# 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。

```
* 异邪的海不
  8
        */
  9
       public class Test {
Structure
          public static void main(String[] args) {
... 12
                int[] arr = {10, 20, 30}; // index 0 - 2
 13
                System.out.println(arr[3]); // ArrayIndexOutOfBoundsException
 14
 15
            }
                                                                                 异常的类型
 Run: __ Test ×
  D:\tools\java\jdk1.8.0_181\bin\java.exe ...
  Exception in thread "main" java.lang ArrayIndexOutOfBoundsException: 3
 药 1
           at com.iweb.lesson01.Test.main(<u>Test.java:13</u>)
 = ↓
                                                                       异党发生的位置
       Process finished with exit code 1
```

## 1.2异常的分类

```
1
   package com.iweb.lesson02;
2
3
   import java.io.FileInputStream;
4
   /**
5
6
    * 作者: jack
    * 时间: 2021-04-26 0026 08:45
7
    * 描述: Test
9
    * 异常的分类
   * 运行时异常和检查异常:
10
    * 1. 运行时异常: 程序在运行的过程中产生异常,分类为: RuntimeException 和
11
   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
12
   常。
    */
13
14
   public class Test {
15
       public static void main(String[] args) {
16
17
          // 运行时异常
18
19
          // String str = null;
20
          // System.out.println(str.equals("abc")); // 比较2个字符串的值
21
          // 检查异常: 文件读取
22
23
          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
24
   部的内容,和程序本身无关
              // FileNotFoundException
25
```

```
26
               FileInputStream in = new
    FileInputStream("C:\\Users\\jack\\Desktop\\java08\\笔记\\word.txt");
27
            } catch (Exception e) {
28
               // 异常发生的时候执行
29
               e.printStackTrace(); // 打印异常栈追踪的内容
30
            }
31
        }
32
33
34
   }
```

### 1.3异常的处理

### 1.3.1 try{}catch{}: 捕获

#### 通过异常的父类来管理所有异常

```
1
    try {
2
        System.out.println(1 / 1);
 3
        int[] arr = {10, 20, 30};
4
        System.out.println(arr[3]);
 5
6
        System.out.println("后面的代码");
7
    } catch (Exception e) {
        // System.out.println(e.getMessage()); // 异常的信息
9
        System.out.println("算术");
        e.printStackTrace(); // 打印异常栈追踪的内容
10
11
   }
```

#### 通过多个catch块来对异常进行分类管理

```
1
    try {
2
        System.out.println(1 / 1);
 3
        int[] arr = {10, 20, 30};
4
        System.out.println(arr[3]);
 5
6
        System.out.println("后面的代码");
 7
    } catch (ArithmeticException e) {
8
        // System.out.println(e.getMessage()); // 异常的信息
9
        System.out.println("算术");
10
        e.printStackTrace(); // 打印异常栈追踪的内容
11
    } catch (ArrayIndexOutOfBoundsException e) {
12
        System.out.println("数组越界");
13
        e.printStackTrace();
14
    }
```

#### finally块: 无论是否发生异常,都会执行

```
1
    try {
2
        System.out.println(1 / 1);
 3
        int[] arr = {10, 20, 30};
 4
        System.out.println(arr[3]);
 5
 6
        System.out.println("后面的代码");
 7
    } catch (ArithmeticException e) {
8
        // System.out.println(e.getMessage()); // 异常的信息
9
        System.out.println("算术");
10
        e.printStackTrace(); // 打印异常栈追踪的内容
11
    } catch (ArrayIndexOutOfBoundsException e) {
12
        System.out.println("数组越界");
13
        e.printStackTrace();
14
   } finally {
        System.out.println("finally"); // 无论是否发生异常都会执行
15
16
    }
```

#### 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
 2
    public int fn(int num) {
 3
       int i = 1;
 4
        try {
 5
            int res = 10 / num;
            return ++i; // ++i 会执行 , return不会执行
6
        } catch (Exception e) {
8
            e.printStackTrace();
9
            return ++i;
10
        } finally {
11
            return ++i; // 执行 return
12
        }
13
    }
```

### 1.3.2 throws: 抛出异常类型

```
public class Test {
 2
 3
        // 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 {
4
 5
            FileInputStream file = new FileInputStream("E:\\1.txt");
 6
        }
 7
        public static void main(String[] args) {
8
9
            Test test = new Test();
10
            try {
11
                test.readFile();
12
            } catch (Exception e) {
```

### 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;
 1
 2
 3
    /**
 4
    * 作者: jack
 5
    * 时间: 2021-04-26 0026 09:22
 6
     * 描述: Test
 7
     */
 8
    public class Test {
 9
10
        public boolean login(String username, String password) {
11
12
            if (!username.equals("admin")) {
13
                throw new RuntimeException("用户不存在");
            }
14
15
16
            if (!password.equals("admin")) {
17
                throw new RuntimeException("密码错误");
18
19
20
            return true;
21
        }
22
23
        public static void main(String[] args) {
24
25
            Test test = new Test();
            boolean res = test.login("admin", "user");
26
27
            System.out.println(res);
28
29
        }
30
31 }
```

# 1.3.4throw 和 throws 的区别 (面试)

# 1.4自定义异常

```
1
   public class UserNameException extends RuntimeException {
2
3
       public UserNameException(String message) {
4
           super(message);
5
       }
6
7
       public UserNameException() {
8
       }
9
   }
```

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

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

#### 使用自定义异常

```
public class Test {
 1
 2
        public boolean login(String username, String password) {
 3
 4
            if (!username.equals("admin")) {
 5
                throw new UserNameException("用户不存在");
 6
            }
 7
 8
            if (!password.equals("admin")) {
 9
                throw new PasswordException("密码错误");
10
11
12
            return true;
13
        }
14
15
        public static void main(String[] args) {
16
17
            Test test = new Test();
18
            boolean res = test.login("user", "user");
19
            System.out.println(res);
20
21
        }
22
23
   }
```