内容

- Java 异常的概念
- Java 异常的分类
- 异常的捕获和处理
- 运行期出现的错误
- 注意观察错误的名字和行号

一、 异常的概念

异常(Exception)即例外，程序没有按自己预想的结果运行出来，出现了非正常情况，即“程序得病了”。怎么让我们写的程序做出合理的处理，不至于崩溃是我们关注的核心。 异常机制就是当程序出现错误，程序如何安全退出的机制。

所谓错误是指在程序运行的过程中发生的一些例外事件（如：除 0，数组下标越界，所要读取的文件不存在）。

Java 异常是 Java 提供的用于处理程序中错误的一种机制。

设计良好的程序应该在异常发生时提供处理这些错误的方法，使得程序不会因为异常的发生而阻断或发生不可预见的结果。

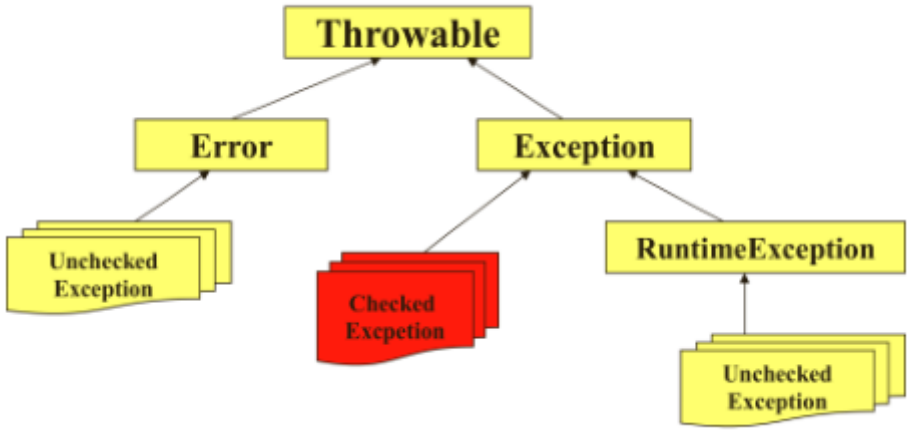
Java 程序的执行过程中如出现例外事件，可以生成一个异常类对象，该异常对象封装了例外事件的信息并将被提交给 Java 运行时系统，这个过程称为**抛出(throw)异常**。

当 Java 运行时系统接收到异常对象时，会寻找能处理这一异常的代码并把当前异常对象交给其处理，这一过程称为**捕获（catch）异常**。

二、 异常的分类

JDK 中定义了很多异常类，这些类对应了各种各样可能出现的例外事件

我开着车走在路上，一头猪冲在路中间，我刹车，这叫一个**异常**。我开着车在路上，发动机坏了，我停车，这叫**错误**。系统处于不可恢复的崩溃状态。发动机什么时候坏？我们普通司机能管吗？不能。发动机什么时候坏是汽车厂发动机制造商的事。



Error:称为错误，由 Java 虚拟机生成并抛出，包括动态链接失败、虚拟机错误等，程序对其不做处理。Error 类层次描述了 Java 运行时系统内部错误和资源耗尽错误。这类错误是我们无法控制的，同时也是非常罕见的错误。所以在编程中，不去处理这类错误。Error 表明系统 JVM 已经处于不可恢复的崩溃状态中。我们不需要管他。 如: 写代码时突然断电，或者内存溢出。

Exception:所有异常的父类，其子类对应了各种各样可能出现的异常事件，一般需要用户显示地声明或捕获。

Runtime Exception

Runtime Exception 类是 Exception 类的子类，叫做运行时异常，Java 中的所有运行时异常都会直接或者间接地继承自 RuntimeException 类。

这一类特殊的异常，如被 0 除、数组下标超范围等，其产生比较频繁，处理麻烦，如果显示的声明或捕获将会对程序可读性可运行效率影响很大。因此由系统自动检测并将它们交给缺省的异常处理程序（用户可不必对其处理）。

我们可以通过程序的健壮性来处理，不推荐使用异常处理机制来处理。

例如：

NullPointerException: 当程序访问只有引用没有对象的成员属性或成员方法。怎么处理？

ArithmeticException: 除数为 0

ClassCastException:多态时类型转换错误

ArrayIndexOutOfBoundsException: 访问的元素下表超过数组长度

NumberFormatException: 数字格式异常！

CheckException

Java 中凡是继承自 Exception，而不继承自 RuntimeException 类的异常都是非运行时异常，也叫**检查时异常**。 如:IOException。 **必须要对其进行处理，否则无法通过编译**

这类异常的产生不是程序本身的问题，通常由外界因素造成的。 为了预防这些异常产生时，造成程序的中断或得到不正确的结果，Java 要求编写可能产生这类异常的程序代码时，一定要去做异常的处理。

三、 异常产生

之所以出现异常，是因为内部抛出了异常对象，这个异常对象的产生分为系统内部产生，或程序员手动抛出异常。

```
public static void test() throws Exception{
        System.out.println("Hello World");
        // 抛出异常
        throw new Exception();
}
```

四、 异常处理和捕获

对于编译（非运行）时异常（ checked exception ），必须要对其进行处理，否则无法通过编译。处理方式有两种：

1 、异常捕获 try catch finally

```
try{
        // 可能发生异常的代码
        // 如果发生了异常，那么异常之后的代码都不会被执行
}catch (XXxException e){
      // 异常处理代码
}catch (Exception e){
      // 异常处理代码
}finally{
      // 不管有没有发生异常，finally 语句块都会被执行
}
```

try 代码段包含可能产生例外的代码

try 代码段后跟有一个或多个 catch 代码段

每个 catch 代码段声明其能处理的一种特定类型的异常并提供处理的方法

当异常发生时，程序会中止当前的流程，根据获取异常的类型去执行相应的 catch 代码段

一个 try 后面可以跟多个 catch，但不管多少个， 最多只会有一个 catch 块被执行。

finally 段的代码无论是否发生异常都有执行

try 语句

try{…}语句制定了一段代码，该段代码就是一次捕获并处理意外的范围。

在执行过程中，该段代码可能会产生并抛出一种或几种类型的异常对象，它后面的 catch 语句要分别对这些异常做相应的处理。

如果没有意外产生，所有的 catch 代码段都被略过不执行。

catch 语句

在 catch 语句块中是对异常进行处理的代码，每个 try 语句块可以伴随一个或多个 catch 语句，用于处理可能产生的不同类型的异常对象。

在 catch 中声明的异常对象封装了异常事件发生的信息，在 catch 语句块中可以使用这个对象的一些方法获取这些信息

例如：

getMessage()方法，用来得到有关异常事件的信息

printStackTrace()方法，用来跟踪异常事件发生时执行堆栈的内容

finally 语句

finally 语句为异常处理提供一个**统一的出口**，使得在控制流程转到程序的其它部分以前，能够对程序的状态作统一的管理。

无论 try 所制定的程序块中是否抛出异常，finally 所指定的代码都要被执行。

通常在 finally 语句中可以进行资源的清除工作，如：

关闭打开的文件

删除临时文件

2、 向外 声明(抛出) 异常 throws

```
public static void test() throws Exception{
        // 可能发生异常的代码
       // 如果发生了异常，那么异常之后的代码都不会被执行
 }
```

在产生异常的方法声明后面写上 throws 某一个 Exception 类型，如 throws Exception，将异常抛出到外面一层去

异常与重写

子类声明的异常范围不能超出父类的异常范围

```
class P{
    public void test() throws IOException,SQLException{

    }
}
class C extends P{
    public void test() throws Exception,SQLException{

    }
}
```

五、 使用自定义的异常

所谓自定义异常，通常就是定义一个类，去继承 Exception 类或者它的子类。因为异常必须直接或者间接地继承自 Exception 类。通常情况下，会直接继承自 Exception 类，一般不会继承某个运行时的异常类。

```
public class MyException extends Exception{
    public MyException(){
        super();
    }
    public MyException(String message){
        super(message);
    }
}
```

使用自定义异常一般有如下步骤：

1、 通过继承 java.lang.Exception 类声明自己的异常类

2、 在方法适当的位置生成自定义异常的实例，并用 throw 语句抛出

3、 在方法声明部分用 throws 语句声明该方法可能抛出的异常

六、 总结

一个图: 异常体系

五个关键字： try catch finally throws throw

先逮小的，再逮大的

异常和重写的关系

思考作业

1、java 中异常处理的两种方式是？

2、如果碰到 NullPointerException,我们一般应该如何查错？如何做处理？

3、Error 和 Exception 的联系与区别

4、RuntimeException 与 CheckedException 的区别是什么？

5、RuntimeException 应该如何处理?

6、CheckedException 应该如何处理?

7、Throwable 是一个类还是一个接口？

8、如果 catch 是两个异常，一个是父类、一个是子类。这两个异常的 catch 顺序如何确定？

9、finally 里面一般放置什么代码？

10、eclipse 中，增加 try-catch 块的操作时怎么做的？

11、自定义一个异常类，如用户未找到

12、方法重写时，子类声明异常能否超出父类的范围？

13、下面的代码，有什么问题？请解释原因：

```
class A{
        public void method() throws IOException{
        }
}
class B extends A{
        public void method() throws Exception{
        }
}
```

14、思考 方法终止|结束 有几种形式?

15、说出现在接触的几个异常，尽量多写。

16.以下程序片段

```
public class Main{
	public static void main(String[] args){
		try{
			int num =Integer.parseInt(args[0]);
			System.out.println(num);
		}catch(NumberFormatException ex){
			System.out.println("必须输入数字");
		}
	}
}
```

执行时没有指定命令行参数，发生何种情况()

a.编译错误

b.显示“必须输入数字”

c.显示 ArrayIndexOutOfBoundsException 异常信息

d 不显示任何信息

17.

```
public class Main{
	public static void main(String[] args){
		try{
			Object obj = "12";
			Integer i =(Integer)obj;
			System.out.println(i);
		}catch(NumberFormatException ex){
			System.out.println("必须输入数字");
		}
	}
```

}

描述正确是()

a.编译错误

b.显示 12

c 显示 ClassCastException 异常信息

d 不显示任何信息

18.public class Main{

```
    public static void main(String[] args){
        try{
            int num =Integer.parseInt(args[0]);
            System.out.println(num++);
        }catch(NumberFormatException ex){
            System.out.println("必须输入数字");
        }
    }
}
```

执行时指定命令行参数为 second，发生何种情况()

a.编译错误

b.显示“必须输入数字”

c.显示 ArrayIndexOutOfBoundsException 异常信息

d 不显示任何信息

寄语

学习这件伟大的事业不是靠力气、

速度和身体的敏捷完成的；

而是靠性格、意志和知识的力量完成的。

谢谢大家