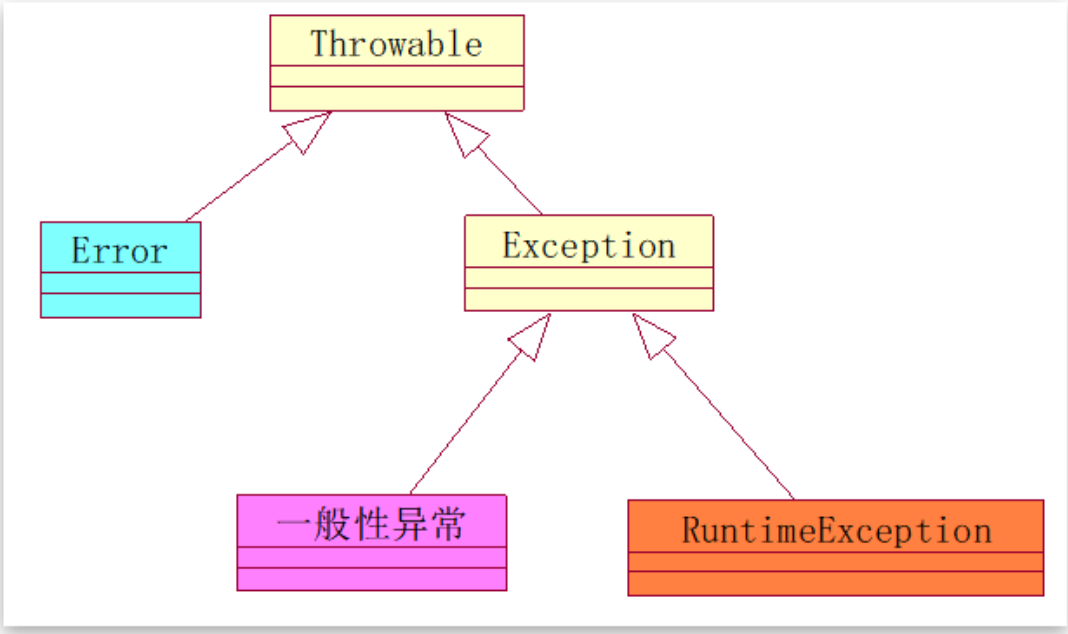
所有的异常都继承与Throwable

3.1 异常的类图结构

先来整体认识下异常的类图结构：



继承exception的异常分为两种 error和Exception

error是指程序不能恢复的错误，系统崩溃，虚拟机错误，内存空间不足，方法调用栈溢出等。对于这类错误的导致的应用程序中断，仅靠程序本身无法恢复和预防，遇到这样的错误，建议让程序终止
Exception是指程序可以恢复的错误,表示程序可以处理的异常，可以捕获且可能恢复。
这种异常是由与程序设计的不完善而出现的问题，遇到这类异常，应该尽可能处理异常，使程序恢复运行，而不应该随意终止异常。

Exception又分成两种
一种是受控错误：受控错误是自己来throws或者try ...catch...来实现的。这个在编译的时候就会检查。主要是处理一些已知的错误。
一种是非受控错误：不用专门写try...catch之类的来处理异常，这种是系统运行时候的报错。其实就是处理一些意外情况的报错，比如文件不存在，除数为零等。

有一个特别要注意的机制就是
try{}
catch{}
finally{}
这种结构的时候，如果在try中报错进入catch之后，在catch中遇到了return 之后会限制性fally中的代码，如果fally中也是return那么程序就会直接中止

5.7 调用下面的方法，得到的返回值是什么?

这个问题，估计大神也可能回答错误。

```
1    public static int getNum() {
2        try {
3            int a = 1 / 0;
4            return 1;
5        } catch (Exception e) {
6            return 2;
7        } finally {
8            return 3;
9        }
10   }
```

上述代码调用后，返回的结果是3，代码分析如下：

public static void main(String[] args) {

//代码走到第 3 行的时候遇到了一个MathException，这时第 4 行的代码就不会执行了，代码直接跳转到catch语句中，走到第 6 行的时候，

//异常机制有一个原则：如果在catch中遇到了return或者异常等能使该函数终止的话那么有finally就必须先执行完finally代码块里面的代码然后再返回值。

//因此代码又跳到第 8 行，可惜第 8 行是一个return语句，那么这个时候方法就结束了，因此第 6 行的返回结果就无法被真正返回。

//如果finally仅仅是处理了一个释放资源的操作，那么该道题最终返回的结果就是2，因此上面这道题返回值是3。

System.out.println(getNum());

}