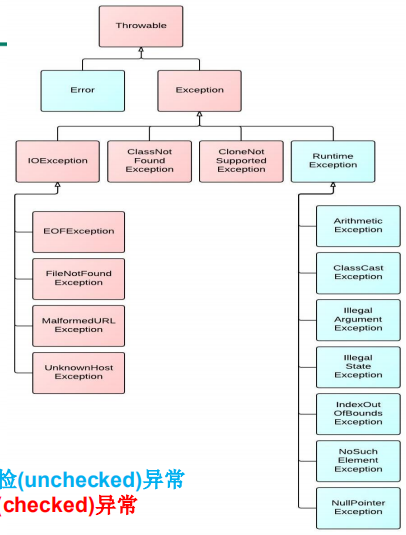
# 异常体系结构

红色为编译时异常

蓝色为运行时异常



# 异常处理过程

***过程一***："抛"：程序在正常执行的过程中，一旦出现异常，就会在异常代码处**生成**一个对应**异常类的对象**。并将此对象抛出。一旦抛出对象以后，**其后的代码就不再执行！！！**

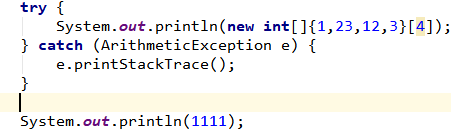
关于异常对象的产生：① 系统自动生成的异常对象

② 手动的生成一个异常对象，并抛出（throw）

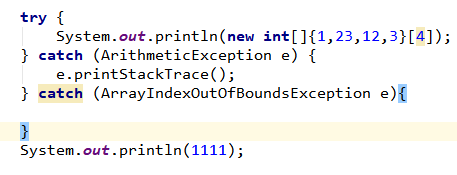
***过程二***："抓"：可以理解为异常的处理方式：① try-catch-finally ② throws

# 异常处理的两种方式

## 方式一、 try-catch -finally



**无法执行打印，因为没有捕获到 ArithmeticException 异常**



**可以执行打印**

### **说明**

2. 使用try将可能出现异常代码包装起来，在执行过程中，一旦出现异常，就会生成一个对应异常类的对象，根据此对象

的类型，去catch中进行匹配

3. 一旦try中的异常对象匹配到某一个catch时，就进入catch中进行异常的处理。一旦处理完成，就跳出当前的try-catch结构（在没有写finally的情况）。继续执行其后的代码

4. catch中的异常类型如果没有子父类关系，则谁声明在上，谁声明在下无所谓。

catch中的异常类型如果满足子父类关系，则**要求子类一定声明在父类的上面**。否则，报错

5. 常用的异常对象处理的方式： ① String getMessage() ② printStackTrace()

6. 在try结构中声明的变量，再出了try结构以后，就不能再被调用

7. try-catch-finally结构可以嵌套

### finally

1.finally是可选的

2.finally中声明的是一定会被执行的代码。即使catch中又出现异常了，try中有return语句，catch中有return语句等情况。

3.像数据库连接、输入输出流、网络编程Socket等资源，JVM是不能自动的回收的，我们需要自己手动的进行资源的释放。此时的资源释放，就需要声明在finally中。

## 方式二、 throws + 异常类型

1. "throws + 异常类型"写在方法的声明处。指明此方法执行时，可能会抛出的异常类型。一旦当方法体执行时，出现异常，仍会在异常代码处生成一个异常类的对象，此对象满足throws后异常类型时，就会被抛出。异常代码后续的代码，就不再执行！

2. 体会：try-catch-finally:真正的将异常给处理掉了。throws的方式只是将异常抛给了方法的调用者。 并没有真正将异常处理掉。

# 4、如何选择两种异常处理方式

3. 开发中如何选择使用try-catch-finally 还是使用throws？

3.1 如果父类中被重写的方法没有throws方式处理异常，则子类重写的方法也不能使用throws，意味着如果子类重写的方法中有异常，必须使用try-catch-finally方式处理。

3.2 执行的方法a中，先后又调用了另外的几个方法，这几个方法是递进关系执行的。我们建议这几个方法使用throws的方式进行处理。而执行的方法a可以考虑使用try-catch-finally方式进行处理。

# 5、手动抛出异常



RuntimeException的不用处理

Exception 因为包含编译时异常所以要处理

# 6、自定义异常

1、继承于现有的异常结构：RuntimeException 、Exception  
 2、提供全局常量：serialVersionUID

1. 提供重载的构造器

