Lab Assignment 5

Task 1:
Relationships Between GoF and GRASP Patterns:

Factory: The Factory pattern is similar to a skilled "Factory" worker dedicated to object creation, just like the "Pure Fabrication" principle in GRASP. This segregation of object creation responsibilities enhances cohesion and maintainability.

Strategy: The Strategy pattern resembles an array of strategies, similar to a toolbox of problem-solving tools. This mirrors the "Polymorphism" principle in GRASP. By defining strategies with a common interface, code flexibility is enriched, allowing dynamic strategy switching.

Composite: Treating a group of objects as a unified entity aligns with the "Polymorphism" concept in GRASP, where individual objects and composite structures comply to the same interface.

Adapter: An Adapter can be envisioned as a universal translator enabling communication among objects with differing "languages" (interfaces). This relates to both "Indirection" and "Polymorphism" in GRASP, bridging incompatible interfaces to promote interoperability.

Facade: The Facade can be likened to a welcoming guide shielding users from complex system intricacies, aligning with "Indirection" and "Protected Variations" in GRASP. It simplifies interactions with intricate subsystems, streamlining user experiences.

Observer: A notification system that informs subscribers of intriguing events corresponds to "Polymorphism" and "Protected Variations" in GRASP. It allows subscribers to respond uniquely to events, fostering flexibility.

Challenges and Solutions:
Task 2:

Scenario 1 - Managing Game State:
When dealing with game state saving/loading, we have a choice to make. The "Board" class knows a lot about the game board, but adding this task might make it less focused. To keep things tidy and focused, we bring in a new class that is like a game-saving expert, a specialized part of the "Board" family. An example name for

that class is: "GameStateManager." This way, the main "Board" can stay great at building the game board.

task 3:

Scenario 2 - Networked Gameplay:
In online gaming, we want to show the same error messages on different systems. To do this, we use "Polymorphism," "Protected Variations," and the "Strategy" pattern.

We make a network tool that figures out what system (OS) you're using, thanks to "Protected Variations." Then, we use the "Strategy" pattern to send the right error message for that system. This makes our code flexible and easy to adapt.

These design ideas and rules make our code simpler, more flexible, and organized. The "Strategy" pattern is a versatile tool for solving complex problems.