OO目的

封装

归一化

OO原则

|  |  |
| --- | --- |
| 名称 | 内容 |
|  | 将变化的部分与不变的部分分离 |
|  | 针对接口编程，而不是针对实现编程 |
|  | 多用组合，少用继承 |
|  | 减小交互对象之间的耦合 |
| 开放-关闭原则 | 类对扩展开放，对修改关闭 |
| Dependency Inversion Principle  依赖倒置原则 | 依赖抽象，不依赖具体类 |
| Least Knowledge Principle  最少知识原则（得墨忒耳法则、迪米特法则） | 减少对象之间的交互 |
| 好莱坞原则 | 高层组件依赖底层组件，低层组件不能依赖高层组件 |
|  | 每个类只有1个责任，也就只有1个被改变的原因 |

|  |
| --- |
| 设计模式 |
| 架构模式 |
| 应用模式 |
| 领域特定模式 |
| 业务流程模式 |
| 组织模式 |
| 用户界面设计模式 |

OO设计模式

1、创建型模式

|  |  |
| --- | --- |
| 名称 | 实现方法 |
| 简单工厂 | 将创建子类对象（产品类）返回超类对象（产品类）的部分封装 |
| Factory method pattern  工厂方法模式 | 把实例化推迟到子类   |  |  |  |  |  | | --- | --- | --- | --- | --- | | OO | 创建者类 | | 产品类 | | | 抽象基类 | 具体子类 | 抽象基类 | 具体子类 | | 不实现创建对象（产品类）部分（或实现一个默认的工厂方法） | 实现创建对象（产品类）部分 |  |  | |
| Abstract factory pattern  抽象工厂模式 | 客户用抽象基类（创建者类）创建相关的一组对象（产品类）  使用了工厂方法模式   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | OO | 创建者类 | | 产品1类 | | ... | 产品n类 | | | 抽象基类 | 具体子类 | 抽象基类 | 具体子类 | 抽象基类 | 具体子类 | | / | 实现创建对象组（产品类）部分 | / |  | / |  | |
| Builder Pattern  生成器模式 | 先定义实例化对象可能用到的步骤，每次按具体情况确定按什么顺序使用哪些步骤   |  |  |  |  | | --- | --- | --- | --- | | OO | 创建者类 | | 产品类 | | 抽象基类 | 具体子类 | | 可能用到的步骤 | 设置对象的属性 |  | |
| Singleton pattern  单例模式（单件模式） | 一个类只有一个实例并提供一个全局的访问点  减少了命名空间污染   |  |  | | --- | --- | | 对资源敏感且有可能始终不被使用的对象 | 延迟实例化  多线程：同步方法/双重检查加锁 | | 总是被使用的对象 | 急切实例化 | |
| Prototype pattern  原型模式 |  |

2、结构型模式

|  |  |  |
| --- | --- | --- |
| 名称 | 实现方法 | 其它用途 |
| Decorator pattern  修饰模式（装饰者模式） | 动态地将属性附加到对象上   |  |  |  |  | | --- | --- | --- | --- | | OO | 抽象基类 | 具体子类 | | | 被装饰组件 | 装饰者 | | / |  | has-a抽象基类对象 |   装饰者不知道装饰链中其它装饰者的存在，不能依赖装饰次数或次序 |  |
| Adapter pattern  适配器模式 | 将被适配类的接口转换成可被客户类使用的接口（目标接口）   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | OO | 对象适配器 | |  |  |  | | --- | --- | --- | | 客户类 | 目标类 | 被适配对象 | | 适配器对象 | | 使用目标类对象 | has-a被适配对象 |  | | | 类适配器 | |  |  |  | | --- | --- | --- | | 客户类 | 目标类 | 被适配类 | | 适配器类 | | | 使用目标类对象 |  | | | |  |
| Facade pattern  外观模式 | 将复杂的接口转换为简单的接口  与适配器模式类似 |  |
| Composite Pattern  组合模式 | 将对象组织成树形结构   |  |  |  | | --- | --- | --- | | OO | 组件类 | | | 叶节点类 | 组合类 | |  | has-a不定个数个组件对象 | |  |
| Proxy pattern  代理模式 | 为对象提供1个替身来控制对其的访问   |  |  |  |  | | --- | --- | --- | --- | | OO | 主题类 | | | | 抽象基类 | 具体子类 | | | 代理类 | 真正的主题类 | | / | has-a服务类对象 |  | | 远程代理：远程的对象  虚拟代理：创建开销大的对象  保护代理：需要安全控制的对象 |
| Bridge pattern  桥接模式 |  |  |
| Flyweight pattern  享元模式（蝇量模式） |  |  |

3、行为型模式

|  |  |  |
| --- | --- | --- |
| 名称 | 实现方法 | 其它用途 |
| Strategy pattern  策略模式 | 将算法封装，算法之间可以相互替换   |  |  |  | | --- | --- | --- | | P | 回调函数 | | | OO | 抽象基类 | 具体子类 | | / | 算法族中的一个算法 | |  |
| Template method pattern  模板方法模式 | 先定义算法的骨架，每次按具体情况确定算法的某几个步骤  钩子：默认情况下不做任何操作的步骤   |  |  |  | | --- | --- | --- | |  | 编译 | 运行 | | P | 函数模板 | 函数指针 | | OO | 类模板 | |  |  | | --- | --- | | 抽象基类 | 具体子类 | | 算法的骨架 | 填补或覆盖算法的某几个步骤 | | |  |
| State pattern  状态模式 | 对象内部状态改变时，对相同调用产生不同的反应   |  |  |  |  | | --- | --- | --- | --- | | P | switch、select case | | | | OO | 上下文类 | 状态类 | | | 抽象基类 | 具体子类 | | has-a固定个数个具体子类实例（状态类） | / |  | |  |
| Observer Pattern  观察者模式 | 对象状态改变时通知所有依赖它的对象   |  |  |  |  | | --- | --- | --- | --- | | P | 回调函数 | | | | OO | 主题类 | 观察者类 | | | 抽象基类 | 具体子类 | | has-a不定个数个抽象基类实例（观察者类） | / |  |   注册、取消注册、通知  通知时：主题对象推状态 和/或 通知后观察者对象拉状态  不能依赖通知次序 |  |
| Command pattern  命令模式 | 将请求封装   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | OO | 客户对象 | 命令类 | | 调用者对象 | 接收者对象 | | 抽象基类 | 具体子类 | | 创建具体子类实例（命令类）给调用者对象 | / | 含有接收者对象、接收者对象要执行的动作 | has-a抽象基类实例（命令类） |  |   可能不存在接收者对象，所有工作都是命令对象完成的 | 撤销（命令类对象的1个函数成员）  重复  宏命令：成组的命令  队列  日志（命令类对象的2个函数成员） |
| Iterator pattern  迭代器模式 | 将遍历集合的方法封装  使用了工厂方法模式   |  |  |  |  |  | | --- | --- | --- | --- | --- | | OO | 集合类 | | 迭代器类 | | | 抽象基类 | 具体子类 | 抽象基类 | 具体子类 | |  | 负责实例化一种具体迭代器 |  |  |   不能依赖迭代器返回的元素的顺序 |  |
| Chain of responsibility pattern  责任链模式 |  |  |
| Interpreter pattern  解释器模式 |  |  |
| Visitor pattern  访问者模式 |  |  |
| Mediator pattern  中介者模式 |  |  |
| Memento pattern  备忘录模式 |  |  |

其它模式

|  |  |
| --- | --- |
| 名称 | 实现方法 |
| Null object pattern  空对象 | 表示没有对象  负责处理null、移除了判断是否为null的条件语句 |

Compound pattern复合模式

|  |  |  |
| --- | --- | --- |
| 名称 | 实现方法 | 其它用途 |
| Model–view–controller（MVC）  模型–视图–控制器 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | OO | 模型类 | 控制器类 | | 视图类 | | 抽象基类 | 具体子类 | |  | / | has-a模型对象，has-a视图对象  初始化时创建视图对象 | 使用了组合模式  可能has-a模型对象  has-a控制器对象 |   模型类可以同时使用多个视图，视图是模型的观察者：观察者模式  视图类使用不同的控制器类：策略模式 |  |
| Model 2 | 使用了MVC   |  |  |  |  | | --- | --- | --- | --- | | 模型 | Bean | Servlet | JSP | | 封装在Bean背后 | 将请求转发给模型，返回模型的处理结果 | 控制器，接收HTTP请求 | 视图，发送HTTP回应 | |  |