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

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

   The toughest part of the SwinAdventure case study was deciding and putting in place the command parser framework in those later iterations, specifically 6.1P_Iteration4 and 7.1P_Iteration5. It was difficult because I had to devise a mechanism that was able to parse natural language commands but at the same time be flexible in different game objects. So, as the game expanded from having just those simple things in 2.4P_Iteration1 to having containers in 4.2P_Iteration2, then places in 7.1P_Iteration5, and locations in 7.2C_Iteration6, the command system needed to become much more refined. Every command needed to cope with the larger and larger combination of objects using the IHaveInventory interface, and I had to carefully consider how commands would manage multiple object references, search through nested containers, and provide useful error messages. Balancing the need for flexibility for future additions with maintaining clean, concise code required me to fully commit object-oriented design principles throughout every iteration of the project.

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

   The most valuable thing I've learned in this unit is how to effectively apply object-oriented design principles to create flexible, maintainable software systems. Through the progressive iterations of SwinAdventure (from 2.4P_Iteration1 through 7.2C_Iteration6), I've gained practical experience in structuring code through abstraction, inheritance, and polymorphism. Starting with simple classes and gradually building a complex system taught me how thoughtful design decisions early on can make future extensions much easier. The implementation of the Command pattern in 6.1P_Iteration4 particularly demonstrated how proper OOP design allows for adding new functionality without disrupting existing code, a principle I saw in action when adding new commands and game elements in later iterations.

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

   I plan to continue applying several effective strategies that proved valuable during my SwinAdventure project. I'll maintain the incremental development approach demonstrated across iterations 2.4P through 7.2C, building small, functional components before expanding. I'll continue implementing test-driven development, creating unit tests before implementing features to catch issues early. Planning ahead by sketching class diagrams before coding will help me visualize object relationships as it did in iterations like 5.2P and 7.1P. Finally, I'll regularly review previous work to identify reusable patterns, just as my command structure in 6.1P_Iteration4 built upon earlier foundations. These strategies will help me tackle complex programming challenges while maintaining code quality in future units.

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.

Yes, I've heard of design patterns, and I think I implemented two in my SwinAdventure case study:

The most prominent design pattern I used was the Command Pattern in 6.1P_Iteration4. I created an abstract Command class with an Execute method, then implemented concrete commands like LookCommand that encapsulated specific actions. This pattern allowed me to separate the invoker (game loop) from the receiver (game objects), making it easy to add new commands in next iteration (like MoveCommand) without modifying existing code.

I also implemented the Composite Pattern starting in 4.2P_Iteration2 with the IHaveInventory interface. This allowed me to treat individual objects and containers (like Player, Bag, and later Location in 7.1P_Iteration5) through the Locate method. This pattern enabled recursive searching through nested containers, allowing commands to work consistently regardless of where items were located.