异常

- What?Why?How?

本章概述

- 异常的概念
- 异常的分类
- 异常处理（try, catch, finally，throws，　throw）
- 异常和重写的关系
- 自定义异常
- Jdk7-12异常处理

生活中的异常

- 正常情况下，小王每日开车去上班，耗时大约30分钟

一路畅通

但是，异常情况迟早要发生！

堵车！

撞车！

程序中的异常2-1

- 以下程序运行时会出现错误吗?

```
public class Test1 {
	public static void main(String[] args) {
		Scanner in = new Scanner(System.in);
		System.out.print("请输入被除数:");
		int num1 = in.nextInt();
		System.out.print("请输入除数:");
		int num2 = in.nextInt();
		System.out.println(String.format("%d / %d = %d",
					num1, num2, num1/ num2));
		System.out.println("感谢使用本程序！ ");
	}
}
```

程序中的异常2-2

- 如何解决该问题呢?

尝试通过if-else来解决异常问题

```
public class Test2 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        …
        System.out.print("请输入除数:");
        int num2 = 0;
        if (in.hasNextInt()) { // 如果输入的除数是整数
            num2 = in.nextInt();
            if (0 == num2) { // 如果输入的除数是0
                System.err.println("输入的除数是0，程序退出。");
                System.exit(1);
            }
        } else { // 如果输入的除数不是整数
            System.err.println("输入的除数不是整数，程序退出。");
            System.exit(1);
        }
        …
    }
}
```

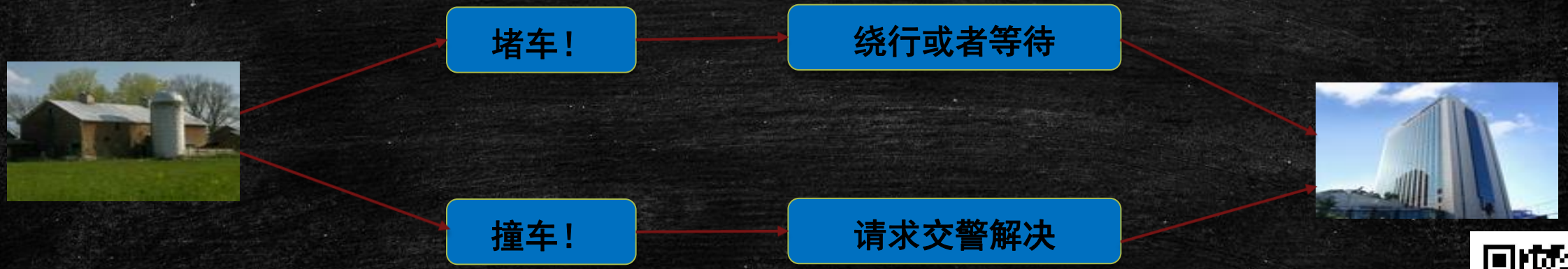
使用异常机制

弊端：
1、代码臃肿
2、程序员要花很大精力“堵漏洞”
3、程序员很难堵住所有“漏洞”

什么是异常

- 异常是指在程序的运行过程中所发生的不正常的事件，它会中断正在运行的程序

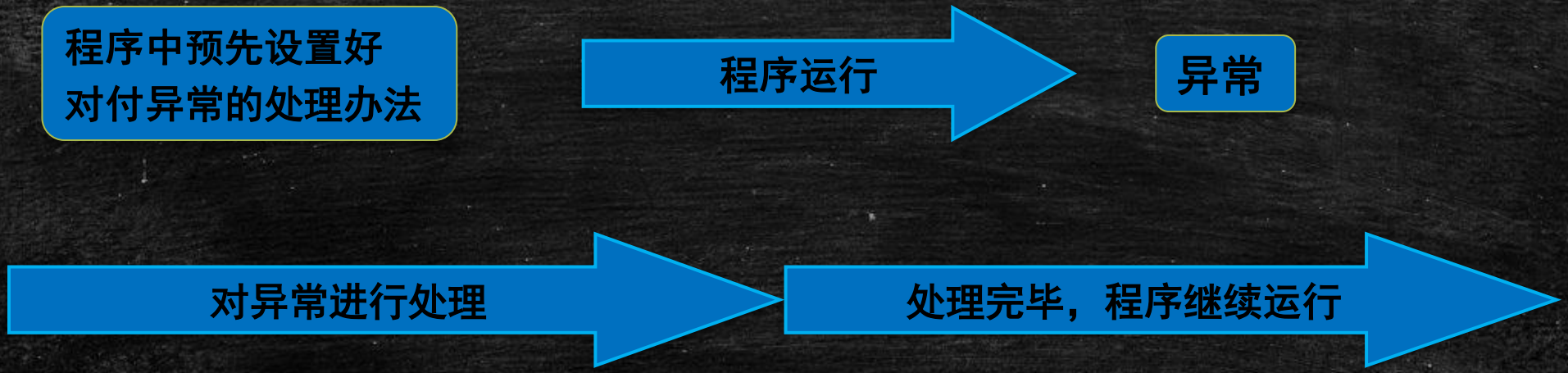
生活中面对异常通常会这样处理



生活中，根据不同的异常进行相应的处理，而不会就此中断我们的生活

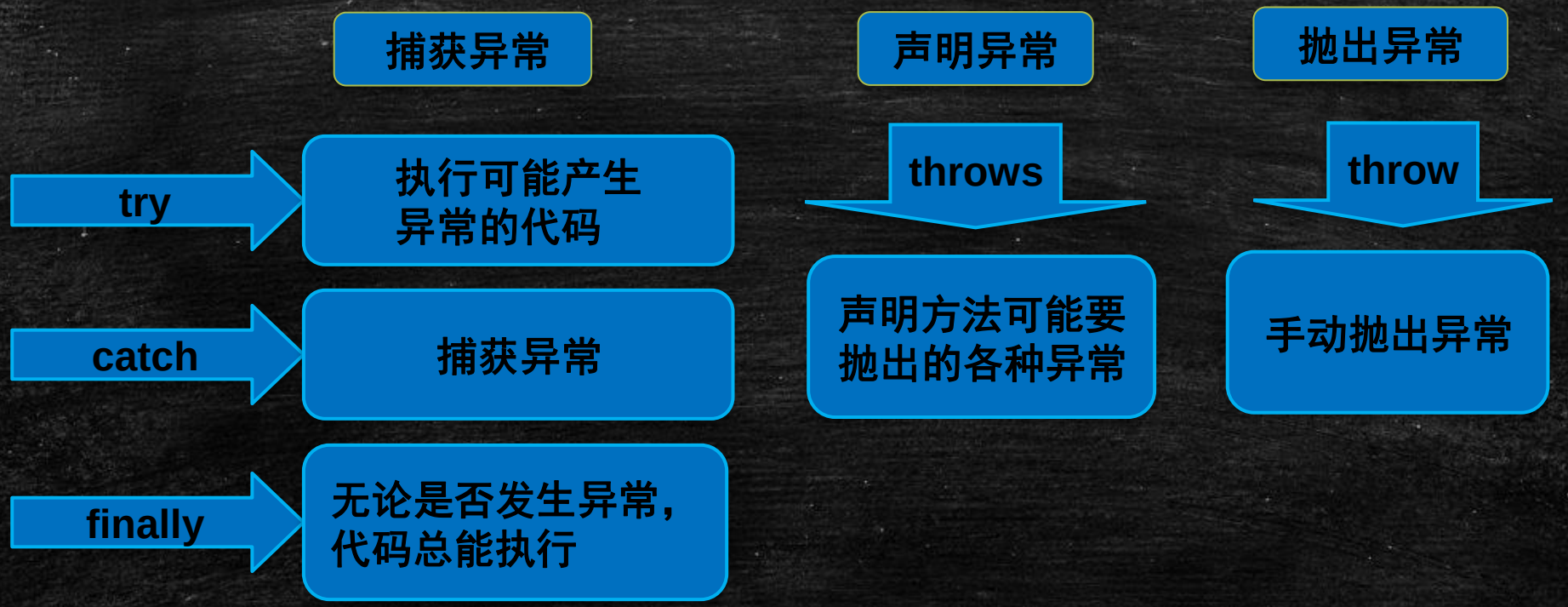
什么是异常处理

- Java编程语言使用异常处理机制为程序提供了错误处理的能力



Java中如何进行异常处理

- Java的异常处理是通过5个关键字来实现的：try、catch、finally、throw、throws

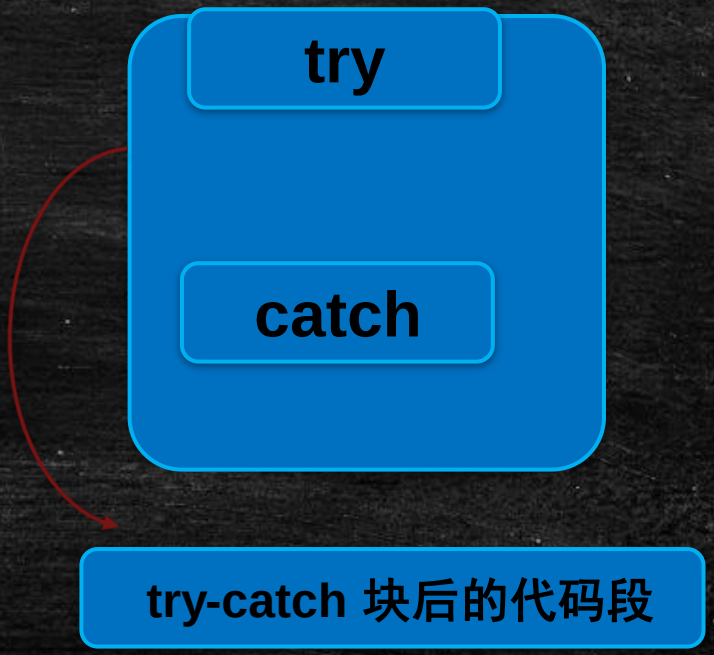


try-catch块5-1

- 使用try-catch块捕获异常，分为三种情况：

第一种情况： 正常

```
public void method(){
    try {
        // 代码段(此处不会产生异常)
    } catch (异常类型 ex) {
        // 对异常进行处理的代码段
    }
    // 代码段
}
```



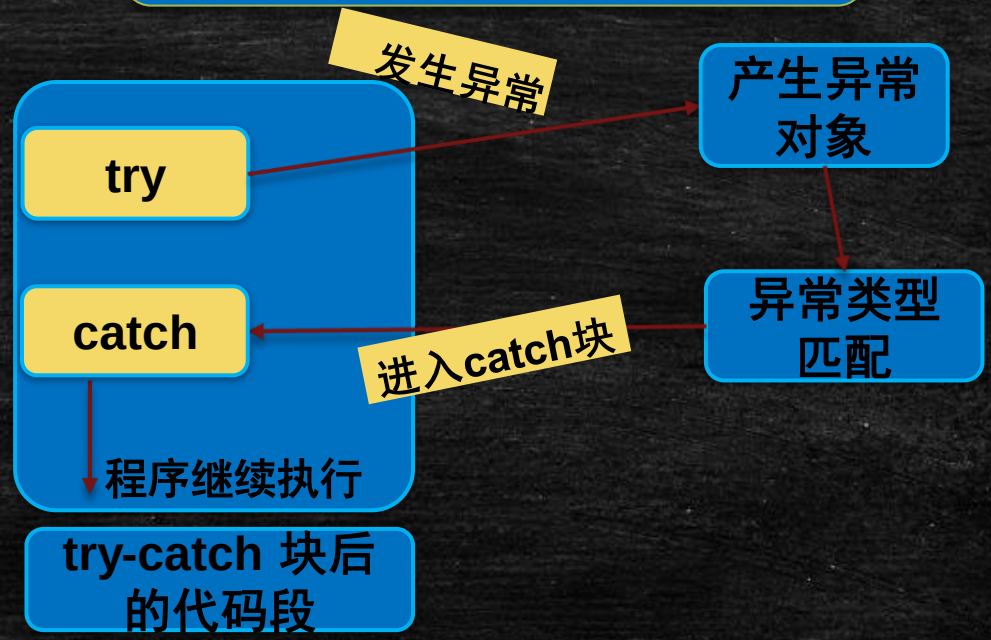
try-catch块5-2

- 使用try-catch块捕获异常，分为三种情况：

第二种情况：出现异常

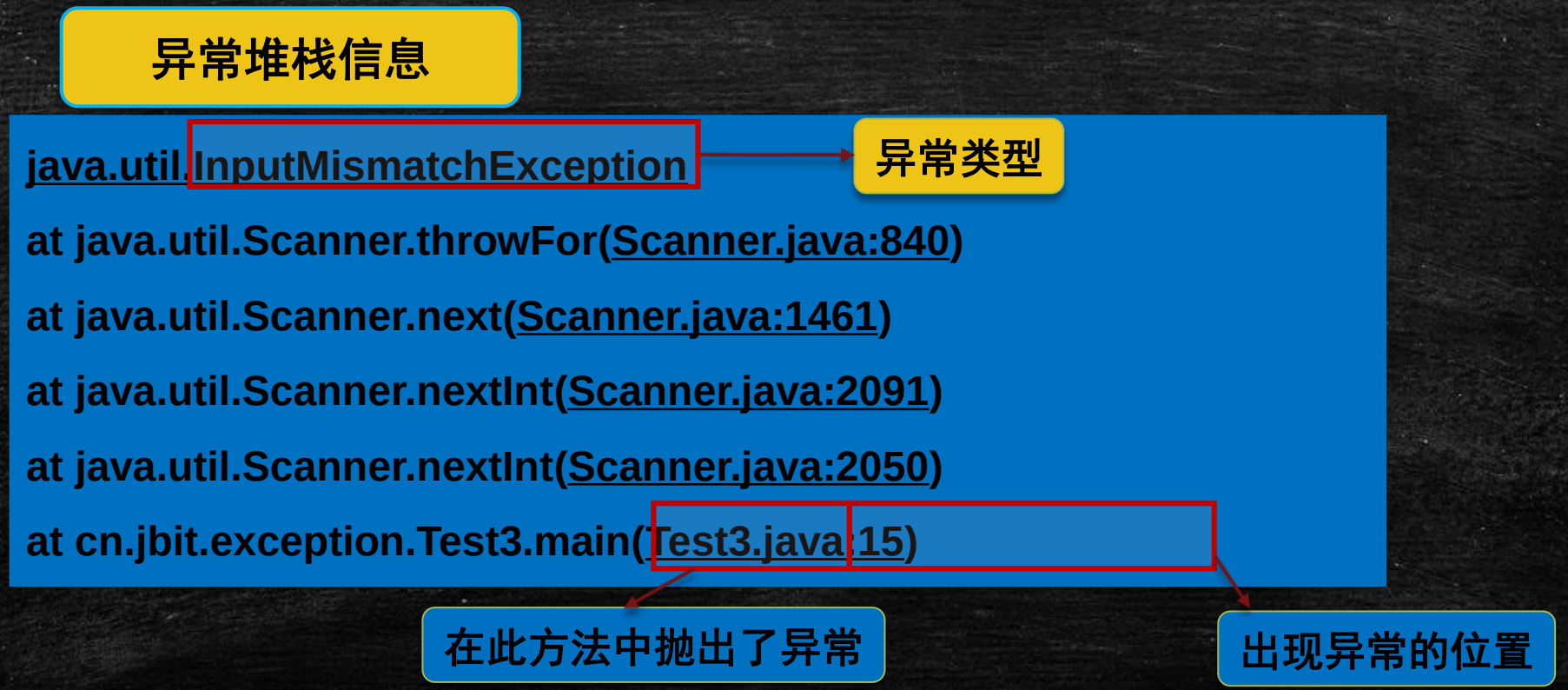
```
public void method(){
        try {
            // 代码段 1
            // 产生异常的代码段 2
            // 代码段 3
        } catch (异常类型 ex) {
            // 对异常进行处理的代码段4
        }
        //代码段5
}
```

异常是一种特殊的对象，类型为java.lang.Exception或其子类



try-catch块5-3

- printStackTrace的堆栈跟踪功能显示出程序运行到当前类的执行流程



try-catch块5-4

- 使用try-catch块捕获异常，分为三种情况：

第三种情况：异常类型不匹配

```
public void method(){
        {
    // 代码段 1
        // 产生异常的代码段 2
    // 代码段 3
    }       (异常类型 ex) {
    // 对异常进行处理的代码段4
    }
    // 代码段5
}
```

try-catch块5-5

```
System.err.println("出现错误：被除数和除数必须是整数， ”
+"除数不能为零。");
```

```
e.printStackTrace();
```

- 调用方法输出异常信息

方法名	说　明
void printStackTrace()	输出异常的堆栈信息
String getMessage()	返回异常信息描述字符串， 是printStackTrace()输出信息的一部分

常见的异常类型

异 常 类 型	说　明
Exception	异常层次结构的父类
ArithmeticException	算术错误情形，如以零作除数
ArrayIndexOutOfBoundsException	数组下标越界
NullPointerException	尝试访问 null 对象成员
ClassNotFoundException	不能加载所需的类
IllegalArgumentException	方法接收到非法参数
ClassCastException	对象强制类型转换出错
NumberFormatException	数字格式转换异常，如把"abc"转换成数字

上机练习1—多重catch的使用

需求说明

编写数据转换类，定义数据转换方法，具有String类型的参数，实现将参数转换为整型数据后输出，要求使用多重catch语句处理异常

编写测试类，调用数据类型转换方法，分别传递参数“a”、20

分析

使用NumberFormatException、Exception异常类型

try-catch-finally 2-1

- 在try-catch块后加入finally块
  - 是否发生异常都执行
  - 不执行的唯一情况

try-catch-finally 2-2

- 存在return的try-catch-finally块

```
public void method(){
        {
        // 代码段 1
        // 产生异常的代码段 2
    }         (异常类型 ex) {
        // 对异常进行处理的代码段3
                              执行return退出方法
    }       {
         // 代码段 4
    }
}
```

try块中有return语句执行过程与此类似

多重catch块

- 引发多种类型的异常
  - 排列catch 语句的顺序：先子类后父类
  - 发生异常时按顺序逐个匹配
  - 只执行第一个与异常类型匹配的catch语句

```
public void method(){
        try {
            // 代码段
            // 产生异常(异常类型2)
        } catch (异常类型1 ex) {
            // 对异常进行处理的代码段
        } catch (异常类型2 ex) {
            // 对异常进行处理的代码段
        } catch (异常类型3 ex) {
            // 对异常进行处理的代码段
        }
        // 代码段
}
```

小结

- **面试题：try-catch块中存在return语句，是否还执行finally块，如果执行，说出执行顺序**
- try-catch- finally块中， finally块唯一不执行的情况是什么？

上机练习2——根据编号输出课程名称

- 需求说明：
  - 按照控制台提示输入1～3之间任一个数字，程序将输出相应的课程名称
  - 根据键盘输入进行判断。如果输入正确，输出对应课程名称。如果输入错误，给出错误提示
  - 不管输入是否正确，均输出“欢迎提出建议”语句

```
请输入课程代号(1~3之间的数字):1
C#编程
欢迎提出建议!
```

声明异常

```
public class Test7 {
    public static void divide() throws Exception {
        //可能出现异常的代码
    }
    public static void main(String[] args) {
        try {
            divide();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
//  public static void main(String[] args) throws Exception {
//      divide();
//  }
}
```

声明异常，多个异常用逗号隔开

方式1：调用者处理异常

main方法声明的异常由Java虚拟机处理

方式2：调用者继续声明异常

抛出异常

```
public class Person {
    private String name = "";   // 姓名
    private int age = 0;   // 年龄
    private String sex = "男";  // 性别
    public void setSex(String sex) throws Exception {
        if ("男".equals(sex) || "女".equals(sex))
            this.sex = sex;
        else {
            throw new Exception("性别必须是\"男\"或者\"女\"！ ");
        }
    }
}
```

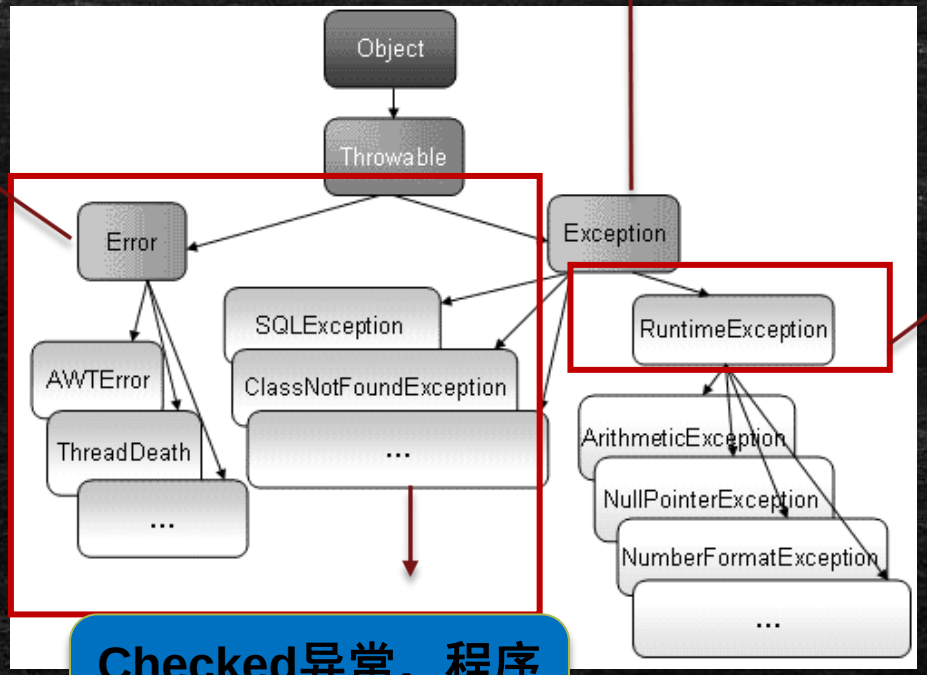
抛出异常

异常的分类2-1

Exception和Error类的父类

由Java应用程序抛出和处理的非严重错误

仅靠程序本身无法恢复的严重错误



运行时异常，不要求程序必须做出处理

Checked异常，程序必须处理该类异常

小结

- **面试题：** 说出5个常见的运行时异常
- throw与throws的区别是什么？

自定义异常

- **何时需要自定义异常？**

当JDK 中的异常类型不能满足程序的需要时，可以自定义异常类

使用自定义异常的步骤

1. 定义异常类
2. 编写构造方法，继承父类的实现
3. 实例化自定义异常对象
4. 使用throw抛出

继承Excepion 或者RuntimeException

上机练习4—自定义异常

需求说明

1、自定义异常

2、在setAge(int age) 中对年龄进行判断，如果年龄介于1到100直接赋值，否则抛出自定义异常

3、在测试类中创建对象并调用setAge(int age)方法，使用try-catch捕获并处理异常

小结

- 异常分为Checked异常和运行时异常
  - Checked异常必须捕获或者声明抛出
  - 运行时异常不要求必须捕获或者声明抛出
- try-catch-finally中存在return语句的执行顺序
- finally块中语句不执行的情况
- throw和throws关键字的区别