Slow6502：设计模式报告（初稿）

Meow

软件学院

版本：0.10

更新：December 5, 2022

目录

1 项目简介 **3**

2 设计模式详述 **3**

2.1 单例（Singleton） . . . 3
2.2 原型（Prototype） . . . 3
2.3 工厂方法（Factory Method） . . . 3
2.4 抽象工厂（Abstract Factory） . . . 3
2.5 建造者（Builder） . . . 3
2.6 代理（Proxy） . . . 3
2.7 适配器（Adapter） . . . 3
2.8 桥接（Bridge） . . . 3
2.9 装饰（Decorator） . . . 4
2.10 外观（Facade） . . . 4
2.11 享元（Flyweight） . . . 4
2.12 组合（Composite） . . . 4
2.13 模板方法（TemplateMethod） . . . 4
2.14 策略（Strategy） . . . 4
2.15 命令（Command） . . . 4
2.16 职责链（Chain of Responsibility） . . . 4
2.17 状态（State） . . . 4
2.18 观察者（Observer） . . . 4
2.19 中介者（Mediator） . . . 5
2.20 迭代器（Iterator） . . . 5
2.21 访问者（Visitor） . . . 5

2.22 备忘录（Memento） 5
2.23 解释器（Interpreter） 5
2.24 MVP 模式 5
2.25 数据访问对象模式（Data Access Object Pattern） 5
2.26 传输对象模式 5
2.27 业务代表模式 5
2.28 前端控制器模式 5

3 项目总结 **5**

1 项目简介

2 设计模式详述

2.1 单例（Singleton）

模式某个类只能生成一个实例，该类提供了一个全局访问点供外部获取该实例，其拓展是有限多例模式。

2.2 原型（Prototype）

模式将一个对象作为原型，通过对其进行复制而克隆出多个和原型类似的新实例。

2.3 工厂方法（Factory Method）

模式定义一个用于创建产品的接口，由子类决定生产什么产品。

2.4 抽象工厂（Abstract Factory）

模式提供一个创建产品族的接口，其每个子类可以生产一系列相关的产品。

2.5 建造者（Builder）

模式将一个复杂对象分解成多个相对简单的部分，然后根据不同需要分别创建它们，最后构建成该复杂对象。

2.6 代理（Proxy）

模式为某对象提供一种代理以控制对该对象的访问。即客户端通过代理间接地访问该对象，从而限制、增强或修改该对象的一些特性。

2.7 适配器（Adapter）

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

2.8 桥接（Bridge）

模式将抽象与实现分离，使它们可以独立变化。它是用组合关系代替继承关系来实现，从而降低了抽象和实现这两个可变维度的耦合度。

2.9 装饰（Decorator）

模式动态的给对象增加一些职责，即增加其额外的功能。

2.10 外观（Facade）

模式为多个复杂的子系统提供一个一致的接口，使这些子系统更加容易被访问。

2.11 享元（Flyweight）

模式运用共享技术来有效地支持大量细粒度对象的复用。

2.12 组合（Composite）

模式将对象组合成树状层次结构，使用户对单个对象和组合对象具有一致的访问性。

2.13 模板方法（TemplateMethod）

模式定义一个操作中的算法骨架，而将算法的一些步骤延迟到子类中，使得子类可以不改变该算法结构的情况下重定义该算法的某些特定步骤。

2.14 策略（Strategy）

模式定义了一系列算法，并将每个算法封装起来，使它们可以相互替换，且算法的改变不会影响使用算法的客户。

2.15 命令（Command）

模式将一个请求封装为一个对象，使发出请求的责任和执行请求的责任分割开。

2.16 职责链（Chain of Responsibility）

模式把请求从链中的一个对象传到下一个对象，直到请求被响应为止。通过这种方式去除对象之间的耦合。

2.17 状态（State）

模式允许一个对象在其内部状态发生改变时改变其行为能力。

2.18 观察者（Observer）

模式多个对象间存在一对多关系，当一个对象发生改变时，把这种改变通知给其他多个对象，从而影响其他对象的行为。

2.19 中介者（Mediator）

模式定义一个中介对象来简化原有对象之间的交互关系，降低系统中对象间的耦合度，使原有对象之间不必相互了解。

2.20 迭代器（Iterator）

模式提供一种方法来顺序访问聚合对象中的一系列数据，而不暴露聚合对象的内部表示。

2.21 访问者（Visitor）

模式在不改变集合元素的前提下，为一个集合中的每个元素提供多种访问方式，即每个元素有多个访问者对象访问。

2.22 备忘录（Memento）

模式在不破坏封装性的前提下，获取并保存一个对象的内部状态，以便以后恢复它。

2.23 解释器（Interpreter）

模式提供如何定义语言的文法，以及对语言句子的解释方法，即解释器。

2.24 MVP 模式

2.25 数据访问对象模式（Data Access Object Pattern）

2.26 传输对象模式

2.27 业务代表模式

2.28 前端控制器模式

3 项目总结