实现一个较为复杂的线性代数计算器

* 实验项目类型：设计性
*此表由学生按顺序填写

课程名称 面向对象程序设计/JAVA 语言 成绩评定
实验项目名称 实现一个较为复杂的线性代数计算器 指导老师 于晓聪
实验项目编号 1 实验项目类型 设计性 实验地点 数学系机房
学生姓名 郭彦培 学号 2022101149
学院 信息科学技术学院 系 数学系 专业 信息管理与信息系统
实验时间 2023 年 10 月 27 日上午 ~ 2023 年 10 月 27 日中午

一、 实验目的

综合运用面向对象设计工具、字符串、异常处理与对象的抽象等知识，结合线性代数中的算法，基于 CAS 系统实现一个较为复杂的线性代数计算器。

二、 实验环境

计算机：PC X64

操作系统：Windows

编程语言：Java

IDE：Visual Studio Code

三、 程序原理

实现了以下一些类

myLinearLib	线性计算主类	myLinearEntire
线性空间元素	抽象类：具有线性性质的基础运算	myLinearSpace
线性空间	1. 基础运算 2. 基 3. 计算秩 4. 对应矩阵	myMatrix
矩阵	1. 基础运算 2. 求秩 3. 行列式 4. 逆 5. 转置 6. 对角化	myPolynomial
多项式	1. 基础运算 2. 求值	myRealNum
实数	1. 基础运算	

实现的功能：控制台指令如下： 中括号内为需要填入字符串
尖括号为可选参数，默认为列表第一个

`addMat [columns] [rows] [digit(1,1)] ... [digit(c,r)]`	添加一个列数为 columns，行数为 rows 的矩阵
`addPol [dim] [digit 1] ... [digit r]`	添加一个阶数为 r 的多项式
`addLS [rank] [LE name 1] ... [LE name r]`	添加一个以`LE 1~r`为基的线性空间

`show <scope :"all" | "Mat" | "Pol" | "LS">`	列出对应的所有元素
`cacuMat [Mat name] <type :"Det" | "Rank" | "inverse" | "transpose" | "Eig">`	计算矩阵的行列式、秩、逆、转置和对角化
`cacuPol [Pol name] [digit]`	计算多项式的值
`op [LE name1] <operator :"+" | "-" | "*"> [LE name2]`	对线性空间元素进行基础运算
`getUID <type: "date" | "code" | "seq"> [name]`	申请对应类型的新 UID，并与一个引用名称 name 绑定
`secUID <type: "date" | "code" | "seq"> [UID]`	查找 UID 对应的引用名称

四、 程序代码

文件：`sis10\myLinearEntire.java` 实现了 `myLinearEntire` 类

```
package sis10;

import java.util.ArrayList;

public abstract class myLinearEntire {
    protected int dim;
    protected ArrayList<myLinearEntire> coordinate;

    public myLinearEntire() {
        this.dim = 0;
        this.coordinate = new ArrayList<myLinearEntire>();
    }

    public myLinearEntire(int dim, ArrayList<myLinearEntire> coordinate) {
        this.dim = dim;
        this.coordinate = new ArrayList<myLinearEntire>(coordinate);
    }
```

```
    public myLinearEntire(myLinearEntire other) {
        this.dim = other.dim;
        this.coordinate = new ArrayList<myLinearEntire>(other.coordinate);
    }

    public myLinearEntire(int _dim)
    {
        this.dim = _dim;
        this.coordinate = new ArrayList<myLinearEntire>();
    }

    public int getDim() {
        return this.dim;
    }

    public ArrayList<myLinearEntire> getCoordinate() {
        return this.coordinate;
    }

    abstract public myLinearEntire add(myLinearEntire other)  throws
Exception;

    abstract public myLinearEntire multiply(myLinearEntire other) throws
Exception;

    abstract public <N> myLinearEntire multiply(N other) throws Exception;

    abstract public myLinearEntire negative() throws Exception;

    abstract public void print() throws Exception;

    @Override
    abstract public boolean equals(Object other);

    @Override
    abstract public int hashCode();

    @Override
    abstract public String toString();

    abstract public double getValue() throws Exception;
}
```

文件：`sis10\myLinearLib.java`实现了`myLinearLib`类

```java
package sis10;

import java.util.ArrayList;
import java.util.HashMap;

public class myLinearLib {
    private HashMap<String, myLinearEntire> lib;
    private StringBuffer nowName;

    private void nextName()
    {
        int i = nowName.length() - 1;
        for(; i >= 0; i--)
        {
            if(nowName.charAt(i) != 'z')
            {
                nowName.setCharAt(i, (char)(nowName.charAt(i) + 1));
                break;
            }
            else nowName.setCharAt(i, 'a');
        }
        if(i < 0) nowName.append('a');
    }

    public myLinearLib()
    {
        lib = new HashMap<String, myLinearEntire>();
        nowName = new StringBuffer("a");
    }

    public myLinearLib(myLinearLib other)
    {
        lib = new HashMap<String, myLinearEntire>(other.lib);
        nowName = new StringBuffer(other.nowName);
    }

    public StringBuffer add(myLinearEntire obj)
    {
        StringBuffer rt = new StringBuffer(nowName);
        lib.put(rt.toString(), obj);
        nextName();
        return rt;
    }
```

```
    public myLinearEntire get(String name)
    {
        return lib.get(name);
    }

    public void remove(String name)
    {
        lib.remove(name);
    }

    public void clear()
    {
        lib.clear();
    }

    public int size()
    {
        return lib.size();
    }

    public String parseCommand(String cmd) throws Exception
    {
        String[] cmdList = cmd.split(" ");
        if(cmdList[0].equals("addMat"))
        {
            int col = Integer.parseInt(cmdList[1]);
            int row = Integer.parseInt(cmdList[2]);
            myMatrix newMat = new myMatrix(row, col);
            for(int i = 0; i < row; i++)
            {
                for(int j = 0; j < col; j++)
                {
                    newMat.set(i, j, new
myRealNum(Double.parseDouble(cmdList[3 + i * col + j])));
                }
            }
            return "成功添加矩阵: " + add(newMat).toString();
        }
        else if(cmdList[0].equals("addPol"))
        {
            int dim = Integer.parseInt(cmdList[1]);
            ArrayList<myLinearEntire> newCoordinate = new
ArrayList<myLinearEntire>();
            for(int i = 0; i < dim; i++)
```

```
            {
                newCoordinate.add(new
myRealNum(Double.parseDouble(cmdList[2 + i])));
            }
            myPolynomial newPol = new myPolynomial(dim, newCoordinate);
            return "成功添加多项式: " + add(newPol).toString();
        }
        else if(cmdList[0].equals("addLS"))
        {
            int rank = Integer.parseInt(cmdList[1]);
            ArrayList<myLinearEntire> newBasis = new
ArrayList<myLinearEntire>();
            for(int i = 0; i < rank; i++)
            {
                newBasis.add(get(cmdList[2 + i]));
            }
            myLinearSpace newLS = new myLinearSpace(rank, newBasis);
            add(newLS);
            return nowName.toString();
        }
        else if(cmdList[0].equals("show"))
        {
            if(cmdList[1].equals("all"))
            {
                return "Mats\n" + parseCommand("show Mat") +
                       "Pols\n" + parseCommand("show Pol") +
                       "LSs\n"+ parseCommand("show LS");
            }
            else if(cmdList[1].equals("Mat"))
            {
                String str = "";
                for(String key : lib.keySet())
                {
                    if(lib.get(key) instanceof myMatrix)
                    {
                        str += key + " : " + lib.get(key).toString() +
"\n";
                    }
                }
                return str;
            }
            else if(cmdList[1].equals("Pol"))
            {
                String str = "";
```

```
                for(String key : lib.keySet())
                {
                    if(lib.get(key) instanceof myPolynomial)
                    {
                        str += key + " : " + lib.get(key).toString() +
"\n";
                    }
                }
                return str;
            }
            else if(cmdList[1].equals("LS"))
            {
                String str = "";
                for(String key : lib.keySet())
                {
                    if(lib.get(key) instanceof myLinearSpace)
                    {
                        str += key + " : " + lib.get(key).toString() +
"\n";
                    }
                }
                return str;
            }
            else throw new Exception("非法的 show 指令。");
        }
        else if(cmdList[0].equals("cacuMat"))
        {
            if(get(cmdList[1]) instanceof myMatrix == false) throw new
Exception("目标不是矩阵");
            myMatrix cacuMatrix = (myMatrix)get(cmdList[1]);
            if(cmdList[2].equals("Det"))
            {
                return String.valueOf(cacuMatrix.det());
            }
            else if(cmdList[2].equals("Rank"))
            {
                return String.valueOf(cacuMatrix.getRank());
            }
            else if(cmdList[2].equals("inverse"))
            {
                return cacuMatrix.inverse().toString();
            }
            else if(cmdList[2].equals("transpose"))
            {
```

```
                return cacuMatrix.transpose().toString();
            }
            else if(cmdList[2].equals("Eig"))
            {
                return cacuMatrix.Eig().toString();
            }
            else throw new Exception("非法的 cacuMat 指令。");
        }
        else if(cmdList[0].equals("cacuPol"))
        {
            if(get(cmdList[1]) instanceof myPolynomial == false) throw new
Exception("目标不是多项式。");
            myPolynomial cacuPolynomial = (myPolynomial)get(cmdList[1]);
            return
String.valueOf(cacuPolynomial.cacu(Double.parseDouble(cmdList[2])));
        }
        else if(cmdList[0].equals("op"))
        {
            if(cmdList[2].equals("+"))
            {
                if(get(cmdList[1]).getClass() !=
get(cmdList[3]).getClass()) throw new Exception("两个元素类型不同。");
                myLinearEntire ans = get(cmdList[1]).add(get(cmdList[3]));
                return add(ans).toString() + ":\n" + ans.toString();
            }
            else if(cmdList[2].equals("-"))
            {
                if(get(cmdList[1]).getClass() !=
get(cmdList[3]).getClass()) throw new Exception("两个元素类型不同。");
                myLinearEntire ans =
get(cmdList[1]).add(get(cmdList[3]).multiply(-1));
                return add(ans).toString() + ":\n" + ans.toString();
            }
            else if(cmdList[2].equals("*"))
            {
                if(get(cmdList[1]).getClass() !=
get(cmdList[3]).getClass()) throw new Exception("两个元素类型不同。");
                myLinearEntire ans =
get(cmdList[1]).multiply(get(cmdList[3]));
                return add(ans).toString() + ":\n" + ans.toString();
            }
            else throw new Exception("非法的 op 指令。");
        }
        else throw new Exception("非法的指令。");
```

```
    }
}
```

文件：`sis10\myLinearSpace.java`实现了`myLinearSpace`类

```
package sis10;

import java.util.ArrayList;

public class myLinearSpace extends myLinearEntire {

    private int rank;
    private ArrayList<myLinearEntire> basis;

    //↓覆写超类方法
    @Override
    public boolean equals(Object other){
        if(other instanceof myLinearSpace){
            myLinearSpace mOther = (myLinearSpace)other;
            if(this.rank != mOther.rank) return false;
            for(int i = 0; i < this.rank; i++){
                if(!this.basis.get(i).equals(mOther.basis.get(i))) return
false;
            }
            return true;
        }
        else return false;
    }

    @Override
    public int hashCode(){
        int hash = 0;
        for(int i = 0; i < this.rank; i++){
            hash += this.basis.get(i).hashCode();
        }
        return hash;
    }

    @Override
    public String toString(){
        String str = "";
        for(int i = 0; i < this.rank; i++){
            str += this.basis.get(i).toString() + "\n";
```

```
        }
        return str;
    }

    //↓构造函数
    public myLinearSpace(int rank) {
        this.rank = rank;
        this.basis = new ArrayList<myLinearEntire>();
    }

    public myLinearSpace(int rank, ArrayList<myLinearEntire> _basis) {
        this.rank = rank;
        this.basis = new ArrayList<myLinearEntire>(_basis);
    }

    public myLinearSpace(myLinearSpace other) {
        this.rank = other.rank;
        this.basis = new ArrayList<myLinearEntire>(other.basis);
    }

    //↓覆写父类方法
    public myLinearSpace add(myLinearEntire other) throws Exception
    {
        if (other instanceof myLinearSpace) {
            myLinearSpace mOther = (myLinearSpace) other;
            if (this.getRank() != mOther.getRank()) {
                throw new IllegalArgumentException("秩不同的线性空间不能相
加。");
            }
            ArrayList<myLinearEntire> newBasis = new
ArrayList<myLinearEntire>();
            for (int i = 0; i < this.rank; i++) {
                newBasis.add(this.basis.get(i).add(mOther.basis.get(i)));
            }
            return new myLinearSpace(this.rank, newBasis);
        } else {
            throw new Exception("非法的线性空间相加。");
        }
    }

    public myLinearSpace multiply(myLinearEntire other) throws Exception
    {
        if (other instanceof myLinearSpace) {
            myLinearSpace mOther = (myLinearSpace) other;
```

```
            if (this.getRank() != mOther.getRank()) {
                throw new IllegalArgumentException("秩不同的线性空间不能相
乘。");
            }
            ArrayList<myLinearEntire> newBasis = new
ArrayList<myLinearEntire>();
            for (int i = 0; i < this.rank; i++) {

newBasis.add(this.basis.get(i).multiply(mOther.basis.get(i)));
            }
            return new myLinearSpace(this.rank, newBasis);
        } else {
            throw new Exception("非法的线性空间相乘。");
        }
    }

    public <N> myLinearSpace multiply(N other) throws Exception
    {
        throw new Exception("线性空间的数乘无意义");
    }

    public double getValue() throws Exception
    {
        throw new Exception("线性空间的值无意义");
    }

    public myLinearSpace negative() throws Exception
    {
        ArrayList<myLinearEntire> newBasis = new
ArrayList<myLinearEntire>();
        for (int i = 0; i < this.rank; i++) {
            newBasis.add(this.basis.get(i).negative());
        }
        return new myLinearSpace(this.rank, newBasis);
    }

    //↓线性空间特有方法
    public int getRank()
    {
        return this.rank;
    }

    public ArrayList<myLinearEntire> getBasis()
    {
```

```
        return new ArrayList<myLinearEntire>(basis);
    }

    public void print() throws Exception
    {
        System.out.println("线性空间的秩为" + this.rank);
        System.out.println("线性空间的基为");
        for (int i = 0; i < this.rank; i++) {
            System.out.print("第" + i + "个基向量为");
            this.basis.get(i).print();
        }
    }
}
```

文件：`sis10\myMatrix.java`实现了`myMatrix.java`类

```
package sis10;

import java.util.ArrayList;

public class myMatrix extends myLinearEntire{
    private int row;
    private int col;
    private ArrayList<ArrayList<myLinearEntire>> matrix;

    //↓覆写超类方法
    @Override
    public boolean equals(Object other){
        if(other instanceof myMatrix){
            myMatrix mOther = (myMatrix)other;
            if(this.row != mOther.row || this.col != mOther.col) return
false;
            for(int i = 0; i < this.row; i++){
                for(int j = 0; j < this.col; j++){
                    if(!
this.matrix.get(i).get(j).equals(mOther.matrix.get(i).get(j))) return
false;
                }
            }
            return true;
        }
        else return false;
    }
```

```
    @Override
    public int hashCode(){
        int hash = 0;
        for(int i = 0; i < this.row; i++){
            for(int j = 0; j < this.col; j++){
                hash += this.matrix.get(i).get(j).hashCode();
            }
        }
        return hash;
    }

    @Override
    public String toString(){
        String str = "";
        for(int i = 0; i < this.row; i++){
            for(int j = 0; j < this.col; j++){
                str += this.matrix.get(i).get(j).toString() + " ";
            }
            if(i != this.row-1)str += "\n";
        }
        return str;
    }

    //↓构造函数
    public myMatrix(int row, int col) {
        this.row = row;
        this.col = col;
        this.matrix = new ArrayList<ArrayList<myLinearEntire>>();
        for (int i = 0; i < row; i++) {
            ArrayList<myLinearEntire> newRow = new
ArrayList<myLinearEntire>();
            for (int j = 0; j < col; j++) {
                newRow.add(new myRealNum(0));
            }
            this.matrix.add(newRow);
        }
    }

    public myMatrix(int row, int col, ArrayList<ArrayList<myLinearEntire>>
matrix) {
        this.row = row;
        this.col = col;
        this.matrix = new ArrayList<ArrayList<myLinearEntire>>(matrix);
```

```
    }

    public myMatrix(myMatrix other) {
        this.row = other.row;
        this.col = other.col;
        this.matrix = new
ArrayList<ArrayList<myLinearEntire>>(other.matrix);
    }

    //↓ 覆写父类方法
    public myMatrix add(myLinearEntire other) throws Exception {
        if (other instanceof myMatrix) {
            myMatrix mOther = (myMatrix) other;
            if (this.getRow() != mOther.getRow() || this.getCol() !=
mOther.getCol()) {
                throw new IllegalArgumentException("行列不同的矩阵不能相
加。");
            }
            ArrayList<ArrayList<myLinearEntire>> newMatrix = new
ArrayList<ArrayList<myLinearEntire>>();
            for (int i = 0; i < this.row; i++) {
                ArrayList<myLinearEntire> newRow = new
ArrayList<myLinearEntire>();
                for (int j = 0; j < this.col; j++) {

newRow.add(this.matrix.get(i).get(j).add(mOther.matrix.get(i).get(j)));
                }
                newMatrix.add(newRow);
            }
            return new myMatrix(this.row, this.col, newMatrix);
        } else {
            throw new Exception("非法的矩阵相加。");
        }
    }

    public myMatrix multiply(myLinearEntire other) throws Exception {
        if (other instanceof myMatrix) {
            myMatrix mOther = (myMatrix) other;
            if (this.getCol() != mOther.getRow()) {
                throw new IllegalArgumentException("行列不匹配的矩阵不能相
乘。");
            }
            ArrayList<ArrayList<myLinearEntire>> newMatrix = new
ArrayList<ArrayList<myLinearEntire>>();
```

```
            for (int i = 0; i < this.row; i++) {
                ArrayList<myLinearEntire> newRow = new
ArrayList<myLinearEntire>();
                for (int j = 0; j < mOther.col; j++) {
                    myLinearEntire sum = new myRealNum(0);
                    for (int k = 0; k < this.col; k++) {
                        sum =
sum.add(this.matrix.get(i).get(k).multiply(mOther.matrix.get(k).get(j)));
                    }
                    newRow.add(sum);
                }
                newMatrix.add(newRow);
            }
            return new myMatrix(this.row, mOther.col, newMatrix);
        } else {
            throw new Exception("非法的矩阵相乘。");
        }
    }

    public <N> myMatrix multiply(N other) throws Exception {
        if (other instanceof myRealNum) {
            myRealNum rOther = (myRealNum) other;
            ArrayList<ArrayList<myLinearEntire>> newMatrix = new
ArrayList<ArrayList<myLinearEntire>>();
            for (int i = 0; i < this.row; i++) {
                ArrayList<myLinearEntire> newRow = new
ArrayList<myLinearEntire>();
                for (int j = 0; j < this.col; j++) {
                    newRow.add(this.matrix.get(i).get(j).multiply(rOther));
                }
                newMatrix.add(newRow);
            }
            return new myMatrix(this.row, this.col, newMatrix);
        } else {
            throw new Exception("非法的矩阵数乘。");
        }
    }

    public double getValue()
    {
        try {
            return this.det();
        } catch (Exception e) {
            e.printStackTrace();
```

```
            return 0;
        }
    }

    public void print() throws Exception {
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col - 1; j++) {
                this.matrix.get(i).get(j).print();
                System.out.print(" ");
            }
            this.matrix.get(i).get(this.col - 1).print();
            System.out.println();
        }
    }

    public myLinearEntire negative() throws Exception {
        ArrayList<ArrayList<myLinearEntire>> newMatrix = new
ArrayList<ArrayList<myLinearEntire>>();
        for (int i = 0; i < this.row; i++) {
            ArrayList<myLinearEntire> newRow = new
ArrayList<myLinearEntire>();
            for (int j = 0; j < this.col; j++) {
                newRow.add(this.matrix.get(i).get(j).negative());
            }
            newMatrix.add(newRow);
        }
        return new myMatrix(this.row, this.col, newMatrix);
    }

    //↓矩阵特有方法
    public int getRow() {
        return this.row;
    }

    public int getCol() {
        return this.col;
    }

    public double det() throws Exception //计算行列式
    {
        if (this.row != this.col) {
            throw new Exception("非方阵无法求行列式。");
        }
        if (this.row == 1) {
```

```
            return ((myRealNum) this.matrix.get(0).get(0)).getNum();
        }
        double ans = 0;
        for (int i = 0; i < this.row; i++) {
            myMatrix subMatrix = new myMatrix(this.row - 1, this.col - 1);
            for (int j = 1; j < this.row; j++) {
                for (int k = 0; k < this.col; k++) {
                    if (k < i) {
                        subMatrix.matrix.get(j -
1).add(this.matrix.get(j).get(k));
                    } else if (k > i) {
                        subMatrix.matrix.get(j -
1).add(this.matrix.get(j).get(k));
                    }
                }
            }
            ans += ((myRealNum) this.matrix.get(0).get(i)).getNum() *
subMatrix.det() * (int) Math.pow(-1, i);
        }
        return ans;
    }

    public int getRank() throws Exception //计算秩
    {
        if (this.row == 1) {
            for (int i = 0; i < this.col; i++) {
                if (((myRealNum) this.matrix.get(0).get(i)).getNum() != 0)
{
                    return 1;
                }
            }
            return 0;
        }
        int ans = 0;
        for (int i = 0; i < this.col; i++) {
            myMatrix subMatrix = new myMatrix(this.row - 1, this.col - 1);
            for (int j = 1; j < this.row; j++) {
                for (int k = 0; k < this.col; k++) {
                    if (k < i) {
                        subMatrix.matrix.get(j -
1).add(this.matrix.get(j).get(k));
                    } else if (k > i) {
                        subMatrix.matrix.get(j -
1).add(this.matrix.get(j).get(k));
```

```
                    }
                }
            }
            if (((myRealNum) this.matrix.get(0).get(i)).getNum() != 0) {
                ans += subMatrix.getRank();
            }
        }
        return ans;
    }

    public myMatrix inverse() throws Exception //计算逆矩阵
    {
        if (this.row != this.col) {
            throw new Exception("非方阵无法求逆。");
        }
        if (this.det() == 0) {
            throw new Exception("行列式为 0，无法求逆。");
        }
        myMatrix ans = new myMatrix(this.row, this.col);
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                myMatrix subMatrix = new myMatrix(this.row - 1, this.col -
1);
                for (int k = 0; k < this.row; k++) {
                    for (int l = 0; l < this.col; l++) {
                        if (k < i && l < j) {

subMatrix.matrix.get(k).add(this.matrix.get(k).get(l));
                        } else if (k < i && l > j) {

subMatrix.matrix.get(k).add(this.matrix.get(k).get(l));
                        } else if (k > i && l < j) {
                            subMatrix.matrix.get(k -
1).add(this.matrix.get(k).get(l));
                        } else if (k > i && l > j) {
                            subMatrix.matrix.get(k -
1).add(this.matrix.get(k).get(l));
                        }
                    }
                }
                ans.matrix.get(i).add(new myRealNum(subMatrix.det() * (int)
Math.pow(-1, i + j)));
            }
        }
```

```
        ans = ans.transpose();
        ans = ans.multiply(new myRealNum(1 / this.det()));
        return ans;
    }

    public myMatrix transpose() throws Exception //计算转置矩阵
    {
        myMatrix ans = new myMatrix(this.col, this.row);
        for (int i = 0; i < this.col; i++) {
            for (int j = 0; j < this.row; j++) {
                ans.matrix.get(i).add(this.matrix.get(j).get(i));
            }
        }
        return ans;
    }

    public myMatrix Eig() throws Exception//进行对角化
    {
        if (this.row != this.col) {
            throw new Exception("非方阵无法求特征值。");
        }
        myMatrix ans = new myMatrix(this.row, this.col);
        for (int i = 0; i < this.row; i++) {
            ans.matrix.get(i).add(this.matrix.get(i).get(i));
        }
        return ans;
    }

    public void set(int i, int j, myRealNum myRealNum) //重设矩阵某个元素
    {
        this.matrix.get(i).set(j, myRealNum);
    }

}
```

文件：`sis10\myPolynomial.java`实现了`myPolynomial`类

```
package sis10;

import java.util.ArrayList;

public class myPolynomial extends myLinearEntire{
```

```
    int xValue = 1;

    //↓覆写超类方法
    @Override
    public boolean equals(Object other){
        if(other instanceof myPolynomial){
            myPolynomial mOther = (myPolynomial)other;
            if(this.dim != mOther.dim) return false;
            for(int i = 0; i < this.dim; i++){
                if(!
this.coordinate.get(i).equals(mOther.coordinate.get(i))) return false;
            }
            return true;
        }
        else return false;
    }

    @Override
    public int hashCode(){
        int hash = 0;
        for(int i = 0; i < this.dim; i++){
            hash += this.coordinate.get(i).hashCode();
        }
        return hash;
    }

    @Override
    public String toString(){
        String str = "";
        for(int i = 0; i < this.dim; i++){
            str += this.coordinate.get(i).toString() + "\n";
        }
        return str;
    }

    //↓构造函数
    public myPolynomial() {
        super();
    }

    public myPolynomial(int _dim, ArrayList<myLinearEntire> _coordinate) {
        super(_dim, _coordinate);
    }
```

```
    public myPolynomial(myPolynomial other) {
        super(other);
    }

    public myPolynomial(int _dim)
    {
        super(_dim);
        for(int i = 0;i < _dim;i ++)
            coordinate.add(new myRealNum(0));
    }

    //↓覆写父类方法
    public myPolynomial add(myLinearEntire other) throws Exception {
        if (other instanceof myPolynomial) {
            myPolynomial mOther = (myPolynomial) other;
            ArrayList<myLinearEntire> newCoordinate = new
ArrayList<myLinearEntire>();
            for (int i = 0; i < Math.min(this.getDim(),other.getDim()); i+
+) {

newCoordinate.add(this.getCoordinate().get(i).add(mOther.getCoordinate().get(i)));
            }
            for (int i = Math.min(this.getDim(),other.getDim()); i <
Math.max(this.getDim(),other.getDim()); i++) {
                if (this.getDim() > other.getDim()) {
                    newCoordinate.add(this.getCoordinate().get(i));
                } else {
                    newCoordinate.add(mOther.getCoordinate().get(i));
                }
            }
            return new myPolynomial(this.getDim(), newCoordinate);
        } else {
            throw new Exception("非法的多项式相加。");
        }
    }

    public myPolynomial multiply(myLinearEntire other) throws Exception {
        if (other instanceof myPolynomial) {
            myPolynomial mOther = (myPolynomial) other;
            ArrayList<myLinearEntire> newCoordinate = new
ArrayList<myLinearEntire>();
            int aimDim = this.getDim() + mOther.getDim() - 1;
            for (int i = 0; i < aimDim; i++) {
                newCoordinate.add(new myRealNum(0));
```

```
            }
            for (int i = 0; i < this.getDim(); i++) {
                for (int j = 0; j < mOther.getDim(); j++) {
                    newCoordinate.set(i + j, newCoordinate.get(i +
j).add(this.getCoordinate().get(i).multiply(mOther.getCoordinate().get(j))));
                }
            }
            return new myPolynomial(aimDim, newCoordinate);
        } else {
            throw new Exception("非法的多项式相乘。");
        }
    }

    public <N> myPolynomial multiply(N other) throws Exception {
        ArrayList<myLinearEntire> newCoordinate = new
ArrayList<myLinearEntire>();
        for (int i = 0; i < this.getDim(); i++) {
            newCoordinate.add(this.getCoordinate().get(i).multiply(other));
        }
        return new myPolynomial(this.getDim(), newCoordinate);
    }

    public void print() throws Exception {
        for (int i = 0; i < this.getDim(); i++) {
            this.getCoordinate().get(i).print();
        }
    }

    public double getValue()
    {
        try {
            return this.cacu(1);
        } catch (Exception e) {
            e.printStackTrace();
            return 0;
        }
    }

    public myLinearEntire negative() throws Exception {
        ArrayList<myLinearEntire> newCoordinate = new
ArrayList<myLinearEntire>();
        for (int i = 0; i < this.getDim(); i++) {
            newCoordinate.add(this.getCoordinate().get(i).negative());
        }
```

```
        return new myPolynomial(this.getDim(), newCoordinate);
    }

    //↓多项式特有方法
    public double cacu(double x) throws Exception { //给定 x 的值计算多项式的
值
        double result = 0;
        for (int i = 0; i < this.getDim(); i++) {
            result += this.getCoordinate().get(i).getValue();
        }
        return result;
    }

    public int getDim() {
        return super.dim;
    }

    public ArrayList<myLinearEntire> getCoordinate() {
        return super.coordinate;
    }

    public void set(int index, myLinearEntire value) //设置多项式的某一项
    {
        super.coordinate.set(index, value);
    }
}
```

文件：`sis10\myRealNum.java` 实现了 `myRealNum` 类

```
package sis10;

public class myRealNum extends myLinearEntire{
    private double value;

    //↓覆写超类方法
    @Override
    public boolean equals(Object other){
        if(other instanceof myRealNum){
            myRealNum mOther = (myRealNum)other;
            if(this.value != mOther.value) return false;
            else return true;
        }
        else return false;
```

```
    }

    @Override
    public int hashCode(){
        return (int)this.value;
    }

    @Override
    public String toString(){
        return String.valueOf(this.value);
    }

    //↓构造函数
    public myRealNum(double value) {
        this.value = value;
    }

    public myRealNum(myRealNum other) {
        this.value = other.value;
    }

    //↓覆写父类方法
    public myLinearEntire add(myLinearEntire other) throws Exception {
        if (other instanceof myRealNum) {
            myRealNum mOther = (myRealNum) other;
            return new myRealNum(this.value + mOther.value);
        } else {
            throw new Exception("非法的实数相加。");
        }
    }

    public <N> myLinearEntire multiply(N other) throws Exception  {
        return new myRealNum(value * (double) other);
    }

    public myLinearEntire multiply(myLinearEntire other) throws Exception {
        if (other instanceof myRealNum) {
            myRealNum mOther = (myRealNum) other;
            return new myRealNum(this.value * mOther.value);
        } else {
            throw new Exception("非法的实数相乘。");
        }
    }
```

```
    public void print() throws Exception {
        System.out.println(this.value);
    }

    public double getValue() {
        return this.value;
    }

    public double getNum() {
        return this.value;
    }

    public myLinearEntire negative() throws Exception {
        return new myRealNum(-this.value);
    }

}
```

文件：`sis10\Test.java`实现了`Test`类，作为入口与测试用

```
package sis10;

import java.util.Scanner;


public class Test {
    public static void main(String[] args) {
        try (Scanner sc = new Scanner(System.in)) {
            myLinearLib lib = new myLinearLib();
            for(;;)
            {
                String s = sc.nextLine();
                try {
                    System.out.println(lib.parseCommand(s));
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }

}
```

五、 出现的问题、原因与解决方法

由于代码量较大，且没有开源项目可参考，编码过程中遇到了不少阻碍。

其中最大的困难是难以将所有具有线性性的元素归入统一的框架内。好在JAVA 提供了抽象类这一工具。于是我构造了一个抽象类`myLinearEntire`，所有的其他类型具有线性性的元素都可以继承此抽象类，并且需要实现抽象类所规定的抽象成员方法。

这样做，可以很方便地统一各个线性元素之间、不同元素之间的运算、存储、管理等操作，并且为程序提供了良好的可扩展性。

在不同的类之间，我使用了保护性传值和深拷贝等技巧，保证了所有成员变量的安全性，同时有完备的异常处理，程序可以以较高的鲁棒性运行。

有一个很小的问题，虽然是实现此系统过程中无数小问题中的一个，但我认为值得关注。在实现矩阵间运算时，我需要判断传入的`myLinearEntire`类型的对象是否为矩阵。经实验，嵌套`instanceof`并不能实现此功能。查阅资料得在`Objec`超类中实现了`getClass()`方法，可以运行时获取对象的类型。

在后续的编码过程中，我加深了对此方法的理解，意识到这个方法对 JAVA 的面向对象系统具有重要意义，于是查阅了资料，研究了此方法的原理及其对 JAVA 面向对象系统的意义，总结如下：

1. Object.getClass() 方法的作用：
   获取对象的运行时类： 返回一个 Class 对象，该对象包含了与调用该方法的对象的实际类相关的信息。
   支持运行时类型检查： 可以使用 getClass()方法进行运行时的类型检查。例如，可以在运行时确定一个对象是否属于特定的类或接口。
2. 原理：
   Object.getClass()方法的原理是基于 Java 的反射机制。反射是在运行时动态获取类的信息的机制。当调用 getClass()方法时，会返回一个 Class 对象，该对象包含有关类的信息，如类的名称、字段、方法等。
   在 Java 中，每个类都有一个对应的 Class 对象，这个对象是在加载类时由 Java 虚拟机创建的。因此，getClass()方法实际上是返回对象所属类的 Class 对象的引用。
3. 对 Java 面向对象体系的贡献：
   运行时类型信息（RTTI）： Object.getClass()方法提供了一种在运行时获取对

象类型信息的机制，使得可以在程序运行时动态地了解和操作对象的类型。
动态创建对象： 反射机制通过 Class 对象可以在运行时动态创建类的实例，这对一些框架、库和工具的开发是非常有用的。
框架和工具的实现： 反射机制为许多框架和工具提供了基础，例如，通过反射可以在运行时动态地加载和管理类，实现插件系统，以及处理配置文件中的类名等。
动态代理： 反射机制支持动态代理的实现，允许在运行时生成代理类来实现特定接口或继承特定类的代理对象。

六、 测试数据与运行结果

输入	输出	解释

`addMat 2 2 1 2 3 4`
`addMat 2 2 1 0 0 1`
`addMat 2 3 1 1 2 3 4 5 6`
`addMat 2 3 1 0 0 0 1 2 3`
	成功添加矩阵：a
	成功添加矩阵：b
	成功添加矩阵：c
	成功添加矩阵：d
		添加矩阵

`addPol 2 1 2` `addPol 2 1 0`	成功添加多项式：e 成功添加多项式：f	添加多项式
`addLS 2 e f`	`h`	添加线性空间

`show all`	`Mats`
	`a : 1.0 2.0`
	`3.0 4.0`
	`b : 1.0 0.0`
	`0.0 1.0`
	`c : 1.0 1.0`
	`2.0 3.0`
	`4.0 5.0`
	`d : 1.0 0.0`
	`0.0 0.0`
	`1.0 2.0`
	`Pols`
	`e : 1.0·x^ 0+ 2.0·x^ 1`
	`f : 1.0·x^ 0+ 0.0·x^ 1`
	`LSs`
	`g : 1.0·x^ 0+ 2.0·x^ 1`
	`1.0·x^ 0+ 0.0·x^ 1`
		显示内存中储存了的所有元素

		续表

输入	输出	解释

`op a + b`	`i:`
	`2.0 2.0`
	`3.0 5.0`
		计算矩阵 a+b
		输出结果并记为 i

`op a * b`	`j:`
	`1.0 2.0`
	`3.0 4.0`
		计算矩阵 a * b
		输出结果并记为 j

`op c * d`	行列不匹配的矩阵不能相乘。	

`op c * a`	`k:`
	`4.0 6.0`
	`11.0 16.0`
	`19.0 28.0`
		计算矩阵 c * a
		输出结果并记为 k

`cacuMat a Rank`	`2`	a 的秩为 2
`cacuMat d Rank`	`1`	b 的秩为 2
`cacuMat c Det`	非方阵无法求行列式。	
`cacuMat a Det`	`-2.0`	矩阵 a 的行列式为-2
`cacuPol e 2`	`3.0`	将$x=2$代入多项式 e

`op e * f`	`l:`
	`1.0·x^ 0+ 2.0·x^ 1+`
	`0.0·x^ 2`
		计算多项式 e * f
		输出结果并记为 l

`cacuPol l 1`	`3.0`	将$x=1$代入多项式 l
`cacuPol l 3`	`7.0`	将$x=3$代入多项式 l
`addMat 3 3 2 1 -1 1 1 1 1 2 3`	成功添加矩阵：`m`	

`cacuMat m inverse`	`n:`
	`-1.0 5.0 -2.0`
	`2.0 -7.0 3.0`
	`-1.0 3.0 -1.0`
		计算矩阵 m 的逆

`addMat 3 3 2 1 -1 1 2 3 1 2 3`	成功添加矩阵：`o`	
`cacuMat o inverse`	行列式为`0`，无法求逆。	
`cacuMat o Rank`	`2`	果然不满秩