

COS20007: Object Oriented Programming

Pass Task 5.2: Case Study — Iteration 3: Bags

Show Wai Yan/105293041

Bag.cs

```
namespace SwinAdventure
{
    public class Bag : Item
    {
        // Fields
        private Inventory _inventory;

        // Constructor
        public Bag(string[] ids, string name, string desc) : base(ids, name, desc)
        {
            _inventory = new Inventory();
        }

        // Methods
        public GameObject? Locate(string id)
        {
            if (AreYou(id)) return this;
            return _inventory.Fetch(id);
        }

        // Properties
        public override string FullDescription
        {
            get { return $"{base.FullDescription}.\nYou look in the {Name.ToLower()} and see:\n{_inventory.ItemList}"; }
        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }
    }
}
```

TestBag.cs

```
using System;
using SwinAdventure;
using NUnit.Framework;
using NUnit.Framework.Legacy;

namespace UnitTests
{
    [TestFixture]
    public class TestBag
    {
        private Bag testBag;
        private Item sword;
        private Item potion;
        private Item key;

        [SetUp]
        public void Setup()
        {
            testBag = new Bag(new string[] { "bag", "backpack" }, "Leather Bag", "A sturdy leather bag to carry items");

            sword = new Item(new string[] { "sword", "weapon" }, "Steel Sword", "A sharp steel sword");
            potion = new Item(new string[] { "potion", "health" }, "Health Potion", "A red potion that restores health");
            key = new Item(new string[] { "key", "golden" }, "Golden Key", "A shiny golden key");

            testBag.Inventory.Put(sword);
            testBag.Inventory.Put(potion);
            testBag.Inventory.Put(key);
        }

        [Test]
        public void TestBagLocatesItems()
        {
            ClassicAssert.That(testBag.Locate("key"), Is.EqualTo(key));
            ClassicAssert.That(testBag.Locate("weapon"), Is.EqualTo(sword));
            ClassicAssert.That(testBag.Locate("potion"), Is.EqualTo(potion));
        }

        [Test]
        public void TestBagLocatesItself()
        {
            ClassicAssert.That(testBag.Locate("backpack"), Is.EqualTo(testBag));
        }

        [Test]
        public void TestBagLocatesNothing()
        {
        }
    }
}
```

```

    {
        ClassicAssert.That(testBag.Locate("Shield"), Is.EqualTo(null));
    }

[Test]
public void TestBagFullDescription()
{
    string description = $"A sturdy leather bag to carry items.\nYou look in the leather bag and
see:\n\t{sword.ShortDescription}\n\t{potion.ShortDescription}\n\t{key.ShortDescription}\n";
    ClassicAssert.That(testBag.FullDescription, Is.EqualTo(description));
}

[Test]
public void TestBagInBag()
{
    Bag b1 = testBag;
    Bag b2 = new Bag(new string[] { "pouch", "small bag" }, "Small Pouch", "A small cloth pouch for tiny items");
    Item coin = new Item(new string[] { "coin", "gold" }, "Gold Coin", "A shiny gold coin");
    Item gem = new Item(new string[] { "gem", "ruby" }, "Ruby", "A sparkling red gemstone");

    b2.Inventory.Put(coin);
    b2.Inventory.Put(gem);

    b1.Inventory.Put(b2);

    // Locate b2
    ClassicAssert.That(b1.Locate("small bag"), Is.EqualTo(b2));

    //Locate other item in b1
    ClassicAssert.That(b1.Locate("sword"), Is.EqualTo(sword));
    ClassicAssert.That(b1.Locate("potion"), Is.EqualTo(potion));

    // Locate other item in b2 from b1
    ClassicAssert.That(b1.Locate("coin"), Is.EqualTo(null));
    ClassicAssert.That(b1.Locate("gem"), Is.EqualTo(null));
}

[Test]
public void TestBagInBagWithPrivileged()
{
    Bag b1 = testBag;
    Bag b2 = new Bag(new string[] { "pouch", "small bag" }, "Small Pouch", "A small cloth pouch for tiny items");
    Item privilegedItem = new Item(new string[] { "important item", "privileged item" }, "Id", "A very important item");

    // Using PrivilegedEscalation()
    privilegedItem.PrivilegeEscalation("3041");
    ClassicAssert.That(privilegedItem.FirstId, Is.EqualTo("105293041"));

    b2.Inventory.Put(privilegedItem);

    b1.Inventory.Put(b2);

    // Testing Privileged Item
    ClassicAssert.That(b1.Locate(privilegedItem.FirstId), Is.EqualTo(null));
}
}
}

```

Screenshot of the Test Explorer showing your unit test running

The screenshot displays the Visual Studio IDE with the Test Explorer on the right and the Test Runner at the bottom. The main editor shows the `TestBag.cs` file with a unit test `TestBagInBagWithPrivileged` running. The test is currently failing, as indicated by the red 'X' icon next to the test name in the Test Explorer.

Test Explorer (Right Panel):

- Filter: (e.g. test, exclude, @tag)
- 27/27 tests, 1.2s
- SwInAdventure.Tests (net9.0) 2.0ms
 - UnitTests 2.0ms
 - TestBag 2.0ms
 - TestBagLocatesItems 0.0ms
 - TestBagLocatesItself 0.0ms
 - TestBagLocatesNothing 0.0ms
 - TestBagFullDescription 2.0ms
 - TestBagInBag 0.0ms
 - TestBagInBagWithPrivileged 0.0ms
 - TestIndentifiableObject 0.0ms
 - TestInventory 0.0ms
 - TestItem 0.0ms
 - TestPlayer 0.0ms

Test Runner (Bottom Panel):

SwInAdventure.Tests -> /Users/showmian/Dev/COS20007/PassTask/5.2P_iteration3/SwInAdventure.Tests/bin/Debug/net9.0/SwInAdventure.Tests.dll

Executing task: dotnet: build /Users/showmian/Dev/COS20007/PassTask/5.2P_iteration3/SwInAdventure.Tests/SwInAdventure.Tests.csproj

dotnet build /Users/showmian/Dev/COS20007/PassTask/5.2P_iteration3/SwInAdventure.Tests/SwInAdventure.Tests.csproj /property:GenerateFullPaths:true /consoleLoggerParameters:NoSummary /p:Configuration=Debug /p:Platform=AnyCPU

Determining projects to restore...

All projects are up-to-date for restore.

SwInAdventure -> /Users/showmian/Dev/COS20007/PassTask/5.2P_iteration3/SwInAdventure/bin/Debug/net9.0/SwInAdventure.dll

SwInAdventure.Tests -> /Users/showmian/Dev/COS20007/PassTask/5.2P_iteration3/SwInAdventure.Tests/bin/Debug/net9.0/SwInAdventure.Tests.dll

Terminal will be reused by tasks, press any key to close it.