

**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 | Exp5 Exception Handling | | |
| Time  2021.12.2 |  | Type | □Verification □Design □Synthetical |
| 1. Answer the questions  (1) Under what conditions will the finally block be executed?  After entering the try block, the program will execute the contents of the finally block.  Unless System. exit (0) is called  (2) What is the difference between the keyword throw and throws?  **Throw** keyword:  Used inside the method  An exception object is thrown (it must be a subclass of Exception / RuntimeException). Throw can only throw one exception object at a time.  When a runtime exception is thrown, it does not need to be processed and is directly handed over to the JVM for processing  When a compile time exception is thrown, you need to handle the exception. There are two methods: throw and try.. catch ()  **Throws** keyword:  After using the method, a method can throw multiple exception objects at one time  The exception thrown is handled by the caller of the method  (3) Can the main method declare an exception?  The main method can declare exceptions. The exception in the main function goes to the default exception handler for the thread, which still exists.  (4) Other experience.  RuntimeException  At runtime, if the divisor is 0 and the array subscript is out of bounds, it is frequent and troublesome. If the declaration or capture is displayed, it will have a great impact on the readability and running efficiency of the program. Therefore, the system automatically detects them and gives them to the default exception handler. Of course, if you have processing requirements, you can also display and capture them.  2. All Codes  **Exp1**.integerInput  import java.util.InputMismatchException; import java.util.Scanner;  public class integerInput {   public static void main(String[] args) {  *sumTwoIntegers*();  }   private static void sumTwoIntegers() {  Scanner input = null;  try {  input = new Scanner(System.*in*);  System.*out*.print("input num1: ");  int num1 = input.nextInt();  System.*out*.print("input num2: ");  int num2 = input.nextInt();  System.*out*.printf("%d+%d=%d", num1, num2, num1 + num2);  } catch (InputMismatchException e) {  System.*out*.println("Input does not match the integer type, please enter again!");  *sumTwoIntegers*();  } finally {  input.close();  }  }  }  **Process screenshot**：    **Exp2**.Triangle  import java.util.Scanner;  public class mainTriangle {  public static void main(String[] args) {  Scanner input = new Scanner(System.*in*);  System.*out*.println("输入边1：");  double side1 = input.nextDouble();  System.*out*.println("输入边2");  double side2 = input.nextDouble();  System.*out*.println("输入边3");  double side3 = input.nextDouble();  try {  Triangle t = new Triangle(side1, side2, side3);  System.*out*.println("成功输入");  System.*out*.println(t.toString());  System.*out*.println("面积：" + t.getArea());  System.*out*.println("周长：" + t.getPerimeter());  } catch (IllegalTriangleException e) {  e.printStackTrace();  }  } }  class Triangle {  private double side1,side2,side3;   Triangle() {  side1 = 1.0;  side2 = 1.0;  side3 = 1.0;  }   Triangle(double side1, double side2, double side3) throws IllegalTriangleException {  this.side1 = side1;  this.side2 = side2;  this.side3 = side3;  if (side1 <= 0 || side2 <= 0 || side3 <= 0)  throw new IllegalTriangleException("有一边小于0");  else if (side1 + side3 <= side2)  throw new IllegalTriangleException("side1 + side3 <= side2");  else if (side1 + side2 <= side3)  throw new IllegalTriangleException("side1 + side2 <= side3");  else if (side2 + side3 <= side1)  throw new IllegalTriangleException("side2 + side3 <= side1");  }   public double getSide1() {  return side1;  }   public double getSide2() {  return side2;  }   public double getSide3() {  return side3;  }   public double getArea() {  return 0.25 \* Math.*sqrt*((side1 + side2 + side3) \* (side1 + side2 - side3) \* (side1 + side3 - side2) \* (side2 + side3 - side1));  }   public double getPerimeter() {  return side1 + side2 + side3;  }   public String toString() {  return "side1 = " + side1 + " side2 = " + side2 + " side3 =" + side3;  } }  class IllegalTriangleException extends Exception {  public IllegalTriangleException() {  super();  }   public IllegalTriangleException(String str) {  super(str);  } }  **Process screenshot**        **Operation results**.  **EXP1.**      **EXP2** | | | |

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