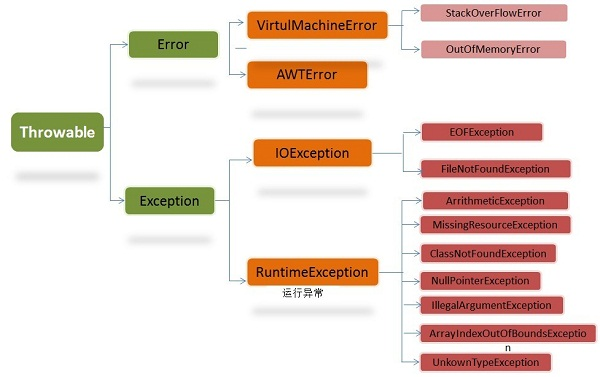
<http://blog.csdn.net/hguisu/article/details/6155636>

Java中的异常层次如下图所示：



Java对异常的处理机制：抛出异常，捕获异常。

捕获异常：

Try/catch序列：try中是可能产生异常的代码；

Catch后的括号项是必需的，如

Try{

//code

}

catch（NumberFormatException e）

{System.out.println(e)}

对象e中提供了一些错误的信息供打印。

Try/catch/finally

必须要清空一些在try中创建的对象。例如，在离开try块之前，应该先将try块中打开的文件关闭。之所以这样做是因为有这么一个问题：如果在执行try块期间，抛出了异常对象，因为异常立即使try块中断，可能会省略了清空。虽然我们可以将清空紧随在catch字句，但是不能保证所有的异常都能够被捕获，因此需要用finally来进行清空操作。

也就是说，finally就是无论是否发生异常，都必须要进行的清空操作放在这里面。

一般又以下三种方案：

1. 如果执行try块没有异常，则控制传递给finally块。即使try通过return、break或continue在最后的语句退出，控制也是这样传递；
2. 如果在try内遇到未捕获的异常，则控制传递给finally快，然后，在执行finally块之后，将异常传播；
3. 如果在try内遇到捕获的异常，则控制传递给相应的catch，然后执行catch快后再执行finally。

抛出异常：

方法名后的throws Exception1,Exception2,...,ExceptionN 为声明要抛出的异常列表。当方法抛出异常列表的异常时，方法将不对这些类型及其子类类型的异常作处理，而抛向调用该方法的方法，由他去处理。

throw总是出现在函数体中，用来抛出一个Throwable类型的异常。程序会在throw语句后立即终止，它后面的语句执行不到，然后在包含它的所有try块中（可能在上层调用函数中）从里向外寻找含有与其匹配的catch子句的try块。我们知道，异常是异常类的实例对象，我们可以创建异常类的实例对象通过throw语句抛出。该语句的语法格式为：

throw new exceptionname;

例如抛出一个IOException类的异常对象：

throw new IOException;

要注意的是，throw 抛出的只能够是可抛出类Throwable 或者其子类的实例对象。下面的操作是错误的：

throw new String("exception");

这是因为String 不是Throwable 类的子类。

Throw&throws