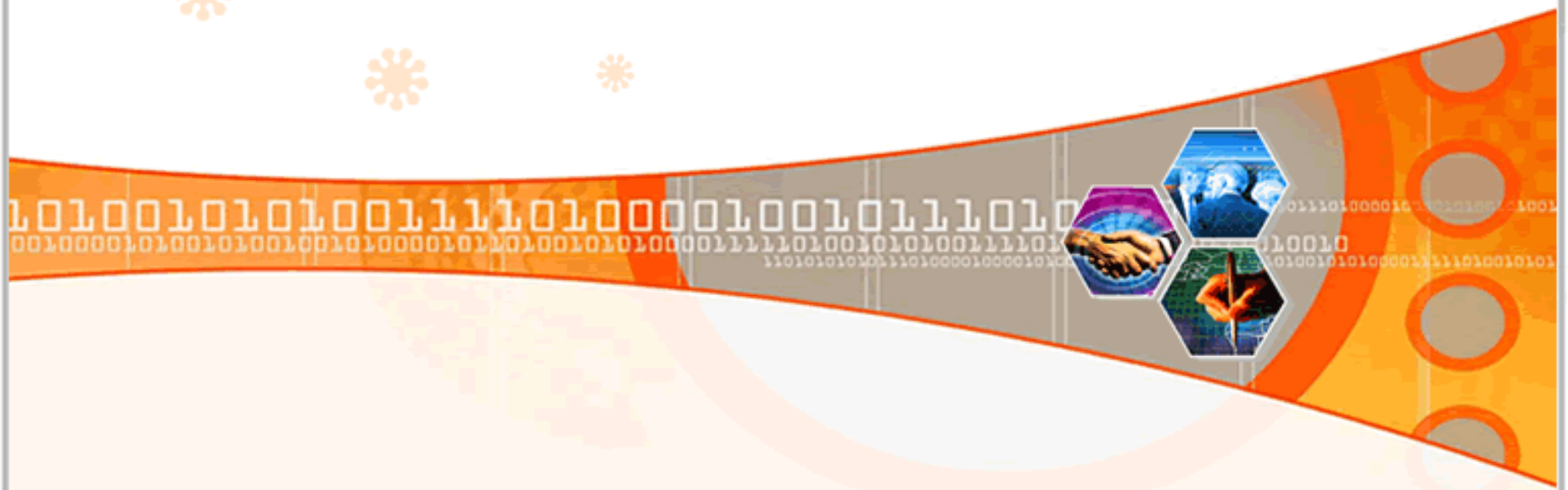
第17章

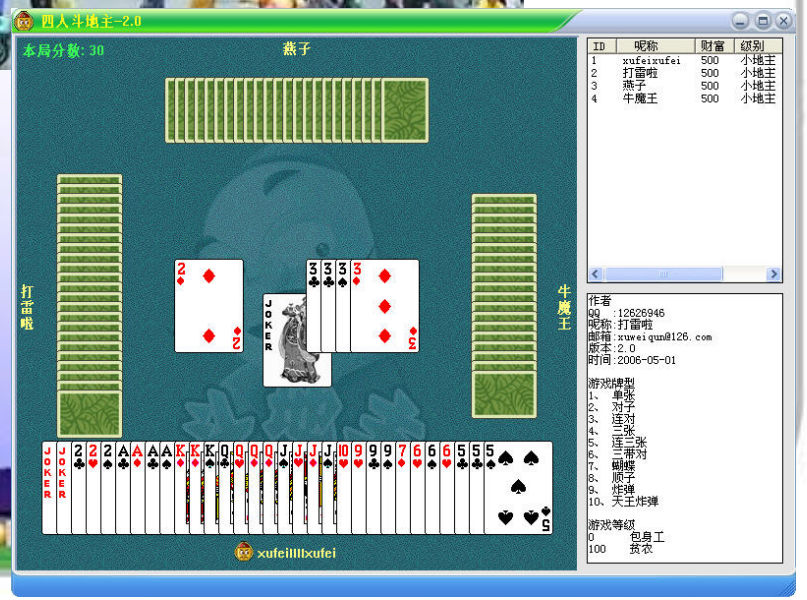
职责链模式



主讲教师：程细柱　韶关学院计算机系
本书主编：刘　伟　　清华大学出版社

本章教学内容

- 行为型模式
  - ✓ 行为型模式概述
  - ✓ 行为型模式简介
- 职责链模式
  - ✓ 模式动机与定义
  - ✓ 模式结构与分析
  - ✓ 模式实例与解析
  - ✓ 模式效果与应用
  - ✓ 模式扩展

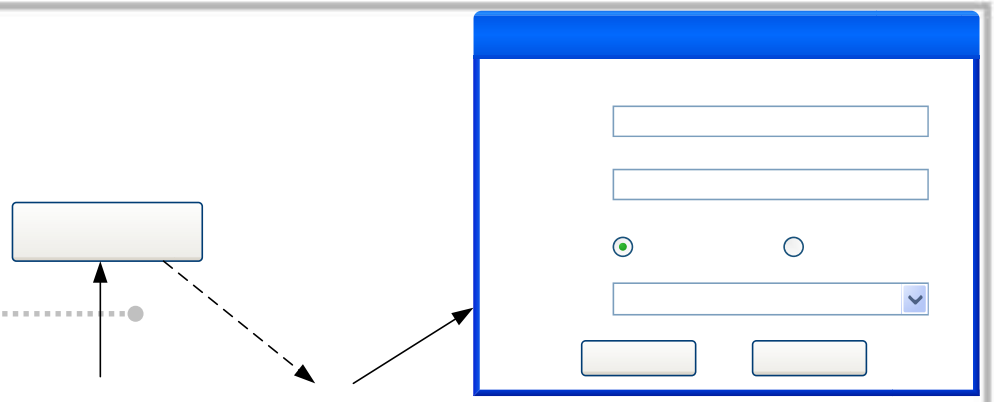


行为型模式

- 行为型模式概述
  - ✓ 行为型模式(Behavioral Pattern)是对在不同的对象之间划分责任和算法的抽象化。
  - ✓ 行为型模式不仅仅关注类和对象的结构，而且重点关注它们之间的相互作用。

  - ✓ 通过行为型模式，可以更加清晰地划分类与对象的职责，并研究系统在运行时实例对象之间的交互。在系统运行时，对象并不是孤立的，它们可以通过相互通信与协作完成某些复杂功能，一个对象在运行时也将影响到其他对象的运行。

行为型模式



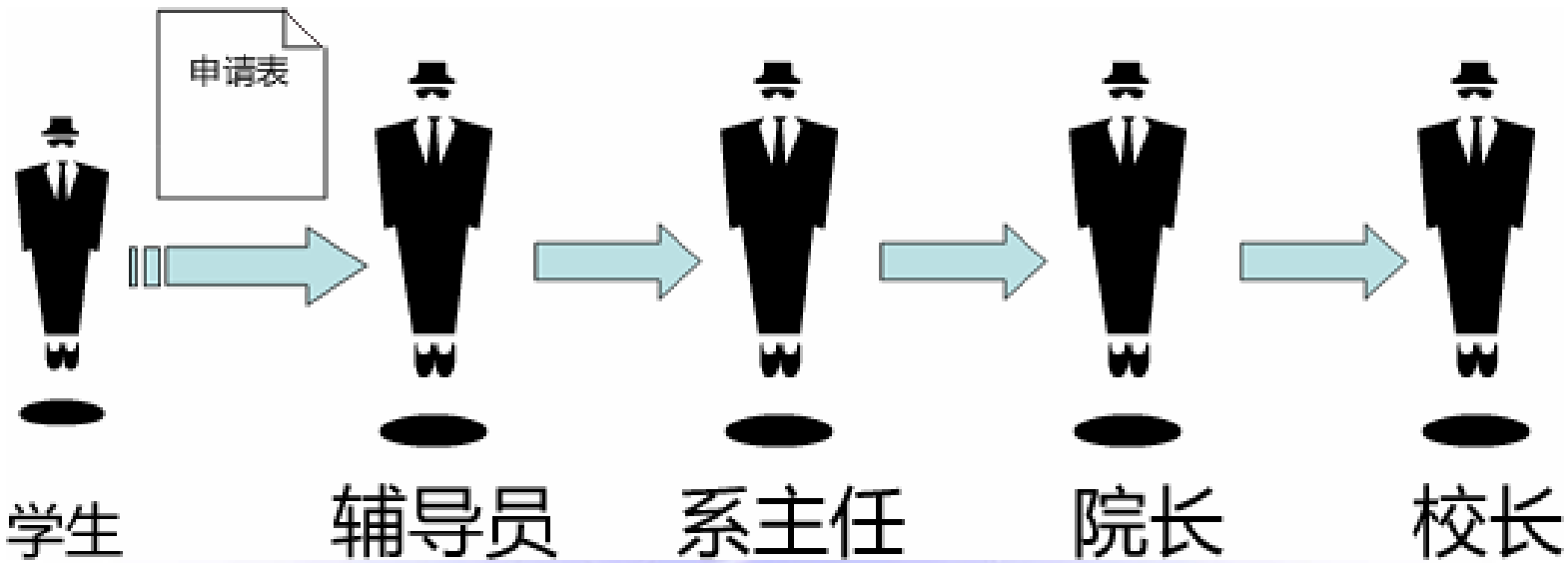
- **行为型模式概述**
  - ✓ 行为型模式分为类行为型模式和对象行为型模式两种：
    - 类行为型模式：类的行为型模式使用继承关系在几个类之间分配行为，类行为型模式主要通过多态等方式来分配父类与子类的职责。
    - 对象行为型模式：对象的行为型模式则使用对象的聚合关联关系来分配行为，对象行为型模式主要是通过对象关联等方式来分配两个或多个类的职责。根据“合成复用原则”，系统中要尽量使用关联关系来取代继承关系，因此大部分行为型设计模式都属于对象行为型设计模式。

行为型模式

- 行为型模式简介
  - ✓ 职责链模式(Chain of Responsibility)
  - ✓ 命令模式(Command)
  - ✓ 解释器模式(Interpreter)
  - ✓ 迭代器模式(Iterator)
  - ✓ 中介者模式(Mediator)
  - ✓ 备忘录模式(Memento)
  - ✓ 观察者模式(Observer)
  - ✓ 状态模式(State)
  - ✓ 策略模式(Strategy)
  - ✓ 模板方法模式(Template Method)
  - ✓ 访问者模式(Visitor)

职责链模式

- 职责链模式的模式动机



职责链模式

- **模式动机**
  - ✓职责链可以是一条直线、一个环或者一个树形结构，最常见的职责链是直线型，即沿着一条单向的链来传递请求。
  - ✓链上的每一个对象都是请求处理者，职责链模式可以将请求的处理者组织成一条链，并使请求沿着链传递，由链上的处理者对请求进行相应的处理，客户端无须关心请求的处理细节以及请求的传递，只需将请求发送到链上即可，将请求的发送者和请求的处理者解耦。这就是职责链模式的模式动机。

职责链模式

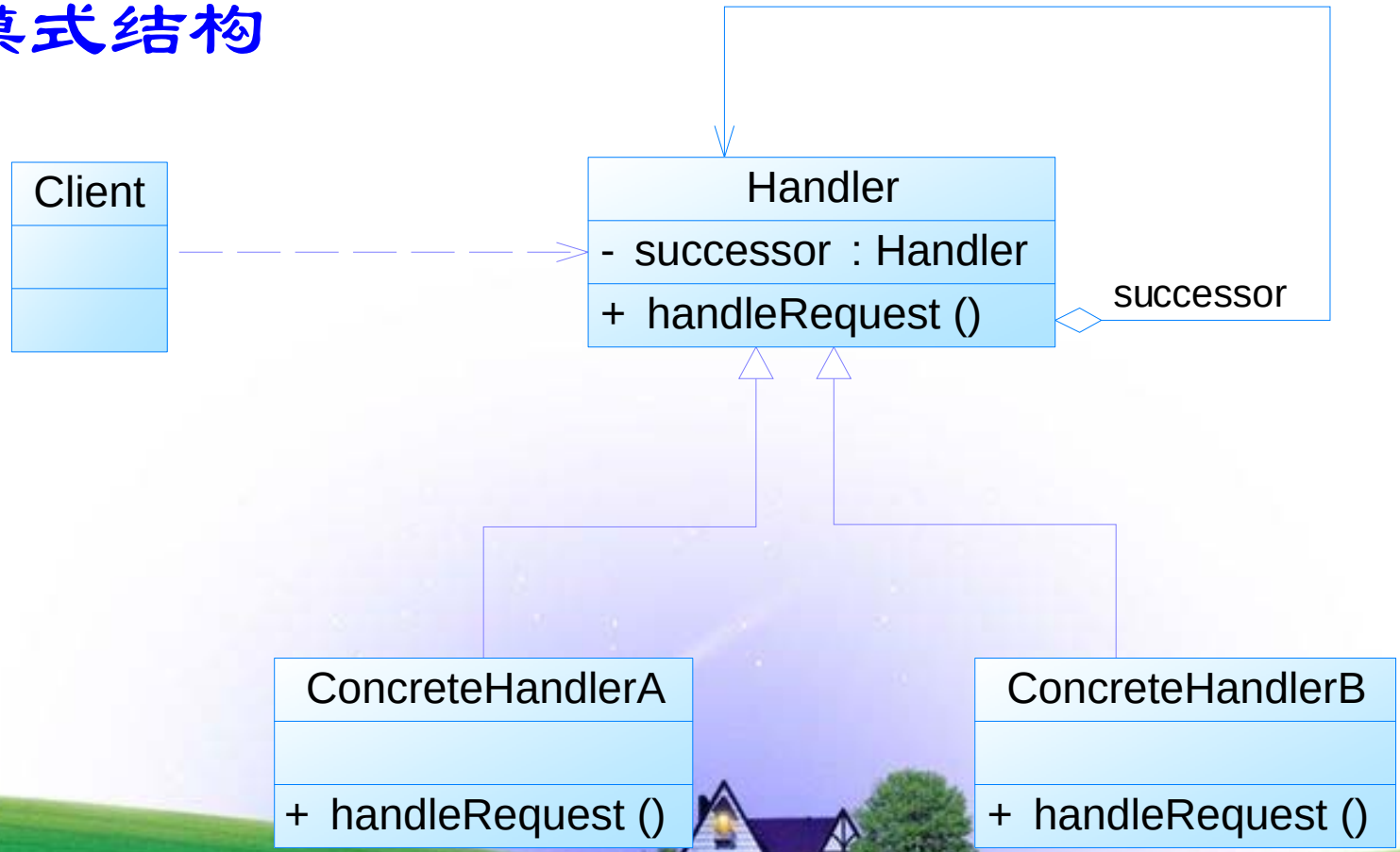
- **模式定义**
  - ✓职责链模式(**Chain of Responsibility Pattern**)：避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止。由于英文翻译的不同，职责链模式又称为责任链模式，它是一种对象行为型模式。

职责链模式

- 模式定义
  - ✓ **Chain of Responsibility Pattern: Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.**
  - ✓ **Frequency of use: medium low**

职责链模式

- 模式结构


Client
Handler
- successor : Handler
+ handleRequest ()
successor
ConcreteHandlerA
+ handleRequest ()
ConcreteHandlerB
+ handleRequest ()

职责链模式

- 模式结构
  - ✓职责链模式包含如下角色:
    - Handler: 抽象处理者
    - ConcreteHandler: 具体处理者
    - Client: 客户类

职责链模式

- 模式分析
  - ✓在职责链模式里，很多对象由每一个对象对其下家的引用而连接起来形成一条链。
  - ✓请求在这条链上传递，直到链上的某一个对象处理此请求为止。
  - ✓发出这个请求的客户端并不知道链上的哪一个对象最终处理这个请求，这使得系统可以在不影响客户端的情况下动态地重新组织链和分配责任。

职责链模式

- 模式分析
  - ✓典型的抽象处理者代码:

```
public abstract class Handler
{
    protected Handler successor;
    public void setSuccessor(Handler successor)
    {     this.successor=successor;     }
    public abstract void handleRequest(String request);
}
```

职责链模式

- 模式分析
  - ✓典型的具体处理者代码:

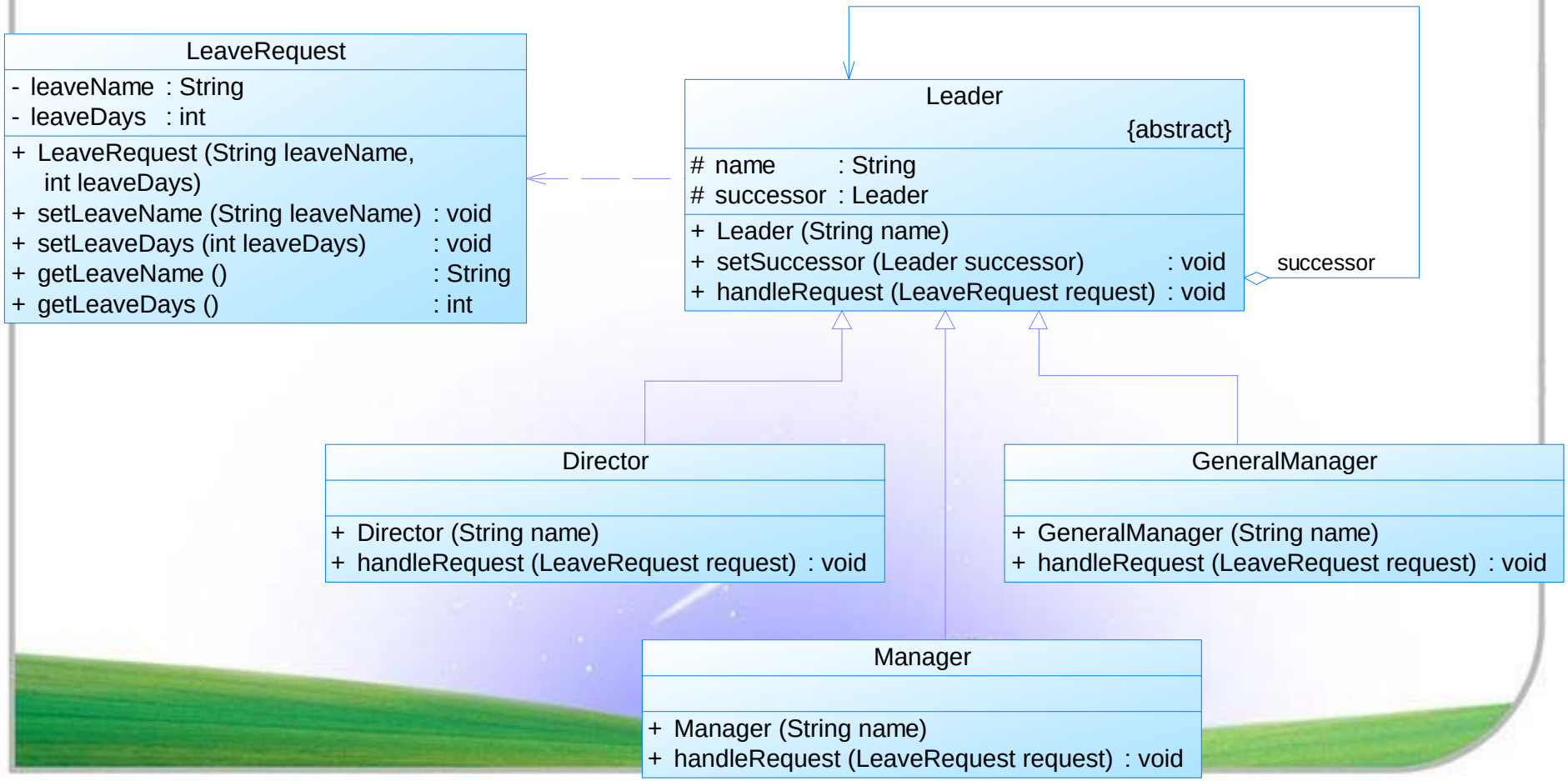
```
public class ConcreteHandler extends Handler
{
    public void handleRequest(String request)
    {
        if(请求request满足条件)
        {       ......  //处理请求;      }
        else{
                this.successor.handleRequest(request); //转发请求
        }
    }
}
```

职责链模式

- **职责链模式实例与解析**
  - ✓ 实例：审批假条
    - 某OA系统需要提供一个假条审批的模块，如果员工请假天数小于3天，主任可以审批该假条；如果员工请假天数大于等于3天，小于10天，经理可以审批；如果员工请假天数大于等于10天，小于30天，总经理可以审批；如果超过30天，总经理也不能审批，提示相应的拒绝信息。

职责链模式

- 职责链模式实例与解析



职责链模式

- 职责链模式实例与解析
  - ✓ 实例：审批假条
    - 参考代码：Chapter 17 CoR\sample01
    - 下载地址：http://download.csdn.net/user/cflynn

演示.....

职责链模式

- **模式优缺点**
  - ✓职责链模式的优点
    - 降低耦合度
    - 可简化对象的相互连接
    - 增强给对象指派职责的灵活性
    - 增加新的请求处理类很方便

职责链模式

- **模式优缺点**
  - ✓职责链模式的缺点
    - 不能保证请求一定被接收。
    - 系统性能将受到一定影响，而且在进行代码调试时不太方便；可能会造成循环调用。

职责链模式

- **模式适用环境**

  ✓在以下情况下可以使用职责链模式：

  - 有多个对象可以处理同一个请求，具体哪个对象处理该请求由运行时刻自动确定。
  - 在不明确指定接收者的情况下，向多个对象中的一个提交一个请求。
  - 可动态指定一组对象处理请求。

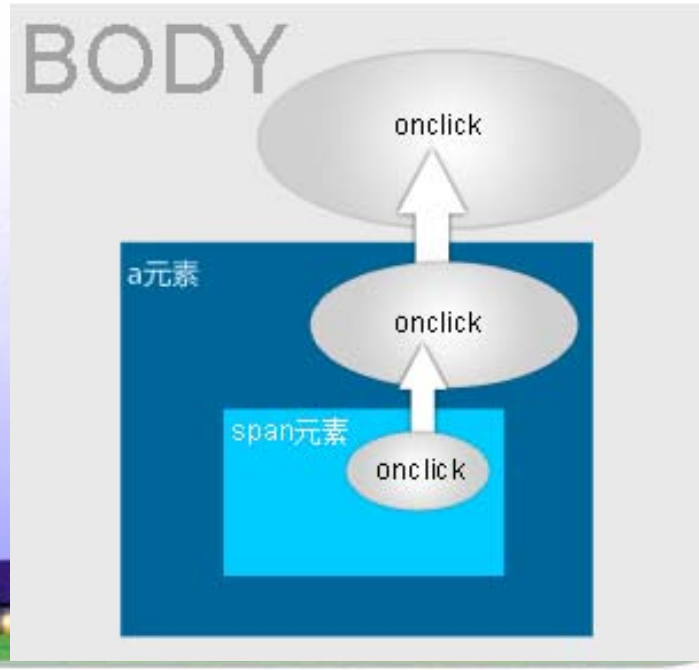
职责链模式

- 模式应用
  - ✓ (1) Java中的异常处理机制

```
try
{   ……  }
catch(ArrayIndexOutOfBoundsException e1)
{   ……  }
catch(ArithmeticException e2)
{   ……  }
catch(IOException e3)
{   ……  }
finally
{   ……  }
```

职责链模式

- **模式应用**
  - ✓ **(2) 早期的Java AWT事件模型(JDK 1.0及更早)：事件浮升(Event Bubbling)机制。**
  - ✓ **JavaScript事件浮升机制：**



职责链模式

◆ 模式扩展

✓ 纯与不纯的职责链模式

- 一个纯的职责链模式要求一个具体处理者对象只能在两个行为中选择一个：一个是承担责任，另一个是把责任推给下家。不允许出现某一个具体处理者对象在承担了一部分责任后又将责任向下传的情况。
- 在一个纯的职责链模式里面，一个请求必须被某一个处理者对象所接收；在一个不纯的职责链模式里面，一个请求可以最终不被任何接收端对象所接收。

本章小结

- 行为型模式是对在不同的对象之间划分责任和算法的抽象化。行为型模式不仅仅关注类和对象的结构，而且重点关注它们之间的相互作用。通过行为型模式，可以更加清晰地划分类与对象的职责，并研究系统在运行时实例对象之间的交互。行为型模式可以分为类行为型模式和对象行为型模式两种。
- 职责链模式可以避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止。它是一种对象行为型模式。

本章小结

- 职责链模式包含两个角色：抽象处理者定义了一个处理请求的接口；具体处理者是抽象处理者的子类，它可以处理用户请求。
- 在职责链模式里，很多对象由每一个对象对其下家的引用而连接起来形成一条链。请求在这个链上传递，直到链上的某一个对象决定处理此请求。发出这个请求的客户端并不知道链上的哪一个对象最终处理这个请求，这使得系统可以在不影响客户端的情况下动态地重新组织链和分配责任。

本章小结

- 职责链模式的主要优点在于可以降低系统的耦合度，简化对象的相互连接，同时增强给对象指派职责的灵活性，增加新的请求处理类也很方便；其主要缺点在于不能保证请求一定被接收，且对于比较长的职责链，请求的处理可能涉及到多个处理对象，系统性能将受到一定影响，而且在进行代码调试时不太方便。
- 职责链模式适用情况包括：有多个对象可以处理同一个请求，具体哪个对象处理该请求由运行时刻自动确定；在不明确指定接收者的情况下，向多个对象中的一个提交一个请求；可动态指定一组对象处理请求。

END

Thanks!