I/O Operation

1 Introduction

Through this training, to master binary I/O operation and text I/O operation.

1.1 Evaluation

- Code Correctness: 60%
- Experimental Report: 40%

1.2 Knowledge Points

- InputStream and OutputStream
- Binary I/O Operation
- Text I/O Operation

2 Demonstration

2.1 Binary I/O Operation

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
class Test {
    public static void main(String args[]){
        InputStream is = null;
        OutputStream os = null;
        try {
            is = new FileInputStream("C:\\data.txt");
            os = new FileOutputStream("C:\\data_copy.txt");
            int c;
            // Read display and copy the data to a new file
            while((c = is.read()) != -1) {
                System.out.printf("%-2s", (char)c);
                os.write(c);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            // Close streams
            try {
                if(is != null) is.close();
                if(os != null) os.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
}
```

2.2 Scanner and PrintWriter

Create a `Scanner` object with `FileInputStream`, then we can read text data from a file using `next..()` methods. We can create a `PrintWriter` object and write data to a file using `print..()` methods.

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
class Test {
    public static void main(String args[]){
        Scanner sc = null;
        PrintWriter pw = null;
        try {
            // Create a scanner object and set a file inputstream
            sc = new Scanner(new FileInputStream("C:\\data.txt"));
            // Create a printwriter object for file writing
            pw = new PrintWriter("C:\\data_copy.txt");
            while(sc.hasNextLine()) {
                String line = sc.nextLine();
                System.out.println(line);
                pw.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            sc.close();
            pw.close();
        }

    }
}
```

2.3 Reader and Writer

Two classes `Reader` and `Writer` and their sub-classes are used for reading and writing character (The basis of the Text) streams. They are designed according to the `decorator` pattern, so they can choose matching methods according to different application scenarios.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.nio.charset.Charset;
class Test {
    public static void main(String args[]){
        BufferedReader br = null;
        BufferedWriter bw = null;
        try {
            // Create a BufferedReader object for reading by line
            FileInputStream fis = new FileInputStream("C:\\data.txt");
            br = new BufferedReader(new InputStreamReader(fis,
Charset.forName("UTF-8")));
            // Create a BufferedWriter for writing
            FileOutputStream fos = new FileOutputStream("C:\\data_copy.txt");
            bw = new BufferedWriter(new OutputStreamWriter(fos,
Charset.forName("UTF-8")));
            String line = null;
            while((line = br.readLine()) != null) {
                System.out.println(line);
                bw.write(line + "\n");
            }
        } catch (IOException e) {
            System.out.println("Reading or Writing Error!");
        } finally {
            try {
                if(br != null)  br.close();
                if(bw != null)  bw.close();
            } catch (IOException e) {
                System.out.println("Stream Close Error!");
            }
        }

    }
}
```

3 Experiment Content

3.1 Geometries Sorting

Read geometries from a data file and sort them by area, then output the sorting resluts to a new file. We mainly consider three geometries: circle, rectangle and triangle. The format of the data file is as follows:

```
id, type, [attribute1[attribute2[..]]]
001, circle, 1.2
002, rectangle, 1.0, 2.0
003, triangle, 2.0, 3.0, 3.0
```

The format of the result file is as follows:

```
id, type, area
002, circle, 1.2
003, rectangle, 1.6
001, triangle, 1.8
```

1.Define the GeometricObject class which implements the Comparable interface.

```
public abstract class GeometricObject implements Comparable<GeometricObject>{
    public abstract double getArea();
    @Override
    public int compareTo(GeometricObject g) {
        //-1, 0, or 1 as the area if this geometry is less than, equal to, or
greater than the specified geometry.
    }
}
```

2.Define three classes Circle, Triangle and Rectangle.

```
public class Circle extends GeometricObject{
    private int id;
    private double radius;

    public Circle(int id) { this.id = id;}
    public Circle(int id, double r) {
        this.id = id;
        radius = r;
    }

    @Override
    public double getArea() {
        // Implement
    }

    // Implement getter and setter methods

    // Return information format: "id,circle,area"
    @Override
    public String toString() {
        // Implement
    }
}
```

```
public class Rectangle extends GeometricObject{
    private int id;
    private double width;
    private double height;
    public Rectangle(int id) { this.id = id;}
    public Rectangle(int id, double width, double height) {
        this.id = id;
        this.width = width;
        this.height = height;
    }
    @Override
    public double getArea() {
        // Implement
    }

    // Implement getter and setter methods

    // Return information format: "id,rectangle,area"
    public String toString() {
        // Implement
    }
}
```

```
public class Triangle extends GeometricObject{
    private int id;
    private double a;
    private double b;
    private double c;
    public Triangle(int id) {   this.setId(id);}
    public Triangle(int id, double a, double b, double c){
        this.setId(id);
        this.setA(a);
        this.setB(b);
        this.setC(c);
    }
    @Override
    public double getArea() {
        // Implement
    }

    // Implement getter and setter methods

    // Return information format: "id,triangle,area"
    public String toString() {
        // Implement
    }
}
```

3.Sort and export geometric objects to a file.

```
// Create a geometry by a string information like "1001, circle, 1.2"
public static GeometricObject CreateGeoByString(String s) {
    GeometricObject g = null;
    String[] items = s.split(",");
    int id = Integer.parseInt(items[0]);
    String type = items[1];
    switch(type) {
        case "circle": {
            double r = Double.parseDouble(items[2]);
            g = new Circle(id, r);
        } break;
        case "rectangle": {
            // Implement
        } break;
        case "triangle": {
            // Implement
        } break;
    }
    return g;
}
```

```
    // Sort geometries from source file and write the sorted results to the target
    file
    public static void sort(String dataPath, String resultsPath) {
        Scanner input = null;
        PrintWriter pw = null;
        try {
            input = new Scanner(new FileInputStream(dataPath));
            pw = new PrintWriter(resultsPath);

            ArrayList<GeometricObject> geoList = new ArrayList<GeometricObject>();
            // Reader all lines from the data file and create geometries

            // Sort the list by Collections.sort()

            // Output the sorted geometry list to target file

        } catch (IOException e) {
            System.out.println("Reading or Writing Error!");
        } finally {
            input.close();
            pw.close();
        }
    }
```

4.Test the sort method.

```
    public static void main(String args[]){
        sort("data.txt", "results.txt");
    }
```

5.Check the sorted file.

```
10146,rectangle,12.83
19771,triangle,12.86
16332,triangle,12.9
18675,rectangle,12.9
15593,circle,12.94
10295,triangle,12.98
14251,triangle,13.0
15088,rectangle,13.0
17008,triangle,13.0
16468,rectangle,13.02
19059,triangle,13.02
16762,triangle,13.02
19951,triangle,13.04
10942,triangle,13.05
17955,triangle,13.05
18076,circle,13.07
18782,circle,13.07
16872,triangle,13.08
12312,rectangle,13.08
```

4 Experiment Report Requirements

4.1 Think and answer the question

(1) What is the difference between `System.in` and `new FileInputStream(..)` as the argument of `new Scanner(..)`.

(2) How can we use the `Collections.sort()` method to sort all geometric objects?

(3) If we do not close `OutputStream`, will it affect the output file content?

(4) Other experience.

4.2 Experiment report content

(1) Answer the above questions.

(2) All codes.

4.3 Submission method

(1) Upload the report by ftp:(Address:ftp://172.18.5.102; UserName:wangxiaomeng; Password: wangxiaomeng)

(2) File name format: StudentID+Name. For example, `20191234小明.docx`

4.4 Other Instructions

You can obtain experiment course resources through the web platform (URL: https://www.lanqiao.cn; InvitationCode: `ZF0XA4Y1`)