

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering | | | | | **Class:** | 2020 |
|  |  | | | | |  |  |
| **Student name:** | 温长锟 | | **Student No.:** | | 222020321062106 | | |
|  |  | |  | |  | | |
| **Teacher:** | 詹成 | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp6 IO Operation | | |
| Time | 2021.12.9 | Type | □Verification □Design □Synthetical |
| 1. Answer the questions  (1) What is the difference between System.in and new FileInputStream(..) as the argument of new Scanner(..).  System. In reads standard input device data (data is obtained from standard input, usually keyboard), and its data type is InputStream.  FileInputStream stream is called file byte input stream, which means to read file data in the form of bytes, such as reading pictures and videos  (2) How can we use the Collections.sort() method to sort all geometric objects?  This object is required to implement the Comparable < T > interface. Then override the ompareto () method inside.  (3) If we do not close OutputStream, will it affect the output file content?  You must absolutely close the target file to ensure that all caches from the JVM to the OS are flushed and that the file can be used by the reader.   1. Other experience.   Stream is used to read and write transmission data. Input and output are for the program itself, but the program needs to read in the data and use the read in stream. The program needs to save the data output and use the output stream.  Stream is equivalent to a variety of different pipelines to transmit data. According to read in and output, it is divided into input stream and output stream. It is divided into byte stream and character stream according to the type of transmission. According to the relationship between pipeline and pipeline, it can be divided into node flow and processing flow.  OutputStream is a byte output stream in Java, which can be used to output files or strings to new files.  OutputStream class is used for output in byte stream and InputStream class is used for input.  2. All Codes  **(1).GeometricObject.java**  public abstract class GeometricObject implements Comparable<GeometricObject> {  public abstract double getArea();  @Override  public int compareTo(GeometricObject geo) {  return Double.*compare*(this.getArea(), geo.getArea());  } }  **(2)Circle.java**  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() {  return Math.*PI* \* radius \* radius;  }   // Implement getter and setter methods  // Return information format: "id,circle,area"  public int getId() {  return id;  }   public double getRadius() {  return radius;  }   public void setId(int id) {  this.id = id;  }   public void setRadius(double radius) {  this.radius = radius;  }   @Override  public String toString() {  return id + ",Circle," + getArea();  } }  **(3)Rectangle.java**  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() {  return width \* height;  }  // Implement getter and setter methods   public int getId() {  return id;  }   public double getWidth() {  return width;  }   public double getHeight() {  return height;  }   public void setId(int id) {  this.id = id;  }   public void setWidth(double width) {  this.width = width;  }   public void setHeight(double height) {  this.height = height;  }   // Return information format: "id,rectangle,area"  public String toString() {  return id + ",Rectangle," + getArea();  } }  **(4)Triangle.java**  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() {  double p = (a + b + c) / 2;  return Math.*sqrt*(p \* (p - a) \* (p - b) \* (p - c));  }  // Implement getter and setter methods   public int getId() {  return id;  }   public double getA() {  return a;  }   public double getB() {  return b;  }   public double getC() {  return c;  }   public void setId(int id) {  this.id = id;  }   public void setA(double a) {  this.a = a;  }   public void setB(double b) {  this.b = b;  }   public void setC(double c) {  this.c = c;  }   // Return information format: "id,triangle,area"  public String toString() {  return id + ",Triangle," + getArea();  } }   1. **IO.java**   import java.io.FileInputStream; import java.io.IOException; import java.io.PrintWriter; import java.util.ArrayList; import java.util.Collections; import java.util.Scanner;  public class IO {  public static void main(String[] args) {  *sort*("C:/Users/温长锟/Desktop/Exp6/src/data.txt", "C:/Users/温长锟/Desktop/Exp6/src/Result.txt");  }   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": {  double width = Double.*parseDouble*(items[2]);  double height = Double.*parseDouble*(items[3]);  g = new Rectangle(id, width, height);  }  break;  case "triangle": {  double a = Double.*parseDouble*(items[2]);  double b = Double.*parseDouble*(items[3]);  double c = Double.*parseDouble*(items[4]);  g = new Triangle(id, a, b, c);  }  break;  }  return g;  }   public static void sort(String dataPath, String resultsPath) {  try (Scanner input = new Scanner(new FileInputStream(dataPath));  PrintWriter pw = new PrintWriter(resultsPath)) {  ArrayList<GeometricObject> geoList = new ArrayList<GeometricObject>();  while (input.hasNextLine()) {  geoList.add(*CreateGeoByString*(input.nextLine()));  }  Collections.*sort*(geoList);  for (int i = 1; i != geoList.toArray().length; i++) {  pw.println(geoList.get(i).toString());  }  // 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!");  e.printStackTrace();  }  } }  **Experimental process** | | | |
| **Result** | | | |

|  |  |  |
| --- | --- | --- |
| Evaluation | Code Correctness (60%): |  |
| Experience (40%): |  |
| Score： | |