输入输出流程序设计基础

兰州大学信息科学与工程学院 徐宇奇 320190902531

第一部分

1.3 第三题

```
package lesson5;
/*文件的字符编码转换
运行此程序时，需要提前写一个test.txt文件放在当前目录；
程序主方法类中有一处错误的代码，需要修改编码方式；

此程序利用先从test.txt读取字符串，然后利用不同的编码方式复制文件
*/

import java.io.*;

public class FileConvert {
    public static void main(String[] args) throws IOException {
        FileConvert myapp = new FileConvert();
        myapp.readFile("src/lesson5/test.txt");
        myapp.copyFile("src/lesson5/test.txt", null, "src/lesson5/unicode.txt",
"Unicode");
        myapp.copyFile("src/lesson5/test.txt", null, "src/lesson5/utf8.txt",
"UTF-8");
        myapp.readFile("src/lesson5/unicode.txt");    //按照本地平台的编码读取字符，读
取到错误的数据
        //myapp.readFile("src/lesson5/unicode.txt","Unicode");  //应该修改为此句话
        myapp.readFile("src/lesson5/utf8.txt", "UTF-8");
    }

    /**
     * 从一个文件中逐行读取字符串，使用本地字符编码
     */
    public void readFile(String fileName) throws IOException {        //声明此方法
可能会产生异常
        readFile(fileName, null);        //使用本地字符编码读取文件
    }

    /**
     * 从一个文件中逐行读取字符串，参数charsetName用于指定文件的字符编码
     */
    public void readFile(String fileName, String charsetName) throws IOException
{
        InputStream in = new FileInputStream(fileName);        //父类引用指向子类对
象
        InputStreamReader reader;    //字节流转换成字符流

        if (charsetName == null)
            reader = new InputStreamReader(in);
```

```
        else
            reader = new InputStreamReader(in, charsetName);

        BufferedReader br = new BufferedReader(reader);    //缓冲字符流
        String data;
        while ((data = br.readLine()) != null)        //逐行读取数据
            System.out.println(data);
        br.close();
    }

    /**
     * 把一个文件中的字符内容复制到另一个文件中，并且进行了相关的字符编码转换
     */
    public void copyFile(String from, String charsetFrom, String to, String
charsetTo) throws IOException {
        InputStream in = new FileInputStream(from);    //父类引用指向子类对象
        InputStreamReader reader;        //字节流转换成字符流
        if (charsetFrom == null)
            reader = new InputStreamReader(in);
        else
            reader = new InputStreamReader(in, charsetFrom);
        BufferedReader br = new BufferedReader(reader);    //缓冲字符流
        OutputStream out = new FileOutputStream(to);    //父类引用指向子类对象
        OutputStreamWriter writer = new OutputStreamWriter(out, charsetTo);    //
字节流转换成字符流
        BufferedWriter bw = new BufferedWriter(writer);
        String data;
        while ((data = br.readLine()) != null)    //向目标文件逐行写数据
            bw.write(data + "\n");
        br.close();
        bw.close();
    }
}
```

1.4 第四题

输出对象 读取对象

Java 提供了一种对象序列化的机制，该机制中，一个对象可以被表示为一个字节序列，该字节序列包括该对象的数据、有关对象的类型的信息和存储在对象中数据的类型。

将序列化对象写入文件之后，可以从文件中读取出来，并且对它进行反序列化，也就是说，对象的类型信息、对象的数据，还有对象中的数据类型可以用来在内存中新建对象。

整个过程都是 Java 虚拟机（JVM）独立的，也就是说，在一个平台上序列化的对象可以在另一个完全不同的平台上反序列化该对象。

类 ObjectInputStream 和 ObjectOutputStream 是高层次的数据流，它们包含反序列化和序列化对象的方法。

第二部分

2.2 第二题

```
package lesson5;
import java.io.*;
public class RandomTest{
    public static void main(String args[]){
        File f=new File("RandomTest.java");
        try{
            RandomAccessFile random=new RandomAccessFile(f,"r");
            //创建了指向文件f的对象random，只读方式打开
            //在生成一个随机文件对象时，除了要指明文件对象之外，还需要指明访问文件的模式。
            long l=random.length();//取此文件的长度
            char ch;
            for (long i=l-1;i>=0;i--) {
                random.seek(i); //seek方法可以将指针定位到i处
                ch = (char) random.read();  //转化为字符型
                System.out.print(ch);
            }
            random.close();
        }catch(Exception e){
            System.out.println("IOException! ");
        }
    }
}
```

第三部分

3.2 第二题

```
package lesson5;
/*
从文本文件中读取数据，通过第一行存储的行数和列数获取矩阵的行和列。
之后，从第二行开始读取矩阵的数据，数据之间以空格分隔。
同时，提供矩阵的加减乘运算。
另外，当两个矩阵不符合矩阵的运算规则时，抛出异常。
*/

import java.io.*;

public class Matrix {
    int Row = 0;
    int Col = 0;
    double[][] data;
    //构造函数
    public Matrix() {
        Row = 0;
        Col = 0;
        data = null;
    }

    /*public Matrix(String fileName){
        Matrix result=new Matrix()
    }*/
    public Matrix(int row, int col, double[][] data) {
```

```
        Row = row;
        Col = col;
        data = data;
    }

    public static void main(String[] args) throws Exception {
        Matrix first = new Matrix();
        first = first.read("src/lesson5/first.txt");
        Matrix second = new Matrix();
        second = second.read("src/lesson5/second.txt");
        Matrix third = new Matrix();
        third = third.read("src/lesson5/third.txt");
        Matrix fourth = new Matrix();
        fourth = fourth.read("src/lesson5/fourth.txt");
        Matrix result = new Matrix();
        result = result.add(first, second);
        result.write(result, "src/lesson5/result.txt");
        result = result.sub(first, second);
        result.write(result, "src/lesson5/result.txt");
        result = result.mul(first, third);
        result.write(result, "src/lesson5/result.txt");
        result = result.div(first, fourth);
        result.write(result, "src/lesson5/result.txt");
    }

    //读取行列
    public int getRow(String fileNameFrom) throws Exception {
        FileReader reader = new FileReader(fileNameFrom);
        BufferedReader br = new BufferedReader(reader);
        String[] ss = br.readLine().split("\\s+");
        Row = Integer.valueOf(ss[0]);
        return Row;
    }

    public int getCol(String fileNameFrom) throws Exception {
        FileReader reader = new FileReader(fileNameFrom);
        BufferedReader br = new BufferedReader(reader);
        String[] ss = br.readLine().split("\\s+");
        Col = Integer.valueOf(ss[1]);
        return Col;
    }

    //set行列
    public void setRow(int row) {
        Row = row;
    }

    public void setCol(int col) {
        Col = col;
    }

    //读取数据
    public Matrix read(String fileNameFrom) throws Exception {
        System.out.println("开始读取数据.......");
        Matrix result = new Matrix();
```

```
        //int[][] data;
        try {
            FileReader reader = new FileReader(fileNameFrom);
            BufferedReader br = new BufferedReader(reader);
            String[] ss = br.readLine().split("\\s+");
            result.Row = Integer.valueOf(ss[0]);
            result.Col = Integer.valueOf(ss[1]);
            System.out.println(result.Row + "行");
            System.out.println(result.Col + "列");
            String[] sss = null;
            result.data = new double[result.Row][result.Col];
            for (int i = 0; i < result.Row; i++) {
                sss = br.readLine().split("\\s+");
                for (int j = 0; j < result.Col; j++) {
                    result.data[i][j] = Double.valueOf(sss[j]);
                }
            }
            br.close();
            return result;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    //写出数据
    public void write(Matrix result, String fileNameTo) throws Exception {
        File file = new File(fileNameTo);
        FileWriter writer = new FileWriter(file);
        BufferedWriter br = new BufferedWriter(writer);
        int Row = result.Row;
        int Col = result.Col;
        double[][] data = result.data;
        br.write(Row + " " + Col);
        br.newLine();
        for (int i = 0; i < Row; i++) {
            String str = "";
            for (int j = 0; j < Col; j++) {
                str += data[i][j] + "\t";
            }
            br.write(str);
            br.newLine();
            br.flush();
        }
        System.out.println("结果已经存储。");
    }


    //加法运算
    public Matrix add(Matrix first, Matrix second) throws Exception {

        Matrix firstm = new Matrix();
        firstm = first;
        Matrix secondm = new Matrix();
        secondm = second;
```

```
        Matrix result = new Matrix();
        try {
            if (first.Row != second.Row || second.Col != second.Col)
                throw new Exception("不符合矩阵加法的运算规则! ");
        } catch (Exception e) {
            e.getMessage();
        }
        result.Row = firstm.Row;
        result.Col = firstm.Col;
        result.data = new double[result.Row][result.Col];
        for (int i = 0; i < result.Row; i++) {
            for (int j = 0; j < result.Col; j++) {
                result.data[i][j] = firstm.data[i][j] + secondm.data[i][j];
            }
        }

        System.out.println("已经完成加法运算");
        return result;

    }

    //减法运算
    public Matrix sub(Matrix first, Matrix second) throws Exception {

        Matrix firstm = new Matrix();
        firstm = first;
        Matrix secondm = new Matrix();
        secondm = second;
        try {
            if (first.Row != second.Row || second.Col != second.Col)
                throw new Exception("不符合矩阵减法的运算规则! ");
        } catch (Exception e) {
            e.getMessage();
        }

        Matrix result = new Matrix();
        result.Row = firstm.Row;
        result.Col = firstm.Col;
        result.data = new double[result.Row][result.Col];
        for (int i = 0; i < result.Row; i++) {
            for (int j = 0; j < result.Col; j++) {
                result.data[i][j] = firstm.data[i][j] - secondm.data[i][j];
            }
        }
        System.out.println("已经完成减法运算");
        return result;

    }

    //乘法运算
    public Matrix mul(Matrix first, Matrix second) throws Exception {

        Matrix firstm = new Matrix();
        firstm = first;
```

```
        Matrix secondm = new Matrix();
        secondm = second;
        Matrix result = new Matrix();

        try {
            if (firstm.Col != secondm.Row)
                throw new Exception("不符合矩阵乘法的运算规则! ");
        } catch (Exception e) {
            e.getMessage();
        }

        result.Row = firstm.Row;
        result.Col = second.Col;
        result.data = new double[result.Row][result.Col];
        for (int i = 0; i < result.Row; i++) {
            for (int j = 0; j < result.Col; j++) {
                double sum = 0;
                for (int k = 0; k < result.Row; k++) {
                    sum += first.data[i][k] * second.data[k][j];
                }
                result.data[i][j] = sum;
                //System.out.println(sum);
            }
        }
        System.out.println("已经完成乘法运算");
        return result;
    }

    public Matrix div(Matrix first, Matrix second) throws Exception {
        Matrix firstm = new Matrix();
        firstm = first;
        Matrix secondm = new Matrix();
        secondm = second;
        Matrix result = new Matrix();
        try {
            if (first.Row != first.Col && second.Col != second.Col)
                throw new Exception("不是方阵,不符合矩阵除法的必要条件! ");
        } catch (Exception e) {
            e.getMessage();
        }
        int a = firstm.Row;
        int b = secondm.Row;
        double[][] aim = new double[b][b];
        aim = secondm.data;
        double c = this.getMartrixResult(aim);
        try {
            if (c < 1e-6)
                throw new Exception("行列式的值为0,不符合矩阵除法的必要条件! ");
        } catch (Exception e) {
            e.getMessage();
        }
        result.Row = firstm.Row;
        result.Col = firstm.Col;
        result.data = new double[result.Row][result.Col];
        result.data = this.getReverseMartrix(aim);
```

```
        result = this.mul(firstm, result);
        System.out.println("已经完成除法运算");
        return result;

    }

    public double getMartrixResult(double[][] data) {
        if (data.length == 2) {
            return data[0][0] * data[1][1] - data[0][1] * data[1][0];
        }
        double result = 0;
        int num = data.length;
        double[] nums = new double[num];
        for (int i = 0; i < data.length; i++) {
            if (i % 2 == 0) {
                nums[i] = data[0][i] * this.getMartrixResult(getConfactor(data,
1, i + 1));
            } else {
                nums[i] = -data[0][i] * this.getMartrixResult(getConfactor(data,
1, i + 1));
            }
        }
        for (int i = 0; i < data.length; i++) {
            result += nums[i];
        }
        return result;
    }

    public double[][] getConfactor(double[][] data, int h, int v) {
        int H = data.length;
        int V = data[0].length;
        double[][] newdata = new double[H - 1][V - 1];
        for (int i = 0; i < newdata.length; i++) {
            if (i < h - 1) {
                for (int j = 0; j < newdata[i].length; j++) {
                    if (j < v - 1) {
                        newdata[i][j] = data[i][j];
                    } else {
                        newdata[i][j] = data[i][j + 1];
                    }
                }
            } else {
                for (int j = 0; j < newdata[i].length; j++) {
                    if (j < v - 1) {
                        newdata[i][j] = data[i + 1][j];
                    } else {
                        newdata[i][j] = data[i + 1][j + 1];
                    }
                }
            }
        }
        return newdata;
    }

    public double[][] getReverseMartrix(double[][] data) {
```

```
        double[][] newdata = new double[data.length][data[0].length];
        double A = this.getMartrixResult(data);
        for (int i = 0; i < data.length; i++) {
            for (int j = 0; j < data[0].length; j++) {
                if ((i + j) % 2 == 0) {
                    newdata[i][j] = this.getMartrixResult(this.getConfactor(data,
i + 1, j + 1)) / A;
                } else {
                    newdata[i][j] = -
this.getMartrixResult(this.getConfactor(data, i + 1, j + 1)) / A;
                }

            }
        }
        newdata = this.trans(newdata);
        return newdata;
    }

    public double[][] trans(double[][] newdata) {
        // TODO Auto-generated method stub
        double[][] newdata2 = new double[newdata[0].length][newdata.length];
        for (int i = 0; i < newdata.length; i++)
            for (int j = 0; j < newdata[0].length; j++) {
                newdata2[j][i] = newdata[i][j];
            }
        return newdata2;
    }
}
```