中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC

Design Patterns

中国科学技术大学软件学院
孟宁

- 设计模式和面向对象的程序设计曾经承诺

让软件设计开发者的工作更加轻松！

不是让学习更加轻松！

相反让学习更困难！

Contents

- 面向对象OO
- Design Patterns

面向对象OO

- 对象的真正威力不是继承而是“行为封装”。
  - “面向对象”范式仅仅告诉开发者在需求语句中寻找“名词”，并将这些名词构造成程序中的对象。
  - 在这种范式中“封装”仅仅被定义为“数据隐藏”，“对象”也只是被定义为“包含数据及访问这些数据的东西”。
- 继承-代码重用；封装-模块化

功能分解

- 功能分解是处理复杂问题的一种自然的方法
- 需求总是在发生变化。
- 功能分解不能帮助我们为未来可能的变化做准备，也不能帮助我们的代码优雅地演化。
  - 你想在代码中做一些改变，但又不敢这么做，因为你知道对一个地方代码的修改可能在另一个地方造成破坏。

包容变化

- 与其抱怨总是变化的需求，我们不如改进开发过程，这样我们可以更有效地应付需求的变化。
- 用模块化来包容变化
  - 高内聚低耦合
  - 抽象类
  - 可见性
  - 封装
  - 多态

Design Patterns

- A design pattern codifies design decisions and best practices for solving a particular design problem according to design principles
- Design patterns are not the same as software libraries; they are not packaged solutions that can be used as is. Rather, they are templates for a solution that must be modified and adapted for each particular use
- 模式是在某一情景下的问题解决方案。

设计模式的组成

- 模式的名称
- 模式的目的，它要解决的问题
- 我们如何实现它
- 为了实现它我们必须考虑的限制和约束

Facade（外观）模式

- 模式的名称
  - Facade（外观）模式
- 模式的目的，它要解决的问题
  - 意图：希望简化现有系统的使用方法，需要定义一个一致的高层接口
  - 问题：只需使用一个复杂系统的一个子集。或者，需要用一种特殊的方式与系统交互。
- 我们如何实现它
  - 向使用者展现使用现有系统的一个新的接口
  - 定义一个或一组新的类来提供所需的接口
  - 让新的类来使用现有系统
- 为了实现它我们必须考虑的限制和约束

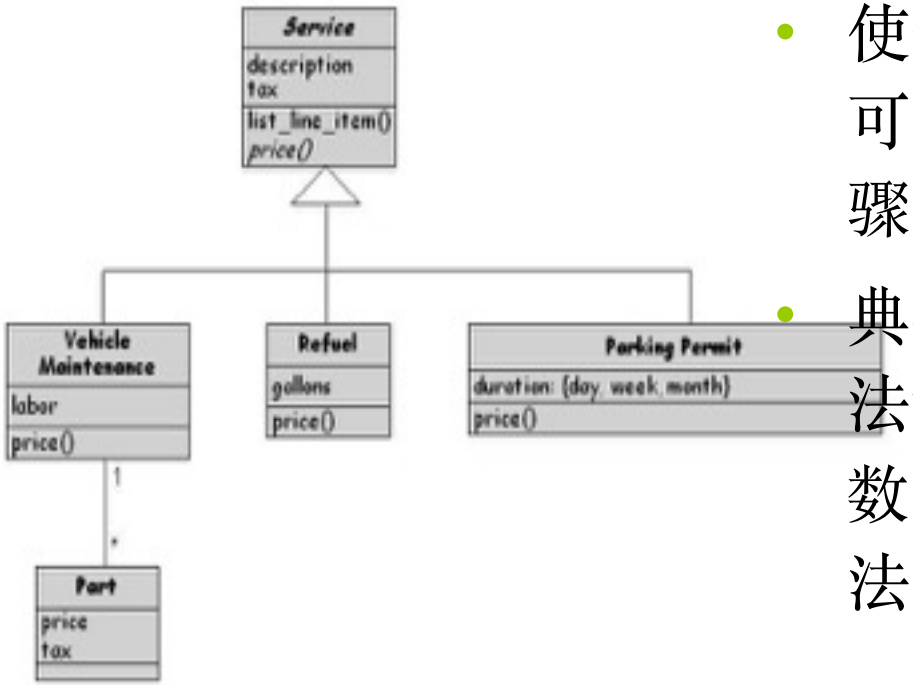
Adapter（适配器）模式

- 将一个类的接口转换成客户所希望的另外一个接口。
- 将一个无法控制的现有对象与一个特定接口相匹配。
- 用我们需要的接口对无法修改的类进行包装

Template Method Pattern

- The Template Method pattern aims to reduce the amount of duplicate code among subclasses of the same parent class
  - It is particularly useful when multiple subclasses have similar but not identical implementations of the same method
  - This pattern addresses this problem by localizing the duplicate code structure in an abstract class from which the subclasses inherit
- The abstract class defines a template method that implements the common steps of an operation, and declares abstract primitive operations that represent the variation points

Template Method Pattern



- 定义一个操作中的骨架，而将一些步骤延迟到子类中
- 使子类可以不改变算法的结构即可重新定义该算法的某些特定步骤
- 典型实现：抽象类中定义模板方法和虚拟方法（C++的纯虚函数？），模板方法中调用虚拟方法，在子类中重载虚拟方法。

Strategy Pattern

- The Strategy pattern allows algorithms to be selected at runtime
- It is useful when various algorithms are available to an application but the chose of best algorithm is not known
- 将算法的选择和算法的实现分离，让客户可以基于场景做出选择。
- 典型实现：让使用算法的类（Context/客户）包含一个抽象类（策略Strategy类），抽象类有一个抽象方法指定如何调用算法，每个子类根据需要实现算法。
- 策略模式与模板方法、工厂方法有不同吗？需求场景上有什么不同？技术实现上有什么不同？
- 策略模式范例：商场促销商品

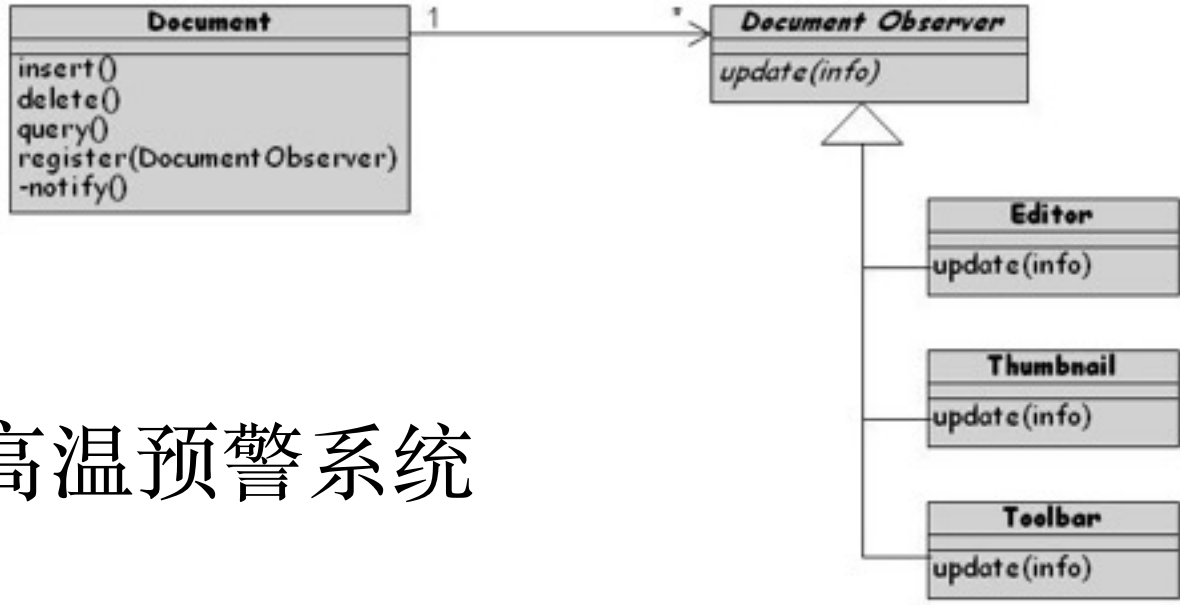
Bridge（桥接）模式

- 将抽象部分和实现部分分离，使它们都可以独立地变化
- 桥接模式范例：开关与电灯的桥接模式实现

Observer Pattern

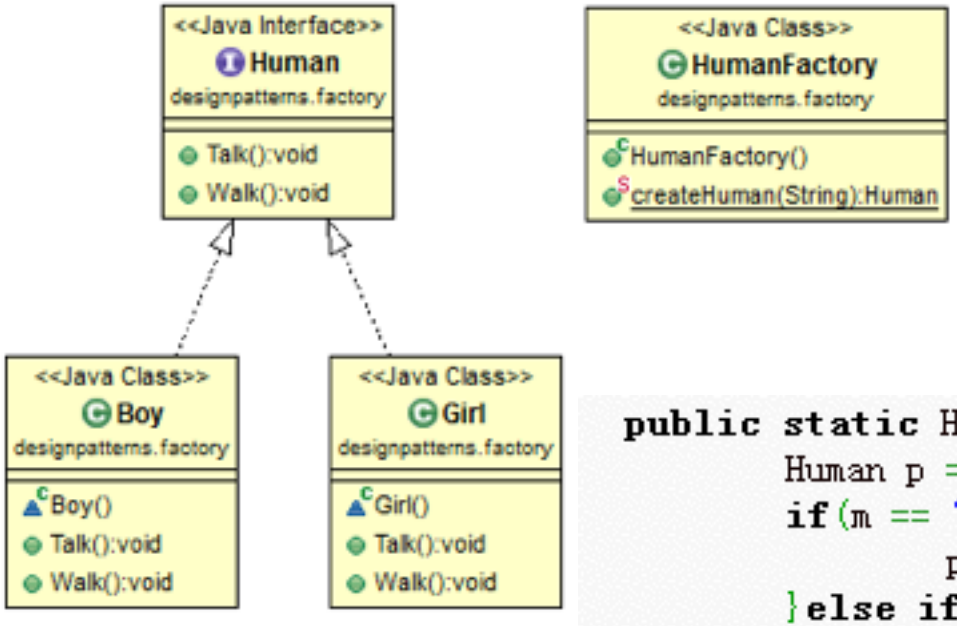
The Observer pattern is an application of the publish–subscribe architecture style

Useful when software needs to notify multiple objects of key events



范例：高温预警系统

简单工厂模式



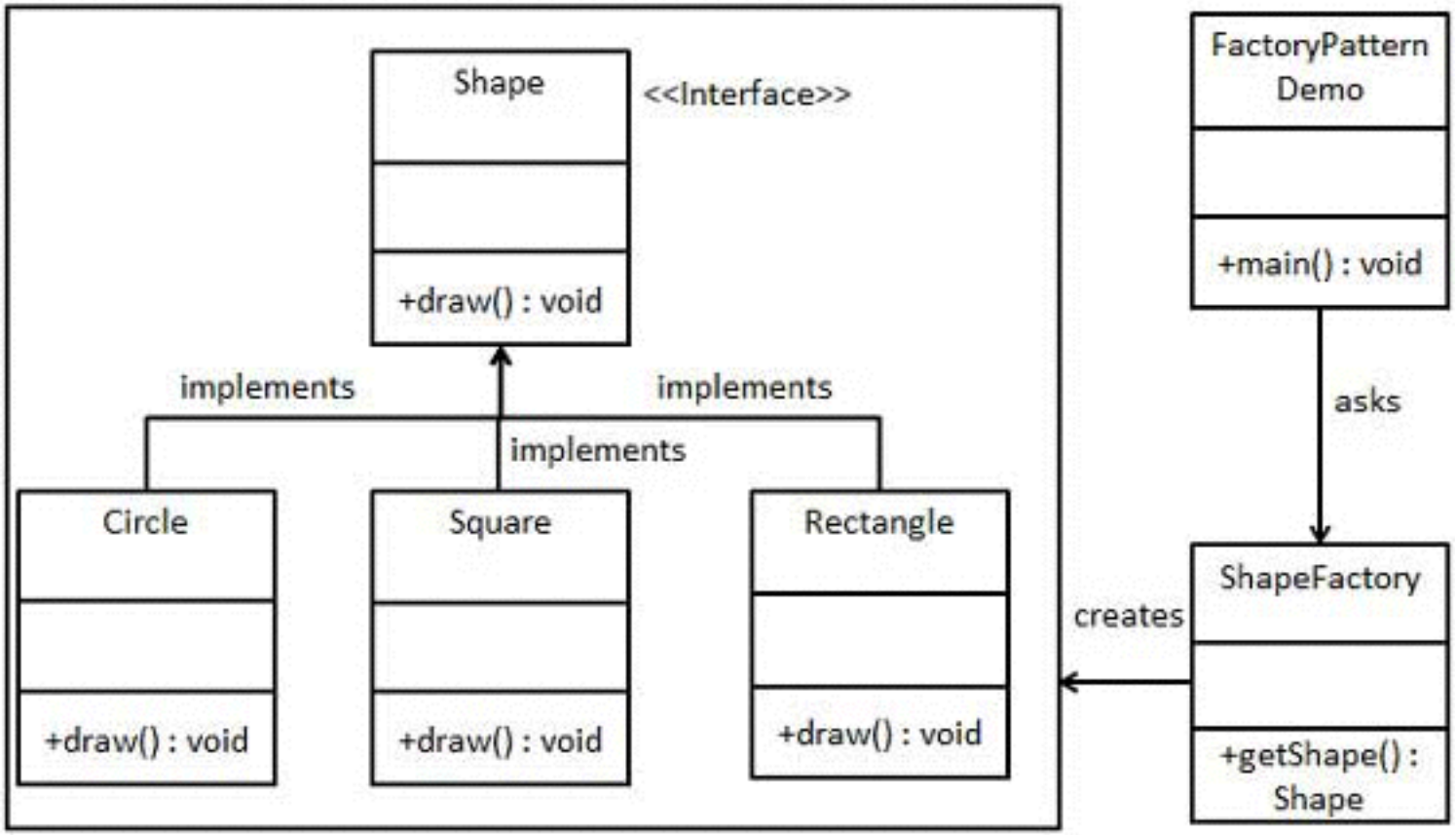

```
public static Human createHuman(String m){
        Human p = null;
        if(m == "boy"){
                p = new Boy();
        }else if(m == "girl"){
                p = new Girl();
        }

        return p;
}
```

Factory Method Pattern

- The Factory Method pattern is used to encapsulate the code that creates objects
- The factory Method pattern is similar to the Template method pattern
- The similar but not identical methods are the constructor methods that instantiate objects

- 定义一个用于创建对象的接口（或构造函数），让子类决定实例化哪一个类。
- 典型实现：在抽象类中定义一个抽象方法（C++的纯虚函数？），这个方法延迟到实例化一个子类的对象的时候，抽象类不知道需要哪个特定对象。
- 模板方法和工厂方法都是将业务划分抽象层和具体层

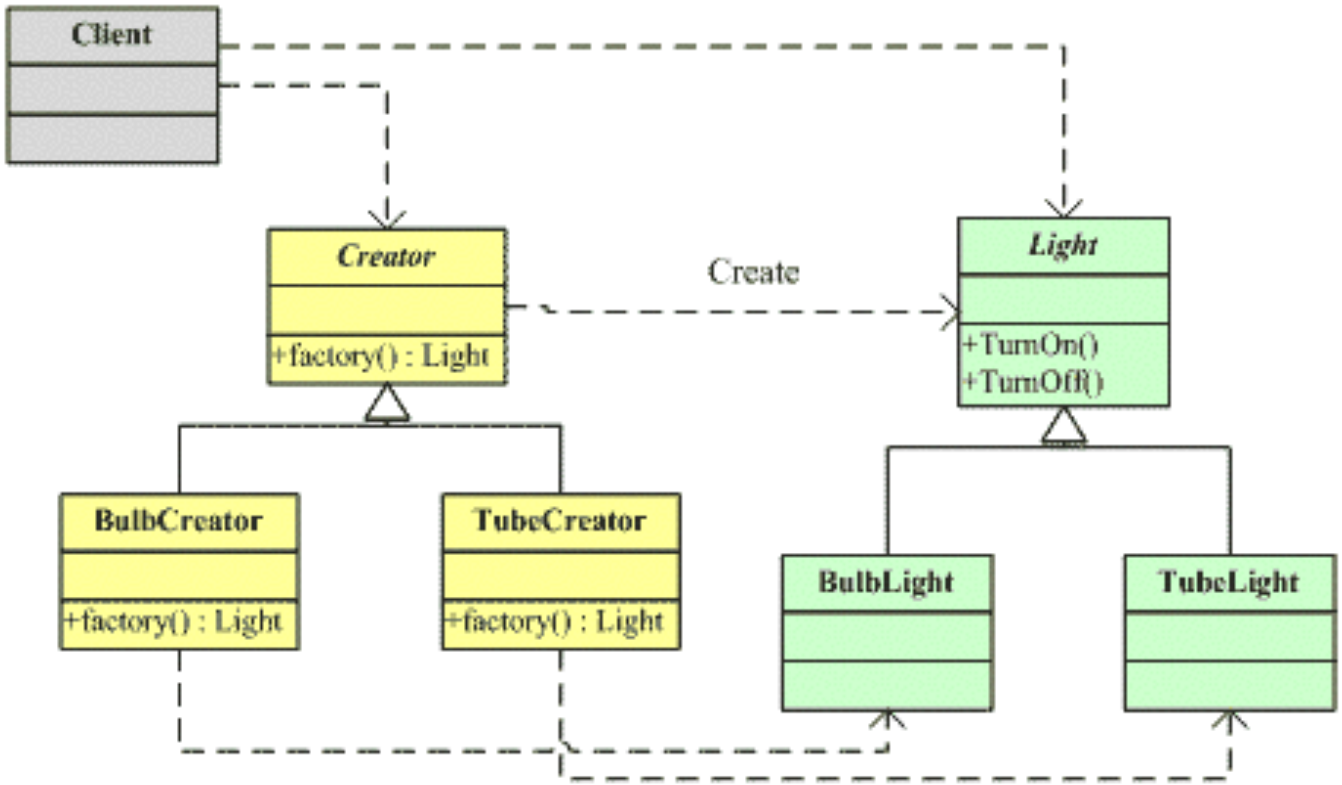
Factory Method Pattern



Abstract Factory Pattern

- 提供一个创建一系列相关或相互依赖的对象的接口，而不需指定它们具体的类。
- 你需要为特定的客户提供特定系列的对象。一系列相关的对象需要被实例化。
- 将使用哪些对象的规则与如何使用这些对象的逻辑相隔离。
- 典型实现：定义一个抽象类来指定哪些对象将被创建，然后为每个系列实现一个具体类

Abstract Factory Pattern



Abstract Factory Pattern

- Abstract Factory Pattern的3个关键策略
  - 发现并封装变化点：使用哪个对象（或系列对象）的选择是变化的，所以将它封装在抽象Factory中。
  - 优先使用对象组合，而不是类继承
  - 针对接口设计，而不是针对实现设计
- 抽象工厂模式范例：Warcraft：Orcs & Humans

练习题

- 你有一个Smartphone接口类和它派生类iPhone、 AndroidPhone、WindowsPhone等，以及以品牌名称命名的派生类如SonyPhone和手机版本iPhone 5c等。你该如何设计这一类的系统？
  - 这是一种设计模式的练习，你充分利用面向对象的设计技巧，保证它有足够的灵活性来支持新产品，并且在现有模型改变时能够保证足够的软件架构稳定性。

中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC

谢谢大家！

References

设计模式精解，**Alan Shalloway & James R.Trott**著，熊节 译 清华大学出版社

软件工程 - 理论与实践（第四版 影印版） **Software Engineering: Theory and Practice (Fourth Edition),Shari Lawrence Pfleeger,Joanne M. Atlee ,**高等教育出版社

软件工程 - 理论与实践（第四版） **Software Engineering: Theory and Practice (Fourth Edition),Shari Lawrence Pfleeger,Joanne M. Atlee,**杨卫东译,人民邮电出版社

软件工程—实践者的研究方法（**Software Engineering-A Practitioner's Approach**）; (美) **Roger S. Pressman** 著； 机械工业出版社**ISBN**：**7-111-07282-0**

http://code.google.com/p/advancedsoftwareengineering/