异常链

Java 从 1.4 版本以后有了异常链这个概念，显然异常链的出现是为了解决某种设计上出现的问题。那么这个问题到底是什么呢？

我们知道如果 A 方法调用 B 方法，B 方法抛出异常，那么 A 方法有几种处理的方式。一种是再次将这个异常抛出，另外一种是使用 try-catch 进行处理。那么异常链是什么呢？异常链指得就是在 try-catch 处理异常的时候，在 catch 里面不做细致的处理，而仅仅是将异常再次抛出来。语法如下所示。

语法

```
try {
    //会抛出异常的代码
}catch(XXXException e) {
    throw new XXXException(e);
}
```

然后通过方法声明 throws 再声明该方法继续抛出某种异常。这就是异常链，就是 A 方法调用 B 方法，B 方法抛出异常。A 方法使用 try-catch 处理了这个异常,同时在 catch 当中将异常再次抛出。这样就形成了异常链,在 catch 当中我们将异常再次抛出需要有一个选择。就是继续抛出原有的异常还是抛出一个新的异常呢？我们可以通过具体的分析来看一下。

首先，如果是抛出原有的异常，那么这显然是一个很糟糕的设计。因为 A 方法与 B 方法进行了关联，它们通过抛出同一种类型的异常关联了起来。这样不便于代码的修改和维护，那么另外一种方式。我们抛出一种新的异常,这显然解决了 A 方法和 B 方法相互关联的问题，但是也带来了一种新的问题，就是原有的异常信息丢失了。那么异常链是如何解决这个问题的呢？使用异常链我们可以抛出一个新的异常的同时保留原有的异常信息。在自定义异常时，使用到了 Throwable 的两个构造方法。

```
public MyException(String message, Throwable cause) {
    super(message, cause);
}
public MyException(Throwable cause) {
    super(cause);
}
```

这两个构造方法中都有一个参数类型是 Throwable，参数名是 cause。这个实际上就是对于异常链而使用的。就是我们抛出一个新的异常类型的时候，在构建这个新的异常对象的时候，我们可以通过这两个构造方法传入一个 Throwable 类型的对象。这个 Throwable 类型的对象代表的是引起现在新的异常的原有的异常类型，这样就保留了原有的异常信息。

如果我们在编写自定义异常的时候没有添加后两个构造方法，那么也可以通过 Throwable 里的另外一个方法来添加异常链的信息，就是 initCause()这个方法，通过 initCause()也可以指定引起新的异常的异常原因。这里要注意：如果 cause 指向自己，说明该属性没有初始化或者呢该异常就已经是异常的源头了。

下面我们通过代码示例来学习什么是异常链,以及如何使用异常链,代码如示例 1 所示。

示例 1：

```
/**
 * 异常链
 * @author 北大青鸟
 */

public class MyException extends Exception {
   public MyException() {
       super();
   }
   public MyException(String message) {
       super(message);
   }
   public MyException(String message, Throwable cause) {
       super(message, cause);
   }
   public MyException(Throwable cause) {
       super(cause);
   }
   public static void main(String[] args) throws MyException {
       try{
           throw new Exception("我的异常信息");
       }catch(Exception e){
           //throw new MyException(e);
           throw new MyException("新的异常信息", e);
       }
   }
}
```

示例 1 代码中，MyException 是自定义异常类，如果要使用异常链，需要编写 public MyException(String message, Throwable cause) {}和 public MyException(String message, Throwable cause) {}这两个构造方法。也就是包含 Throwable 这个参数类型的构造方法。然后让这两个构造方法，调用父类的已经实现好的相应的构造方法，这样我们在创

建新的异常对象的时候，就可以使用这两个构造方法传递异常的原因。若要使用它形成异常链，可以在 main 方法中创建 Exception 类型的异常对象，使用 try-catch 对它进行捕获和处理。在 catch 中，要抛出一个新的异常类型，又不想损失原有的异常信息，这时可以使用异常链。即在创建新的异常类型时，传递一个参数是原有的异常对象，也就是说引起现在这个 MyException 类型异常的原因，是 Exception 这种异常的原因。

如果还有其他的方法调整当前这个方法，其他方法也可以知道这个方法产生异常的原因，就不会丢失异常的信息了