9.1P: In Person Check-in 3 – Answer Sheet

1. What was the most challenging aspect of the case study tasks? Why?

From my perspective, during several iteration phases in the SwinAdventure case study, the most challenging aspects were drawing sequence diagrams and understanding complex code and how it works. In the early weeks, the iteration descriptions were quite easy and detailed, while subsequent iterations became shorter and more complex. To be honest, understanding UML diagrams, sequence diagrams, and the code itself were the most challenging parts of the case study.

2. What is the most valuable thing you have learned in this unit so far?

In my opinion, the most valuable thing I've learned in this unit so far is logical thinking to solve complex problems and how to write good, structured, and clean code. Interestingly, during this course, I realized that object-oriented programming requires students to understand the code they are writing and make it clean. In summary, the vaulable thing I have learned is the logical thinking and how to make code clean and easy understand.

3. What are some strategies for success you can start or continue using for the remainder of the semester and in future units?

From my point of view, I frequently utilize powerful applications and tools including AI Assisstant and Trello, analyzing, guilding me through several challenge questions and time management effective for the remainder of the semester. In the future courses, I will try to latest tools to improve not only my academic performance but also in my future career

4. Have you heard of design patterns? Did you use any design patterns in the Swin Adventure case study? If yes, please explain where they were used.

To be honest, I haven’t heard “design patterns” but after researching, finally I knnow that “design patterns” reusable blueprints for solving common software design problems. In the Swin Adventure case study, two notable patterns were used. The **Composite Pattern** is evident through the IHaveInventory interface, allowing Player, Location, and Bag to be treated uniformly when locating items. This simplifies the LookCommand's ability to search within different containers. The **Strategy Pattern** is used with the Command abstract class and its implementations like LookCommand, where each command encapsulates its specific execution logic. This makes adding new commands easy without altering the core game structure.