Java中“ 异常机制 ”深入研究（二） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_Java_E4_B8_AD_E2_80_9C_c104_145216.htm 三、"异常"的处理方法 有两种方法处理"异常"：第一种如例2，将含有"异常"出口的方法直接放到try块中，然后由紧随其后的catch块捕捉。第二种是不直接监听捕捉被引用方法的"异常"，而是将这个"异常"关联传递给引用方法，同时监听捕捉工作也相应向上传递。 例3： int myMethod2(int dt) { int data = 0. try{ data = myMethod(dt) }catch(LowZeroException e){ System.out.println("发生数据错误！"). e.printStackTrace(). } return data. } int myMethod(int dt) throws LowZeroException { int data = isLegal(dt). // 此处引用isLegal()方法，但并没有捕捉它的"异常" return data. } 从上例中可以看到方法myMethod()与它引用的方法isLegal()产生的"异常"LowZeroException建立了关联，也就是完成了将"异常"关联的向上传递，此时的myMethod()方法体中虽然只有一个return返回语句，但它事实上同样有两种方式的函数出口，一种是由return返回的整形值，另一种则是返回方法名中的throws关键字所指的"异常类"的实例对象。相应的，监听捕捉的工作交给了上一层方法myMethod2()。同样的道理，myMethod2()也可以将"异常"通过throws的关联继续向上传递。这样的话，一旦一个"异常"被捕捉到时，这个"异常"必有一个传递路径，而如果我们在捕捉点的catch程序块中加入printStackTrace()方法，便能清楚的看到这个"异常"是怎样传递过来的。例如在例3如果有"异常"被捕捉到，e.printStackTrace()打印出来的结果将是： LowZeroException:

at Example.isLegal at Example myMethod at Example.myMethod2 at Example main 从上结果中我们可以看到，从LowZeroException"异常"产生点，即包含throw new LowZeroException().子句的方法开始，然后一直追溯到产生当前线程的方法（注意：printStackTrace()并不是追溯到捕捉点结束，而是到产生当前线程的方法结束）。"异常"产生点产生的LowZeroException"异常"对象，首先被赋给了isLegal()关联的LowZeroException类的无名引用，然后继续赋给myMethod()关联的LowZeroException类的无名引用，再继续赋给myMethod2()中的catch块中的形参e，最后在这里被处理掉，这个"异常"对象随即消失。可以说，catch(){}就是"异常"对象的生命终结点。 另外还要注意一点，方法与"异常"的关联可以一直向上传递，当传递到与main方法关联后，即在main()方法的定义中使用了throws Exception，这时除了虚拟机没有其它方法能够引用main()方法，且在程序中可能看不到try…catch程序块，但并不会产生错误，因为此时虚拟机会捕捉"异常"，并且会默认的调用printStackTrace()方法打印出"异常"路径。总之只要一个方法关联了"异常"，可以将这个"异常"关联向上传递，但是最终必须使用catch来终止"异常"，或者一直传递到main()方法交给Java虚拟机来结束"异常"对象的生命，否则是通不过编译的。 四、使用"异常机制"的需要注意的几点 1. 一个方法中可能会产生多种不同的异常，你可以设置多个"异常"抛出点来解决这个问题。 2. "异常"对象从产生点产生后，到被捕捉后终止生命的全过程中，实际上是一个传值过程，所以你可以根据需要，来合理的控制检测到"异常"的粒度。例如在例3中，如果你并不需要知道具体产生的

是LowZeroException"异常"，那么你可以使用"异常"的公共父类Exception来结合"异常"对象，即catch(Exception e){…}。同样在"异常"与方法关联的传递过程中，也可以根据需要控制关联"异常"的粒度，即throws后面跟上异常对象的父类名。 3. "异常机制"中还有一种特殊情况——RuntimeException"异常类"（见图1），这个"异常类"和它的所有子类都有一个特性，就是"异常"对象一产生就被Java虚拟机直接处理掉，即在方法中出现throw 子句的地方便被虚拟机捕捉了。因此凡是抛出这种"运行时异常"的方法在被引用时，不需要有try…catch语句来处理"异常"。 