

Lab 08 Assignment

班级：202111

学号：19241027

姓名：胡峻诚

Question1 Java中的检查型异常 (checked exception) 和非检查型异常 (unchecked exception) 有什么区别?

简答

- 检查型异常继承自 `Exception`，必须对其采用 `try...catch` 处理或者 `throws` 这个异常，接受 `throws` 的上级方法也必须对其处理，否则都不能通过编译。
- 非检查型异常继承自 `Runtime Exception` 或 `Error`，可以不用捕获，可以不在函数声明中添加 `throws` 语句，调用函数也不做强制处理，JVM 会自动处理。

Question2 简述Java异常处理中 `throws` 和 `throw` 关键字的作用。

简答

- `throws` 关键字写在方法声明后面，表示这个方法可能会出现对应的异常，但没有对其处理，而是抛出异常到上一级去处理。
- `throw` 常用于主动抛出自定义异常，即使编译器可能并不觉得这是个异常。程序会在 `throw` 语句后立即终止，它后面的语句执行不到，然后在包含它的所有 `try` 块中（可能在上层调用函数中）从里向外寻找含有与其匹配的 `catch` 子句的 `try` 块。

Question3 请列出2个常见的运行时异常和2个非运行时异常。

简答

- 运行时异常：`NullPointerException`（空指针），`IndexOutOfBoundsException`（下标越界）
- 非运行时异常：`IOException`，`SQLException`

Question4 指出下列程序的错误并改正。

改错

`RuntimeException` 是 `Exception` 的一个子类，`Exception` 已经将 `RuntimeException` 捕捉到，导致编译失败。

修改后的代码如下：

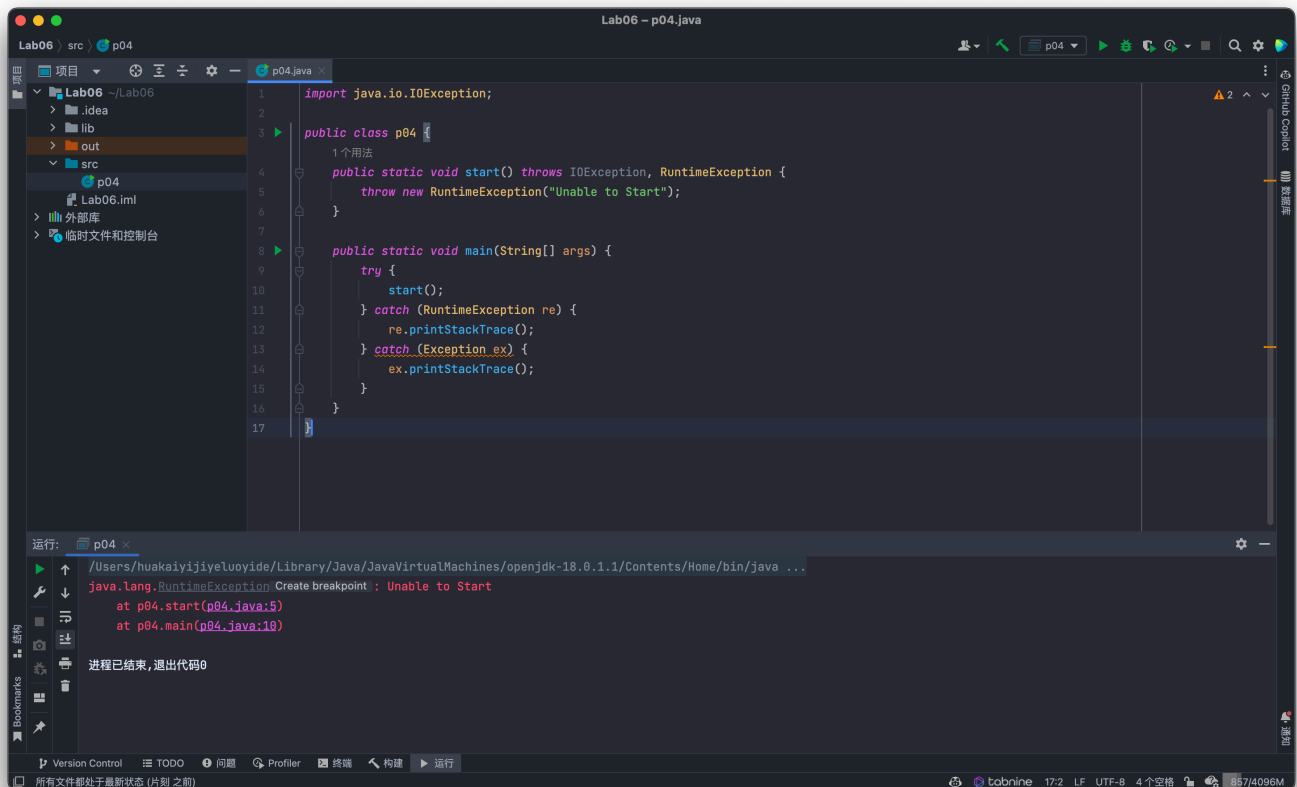
```
1 import java.io.IOException;
2
3 public class p04 {
4     public static void start() throws IOException, RuntimeException {
5         throw new RuntimeException("Unable to Start");
6     }
7
8     public static void main(String[] args) {
9         try {
```

```

10         start();
11     } catch (RuntimeException re) {
12         re.printStackTrace();
13     } catch (Exception ex) {
14         ex.printStackTrace();
15     }
16 }
17 }

```

修改后可以运行的截图如下：



Question5 指出下列程序的错误并改正。 改错

- SubClass 是 SuperClass 的继承，但是 SubClass 中的 start() 无法覆盖 SuperClass 中的 start()，原因是被覆盖的方法未抛出 java.lang.Exception 异常。
- SubClass 使用了 FileInputStream 函数，但是该函数必须考虑 java.io.FileNotFoundException 异常，必须对其进行捕获或声明以便抛出。

修改后的代码如下：

```

1 //SuperClass.java
2
3 import java.io.IOException;
4
5 public class SuperClass {

```

```
6     public void start() throws IOException {
7         throw new IOException("Unable to start");
8     }
9 }
10
11 //SubClass.java
12
13 import java.io.FileInputStream;
14 import java.io.FileNotFoundException;
15 import java.io.IOException;
16
17 public class SubClass extends SuperClass {
18     public void start() throws IOException {
19         throw new IOException("Unable to open file");
20     }
21
22     public void open(String fileName) throws FileNotFoundException {
23         FileInputStream fis = new FileInputStream(fileName);
24     }
25
26     public static void main(String[] args) {
27         System.out.println("Success.");
28     }
29 }
```

修改后可以运行的截图如下：

The screenshot shows an IDE window titled "Lab06 - SubClass.java". The code defines a class `SubClass` that extends `SuperClass`. It includes imports for `java.io.FileInputStream`, `java.io.FileNotFoundException`, and `java.io.IOException`. The `start()` method throws an `IOException` with the message "Unable to open file". The `open()` method creates a `FileInputStream` for a given file name. The `main()` method prints "Success.". The execution output at the bottom shows the command `java ...` and the output `Success.`, followed by the message "进程已结束,退出代码0".

```
//SubClass.java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class SubClass extends SuperClass {
    public void start() throws IOException {
        throw new IOException("Unable to open file");
    }

    public void open(String fileName) throws FileNotFoundException {
        FileInputStream fis = new FileInputStream(fileName);
    }

    public static void main(String[] args) {
        System.out.println("Success.");
    }
}
```

运行: SubClass
 /Users/huakaiyijiye/luoyide/Library/Java/JavaVirtualMachines/openjdk-18.0.1.1/Contents/Home/bin/java ...
 Success.
 进程已结束,退出代码0

Question6 写出以下程序的输出。 程序输出

The screenshot shows an IDE window titled "Lab06 - p06.java". The code defines a class `p06` with a `main()` method that calls `methodA()` and `methodB()` within a try-catch block. `methodA()` throws a `RuntimeException` and prints "methodA抛出一个异常!". `methodB()` prints "methodB执行!". The execution output shows the sequence of prints and the exception message, followed by "进程已结束,退出代码0".

```
public class p06 {
    public static void main(String[] args) {
        try {
            methodA();
        } catch (Exception e) {
            methodB();
        }
    }

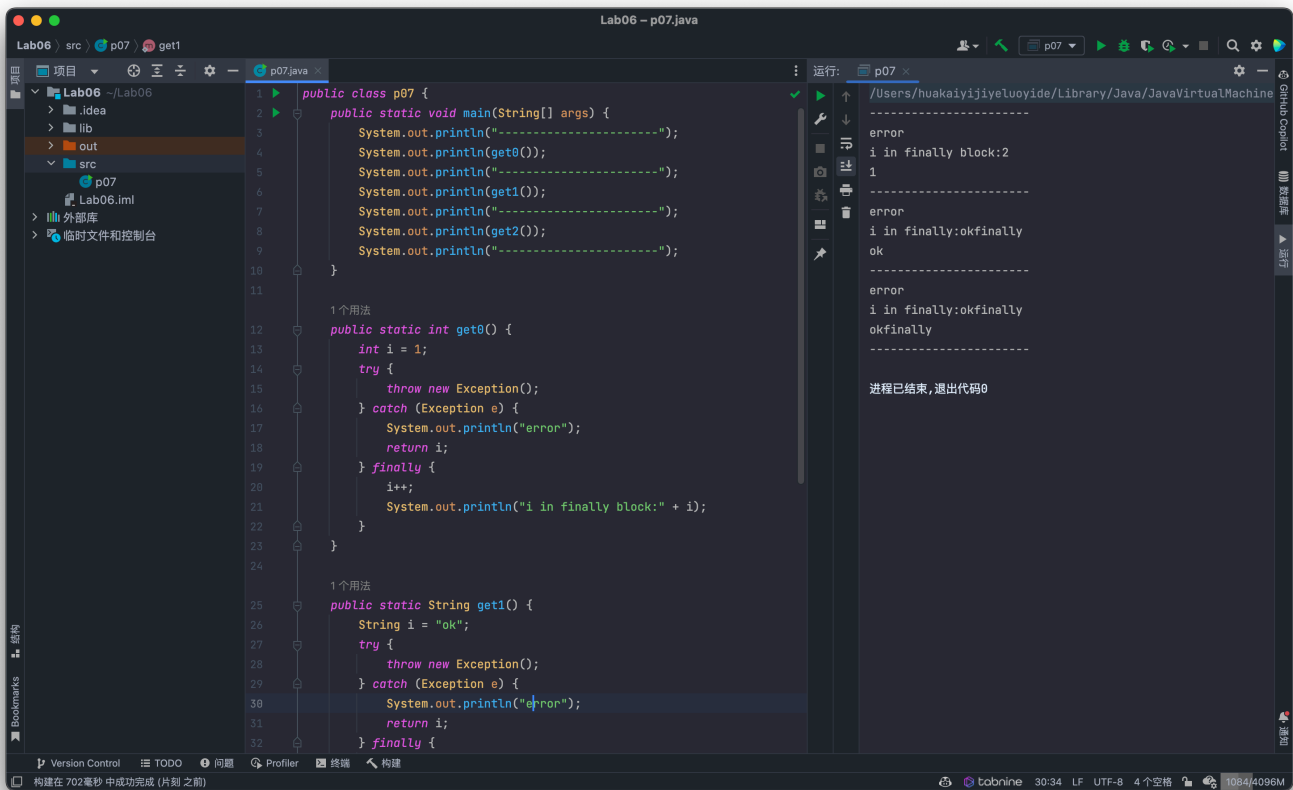
    1个用法
    private static void methodA() {
        try {
            System.out.println("methodA抛出一个异常! ");
            throw new RuntimeException();
        } finally {
            System.out.println("执行methodA的finally!");
        }
    }

    1个用法
    private static void methodB() {
        try {
            System.out.println("methodB执行! ");
        } finally {
            System.out.println("执行methodB的finally!");
        }
    }
}
```

运行: p06
 /Users/huakaiyijiye/luoyide/Library/Java/JavaVirtualMachines/openjdk-18.0.1.1/Contents/Home/bin/java ...
 methodA抛出一个异常!
 执行methodA的finally!
 methodB执行!
 执行methodB的finally!
 进程已结束,退出代码0

Question7 写出以下程序的输出，试着解释三个函数不同输出的原因。

程序输出



1. 执行顺序: try -> catch -> finally, finally 当中的语句必定被执行。
2. 一般 finally 不会修改返回值, 除非 finally 修改了该引用指向的实际内存内容。
 - o `get0()` 中程序返回的是 `int` 型数据, `int` 类型是基本数据类型, 返回时会在内存中生成一份与原来的值相同的 `int` 类型的拷贝, 故输出 2 和 1
 - o `get1()` 中, 程序返回的是 `String` 型数据, `String` 类型是引用数据类型, 返回时会在内存中生成一份与原来的引用相同的 `String` 类型的拷贝, 但由于 `String` 类型是不可变的数据类型, 故在执行 `i += "finally"` 后, `i` 的引用发生了变化, 故输出 `okfinally` 和 `ok`
 - o `get2()` 中, 程序返回的是 `StringBuilder` 型数据, `StringBuilder` 类型是引用数据类型, 返回时会在内存中生成一份与原来的引用相同的 `StringBuilder` 类型的拷贝, 在执行 `append` 后, 方法中的引用和返回值的引用都没有发生变化, 故输出 `okfinally` 和 `okfinally`

Question8 编写程序完成以下要求。

编程

代码如下:

```
1 class Triangle {
2     double x, y, z;
3
4     public Triangle(double x, double y, double z) {
5         this.x = x;
6         this.y = y;
```

```

7         this.z = z;
8     }
9
10    public void checkWhetherTriangle() throws NotTriangleException {
11        if (x + y <= z || x + z <= y || y + z <= x) {
12            throw new NotTriangleException(x, y, z);
13        }
14    }
15
16    public double getArea() throws NotTriangleException {
17        checkWhetherTriangle();
18        double p = (x + y + z) / 2;
19        return Math.sqrt(p * (p - x) * (p - y) * (p - z));
20    }
21
22    public void showInfo() throws NotTriangleException {
23        checkWhetherTriangle();
24        System.out.println("x = " + this.x);
25        System.out.println("y = " + this.y);
26        System.out.println("z = " + this.z);
27    }
28 }
29
30 public class Triangle_Test {
31     public static void main(String[] args) throws NotTriangleException {
32         Triangle t1 = new Triangle(3, 4, 5);
33         System.out.println(t1.getArea());
34         t1.showInfo();
35
36         Triangle t2 = new Triangle(2, 4, 6);
37         System.out.println(t2.getArea());
38         t2.showInfo();
39     }
40 }
41
42 class NotTriangleException extends Exception {
43     public NotTriangleException(double x, double y, double z) {
44         super(x + " " + y + " " + z + " 这似乎好像不能组成一个三角形捏~");
45     }
46 }

```

运行截图如下：

