GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

C++设计模式

Template Method

李建忠

GOF-23 模式分类

- 从目的来看：
  - 创建型（Creational）模式：将对象的部分创建工作延迟到子类或者其他对象，从而应对需求变化为对象创建时具体类型实现引来的冲击。
  - 结构型（Structural）模式：通过类继承或者对象组合获得更灵活的结构，从而应对需求变化为对象的结构带来的冲击。
  - 行为型（Behavioral）模式：通过类继承或者对象组合来划分类与对象间的职责，从而应对需求变化为多个交互的对象带来的冲击。
- 从范围来看：
  - 类模式处理类与子类的静态关系。
  - 对象模式处理对象间的动态关系。

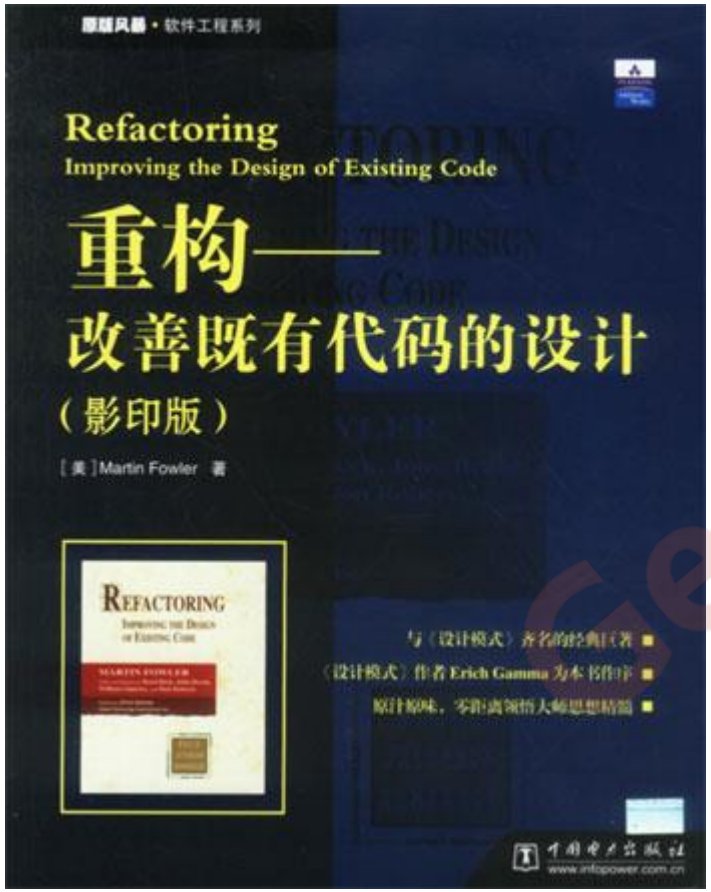
从封装变化角度对模式分类

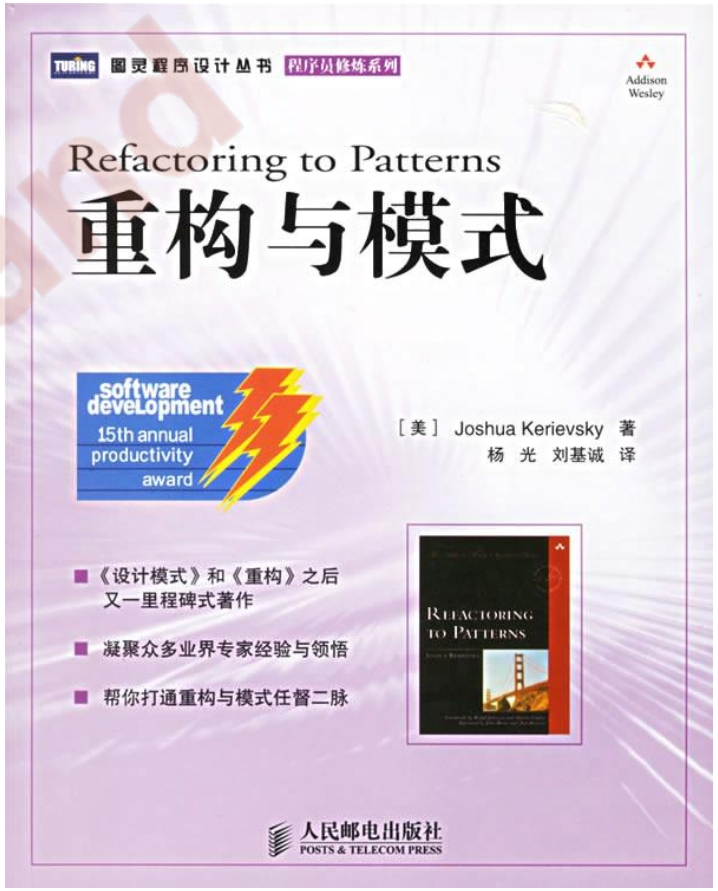
- 组件协作：
  - Template Method
  - Observer / Event
  - Strategy
- 单一职责：
  - Decorator
  - Bridge
- 对象创建:
  - Factory Method
  - Abstract Factory
  - Prototype
  - Builder
- 对象性能：
  - Singleton
  - Flyweight
- 接口隔离:
  - Façade
  - Proxy
  - Mediator
  - Adapter
- 状态变化：
  - Memento
  - State
- 数据结构：
  - Composite
  - Iterator
  - Chain of Resposibility
- 行为变化：
  - Command
  - Visitor
- 领域问题：
  - Interpreter

重构获得模式 Refactoring to Patterns

- 面向对象设计模式是“好的面向对象设计”，所谓“好的面向对象设计”指是那些可以满足“应对变化，提高复用”的设计。
- 现代软件设计的特征是“需求的频繁变化”。设计模式的要点是“寻找变化点，然后在变化点处应用设计模式，从而来更好地应对需求的变化”.“什么时候、什么地点应用设计模式”比“理解设计模式结构本身”更为重要。
- 设计模式的应用不宜先入为主，一上来就使用设计模式是对设计模式的最大误用。没有一步到位的设计模式。敏捷软件开发实践提倡的“Refactoring to Patterns”是目前普遍公认的最好的使用设计模式的方法。

推荐图书





重构关键技法

- 静态 → 动态
- 早绑定 → 晚绑定
- 继承 → 组合
- 编译时依赖 → 运行时依赖
- 紧耦合 → 松耦合

“组件协作”模式:

- 现代软件专业分工之后的第一个结果是“框架与应用程序的划分”，“组件协作”模式通过晚期绑定，来实现框架与应用程序之间的松耦合，是二者之间协作时常用的模式。

- 典型模式
  - Template Method
  - Observer / Event
  - Strategy

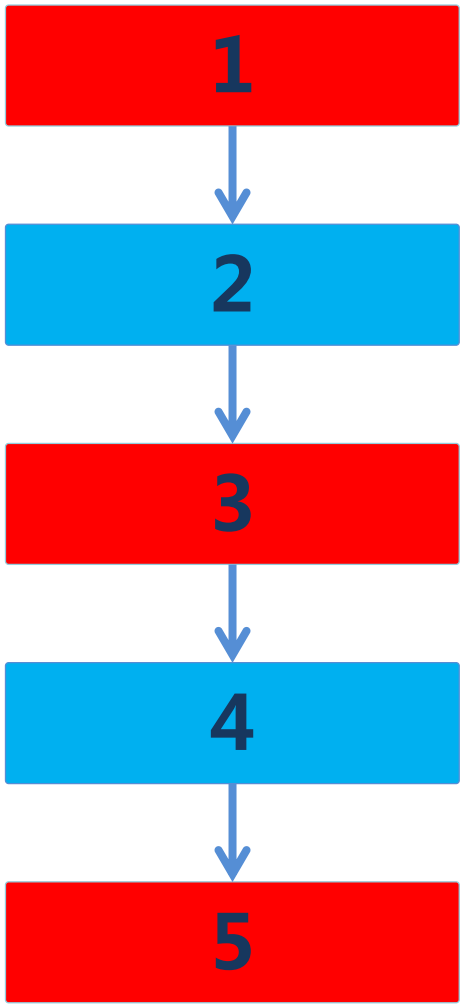
Template Method 模式

动机（Motivation）

- 在软件构建过程中，对于某一项任务，它常常有稳定的整体操作结构，但各个子步骤却有很多改变的需求，或者由于固有的原因（比如框架与应用之间的关系）而无法和任务的整体结构同时实现。

- 如何在确定稳定操作结构的前提下，来灵活应对各个子步骤的变化或者晚期实现需求？

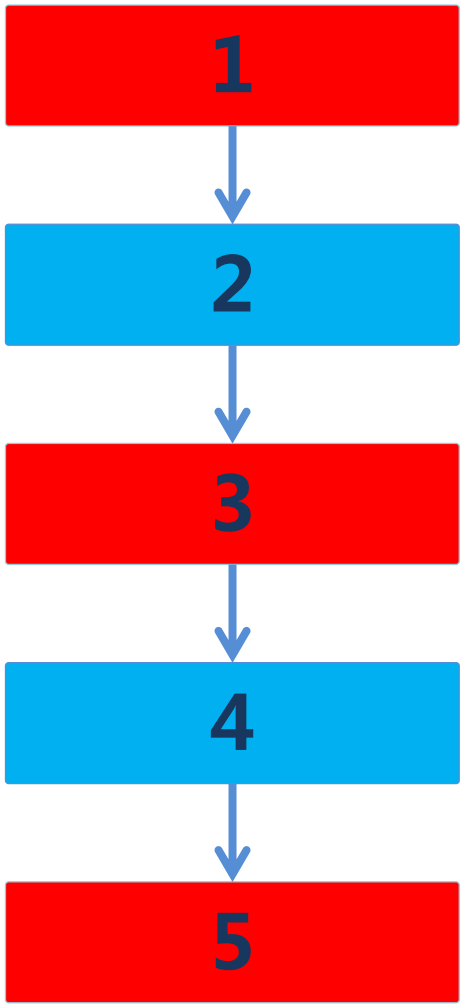
结构化软件设计流程



Library开发人员：
(1) 开发1、3、5 三个步骤

Application开发人员
(1) 开发2、4两个步骤
(2) 程序主流程

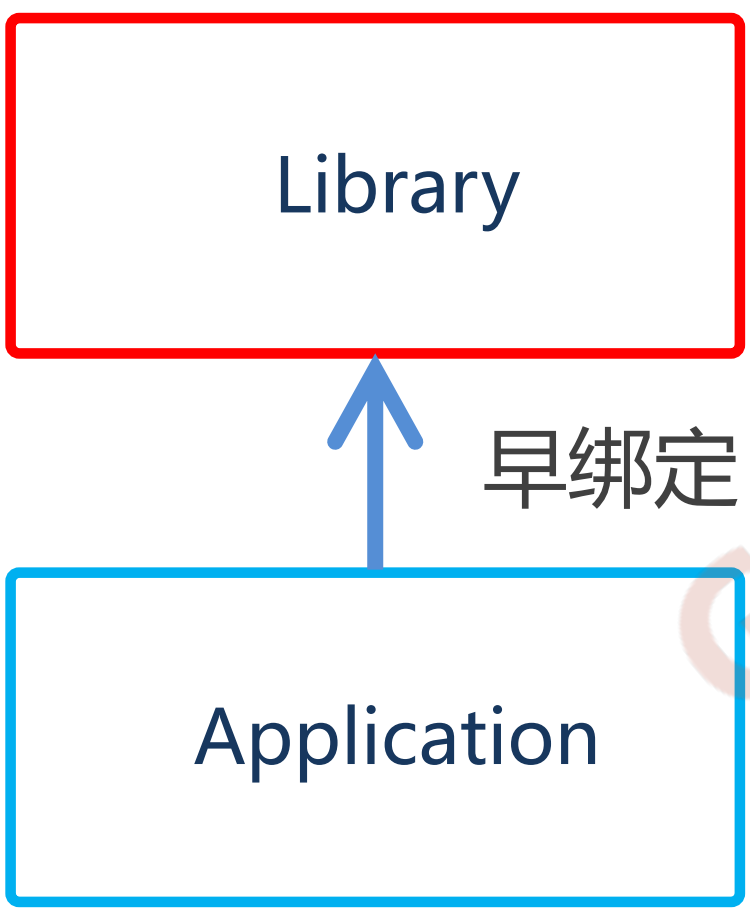
面向对象软件设计流程

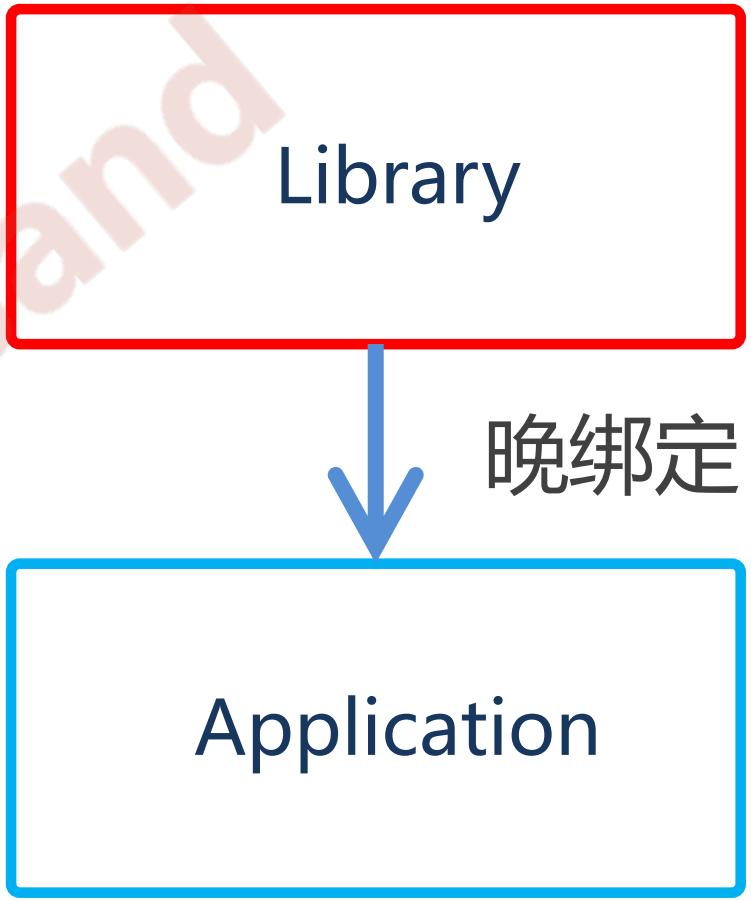


Library开发人员：
(1) 开发1、3、5 三个步骤
(2) 程序主流程

Application开发人员
(1) 开发2、4两个步骤

早绑定与晚绑定



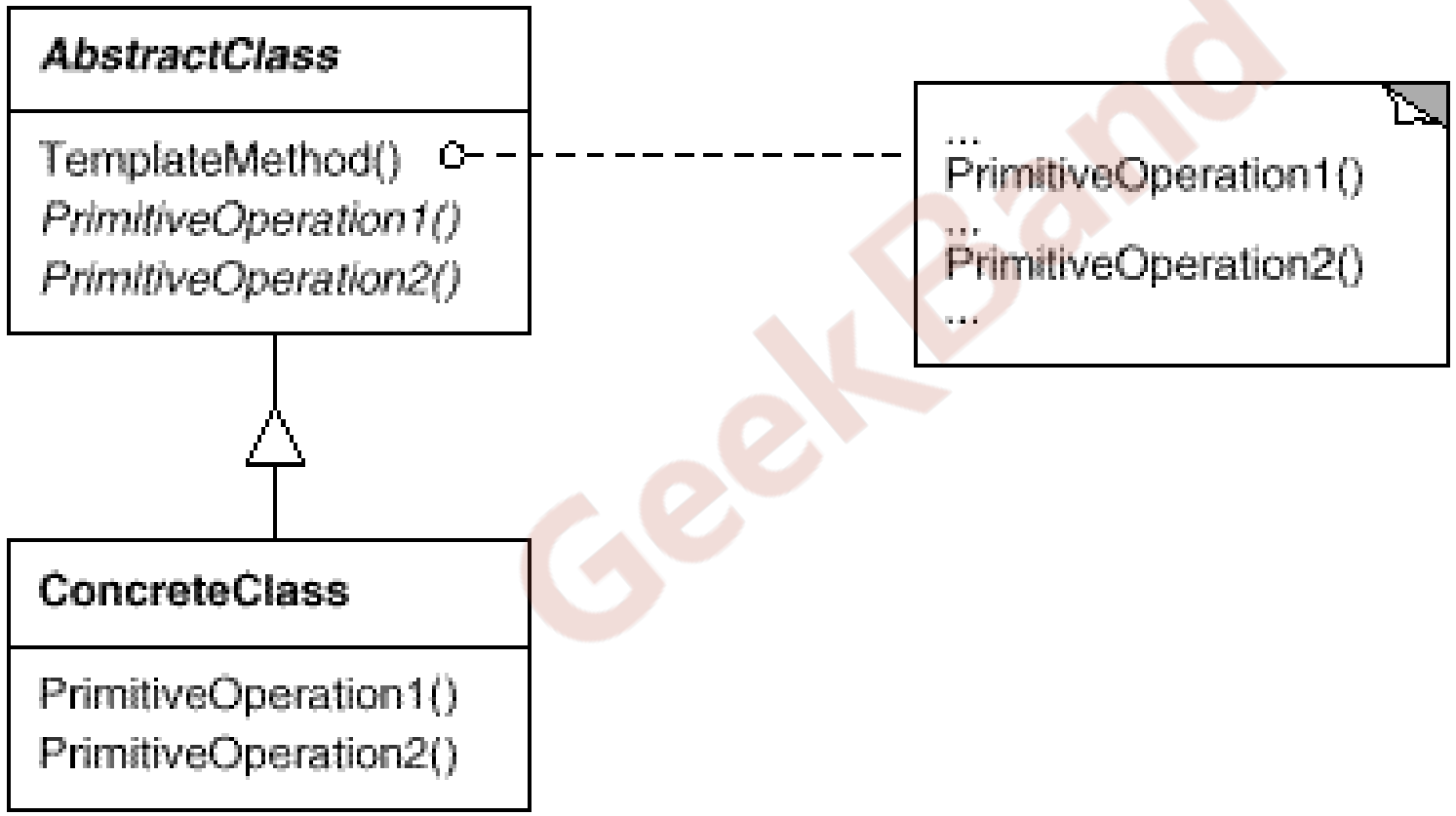


模式定义

定义一个操作中的算法的骨架 (稳定)，而将一些步骤延迟(变化)到子类中。Template Method使得子类可以不改变(复用)一个算法的结构即可重定义(override 重写)该算法的某些特定步骤。

——《设计模式》GoF

结构（Structure）



要点总结

- Template Method模式是一种非常基础性的设计模式，在面向对象系统中有着大量的应用。它用最简洁的机制（虚函数的多态性）为很多应用程序框架提供了灵活的扩展点，是代码复用方面的基本实现结构。

- 除了可以灵活应对子步骤的变化外，“不要调用我，让我来调用你”的反向控制结构是Template Method的典型应用。

- 在具体实现方面，被Template Method调用的虚方法可以具有实现，也可以没有任何实现（抽象方法、纯虚方法），但一般推荐将它们设置为protected方法。