异常：
在某些情况下，因为用户的一些原因，例如：配置问题、错误输入的问题、用户磁盘空间满了等因素，导致程序无法正常运行。

不是异常：（1）语法错误（2）逻辑错误

java 的程序如果出现异常，并且在代码中，没有做相应的处理，那么程序会“崩溃”，“挂了”

Java 如何处理异常？或 Java 异常处理的过程？
1）当程序运行到某一句时，发生了异常，那么程序会先停下来
2）程序会在这句代码处，查看原因，生成一个合理“异常对象”，然后“抛”出
3）JVM 会检测在这句代码的外围，是否有 try..catch 结构，可以“捕获”它，
如果可以捕获，那么程序再处理完异常后，继续下面的运行，不会崩溃；
如果不能捕获，那么会把这个异常继续抛给“上级”，如果“上级”能处理，那么程序从“上级"处理完的代码后面继续运行；
如果上级也不能处理，那么继续往上抛，一直到达 JVM，那么就“崩溃”

所有类的根父类是 Object。
枚举的公共父类是 Enum，根父类仍然是 Object
异常的公共父类是 Throwable，根父类仍然是 Object
1、异常的公共父类：java.lang.Throwable
(1)只有当对象是此类（或其子类之一）的实例时，才能通过 Java 虚拟机或者 Java throw 语句“抛”出。
(2)只有此类或其子类之一才可以是 catch 子句中的参数类型。

2、Throwable 又分为两大派别：
（1）Error：错误
一般指严重错误，一般合理的应用程序不应该试图去捕获它。
如果出现这个问题，要么需要升级程序，要么需要升级架构，要么需要升级硬件。
例如：报了一个 OutOfMemoryError
经典代表：VirtualMachineError（堆内存溢出 OutOfMemoryError，栈内存溢出 StackOverflowError）
（2）Exception：异常
一般异常，合理的应用程序应该试图去捕获它。

3、Exception 还可以分为两大类：
（1）运行时异常(RuntimeException 或它子类)：又称为非受检异常
编译时，编译器是不会提醒你做处理的，只有运行期间，才会发生。
运行时异常是不建议用 try...catch，因为它发生频率太高，而且一般都是很不应该发生的问题。
例如：空指针异常，数组下标越界异常，类型转换异常等，这些异常完全可以避免掉。
但是如果实在没有考虑到，也可以通过 try...catch 处理。

（2）编译时异常，除了 RuntimeException 系列以外的，都是编译时异常。又称为受检异常。

编译时，编译器会强制要求程序员编写处理的代码，如果你不编写，那么就编译不通过。

例如：FileNotFoundException，IOException 等

finally 与 return 混用：

（1）不管 try 中是否发生异常，也不管 catch 是否可以捕获异常，也无论 try 或 catch 中是否有 return。

finally 中的代码都必须执行

（2）如果 finally 中有 return，就从 finally 块的的 return 回去。

（3）如果 finally 中没有 return，那么先把 try 或 catch 中该执行的执行完，

在 return 结束当前方法之前，先走一下 finally，然后回去结束当前方法

一、异常的处理方式：try...catch

1、语法格式：

```
try{
        可能发生异常的代码
}catch(异常类型 1  异常对象名){//异常对象名绝大多数都是写 e
        处理这个异常的代码
}catch(异常类型 2  异常对象名){//异常对象名绝大多数都是写 e
        处理这个异常的代码
}catch(异常类型 3  异常对象名){//异常对象名绝大多数都是写 e
        处理这个异常的代码
}
。。。。
```

2、异常对象的常用方法

（1）e.printStackTrace();

打印异常的详细信息，包括追踪跟踪信息，即这个异常对象一路经过了哪些方法

（2）e.getMessage();

返回异常对象中简单的错误信息提示

3、打印异常/错误信息

System.err.println(xx);打印错误信息

System.out.println(xx);打印正常信息

4、多个 catch 分支，如何匹配和执行的？

从上到下依次判断，一旦有一个满足，后面就不看了。

建议：如果多个 catch 中的异常类型有大小包含关系，那么小的在上，大的在下，如果没有大小包含关系，顺序随意。

5、如果 catch，可以捕获 try 中发生的异常，那么程序，会从 try...catch 下面的代码继续运行 ，不会崩溃。

如果 catch 无法捕获 try 中发生的异常，那么就会导致当前方法结束，并把异常对象抛出调用者，

如果调用者可以处理，那么从调用者处理代码的后面继续运行，否则继续往上抛，最终

到达 JVM，程序就崩溃了。
二、try..catch 的形式二
```
try{
        可能发生异常的代码
}catch(异常类型 1  异常对象名){//异常对象名绝大多数都是写 e
        处理这个异常的代码
}catch(异常类型 2  异常对象名){//异常对象名绝大多数都是写 e
        处理这个异常的代码
}catch(异常类型 3  异常对象名){//异常对象名绝大多数都是写 e
        处理这个异常的代码
}
。。。。
finally{
        不管 try 中是否发生异常，也不管 catch 是否可以捕获异常，这里代码都必须执行
}
```
一般用于编写释放资源，断开连接等代码
特殊情况：可以没有 catch 部分
```
try{
 * }finally{
 * }
```

(1)finally 里面有 return，就从 finally 的 return 回去了
(2)类初始化：main 所在的类要先初始化，才能执行 main 方法
由①静态变量的显示赋值（这里没有）②静态代码块
（3）实例初始化：必须要 new 对象才会有，这里没有创建 TestExer4 的对象，所以不走
异常处理的方式之一：在当前方法中直接用 try...catch 处理
异常处理的方式之二：在当前方法中不处理，扔/抛给调用者处理
throws 的好处：
（1）throws：告知被调用者，我这个方法可能会抛出哪些异常，使得调用者可以明确的知道应该 catch 什么异常
如果没有 throws，那么调用者就不清楚，只能按照 Exception 处理，或者根据错误经验来处理。
（2）编译时异常，如果在当前方法中不用 try..catch 处理，编译不通过，那么可以通过 throws 明确的说明，抛给调用者处理
throws 的语法格式：
```
【修饰符】 返回值类型  方法名(【形参列表】)throws 异常列表们{
}
```
说明：throws 后面可以跟好几个异常，顺序无所谓，每一个异常之间使用,分割
关于方法重写时，对 throws 抛出的异常的要求：
子类重写的方法抛出的异常类型必须<=父类被重写的方法抛出的异常类型。
例如：Exception > RuntimeException > ArrayIndexOutOfBoundsException
整理重写的要求：
（1）方法名：相同
（2）形参列表：相同

（3）返回值类型：

基本数据类型与 void：相同

引用数据类型：<=

（4）修饰符

权限修饰符：>=

其他修饰符：不能是 final,private,static

（5）抛出的异常类型：<=

异常的对象的创建和抛出有两种方式：

（1）JVM 创建并抛出

（2）程序员 new 出来，然后由 throw 抛出。

Throwable：

只有当对象是此类（或其子类之一）的实例时，才能通过 Java 虚拟机或者 Java throw 语句抛出。

类似地，只有此类或其子类之一才可以是 catch 子句中的参数类型。

throw：

用于手动抛出异常对象。

语法格式：

throw 程序员 new 的异常对象;

可以代替 return 语句，结束当前的方法

面试题：throw 和 throws 什么区别?

（1）throw 用于手动抛出异常对象，是个可执行的语句

（2）throws，在方法签名中，声明方法可能抛出什么异常，让调用者来处理这些异常。

自定义异常：

如果系统预定义的异常类型，

例如：ArrayIndexOutOfBoundsException

ClassCastException

NullPointerException

ArithmeticException

InputMisMatchException

IllegalAugumentException

....

发现不能准确的表达你当前的异常类型的意思时，你可以选择自定义。

面试题：列出常见的异常类型，已经什么情况下会发生这个异常，你如何处理?

至少 5 个

1、自定义的要求：

（1）必须继承 Throwable 或它的子类

但是实际开发中，一般继承 RuntimeException 和 Exception

（2）建议大家保留两种构造器的形式

①无参构造

②带给父类的 message 属性赋值的构造器

2、如何使用自定义异常

只能使用 throw 语句进行手动抛出。它不能由 JVM 自动抛出。

3、建议

大家在自定义异常时，异常的类型名非常重要，见名知意。