异常：就是在程序运行时候出现的不正常情况。在java中将其封装成类。

在javax.lang 包中

Throwable

|--Error

|--Exception

# **异常分类**：

1. 可处理非严重异常Exception
2. 不可处理异常Error

# 对异常的处理。

处理异常的语句：

try{

需要被检测的代码

}

catch(Exception e){

处理异常的代码

}

finally{

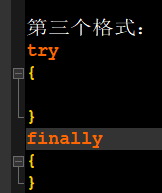
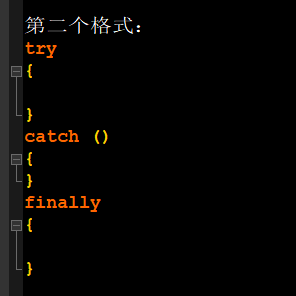
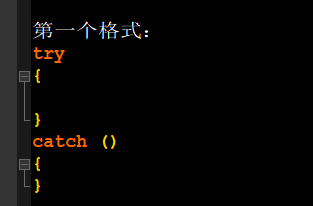
通常是释放资源的操作，

无论如何都要执行的代码 就算在try和catch中有return 也要执行finally里的语句

但是特殊情况，如果出现System.exit(0); 就不会执行finally块中的语句，因为JVM直接

退出了

}



在函数上声明异常。

便于提高安全性，让调用出进行处理。不处理编译失败。

1，声明异常时，建议声明更为具体的异常。这样处理的可以更具体。

2，对方声明几个异常，就对应有几个catch块。不要定义多余的catch块。

如果多个catch块中的异常出现继承关系，父类异常catch块放在最下面。

如果父类异常放在子类的上面，异常将会被父类捕捉，而子类无法捕获异常。

建立在进行catch处理时，catch中一定要定义具体处理方式。

不要简单定义一句 e.printStackTrace(),

也不要简单的就书写一条输出语句。

# 自定义异常

需求：在本程序中，对于除数是-1，也视为是错误的是无法进行运算的。

那么就需要对这个问题进行自定义的描述。

当在函数内部出现了throw抛出异常对象，那么就必须要给对应的处理动作。

要么在内部try catch处理。

要么在函数上声明让调用者处理。

一般情况在，函数内出现异常，函数上需要声明。

发现打印的结果中只有异常的名称，却没有异常的信息。

因为自定义的异常并未定义信息。

如何定义异常信息呢？

因为父类中已经把异常信息的操作都完成了。

所以子类只要在构造时，将异常信息传递给父类通过super语句。

那么就可以直接通过getMessage方法获取自定义的异常信息。

自定义异常：

必须是自定义类继承Exception。

继承Exception原因：

异常体系有一个特点：因为异常类和异常对象都被抛出。

他们都具备可抛性。这个可抛性是Throwable这个体系中独有特点。

只有这个体系中的类和对象才可以被throws和throw操作。

throws和throw的区别

throws使用在函数上。

throw使用在函数内。

throws后面跟的异常类。可以跟多个。用逗号隔开。

throw后跟的是异常对象。

# RuntimeException

Exceptoin中有一个特殊的子类异常RuntimeException 运行时异常。

如果在函数内容抛出该异常，函数上可以不用声明，编译一样通过。

如果在函数上声明了该异常。调用者可以不用进行处理。编译一样通过；

之所以不用在函数声明，是因为不需要让调用者处理。

当该异常发生，希望程序停止。因为在运行时，出现了无法继续运算的情况，希望停止程序后，对代码进行修正。

自定义异常时：如果该异常的发生，无法在继续进行运算，

就让自定义异常继承RuntimeException。

对于异常分两种：

1，编译时被检测的异常。

2，编译时不被检测的异常(运行时异常。RuntimeException以及其子类)

# 子父类覆盖：

异常在子父类覆盖中的体现；

1，子类在覆盖父类时，如果父类的方法抛出异常，那么子类的覆盖方法，只能抛出父类的异常或者该异常的子类。

2，如果父类方法抛出多个异常，那么子类在覆盖该方法时，只能抛出父类异常的子集。

3，如果父类或者接口的方法中没有异常抛出，那么子类在覆盖方法时，也不可以抛出异常。