编程入门基础

——异常和保护

主讲教师：耿宇航

如何处理异样的情况？

- 编程的关键在于“防患于未然”
- 我们必须在代码中，事先预料出所有可能的情况，机器在执行中才能“稳健”
- 机器本身并没有任何智能，所有的看起来的“智能”现象，都是程序员事先就精心编写好的。

意外情况的处理思路

- 传统方式
  - 用if语句进行判断，并作相应的个理
- 现代的方式
  - 抛现异常，捕获异常的机制

if 判断法

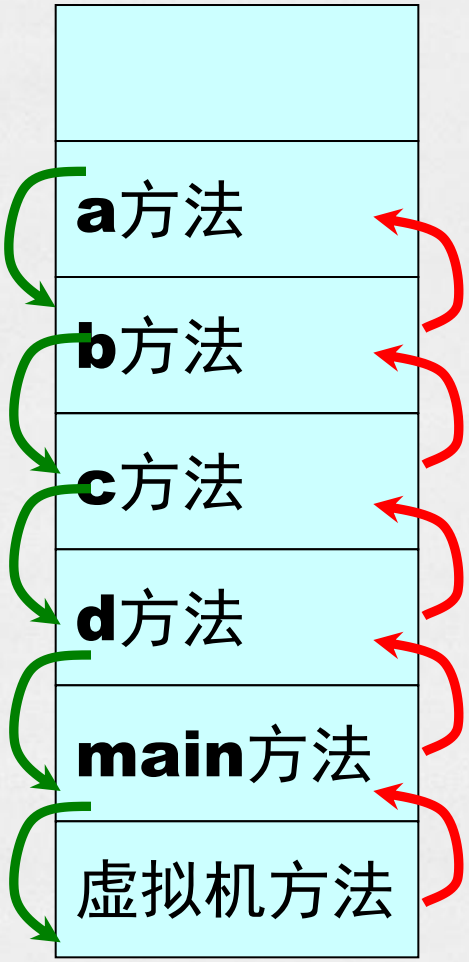
- 一直沿用至今，十分普遍，行之有效

```
String s = “abdcd”;
int n = s.indexOf(“+”);
if(n<0){
  // 处理意外情况...
}
String s1 = s.substring(0,n);
```

被谁调用?

- a方法为什么会执行?
  - 因为b方法调用了它
- b方法为什么会执行?
  - 因为c方法调用了它
- c方法为什么会执行?
  - 因为d方法调用了它
- 到底谁是第一个执行的方法?
  - main() ??
  - 虚拟机方法

调用栈



表示调用

表示返回

控制异常的新方法

- 分析
  - 发现异常的人可能不知道该如何处置异常
  - 有能力处理异常的人不想陷入异常产生的细节代码
- 目标
  - 把异常的发现和处理相分离

分工

- 人类社会发展的过程，就是一个生产不断社会化的过程。
  - 分工越来越细，专门的人做专门的事
- 程序语言的设计也模仿了人类社会的成功经验。
  - 设计与编码的分离
  - 功能和实现的分离
  - 错误的发现与控制的分离

异常语法

```
try{
  语句;
  调用方法();
  ...
}
catch(异常类型 e){
  处理异常
}
catch(异常类型 e){
  处理异常
}
```

try{ } 块表示受到监控的代码。其中一但发生情况，就会按照是哪种情况，跳到**catch**块中执行

e 是产生的异常对象的名字。每个异常发生的时候，都会有对应的异常对象。它记录了异常的详细信息

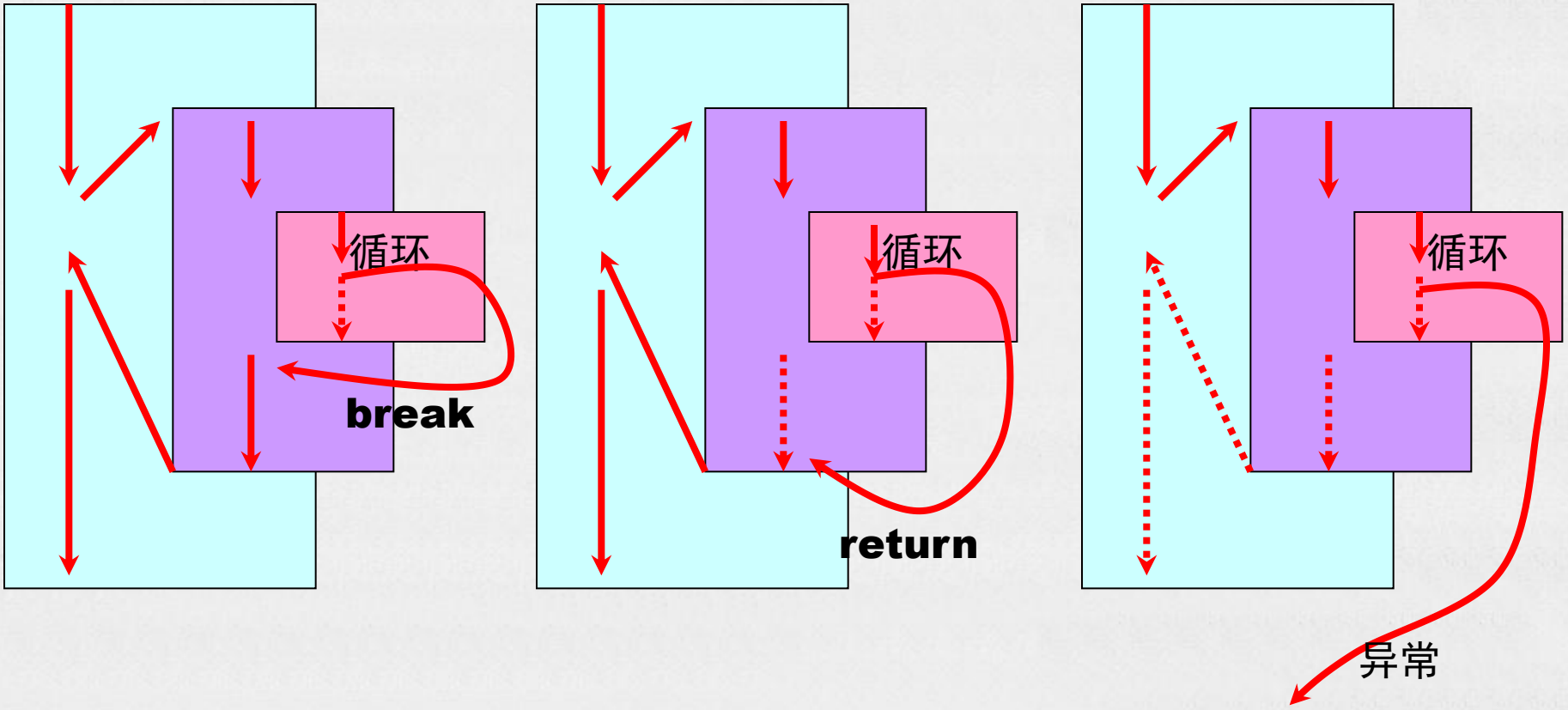
抛出异常

- 方法在进行工作的时候，可以抛出异常，好比“报警”
- 报警后，手头的工作立即停止
- 如果没人理会这个异常，调用它的所有方法的剩余工作，也会立即停止
- 直到撞上了catch，此次事件才平息

异常机制的作用

- 异常是流程转移的有效手段
- 异常使发现错误的人不再尴尬
  - 他不需要处理这个事件
- 异常的代码从在量的if中解救出来

break, return, 异常的比较



异常的执行流程

```
try
{
   1.
   2.
   调用
   3.
}
catch(Exception e)
{
  4
}
  5
```

```
{
  11
  12
  13
}
```

正常顺序：

1,2,11,12,13, 3,5

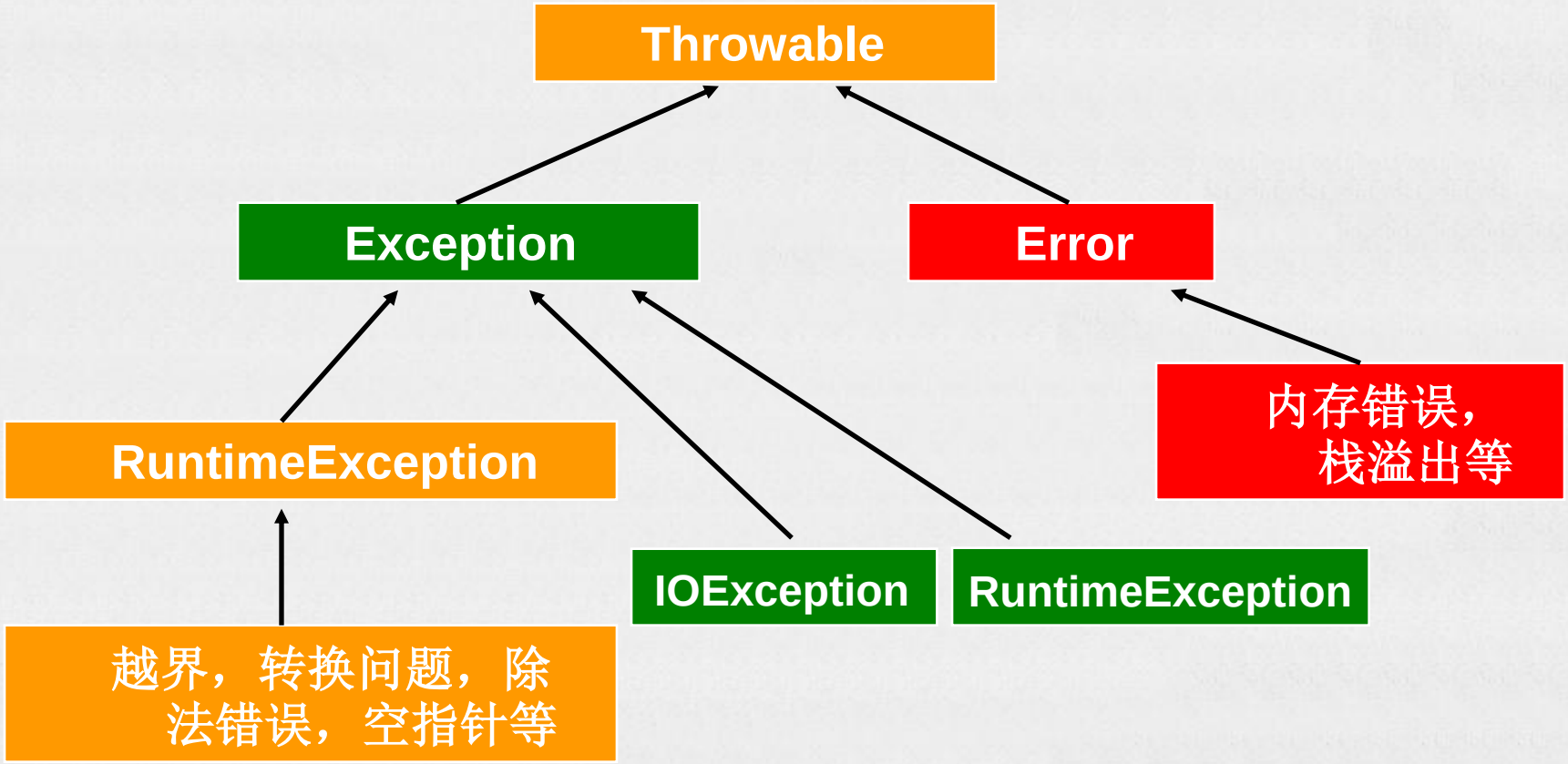
异常顺序：

1,2,11, 4, 5

注意

- catch中只抓住它所对应的try块中产生的异常
- 异常一经捕获，就消失了，后续的代码，正常执行

异常的分类



异常类型的差别

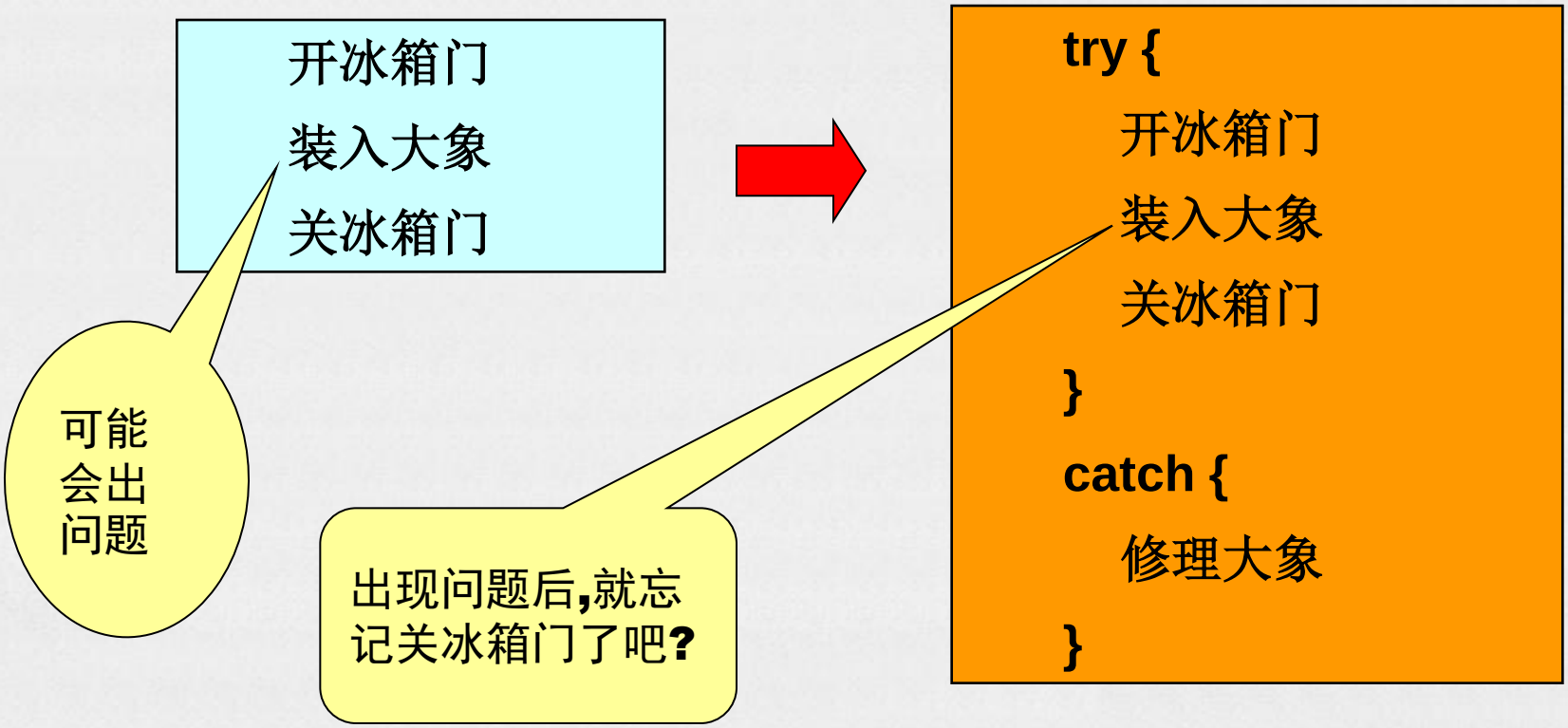
- Exception类型
  - 可以被catch住
- Error
  - 不可恢复的“灾难”，不必料理后事
- RuntimeException
  - 可以被cath，但不强迫处理

try..finally

- catch的目的
  - 抓住出现的异常情况，使工作维持正轨
- finally的目的
  - 有某些事情，相当重要，即便是发生了异常，也不应该被跳过
  - 比如：还债

finally到底用在哪？

- 宋丹丹的失误



完美的方案

```
打开冰箱门
try
{
    装入大象
}
finally
{
    关闭冰箱门
}
```

```
打开冰箱门
try
{
    装入大象
}
catch(...)
{
    修理大象
}
finally
{
    关闭冰箱门
}
```

注意

- try..catch，try..finally是两套互无关系的机构，可以相互嵌套
- try..finally..catch只是一种省略的写法。

注意

- 经典的错误

```
try
{
    打开冰箱门
    装入大象
}
finally
{
    关闭冰箱门
}
```

如果,打开冰箱门时,出现异外情况,会错误地执行”关闭冰箱门”的动作

考考你

请分别说出正常与异常执行的流程

```
1...

try{
   2...
   3...
   4...
}
finally{
   5...
}

   6...
```

正常时: 1,2,3,4,5,6

1..发生异常时: 1

3..发生异常时: 1,2,3,5

韩信点兵问题

- 韩信知道部队人数大约1000多，具体数字不详。
- 使用如下办法
  - 5人一组 余 1
  - 7人一组 余 2
  - 8人一组 余 3
- 考虑所有可能的人数 1000 - 2000

谢　谢！