第七章作业

2017221105003 黄文杰

1.Java语言中的异常处理是什么含义？

异常就是程序执行过程中出现的不正常现象。任何一个程序都可能出现异常，如果要打开的文件不存在、内存不够、数组访问超界等。Java作为一种面向对象语言，它设计了很多标准的异常处理类，采用throw-catch（抛出-捕获）方式，捕获并处理异常。Java使异常处理标准化，使程序设计思路更清楚，理解更容易。

2.在异常处理中由哪些部分组成，它们各有什么作用？

最基本的异常处理结构形式如下：

try {

正常程序段;

}

catch(异常类1 异常变量) {

与异常类1有关的处理程序段;

}

catch(异常类2 异常变量) {

与异常类2有关的处理程序段;

}

……

finally {

退出异常处理了程序段;

}

在这个结构中使用了三个关键字：try，catch，finally。try后大括号中的程序段称为保护代码，可能产生某种异常。如果这些异常是Java定义的标准异常，如FIleNotFoundException等，将由Java自动抛出这个类的一个对象。无论是哪种情况发生，在try{}中都不进行任何异常处理。紧接yry之后的是一系列catch语句，这才是处理异常的正确地方。catch与一个方法定义相似，需要一个参数，该参数必须是Throwable类或其子类的一个对象，或一个接口。catch的作用是当系统或应用程序抛出一个异常后，通过与catch的参数比较，如果相匹配，就执行catch方法中的异常处理程序段，否则与下一catch继续进行相匹配的比较。

3.在异常处理的catch语句中，异常类在安排次序是有什么要注意的？

catch检查匹配是顺序进行的，当程序中包含多条catch语句时，如果catch安排的顺序不当可能导致某些catch后的处理程序段永远不会被执行。例如如下程序段：

catch(IOException e) {

异常处理1;

}

catch(FileNotFoundException e) {

异常处理2;

}

如果程序产生了一个FileNotFoundException异常，在进行第一次匹配比较时，根据匹配的第二条含义，Java认为它们已经匹配，则执行异常处理1，而异常处理2永远不会被执行。所以在使用catch语句时，应该清楚所有参数所属类的层次，以便正确安排catch语句的顺序。如果出现这种错误，javac编译器会发出错误信息：catch no reached。

4.什么情况下必须使用throw语句？举例说明。

如果是应用程序自己设计的异常，必须使用throw语句抛出自定义的异常对象。

例如：

|  |
| --- |
| testThrow.java |
| import java.util.Scanner;  class IllegalInputException extends Exception {  private int grade;  IllegalInputException(int grade) {  this.grade = -1 \* grade;  }  int getGrade() {  return grade;  }  }  public class testThrow {  public static void main(String[] args) {  System.out.println("Input the grade:");  Scanner input = new Scanner(System.in);  int grade = input.nextInt();  try {  if(grade >= 0) {  System.out.println("Grade is " + grade);  }  else {  throw new IllegalInputException(grade);  }  }  catch(IllegalInputException e) {  System.out.println("Grade can not be a negative number.");  System.out.println("You maybe input this grade: " + e.getGrade());  }  }  } |
| 运行结果：  $ java testThrow  Input the grade:  -60  Grade can not be a negative number.  You maybe input this grade: 60 |

5.throws子句在什么位置使用，它表示什么含义？举例说明。

在异常处理时，抛出异常和处理异常可以在同一个方法中，但有时要求抛出异常的方法并不处理该异常，而是由调用该方法的另一方法来处理，那么这时可以使用throws语句给方法声明一个例外情况，其声明格式为：

方法 throws 异常类1, 异常类2, ...

例如：

|  |
| --- |
| testThrows.java |
| import java.util.Scanner;  class IllegalInputException extends Exception {  private int grade;  IllegalInputException() {  System.out.println("Grade can not be a negative number.");  }  }  class gradeInfo {  private int grade;  void setGrade(int grade) throws IllegalInputException {  if(grade < 0)  throw new IllegalInputException();  else  this.grade = grade;  }  int getGrade() {  return this.grade;  }  }  public class testThrows {  public static void main(String[] args) {  System.out.println("Input the grade:");  Scanner input = new Scanner(System.in);  gradeInfo g = new gradeInfo();  int grade = input.nextInt();  try {  g.setGrade(grade);  }  catch(IllegalInputException e) {    }  finally {  System.out.println("The grade is: " + g.getGrade());  }    }  } |
| 运行结果：  $ java testThrows  Input the grade:  9  The grade is: 9  $ java testThrows  Input the grade:  -99  Grade can not be a negative number.  The grade is: 0 |

6.编写一个含有ArithmrticException、IndexOutOfBoundsException和NullPointerException异常处理程序。

|  |
| --- |
| S7E6.java |
| class sample {  private String name;  sample(){}  String getName(){  return name;  }  }  public class S7E6 {  public static void main(String[] args) {  int[] array = new int[3];  int num=2;  for(int i=1; i<=3; i++) {  try {  System.out.println("Case" + i + ": ");  switch(i) {  case 1:  num /= 0;  break;  case 2:  array[3] = num;  break;  case 3:  new sample().getName().length();  break;  }  }  catch(ArithmeticException e) {  System.out.println(e);  }  catch(IndexOutOfBoundsException e) {  System.out.println(e);  }  catch(NullPointerException e) {  System.out.println(e);  }  finally {  System.out.println("Finish Case" + i + ".");  }  }  }  } |
| $ java S7E6  Case1:  java.lang.ArithmeticException: / by zero  Finish Case1.  Case2:  java.lang.ArrayIndexOutOfBoundsException: 3  Finish Case2.  Case3:  null  Finish Case3. |