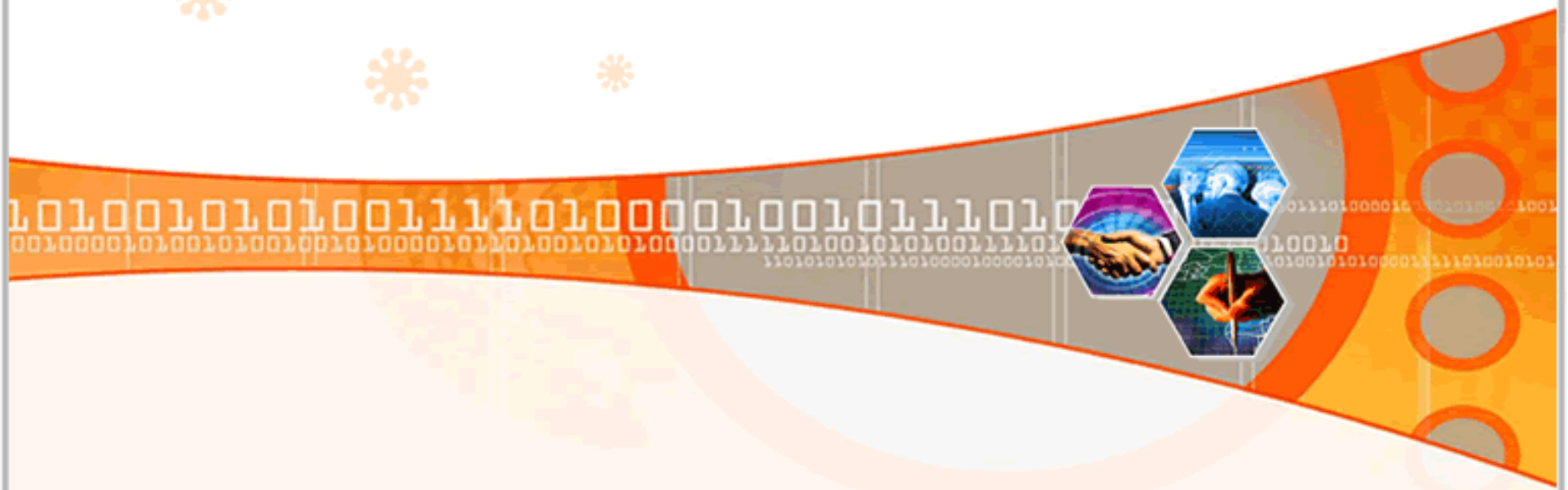
第16章

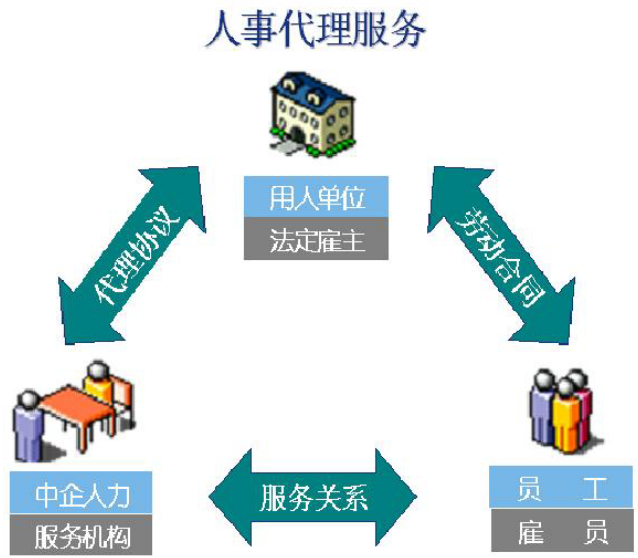
代理模式



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

- 代理模式
  - ✓ 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



代理模式

- **模式动机**
  - ✓在某些情况下，一个客户不想或者不能直接引用一个对象，此时可以通过一个称之为“代理”的第三者来实现间接引用。代理对象可以在客户端和目标对象之间起到中介的作用，并且可以通过代理对象去掉客户不能看到的内容和服务或者添加客户需要的额外服务。

代理模式

- 模式动机

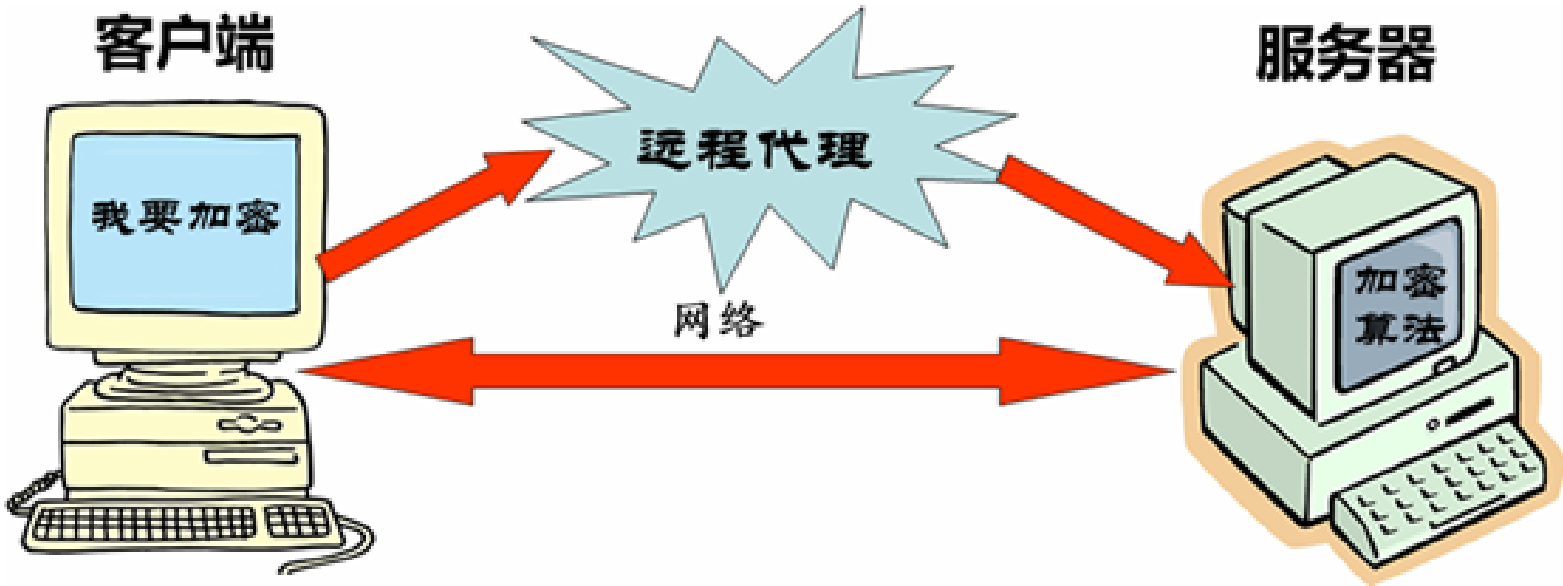
小图片

代理

大图片

代理模式

- 模式动机



代理模式

◆ 模式动机

✓通过引入一个新的对象（如小图片和远程代理对象）来实现对真实对象的操作或者将新的对象作为真实对象的一个替身，这种实现机制即为代理模式，通过引入代理对象来间接访问一个对象，这就是代理模式的模式动机。

代理模式

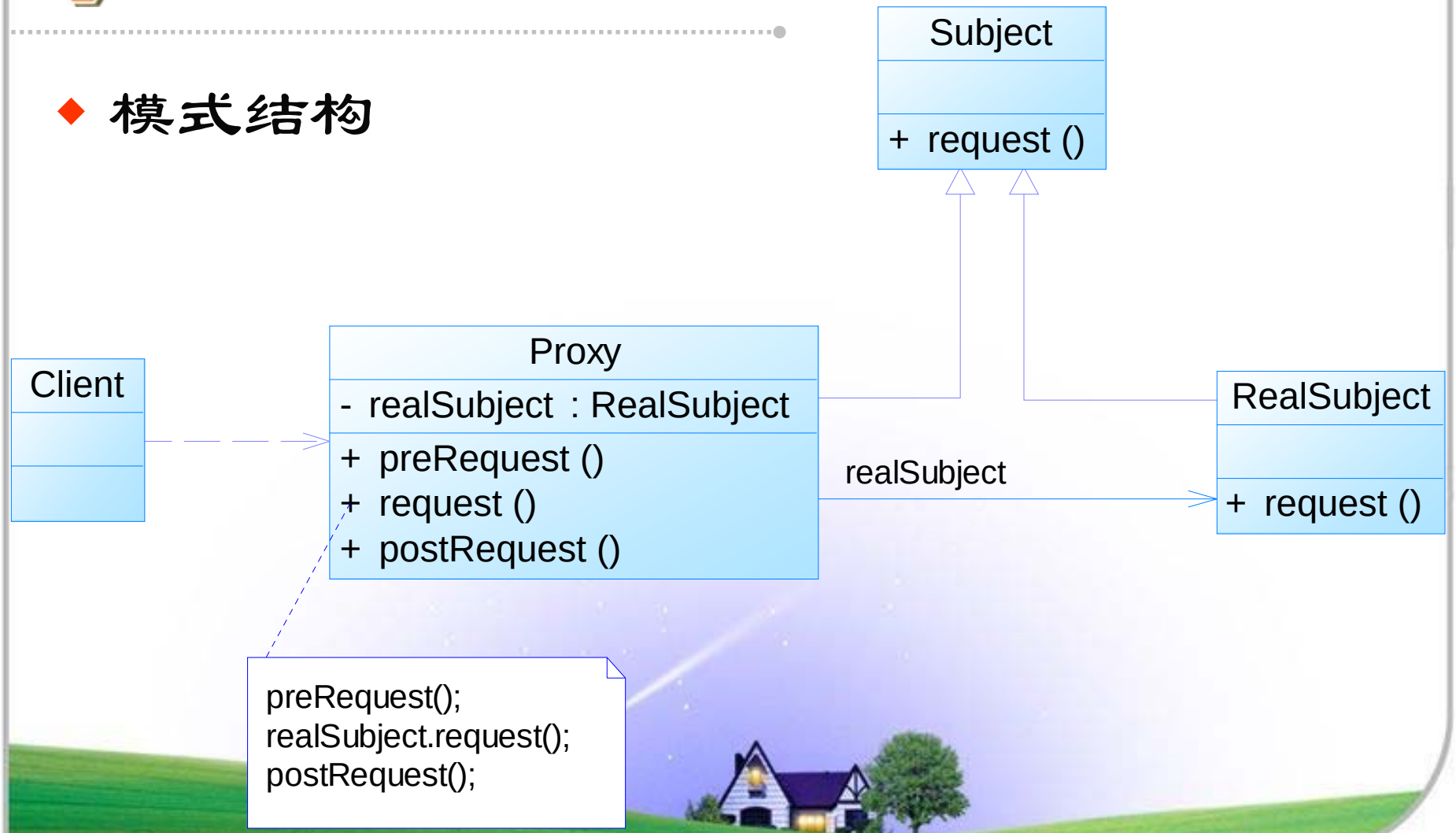
- 模式定义
  - ✓代理模式(Proxy Pattern)：给某一个对象提供一个代理，并由代理对象控制对原对象的引用。代理模式的英文叫做Proxy或Surrogate，它是一种对象结构型模式。

代理模式

- 模式定义
  - ✓ **Proxy Pattern:** Provide a **surrogate or placeholder** for **another** object to **control access** to it.
  - ✓ **Frequency of use: medium high**

代理模式

- 模式结构

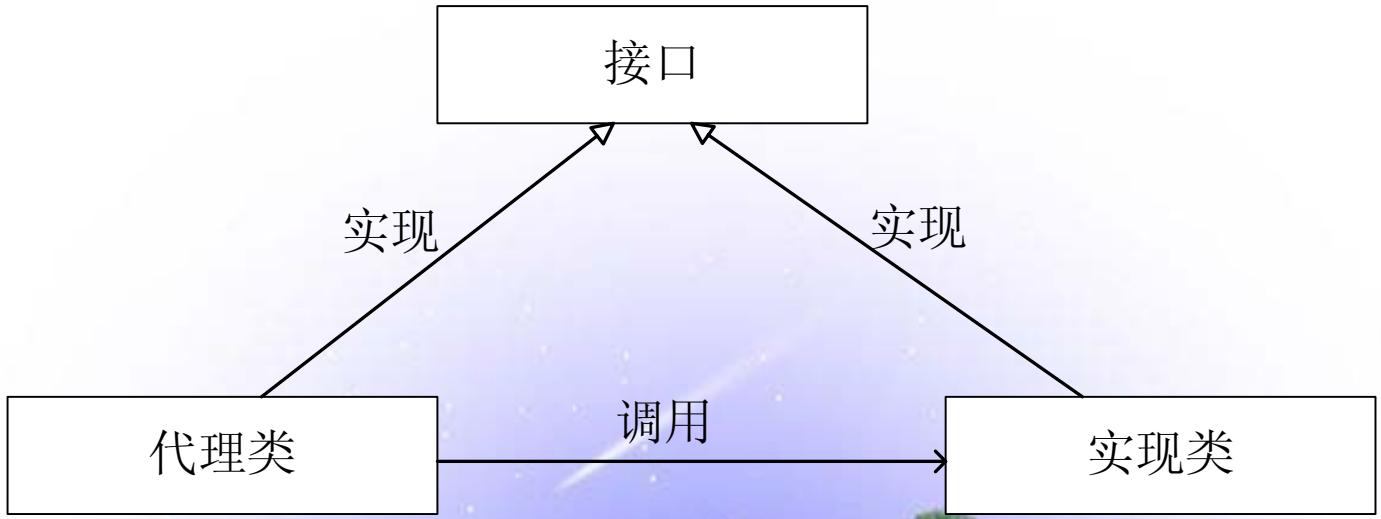

Subject
+ request ()
Client
Proxy
- realSubject : RealSubject
+ preRequest ()
+ request ()
+ postRequest ()
realSubject
RealSubject
+ request ()
preRequest();
realSubject.request();
postRequest();

代理模式

- **模式结构**
  - ✓代理模式包含如下角色:
    - **Subject: 抽象主题角色**
    - **Proxy: 代理主题角色**
    - **RealSubject: 真实主题角色**

代理模式

- **模式分析**
  - ✓代理模式示意结构图比较简单，一般可以简化为如下图所示，但是在现实中要复杂很多。



代理模式

- 模式分析
  - ✓ 典型的代理类实现代码:

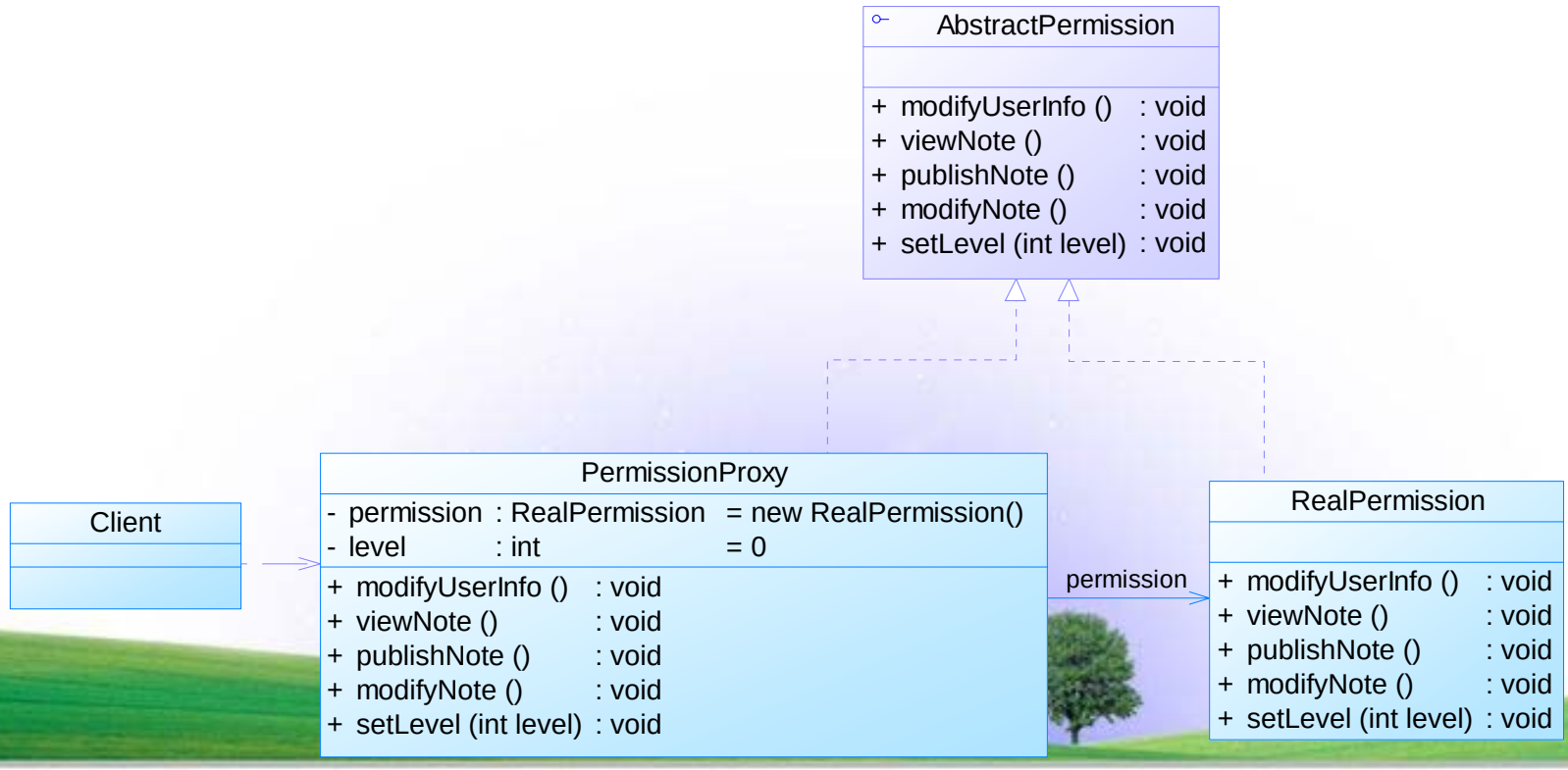
```
public class Proxy implements Subject {
    private RealSubject realSubject = new RealSubject();
    public void preRequest()
    {……}
    public void request()  {
        preRequest();
        realSubject.request();
        postRequest();
    }
    public void postRequest()
    {……}
}
```

代理模式

- **代理模式实例与解析**
  - ✓ 实例一：论坛权限控制代理
    - 在一个论坛中已注册用户和游客的权限不同，已注册的用户拥有发帖、修改自己的注册信息、修改自己的帖子等功能；而游客只能看到别人发的帖子，没有其他权限。使用代理模式来设计该权限管理模块。
    - 在本实例中我们使用代理模式中的保护代理，该代理用于控制对一个对象的访问，可以给不同的用户提供不同级别的使用权限。

代理模式

- 代理模式实例与解析
  - ✓ 实例一：论坛权限控制代理



代理模式

- **代理模式实例与解析**
  - ✓ 实例一：论坛权限控制代理
    - 参考代码：**Chapter 16 Proxy\sample01**
    - 下载地址：**http://download.csdn.net/user/cflynn**

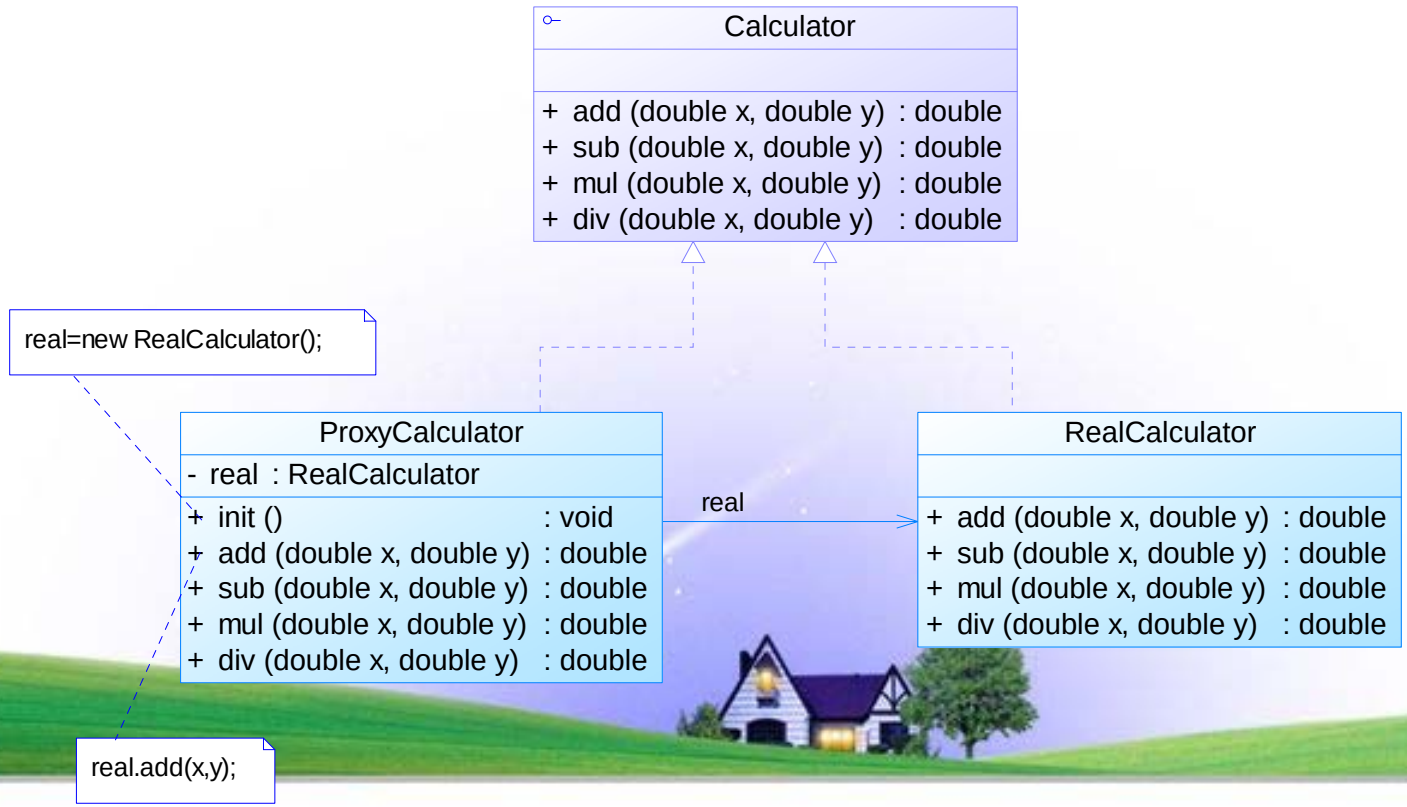
演示.....

代理模式

- **代理模式实例与解析**
  - ✓ 实例二：数学运算代理
    - 模拟应用远程代理来访问另外一个应用程序域中的对象，如果在远程实现了加减乘除等运算，在本地需要调用，那么可以考虑在本地设置一个代理。

代理模式

- 代理模式实例与解析
  - ✓实例二：数学运算代理



代理模式

- **模式优缺点**
  - ✓代理模式的优点
    - 代理模式能够协调调用者和被调用者，在一定程度上降低了系统的耦合度。
    - 远程代理使得客户端可以访问在远程机器上的对象，远程机器可能具有更好的计算性能与处理速度，可以快速响应并处理客户端请求。
    - 虚拟代理通过使用一个小对象来代表一个大对象，可以减少系统资源的消耗，对系统进行优化并提高运行速度。
    - 保护代理可以控制对真实对象的使用权限。

代理模式

- **模式优缺点**

  ✓代理模式的缺点

  - 由于在客户端和真实主题之间增加了代理对象，因此有些类型的代理模式可能会造成请求的处理速度变慢。
  - 实现代理模式需要额外的工作，有些代理模式的实现非常复杂。

代理模式

模式适用环境

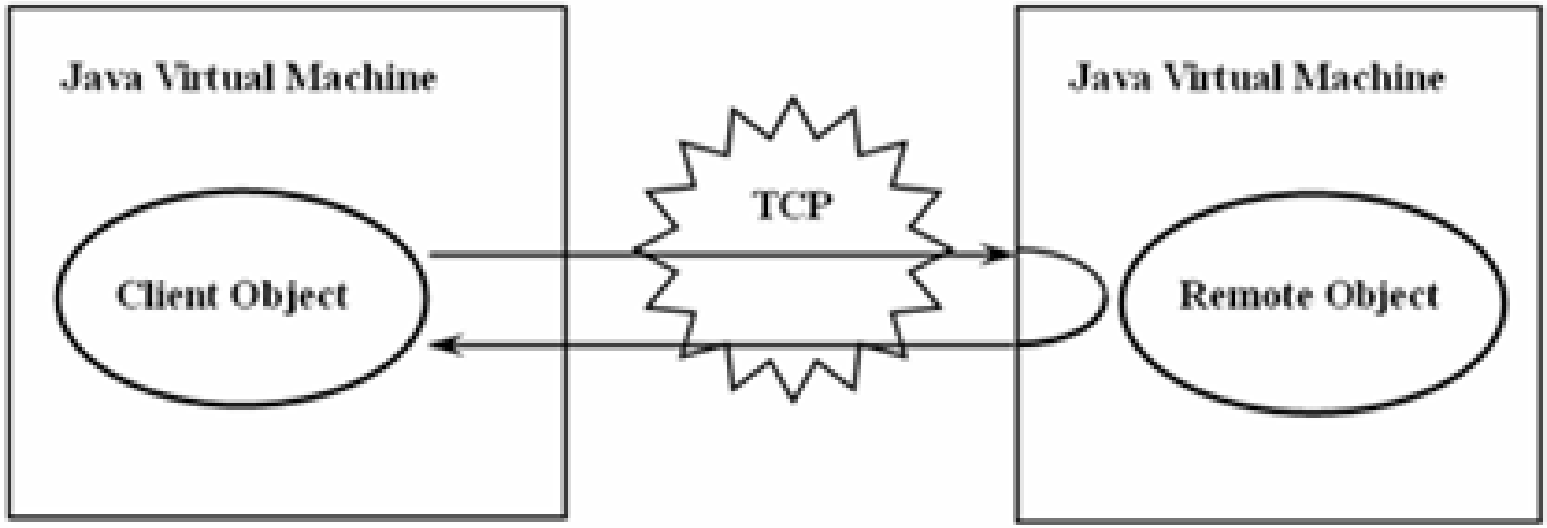
- 根据代理模式的使用目的，常见的代理模式有以下几种类型：
  - 远程(Remote)代理：为一个位于不同的地址空间的对象提供一个本地的代理对象，这个不同的地址空间可以是在同一台主机中，也可是在另一台主机中，远程代理又叫做大使(Ambassador)。
  - 虚拟(Virtual)代理：如果需要创建一个资源消耗较大的对象，先创建一个消耗相对较小的对象来表示，真实对象只在需要时才会被真正创建。
  - Copy-on-Write代理：它是虚拟代理的一种，把复制（克隆）操作延迟到只有在客户端真正需要时才执行。一般来说，对象的深克隆是一个开销较大的操作，Copy-on-Write代理可以让这个操作延迟，只有对象被用到的时候才被克隆。

代理模式

- **模式适用环境**
  - ✓ 根据代理模式的使用目的，代理模式有以下几种类型（续）：
    - 保护(Protect or Access)代理：控制对一个对象的访问，可以给不同的用户提供不同级别的使用权限。
    - 缓冲(Cache)代理：为某一个目标操作的结果提供临时的存储空间，以便多个客户端可以共享这些结果。
    - 防火墙(Firewall)代理：保护目标不让恶意用户接近。
    - 同步化(Synchronization)代理：使几个用户能够同时使用一个对象而没有冲突。
    - 智能引用(Smart Reference)代理：当一个对象被引用时，提供一些额外的操作，如将此对象被调用的次数记录下来等。

代理模式

- 模式应用
  - ✓ (1) Java RMI (Remote Method Invocation，远程方法调用)。



代理模式

- 模式应用
  - ✓ (2) EJB、Web Service等分布式技术都是代理模式的应用。在EJB中使用了RMI机制，远程服务器中的企业级Bean在本地有一个桩代理，客户端通过桩来调用远程对象中定义的方法，而无须直接与远程对象交互。在EJB的使用中需要提供一个公共的接口，客户端针对该接口进行编程，无须知道桩以及远程EJB的实现细节。

代理模式

- 模式应用
  - ✓ (3) Spring 框架中的AOP技术也是代理模式的应用，在Spring AOP中应用了动态代理(Dynamic Proxy)技术。

代理模式

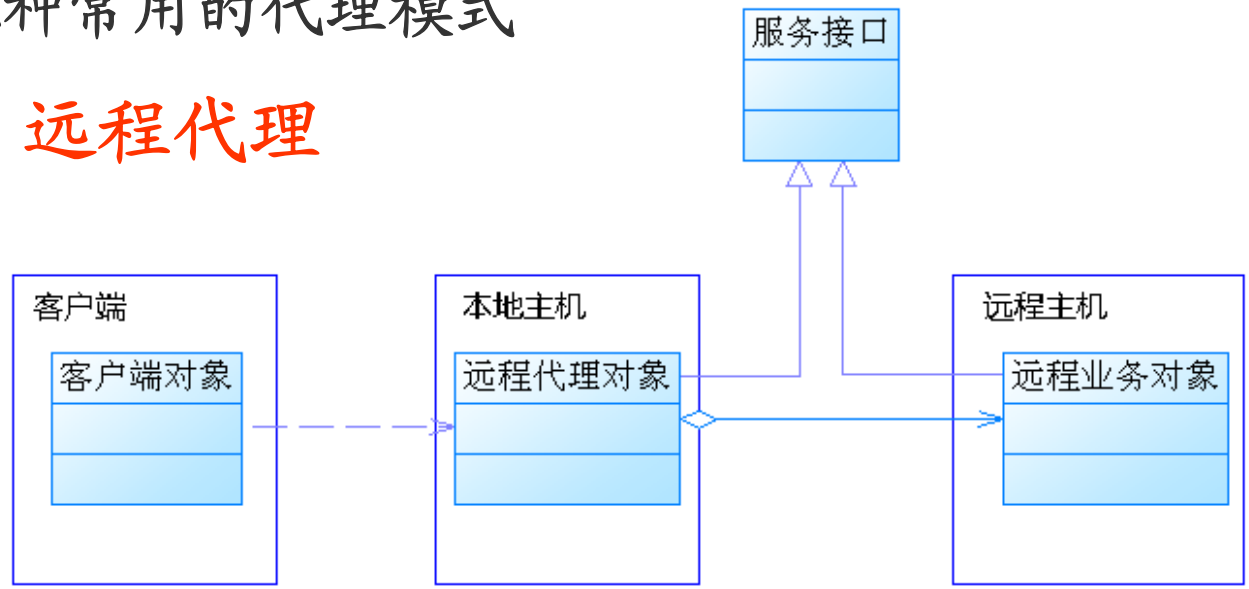
- **模式扩展**
  - ✓ 几种常用的代理模式
    - 图片代理：一个很常见的代理模式的应用实例就是对大图浏览的控制。
    - 用户通过浏览器访问网页时先不加载真实的大图，而是通过代理对象的方法来进行处理，在代理对象的方法中，先使用一个线程向客户端浏览器加载一个小图片，然后在后台使用另一个线程来调用大图片的加载方法将大图片加载到客户端。当需要浏览大图片时，再将大图片在新网页中显示。如果用户在浏览大图时加载工作还没有完成，可以再启动一个线程来显示相应的提示信息。通过代理技术结合多线程编程将真实图片的加载放到后台来操作，不影响前台图片的浏览。

代理模式

- **模式扩展**
  - ✓ 几种常用的代理模式
    - 远程代理：远程代理可以将网络的细节隐藏起来，使得客户端不必考虑网络的存在。客户完全可以认为被代理的远程业务对象是局域的而不是远程的，而远程代理对象承担了大部分的网络通信工作。

代理模式

- **模式扩展**
  - ✓ 几种常用的代理模式
    - 远程代理



代理模式

- **模式扩展**
  - ✓ 几种常用的代理模式
    - 虚拟代理：当一个对象的加载十分耗费资源的时候，虚拟代理的优势就非常明显地体现出来了。虚拟代理模式是一种内存节省技术，那些占用大量内存或处理复杂的对象将推迟到使用它的时候才创建。
    - 在应用程序启动的时候，可以用代理对象代替真实对象初始化，节省了内存的占用，并大大加速了系统的启动时间。

代理模式

- **模式扩展**
  - ✓ 动态代理
    - 动态代理是一种较为高级的代理模式，它的典型应用就是Spring AOP。
    - 在传统的代理模式中，客户端通过Proxy调用RealSubject类的request()方法，同时还在代理类中封装了其他方法(如preRequest()和postRequest())，可以处理一些其他问题。
    - 如果按照这种方法使用代理模式，那么真实主题角色必须是事先已经存在的，并将其作为代理对象的内部成员属性。如果一个真实主题角色必须对应一个代理主题角色，这将导致系统中的类个数急剧增加，因此需要想办法减少系统中类的个数，此外，如何在事先不知道真实主题角色的情况下使用代理主题角色，这都是动态代理需要解决的问题。

代理模式

- **模式扩展**
  - ✓动态代理
    - Java动态代理实现相关类位于java.lang.reflect包，主要涉及两个类：
      - InvocationHandler接口。它是代理实例的调用处理程序实现的接口，该接口中定义了如下方法：
        public Object invoke (Object proxy, Method method, Object[] args) throws Throwable;invoke()方法中第一个参数proxy表示代理类，第二个参数method表示需要代理的方法，第三个参数args表示代理方法的参数数组。

代理模式

- **模式扩展**
  - ✓动态代理
    - – **Proxy类**。该类即为**动态代理类**，该类最常用的方法为：public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces, InvocationHandler h) throws IllegalArgumentException。newProxyInstance()方法用于根据传入的接口类型interfaces返回一个动态创建的代理类的实例，方法中第一个参数loader表示代理类的类加载器，第二个参数interfaces表示代理类实现的接口列表（与真实主题类的接口列表一致），第三个参数h表示所指派的调用处理程序类。

代理模式

- **模式扩展**
  - ✓动态代理
    - 参考代码：**Chapter 16 Proxy\DynamicProxy**
    - 下载地址：**http://download.csdn.net/user/cflynn**

演示.....

本章小结

- 在代理模式中，要求给某一个对象提供一个代理，并由代理对象控制对原对象的引用。代理模式的英文叫做Proxy或Surrogate，它是一种对象结构型模式。
- 代理模式包含三个角色：抽象主题角色声明了真实主题和代理主题的共同接口；代理主题角色内部包含对真实主题的引用，从而可以在任何时候操作真实主题对象；真实主题角色定义了代理角色所代表的真实对象，在真实主题角色中实现了真实的业务操作，客户端可以通过代理主题角色间接调用真实主题角色中定义的方法。

本章小结

- 代理模式的优点在于能够协调调用者和被调用者，在一定程度上降低了系统的耦合度；其缺点在于由于在客户端和真实主题之间增加了代理对象，因此有些类型的代理模式可能会造成请求的处理速度变慢，并且实现代理模式需要额外的工作，有些代理模式的实现非常复杂。
- 远程代理为一个位于不同的地址空间的对象提供一个本地的代表对象，它使得客户端可以访问在远程机器上的对象，远程机器可能具有更好的计算性能与处理速度，可以快速响应并处理客户端请求。

本章小结

- 如果需要创建一个资源消耗较大的对象，先创建一个消耗相对较小的对象来表示，真实对象只在需要时才会被真正创建，这个小对象称为虚拟代理。虚拟代理通过使用一个小对象来代表一个大对象，可以减少系统资源的消耗，对系统进行优化并提高运行速度。
- 保护代理可以控制对一个对象的访问，可以给不同的用户提供不同级别的使用权限。

END

Thanks!