异常引入

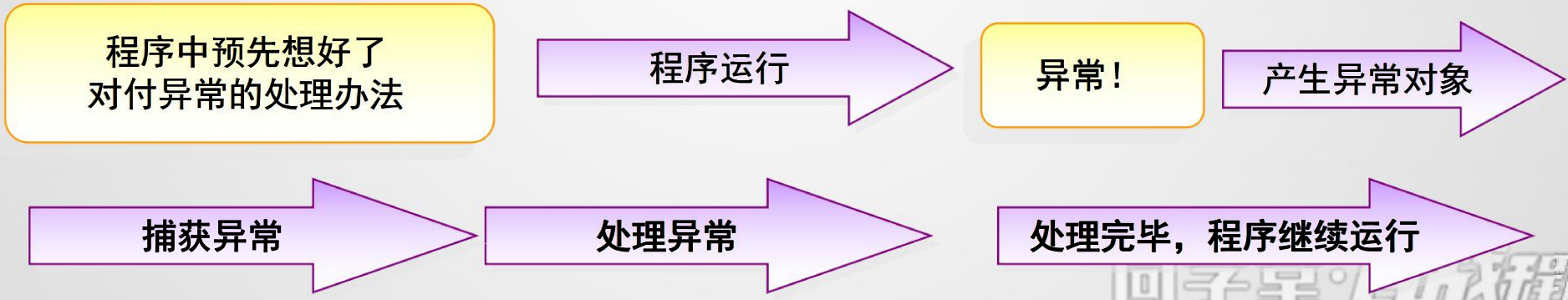
- 程序中的异常
  - 示例1：给出除数和被除数，求商
    - 如果除数为0，出异常
    - 如果除数或者被除数不是数字，出异常
  - 示例2：将d:/a.txt复制到e:/a.txt
    - 如果d:/a.txt不存在
    - 如果e:/存在a.txt
    - 如果e盘空间不足
    - 如果复制过程中出错

```
if(“d:/a.txt”这个文件存在){
    if(e盘的空间大于a.txt文件长度){
        if(文件复制一半IO流断掉){
            停止copy，输出：IO流出问题！
        }else{
            copyFile("d:/a.txt","e:/a.txt");
        }
    }else{
        输出：e盘空间不够存放a.txt！
    }
}else{
    输出：a.txt不存在！
}
```

真正的代码，只有一行！其余都是用于处理例外情况的代码！

异常引入

- 程序中的异常
  - 面对异常该怎么办呢？
  - 方式1：由开发者通过if-else来解决异常问题
    - 代码臃肿：业务代码和异常处理代码放一起
    - 程序员要花很大精力"堵漏洞“
    - 程序员很难堵住所有“漏洞”，对程序员本身要求较高
  - 方式2：开发者不需要通过if-else来解决异常问题，而是Java提供异常处理机制。它将异常处理代码和和业务代码分离，使程序更优雅，更好的容错性，高键壮性

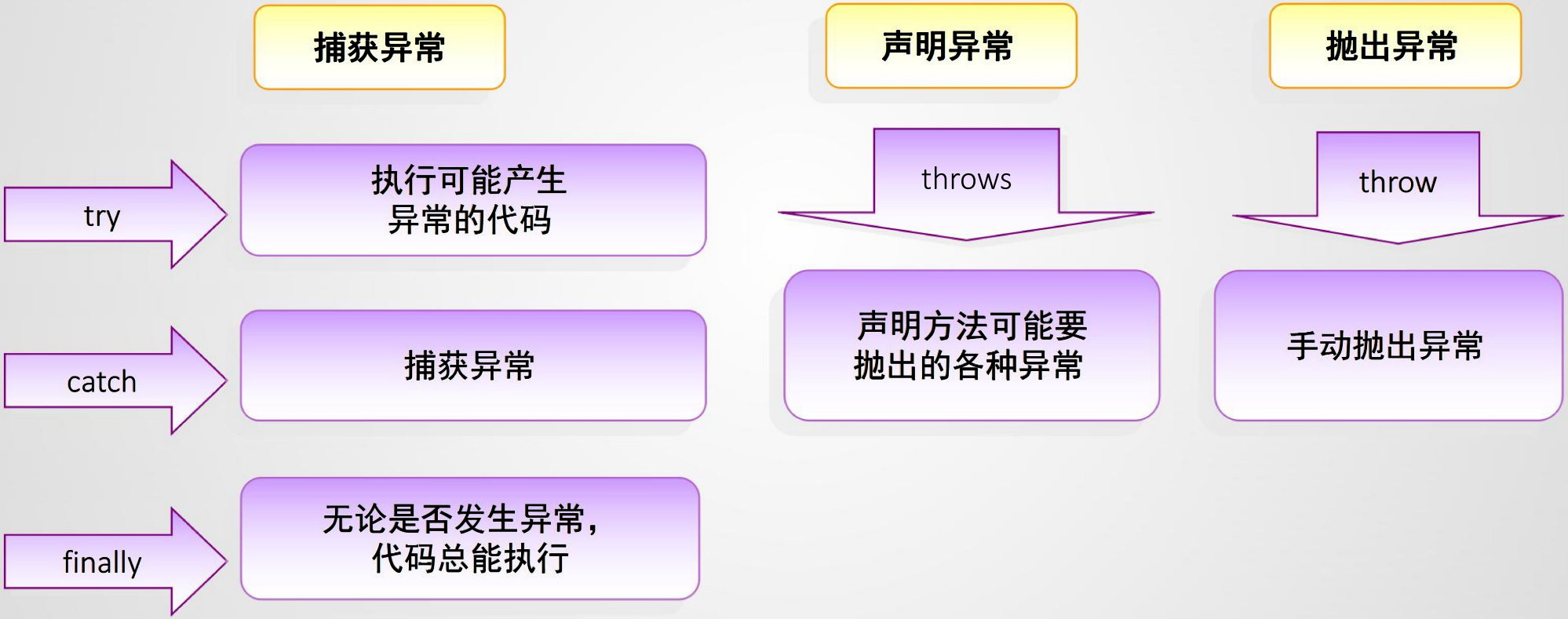


异常引入

- 异常（ Exception 也称例外）就是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
  - 所需文件找不到
  - 网络连接不通或中断
  - 算术运算错 (被零除...)
  - 数组下标越界
  - 装载一个不存在的类或者对null对象操作
  - 类型转换异常
  - .....
- 当Java程序出现以上的异常时，就会在所处的方法中产生一个异常对象。这个异常对象包括异常的类型，异常出现时程序的运行状态以及对该异常的详细描述。

异常引入

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



异常处理

- try-catch
  - 情况1：try块中代码没有出现异常
    - 不执行catch块代码，执行catch块后边的代码
  - 情况2：try块中代码出现异常，catch中异常类型匹配（相同或者父类）
    - 执行catch块代码，执行catch块后边的代码
  - 情况3：try块中代码出现异常， catch中异常类型不匹配
    - 不执行catch块代码，不执行catch块后边的代码，程序会中断运行
  - 注意
    - 出现异常后，Java会生成相应的异常对象，Java系统,寻找匹配的catch块，找到后将异常对象付给catch块异常参数
    - 出现异常后，try块中尚未执行的语句不会执行
    - 出现异常后并处理后，catch块后面的语句还会执行