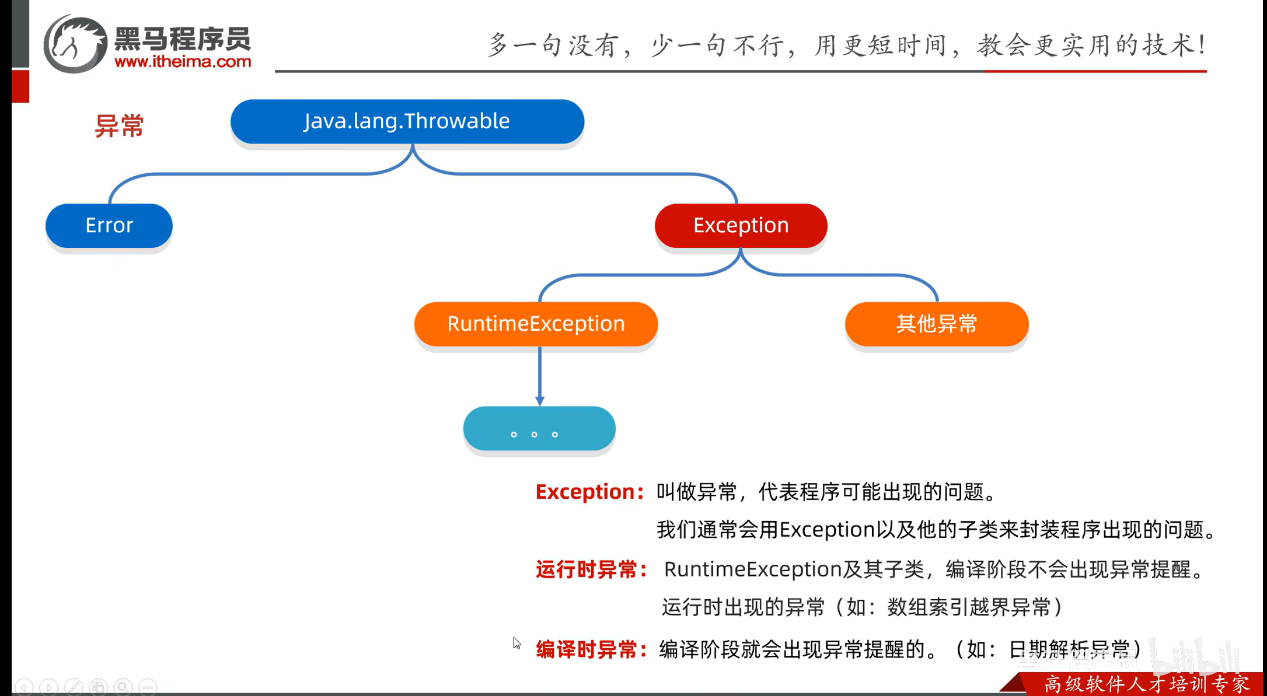
异常

1目的：当程序出现异常的时候应该怎么处理。

2分类：Error：代表系统级别的错误，是给开发人员使用的。



3、区别：RuntimeException是在运行时出现异常，除了RuntimeExecption和他的子类，其它的的错误都是编译时异常

4、作用：查询bug的关键参考信息

可以作为方法内部的一种特殊的返回值，

5、异常的处理：（1）JVM默认的处理方式

（2）自己处理

（3）抛出异常

（1）在出现问题的行数后，系统会停止运行，提示报错的行数

（2）try {有可能出现问题的代码}catch(异常类名变量名){处理的代码} 这里的异常类名，是出什么问题就用该Expection类的名称，变量名随便起。

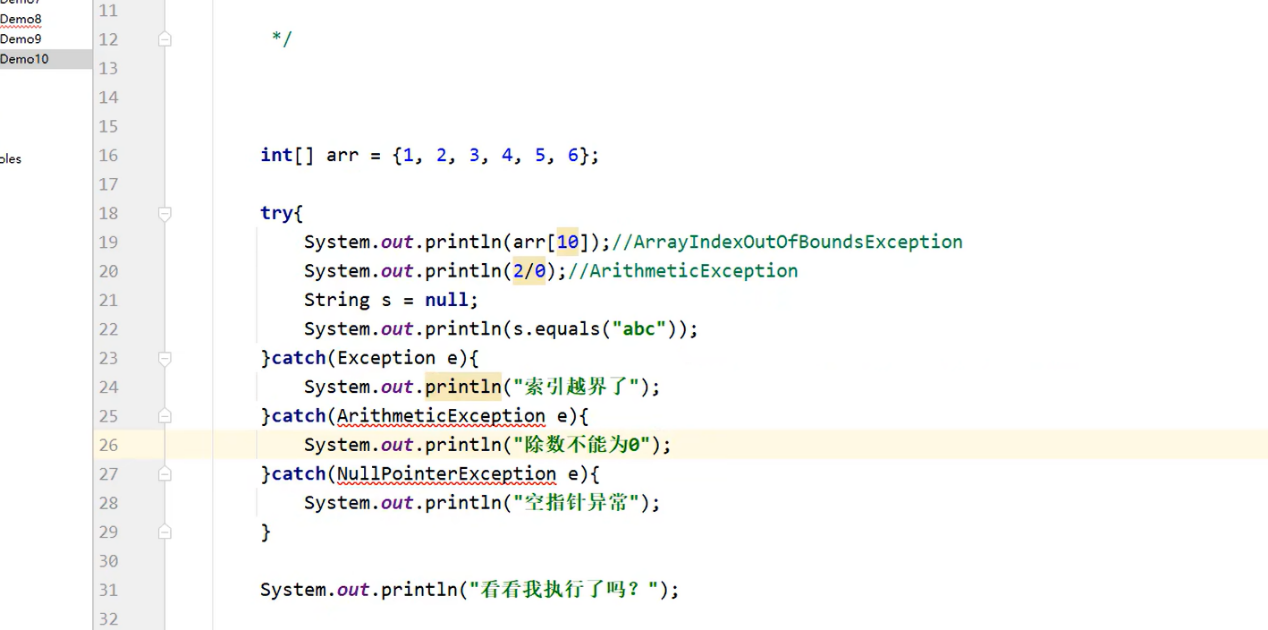
这样做可以使问题代码出现后，依然会运行之后的代码

原理：当在try中的代码出现问题的时候，系统会将将出现的问题与catch中的异常类名进行对比，如果对应，则捕获到，之后会执行Cath中的代码。然后继续执行后面的代码。

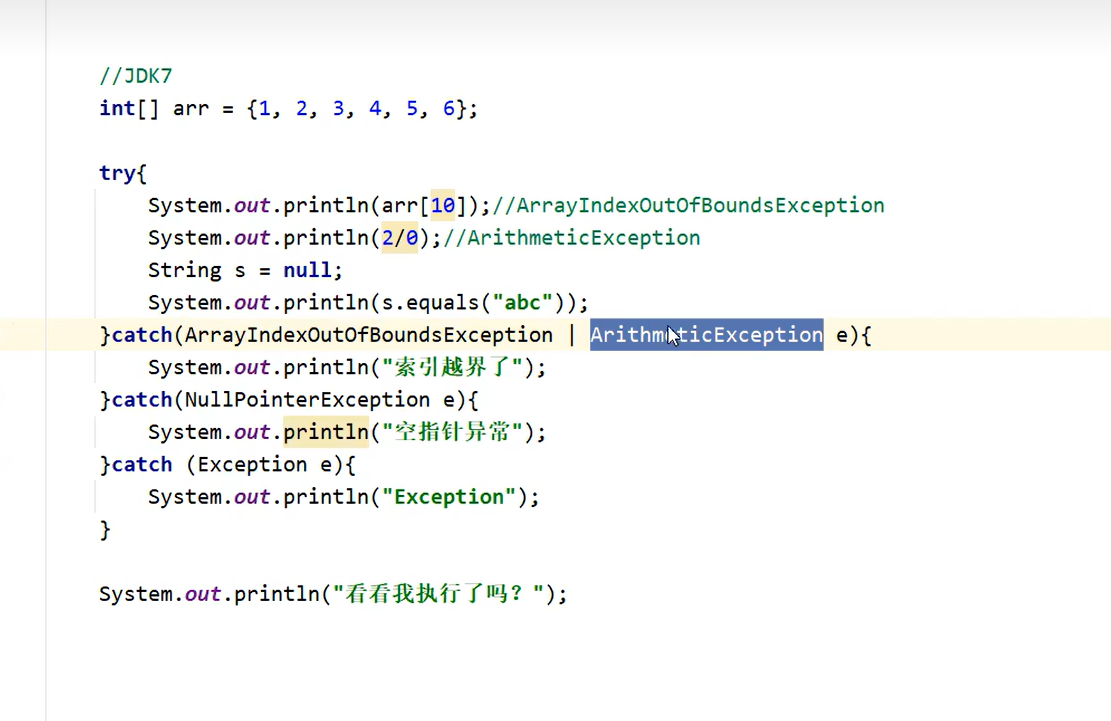
出现的情况：1、当try中的代码没有问题的时候，系统会直接指向try中的所有代码，不会执行catch中的代码。

2、当try中出现多个问题的时候：如果只有一个catch捕获的时候，会先捕获先出现问题的地方，然后跳过try中问题该异常之后的所有代码，当有多个catch出现的时候，会进行对应捕获。

当catch创建的类存在父子关系的时候，父类放在子类的后面。

例如，

在JDK7之后，如果有多个catch类，且解决方案是一致的，则可以



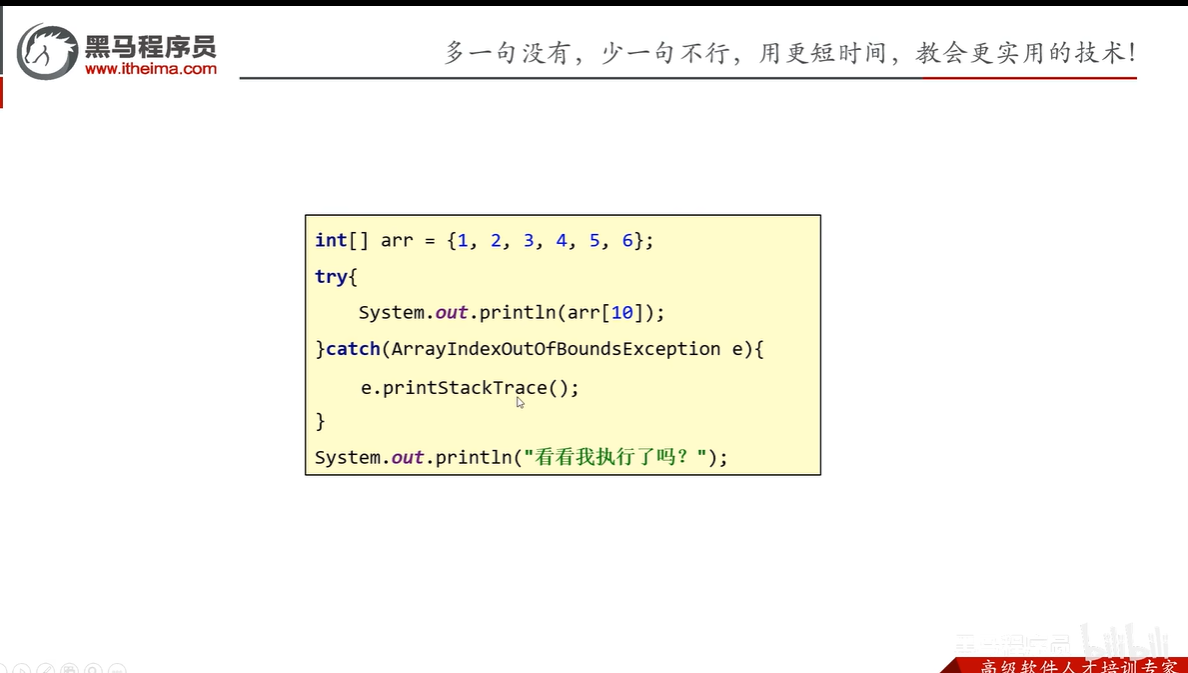
其“|“只能出现一次

3、当catch没有捕获到问题的时候，会采用默认的处理方式

4、在捕获到之后，try中问题后面的代码不会执行

（3）异常的常见处理方法

1、当我们在使用try...catch去捕获异常的时候，我们通常在catch中使用：异常类名.printStackTrace();



这里的方法其实是属于Throwable的成员方法



这三种方法在catch区域可以同时调用。其中最为常用的是第三种，因为第三种可以返回位置、异常等所有信息

2、printStackTrace打印信息为红色的原因：当我们在输出信息的时候，我们通常是System.print.out.println();

但是还有另外一种错误的输出语句（用来打印错误信息）：

System.err.println();这种情况下会打印红色的字体。

而printStackTrace就是以这种方法输出的。（了解）

3、抛出处理（提升代码的健壮性）：

作用：在方法中，出现异常时，停止运行方法，采取抛出处理，让编译者知道方法出现了问题，需要进行处理

4、异常的三种处理方法：

（1）虚拟机默认处理方式，把异常信息以红色字体的形式打印到控制台上，并结束程序。

（2）try catch：一般在调用处，目的是为了让代码能够继续运行下去

（3）throws throw：在方法中，出现异常时，停止运行方法，采取抛出处理，让编译者知道方法出现了问题，需要进行处理

5、自定义异常（能够更好的表达异常）

（1）书写格式：定义异常类

写继承关系

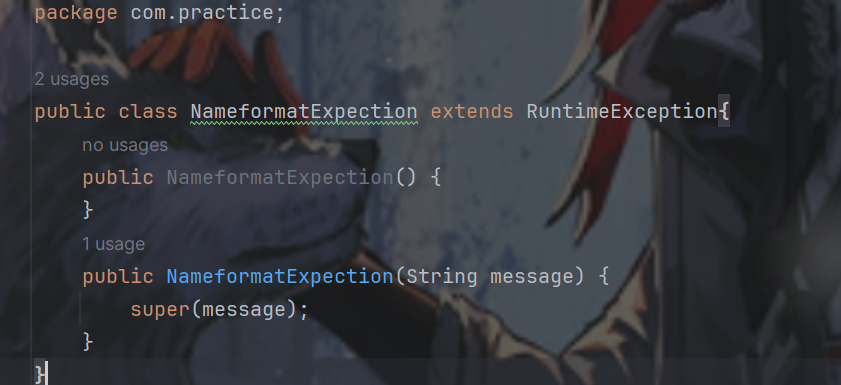
空参构造

带参构造

1. 作用：**提供更具体的异常信息**：通过自定义异常类，您可以为特定的错误或异常情况提供更具体和详细的描述，从而使程序员能够更好地理解和处理异常。
2. **增强可读性和可维护性**：通过使用自定义异常类，您可以为不同的异常情况创建不同的异常类型，这样可以使代码更易于阅读和维护。每个异常类型都能够明确表示一个特定的错误条件。
3. **提供更好的错误处理机制**：自定义异常类允许您将程序中可能发生的不同类型的错误进行分类和组织，这样可以为每种异常情况提供相应的处理机制，从而增强程序的健壮性和可靠性。
4. **与标准异常区分**：有时候，标准异常类无法满足特定的应用需求，因此自定义异常类可以提供更具体的异常类型，从而与标准异常进行区分，并更好地适应特定的应用场景。

样例：

1. 自定义类：



这里的两个方法都来自父类，不需要自己创建

1. 在类中：



可以在参数列表中输入信息

（3）在Main中