

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering | | | | | **Class:** | 2019 |
|  |  | | | | |  |  |
| **Student name:** | 冯春霖 | | **Student No.:** | | 222019321062074 | | |
|  |  | |  | |  | | |
| **Teacher:** | Wang Xiaomeng | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp6 IO Operation | | |
| Time | 2020.12.7 | 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(..).  A: System.in for standard input via commandline and FileInputStream for stream input via file.  (2) How can we use the Collections.sort() method to sort all geometric objects?  A: Collection.sort() is a static method so we can use class name to call the method directly.  (3) If we do not close OutputStream, will it affect the output file content?  A: It may keep consuming computer's memory, but today's modern IDEs usually close it automatically.  (4) Other experience.  A: Learned the different ways of input and output and their advantages and disadvantages.  2. All Codes  Code of GeometricObject class:  public abstract class GeometricObject implements Comparable<GeometricObject>  {  public abstract double getArea();  public int compareTo(GeometricObject g)  {  if (this.getArea() > g.getArea())  return 1;  else if (this.getArea() == g.getArea())  return 0;  else  return -1;  }  }  Code of Circle class:  package week14;  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;  this.radius = r;  }    public double getRadius()  {  return radius;  }    public void setRadius(double \_r)  {  radius = \_r;  }  public double getArea()  {  return (radius \* radius \* Math.***PI***);  }    *@Override*  public String toString()  {  return id + ",circle," + this.getArea();  }  }  Code of Rectangle class:  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;  }    public int getId()  {  return id;  }    public void setId(int id)  {  this.id = id;  }    public double getWidth()  {  return width;  }    public void setWidth(double width)  {  this.width = width;  }    public double getHeight()  {  return height;  }    public void setHeight(double height)  {  this.height = height;  }  *@Override*  public double getArea()  {  return width \* height;  }    public String toString()  {  return id + ",rectangle," + this.getArea();  }  }  Code of Triangle class:  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);  }    public int getId()  {  return id;  }    public void setId(int id)  {  this.id = id;  }    public double getA()  {  return a;  }    public void setA(double a)  {  this.a = a;  }    public double getB()  {  return b;  }    public void setB(double b)  {  this.b = b;  }    public double getC()  {  return c;  }    public void setC(double c)  {  this.c = c;  }    *@Override*  public double getArea()  {  double p = (a + b + c) / 2;  return Math.*sqrt*(p \* (p - a) \* (p - b) \* (p - c));  }  // Return information format: "id,triangle,area"  public String toString()  {  return id + ",triangle," + this.getArea();  }  }  The output file after the program is run: | | | |

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