GeekBand 极客班

互联网人才 + 油站!

C++设计模式

www.geekband.com

C++设计模式

设计模式总结

李建忠

一个目标

管理变化，提高复用！

两种手段

分解 vs. 抽象

八大原则

- 依赖倒置原则（DIP）
- 开放封闭原则（OCP）
- 单一职责原则（SRP）
- Liskov 替换原则（LSP）
- 接口隔离原则（ISP）
- 对象组合优于类继承
- 封装变化点
- 面向接口编程

重构技法

- 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

从封装变化角度对模式分类

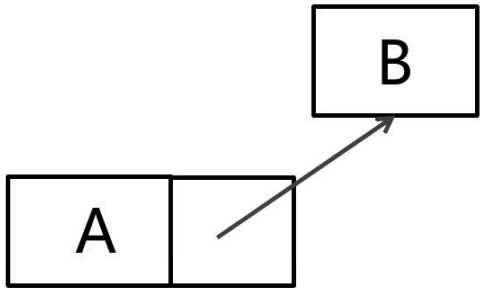
- 组件协作：
  - Template Method
  - Strategy
  - Observer / Event
- 单一职责：
  - Decorator
  - Bridge
- 对象创建:
  - Factory Method
  - Abstract Factory
  - Prototype
  - Builder
- 对象性能：
  - Singleton
  - Flyweight
- 接口隔离:
  - Façade
  - Proxy
  - Mediator
  - Adapter
- 状态变化：
  - Memento
  - State
- 数据结构：
  -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

C++ 对象模型

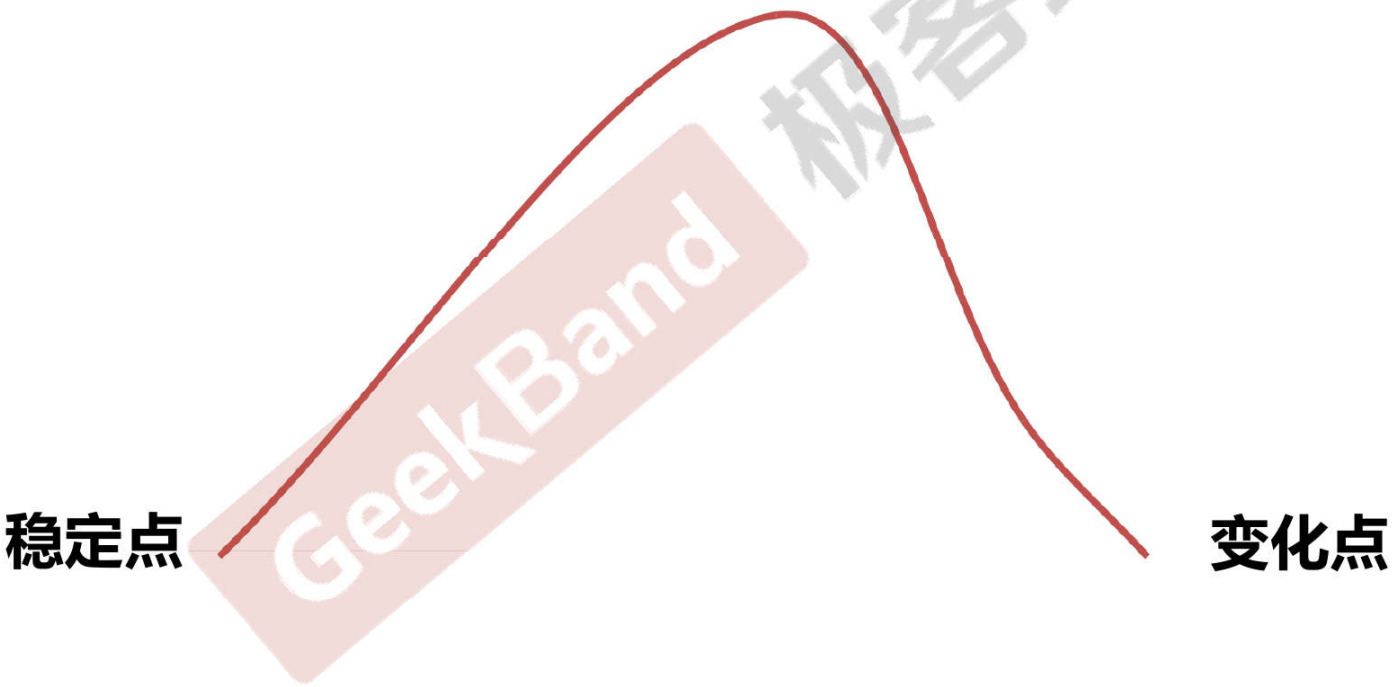
```
class A: B {
    //...
};
```

```
class A {
    B b;
    //...
};
```

```
class A {
    B* pb;
    //...
};
```



关注变化点和稳定点



什么时候不用模式

- 代码可读性很差时
- 需求理解还很浅时
- 变化没有显现时
- 不是系统的关键依赖点
- 项目没有复用价值时
- 项目将要发布时

经验之谈

- 不要为模式而模式
- 关注抽象类 & 接口
- 理清变化点和稳定点
- 审视依赖关系
- 要有Framework 和 Application的区隔思维
- 良好的设计是演化的结果

设计模式成长之路

- “手中无剑,心中无剑” ：见模式而不知
- “手中有剑,心中无剑” ：可以识别模式，作为应用开发人员使用模式
- “手中有剑,心中有剑” ：作为框架开发人员为应用设计某些模式
- “手中无剑,心中有剑” ：忘掉模式，只有原则

祝学习愉快，谢谢同学们！