

# Java 异常处理

---

[前言](#)

[抛出异常](#)

[捕获异常](#)

[处理异常](#)

[finally块](#)

[总结](#)

## 前言


当Java程序中出现错误或异常时，通常会抛出一个异常。Java的异常处理机制使得我们可以在程序运行过程中捕获这些异常并采取相应的措施，以便程序能够正常运行或者优雅地停止。

## 抛出异常

在Java中，当程序发生错误或异常时，可以使用`throw`关键字抛出一个异常。例如：

```
1  if (x == 0) {  
2      throw new ArithmeticException("除数不能为0");  
3  }
```

Java

 复制代码

## 捕获异常

在Java中，我们可以使用`try-catch`语句来捕获异常。在`try`块中编写可能会引发异常的代码，在`catch`块中编写处理异常的代码。例如：

```
1 try {  
2     //可能会抛出异常的代码  
3 } catch (ExceptionType e) {  
4     //处理异常的代码  
5 }
```

其中，`ExceptionType`表示捕获的异常类型，可以是Java中任何一个异常类的名称，如 `ArithmeticException`、`NullPointerException`等等。当在`try`块中的代码发生了与`ExceptionType`相匹配的异常时，程序会跳转到`catch`块中执行相应的代码。

## 处理异常

在`catch`块中，我们可以根据实际需要编写相应的处理代码，比如打印错误信息、重新抛出异常、继续执行其他代码等等。例如：

```
1 try {  
2     //可能会抛出异常的代码  
3 } catch (ArithmeticException e) {  
4     System.out.println("除数不能为0");  
5 } catch (NullPointerException e) {  
6     System.out.println("对象引用为空");  
7 } catch (Exception e) {  
8     System.out.println("未知异常: " + e.getMessage());  
9 }
```

在上面的例子中，我们通过捕获不同的异常类型来处理不同类型的异常，最后一个`catch`块则是处理所有其他未被上面的`catch`块捕获的异常。

## finally块

除了`try-catch`语句之外，Java还提供了`finally`块，用于编写在`try-catch`语句执行完毕之后一定要执行的代码，无论是否发生异常。例如：

```
1 try {  
2     //可能会抛出异常的代码  
3 } catch (ExceptionType e) {  
4     //处理异常的代码  
5 } finally {  
6     //一定会执行的代码  
7 }
```

在上面的例子中，无论**try-catch**块中的代码是否抛出异常，**finally**块中的代码都一定会被执行。在**finally**块中通常会放置释放资源等必须执行的代码。

## 总结

Java异常处理的基本流程是：先抛出异常，然后使用**try-catch**语句捕获异常并进行处理，最后使用**finally**块执行必须执行的代码。熟练掌握异常处理技