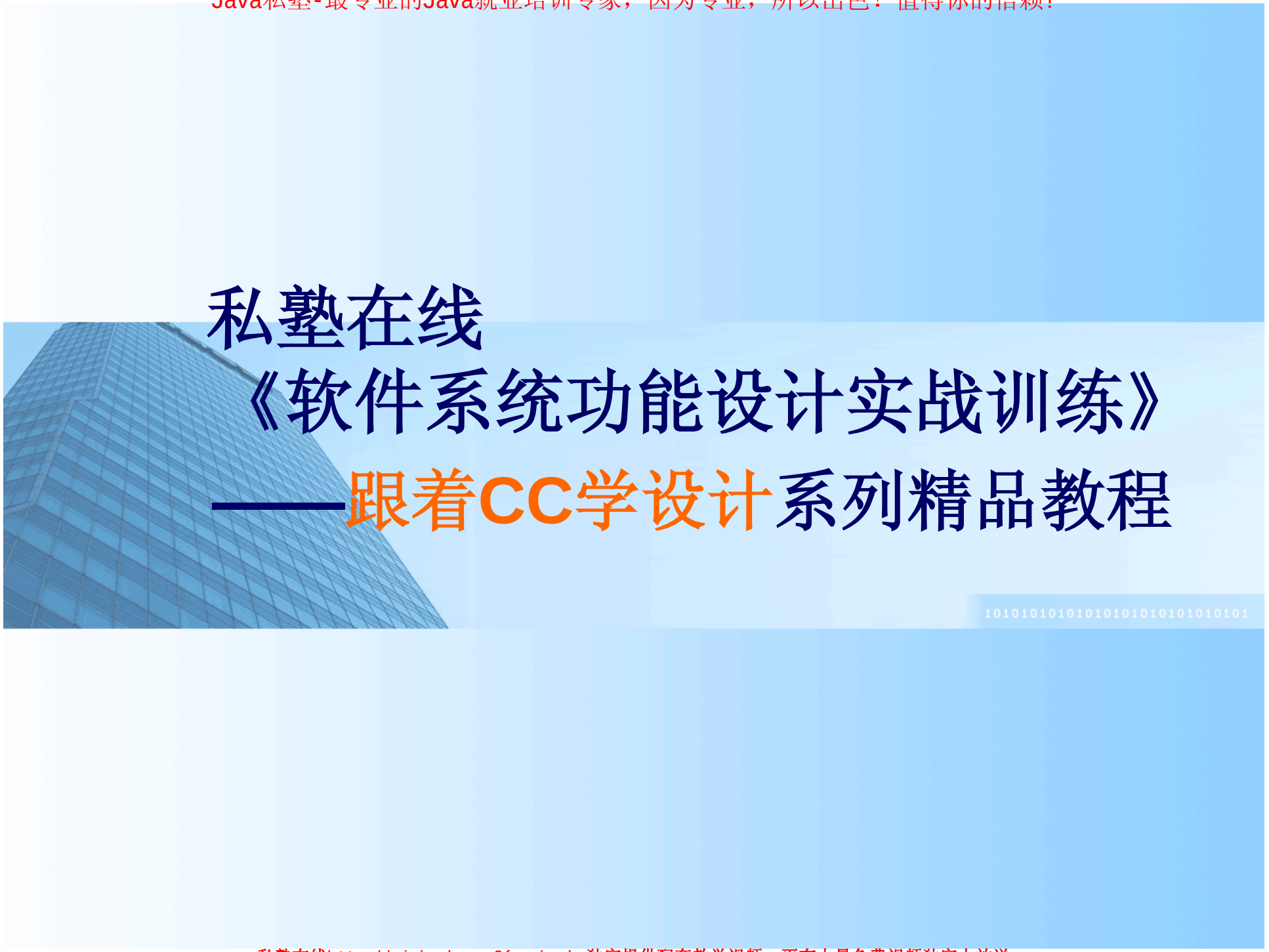


私塾在线
《软件系统功能设计实战训练》
——跟着CC学设计系列精品教程

本周设计作业

- 本周设计作业的项目背景：

  编号/文号生成器——来自中国人寿企业年金系统

- 学习目标：

  （1）在实战中练习设计的理念和方法

  （2）学习如何设计API，如何应对业务的变更，体会应用功能需求的多变性，体会优秀设计的重要性和对变化的支持

  （3）学习缓存的应用，高并发的处理策略等

  （4）综合应用工厂方法/简单工厂模式、桥接模式、策略模式、适配器模式、值对象模式、单例模式、观察者模式等，包含着这些模式但不限于这些模式该设计方式可应用于多种有类似功能的系统，比如：固定资产编号、文件编号、所有需要有格式号的系统……

本周设计作业

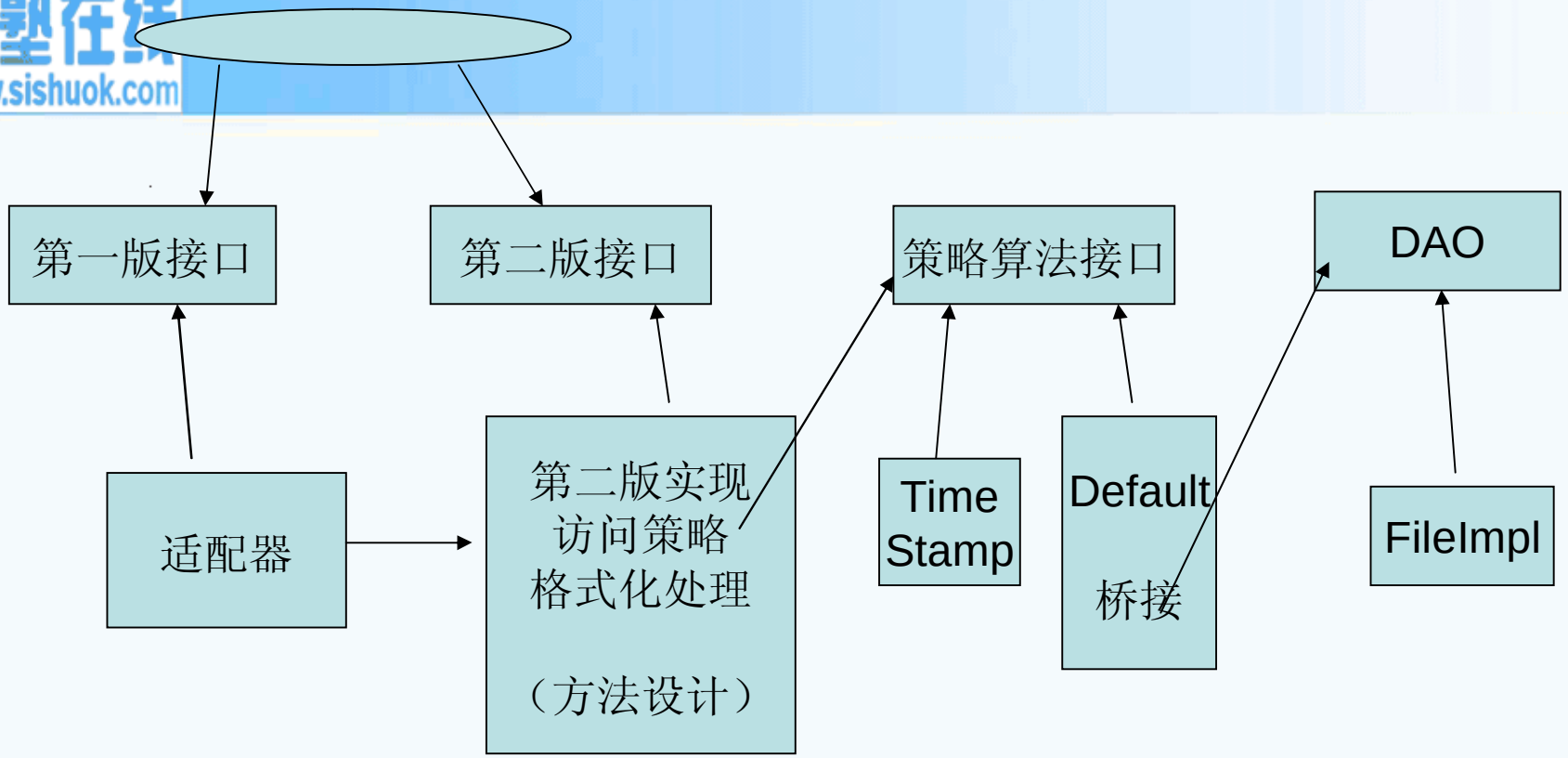
- 基本功能

  1：首先实现一版：能生成不重复的流水号，把流水号数据存放在文件中，既支持统一使用一个流水号，也支持各个业务单独使用自己的流水号，引入简单工厂，桥接和值对象模式

  2：在保证兼容第一版的情况下，实现新的一版，要求能够生成不同格式要求、不同生成算法的编号，引入策略模式和适配器模式

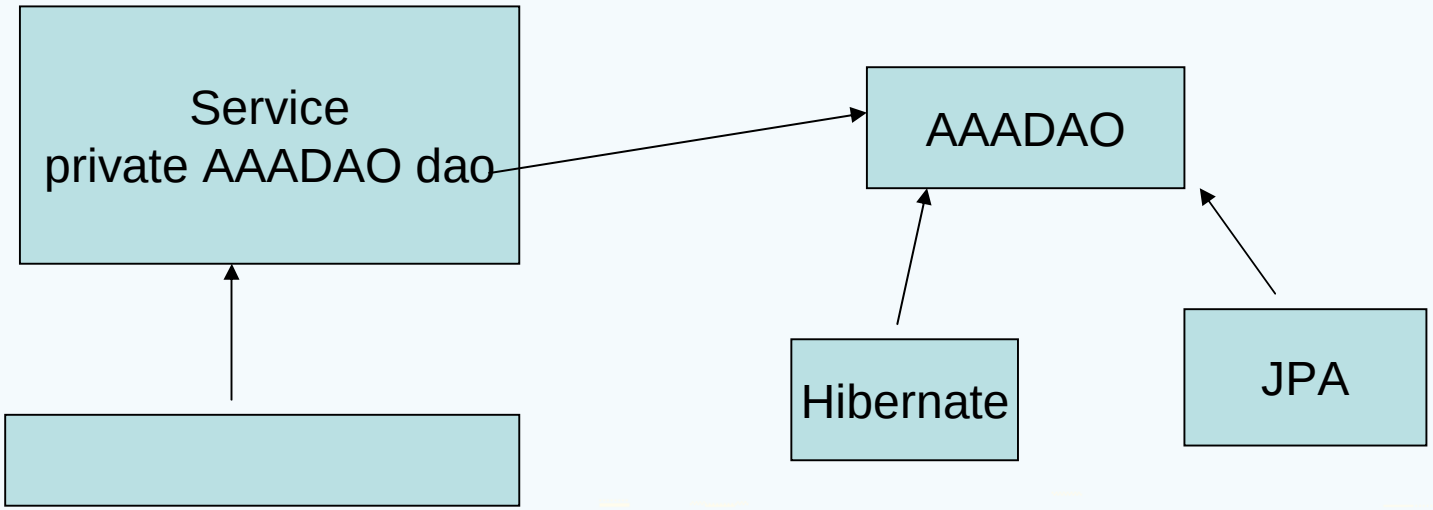
- 作业要求

  1：在老师给定的概要代码基础上，实现上述基本要求的功能设计
  2：重点在接口和功能实现的设计上，无须关注具体实现
  3：对于每个api，可以适当写点样例代码，能够调用运行更佳
  4：考虑合理的结构，职责的划分，以及设计模式的合理使用



- 无意义（指的是跟业务无关）的自然键
- 整个系统用一个流水号，比如：文件管理 -1，档案管理-2，文件管理-3，文件管理-4，档案管理-5
- 各个业务单独的流水号，比如：文件管理 -1，档案管理-1，文件管理-2，文件管理-3，档案管理-2
- 值对象（vo）value object ==== 简化使用，当成数据的封装对象
- 1：私有化所有的属性
- 2：getter/setter
- 3：要有一个public为空参的构造方法
- 4：建议覆盖实现equal、hashCode、toString

- class AAAService{
- @Autowired
- private AAADAO dao = null;
- }

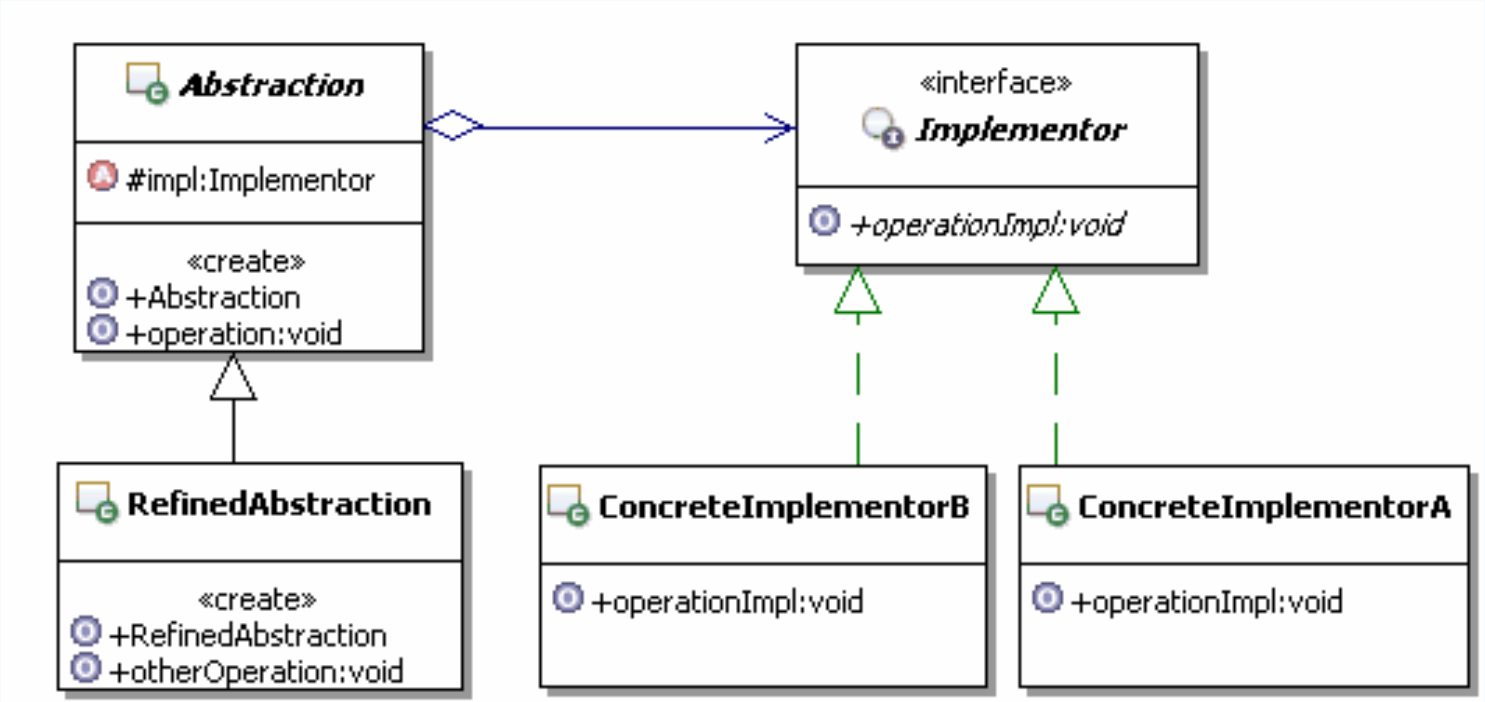


初识桥接模式

- 定义

  将抽象部分与它的实现部分分离，使它们都可以独立地变化。

- 结构和说明



桥接模式的知识要点

- 桥接模式的知识要点

1：桥接是在被分离了的抽象部分和实现部分之间来搭桥，桥接在程序上就体现成了在抽象部分拥有实现部分的接口对象，维护桥接就是维护这个关系

2：桥接模式的意图：使得抽象和实现可以独立变化，都可以分别扩充。

3：桥接模式可以实现运行时动态组合具体的真实实现，从而达到动态变换功能的目的

4：桥接模式适应于两个纬度的变化，而继承适用于一个纬度的变化

5：使用桥接模式的时候，要注意谁来创建Implementor的对象，并把它设置到抽象部分的对象里面去。

6：从某个角度来讲，桥接模式就是对“面向抽象编程”这个设计原则的扩展。

7：桥接模式是可以连续组合使用的，一个桥接模式的实现部分，可以作为下一个桥接模式的抽象部分

思考桥接模式

- 桥接模式的本质是：**分离抽象和实现**
- 何时选用桥接模式

1：如果你不希望在抽象和实现部分采用固定的绑定关系，可以采用桥接模式，来把抽象和实现部分分开，然后在程序运行期间来动态的设置抽象部分需要用到的具体的实现，还可以动态切换具体的实现

2：如果出现抽象部分和实现部分都应该可以扩展的情况，可以采用桥接模式，让抽象部分和实现部分可以独立的变化，从而可以灵活的进行单独扩展，而不是搅在一起，扩展一边会影响到另一边 。

3：如果希望实现部分的修改，不会对客户产生影响，可以采用桥接模式，客户是面向抽象的接口在运行，实现部分的修改，可以独立于抽象部分，也就不会对客户产生影响了，也可以说对客户是透明的

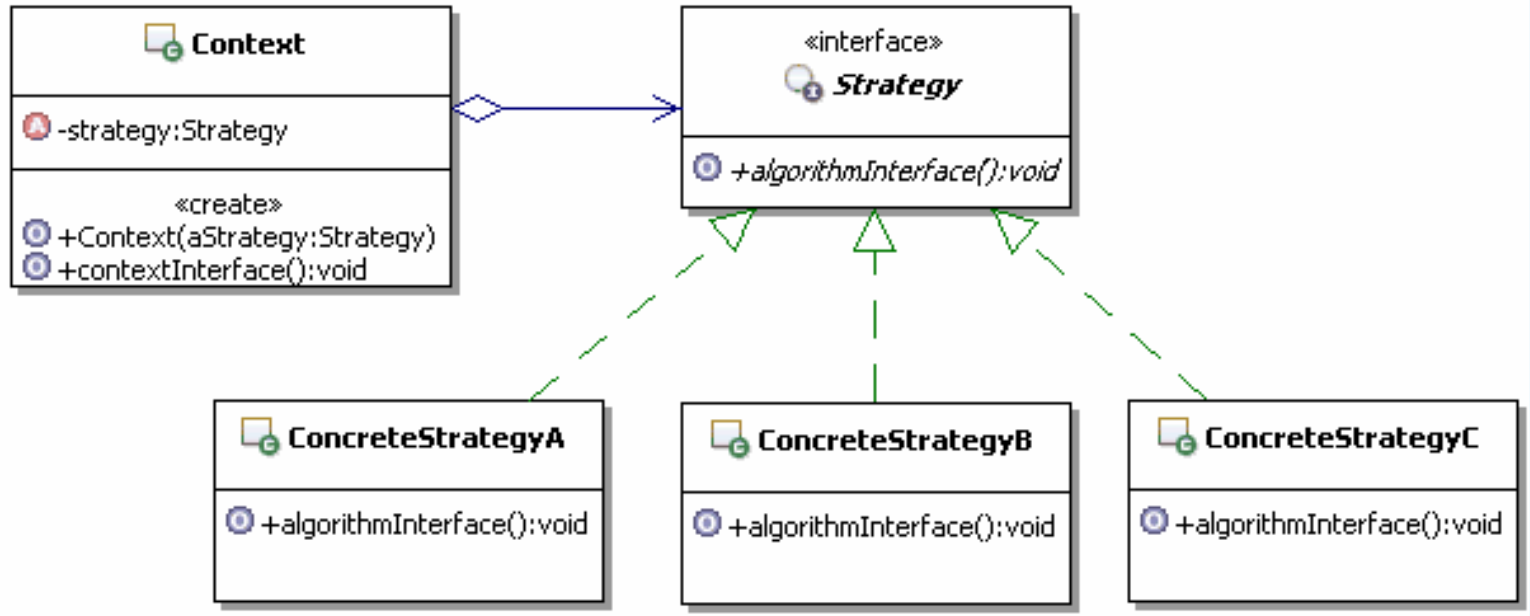
4：如果采用继承的实现方案，会导致产生很多子类，对于这种情况，可以考虑采用桥接模式，分析功能变化的原因，看看是否能分离成不同的纬度，然后通过桥接模式来分离它们，从而减少子类的数目

初识策略模式

- 定义

  定义一系列的算法，把它们一个个封装起来，并且使它们可相互替换。本模式使得算法可独立于使用它的客户而变化 。

- 结构和说明



策略模式的知识要点

n 策略模式的知识要点

1：策略模式的功能是把具体的算法实现，从具体的业务处理里面独立出来，把它们实现成为单独的算法类，从而形成一系列的算法，并让这些算法可以相互替换。

2：策略模式的重心不是如何来实现算法，而是如何组织、调用这些算法，从而让程序结构更灵活、具有更好的维护性和扩展性。

3：策略模式一个很大的特点就是各个策略算法的平等性。所有的策略算法在实现上也是相互独立的，相互之间是没有依赖的。所以可以这样描述这一系列策略算法：策略算法是相同行为的不同实现。

4：使用策略模式的时候，要注意谁来选择策略，可以是客户端，也可以在上下文里面。而跟策略模式类似的状态模式，一般是不会让客户端来选择状态的，状态是内部行为，这是一个很重要的区别。

5：策略模式在每一个时刻只能使用一个具体的策略实现对象，虽然可以动态的在不同的策略实现中切换，但是同时只能使用一个。

6：上下文在策略模式里面有特殊的地位，上下文使用具体的策略实现对象，反过来，策略实现对象也可以从上下文获取所需要的数据。甚至在某些情况下，策略实现对象还可以回调上下文的方法来实现一定的功能，这种使用场景下，上下文变相充当了多个策略算法实现的公共接口，在上下文定义的方法可以当做是所有或者是部分策略算法使用的公共功能。

思考策略模式

- 策略模式的本质

  策略模式的本质是：**分离算法，选择实现**

- 何时选用策略模式

1：出现有许多相关的类，仅仅是行为有差别的情况，可以使用策略模式来使用多个行为中的一个来配置一个类的方法，实现算法动态切换

2：出现同一个算法，有很多不同的实现的情况，可以使用策略模式来把这些“不同的实现”实现成为一个算法的类层次

3：需要封装算法中，与算法相关的数据的情况，可以使用策略模式来避免暴露这些跟算法相关的数据结构

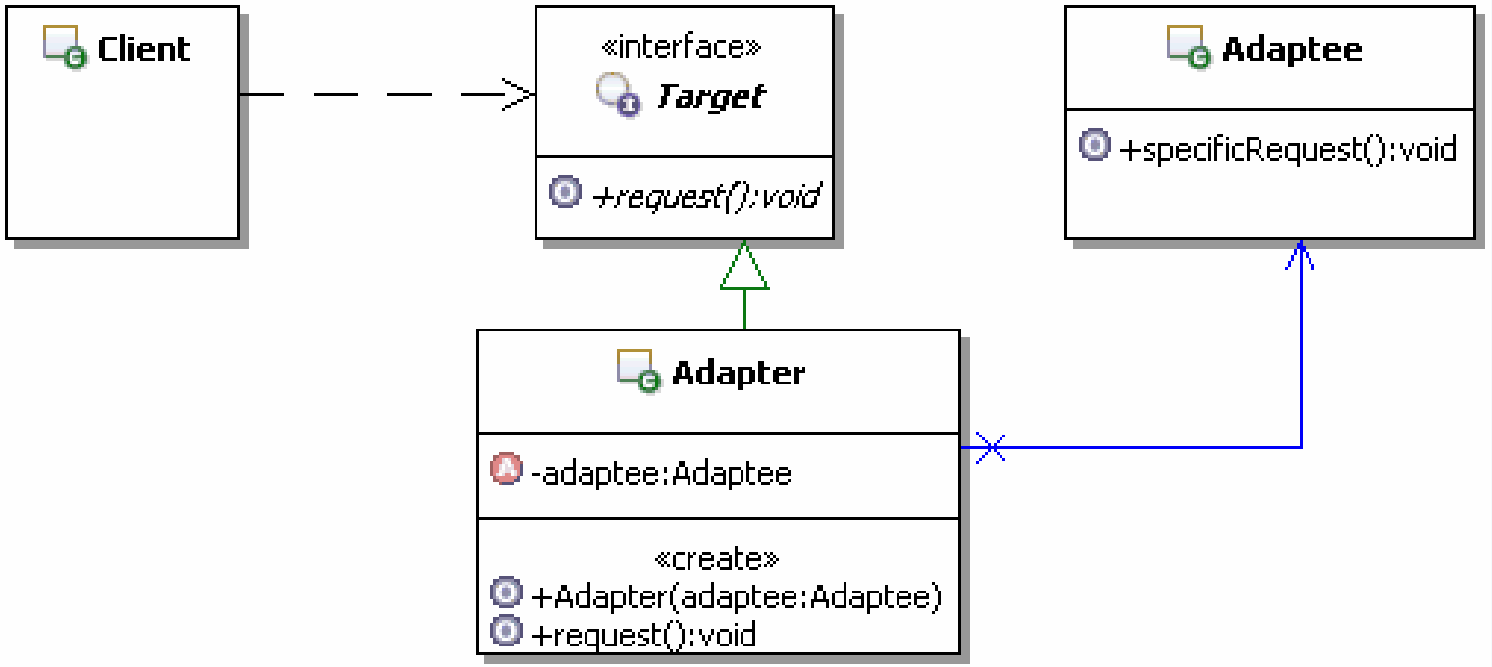
4：出现抽象一个定义了很多行为的类，并且是通过多个if-else语句来选择这些行为的情况，可以使用策略模式来代替这些条件语句

初识适配器模式

- 定义

  将一个类的接口转换成客户希望的另外一个接口。适配器模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

- 结构和说明



适配器模式的知识要点

- 适配器模式的知识要点

1：适配器模式的主要功能是进行转换匹配，目的是复用已有的功能，而不是来实现新的接口。

2：适配器里面可以实现功能，称这种适配器为智能适配器。

3：适配器模式中被适配的接口Adaptee和适配成为的接口Target是没有关联的，也就是说，Adaptee和Target中的方法既可以相同，也可以不同，极端情况下两个接口里面的方法可能是完全不同的，当然极端情况下也可以完全相同。

4：适配器的实现方式其实是依靠对象组合的方式，当客户端调用Target的时候，适配器会把相应的功能，委托给被适配的对象去完成。

5：缺省适配的意思是：为一个接口提供缺省实现。有了它，就不用直接去实现接口，而是采用继承这个缺省适配对象，从而让子类可以有选择的去覆盖实现需要的方法，对于不需要的方法，就使用缺省适配的方法就可以了。

6：适配器也可以实现双向的适配，前面我们讲的都是把Adaptee适配成为Target，其实也可以把Target适配成为Adaptee，也就是说这个适配器可以同时当作Target和Adaptee来使用。

思考适配器模式

- 适配器模式的本质

  适配器模式的本质是：**转换匹配，复用功能**

- 何时选用适配器模式

1：如果你想要使用一个已经存在的类，但是它的接口不符合你的需求，这种情况可以使用适配器模式，来把已有的实现转换成你需要的接口

2：如果你想创建一个可以复用的类，这个类可能和一些不兼容的类一起工作，这种情况可以使用适配器模式，到时候需要什么就适配什么

3：如果你想使用一些已经存在的子类，但是不可能对每一个子类都进行适配，这种情况可以选用对象适配器，直接适配这些子类的父类就可以了