**try-with-resources**

JDK7 之后，Java 新增的 try-with-resource 语法糖来打开资源，并且可以在语句执行完毕后确保每个资源都被自动关闭 。

try-with-resources 是一种异常处理机制，它可以简化资源管理代码的编写。

JDK7 之前所有被打开的系统资源，比如流、文件或者 Socket 连接等，都需要被开发者手动关闭，否则将会造成资源泄露。

try (resource declaration) {

// 使用的资源

} catch (ExceptionType e1) {

// 异常块

}

以上的语法中 try 用于声明和实例化资源，catch 用于处理关闭资源时可能引发的所有异常。

注意：try-with-resources 语句关闭所有实现 AutoCloseable 接口的资源。

实例

import java.io.\*;

public class RunoobTest {

public static void main(String[] args) {

String line;

try(BufferedReader br = new BufferedReader(new FileReader("test.txt"))) {

while ((line = br.readLine()) != null) {

System.out.println("Line =>"+line);

}

} catch (IOException e) {

System.out.println("IOException in try block =>" + e.getMessage());

}

}

}

以上实例输出结果为：

IOException in try block =>test.txt (No such file or directory)

以上实例中，我们实例一个 BufferedReader 对象从 test.txt 文件中读取数据。

在 try-with-resources 语句中声明和实例化 BufferedReader 对象，执行完毕后实例资源，不需要考虑 try 语句是正常执行还是抛出异常。

如果发生异常，可以使用 catch 来处理异常。

再看下不使用 try-with-resources 而改成 finally 来关闭资源，整体代码量多了很多，而且更复杂繁琐了：

实例

import java.io.\*;

class RunoobTest {

public static void main(String[] args) {

BufferedReader br = null;

String line;

try {

System.out.println("Entering try block");

br = new BufferedReader(new FileReader("test.txt"));

while ((line = br.readLine()) != null) {

System.out.println("Line =>"+line);

}

} catch (IOException e) {

System.out.println("IOException in try block =>" + e.getMessage());

} finally {

System.out.println("Entering finally block");

try {

if (br != null) {

br.close();

}

} catch (IOException e) {

System.out.println("IOException in finally block =>"+e.getMessage());

}

}

}

}

以上实例输出结果为：

Entering try block

IOException in try block =>test.txt (No such file or directory)

Entering finally block

try-with-resources 处理多个资源

try-with-resources 语句中可以声明多个资源，方法是使用分号 ; 分隔各个资源：

实例

import java.io.\*;

import java.util.\*;

class RunoobTest {

public static void main(String[] args) throws IOException{

try (Scanner scanner = new Scanner(new File("testRead.txt"));

PrintWriter writer = new PrintWriter(new File("testWrite.txt"))) {

while (scanner.hasNext()) {

writer.print(scanner.nextLine());

}

}

}

}

以上实例使用 Scanner 对象从 testRead.txt 文件中读取一行并将其写入新的 testWrite.txt 文件中。

多个声明资源时，try-with-resources 语句以相反的顺序关闭这些资源。 在本例中，PrintWriter 对象先关闭，然后 Scanner 对象关闭。