School of Science, Computing and Engineering Technologies

# Object Oriented Programming

Credit Task 7.2: Case Study — Iteration 6: Locations

## Overview

Object oriented programming makes best sense with larger programs. The case study will be your opportunity to create a larger program and better see how these abstractions make it easier to create software solutions.

|  |  |
|---|---|
| **Purpose:** | Demonstrate how to use the classes created in the case study to make a command like program, then design the first extension. |
| **Task:** | Understand the case study program and implement iteration 6. |
| **Time:** | **End of Week 9, Friday, 4 July 2025, 23:59:00 (Firmed)** |

### *Submission Details*

All students have access to the Adobe Acrobat tools. Please print your solution to PDF and combine it with the screenshots taken for this task.

- Program source code
- Test source code
- Screenshot of unit tests passing
- Screenshot of program running
- UML class diagram showing what needs to be added
- UML sequence diagram to explain how Locate works in the Player

## Instructions

1. Review the **Case Study Requirements** document. It outlines what you need to create.
2. For this task aim to complete Iteration 6.

Once your tests are working correctly get a screenshot of the tests passing and submit them along with the code. To get it signed off as complete you will need to demonstrate it working to your tutor either during your lab or at the help desk.

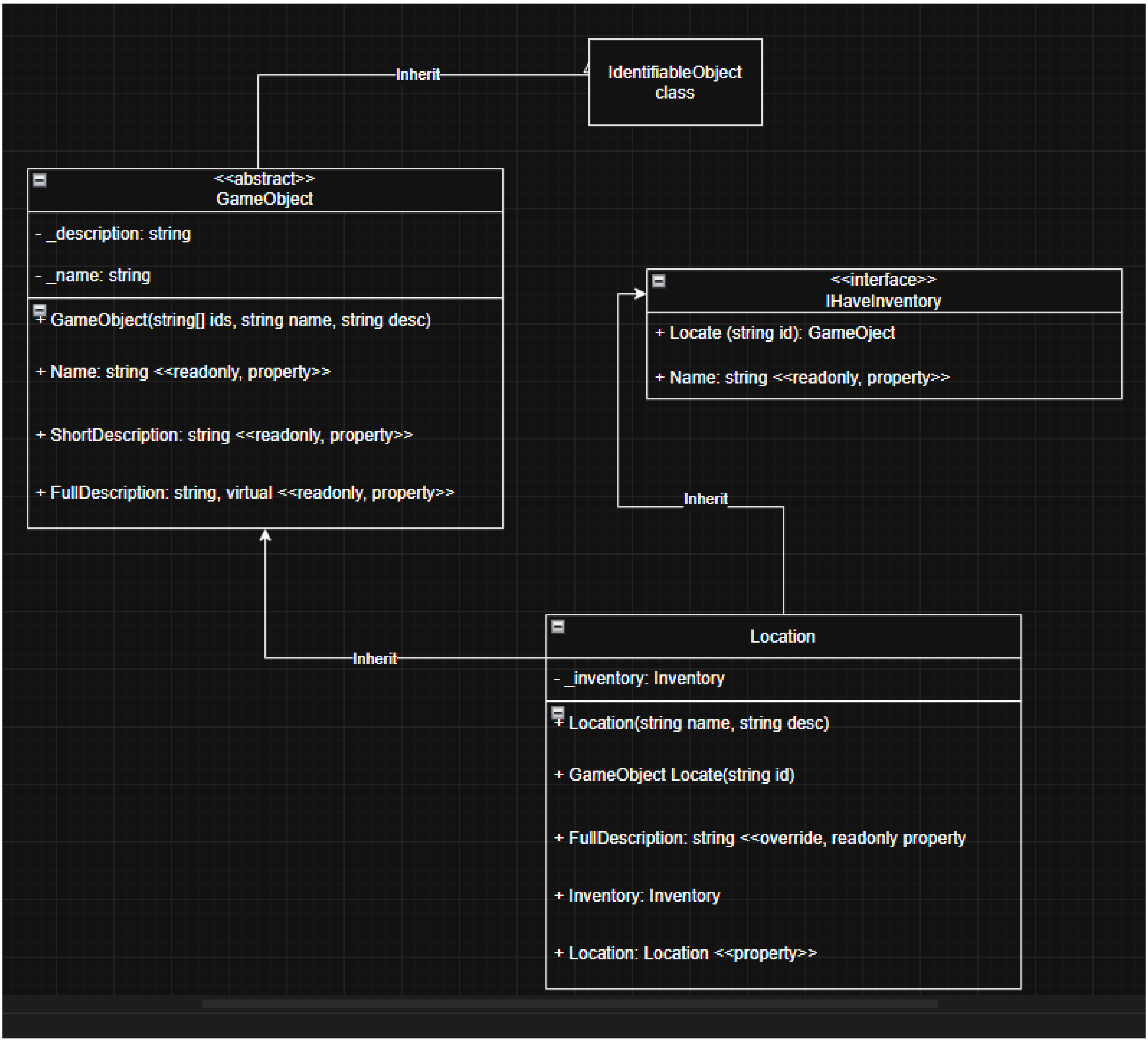
**Note**: Once you have made a submission, this task requires you to have a discussion with your tutor in your lab or at the help desk before it can be signed off as Complete.

### *Assessment Criteria*

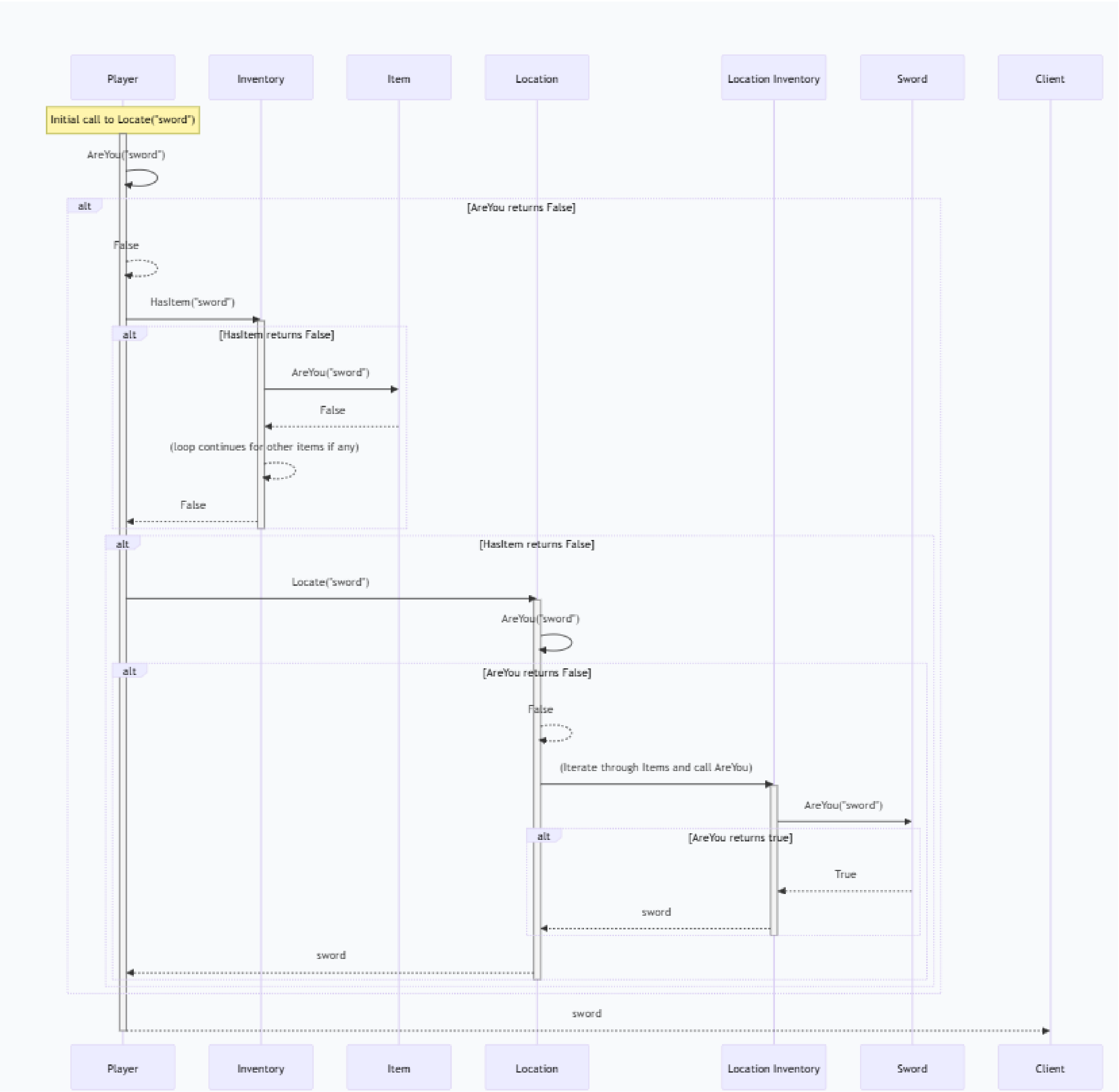
Make sure that your task has the following in your submission:

- The program is implemented correctly based on the case study description.
- The “Universal Task Requirements” (see Canvas) have been met.

## A. UML Class Diagram


Inherit
IdentifiableObject class
<<abstract>>
GameObject
- _description: string
- _name: string
+ GameObject(string[] ids, string name, string desc)
+ Name: string <<readonly, property>>
+ ShortDescription: string <<readonly, property>>
+ FullDescription: string, virtual <<readonly, property>>
<<interface>>
IHaveInventory
+ Locate (string id): GameOject
+ Name: string <<readonly, property>>
Inherit
Inherit
Location
- _inventory: Inventory
+ Location(string name, string desc)
+ GameObject Locate(string id)
+ FullDescription: string <<override, readonly property
+ Inventory: Inventory
+ Location: Location <<property>>

## B. UML Sequence Diagram and explaination



Description:

- Command Execution: The Program sends your "look at sword" command to the LookCommand.
- LookCommand Checks Player: The LookCommand first asks the Player to Locate the "sword."
- Player's Search:
    - The Player checks if they themselves are the "sword" (they aren't).
    - Then, the Player checks their own Inventory to see if it HasItem("sword"). The Inventory iterates through its items, asking each "AreYou 'sword'?"
    - Assuming the sword isn't in the player's inventory, the Player.Locate returns null to the LookCommand.
- LookCommand Checks Location: Since the player didn't have the sword, the LookCommand then asks the Player's current Location to Locate the "sword."
- Location's Search:
    - The Location first checks if it is the "sword" (it isn't).
    - Then, the Location iterates directly through the items in its own Inventory (items on the ground). It asks each item "AreYou 'sword'?"
    - Upon finding the "sword," the Location returns the sword:Item object to the LookCommand.
- Description Display: The LookCommand receives the sword:Item object and asks it for its FullDescription. This

description is then displayed to you.