

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering | | | | | **Class:** | 2019 |
|  |  | | | | |  |  |
| **Student name:** | Fu Ruoxuan | | **Student No.:** | | 222019321062060 | | |
|  |  | |  | |  | | |
| **Teacher:** | Wang Xiaomeng | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp6 IO Operation | | |
| Time | 2020.12.9 | 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(..).  Java.util.Scanner will get the command input. When a Scanner is created with new Scanner(System.in), the console waits for input until it finishes hitting the Enter key, passing the input to the Scanner as the object to be scanned. When a Scanner is created with new Scanner(new FileInputStream(path)),the text in the path will be read .  (2) How can we use the Collections.sort() method to sort all geometric objects?  Collections.sort() is a static method of a tool class that sorts list types. The arguments it requires are built with the idea of generics in mind, so you can pass in objects of your own type. So just follow the tool class specification to implement your own class and override the CompareTo() method to sort all the objects using Collection, sort().  (3) If we do not close OutputStream, will it affect the output file content?  If you don't close the output stream, the IO resources allocated to it will be occupied by it all the time, resulting in wasted resources and memory usage.  (4) Other experience.   * The java.lang.Double.compare() method compares two specified values of double. Returning integer values with the same sign will return the integer by calling. It can be used instead of writing his own compare(). * Java provides a syntactic structure for try that enables automatic closure of streams. The syntax is as follows: try(stream object created). Note that in the parentheses after the try statement, you need to write the complete statement that creates the stream object, with a semicolon. Also, the statement in parentheses can only be one line; you cannot split the statement into two lines of statements for declaring and creating objects.   2. All Codes  GeometricObject.java   1. **package** exp6; 3. **public** **abstract** **class** GeometricObject **implements** Comparable<GeometricObject>{ 4. **public** **abstract** **double** getArea(); 5. **public** **int** compareTo(GeometricObject geo) 6. { 7. **return** Double.compare(**this**.getArea(), geo.getArea()); 8. } 10. }   Circle.java   1. **package** exp6; 3. **class** Circle **extends** GeometricObject{ 4. **private** **int** id; 5. **private** **double** radius; 6. **public** Circle(**int** id) { **this**.id = id;} 7. **public** Circle(**int** id, **double** r) { 8. **this**.id = id; 9. radius = r; 10. } 11. @Override 12. **public** **double** getArea() { 13. **return** Math.PI\*radius\*radius; 14. } 15. // Implement getter and setter methods 16. // Return information format: "id,circle,area" 17. **public** **int** getId() { 18. **return** id; 19. } 21. **public** **double** getRadius() { 22. **return** radius; 23. } 25. **public** **void** setId(**int** id){ 26. **this**.id = id; 27. } 29. **public** **void** setRadius(**double** radius) { 30. **this**.radius = radius; 31. } 33. @Override 34. **public** String toString() { 35. **return** id+",Circle,"+getArea(); 36. } 37. }   Triangle.java   1. **package** exp6; 3. **class** Triangle **extends** GeometricObject{ 4. **private** **int** id; 5. **private** **double** a; 6. **private** **double** b; 7. **private** **double** c; 8. **public** Triangle(**int** id) { **this**.setId(id);} 9. **public** Triangle(**int** id, **double** a, **double** b, **double** c){ 10. **this**.setId(id); 11. **this**.setA(a); 12. **this**.setB(b); 13. **this**.setC(c); 14. } 15. @Override 16. **public** **double** getArea() { 17. **double** p = (a+b+c)/2; 18. **return** Math.sqrt(p\*(p-a)\*(p-b)\*(p-c)); 19. } 20. // Implement getter and setter methods 22. **public** **int** getId() { 23. **return** id; 24. } 26. **public** **double** getA() { 27. **return** a; 28. } 30. **public** **double** getB() { 31. **return** b; 32. } 34. **public** **double** getC() { 35. **return** c; 36. } 38. **public** **void** setId(**int** id) { 39. **this**.id = id; 40. } 42. **public** **void** setA(**double** a) { 43. **this**.a = a; 44. } 46. **public** **void** setB(**double** b) { 47. **this**.b = b; 48. } 50. **public** **void** setC(**double** c) { 51. **this**.c = c; 52. } 54. // Return information format: "id,triangle,area" 55. **public** String toString() { 56. **return** id+",Triangle,"+getArea(); 57. } 58. }   Rectangle.java   1. **package** exp6; 3. **class** Rectangle **extends** GeometricObject{ 4. **private** **int** id; 5. **private** **double** width; 6. **private** **double** height; 7. **public** Rectangle(**int** id) { **this**.id = id;} 8. **public** Rectangle(**int** id, **double** width, **double** height) { 9. **this**.id = id; 10. **this**.width = width; 11. **this**.height = height; 12. } 13. @Override 14. **public** **double** getArea() { 15. **return** width\*height; 16. } 17. // Implement getter and setter methods 19. **public** **int** getId() { 20. **return** id; 21. } 23. **public** **double** getWidth() { 24. **return** width; 25. } 27. **public** **double** getHeight() { 28. **return** height; 29. } 31. **public** **void** setId(**int** id) { 32. **this**.id = id; 33. } 35. **public** **void** setWidth(**double** width) { 36. **this**.width = width; 37. } 39. **public** **void** setHeight(**double** height) { 40. **this**.height = height; 41. } 43. // Return information format: "id,rectangle,area" 44. **public** String toString() { 45. **return** id+",Rectangle,"+getArea(); 46. } 47. }   IOTest.java   1. **package** exp6; 3. **import** java.io.FileInputStream; 4. **import** java.io.IOException; 5. **import** java.io.PrintWriter; 6. **import** java.util.ArrayList; 7. **import** java.util.Collections; 8. **import** java.util.Scanner; 10. **public** **class** IOTest{ 11. **public** **static** **void** main(String[]args){ 12. sort("C:/Users/21558/IdeaProjects/untitled/src/exp6/data.txt", "C:/Users/21558/IdeaProjects/untitled/src/exp6/results.txt"); 13. } 14. **public** **static** GeometricObject CreateGeoByString(String s) { 15. GeometricObject g = **null**; 16. String[] items = s.split(","); 17. **int** id = Integer.parseInt(items[0]); 18. String type = items[1]; 19. **switch**(type) { 20. **case** "circle": { 21. **double** r = Double.parseDouble(items[2]); 22. g = **new** Circle(id, r); 23. } **break**; 24. **case** "rectangle": { 25. **double** width = Double.parseDouble(items[2]); 26. **double** height = Double.parseDouble(items[3]); 27. g = **new** Rectangle(id,width,height); 28. } **break**; 29. **case** "triangle": { 30. **double** a = Double.parseDouble(items[2]); 31. **double** b = Double.parseDouble(items[3]); 32. **double** c = Double.parseDouble(items[4]); 33. g = **new** Triangle(id,a,b,c); 34. } **break**; 35. } 36. **return** g; 37. } 38. **public** **static** **void** sort(String dataPath, String resultsPath) { 39. **try**( Scanner input = **new** Scanner(**new** FileInputStream(dataPath)); 40. PrintWriter pw = **new** PrintWriter(resultsPath)) { 41. ArrayList<GeometricObject> geoList = **new** ArrayList<GeometricObject>(); 42. **while** (input.hasNextLine()) { 43. geoList.add(CreateGeoByString(input.nextLine())); 44. } 45. Collections.sort(geoList); 46. **for** (**int** i = 1;i!= geoList.toArray().length;i++){ 47. pw.println(geoList.get(i).toString()); 48. } 49. // Reader all lines from the data file and create geometries 50. // Sort the list by Collections.sort() 51. // Output the sorted geometry list to target file 52. } **catch** (IOException e) { 53. System.out.println("Reading or Writing Error!"); 54. e.printStackTrace(); 55. } 56. } 57. } | | | |

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