Yuhang Zhang
CS-665
Final Project
Date: 12/7/2025

Final Project Report

Assignment Overview

This project demonstrates the Builder design pattern by constructing a structured DailyFitnessPlan, which conceptually represents a common type of day-planning scenario involving two independent components:
(1) a workout schedule and
(2) a nutrition schedule.
The focus of the project is software design abstraction, not domain expertise.
The fitness domain is used purely as a concrete example because it conveniently contains:
a complex product composed of multiple sub-objects,
step-by-step construction requirements,
and multiple variations that follow the same construction process but result in different final outputs.
This makes the domain an ideal fit to demonstrate the intent of the Builder pattern.

Implementation Description

For each assignment, please answer the following:

- - This implementation offers high flexibility because the building process is abstracted within the `DailyPlanBuilder` interface, and different concrete Builders (muscle gain, fat loss, beginner) only need to implement their respective details. If new plan types are needed in the future, such as a powerlifting plan or a recovery day plan, simply adding a new Builder class will suffice, without modifying the Director or existing code; the system structure can be extended naturally.
- - The entire design is structured according to the single responsibility principle, with each class responsible for only one specific function: the Product class stores data, the Builder is responsible for assembling the structure, and the Director controls the steps. Class and method names are intuitive and easy to understand, making the construction process "read like natural language." This clear structure makes the code easy to read, understand, and maintain, even for beginners.
- - This implementation avoids code duplication through a unified build process. All planned creation steps are uniformly executed by the Director calling the Builder interface. The specific Builder is only responsible for the differing parts, avoiding

repetition of the overall process logic. This not only reduces redundancy but also means that future modifications to the build process (such as adding default fields) only require changes in one place, significantly reducing maintenance costs and the probability of errors.

- - This project utilizes the Builder design pattern because it is ideally suited for constructing complex objects composed of multiple parts and having multiple versions (in this case, daily fitness and diet plans). This pattern separates the "construction steps" from the "different results of the construction," allowing the same process to generate different types of plans while maintaining a stable and maintainable structure. Choosing the Builder pattern clearly demonstrates abstraction, extensibility, and software design principles, which aligns perfectly with the course objectives.

GitHub Repository

https://github.com/YuhangZhangz/CS-665_Class_Project

UML