第9章 异常处理

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java异常体系
- 异常的捕获和处理
  - try
  - catch
  - Finally
- 使用throws抛出异常
- 自定义异常类
  - 自定义
  - throw抛出

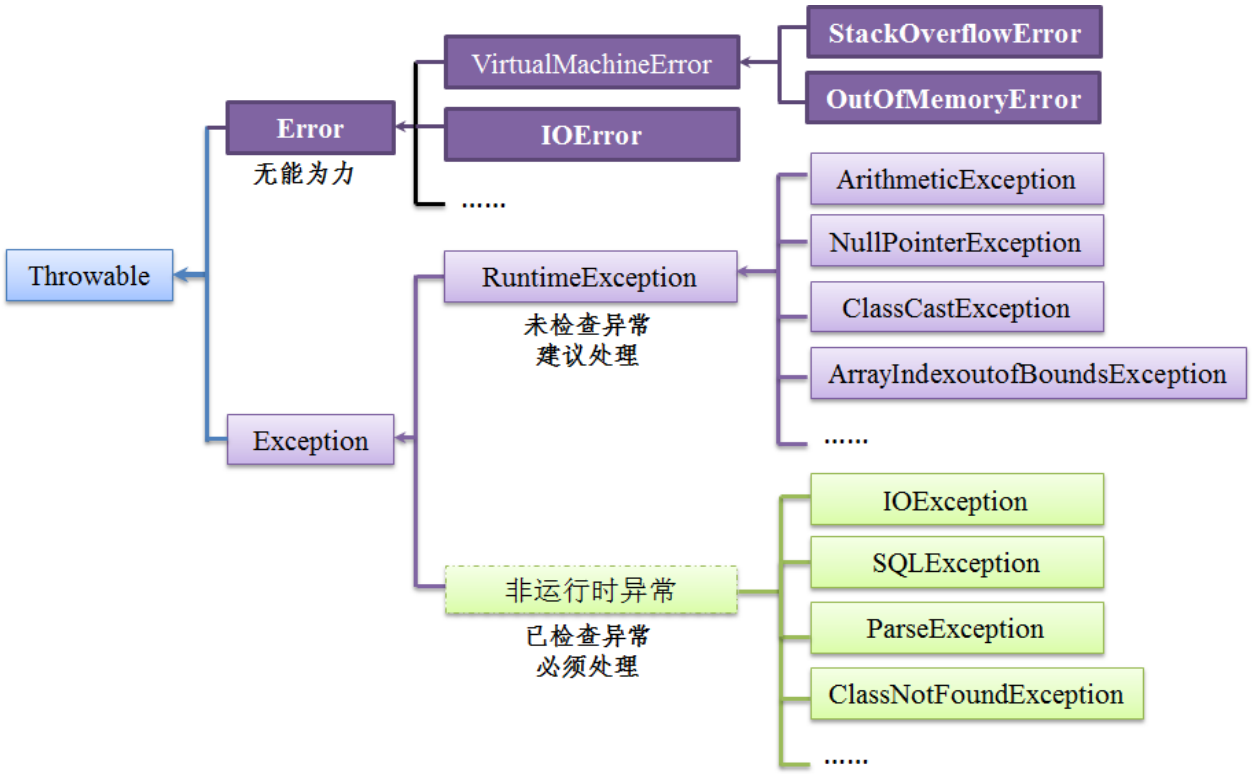
- 异常，是程序在运行期间发生的意外状况。

- 尽管任何一个程序员都不希望意外发生，但没有人能保证自己写的程序永远都不会出错；即便是程序没有错误，也不能保证使用程序的人不会输入程序不想要的数据；即便是使用程序的人十分配合，也不能保证运行程序的环境永远稳定，软件的问题、硬件的问题、网络的问题都有可能随时发生……所有的这些“不能保证”都会引发意外。

- 对于程序设计者需要尽可能地预见可能发生的意外，尽可能地保证程序在各种糟糕的情况下仍可以运行。

- Java语言提供了成熟的异常处理机制。

- Java将所有的错误封装成为一个对象，Throwable类是这个异常体系的根，它有两个子类：Error和Exception。



1．Error

- Error类及其子类对象代表了程序运行时Java系统内部的错误。对于Error，程序设计者无能为力，程序不能从Error恢复，因此不必处理它们，从技术上讲Error不是异常。

2．Exception

- Exception通常是由于某个资源不可用，或者正确执行程序所需的条件不满足所造成的。Exception类及其子类对象是程序设计者应该关心、并尽可能加以处理的部分。

- Java将Exception分为两类
  - RuntimeException（运行时异常，也称为未检查异常Unchecked Exception）
  - 非RuntimeException（也称为已检查异常 checked Exception），以下称未检查异常和已检查异常。

(1) 未检查异常

- 未检查异常包含java.lang.RuntimeException类及其所有子类。
- 未检查异常因为程序员没有进行必要的检查，因疏忽或错误而引起的异常，是可避免的。

(2) 已检查异常

- 除了java.lang.RuntimeException之外的所有异常都属于此类。
- 已检查异常是不可避免的，最常见的如IOException及其子类（像找不到文件的FileNotFoundException，意外到达文件尾部EOFException等都是它的子类）、数据库访问错误SQLException、解析时出现的异常ParseException等。

【例9-1】按“yyyy-MM-dd”格式输入一个人的生日并打印输出。

```
public static void main(String[] args) {
        Scanner scn  = new Scanner(System.in);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        System.out.print("输入生日(yyyy-MM-dd):");
        String birthStr = scn.next();

        try {
                Date birth = sdf.parse(birthStr);  //如果birthStr不可解析,
就抛出异常
                System.out.println("生日:"+sdf.format(birth));
        } catch (ParseException e) { //捕获异常
                System.out.println("日期格式错!");
        }
}
```

（1）try语句块

- 将可能产生异常的代码放在try语句块中尝试执行，异常发生之后的代码不会被执行。
- 在程序执行过程中，该段代码可能会产生并抛出一种或几种异常，这些异常都由它后面的catch负责捕获、处理。所以一个try语句块后面可以跟多个catch语句块（从语法的角度也可以一个catch都没有）。

(2) catch语句块

- 每个catch语句块捕获、处理一种类型的异常。当异常发生时，程序会中断正常的流程，离开try语句块去执行相应的catch语句块。
- 在catch中声明了异常对象（如ParseException e），异常对象封装了异常事件的相关信息，在catch语句块中可以使用这个对象获取这些信息，常用方法包括：
  - getMessage()：返回该异常的详细描述字符串。
  - printStackTrace()：将异常事件的跟踪栈信息输出。

（2）catch语句块

- 一般情况下每个catch语句块用于捕获、处理一种类型的异常。从Java SE 7开始允许catch语句一次捕获多个Exception，即当出现的多个异常采取同样的处理措施时，则将多个异常见用“|”分开写在一个catch的后面，注意这个catch中的多个异常不允许出现继承关系。

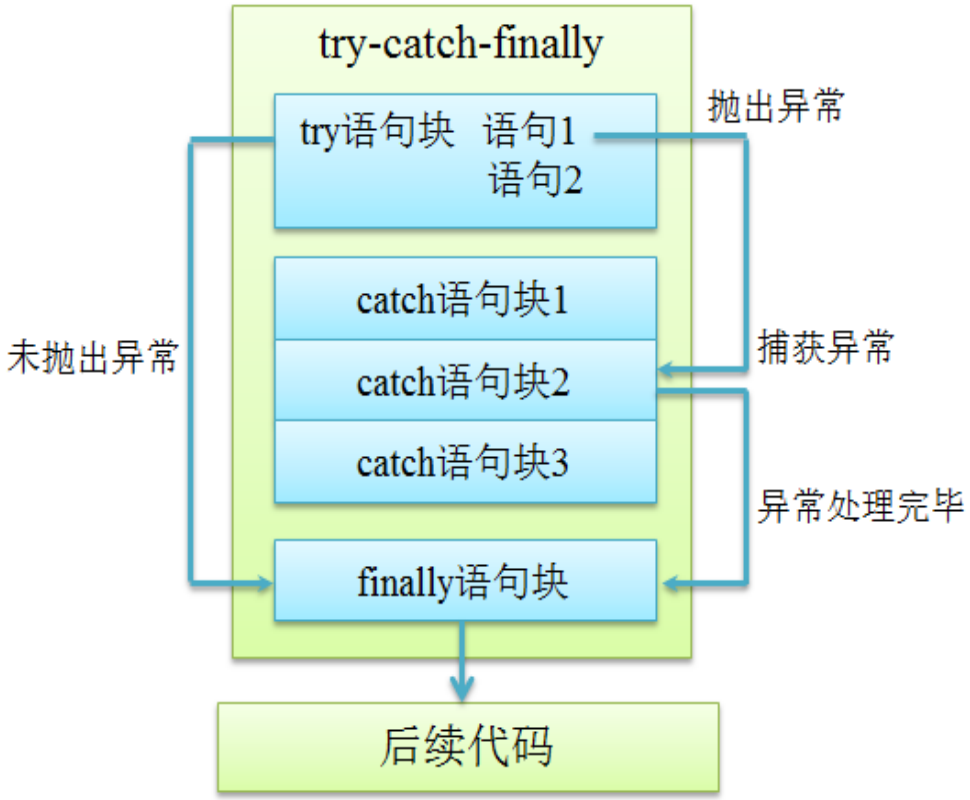
（3）finally语句块

- finally语句块为可选，一旦出现，无论try语句块是否抛出异常，finally代码块都要被执行。finally语句块为异常处理提供统一的善后处理，使流程转到其他部分之前，能够对程序的状态进行统一的管理。通常在finally语句块中进行资源释放的工作，如关闭已打开的文件、关闭数据库连接等。

【例9-2】使用Properties读取配置文件ipConfig.properties，并进行异常的捕获、处理。

【例9-3】用异常完善猜数游戏。

9.2.2 try-catch-finally语句的执行过程



9.2.2 try-catch-finally语句的执行过程

【例9-4】分析带return的try-catch-finally代码块的执行过程。

- 如果当前的方法不知道如何处理捕获到的异常，可以使用throws声明将此异常再次抛出，交给当前方法的上一层调用者，如果上层方法仍不知道如何处理，也可以继续向上抛出，直到某个方法可以处理此异常，或者最终将异常交给了JVM。异常对象沿着方法调用链进行反向传递。JVM对异常的处理方法是，打印异常的跟踪信息栈，并终止程序运行。

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

1. 基本规则

- throws写在方法签名之后，语法格式如下：

  throws Exception1，Exception2……

【例9-5】使用throws抛出异常。

2. 子类方法重写父类方法时throws的规则

- Java规定，子类方法重写父类方法时，子类方法抛出的异常类型不能比父类方法抛出的异常类型更宽泛。也就是说，子类方法可以：
  （1）抛出与父类方法相同的异常。
  （2）抛出父类方法抛出异常的子类。
  （3）不抛出异常。

【例9-6】分析下面代码中哪个类不能通过编译?

```
public class CC {
        void doStuff() throws IOException{}
}
public class CC2 extends CC{
        void doStuff() throws FileNotFoundException {}
}
public class CC3 extends CC{
        void doStuff() throws Exception{}
}
public class CC4 extends CC{
        void doStuff(int x) throws Exception{}
}
public class CC5 extends CC{
        void doStuff(){}
}
```

- 异常的名字是异常信息的一种表现，从名字可以读出异常的原因，所以在应用程序中往往根据业务处理的需要设计与业务状态相关的自定义异常类。

9.4.1 自定义异常类的方法

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 创建自定义异常类：
  （1）为该异常类取一个能标识异常状况的有意义的名字。
  （2）令其继承Exception类。
  （3）在异常类中至少定义两个构造方法：一个是无参的；另一个是带String参数的，将此字符串传递给父类Exception的相同构造方法。这个String将作为该异常对象的描述信息（即getMessage()方法的返回值）。

【例9-7】自定义异常类，标识一个用户管理系统中因用户名（email）已存在而注册失败的情况。

9.4.2 throw抛出异常

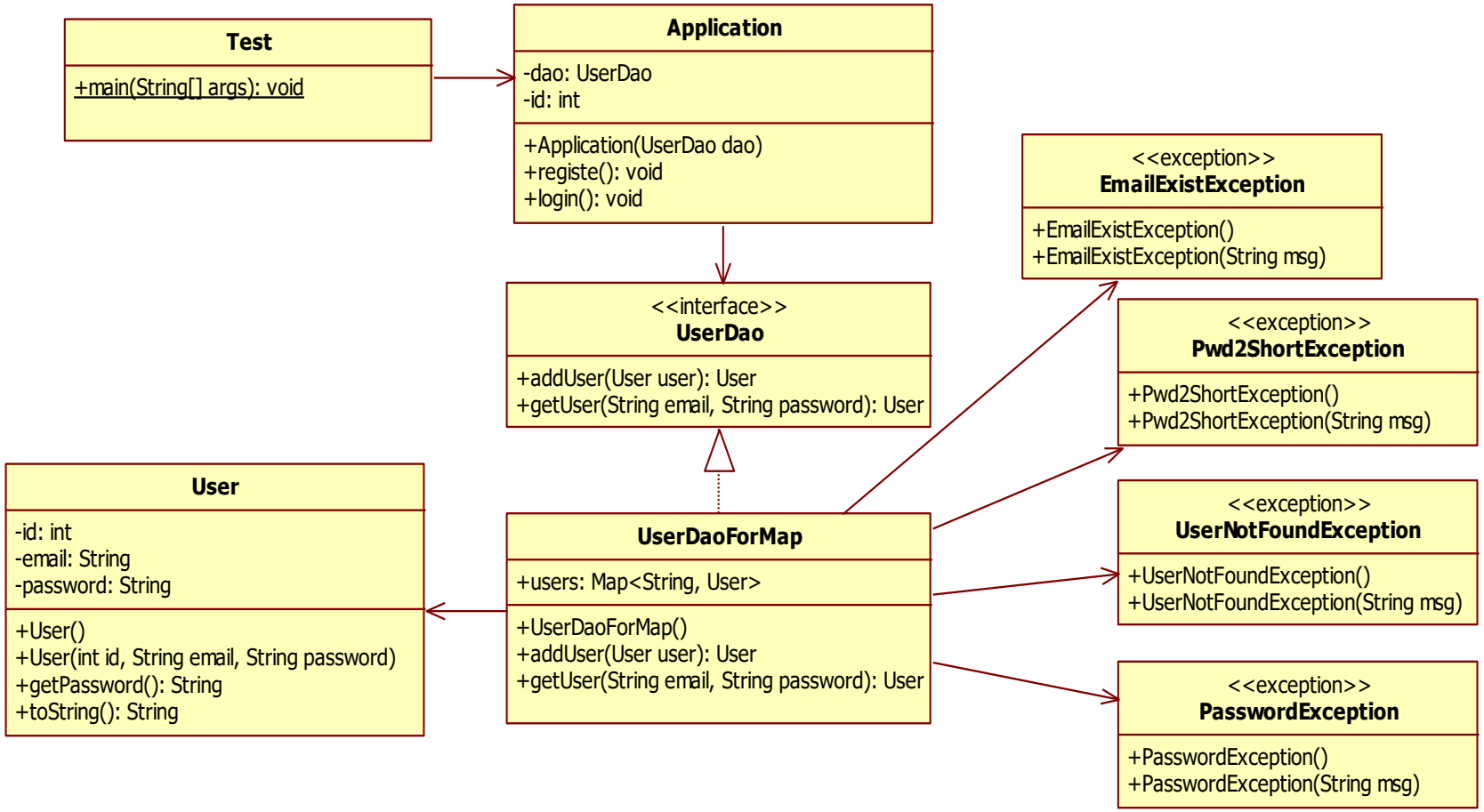
- Java异常体系中的异常都是在运行时由系统抛出的，用户自定义的异常必须自行抛出。
  - 自行抛出异常使用throw语句，它抛出的不是异常类，而是异常对象，每次只能抛出一个异常对象。
  - throw的语法格式如下：

    throw 异常对象;
  - 一旦执行throw语句，其后的代码都不会被执行。

【例9-8】自行抛出自定义异常EmailExistException。

9.4.3 异常处理的5个关键字

9.5 综合实践—用户管理系统及其异常类设计



本章思维导图


异常
I. Java异常体系
Throwable
Error
无需处理
Exception
RuntimeException
建议在程序设计阶段处理
非运行时异常
必须处理
II. 捕获和处理
try
包围可能产生异常的代码段
catch
捕获异常
异常对象
getMessage()
printStackTrace()
异常按先子类异常再父类异常的顺序排列
finally
未必会被执行
对程序状态进行统一管理，通常用于释放资源
return不能阻止finally的执行
III. 再次抛出异常
throws
原则：谁知情谁处理
IV. 自定义异常
意义
标识系统中的意外状况
分离业务代码和意外处理代码
定义异常
用有标识作用的名字命名异常
继承Exception
提供构造方法
抛出自定义异常
throw
V. 案例
【例9-1】输入生日并打印
ParseException
【例9-2】使用Properties读取配置文件
try-catch的组织
【例9-3】用异常完善猜数游戏
【例9-4】分析代码块执行过程
return与finally关系
【例9-5】使用throws抛出异常
【例9-6】分析代码
子类重写父类方法时异常的处理
【例9-7】自定义异常类
【例9-8】抛出自定义异常
【综合实践】用户管理系统及其异常类设计
自定义异常
抛出自定义异常