LightHouseIT

2.5 Java 异常处理

- 异常的基本概念
- Java 中的异常
- 异常的处理

目录

异常

- **异常**[例外]是指程序运行过程中出现的不正常或不符合预期的、需要特殊处理的情况。
- 需要区别异常与**语法错误**[構文エラー]： 语法错误在编译时出现；而异常出现在运行时。
- 一部分异常必须在程序中进行处理。否则，程序会无法继续运行而强制终止。
- 出现语法错误就表示程序写错了，必须通过修正代码来解决。而出现异常**可能**是程序写错了；也**可能**是程序的运行环境出现了问题，此时需要提前判断进行*处理*。

异常的产生和处理

- 为了防止出现致命错误导致程序提前终止，编程时需要考虑可能出现的特殊输入或者环境问题，并**抛出**[スロー]（产生）异常。

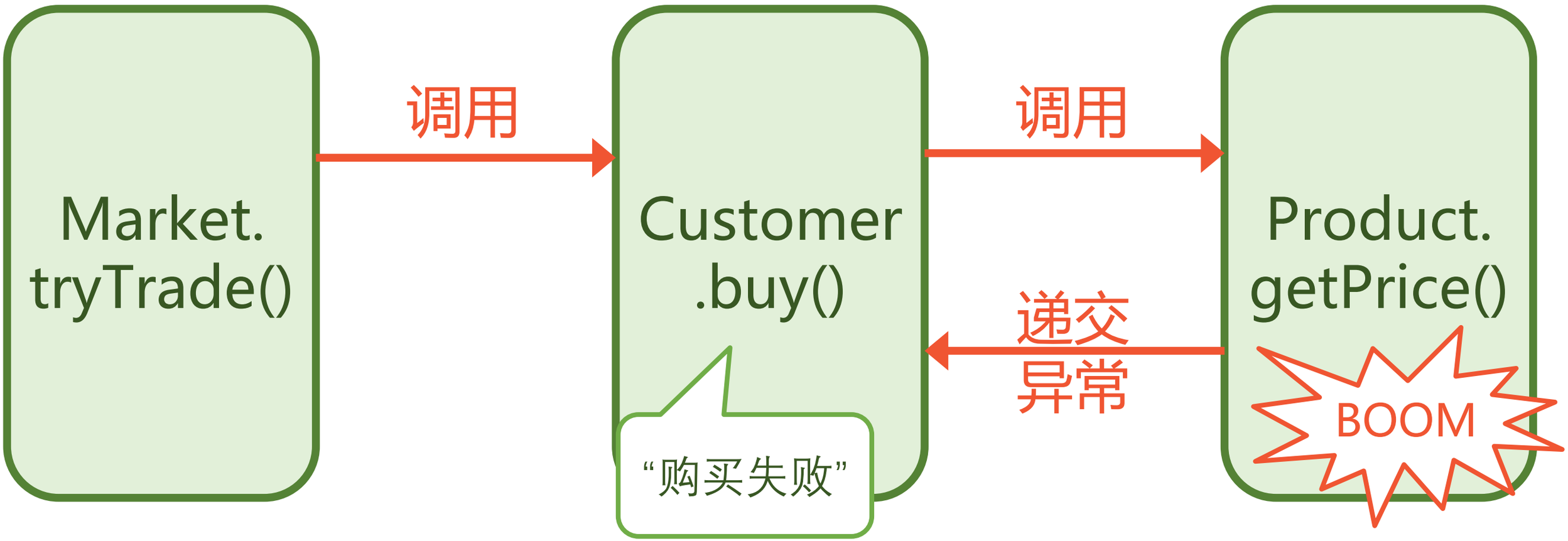
- 而在异常可能出现的位置，一般有两种**处理**异常的方法：
  1. 当你知道应对异常的办法时，直接在当前*方法*里编写代码**解决**问题。
  2. 当你不知道应对异常的办法时，将异常**交给**调用当前*方法*的*方法*处理。

接次页

接前页

Example

在交易系统中，顾客交易时可能出现商品的价格没有被设定的问题。



Question and answer

LightHouseIT

Java 中的异常

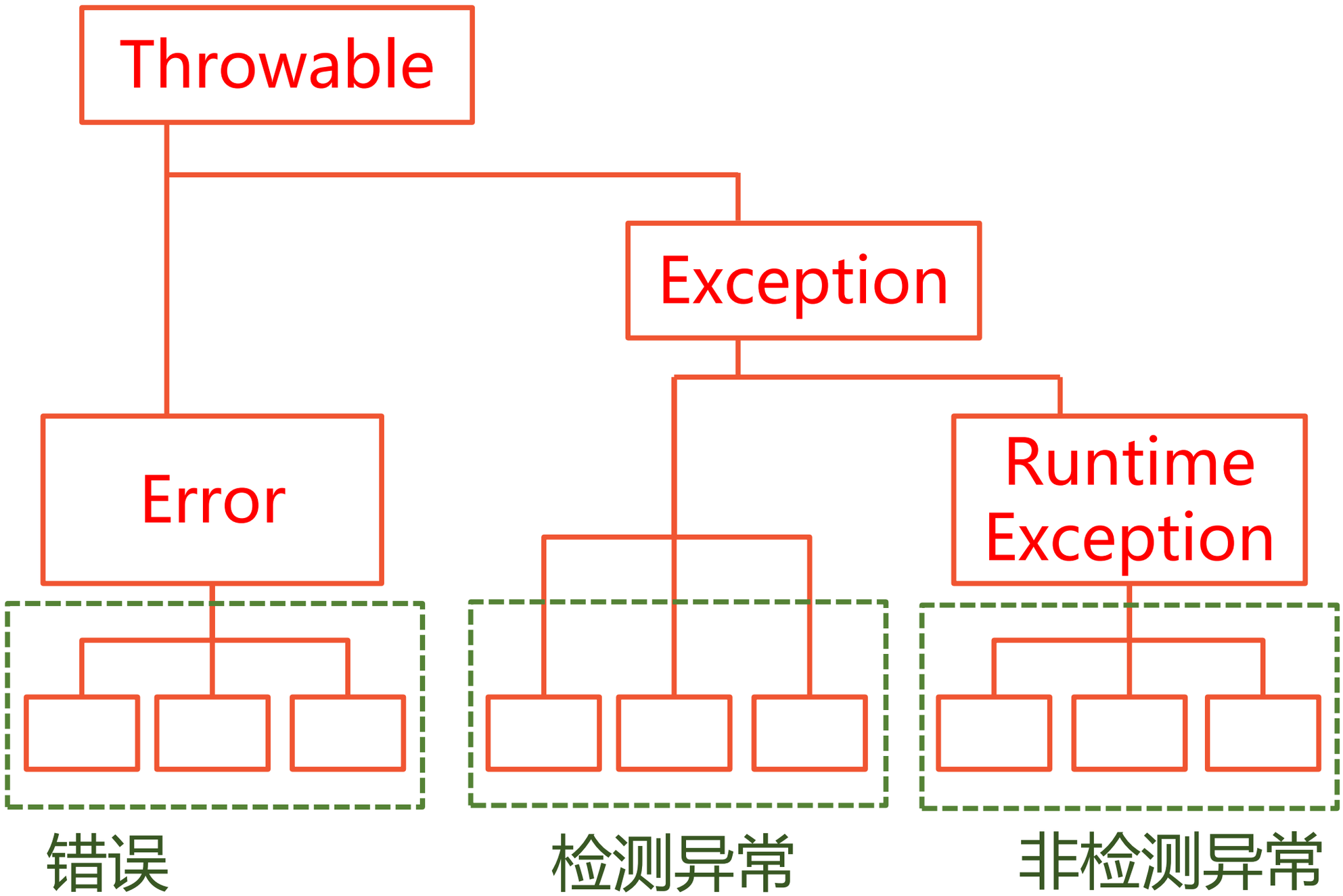
- Java 中的所有异常都由一个 Java 类表示。所有异常类都继承 Throwable 类。

- Java 中的异常大体可以分为三种：**错误**[エラー]（Error）、**非检测异常**[非チェック例外]（Unchecked Error）、**检测异常**[チェック例外]（Checked Error）。

- 下图展示了它们之间的关系：

接次页

接前页



错误

- **错误**[エラー] 是指运行时会出现的，由于系统或 Java 本身出现的异常问题。这种类型的异常通常是**致命**的，因此**无法**也**无需**在代码中*处理*。

Example

内存不足 OutOfMemoryError
堆栈溢出 StackOverflowError

- 错误类的名称都以 “Error” 结尾。
- 我们只能**修改代码**或者**调整环境**以避免错误的出现。

非检测异常

- **非检测异常**[非チェック例外]或**实行时异常**[実行時例外]是指比较基本的、经常会出现的一类异常。

- 很多基本的语法都有可能产生非检测异常，比如数组元素的操作、除法运算或对象方法调用等。如果这些异常全部都要处理，那代码会变得非常冗长复杂，因此非检测异常**不需要**在代码中被*处理*。

- 另一方面，非检测异常的出现很有可能是程序本身的逻辑错误或者考虑不周全，我们需要**修改代码**以避免异常出现。

- 非检测异常类的名称都以 “Exception” 结尾，并且均继承 RuntimeException 类。

常见的非检测异常

- 下表列出一些常见的非检测异常:

异常类型	类名	例子
算术异常	ArithmeticException	`double a = 1 / 0;`
数组索引超出范围异常	ArrayIndexOutOfBoundsException	`int arr = new int[3];` `arr[3] = 1;`
空指针异常	NullPointerException	`String a = null;` `int b = a.length();`
不支持功能异常	UnsupportedOperationException	`List list = List.of(1, 2);` `list.add(3);`
负数组大小异常	NegativeArraySizeException	`Int[] a = new int [-2];`

检测异常

- **检测异常**[チェック例外]就是除了刚刚两种特殊情况外的一般异常。
- *检测异常*可能出现在任何实际操作的过程，比如文件读写、网络通信或数据库查询等。
- 如果你没有在检测异常可能产生的位置*处理*它，就会产生语法错误，程序将无法运行。
- 检测异常类的名称都以 “Exception” 结尾，并且**不**继承 RuntimeException 类。
- 常见的检测异常类包括 FileNotFoundExeption，FileIOException 等。我们会在接下来的学习过程中还遇到很多这样的异常。

不同异常 / 错误类型总结

类型	父类	出现时机	处理办法
语法错误	-	编译时	修改代码
错误	Error	运行时	修改代码或调整环境
非检测异常	Runtime Exception	运行时	修改代码或异常处理
检测异常	Exception	运行时	异常处理

Question and answer

目录

Java 中的异常处理

- 在 Java 中，很多自带的类和方法都可能产生一些异常。同时，我们之后可能使用到的一些外部包也会产生异常（一般都是检测异常）。你也可以在自己代码中手动产生异常。

- 现阶段我们需要先记住当异常可能出现时，应该如何处理。

- Java 中的异常处理有两种选择：
  - try-catch 语句，用于在当前方法里**解决**问题。
  - throws 关键字，用于将异常**交给**调用当前方法的代码处理。

Note

再次强调：
检测异常必须被处理；非检测异常可以被处理。

try-catch 语句

- 先看句法:

```
1 try {
2     codes;
3 } catch (Exception e) {
4     codes2;
5 }
```

- 在 codes 的位置写入可能**产生异常**的代码，在 codes2 的位置写入**解决问题**的代码。

用 try-catch 语句处理异常

- 下面的代码会产生一个 ArrayIndexOutOfBoundsException:

```
1 int[] arr = new int[10];
2 System.out.println(arr[10]);
```

- 我们可以把可能产生错误的代码放入 try-catch 语句中:

```
1 int[] arr = new int[10];
2 try {
3     System.out.println(arr[10]);
4 } catch (Exception e) {
5     System.out.println("Failed to print number");
6 }
```

Try
01011
11010
01011
Catch.java

多个 try 块

- 如果一段代码里可能出现多种异常，我们可能需要不同的代码以应对不同的异常类型。
- 句法：

```
1 try {
2     codes;
3 } catch (Exception1 e) {
4     codes2;
5 } catch (Exception2 e) {
6     codes3;
7 } catch (Exception3 e) {
8     codes4;
9 }
```

- 其中，Exception1、Exception2 和 Exception3 是不同的异常类。

多重捕获

- 也可以用同一个 catch 块处理多种异常，只要将多个异常类型以 “|” 隔开书写。这被称为**多重捕获**[マルチキャッチ]：

```
1 try {
2     codes;
3 } catch (Exception1 | Exception2 e) {
4     codes2;
5 } catch (Exception3 e) {
6     codes3;
7 } catch (Exception4 e) {
8     codes4;
9 }
```

try-catch 语句的执行顺序

- 要注意 try-catch 语句的执行顺序：如果在执行 try 块时出现了异常，try 块会立刻结束。随后，只执行和异常类型相符的一个 catch 块中的语句。

- 如果执行 try 块时没有出现异常，代码会正常结束，不会有 catch 块被执行。

finally 语句

- 有时，我们希望不管异常是否发生，都执行某些语句。
- 比如，我们会在 try 块中将把某些文件资源放入内存。我们希望代码结束时，无论是否发生异常，这些资源都被释放出去。
- 此时，可以使用 **finally** 语句：

```
1 try {
2     codes;
3 } catch (Exception e) {
4     codes2;
5 } finally {
6     codes3;
7 }
```

finally 语句的执行顺序

- 无论何种情况，finally 中的语句都一定会在 try-catch 语句结束后执行。
- 即使 try 或 catch 语句使用 return 语句立刻终止方法，finally 块中的语句也会被执行（会在返回值被使用前执行）。

- 还有一种确保释放资源的方法是 try-with-resource 语句。我们会在之后实际使用到时（§ 3.3.3）介绍。

throws 关键字

- 当我们不知道当前方法中如何解决异常时，可以用 **throws** 关键字将其交（“抛”）给调用者处理。
- 在方法定义的括号 “)” 后加上 throws 关键字和异常类型:

```
1 public static void test() throws ArrayIndexOutOfBoundException {
2     int[] arr = new int[10];
3     arr[10] = 100;
4 }
```

- 你也可以写上多个异常类型，用逗号隔开:

```
void test() throws IOException, ArithmeticException {}
```

throws 关键字的运行机制

- 如果带有 throws 关键字的某个方法中出现了异常，整个方法会立即终止。同时，异常会出现在调用这个方法的地方。
- 因此，添加了 throws 关键字的方法本身可能产生对应的异常。我们需要用同样的方式（try-catch 或 throws）进行*处理*。

throw 语句

- 当你想要手动**抛出**（产生）一个异常时，可以使用 **throw** 语句：

```
throw new Exception();
```

- 其中，Exception 可以换成任何其他异常类。你也可以通过继承 Exception 定义自己的异常类来使用。

- 注意辨别它和 throws 语句。

Question and answer

总结

Sum Up

1. 异常的定义和异常处理的思想：原地解决或交给调用者处理。

2. Java 中异常的种类：
   ① 错误：无法处理。
   ② 非检测异常：可以不处理。

3. Java 中异常处理的方法：
   ① 原地解决：try-catch-finally 语句。
   ② 交给调用者：throws 关键字。

LightHouseIT

THANK YOU!