School of Science, Computing and Engineering Technologies

Object Oriented Programming

Week 9: SwinAdventure — Iteration 6

Overview

In this week, there are two tasks 9.1 and 9.2. Each task contributes 2% to your final grade. Noting that you need to complete this task before coming to your allocated lab. In the lab, there will be a verification task and short interview to verify your understanding.

These tasks extend from your current SwinAdventure application that you created in Week 8.

Purposes

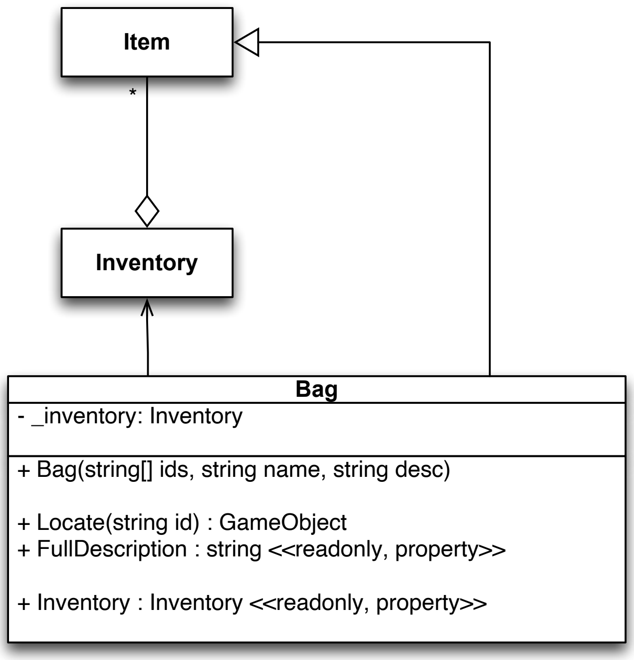
Task 9.1	This task aims to develop a Bag class for the SwinAdventure game. You will then create unit test cases for this class.
Task 9.2	This task aims to create a C# interface called I Have Inventory interface/ protocol. Objects that implement this interface must be able to perform certain tasks.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Instructions - Task 9.1

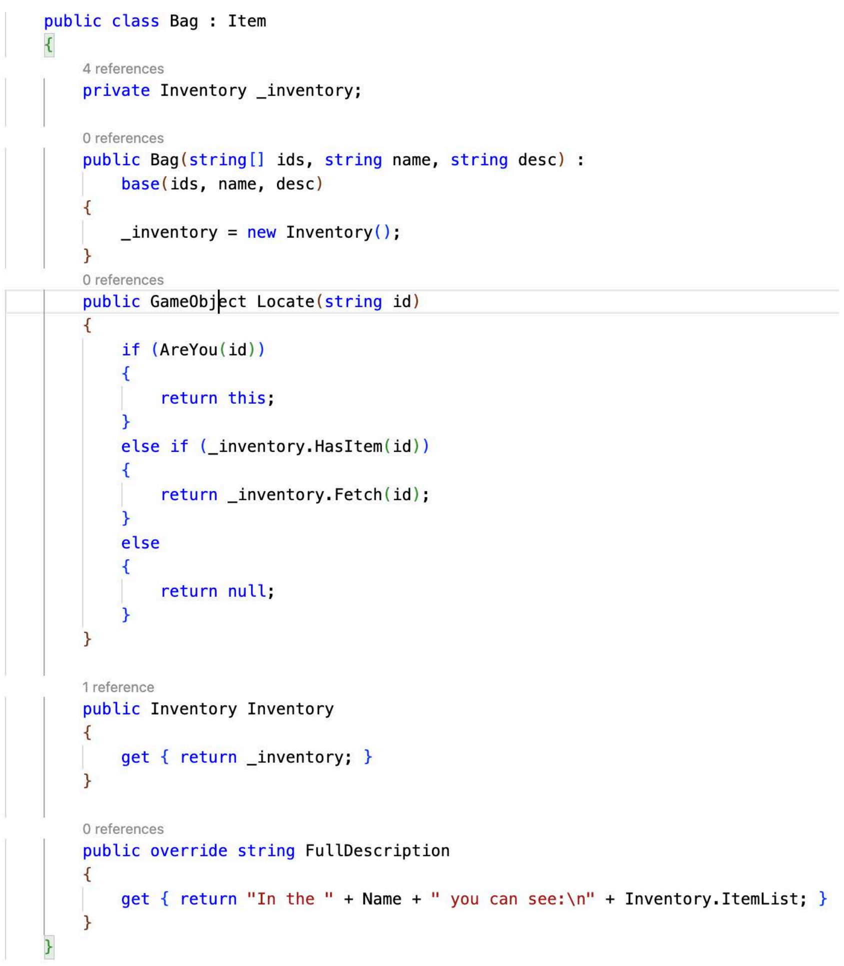
1. Review the ***Case Study Requirements*** document. It outlines what you need to create.

2. In this iteration you will add a Bag class to make it possible to have items that contain other items.
3. The Bag abstraction is a special kind of item, one that contains other items in its own Inventory. This is a version of **the composite pattern**, which allows flexible arrangements of bags and items, for example a bag to contain another bag.
4. We provide you with an UML class diagram for the Bag class as follows.


Item
*
Inventory
Bag
- _inventory: Inventory
+ Bag(string[] ids, string name, string desc)
+ Locate(string id) : GameObject
+ FullDescription : string <<readonly, property>>
+ Inventory : Inventory <<readonly, property>>

5.The Locate method in the Bag class allows to locate itself it the input id matches to one of its identifier. Alternatively, the method can locate matched items from its inventory.

6. The below sample code demonstrates how to implement this class.

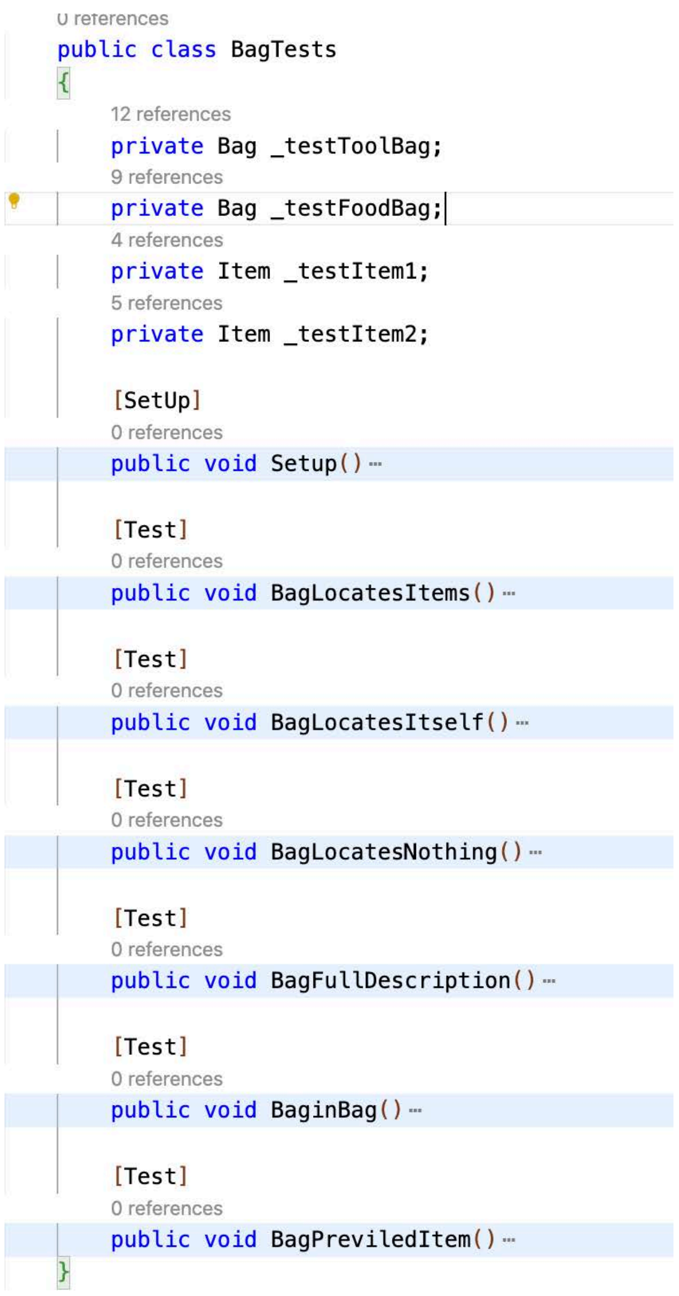


```
public class Bag : Item
{
    4 references
    private Inventory _inventory;

    0 references
    public Bag(string[] ids, string name, string desc) :
        base(ids, name, desc)
    {
        _inventory = new Inventory();
    }
    0 references
    public GameObject Locate(string id)
    {
        if (AreYou(id))
        {
            return this;
        }
        else if (_inventory.HasItem(id))
        {
            return _inventory.Fetch(id);
        }
        else
        {
            return null;
        }
    }

    1 reference
    public Inventory Inventory
    {
        get { return _inventory; }
    }

    0 references
    public override string FullDescription
    {
        get { return "In the " + Name + " you can see:\n" + Inventory.ItemList; }
    }
}
```

Then, please develop following test cases.

Bag Unit Tests	
Test Bag Locates Items	You can add items to the Bag, and locate the items in its inventory, this returns items the bag has and the item remains in the bag's inventory.
Test Bag Locates itself	The bag returns itself if asked to locate one of its identifiers.
Test Bag Locates nothing	The bag returns a null/nil object if asked to locate something it does not have.
Test Bag Full Description	The bag's full description contains the text "In the <name> you can see:" and the short descriptions of the items the bag contains (from its inventory's item list)
Test Bag in Bag	Create two Bag objects (b1, b2), put b2 in b1's inventory. Test that b1 can locate b2. Test that b1 can locate other items in b1. Test that b1 **can not** locate items in b2.
Test Bag in Bag with a privileged item	Create two Bag objects (b1,b2), put b2 in b1's inventory. Put a privileged item into b2 by using PrivilegeEscalation method. Then, please test that b1 **can not** locate the privileged item in b2.

7. The source code template for the BagTest class is as below.

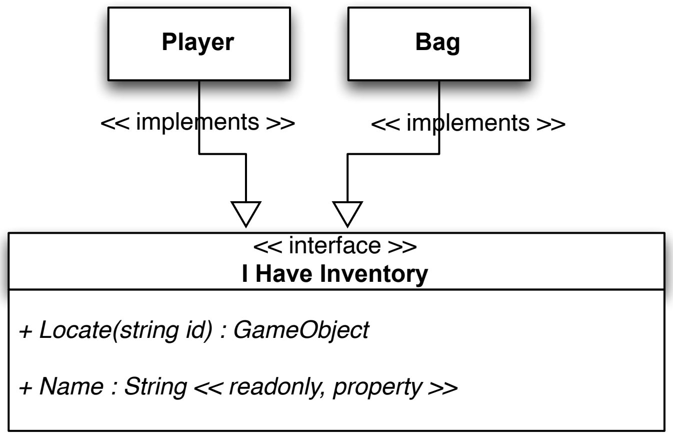


```
0 references
public class BagTests
{
    12 references
    private Bag _testToolBag;
    9 references
    private Bag _testFoodBag;
    4 references
    private Item _testItem1;
    5 references
    private Item _testItem2;

    [SetUp]
    0 references
    public void Setup()…

    [Test]
    0 references
    public void BagLocatesItems()…

    [Test]
    0 references
    public void BagLocatesItself()…

    [Test]
    0 references
    public void BagLocatesNothing()…

    [Test]
    0 references
    public void BagFullDescription()…

    [Test]
    0 references
    public void BaginBag()…

    [Test]
    0 references
    public void BagPreviledItem()…
}
```

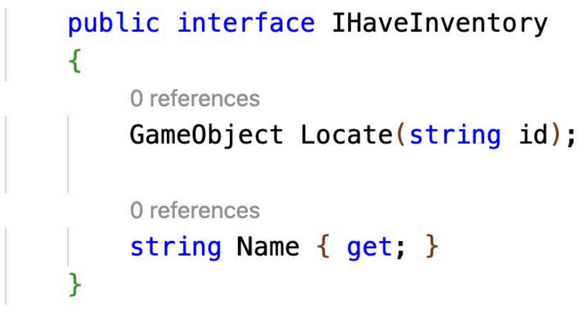
Instructions - Task 9.2

1. Review the ***Case Study Requirements*** document. It outlines what you need to create.

2. In this task, we will create an Interface named IHaveInventory. By using this interace, whichever objects that implement this interface will be treated in the same way: as objects that container other items. Thanks to polymorphism. This is one of the key principles in OOP.
3. Specifically, we will modify the class Player and Bag to implement this interface.
4. We provide you with an UML class diagram for this interface as follows.


Player
Bag
<< implements >>
<< implements >>
<< interface >>
I Have Inventory
+ Locate(string id) : GameObject
+ Name : String << readonly, property >>

5. Please implement the above interface using the following sample source code.



```
public interface IHaveInventory
{
    0 references
    GameObject Locate(string id);

    0 references
    string Name { get; }
}
```

6. The classes Player and Bag then needed to be updated accordingly with the following code
   - public class Bag : Item, IHaveInventory
   - public class Player : GameObject, IHaveInventory

7. Review your entire SwinAdventure project implementation to ensure you understand how these classes collaborate to each other.

8. In the file Program.cs of your SwinAdventure, please create a container to exercise polymorphism

```
List<IHaveInventory> myContainers = new List<IHaveInventory>();

//define a player object and add this object into the list myContainers
Player _testPlayer;
_testPlayer = new Player("James", "an explorer");

myContainers.Add(_testPlayer);

//define a bag object and an item, then add the item into the inventory of the bag.
Bag _testToolBag;
_testToolBag = new Bag(new string[] { "bag", "tool" }, "Tools Bag", "A bag that contains tools");
Item _testItem2;
_testItem2 = new Item(new string[] { "stew", "beef" }, "A Beef Stew", "A hearty beef stew");

_testToolBag.Inventory.Put(_testItem2);
//add the bag into the list myContainers
myContainers.Add(_testToolBag);
```

9. Then, please create a for-loop to print out the information of the player and the bag objects.

Hint. You need to convert an object in the list to the correct type before printing out the information.