


在线考研模拟系统 -- 顶峰考研云

软件设计模式期末项目答辩

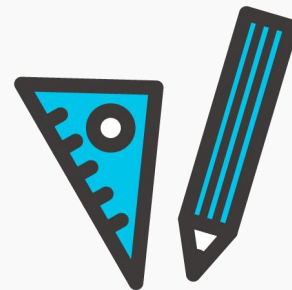
 汇报人：沙坚 焦骛

 指导老师：史扬

小组成员：

储岱泽 傅佳恒 夏尧民 韩嘉睿 段婷婷 莫益萌 焦骛 王宜沣 周文玥
徐嘉琪 沙坚





目录

CONTENTS

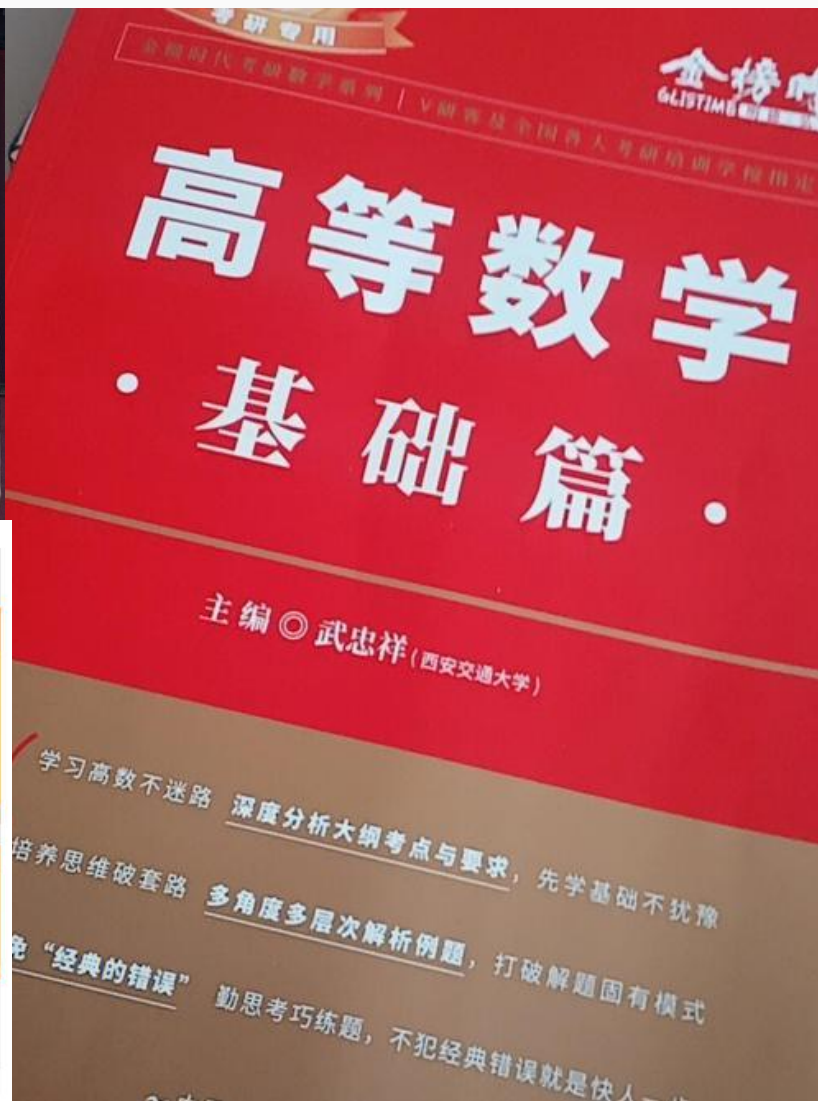
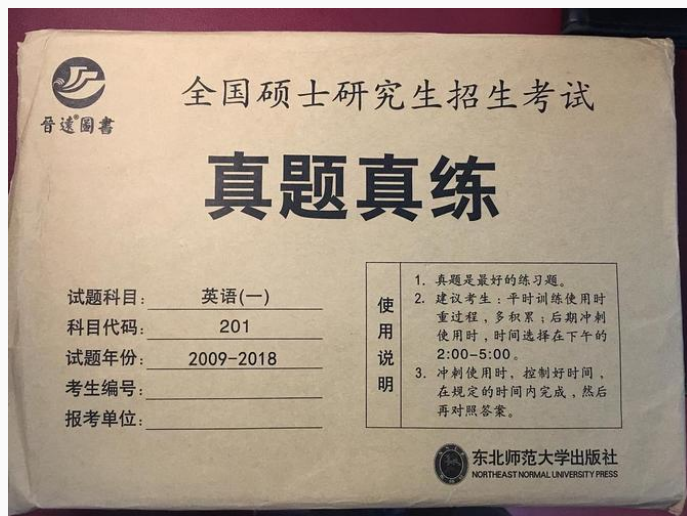


1

项目背景

2

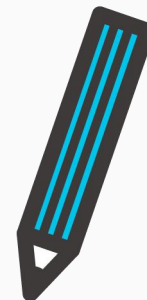
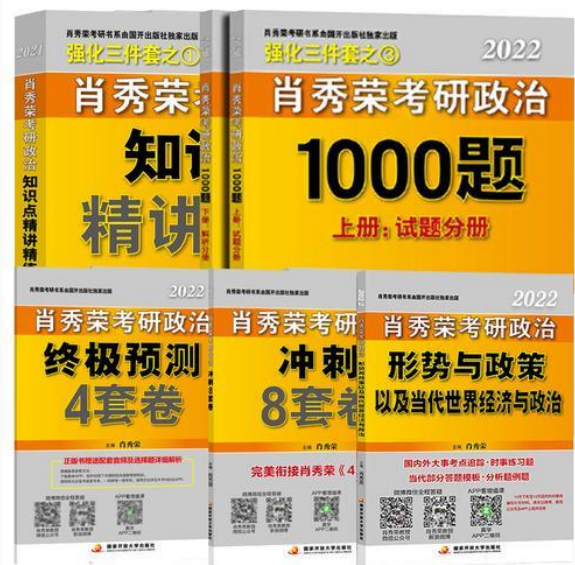
设计模式简介



2024年考研刚刚结束，新同学们可以开始准备2025年的考研啦！
为了在新一年的考研中取得理想的成绩，同济大学民间推出了一款名叫

“顶峰考研云”

的应用，在该应用中，系统存储了大量往年的考研真题以及相关习题，并邀请名师进行考题预测，进行模拟试卷的编写，同学们可以在该应用中和研友一起进行模拟考试PK，考题交流，同时也有进行教辅资料的购买等功能



1

学生管理模块

- 学生注册
- 获取学生信息列表
- 修改学生姓名
- 获得学生物品信息
- 学生交流
- 加入学习小组
- 获得学习小组信息
- 输出学生状态
- 打印成绩单

2

教师管理模块

- 创建教师信息
- 创建课程考试
- 输出每一场考试的信息
- 进行测试
- 获取所有学生编号和姓名

3

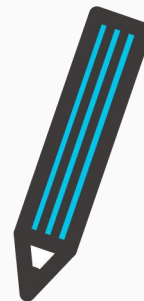
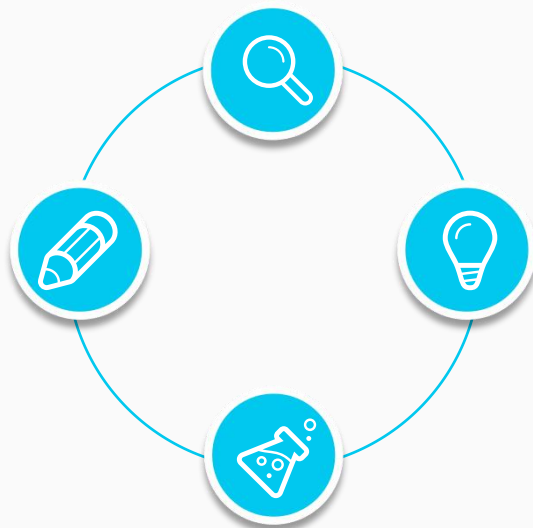
考试管理模块

- 学生参加考试
- 输出考试准备信息
- 查看指定课程考试指定学生的排名
- 获得考试学生列表
- 查看学生成绩

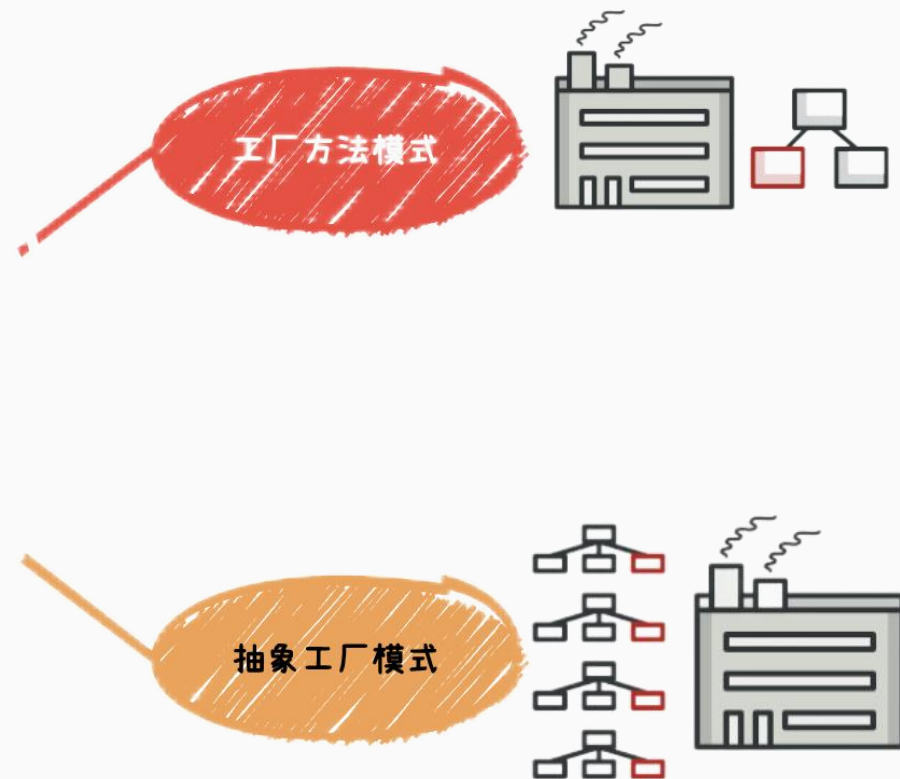
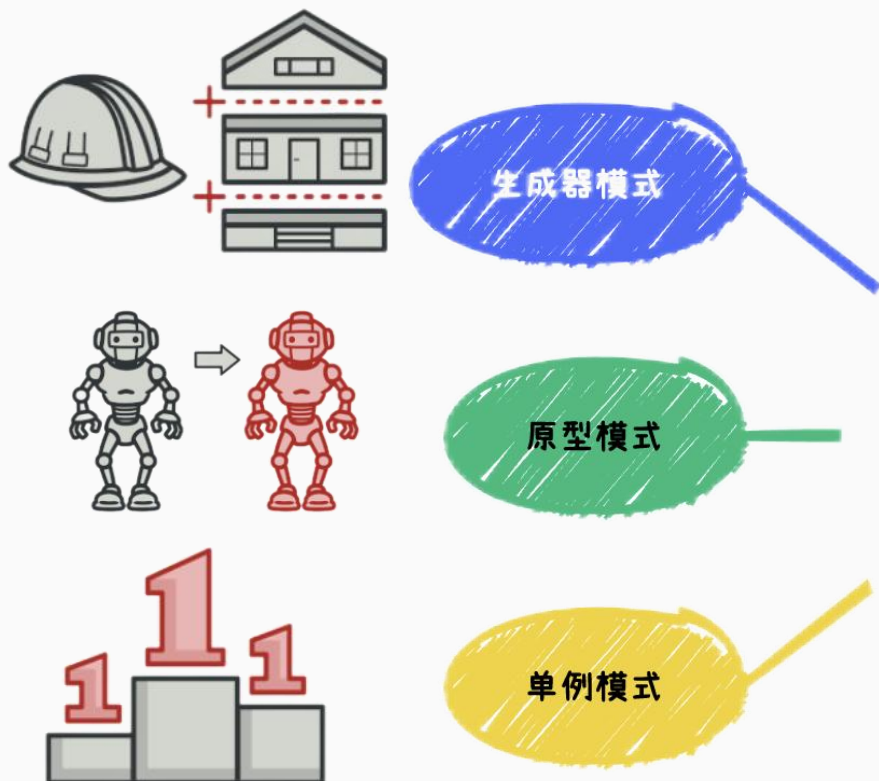
4

教辅资料模块

- 进行教辅资料采购
- 输出教辅资料采购信息
- 创建物品
- 输出教辅资料点单信息
- 支付

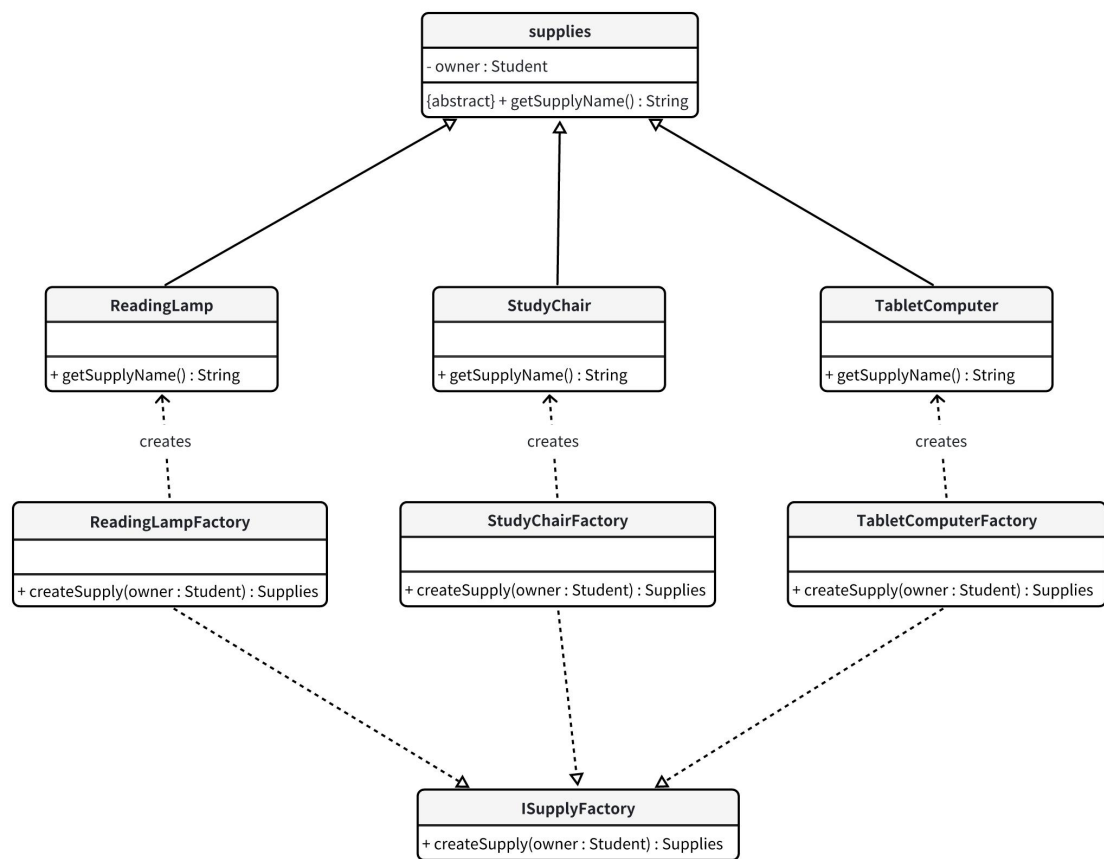
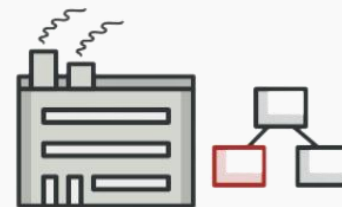


创建型模式



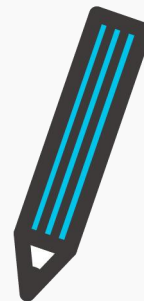
工厂方法模式 - - 类图分析

Factory Method



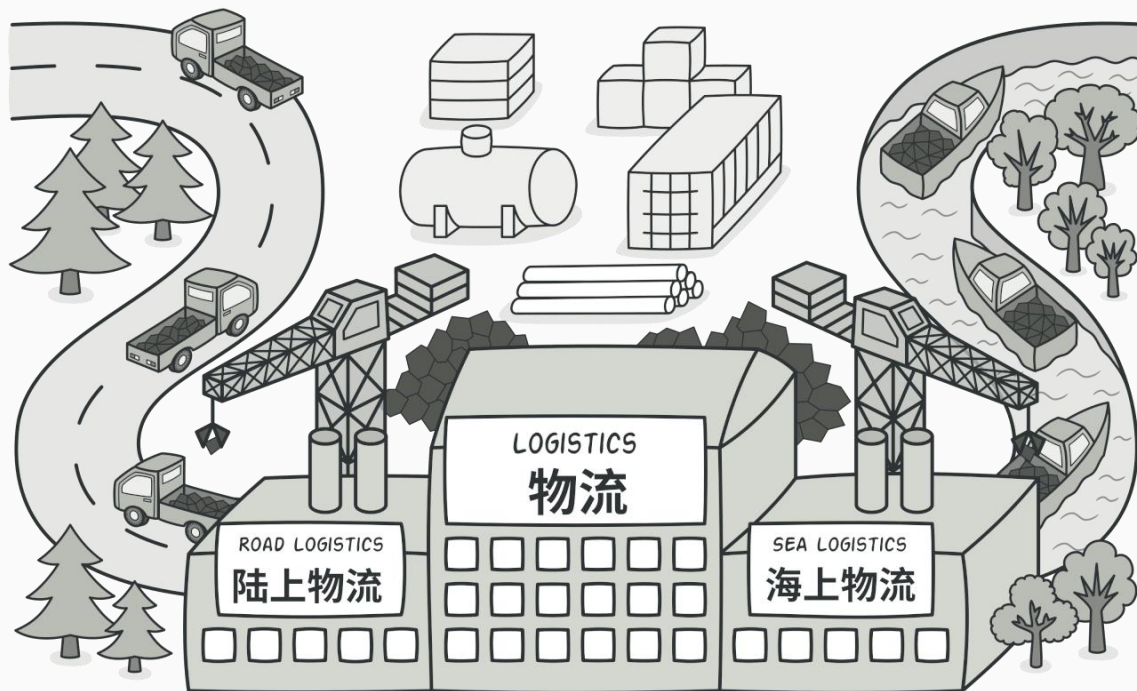
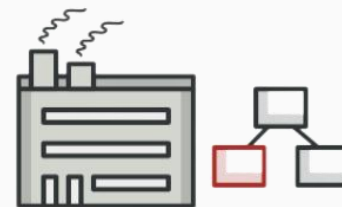
工厂方法模式（Factory Method Pattern）是一种创建型设计模式，其核心在于定义一个用于创建对象的接口，但让子类决定要实例化的类是哪一个。工厂方法模式使一个类的实例化延迟到其子类。

在“顶峰考研云”系统中，工厂方法模式的实现可以针对不同类型的学习资源或工具进行优化。该系统可能包含多种类型的学习辅助工具和资源，如阅读灯、学习椅、电脑等，每种都有其特定的属性和功能。



工厂方法模式 - - 代价分析

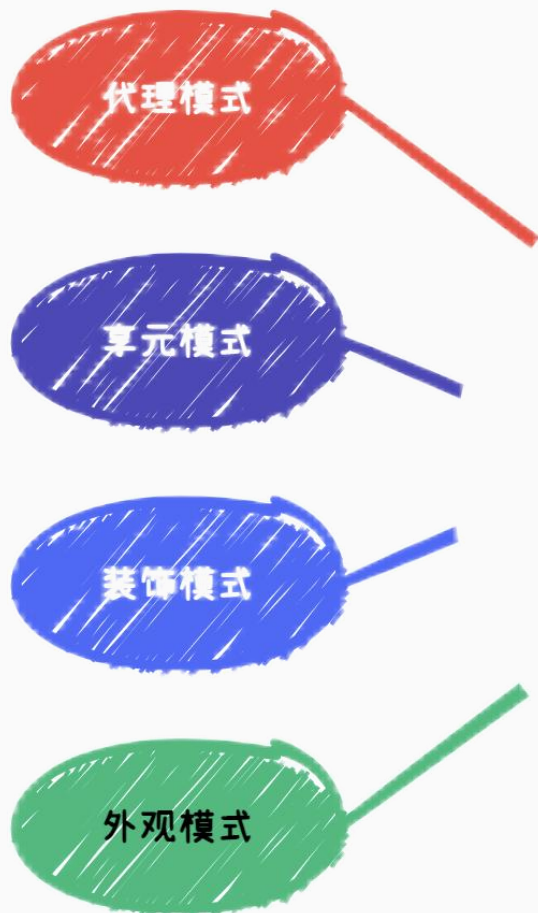
Factory Method



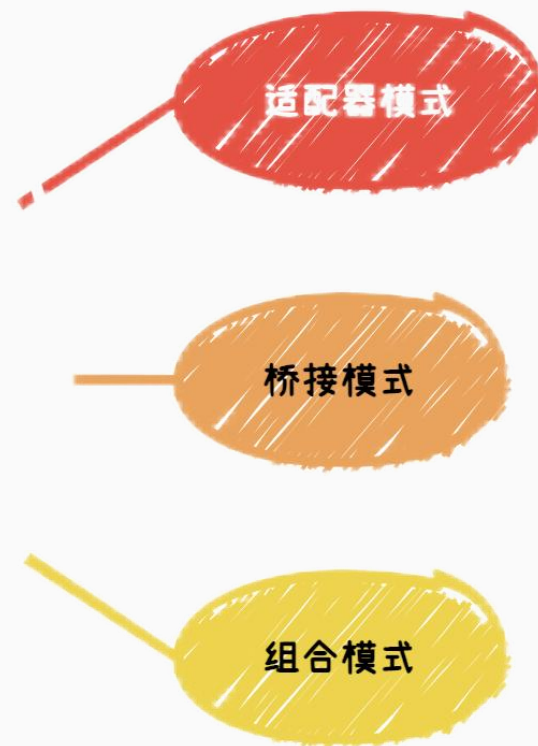
- 实现复杂度先增加后减少
- 可维护性先增加后减少
- 可扩展性增加
- 性能影响较小

结构型模式

Structural Design Patterns

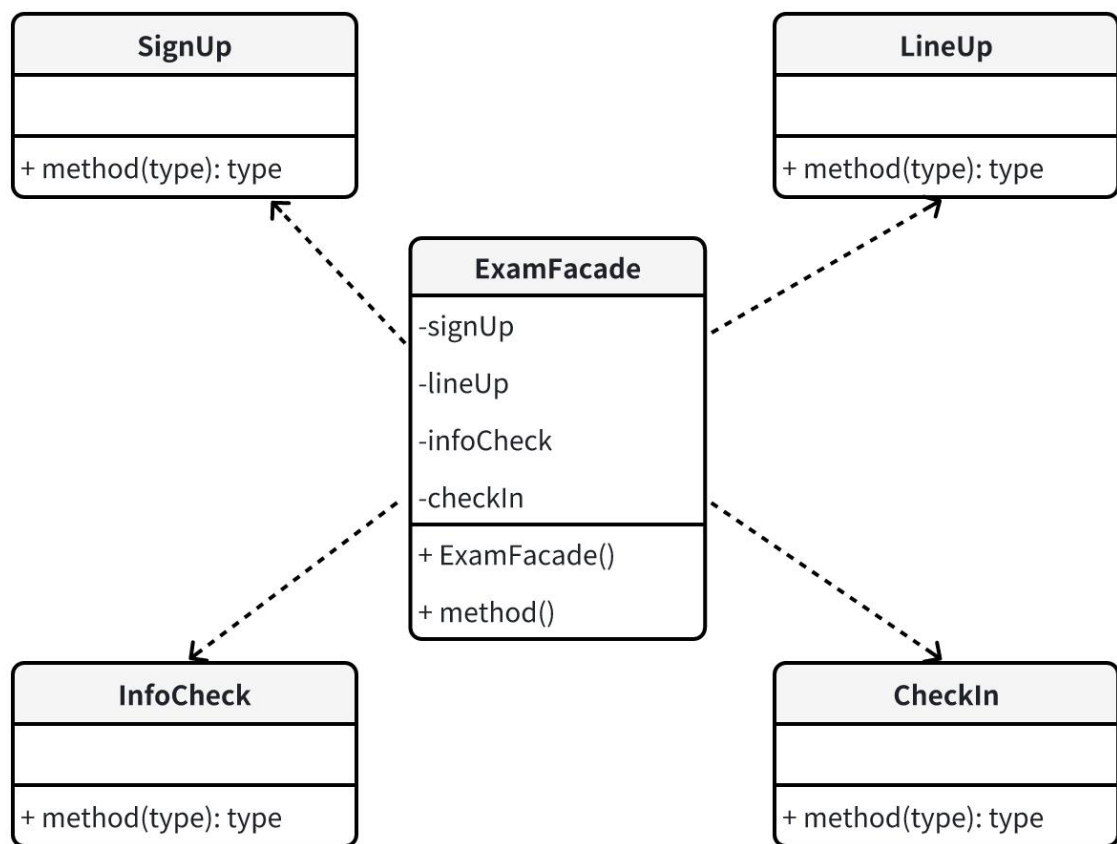


结构模式



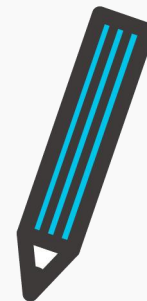
外观模式 - - 类图分析

Facade Pattern



外观模式 (Facade Pattern) 是一种常用的软件设计模式，外部代码可以通过这个“外观”接口与系统交互，而无需直接处理复杂的内部逻辑。

在顶峰考研云的系统中，当学生选择参加考试后，他们需要完成报名、排队、信息检查和入场等四个步骤，方可正式开始考试。这四个步骤虽然各自独立，但都是考试准备过程的一部分，且最终目的一致。因此，我们可以将这些步骤抽象化，并设计一个统一的高层接口，以简化对学生而言可能较为复杂的报名流程。



外观模式 - - 代价分析

Facade Pattern



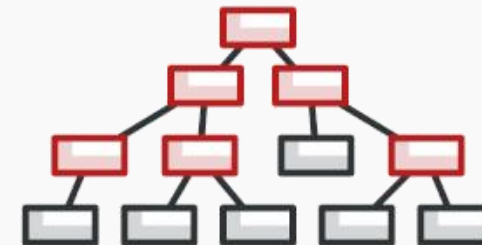
- 不符合开放封闭原则
- 隐藏系统复杂性
- 过度使用可能导致系统变得难以理解
- 性能开销



- 简化接口
- 降低耦合度
- 提高可用性
- 易于维护和扩展

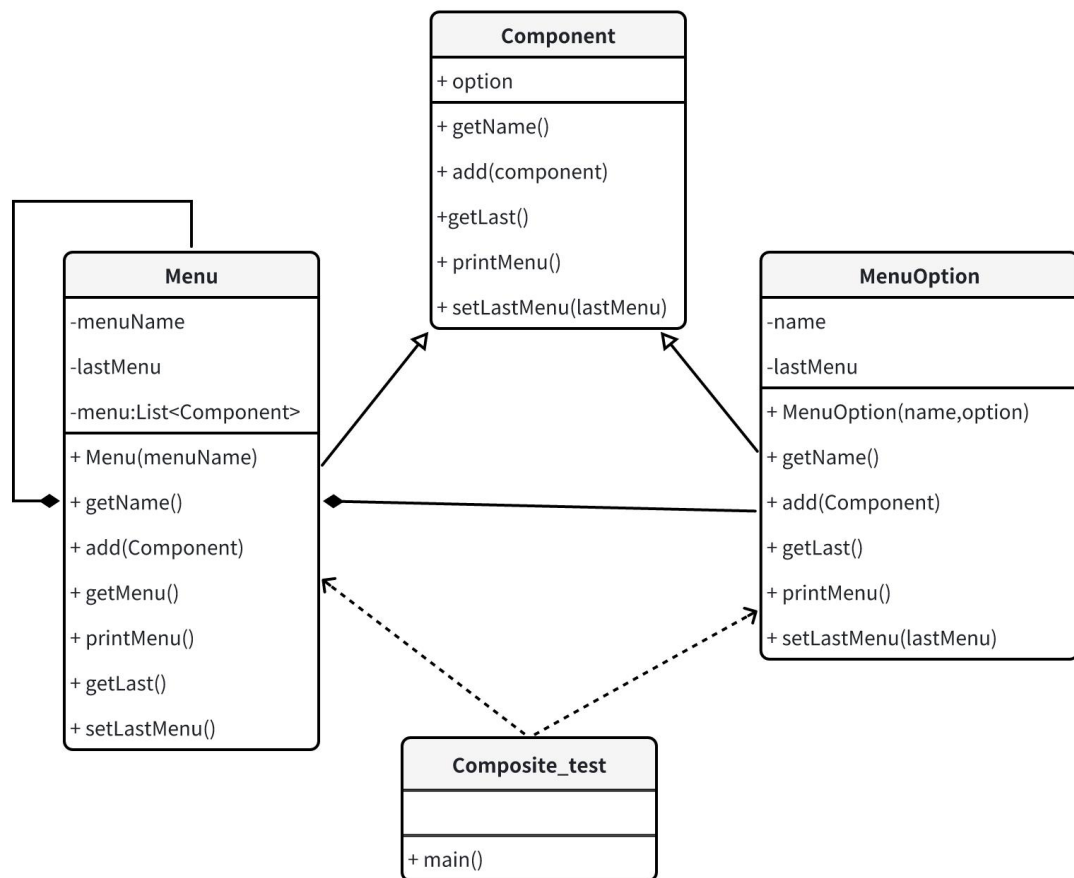
组合模式 - - 类图分析

Composite Pattern



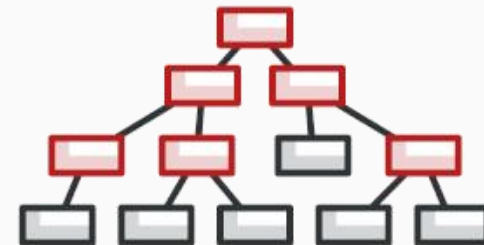
组合模式（Composite Pattern），也被称为部分-整体模式，是一种结构型设计模式，用于将一组相似的对象视为一个单一的对象。该模式基于树形结构来组织对象，以表示对象的部分及整体层次关系。

在顶峰考研云的系统中，在 CompositeTest 类里，我们采用了组合模式来模拟文件与文件夹的关系，通过设计 MenuOption 和 Menu 两类，这两者都继承自 Component 类，来分别表示菜单中的单个选项和二级菜单。



组合模式 - - 代价分析

Composite Pattern



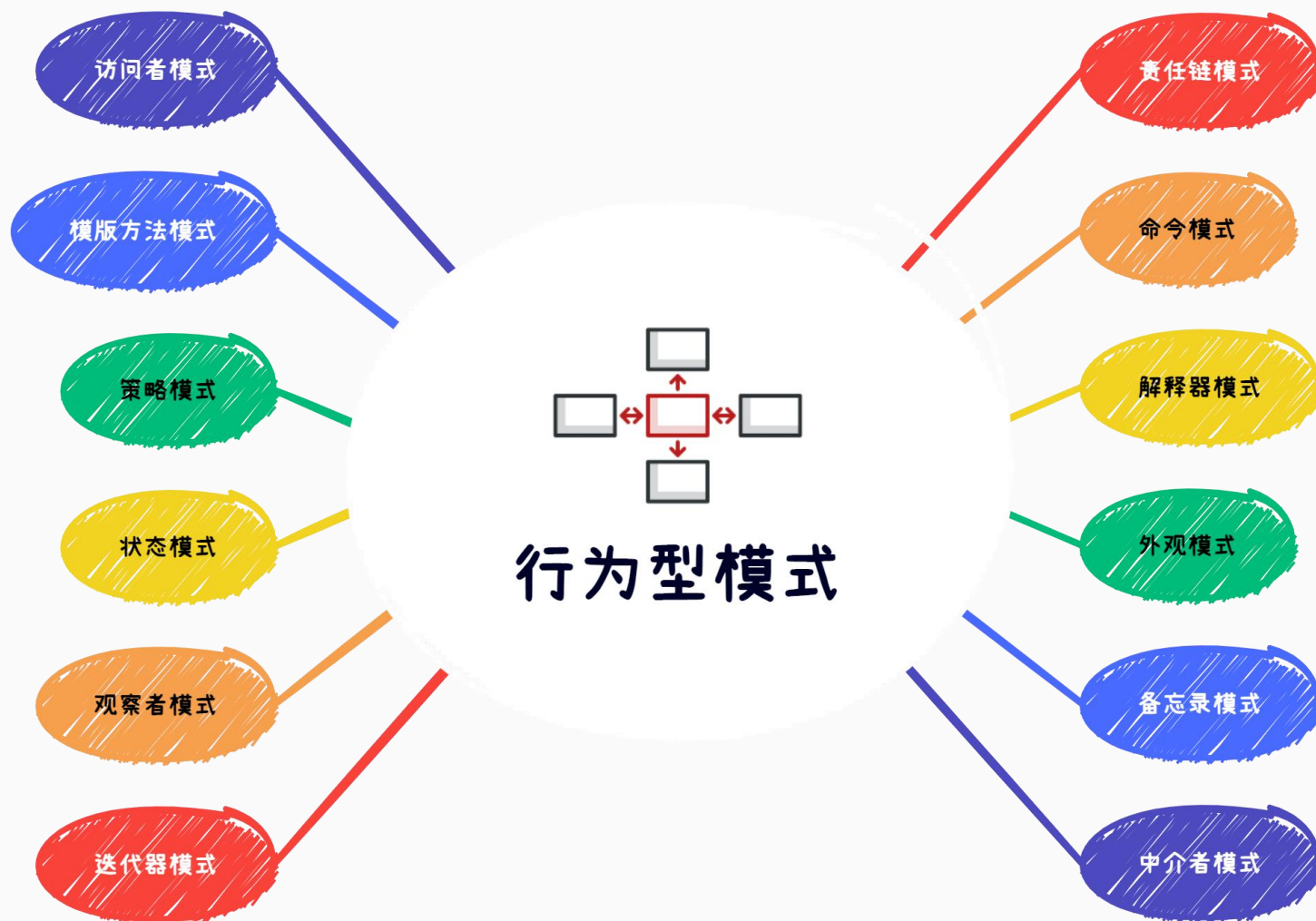
- 设计的复杂性
- 难以限制组件
- 性能开销
- 过度一般化



- 统一的对象处理
- 清晰的层次结构
- 增强的灵活性
- 简化的设计

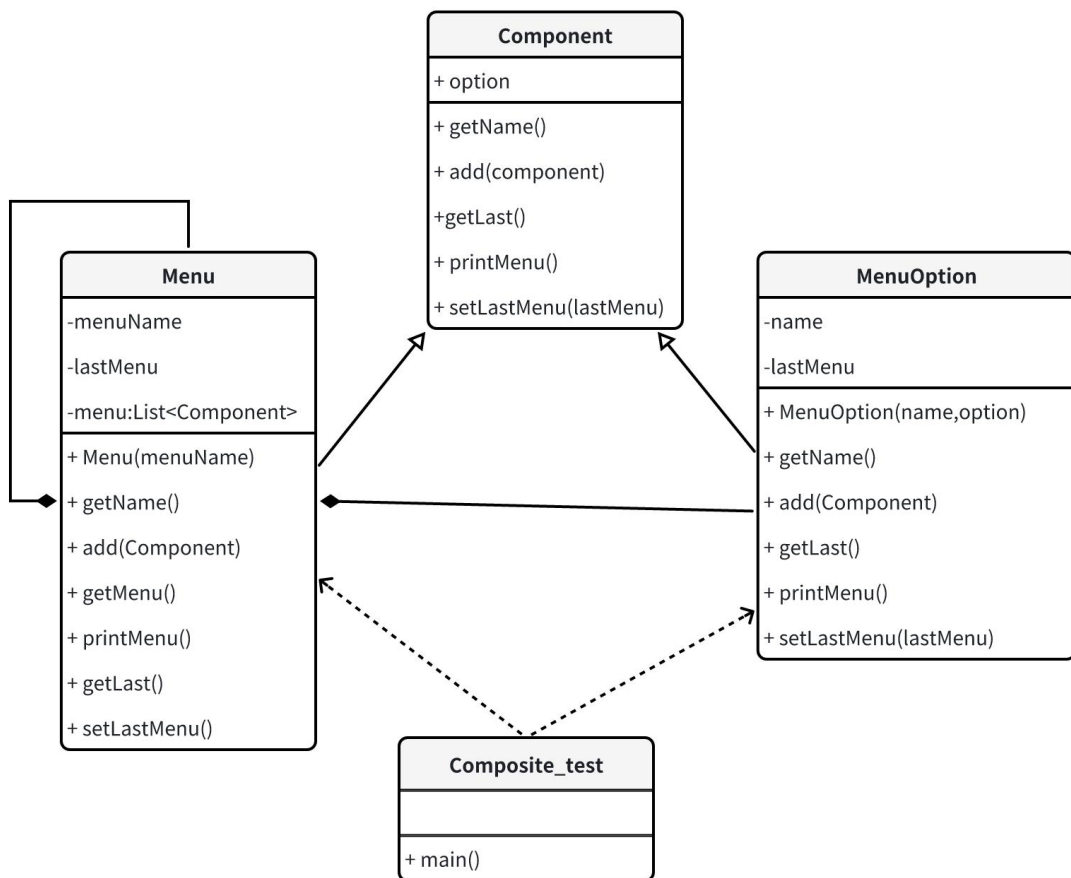
行为型模式

Behavioral Pattern



命令模式 - - 类图分析

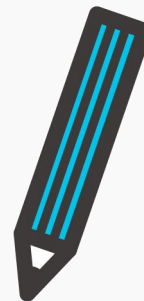
Command Pattern



命令模式(Command Pattern)是一种行为型设计模式，它允许将请求封装成一个独立的对象，并将请求的发送者和接收者解耦，使得请求可以被保存、传递、记录、撤销或重做，从而提供更大的灵活性和可扩展性。

在顶峰考研云中，注册用户可以进行教辅资料的采购，在顶峰资料采购中心中，用户可以在商城购买书籍用于复习和刷题，用户在购买书籍后可以顺便购买一些学习辅助材料进行学习刷题的辅助。

用户可以对自已的点单进行**撤销**操作，所以为了能够方便对于同一条命令的多次调用以及对于调用命令的撤销，我们选择使用命令模式进行该功能的实现。



命令模式 - - 代价分析

Composite Pattern



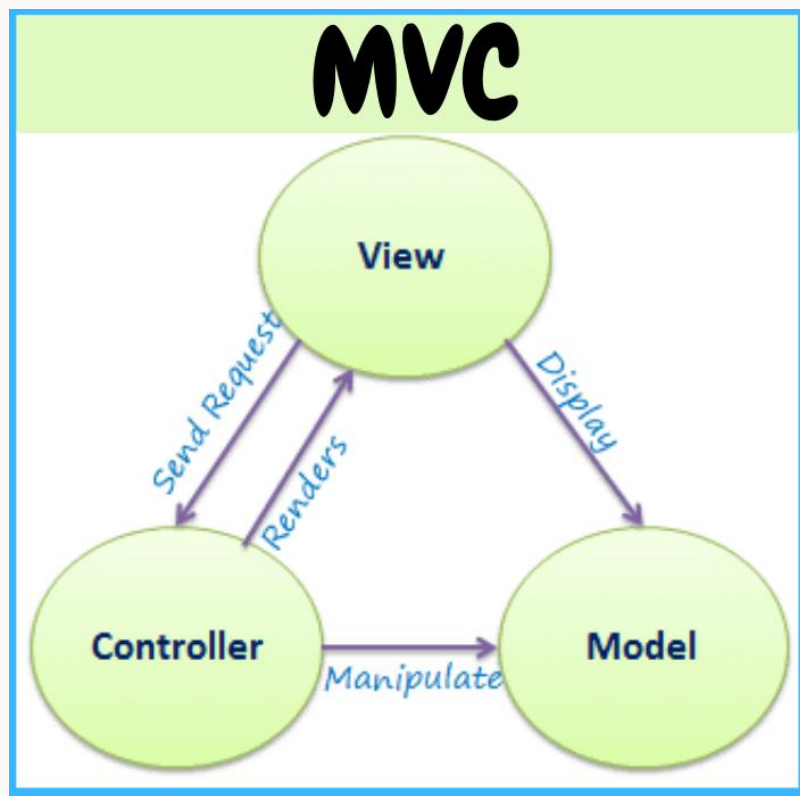
- 设计的复杂性
- 难以限制组件
- 性能开销
- 过度一般化



- 统一的对象处理
- 清晰的层次结构
- 增强的灵活性
- 简化的设计

MVC模式

MVC Pattern

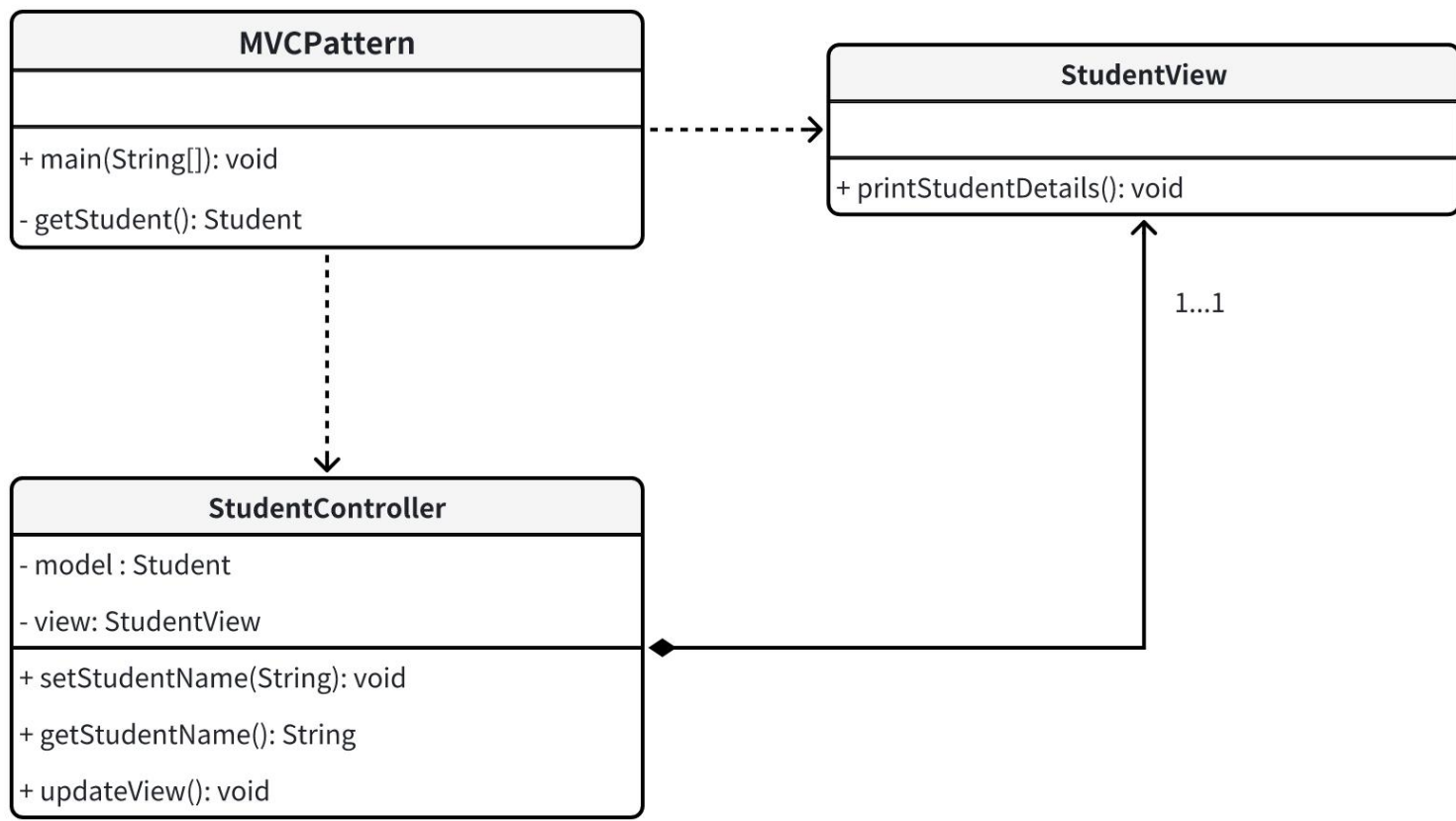


MVC模式 (Model-View-Controller Pattern) 是一种架构设计模式，用于将应用程序分为三个主要的逻辑组件：模型 (Model)、视图 (View) 和控制器 (Controller)。这种分离有助于管理复杂的应用程序，因为可以独立地维护或修改每个部分。

在“顶峰考研云”项目中，MVC模式被用于管理学生信息的展示和更新。该模式的实现涉及三个主要组件：Student模型、StudentView视图和StudentController控制器。

MVC模式

MVC Pattern



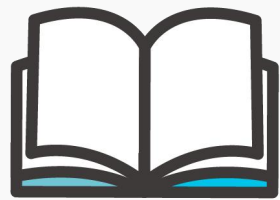
MVC模式

MVC Pattern

- 增加了系统的复杂性
- 性能开销
- 过度工程化
- 视图与模型间的潜在耦合风险



- 分离关注点
- 提高可维护性
- 增强可测试性
- 灵活性和可扩展性
- 促进团队协作



2024

感谢各位的观看

