本章内容

- Java 异常概念
- Java 异常的分类
- 异常的捕获和处理

运行期出现的错误

观察错误的名字和行号最重要

异常的概念

- Java 异常是Java提供的用于处理程序中错误的一种机制。
- 所谓错误是指在程序运行的过程中发生的一些异常事件（如：除0溢出，数组下标越界，所要读取的文件不存在）。
- 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产生不可预见的结果。
- Java程序的执行过程中如出现异常事件，可以生成一个异常类对象，该异常对象封装了异常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。

```
public class Test {
    public static void main(String[] args) {
        String friends[] = { "Tom", "John", "Jenni" };
        for (int i = 0; i < 4; i++) {
            System.out.println(friends[i]);
        }
        System.out.println("\nthis is the end");
    }
}
```

异常的概念

```
public void someMethod()
                        throws SomeException {
    if (someCondition()) {
        throw new SomeException("错误原因");
  }
        … … …
}
```

声明该方法可能抛出的异常

构造并抛出异常对象

调用该方法时试图捕获异常

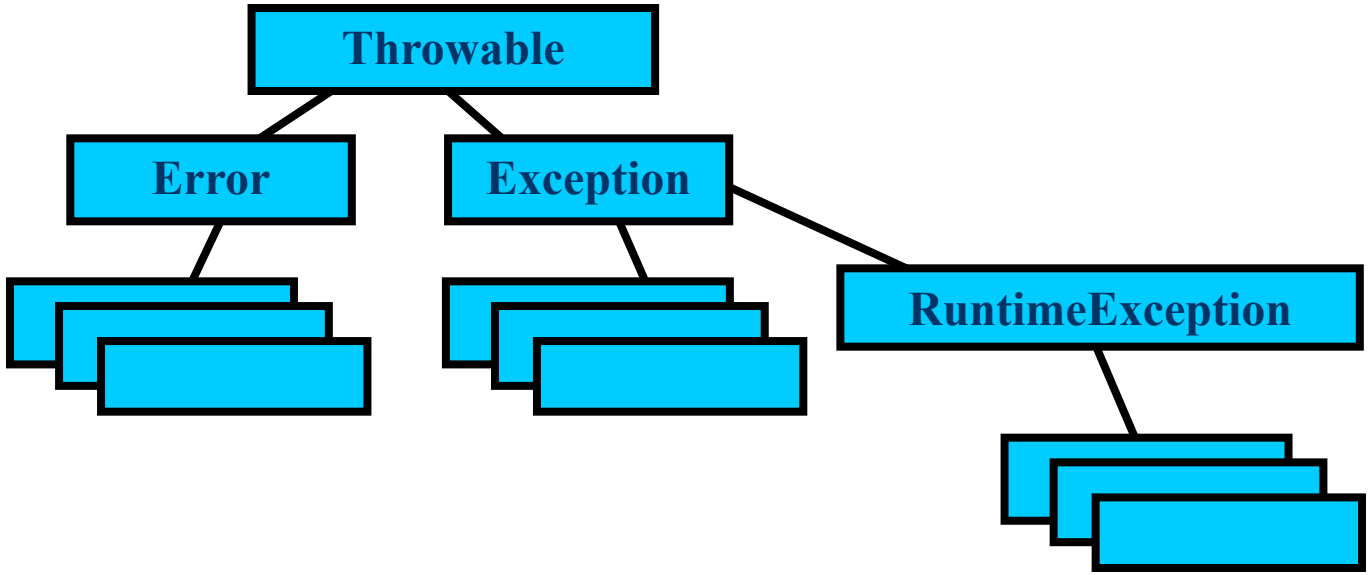
```
        … … …
try {
    someMethod();
} catch (SomeException e) {
    //异常处理代码;
}
        … … …
```

定义处理异常的代码

方法是可能抛出异常的

异常的分类

- J2SDK 中定义了很多异常类，这些类对应了各种各样可能出现的异常事件。



TestEx.java

异常的分类

- 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。
- Exception：所有异常类的父类，其子类对应了各种各样可能出现的异常事件，一般需要用户显式的声明或捕获。
- Runtime Exception：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，如果显式的声明或捕获将会对程序可读性和运行效率影响很大。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）。

- **Exception (in java.lang)**
  - **ClassNotFoundException**
    IOException
    InterruptedException
    RuntimeException
    - **ArithmeticException NullPointerException**
    - **IndexOutOfBoundsException**
      - **ArrayIndexOutObBoundsException**
      - **StringIndexOutObBoundsException … …**

异常的捕获和处理

```
try {
//可能抛出异常的语句
} catch ( SomeException1 e)
{
    … … …
} catch ( SomeException2 e)
{
    … … …
} finally {
    … … …
}
```

- 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。
- finally段的代码无论是否发生异常都有执行。

异常的捕获和处理

➢捕获SomeException2时：

```
try {
   语句 1;
   语句 2;
}
catch ( SomeException1 e）
{ … … …}
catch ( SomeException2 e）
{ … … …}
finally { … … …}
  后面的语句;
```

➢没有捕获到异常时：

```
try {
   语句 1;
   语句 2;
}
catch ( SomeException1 e）
{ … … …}
catch ( SomeException2 e）
{ … … …}
finally { … … …}
  后面的语句;
```

异常的捕获和处理

- try语句
  - try {...}语句指定了一段代码，该段代码就是一次捕获并处理例外的范围。
  - 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。
  - 如果没有例外产生，所有的catch代码段都被略过不执行。

异常的捕获和处理

- 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（catch（SomeException e））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。

异常的捕获和处理

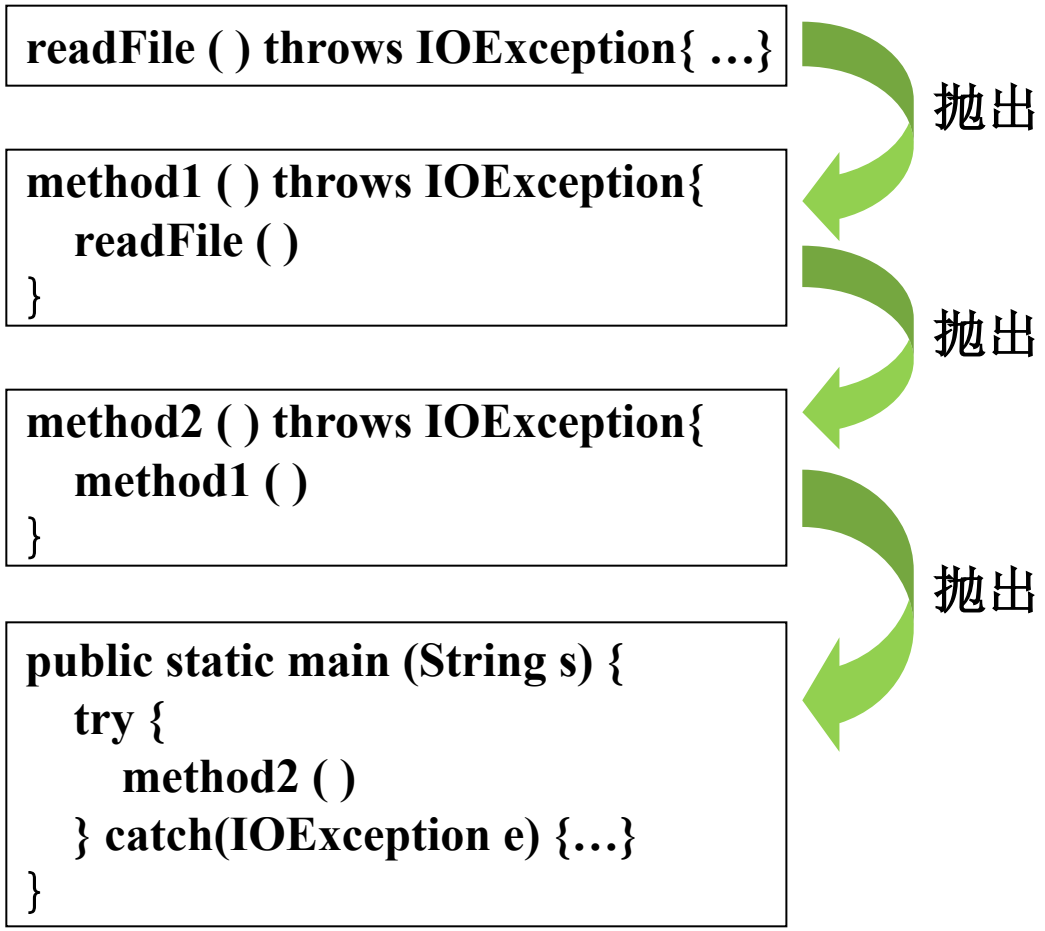
- finally 语句
  - finally语句为异常处理提供一个统一的出口，使得在控制流程转到程序的其他部分以前，能够对程序 的状态作统一的管理。
  - 无论try所指定的程序块中是否抛出例外，finally所指定的代码都要被执行。
  - 通常在finally语句中可以进行资源的清除工作，如：
    - ✓关闭打开的文件
    - ✓删除临时文件
    - ✓…

异常的捕获和处理

```
public class Test {
    public static void main(String[] args) {
        FileInputStream in = null;
        try {
            in = new FileInputStream("myfile.txt");
            int b;
            b = in.read();
            while (b != -1) {
                System.out.print((char) b);
                b = in.read();
            }
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } finally {

            … … … //关闭文件操作
        }
    }
}
```

异常的捕获和处理



- Java的例外处理机制使得例外事件，沿着被调用的顺序往前寻找，只要找到符合该例外种类的例外处理程序.

异常的捕获和处理

```
public class Test {
    public static void main(String args[]) {
        String[][] s = new String[5][];
        try {
            s[5] = new String[3]; s[5][1] = "hello";
        } catch( Exception e2){
            System.out.println("数组下标越界了");
        } catch ( ArrayIndexOutOfBoundsException e1) {
            System.out.println("有异常发生了");
        }
    }
}
```

会有编译错误，这个异常不会被捕获，在一个try语句块中，基类异常的捕获语句不可以写在子类异常捕获语句的上面。

使用自定义的异常

- 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. 在方法的声明部分用throws 语句声明该方法可能抛出的异常。

Test.java

```
class MyException extends Exception {
    private int id;
    public MyException(String message,int id) {
        super(message);
        this.id = id;
    }
    public int getId() {
        return id;
    }
}
```

总结

- 一个图
- 五个关键字
- 先逮小的，再逮大的
- 异常和重写的关系