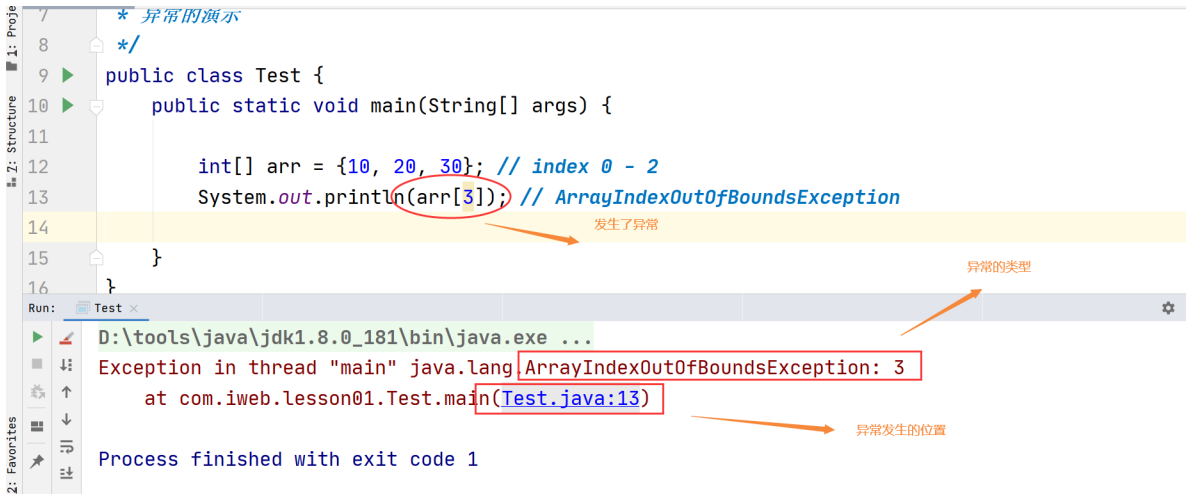
1.异常处理

1.1什么是异常

java中将可能发生的异常进行了分类，根据异常的类型程序员可以定位到异常发生的位置以及发生的原因。



1.2异常的分类

```java
package com.iweb.lesson02;

import java.io.FileInputStream;

/**
 * 作者: jack
 * 时间: 2021-04-26 0026 08:45
 * 描述: Test
 * 异常的分类
 * 运行时异常和检查异常:
 * 1. 运行时异常： 程序在运行的过程中产生异常, 分类为： RuntimeException 和 RuntimeException 的子类
 * 2. 检查异常： 程序本身没有问题，由于外部原因导致的异常。 RuntimeException 以外的异常。
 */
public class Test {

    public static void main(String[] args) {

        // 运行时异常
        // String str = null;
        // System.out.println(str.equals("abc")); // 比较2个字符串的值

        // 检查异常 ： 文件读取
        try {
            // 可能发生检查异常的代码 ： 发生异常的可能性就是 文件不存在. 文件是程序外部的内容,和程序本身无关
            // FileNotFoundException
```

```
            FileInputStream in = new
FileInputStream("C:\\Users\\jack\\Desktop\\java08\\笔记\\word.txt");
        } catch (Exception e) {
            // 异常发生的时候执行
            e.printStackTrace(); // 打印异常栈追踪的内容
        }

    }

}
```

1.3异常的处理

1.3.1 try{}catch{}： 捕获

```
try{

    // 可能发生检查异常的代码

}catch(Exception e){
    // 当异常发生的时候执行的代码
}
```

通过异常的父类来管理所有异常

```
try {
    System.out.println(1 / 1);
    int[] arr = {10, 20, 30};
    System.out.println(arr[3]);

    System.out.println("后面的代码");
} catch (Exception e) {
    // System.out.println(e.getMessage()); // 异常的信息
    System.out.println("算术");
    e.printStackTrace(); // 打印异常栈追踪的内容
}
```

通过多个catch块来对异常进行分类管理

```
try {
    System.out.println(1 / 1);
    int[] arr = {10, 20, 30};
    System.out.println(arr[3]);

    System.out.println("后面的代码");
} catch (ArithmeticException e) {
    // System.out.println(e.getMessage()); // 异常的信息
    System.out.println("算术");
    e.printStackTrace(); // 打印异常栈追踪的内容
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("数组越界");
    e.printStackTrace();
}
```

finally块： 无论是否发生异常, 都会执行

```
try {
    System.out.println(1 / 1);
    int[] arr = {10, 20, 30};
    System.out.println(arr[3]);

    System.out.println("后面的代码");
} catch (ArithmeticException e) {
    // System.out.println(e.getMessage()); // 异常的信息
    System.out.println("算术");
    e.printStackTrace(); // 打印异常栈追踪的内容
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("数组越界");
    e.printStackTrace();
} finally {
    System.out.println("finally"); // 无论是否发生异常都会执行
}
```

一个特殊的例子:

```
// 程序在执行的过程中只要遇到 return 语句方法就终止
public int fn(int num) {
    int i = 1;
    try {
        int res = 10 / num;
        return ++i; // ++i 会执行 , return不会执行
    } catch (Exception e) {
        e.printStackTrace();
        return ++i;
    } finally {
        return ++i; // 执行 return
    }
}
```

1.3.2 throws： 抛出异常类型

```
public class Test {

    // 将异常抛出, 由调用者来处理异常: 多种异常使用逗号隔开
    public void readFile() throws FileNotFoundException {
        FileInputStream file = new FileInputStream("E:\\1.txt");
    }

    public static void main(String[] args) {
        Test test = new Test();
        try {
            test.readFile();
        } catch (Exception e) {
```

```
            System.out.println(e.getMessage());
        }
    }

}
```

1.3.3 throw：抛出异常对象

对于有些异常的分类JAVA给我们提供了,有些特定场景的异常Java没有进行分类。我们需要在编写程序的时候自己进行异常的提示。

```
package com.iweb.lesson06;

/**
 * 作者: jack
 * 时间: 2021-04-26 0026 09:22
 * 描述: Test
 */
public class Test {

    public boolean login(String username, String password) {

        if (!username.equals("admin")) {
            throw new RuntimeException("用户不存在");
        }

        if (!password.equals("admin")) {
            throw new RuntimeException("密码错误");
        }

        return true;
    }

    public static void main(String[] args) {

        Test test = new Test();
        boolean res = test.login("admin", "user");
        System.out.println(res);

    }

}
```

1.3.4throw 和 throws 的区别 （面试）

1.4自定义异常

```java
public class UserNameException extends RuntimeException {

    public UserNameException(String message) {
        super(message);
    }

    public UserNameException() {
    }
}
```

```java
public class PasswordException extends RuntimeException {
    public PasswordException() {
    }

    public PasswordException(String message) {
        super(message);
    }
}
```

使用自定义异常

```java
public class Test {
    public boolean login(String username, String password) {

        if (!username.equals("admin")) {
            throw new UserNameException("用户不存在");
        }

        if (!password.equals("admin")) {
            throw new PasswordException("密码错误");
        }

        return true;
    }

    public static void main(String[] args) {

        Test test = new Test();
        boolean res = test.login("user", "user");
        System.out.println(res);

    }

}
```