Bag code

```
using MiNET.Utils.Skins;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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

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

        public override string GetFullDescription()
        {

            return $" You are {Name}, a brave adventurer. You are holding" + _inventory.ItemList;
        }
        public Inventory Inventory
        {
            get { return _inventory; }
        }

        public GameObject Locate(string id)
        {
```

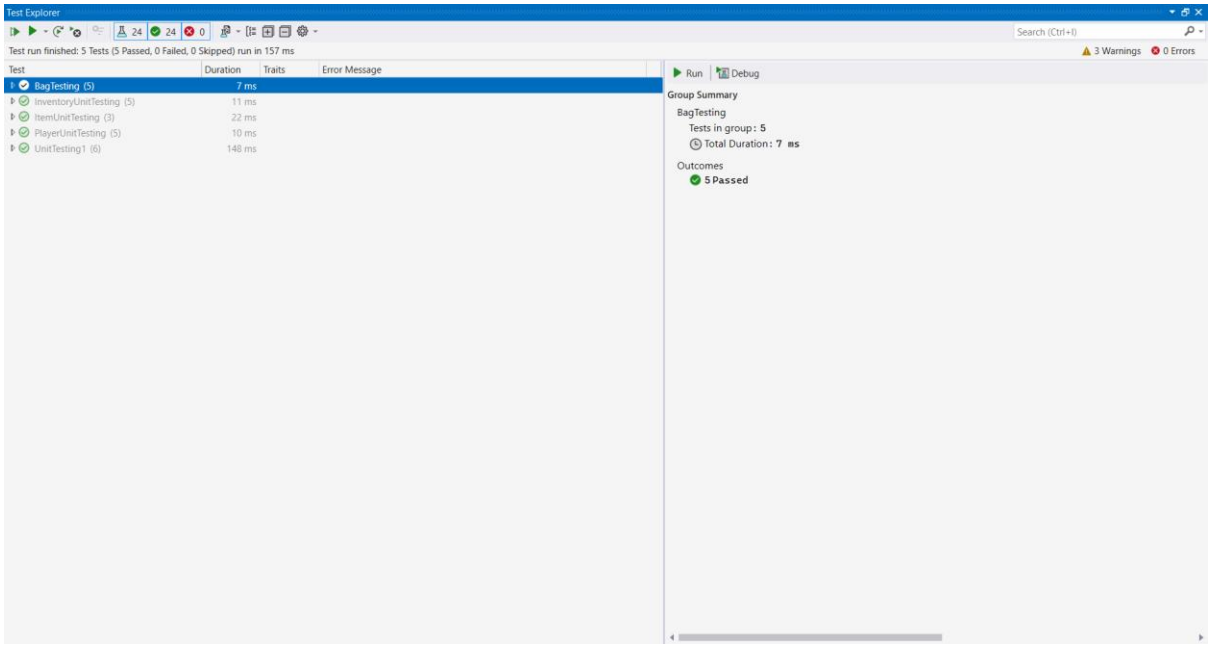
```
            if (AreYou(id))
            {
                return this;
            }
            return _inventory.Fetch(id);
        }
    }

}
```

Bag test code

```
using NUnit.Framework;
using SwinAdventure;

namespace BagTesting
{
    public class BagTesting
    {
        private Bag _bag;
        private Item _item1;
        private Item _item2;
        private Bag _bag2;

        [SetUp]
        public void Setup()
        {
            _bag = new Bag(new string[] { "bag1" }, "Bag 1", "A bag for items");
            _item1 = new Item(new string[] { "item1" }, "Item 1", "An item");
            _item2 = new Item(new string[] { "item2" }, "Item 2", "Another item");
            _bag2 = new Bag(new string[] { "bag2" }, "Bag 2", "Another bag");
        }
```

```
[Test]
public void TestBagLocatesItems()
{
   // Add an item to the bag and verify it's located in the inventory
   _bag.Inventory.Put(_item1);
   Assert.AreEqual(_item1, _bag.Inventory.Fetch("item1"));
}

[Test]
public void TestBagLocatesItself()
{
   // Ensure the bag can locate itself by its identifier
   Assert.AreEqual(_bag, _bag.Locate("bag1"));
}

[Test]
public void TestBagLocatesNothing()
{
   // Ensure the bag returns null for a non-existent item
   Assert.IsNull(_bag.Locate("nonexistent"));
}

[Test]
public void TestBagFullDescription()
{
   // Add an item and check if the full description contains the item
   _bag.Inventory.Put(_item1);
   Assert.IsTrue(_bag.GetFullDescription().Contains("item1"));
}
```

```
    [Test]
    public void TestBagInBag()
    {
        // Add bag2 to bag1's inventory
        _bag.Inventory.Put(_bag2);

        // Verify that bag1 can locate bag2
        Assert.AreEqual(_bag2, _bag.Locate("bag2"));

        // Verify that bag1 can locate items within itself but not inside bag2
        _bag.Inventory.Put(_item1);
        _bag2.Inventory.Put(_item2);

        Assert.AreEqual(_item1, _bag.Locate("item1"));
        Assert.IsNull(_bag.Locate("item2"));
    }

    public void TestPutItemInInventory()
    {
        Item item1 = new Item(new string[] { "item1" }, "Item 1", "A test item");
        _bag.Inventory.Put(item1);

        // Check if the item was added
        Assert.IsTrue(_bag.Inventory.HasItem("item1"));
    }
  }
}
```


Test Explorer
24
24
0
Search (Ctrl+I)
Test run finished: 5 Tests (5 Passed, 0 Failed, 0 Skipped) run in 157 ms
3 Warnings
0 Errors
Test
Duration
Traits
Error Message
BagTesting (5)
7 ms
InventoryUnitTesting (5)
11 ms
ItemUnitTesting (3)
22 ms
PlayerUnitTesting (5)
10 ms
UnitTesting1 (6)
148 ms
Run
Debug
Group Summary
BagTesting
Tests in group: 5
Total Duration: 7 ms
Outcomes
5 Passed