异常的捕获及处理

# 引言

1. 异常的产生分析以及所带来的影响；
2. 异常处理的基本格式；
3. 异常的处理流程；
4. 异常的处理模式；
5. 自定义异常。

# 具体内容

Java中最为优秀的设计就在于异常的处理上，而且很简单。

## 认识异常

异常指的是导致程序中断执行的一种指令流。一旦产生异常并且没有正常处理的话，那么程序将会中断执行。

## 处理异常

如果要在Java中进行异常的处理可以使用三个关键字的组合完成：try、catch、finally。对于这三个关键字的组合，可以有如下的使用语法：

|  |
| --- |
| try {  // 有可能出现异常的语句  } [catchc (异常类型 对象) {  // 异常处理  } [catchc (异常类型 对象) {  // 异常处理  } …] [finally {  // 异常的统一出口代码  }] |

而此时给出的语法也有三种组合模式：try…catch、try…catch…finally、try…finally。

如果要想输出异常信息比较完整一些，则可以使用printStackTrace()方法完成。

## 多个异常的处理

## 异常处理流程

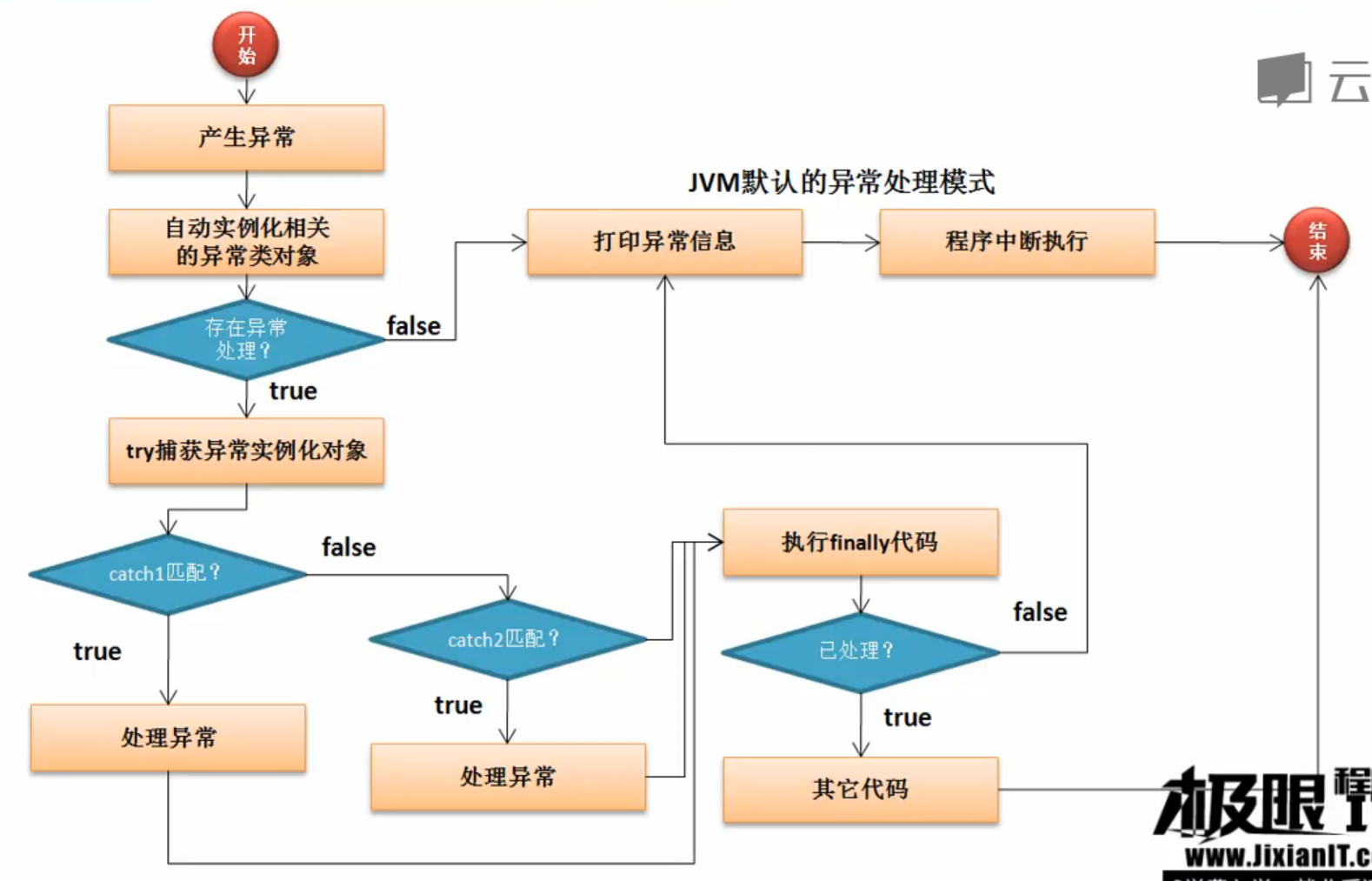
面试题：请接收Throwable下的Error和Exception子类的区别

* Error：是在程序还未执行时出现的错误，一般指的是JVM错误，用户无法处理。
* Exception：指的是在程序运行中出现的异常，异常处理都是针对于此类型完成的。

因此在以后的开发中，如果处理异常，能够处理的最大的父类就是Exception。

分析异常的处理流程：

1. 当程序中产生异常之后，会由JVM自动的根据异常类型实例化一个指定的异常类对象；
2. 程序需要判断当前的代码之中是否存在有异常的处理逻辑，如果没有，则交由JVM默认处理，处理方式就是输出异常信息，而后中断异常执行；
3. 如果程序中存在有异常处理，则try语句会捕获异常类的实例化对象（想象为引用传递）；
4. 捕获到的异常类的实例化对象要与catch中的异常类型进行依次匹配；
5. 如果catch匹配了该异常类型则使用相应代码进行处理，随后执行finally代码，如果没有任何一个catch匹配则直接跳转到finally；
6. 执行完finally代码之后要判断该异常是否已经处理过了，如果已经处理过了，则继续执行后续代码，而如果没有处理，则呈现将继续交由JVM默认处理；



通过以上的分析，可以发现所谓的catch匹配异常，就跟方法的参数传递没什么区别，有一个实例化对象，如果发现类型匹配，则进行接收。那么按照对象向上转型的原则，所有的类对下那个都可以向父类转换，那么也就证明都可以使用Exception进行处理。

在实际的开发过程之中，如果开发团队有明确的要求，那么就分开处理。但是如果要求不明确的情况下，使用Exception处理时最方便的。

## throws关键字

异常的处理语句本身使用并不复杂，但是最为麻烦的是，执行到某些代码的时候，用户如何知道这个代码会产生什么异常？所以在Java中提供有一个throws关键字，这个关键字的主要目的是明确的告诉用户执行某一个方法中由可能会产生哪些异常。所以throws主要是用于方法的声明处。

如果程序中使用了throws声明，那么就表示必须强制性进行处理异常的操作。

## throw关键字

现在为止所有的异常对象都是由Java负责进行实例化对象，我们能够进行的处理只能够进行捕获。但是在Java中允许用户自己去实例化异常类对象，那么如果要抛出这个对象，就必须使用throw关键字。

只要出现了throw，那么就表示代码中产生了异常，此时就两个选择：

* + 方法上使用throws继续抛出；
  + 手工使用异常处理。

面试题：请解释throw与throws的区别？

* throw在方法体中使用，表示手工抛出一个异常类的实例化对象；
* throws在方法的声明中使用，表示此方法调用时必须明确进行异常的处理。

## 异常处理的实际应用

## 断言：assert

所谓的断言指的是在程序编写过程之中，确定代码执行到某行之后数据一定是某个期待的内容。

|  |
| --- |
| assert num == 100:“啦啦啦啦”; |

|  |
| --- |
| java –ea TestDemo |

## 自定义异常类

在Java中提供了大量的类型，但是这些提供的类型几乎都是与语法有关的异常类型，缺少业务有关的类型。

对于异常类型主要有两个：Exception、RuntimeException

|  |
| --- |
| public class ScoreException extend Exception {  public ScoreException(String msg) {  super(msg);  }  } |

# 总结