**异常是什么？**

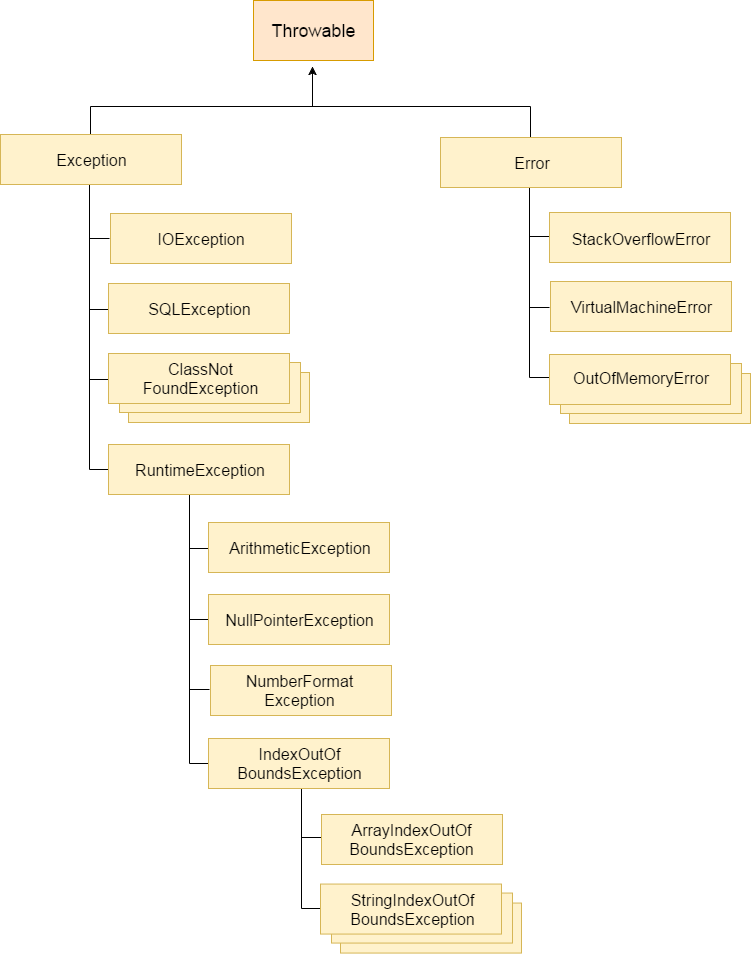
程序运行过程中出现了不正常的情况。

**为什么要处理异常？**

如果不处理异常，程序就会在异常处终止，不会继续执行。

Java的异常处理是一种强大的机制，能够处理运行时的错误，维持程序的正常执行流程。

**Java的异常体系**



**Java的异常分类**

1.Checked Exception（受检异常）

继承自Exception但不是RuntimeException的子类。受检异常在编译时被检查。

1. Unchecked Exception（未受检异常）

继承自RuntimeException。编译时不会被检查。运行时被检查。

1. Error（错误）

不可恢复的异常，程序无法恢复正常状态继续运行。只能退出。

4.异常处理主要是处理checked异常，因为unchecked异常通常都是程序员粗心造成的，通常可以避免。

**怎样处理异常？**

try {

// 可能抛出异常的代码

// try后面必须有catch或者finally中的至少一个

// 一个try后面可以跟随多个catch

// 一个try后面最多只能跟随一个finally

// 被try包围的语句，如果某处发生异常，那么其后面的语句将不会执行

// 因为程序逻辑已经跳转到catch块中。

int a[]=new int[5];

a[5]=30/0;

} catch (ArithmeticException e) {

// 如果异常发生了，程序执行逻辑会转到catch块，在此处处理异常

} catch (ArrayIndexOutOfBoundsException e) {

// 在某个时刻最多只有一个异常会发生，因此只会有一个catch块被执行

} catch (Exception e) {

// 如果有多个catch块，异常排列的顺序必须是从最具体到最一般的。

} finally {

// 只要程序执行到try内，不论异常是否发生，也不论异常发生后是否被处理，finally // 块都会执行，除非程序退出（例如System.exit()）或者出现严重错误导致进程终止。

// finally块一般用来释放资源。

}

**异常的传播**

对于unchecked异常，如果方法1产生了异常但未处理，方法2调用了方法1，那么异常会传递到方法2，依此类推，直到有方法来处理，如果都不处理则JVM处理（打印异常描述信息和栈跟踪信息，终止程序）。

对于checked异常，通过在方法签名中用throws声明异常能够让checked异常传播。如果在方法1中通过throw显式抛出了checked异常，那么方法1必须在其签名中声明该异常。如果方法2调用了声明了checked异常的方法1，那么方法2必须处理异常（不论异常是否发生），要么catch后自己处理，要么在方法签名中声明给调用者处理。

注：如果在方法签名中声明的是unchecked异常，调用者可以不做任何处理。不做处理只是说明在语法上合理，编译可以通过，不代表运行一定正常。程序应该根据实际情况处理unchecked异常，主要是提前做一些参数检查。

**throw和throws关键字**

throw用于显式抛出一个异常，后面跟着一个异常实例对象，能够抛出checked或unchecked异常。

throws用于某个方法声明异常，后面跟一个或多个异常类型，目的是提供信息给调用者，这个方法可能会出现异常，调用者需要妥善处理。

**方法重写中的异常处理**

1. 如果父类方法没有声明任何异常，子类重写父类方法不能声明checked异常，但能声明unchecked异常。
2. 如果父类方法声明了异常，子类重写父类方法可以声明相同的异常或子类异常或不声明异常，不能声明父类异常。

简而言之，子类重写方法声明的异常要比父类方法“小”。

**自定义异常**

一般是继承java.lang.Exception