异常

1.概述

什么是异常

> 异常是指程序运行过程中出现的非正常情况（比如：除以0...）

异常会导致什么情况

> 异常会中断程序的运行，从出现异常的那一行不再继续往下执行

异常的分类

1. 受查异常：必须要处理，否则编译不通过
2. 非受查异常：不是必须处理（最好处理）

2.Java中的异常处理机制

捕获异常

语法

> `try-catch-finally` 用来捕获异常

```
try{
    //可能出现异常的代码
}catch(异常类型 异常变量){
    //处理异常
}catch(异常类型 异常变量){
    //处理异常
}finally{
    //释放资源
}
```

1. `try{}`语句块中放的是要检测的java代码，可能有会抛出异常，也可能会正常执行
2. `catch(异常类型){}`块是当Java运行时系统接收到try块中所抛出异常对象时，会寻找能处理这一异常catch块来进行处理（可以有多个catch块）。
3. `finally{}`不管系统有没有抛出异常都会去执行，一般用来释放资源。除了在之前执行了`System.exit(0)`

上面代码可以有一下几种组合：

1. try-catch
2. try-catch-finally
3. try-finally：可以组合，但是一般不会使用

`printStackTrace()`输出异常信息，包括异常类型和第几行出现的异常，一般在catch块处理异常，会调用异常对象的此方法，来输出详细信息

捕获顺序

> `Exception`类是所有异常的父类，捕获时可以直接捕获Exception异常，如果捕获多个异常，则**Exception**应该最后捕获

例子

示例代码

```
Scanner scanner = new Scanner(System.in);
try {
    System.out.print("输入第一个数：");
    int x = scanner.nextInt();// 可能出现异常的代码
    System.out.print("输入第二个数：");
    int y = scanner.nextInt();
    System.out.println("x/y=" + x / y);// 可能出现异常的代码
    System.exit(0);// 如果执行了这行代码，finally语句块不会执行
} catch (ArithmeticException e) {
    e.printStackTrace();
} catch (InputMismatchException e) {
    e.printStackTrace();
} catch (Exception e) {// 应该最后捕获此异常
    e.printStackTrace();
} finally {
    // 善后工作，不管有没有出现异常，都会执行
    System.out.println("总会执行的finally");
    scanner.close();
}

System.out.println("运算结束");
```

执行结果

```
输入第一个数：4
输入第二个数：0
java.lang.ArithmeticException: / by zero
        at com.oaec.exceptiondemo.ExceptionDemo1.main(ExceptionDemo1.java:14)
总会执行的finally
运算结束
```

抛出异常

> throw/throws

1. `throw`用于手动抛出异常。作为程序员可以在任意位置手动抛出异常。
2. `throws`用于在方法上标识要暴露的异常。抛出的异常交由调用者处理

示例代码

```
class NetBar{
    public void enter(int age) throws IllegalArgumentException{
        if(age < 18){
            //手动抛出异常（参数类型不匹配）
            throw new IllegalArgumentException("未成年人不准进入");
        }else{
            System.out.println("欢迎光临");
        }
    }
}
```

其中，`throw new IllegalArgumentException("未成年人不准进入");`是手动抛出异常，`throws IllegalArgumentException`告诉调用者，此方法会抛出此异常

3.自定义异常

常见的异常

1. `NullPointerException`空指针异常。
2. `ArrayIndexOutOfBoundsException`数组下标越界异常
3. `ClassCastException`类型转换异常
4. `ClassNotFoundException`类找不到异常
5. ……

为什么要自定义异常

> Java提供的异常体系不可能预见所有希望加以报告的错误，所有有时候需要程序员自己定义一下异常类以满足特定的需求

自定义异常的方式

- 自己写一个异常类继承已有的异常类

示例代码：

自定义异常类

```
/**
 * 年龄不满18岁异常
 * @author Kevin
 *
 */
class AgeLessThanEighteenException extends Exception{

    private static final long serialVersionUID = 7505653748337799068L;

    private String message;

    public AgeLessThanEighteenException(String message) {
        super(message);
        this.message = message;
    }

    @Override
    public String getMessage() {
        return message;
    }

}
```

抛出自定义异常的方法

```
/**
 * 网吧
 * @author Kevin
 */
class NetBar {
    public void enter(int age) throws AgeLessThanEighteenException {
        if (age < 18) {
            throw new AgeLessThanEighteenException("未成年人不准进入");
        } else {
            System.out.println("欢迎光临");
        }
    }
}
```

main方法调用

```
public static void main(String[] args) {
    NetBar netBar = new NetBar();
    try {
        netBar.enter(17);
    } catch (AgeLessThanEighteenException e) {
        e.printStackTrace();
    }
    System.out.println("*********");
}
```

执行结果

```
com.oaec.exceptiondemo.AgeLessThanEighteenException: 未成年人不准进入
        at com.oaec.exceptiondemo.NetBar.enter(ExceptionDemo2.java:23)
        at com.oaec.exceptiondemo.ExceptionDemo2.main(ExceptionDemo2.java:7)
```