

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | JAVA Programming | | | | | | |
|  |  | | | | | | |
| **Semester:** | 2-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering Class | | | | | **Class:** | 2018 |
|  |  | | | | |  |  |
| **Student name:** | Song Xingjian | | **Student No.:** | | 222018321062006 | | |
|  |  | |  | |  | | |
| **Teacher:** | 王晓蒙 | | | | | | |

School of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exception handling | | |
| Time | 2019.11.1 | Type | □Verification □Design □Synthetical |
| **1. Objective**  Master the methods of exception handling in Java language, define exception classes, writes exception handling procedures to capture abnormal events.  **2. Requirement**  Programming the following exercise  **Problem 1:**      **Problem 2:**      **3. Content and design of experiment(main content, operations, algorithm description or code of program)**  **Problem 1：**  **Solutions:**   1. 建立异常检查函数inputInt()； 2. 先进行输入尝试，如果不行则进行异常处理，输出“Mismatch!”，并再次输入； 3. 输入成功后关闭输入流，并将输入值返回。   **Codes:**  **import** java.util.Scanner;  **import** java.util.InputMismatchException;  **public** **class** Test\_InputMismatchException {  **public** **static** **void** main(String[] args) {  System.***out***.printf("My input value: %d", *inputInt*());  }    //inputInt异常检查函数  **public** **static** **int** inputInt() {  Scanner input = **new** Scanner(System.***in***);  System.***out***.print("Enter an integer: ");  **try** {  **return** input.nextInt();  }**catch**(InputMismatchException e) {  //如果符合异常，则输出Mismatch!  System.***out***.println("Mismatch!");  **return** *inputInt*();  }**finally** {  input.close(); //关闭input  }  }  }  **Results:**    **Summary:**   1. 之前的异常处理是用一个死循环和break构成的，今天学习到了更加正规的异常处理，使用起来也更加方便； 2. 关键在于try和catch关键词的使用，用try进行尝试，发现异常后进行catch中的语句，成功后运行finally的语句。   **Problem 2：**  **Solutions:**     1. 建立了legalTriangle包，内含4个类，其中一个含主函数为main\_Triangle类，另外3个分别为几何图形GeomatricObject类，三角形类Triangle类，异常处理IllegalTriangleExcepion类； 2. Triangle类是GeomatricObject类的子类，完成关于三角形的三边初始化、三角形周长和边长的计算； 3. IllegalTriangleExcepion类Excepion类的子类，完成异常处理；   **Codes:**   1. **main\_Triangle.java**   **package** legalTriangle;  **import** java.util.Scanner;  **import** legalTriangle.IllegalTriangleException;  **import** legalTriangle.Triangle;  **public** **class** main\_Triangle {  **public** **static** **void** main(String[] args) {  System.***out***.println("Please input a triangle: ");  Triangle tri;// 创建三角形类的实例  tri = *inputTriangle*();// 输入并检验  // 输出结果  System.***out***.println(tri.toString());  System.***out***.printf("Perimeter: %.2f", tri.getPerimeter());  System.***out***.printf("\nArea: %.2f", tri.getArea());  }  // 输入三角形并检验  **public** **static** Triangle inputTriangle() {  **double** a, b, c;  // 输入三边长a，b，c  Scanner input = **new** Scanner(System.***in***);  System.***out***.print("a: ");  a = input.nextDouble();  System.***out***.print("b: ");  b = input.nextDouble();  System.***out***.print("c: ");  c = input.nextDouble();  **try** {// 尝试以啊，a，b，c创建三角形，若成功则将三角形返回  Triangle triangle\_temp = **new** Triangle(a, b, c);  **return** triangle\_temp;  } **catch** (IllegalTriangleException ite) {// 若不成功则调用异常函数，并再次输入  // ite.printStackTrace();  **return** *inputTriangle*();  } **finally** {// 结束异常检验后关闭输入流  input.close(); // 关闭input  }  }  }   1. **Triangle.java**   **package** legalTriangle;  **import** legalTriangle.IllegalTriangleException;  **public** **class** Triangle **extends** GeometricObject {  **private** **double** a, b, c;  **public** Triangle(**double** a, **double** b, **double** c) **throws** IllegalTriangleException {  **if** (a + b > c && b + c > a && a + c > b) {// 如果可以构成三角形，则赋值  **this**.a = a;  **this**.b = b;  **this**.c = c;  } **else** {// 如果不能构成三角形，则返回相应的提示语句  String message = "";  **if** (a + b <= c) {  message = "IllegalTriangleException:a+b<=c";  } **else** **if** (b + c <= a) {  message = "IllegalTriangleException:b+c<=a";  } **else** {  message = "IllegalTriangleException:a+c<=b";  }  // 主动将异常抛出  **throw** **new** IllegalTriangleException(message);  }  }  // 计算三角形面积并返回（海伦公式）  **public** **double** getArea() {  **double** p = 0.5 \* getPerimeter(); // 半周长  **return** Math.*sqrt*(p \* (p - a) \* (p - b) \* (p - c));  }  // 计算矩形周长并返回  **public** **double** getPerimeter() {  **return** (a + b + c);  }  // 转化输出语句  **public** String toString() {  String str;  str = "Triangle:side1 = " + a + "\tside2 = " + b + "\tside3 = " + c;  **return** str;  }  }   1. **IllegalTriangleExcepion.java**   **package** legalTriangle;  **public** **class** IllegalTriangleException **extends** Exception {  **public** IllegalTriangleException(String messagge) {  System.***out***.println(messagge);  System.***out***.println("\nPlease enter again!");  }  }   1. **GeomatricObject.java**   **package** legalTriangle;  **public** **class** GeometricObject {  **double** side1;  **double** side2;  **double** side3;  // 初始化默认构造函数（无参数），三边均为1  GeometricObject() {  side1 = 1;  side2 = 1;  side3 = 1;  }  // 初始化默认构造函数（有参数）  GeometricObject(**double** x, **double** y, **double** z) {  side1 = x;  side2 = y;  side3 = z;  }  }  **Results:**    **Summary:**   1. 再次练习try和catch关键词的使用，用try进行尝试创建一个三角形类的实例，若创建成功则返回，发现异常后进行catch中的语句，抛出异常跳转到异常处理函数IllegalTriangleExcepion输出异常提示语句，并再次提示输入，成功后运行finally的语句关闭input输入流； 2. 三角形类Triangle类中，计算三角形面积使用海伦公式比较方便； 3. IllegalTriangleExcepion需要继承Excepion类，否则会报错。 | | | |

|  |  |  |
| --- | --- | --- |
| Teacher’s  comments | content and design of experiment（A-E）： |  |
| operations, algorithm description or code of program（A-E）： |  |
| results（A-E）： |  |
| summary and analysis of experiment（A-E）： |  |
| Grade（A-E）： | |