Java 最常见的 208 道面试题：第七模块答案

Java团长 1周前

异常

74. throw 和 throws 的区别？

throws是用来声明一个方法可能抛出的所有异常信息，throws是将异常声明但是不处理，而是将异常往上传，谁调用我就交给谁处理。而throw则是指抛出的一个具体的异常类型。

75. final、finally、finalize 有什么区别？

- final可以修饰类、变量、方法，修饰类表示该类不能被继承、修饰方法表示该方法不能被重写、修饰变量表示该变量是一个常量不能被重新赋值。
- finally一般作用在try-catch代码块中，在处理异常的时候，通常我们将一定要执行的代码方法finally代码块中，表示不管是否出现异常，该代码块都会执行，一般用来存放一些关闭资源的代码。
- finalize是一个方法，属于Object类的一个方法，而Object类是所有类的父类，该方法一般由垃圾回收器来调用，当我们调用System的gc()方法的时候，由垃圾回收器调用finalize(),回收垃圾。

76. try-catch-finally 中哪个部分可以省略？

答：catch 可以省略

原因：

更为严格的说法其实是：try只适合处理运行时异常，try+catch适合处理运行时异常+普通异常。也就是说，如果你只用try去处理普通异常却不加以catch处理，编译是通不过的，因为编译器硬性规定，普通异常如果选择捕获，则必须用catch显示声明以便进一步处理。而运行时异常在编译时没有如此规定，所以catch可以省略，你加上catch编译器也觉得无可厚非。

理论上，编译器看任何代码都不顺眼，都觉得可能有潜在的问题，所以你即使对所有代码加上try，代码在运行期时也只不过是在正常运行的基础上加一层皮。但是你一旦对一段代码加上try，就等于显示地承诺编译器，对这段代码可能抛出的异常进行捕获而非向上抛出处理。如果是普通异常，编译器要求必须用catch捕获以便进一步处理；如果运行时异常，捕获然后丢弃并且+finally扫尾处理，或者加上catch捕获以便进一步处理。

至于加上finally，则是在不管有没捕获异常，都要进行的“扫尾”处理。

77. try-catch-finally 中，如果 catch 中 return 了，finally 还会执行吗？

答：会执行，在 return 前执行。

代码示例1：

```
/*
 * java面试题--如果catch里面有return语句，finally里面的代码还会执行吗？
 */
public class FinallyDemo2 {
    public static void main(String[] args) {
        System.out.println(getInt());
    }

    public static int getInt() {
        int a = 10;
        try {
            System.out.println(a / 0);
            a = 20;
        } catch (ArithmeticException e) {
            a = 30;
            return a;
            /*
             * return a 在程序执行到这一步的时候，这里不是return a 而是 retur
             * 但是呢，它发现后面还有finally，所以继续执行finally的内容，a=40
```

```
            * 再次回到以前的路径,继续走return 30，形成返回路径之后，这里的a就
            */
        } finally {
            a = 40;
        }

//      return a;
    }
}
```

执行结果：30

代码示例2：

```
package com.java_02;

/*
 * java面试题--如果catch里面有return语句，finally里面的代码还会执行吗？
 */
public class FinallyDemo2 {
    public static void main(String[] args) {
        System.out.println(getInt());
    }

    public static int getInt() {
        int a = 10;
        try {
            System.out.println(a / 0);
            a = 20;
        } catch (ArithmeticException e) {
            a = 30;
            return a;
            /*
             * return a 在程序执行到这一步的时候，这里不是return a 而是 retur
             * 但是呢，它发现后面还有finally，所以继续执行finally的内容，a=40
```

```
            * 再次回到以前的路径,继续走return 30，形成返回路径之后，这里的a就
            */
        } finally {
            a = 40;
            return a; //如果这样，就又重新形成了一条返回路径，由于只能通过1个r
        }

//      return a;
    }
}
```

执行结果：40

78. 常见的异常类有哪些？

- NullPointerException：当应用程序试图访问空对象时，则抛出该异常。
- SQLException：提供关于数据库访问错误或其他错误信息的异常。
- IndexOutOfBoundsException：指示某排序索引（例如对数组、字符串或向量的排序）超出范围时抛出。
- NumberFormatException：当应用程序试图将字符串转换成一种数值类型，但该字符串不能转换为适当格式时，抛出该异常。
- FileNotFoundException：当试图打开指定路径名表示的文件失败时，抛出此异常。
- IOException：当发生某种I/O异常时，抛出此异常。此类是失败或中断的I/O操作生成的异常的通用类。
- ClassCastException：当试图将对象强制转换为不是实例的子类时，抛出该异常。
- ArrayStoreException：试图将错误类型的对象存储到一个对象数组时抛出的异常。
- IllegalArgumentException：抛出的异常表明向方法传递了一个不合法或不正确的参数。
- ArithmeticException：当出现异常的运算条件时，抛出此异常。例如，一个整数“除以零”时，抛出此类的一个实例。
- NegativeArraySizeException：如果应用程序试图创建大小为负的数组，则

抛出该异常。

- NoSuchMethodException：无法找到某一特定方法时，抛出该异常。
- SecurityException：由安全管理器抛出的异常，指示存在安全侵犯。
- UnsupportedOperationException：当不支持请求的操作时，抛出该异常。
- RuntimeExceptionRuntimeException：是那些可能在Java虚拟机正常运行期间抛出的异常的超类。

（完）