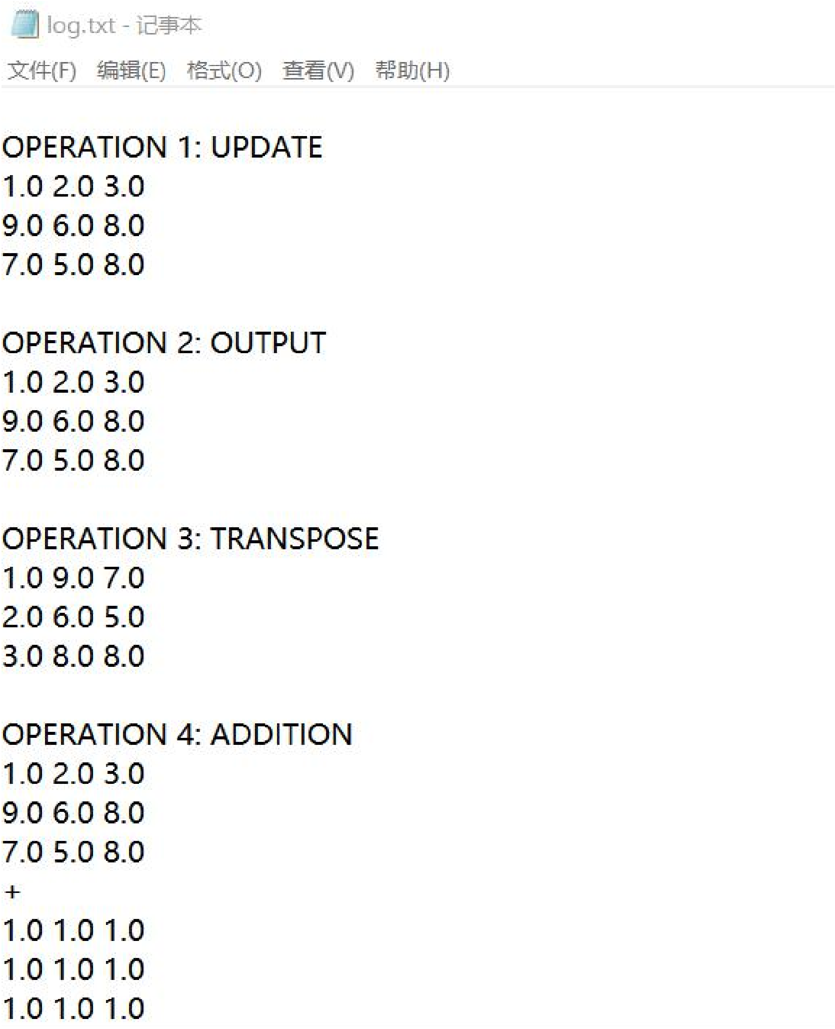
11. 求解线性方程组



```
9
Please enter the information of those equations
The number of the equations
3
The number of the unknown numbers
3
Enter the coefficient of the unknown number. For example, enter"2 3 1 2" means "2x1 + 3x2 + x3 = 2"
1 1 1 8
2 1 1 9
1 2 1 11
The result is as follow:
The solutions of the equations are as follows
x1 = 1.0
x2 = 3.0
x3 = 4.0
```

12. 文件读写

所有的操作记录和相关结果会存储在源程序下的 log.txt 文件中

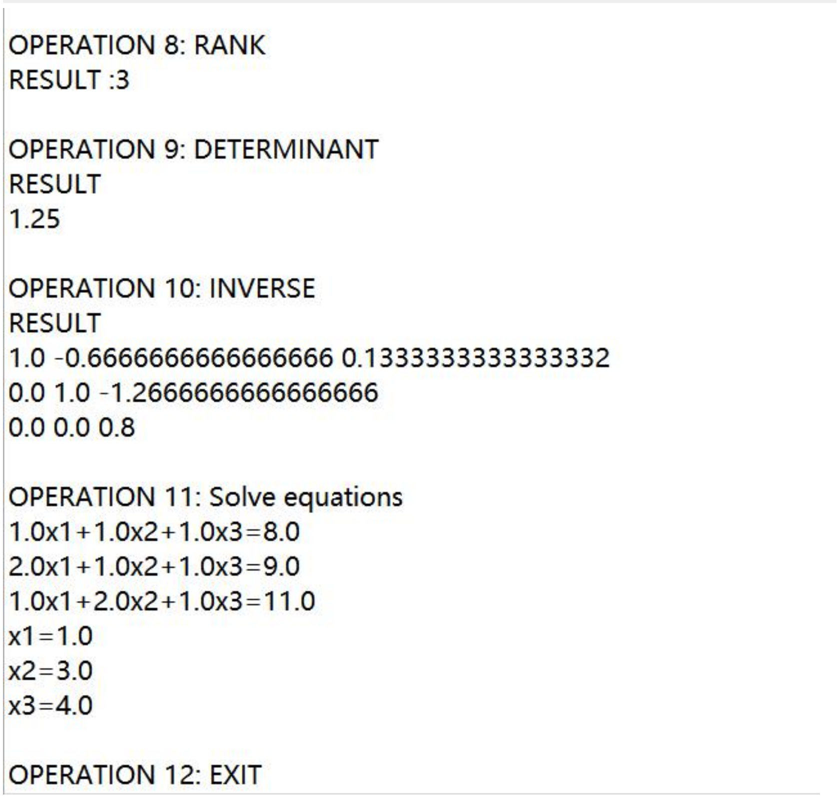


```
OPERATION 1: UPDATE
1.0 2.0 3.0
9.0 6.0 8.0
7.0 5.0 8.0

OPERATION 2: OUTPUT
1.0 2.0 3.0
9.0 6.0 8.0
7.0 5.0 8.0

OPERATION 3: TRANSPOSE
1.0 9.0 7.0
2.0 6.0 5.0
3.0 8.0 8.0

OPERATION 4: ADDITION
1.0 2.0 3.0
9.0 6.0 8.0
7.0 5.0 8.0
+
1.0 1.0 1.0
1.0 1.0 1.0
1.0 1.0 1.0
```



```
OPERATION 8: RANK
RESULT :3

OPERATION 9: DETERMINANT
RESULT
1.25

OPERATION 10: INVERSE
RESULT
1.0 -0.6666666666666666 0.1333333333333332
0.0 1.0 -1.2666666666666666
0.0 0.0 0.8

OPERATION 11: Solve equations
1.0x1+1.0x2+1.0x3=8.0
2.0x1+1.0x2+1.0x3=9.0
1.0x1+2.0x2+1.0x3=11.0
x1=1.0
x2=3.0
x3=4.0

OPERATION 12: EXIT
```

五、源代码

1. 接口代码 Calculate.java

```
    package MATRIXHW;

public interface Calculate {
    Matrix Addition(Matrix p);
    Matrix Multiplication(Matrix p);
    Matrix Inverse();
    double Determinant();
}
```

2. 类代码 Matrix.java

```
    package MATRIXHW;

import javax.swing.plaf.basic.BasicInternalFrameTitlePane;
import java.util.Scanner;


class Matrix implements Calculate{
    int Row, Column;
    double[][] matrix = null;
    Matrix(){}

    Matrix(int Row, int Colummn) {
        matrix = null;
        this.Row = Row;
        this.Column = Colummn;
        matrix = new double[Row][Colummn];
    }

    Matrix(Matrix p) {
        this(p.Row, p.Column);
        for(int i = 0; i < Row; i++) {
            for(int j = 0; j < Column; j++) {
```

```
            this.matrix[i][j] = p.matrix[i][j];
        }
    }
}

void Input() {
    Scanner scan = new Scanner(System.in);
    for(int i = 0; i < Row; i++) {
        for(int j = 0; j < Column; j++) {
            matrix[i][j] = scan.nextDouble();
        }
    }
}

void Output() {
    for (int i = 0; i < Row; i++) {
        for (int j = 0; j < Column; j++) {
            System.out.printf("%5.2f ", matrix[i][j]);
        }
        System.out.println();
    }
}

Matrix Transpose() {
    Matrix New = new Matrix(this.Column, this.Row);
    for(int i = 0; i < Row; i++) {
        for(int j = 0; j < Column; j++) {
            New.matrix[j][i] = this.matrix[i][j];
        }
    }
    return New;
}

@Override
public Matrix Addition(Matrix p) {
    if(Row == p.Row && Column == p.Column) {
        Matrix New = new Matrix(Row, Column);
        for (int i = 0; i < Row; i++) {
            for (int j = 0; j < Column; j++) {
                    New.matrix[i][j] = matrix[i][j] + p.matrix[i][j];
                }
            }
            return New;
        }
        else{
            System.out.println("Cannot perform addition, please try again");
            return null;
        }
    }

    @Override
    public Matrix Multiplication(Matrix p) {
        if(Column == p.Row) {
            Matrix New;
            try{
                New = new Matrix(Row, p.Column);
            }
            catch (NullPointerException e){
                System.out.println("Data illegal, please try again");
                return null;
            }
            for(int i = 0; i < Row; i++){
                for(int j = 0; j < p.Column; j++){
                    double sum = 0;
                    for(int k = 0; k < Column; k++) {
                        sum = sum + matrix[i][k] * p.matrix[k][j];
                    }
                    New.matrix[i][j] = sum;
                }
            }
            return New;
        }
        else{
            System.out.println("Cannot perform multiplication, please try again");
            return null;
        }
    }
```

```java
@Override
public Matrix Inverse() {
    if(Row == Column)
    {
        SquareMatrix a = new SquareMatrix(this);
        return a.Inverse();
    }
    else{
        System.out.println("Cannot perform this operation");
        return null;
    }
}


@Override
public double Determinant() {
    if(Row == Column){
        SquareMatrix a = new SquareMatrix(this);
        return a.Determinant();
    }
    else{
        System.out.println("Can not preform this operation, please try again");
        return -1;
    }
}

int Rank(){
    double[][] tmp = this.matrix;
    int n = this.Column - 1;
    for(int r = 0, c = 0; r < this.Row && c < n; r++, c++){
        int t = r;
        for(int i = r + 1; i < this.Row; i++) {
            if(Math.abs(tmp[i][c]) > Math.abs(tmp[t][c])) t = i;
        }
        for(int i = c; i <= n; i++){
            double temp = tmp[r][i];
            tmp[r][i] = tmp[t][i];
            tmp[t][i] = temp;
        }
        for(int i = n; i >= c; i--){
            if(tmp[r][i] == 0 || tmp[r][c] == 0.0){
                tmp[r][i] = 0;
                continue;
            }
            tmp[r][i] /= tmp[r][c];


        }
        for(int i = r+1; i < this.Row; i++){
            for(int j = n; j >= c; j--){
                tmp[i][j] -= tmp[i][c] * tmp[r][j];
            }
        }
    }
    for(int i = 0; i < this.Row; i++) {
        for(int j = 0; j < i; j++) {
            tmp[i][j] = 0;
        }
    }
    int ret = 0, flg = 0;
    for(int i = 0; i < Row; i++){
        for(int j = 0; j < Column; j++){
            if(tmp[i][j] != 0){
                flg = 1;
                break;
            }
        }
        if(flg == 1) ret++;
        flg = 0;
    }
    return ret;
}

double[] Solve() {
    int equ = Row, var = Column - 1;
    int i, j, k, col, max_r;
    double eps = 0.000000001;
    double[][] a = new double[equ][var];
```

```
for(i = 0; i < Row; i++){
    for(j = 0; j < var; j++){
        a[i][j] = matrix[i][j];
    }
}
double x[] = new double[var];
for(i = 0; i < Row; i++){
    x[i] = matrix[i][var];
}
for(k = 0, col = 0; k < equ && col <var; k++, col++){
    max_r = k;
    for(i = k +1; i < equ; i++){
        if(Math.abs(a[i][col]) > Math.abs(a[max_r][col])){
            max_r = i;
        }
    }
    if(Math.abs(a[max_r][col]) <eps){
        System.out.println("There is no solution to the equations");
        return null;
    }
    if(k != max_r){
        for(j = col; j < var; j++){
            double tmp = a[max_r][j];
            a[max_r][j] = a[k][j];
            a[k][j] = tmp;
        }
        double tmp = x[max_r];
        x[max_r] = x[k];
        x[k] = tmp;
    }
    x[k] /= a[k][col];
    for(j = col + 1; j < var; j++){
        a[k][j] /= a[k][col];
    }
    a[k][col] = 1;
    for(i = 0; i < equ; i++){
        if(i != k){
            x[i] -= x[k] * a[i][col];
            for(j = col + 1; j <var; j++){
```

```
                        a[i][j] -= a[k][j] * a[i][col];
                    }
                    a[i][col] = 0;
                }
            }
        }
        System.out.println("The solutions of the equations are as follows");
        for(i = 0; i < var; i++)
        {
            System.out.println("x" + (i+1) + " = " + x[i]);
        }
        System.out.println();
        return x;
    }
}

class SquareMatrix extends Matrix{
    SquareMatrix(){
        super();
    }

    SquareMatrix(int Row){
        super(Row, Row);
    }

    SquareMatrix(Matrix a) {
        this.matrix = a.matrix;
        this.Row = a.Row;
    }

    boolean CheckDiagonalMatrix(){
        for(int i = 0; i < Row; i++){
            for(int j = 0; j < Row; j++){
                if(i != j && matrix[i][j] != 0){
                    return false;
                }
            }
        }
        return true;
```

```
}

int CheckTriangleMatrix(){
    int flg_up = 1, flg_low = 0;
    for(int i = 0; i < Row; i++){
        for(int j = 0; j < i; j++){
            if(matrix[i][j] != 0){
                flg_up = 0;
                break;
            }
        }
    }
    for(int i = 0; i < Row; i++){
        for(int j = Row - 1; j > i; j--){
            if(matrix[i][j] != 0){
                flg_low = 0;
                break;
            }
        }
    }
    return flg_low | flg_up;
}

SquareMatrix Create(int i, int j, SquareMatrix M){
    if(M.Row <= 1) {
        return M;
    }
    else {
        SquareMatrix New = new SquareMatrix(M.Row - 1);
        int curx = 0, cury = 0;
        for (int ii = 0; ii < M.Row; ii++) {
            if(ii == i) continue;
            for (int jj = 0; jj < M.Row; jj++) {
                if (jj == j) {
                    continue;
                }
                New.matrix[curx][cury++] = M.matrix[ii][jj];
            }
            curx++;
```

```
                cury = 0;
            }
            return New;
        }
    }

    @Override
    public double Determinant() {
        if(CheckDiagonalMatrix()){
            DiagonalMatrix a = new DiagonalMatrix(this);
            return a.Determinant();
        }
        if(CheckTriangleMatrix() == 1){
            TrianglarMatrix a = new TrianglarMatrix(this);
            return a.Determinant();
        }
        if(Row == 1) return matrix[0][0];
        else{
            double sum = 0;
            for(int i = 0; i < Row; i++){
                SquareMatrix tmp = Create(0, i, this);
                sum += Math.pow(-1, i) * matrix[0][i] * tmp.Determinant();
            }
            return sum;
        }
    }

    @Override
    public Matrix Inverse() {
        double[][] tmp = new double[Row][2 * Row];
        for(int i = 0; i < Row; i++) {
            for(int j = 0; j < Row; j++){
                tmp[i][j] = matrix[i][j];
            }
        }
        for(int i = 0; i < Row; i++){
            for(int j = Row; j < 2 * Row; j++){
                if(i + Row == j){
                    tmp[i][j] = 1;
        }
        else{
            tmp[i][j] = 0;
        }
    }
}

int n = 2 * Row - 1;
for(int r = 0, c = 0; r < this.Row && c < n; r++, c++){
    int t = r;
    for(int i = r + 1; i < this.Row; i++) {
        if(Math.abs(tmp[i][c]) > Math.abs(tmp[t][c])) t = i;
    }
    for(int i = c; i <= n; i++){
        double temp = tmp[r][i];
        tmp[r][i] = tmp[t][i];
        tmp[t][i] = temp;
    }
    for(int i = n; i >= c; i--){
        if(tmp[r][i] == 0 || tmp[r][c] == 0.0){
            tmp[r][i] = 0;
            continue;
        }
        tmp[r][i] /= tmp[r][c];
    }

    for(int i = r+1; i < this.Row; i++){
        for(int j = n; j >= c; j--){
            tmp[i][j] -= tmp[i][c] * tmp[r][j];
        }
    }
}

for(int j = Row - 1; j >= 0; j--){
    for(int i = j - 1; i >= 0; i--){
        double time = tmp[i][j] / tmp[j][j];
        for(int k = j; k < 2 * Row; k++){
            tmp[i][k] -= tmp[j][k] * time;
        }
```

```
            }
        }
        Matrix ans = new Matrix(Row, Row);
        for(int i = 0; i < Row; i++){
            for(int j = Row; j < 2 * Row; j++) {
                ans.matrix[i][j - Row] = tmp[i][j];
            }
        }
        return ans;
    }


}

class IdentityMatrix extends SquareMatrix{
    IdentityMatrix(){
        super();
    }

    IdentityMatrix(int Row){
        super(Row);
    }

    IdentityMatrix(Matrix a){
        super(a);
    }

    @Override
    public double Determinant() {
        System.out.println("This is an IdentityMatrix");
        return 1.0;
    }
}

class DiagonalMatrix extends SquareMatrix{
    DiagonalMatrix(){
        super();
    }
    DiagonalMatrix(int Row){
        super(Row);
    }

    DiagonalMatrix(Matrix a){
        super(a);
    }

    boolean CheckIdentityMatrix(){
        for(int i = 0; i < Row; i++){
            if(matrix[i][i] != 1.0){
                return false;
            }
        }
        return true;
    }

    @Override
    public double Determinant() {
        if(CheckIdentityMatrix()){
            IdentityMatrix a = new IdentityMatrix(this);
            return a.Determinant();
        }
        System.out.println("This is a DiagonalMatrix");
        double mul = 1.0;
        for(int i = 0; i < Row; i++) {
            mul *= this.matrix[i][i];
        }
        return mul;
    }
}

class TrianglarMatrix extends SquareMatrix{
    TrianglarMatrix(){
        super();
    }

    TrianglarMatrix(int Row){
        super(Row);
    }

    TrianglarMatrix(Matrix a){
        super(a);
    }

    @Override
    public double Determinant() {
        System.out.println("This is a TriangleMatrix");
        double mul = 1.0;
        for(int i = 0; i < Row; i++) {
            mul *= this.matrix[i][i];
        }
        return mul;
    }

}
```

3. 主函数 Main.java

```
    package MATRIXHW;

import java.awt.event.MouseAdapter;
import java.io.*;
import java.util.Locale;
import java.util.Scanner;


public class Main {
    public static void main(String[] args)
    {
        int count = 1;
        System.out.println("First, let`s generate a matrix:");
        System.out.println("Please enter the row of the matrix: ");
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        System.out.println("Please enter the column of the matrix: ");
        int m = scanner.nextInt();
        System.out.println("Please enter the data of this matrix");
        Matrix ex1 = new Matrix(n, m);
        ex1.Input();
        System.out.println("The matrix is as follow:");
        ex1.Output();
        try {
            String filename = "E:\\桌面\\编写\\java\\MatrixHW\\log.txt";
            File file =new File(filename);
            if(!file.exists())
            {
                try
                {
                    file.createNewFile();
                }
                catch (IOException e){};
            }
            BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(filename));
            int op = 0;
            do {
                bufferedWriter.write("\nOPERATION " + count++ + ": ");

System.out.printf("\n\n--------------------------------------------------------
--\n");
                System.out.println("Now choose the operation you would like to
perform:");
                System.out.println("1.update the matrix");
                System.out.println("2.Output the matrix");
                System.out.println("3.Transpose");
                System.out.println("4.Addition");
                System.out.println("5.Multiplication");
                System.out.println("6.Rank");
                System.out.println("7.Determinant");
                System.out.println("8.Inverse");
                System.out.println("9.Solving linear equations");
                System.out.println("0.Exit");
                op = scanner.nextInt();
                if (op == 1) {
                    bufferedWriter.write("UPDATE\n");
                    System.out.println("Please enter the new infomation of the new
matrix:");
                    System.out.println("Row:");
                    n = scanner.nextInt();
                    System.out.println("Column");
                    m = scanner.nextInt();
                    ex1 = new Matrix(n, m);
                    System.out.println("enter data:");
                    ex1.Input();
                    System.out.println("Update successfully, now the matrix is as
follow:");
                    ex1.Output();
                    StringBuilder data;
                    for(int i = 0; i < ex1.Row; i++){
                        data = new StringBuilder();
                        for(int j = 0; j < ex1.Column; j++)
                        {
                            data.append(ex1.matrix[i][j] + " ");
                        }
                        data.append("\n");
                        bufferedWriter.write(data.toString());
                    }
                } else if (op == 2) {
                    bufferedWriter.write("OUTPUT\n");
                    System.out.println("The matrix is as follow:");
                    ex1.Output();
                    StringBuilder data;
                    for(int i = 0; i < ex1.Row; i++){
                        data = new StringBuilder();
                        for(int j = 0; j < ex1.Column; j++)
                        {
                            data.append(ex1.matrix[i][j] + " ");
                        }
                        data.append("\n");
                        bufferedWriter.write(data.toString());
                    }
                } else if (op == 3) {
                    bufferedWriter.write("TRANSPOSE\n");
                    Matrix ex2 = ex1.Transpose();
                    System.out.println("Transpose matrix is as follow:");
```

```java
    ex2.Output();
    StringBuilder data;
    for(int i = 0; i < ex2.Row; i++){
        data = new StringBuilder();
        for(int j = 0; j < ex2.Column; j++)
        {
            data.append(ex2.matrix[i][j] + " ");
        }
        data.append("\n");
        bufferedWriter.write(data.toString());
    }
} else if (op == 4) {
    bufferedWriter.write("ADDITION\n");
    System.out.println("Please enter another matrix");
    System.out.println("Row:");
    n = scanner.nextInt();
    System.out.println("Column:");
    m = scanner.nextInt();
    System.out.println("enter data:");
    Matrix ex2 = new Matrix(n, m);
    ex2.Input();
    System.out.println("Result");
    Matrix ex3 = ex1.Addition(ex2);
    if (ex3 != null) {
        ex3.Output();
    }
    StringBuilder data;
    for(int i = 0; i < ex1.Row; i++){
        data = new StringBuilder();
        for(int j = 0; j < ex1.Column; j++)
        {
            data.append(ex1.matrix[i][j] + " ");
        }
        data.append("\n");
        bufferedWriter.write(data.toString());
    }
    bufferedWriter.write("+\n");
    for(int i = 0; i < ex2.Row; i++){
        data = new StringBuilder();
```

```
            for(int j = 0; j < ex2.Column; j++)
            {
                data.append(ex2.matrix[i][j] + " ");
            }
            data.append("\n");
            bufferedWriter.write(data.toString());
        }
        bufferedWriter.write("RESULT\n");
        if(ex3 == null){
            bufferedWriter.write("DATA ILLEGAL\n");
        }
        else{
            for(int i = 0; i < ex3.Row; i++){
                data = new StringBuilder();
                for(int j = 0; j < ex3.Column; j++)
                {
                    data.append(ex3.matrix[i][j] + " ");
                }
                data.append("\n");
                bufferedWriter.write(data.toString());
            }
        }

    } else if (op == 5) {
        bufferedWriter.write("MULTIPLICATION\n");
        System.out.println("Please enter another matrix");
        System.out.println("Row:");
        n = scanner.nextInt();
        System.out.println("Column:");
        m = scanner.nextInt();
        System.out.println("enter data:");
        Matrix ex2 = new Matrix(n, m);
        ex2.Input();
        System.out.println("Result");
        Matrix ex3 = ex1.Multiplication(ex2);
        if (ex3 != null) {
            ex3.Output();
        }
        StringBuilder data;
    for(int i = 0; i < ex1.Row; i++){
        data = new StringBuilder();
        for(int j = 0; j < ex1.Column; j++)
        {
            data.append(ex1.matrix[i][j] + " ");
        }
        data.append("\n");
        bufferedWriter.write(data.toString());
    }
    bufferedWriter.write("*\n");
    for(int i = 0; i < ex2.Row; i++){
        data = new StringBuilder();
        for(int j = 0; j < ex2.Column; j++)
        {
            data.append(ex2.matrix[i][j] + " ");
        }
        data.append("\n");
        bufferedWriter.write(data.toString());
    }
    bufferedWriter.write("RESULT\n");
    if(ex3 == null){
        bufferedWriter.write("DATA ILLEGAL\n");
    }
    else{
        for(int i = 0; i < ex3.Row; i++){
            data = new StringBuilder();
            for(int j = 0; j < ex3.Column; j++)
            {
                data.append(ex3.matrix[i][j] + " ");
            }
            data.append("\n");
            bufferedWriter.write(data.toString());
        }
    }
} else if (op == 6) {
    bufferedWriter.write("RANK\n");
    System.out.println("The rank of matrix is :" + ex1.Rank());
    bufferedWriter.write("RESULT :" + ex1.Rank() + "\n");
} else if (op == 7) {
```

```
                    bufferedWriter.write("DETERMINANT\nRESULT\n");
                    if (ex1.Determinant() == -1) {
                        bufferedWriter.write("DATA ILLEGAL\n");
                        continue;
                    }
                    System.out.println("The Determinant value of the matrix is : "
+ ex1.Determinant());
                    StringBuilder data = new StringBuilder();
                    data.append(ex1.Determinant() + "\n");
                    bufferedWriter.write(data.toString());
                } else if (op == 8) {
                    bufferedWriter.write("INVERSE\nRESULT\n");
                    System.out.println("The inverse matrix is as follow");
                    Matrix ex2 = ex1.Inverse();
                    if (ex2 == null) {
                        bufferedWriter.write("DATA ILLEGAL\n");
                    } else {
                        StringBuilder data;
                        ex2.Output();
                        for(int i = 0; i < ex2.Row; i++){
                            data = new StringBuilder();
                            for(int j = 0; j < ex2.Column; j++)
                            {
                                data.append(ex2.matrix[i][j] + " ");
                            }
                            data.append("\n");
                            bufferedWriter.write(data.toString());
                        }
                    }
                } else if (op == 9) {
                    bufferedWriter.write("Solve equations\n");
                    System.out.println("Please enter the information of those
equations");
                    System.out.println("The number of the equations");
                    n = scanner.nextInt();
                    System.out.println("The number of the unknown numbers");
                    m = scanner.nextInt();
                    System.out.println("Enter the coefficient of the unknown number.
For example, enter\"2 3 1 2\" means \"2x1 + 3x2 + x3 = 2\"");
```

```
                    Matrix ex2 = new Matrix(n, m + 1);

                    ex2.Input();
                    StringBuilder data;
                    for(int i = 0; i < ex2.Row; i++){
                        data = new StringBuilder();
                        for(int j = 0; j < ex2.Column; j++)
                        {
                            if(j == 0) data.append(ex2.matrix[i][j] + "x" + (j+1));
                            else if(j < ex2.Column - 1) data.append("+" +
ex2.matrix[i][j] + "x" + (j+1));
                            else data.append("=" + ex2.matrix[i][j]);
                        }
                        data.append("\n");
                        bufferedWriter.write(data.toString());
                    }
                    System.out.println("The result is as follow:");
                    double[] tmp = ex2.Solve();
                    if(tmp == null){
                        bufferedWriter.write("There is no solution to the
equations\n");
                    }
                    else{
                        for(int i = 0; i < ex2.Column - 1; i++){
                            data = new StringBuilder();
                            data.append("x" + (i+1) + "=" + tmp[i] + " ");
                            data.append("\n");
                            bufferedWriter.write(data.toString());
                        }
                    }
                }else if(op == 0){
                    bufferedWriter.write("EXIT");
                    bufferedWriter.flush();
                    bufferedWriter.close();
                }
            } while (op != 0);
        }
        catch (IOException e){
            e.printStackTrace();
            }
        }
    }
```