Java 应用与开发
异常处理

王晓东
wangxiaodong@ouc.edu.cn

中国海洋大学

October 30, 2018

学习目标

1. 掌握 Java 异常的概念和分类
2. 深入理解 Java 异常处理机制

大纲

异常的概念及分类

Java 异常处理机制

C++ 中的异常处理

❖ 《The C++ Programming Language》

- 一个库的作者可以检测出发生了运行时错误，但一般不知道怎样去处理它们（因为和用户具体的应用有关）；
- 另一方面，库的用户知道怎样处理这些错误，但却无法检查它们何时发生（如果能检测，就可以再用户的代码里处理了，不用留给库去发现）。

☛ 提供异常处理机制的的基本思想

让一个函数在发现了自己无法处理的错误时抛出（throw）一个异常，然后它的（直接或者间接）调用者能够处理这个问题。

❖ 《C++ primer》

将问题检测和问题处理相分离。
(Exceptions let us separate problem detection from problem resolution.)

C++ 中的异常处理

❖ 《The C++ Programming Language》

- 一个库的作者可以检测出发生了运行时错误，但一般不知道怎样去处理它们（因为和用户具体的应用有关）；
- 另一方面，库的用户知道怎样处理这些错误，但却无法检查它们何时发生（如果能检测，就可以再用户的代码里处理了，不用留给库去发现）。

☛ 提供异常处理机制的的基本思想

让一个函数在发现了自己无法处理的错误时抛出（throw）一个异常，然后它的（直接或者间接）调用者能够处理这个问题。

❖ 《C++ primer》

将问题检测和问题处理相分离。
(Exceptions let us separate problem detection from problem resolution.)

C++ 中的异常处理

❖ 《The C++ Programming Language》

- 一个库的作者可以检测出发生了运行时错误，但一般不知道怎样去处理它们（因为和用户具体的应用有关）；
- 另一方面，库的用户知道怎样处理这些错误，但却无法检查它们何时发生（如果能检测，就可以再用户的代码里处理了，不用留给库去发现）。

☛ 提供异常处理机制的的基本思想

让一个函数在发现了自己无法处理的错误时抛出（throw）一个异常，然后它的（直接或者间接）调用者能够处理这个问题。

❖ 《C++ primer》

将问题检测和问题处理相分离。
(Exceptions let us separate problem detection from problem resolution.)

C++ 中的异常处理

❖ 《The C++ Programming Language》

- 一个库的作者可以检测出发生了运行时错误，但一般不知道怎样去处理它们（因为和用户具体的应用有关）；
- 另一方面，库的用户知道怎样处理这些错误，但却无法检查它们何时发生（如果能检测，就可以再用户的代码里处理了，不用留给库去发现）。

☛ 提供异常处理机制的的基本思想

让一个函数在发现了自己无法处理的错误时抛出（throw）一个异常，然后它的（直接或者间接）调用者能够处理这个问题。

❖ 《C++ primer》

将问题检测和问题处理相分离。
(Exceptions let us separate problem detection from problem resolution.)

接下来…

异常的概念及分类

Java 异常处理机制

什么是异常

在 Java 语言中，程序运行出错被称为出现异常（Exception）。异常是**程序运行过程**中发生的事件，该事件可以中断程序指令的正常执行流程。

❖ Java 异常分为两大类

1. 错误（Error）是指 JVM 系统内部错误、资源耗尽等严重情况。
2. 违例（Exception）则是指其他因编程错误或偶然的外在因素导致的一般性问题，例如对负数开平方根、空指针访问、试图读取不存在的文件以及网络连接中断等。

☛ 小示例

课程配套代码 ◗ sample.exception.FirstExceptionSample.java

什么是异常

在 Java 语言中，程序运行出错被称为出现异常（Exception）。异常是**程序运行过程**中发生的事件，该事件可以中断程序指令的正常执行流程。

❖ Java 异常分为两大类

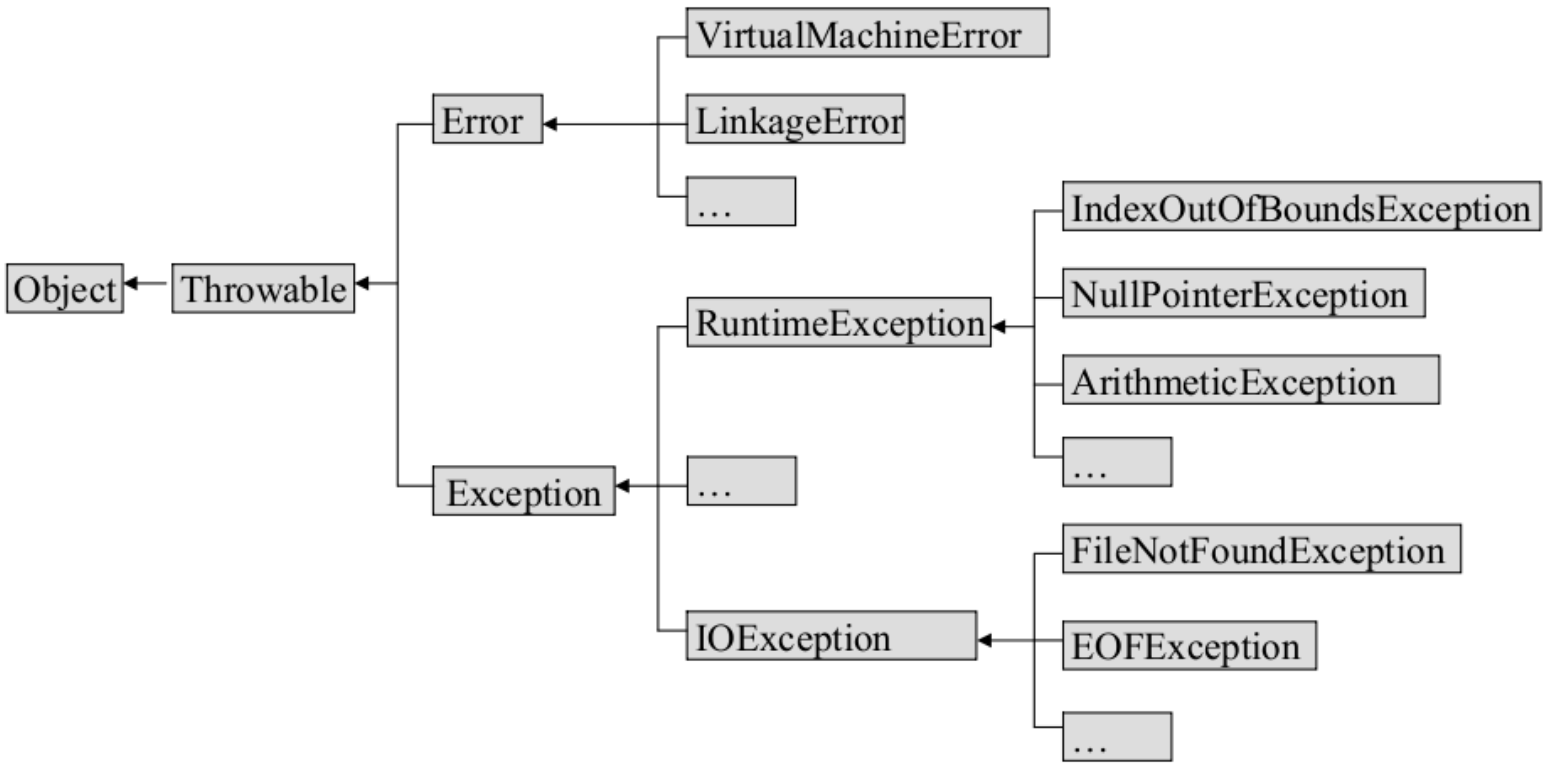
1. **错误**（Error）是指 JVM 系统内部错误、资源耗尽等严重情况。
2. **违例**（Exception）则是指其他因编程错误或偶然的外在因素导致的一般性问题，例如对负数开平方根、空指针访问、试图读取不存在的文件以及网络连接中断等。

☛ 小示例

课程配套代码 ➧ sample.exception.FirstExceptionSample.java

Java 异常分类

Throwable 类是 Java 语言中所有异常类的父类。



常见错误

❖ 链接错误（LinkageError）

是指程序链接错误。例如，一个类中用到另外一个类，在编译前一个类之后，后一个类发生了不相容的改变时，再使用前一个类则会出现链接错误。最常见的就是后一个类的.class 文件被误删除。

❖ 虚拟机错误（VirtualMachineError）

当 Java 虚拟机崩溃或资源耗尽时会抛出该错误。其中比较有代表性的是 StackOverflowError，当应用程序递归太深而导致栈内存溢出时会出现该异常。

课程配套代码 ➧ sample.exception.VMErrorSample.java

常见错误

❖ 链接错误（LinkageError）

是指程序链接错误。例如，一个类中用到另外一个类，在编译前一个类之后，后一个类发生了不相容的改变时，再使用前一个类则会出现链接错误。最常见的就是后一个类的.class 文件被误删除。

❖ 虚拟机错误（VirtualMachineError）

当 Java 虚拟机崩溃或资源耗尽时会抛出该错误。其中比较有代表性的是 StackOverflowError，当应用程序递归太深而导致栈内存溢出时会出现该异常。

课程配套代码 ➧ sample.exception.VMErrorSample.java

常见异常

❖ RuntimeException

- 错误的类型转换
- 数组下标越界
- 空指针访问

空指针异常（NullPointerException）

如果试图访问不指向任何对象的引用变量的成员，将会产生空指针异常。

```
1     Person p = null;
2     System.out.println(p.age);
```

常见异常

❖ IOException

- 从一个不存在的文件中读取数据
- 越过文件结尾继续读取
- 连接一个不存在的 URL

☛ IOException 示例[1]

课程配套代码 ◗ sample.exception.IOExceptionSample.java

[1] 上述代码无法编译：只要是有可能出现 IOException 的 Java 代码，在编译时就会出错，而不会等到运行时才发生。

接下来…

异常的概念及分类

Java 异常处理机制

Java 异常处理的原则

- 返回到一个安全和已知的状态
- 能够使用户执行其他的命令
- 如果可能，则保存所有的工作
- 如果有必要，可以退出以避免造成进一步的危害

Java 异常处理机制

- Java 程序执行过程中如出现异常，系统会监测到并自动生成一个相应的异常类对象，然后再将它交给运行时系统。
- 运行时系统再寻找相应的代码来处理这一异常。如果 Java 运行时系统找不到可以处理异常的代码，则运行时系统将终止，相应的 Java 程序也将退出。
- 程序员通常对错误（Error）无能为力，因而一般只处理违例 (Exception)。

Java 异常处理机制

- Java 程序执行过程中如出现异常，系统会监测到并自动生成一个相应的异常类对象，然后再将它交给运行时系统。
- 运行时系统再寻找相应的代码来处理这一异常。如果 Java 运行时系统找不到可以处理异常的代码，则运行时系统将终止，相应的 Java 程序也将退出。
- 程序员通常对错误（Error）无能为力，因而一般只处理违例 (Exception)。

Java 异常处理机制

- Java 程序执行过程中如出现异常，系统会监测到并自动生成一个相应的异常类对象，然后再将它交给运行时系统。
- 运行时系统再寻找相应的代码来处理这一异常。如果 Java 运行时系统找不到可以处理异常的代码，则运行时系统将终止，相应的 Java 程序也将退出。
- 程序员通常对错误（Error）无能为力，因而一般只处理违例 (Exception)。

异常处理结构

❖ 异常处理结构

```
1     try {
2       ... //可能产生异常的代码，试图捕获异常

4     } catch (ExceptionName1 e) {
5       ... //当产生 ExceptionName1 型异常时的处置措施

7     } catch (ExceptionName2 e) {
8       ... //当产生 ExceptionName2 型异常时的处置措施

10    } finally {
11      ... //无条件执行的语句

13    }
```

课程配套代码 ➧ sample.exception.HandleFirstExceptionSample.java

使用 finally 语句

- finally 语句是可选的
- 作用是为异常处理提供一个统一的出口，使得在控制流转到程序的其他部分以前，能够对程序的状态作统一的管理。
- 不论 try 代码块中是否发生了异常事件，finally 块中的语句都会被执行。当 catch 语句块中出现 return 语句时，finally 语句块同样会执行。

操作异常对象

发生异常时，系统将自动创建异常类对象，并将作为实参传递给匹配的 catch 语句块的形参，这样就可以在语句块中操纵该异常对象。

❖ 异常类的父类 Throwable 中定义的方法

getMessage() 返回描述当前异常的详细消息字符串。

printStackTrace() 用来跟踪异常事件发生时运行栈的内容，并将相关信息输出到标准错误输出设备。本方法比较常用，在没有找到适合的异常处理代码时，系统也会自动调用该方法输出错误信息。

操作异常对象

发生异常时，系统将自动创建异常类对象，并将作为实参传递给匹配的 catch 语句块的形参，这样就可以在语句块中操纵该异常对象。

❖ 异常类的父类 Throwable 中定义的方法

getMessage() 返回描述当前异常的详细消息字符串。

printStackTrace() 用来跟踪异常事件发生时运行栈的内容，并将相关信息输出到标准错误输出设备。本方法比较常用，在没有找到适合的异常处理代码时，系统也会自动调用该方法输出错误信息。

捕获和处理 IOException

❖ 一些知识点

- **异常类型的多态性** FileNotFoundException 是 IOException 的子类，基于多态性机制，后一个 catch 语句也可以处理 FileNotFoundException，因此前一个 catch 语句块可以取消，但这样就无法区分“文件不存在”或其他 I/O 异常了。
- 运行时异常
- 过度处理

课程配套代码 ◗ sample.exception.IOExceptionSample.java

捕获和处理 IOException

❖ 一些知识点

- 异常类型的多态性
- 运行时异常 对于只可能产生 RuntimeException 的代码可以不使用 try-catch 语句进行处理，如果对于这些相对安全的代码仍然采用了 try 语句块的形式，则 try 后可以省略 catch 语句块或 finally 语句块，但不能同时省略。
- 过度处理

课程配套代码 ◗ sample.exception.IOExceptionSample.java

捕获和处理 IOException

❖ 一些知识点

- 异常类型的多态性
- 运行时异常
- **过度处理** 如果试图捕获和处理代码中根本不可能出现的异常，编译器也会指出这种不当行为。

课程配套代码 ◗ sample.exception.IOExceptionSample.java

声明抛出异常

❖ 声明抛弃异常是 Java 中处理 Exception 的第二种方式

- 一个方法中的代码在运行时可能生成某种异常，但在本方法中不必或者不能确定如何处理此类异常时，则可以声明抛弃该异常。
- 此时方法中将不对此类异常进行处理，而是由该方法的调用者负责处理。

课程配套代码 ◗ sample.exception.ThrowsExceptionSample.java

声明抛出异常

❖ 采用声明抛出异常的注意事项

- 除非事先约定，否则在开发过程中不要在自己编写的方法中采用抛出异常的方式。
- 重写方法不允许抛出比被重写方法范围更大的异常类型。例如 IOException 重写后抛出 FileNotFoundException 和 EOFException 被允许，而抛出 Exception 则不被允许。

人工抛出异常

Java 异常类对象除了在程序运行出错时由系统自动生成并抛出之外，也可根据需要人工创建并抛出：

```
1    IOException e = new IOException(); // 创建异常类对象
2    throw e; // 抛出操作，即将该异常对象提交给Java运行环境
```

被抛出的必须是 Throwable 或其子类类型的对象，下述语句在编译时会产生语法错误：

```
1    throw new String("want to throw");
```

课程配套代码 ➧ sample.exception.ManualThrowExceptionSample.java

人工抛出异常

❖ 选择人工抛出异常的原则

- 当明确知道可能出错的地方或能够通过简单的检查而有效防止错误发生，就应该使用 if-else 语句来预防错误发生。
- 只有当我们无法明确知道错误发生之处或无法完全避免异常，才不得不通过异常处理的方式来捕获和处理异常。
- 自定义的异常类对象只能采用人工方式抛出。（自定义异常有兴趣请自行搜索资料学习）

本节习题

1. 总结 Java 的异常处理机制。
2. 什么是运行时异常？
3. 若 try 语句结构中有多个 catch() 子句，这些子句的排列顺序与程序执行效果是否有关？
4. 总结 Java 异常处理机制随 Java 版本的更新不断加入的新特性，并附参考文献或网站链接。（选做）

THE END

wangxiaodong@ouc.edu.cn