**《软件工程》**

**实验报告七：设计模式以及构建设计**

**姓 名： 周子雄 学 号： 202210120127**

**院 系： 计算机与信息学院 专 业： 计算机科学与技术**

**实 验 室： j1307 实验日期： 2024/12/3**

**总评成绩： 审阅教师：**

# 实验目的

1. 了解软件工程工具生成代码的方法，理解代码自动生成的过程，破除自动生成代码“黑科技”
2. 掌握构件设计方法
3. 学习Bridge设计模式，体会设计模式的作用

# 实验环境

idea，PlantUML

# 实验要求

1.参考第7章 设计概念（design principle &pattern）（4学时），学习Bridge设计模式的使用；

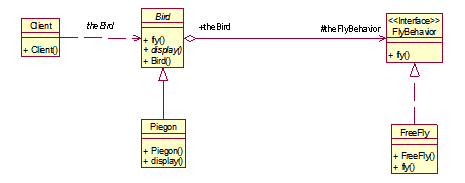
2.学习 Rose 生成代码的方法，理解代码自动生成原理

3.文件以“学号-姓名-软件工程实验七.rar”的方式命名，把模型文件和工程实现代码打包，提交到长江雨课堂软件工程实验七；

# 实验内容

1. 按照给出的 Bridge 设计模式 构建类图

有一个叫 HuntBird 的游戏，里面需要表示各种各样的鸟类。采用Bridge设计模式实现，针对各种鸟类，采用Bridge设计模式建模，如下图：



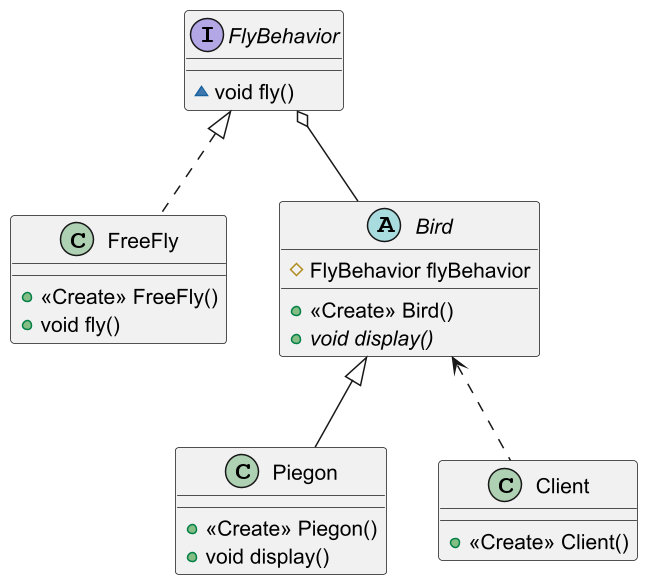
注意：

（1）绘制类之前，首先设置默认语言为Java/C++（本例以Java说明）

（2）Bird类为抽象类

（3）display方法为抽象方法

（4）Bird具有保护类型的FlyBehavior属性；



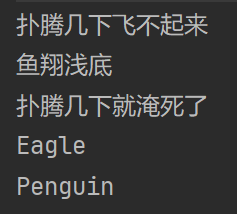
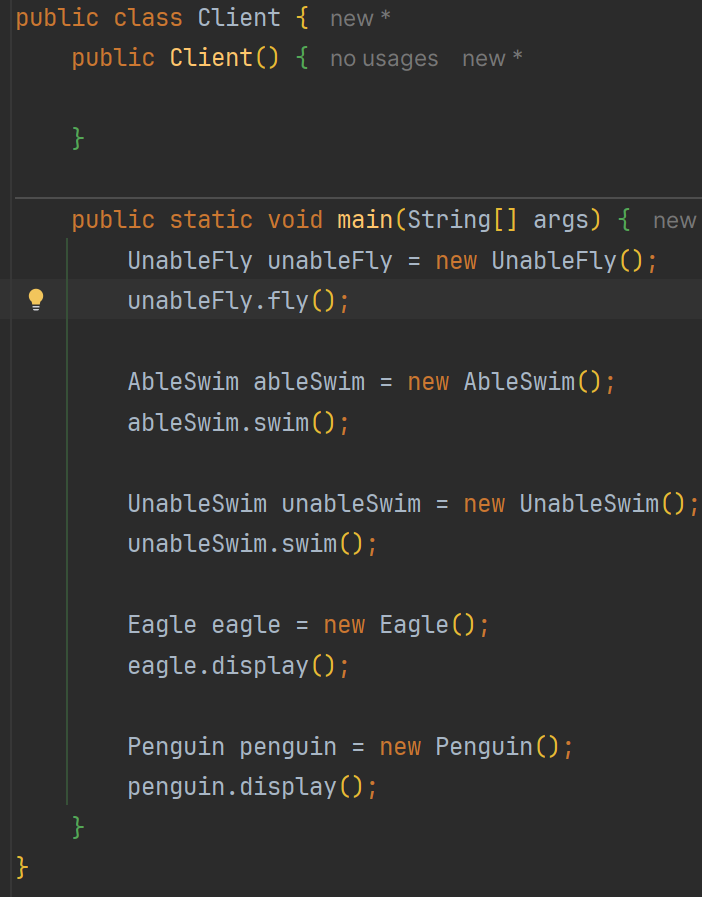
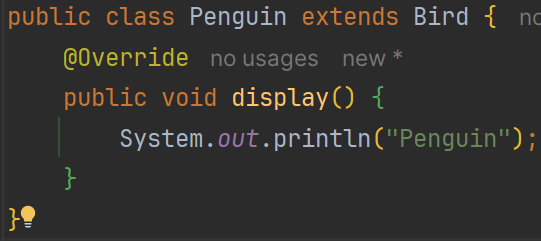
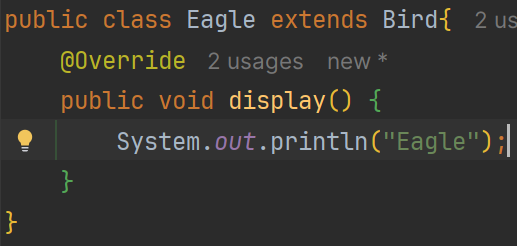
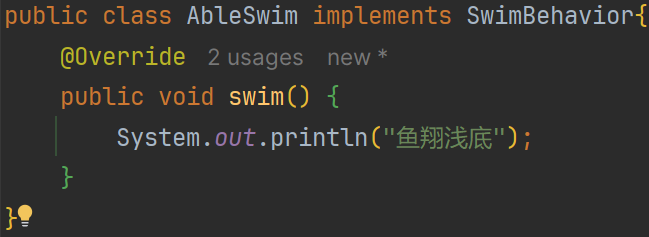
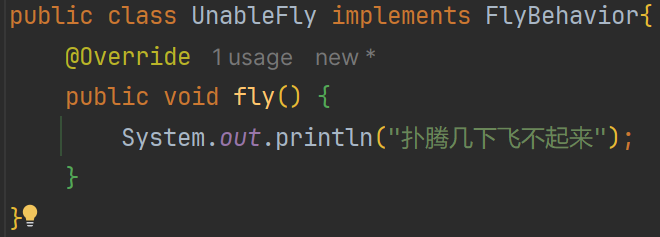
2. 按照以下给出的要求，扩展类图，并生成代码框架，实现代码，测试

(1)添加一个飞翔的行为的实现类UnableFly，实现fly接口方法：扑腾几下飞不起来；

(2)添加游泳的行为SwimBehavior接口，两个实现类AbleSwim，实现swim接口方法：鱼翔浅底；UnableSwim实现类实现swim接口方法：扑腾几下就淹死了：

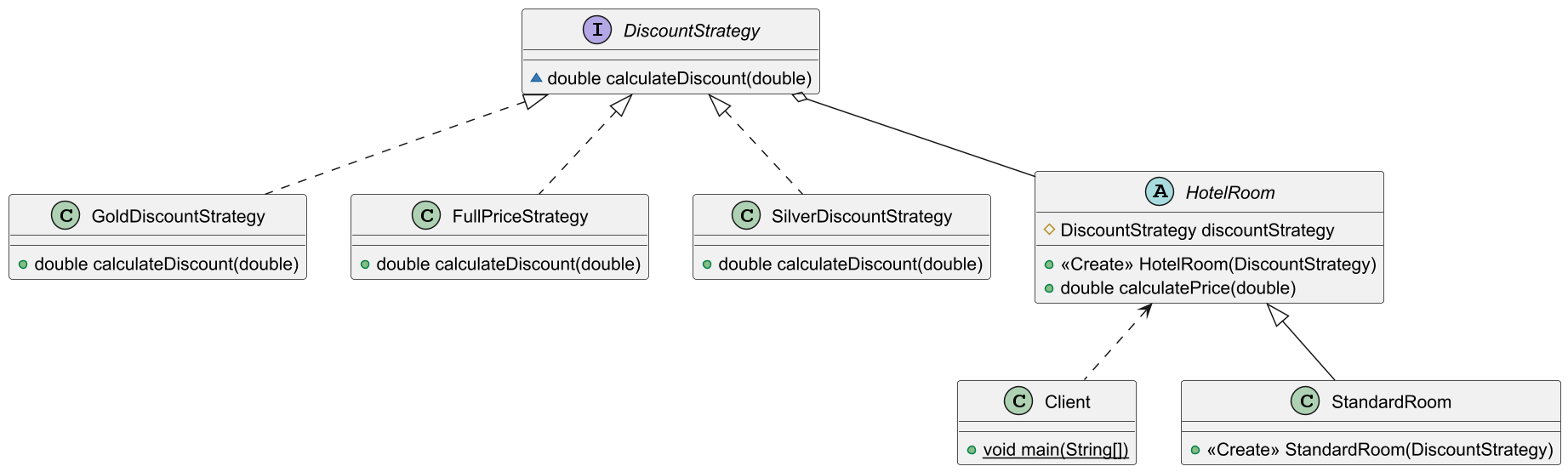
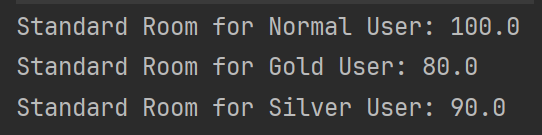
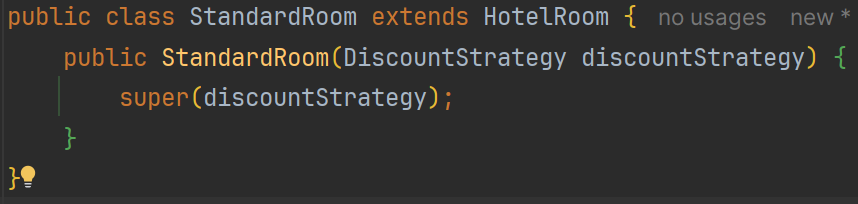
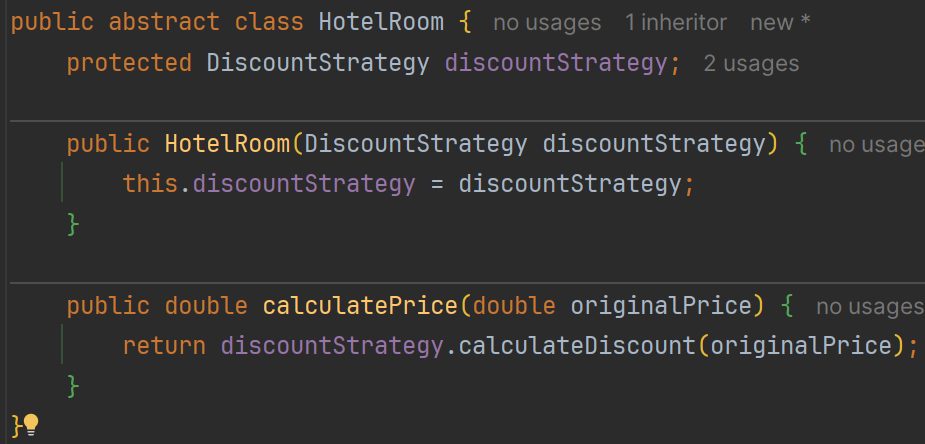
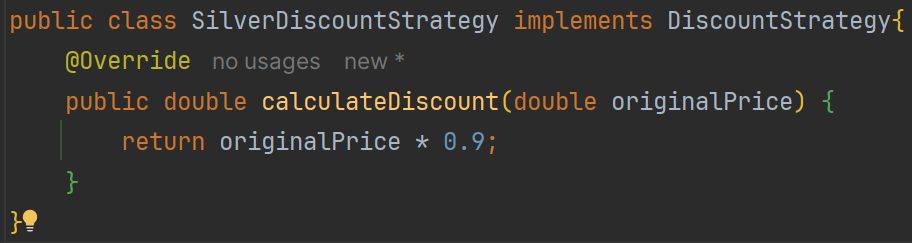
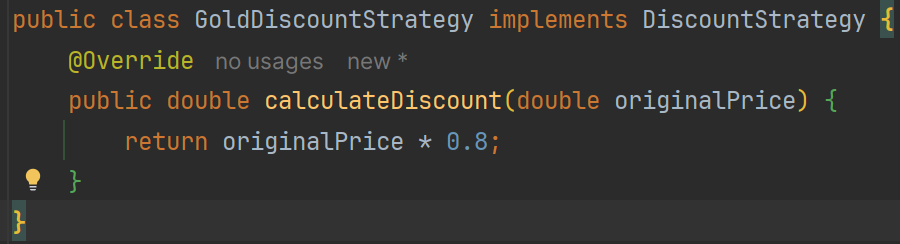
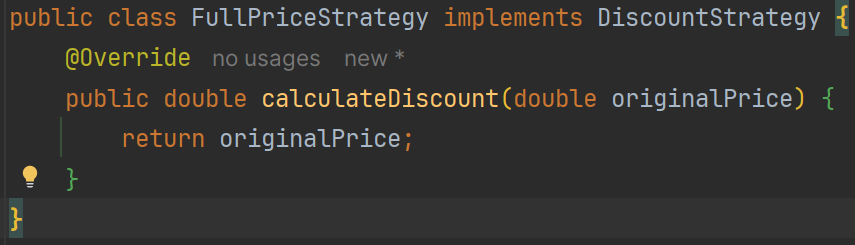
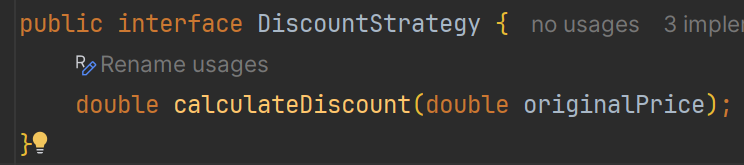
(3)添加两个具体类：老鹰Eagle和企鹅Penguin，并实现相应代码，即构造函数和display抽象方法

(4)使用Client类，测试以上代码



3. 根据以下需求说明，设计符合面向对象设计原则的 UML 类图并实现代码。

假如我们正在开发一个酒店预订系统，针对不同的用户，我们需要计算出不同的房价。比如，普通用户是全价，金卡是 8 折，银卡是 9 折。



# 实验结论

在本次实验中，我深入探索了软件工程工具生成代码的方法，掌握了构件设计方法，并学习了Bridge设计模式。以下是我对这些内容的理解和体会。

首先，通过了解软件工程工具生成代码的方法，我对代码自动生成的过程有了更清晰的认识。过去，我曾将自动生成代码视为一种神秘的“黑科技”，但通过实验，我理解了这一过程实际上是基于一系列预定义的模板和规则。这些工具通过解析需求和设计文档，自动填充模板中的占位符，生成符合特定编程语言规范的代码。这一过程不仅提高了开发效率，还减少了人为错误，提高了代码质量。我认识到，自动生成代码并非不可解释的黑箱操作，而是一种可以被理解和控制的技术。

其次，我掌握了构件设计方法。这种方法强调将软件系统分解为可管理的、可复用的构件。通过实验，我学会了如何识别系统中的关键功能，并将其封装为独立的构件。这样做不仅使得系统结构更加清晰，而且提高了系统的可维护性和可扩展性。我体会到，构件设计方法是一种有效的软件工程实践，它帮助我在设计阶段就考虑到了系统的长远发展。

最后，我学习了Bridge设计模式，并深刻体会到了设计模式的作用。Bridge模式通过将抽象与实现分离，使得两者可以独立变化。在实验中，我通过应用Bridge模式解决了一个实际问题，这让我深刻感受到了设计模式在解决软件设计问题中的强大能力。我认识到，设计模式不仅是一种代码重用的手段，更是一种思考问题和解决问题的方法论。

总结来说，这次实验不仅提升了我的技术能力，也加深了我对软件工程的理解。我学会了如何利用工具和设计模式来提高开发效率和代码质量，同时也认识到了持续学习和实践的重要性。我相信，这些知识和技能将在我的未来工作中发挥重要作用。

仓库地址

https://github.com/Li-Keanu/school-software