

5.2. Case Iteration 3: Bag

I've fixed the mistake in inventory full description function

Inventory File

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class Inventory
    {
        private List<Item> _items = new List<Item>();

        public Inventory()
        {
        }

        public string ItemList
        {
            get
            {
                string list = "";
                foreach (Item item in _items)
                {
                    list += "\t " + item.ShortDescription + "\n";
                }
                return list;
            }
        }

        public bool HasItem(string id)
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id))
                {
                    return true;
                }
            }
        }
    }
}
```

```

    }
    return false;
}

public void Put(Item itm)
{
    _items.Add(itm);
}

public Item? Take(string id)
{
    foreach (Item item in _items)
    {
        if (item.AreYou(id))
        {
            _items.Remove(item);
            return item;
        }
    }
    return null;
}

public Item? Fetch(string id)
{
    foreach (Item item in _items)
    {
        if (item.AreYou(id))
        {
            return item;
        }
    }
    return null;
}
}
}

```

Bag File

```

using System;
using System.Collections.Generic;

```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static SwinAdventure.LookCommand;

namespace SwinAdventure
{
    public class Bag : Item, IHaveInventory
    {
        private Inventory _inventory = new Inventory();

        public Bag(string[] idents, string name, string desc) : base(idents,
name, desc)
        {
        }

        public Item? Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            else
            {
                return _inventory.Fetch(id);
            }
        }

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

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

```

    }

    GameObject? IHaveInventory.Locate(string id)
    {
        return Locate(id);
    }

    string IHaveInventory.Name
    {
        get
        {
            return Name;
        }
    }
}

```

Unit Test File

```

using SwinAdventure;

namespace TestQueue
{
    public class Tests
    {
        Item item1 = new Item(new string[] { "sword" }, "sword", "a sword");
        Item item2 = new Item(new string[] { "shield" }, "shield", "a shield");
        Item item3 = new Item(new string[] { "shiba" }, "shiba", "a shiba");
        Item item4 = new Item(new string[] { "gem" }, "gem", "a gem");

        [SetUp]
        public void Setup()
        {
        }

        // Test the Item class
        [Test]
        public void ItemIdentifiable()
        {

```

```
    Assert.IsTrue(item1.AreYou("sword"));
}

[Test]
public void ShortDescription()
{
    Assert.That(item1.ShortDescription, Is.EqualTo("sword (sword)"));
}

[Test]
public void FullDescription()
{
    Assert.That(item1.FullDescription, Is.EqualTo("a sword"));
}

// Test the Inventory class
[Test]
public void FindItem()
{
    Inventory inventory = new Inventory();
    inventory.Put(item1);

    Assert.IsTrue(inventory.HasItem("sword"));
}

[Test]
public void NoItem()
{
    Inventory inventory = new Inventory();
    Assert.IsFalse(inventory.HasItem("sword"));
}

[Test]
public void FetchItem()
{
    Inventory inventory = new Inventory();
    inventory.Put(item1);

    Assert.That(item1, Is.EqualTo(inventory.Fetch("sword")));
    Assert.IsTrue(inventory.HasItem("sword"));
}
```

```

    }

[Test]
public void TakeItem()
{
    Inventory inventory = new Inventory();
    inventory.Put(item1);

    Assert.That(item1, Is.EqualTo(inventory.Take("sword")));
    Assert.IsFalse(inventory.HasItem("sword"));
}

[Test]
public void ItemList()
{
    Inventory inventory = new Inventory();
    inventory.Put(item1);
    inventory.Put(item2);

    //the list string below is the expected output, consisting of every
    item in the following format: name ( first id)
    Assert.That(inventory.ItemList, Is.EqualTo("\t sword (sword)\n\t
    shield (shield)\n"));
}

// Test the Player class
[Test]
public void PlayerIdentifiable()
{
    Player player = new Player("Tan", "A player");

    Assert.IsTrue(player.AreYou("me"));
    Assert.IsTrue(player.AreYou("inventory"));
}

[Test]
public void PlayerLocate()

```

```

{
    Player player = new Player("Tan", "A player");
    player.Inventory.Put(item1);

    Assert.That(item1, Is.EqualTo(player.Locate("sword")));
}

[Test]
public void PlayerLocateItself()
{
    Player player = new Player("Tan", "A player");
    Assert.That(player, Is.EqualTo(player.Locate("me")));
    Assert.That(player, Is.EqualTo(player.Locate("inventory")));
}

[Test]
public void PlayerLocateNothing()
{
    Player player = new Player("Tan", "A player");
    Assert.That(player.Locate("sword"), Is.Null);
}

[Test]
public void PlayerFullDescription()
{
    Player player = new Player("Tan", "A player");
    player.Inventory.Put(item1);
    player.Inventory.Put(item2);

    //the list string below is the expected output, consisting of every
    item in the following format: name ( first id)
    Assert.That(player.FullDescription, Is.EqualTo("You are Tan A
    player\nYou are carrying:\n\t sword (sword)\n\t shield (shield)\n"));
}

//Test the Bag class
[Test]
public void BagLocate()
{

```

```

        Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a
backpack");
        backpack.Inventory.Put(item1);
        backpack.Inventory.Put(item2);
        backpack.Inventory.Put(item3);

        //ask to return item and item stays in backpack
        Assert.That(item3, Is.EqualTo(backpack.Locate("shiba")));
        Assert.IsTrue(backpack.Inventory.HasItem("shiba"));

    }

    [Test]
    public void BagLocatesItself()
    {
        Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a
backpack");
        Assert.That(backpack, Is.EqualTo(backpack.Locate("backpack")));
    }

    [Test]
    public void BagLocateNothing()
    {
        Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a
backpack");
        Assert.That(backpack.Locate("sword"), Is.Null);
    }

    [Test]
    public void BagFullDescription()
    {
        Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a
backpack");
        backpack.Inventory.Put(item1);
        backpack.Inventory.Put(item2);
        backpack.Inventory.Put(item3);

        //the list string below is the expected output, consisting of every
item in the following format: name ( first id)

```



```
        Assert.That(backpack.FullDescription, Is.EqualTo("In the backpack  
you can see:\n\t sword (sword)\n\t shield (shield)\n\t shiba (shiba)\n"));  
    }  
  
    [Test]  
    public void BagInBag()  
    {  
        Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a  
backpack");  
        Bag satchel = new Bag(new string[] { "satchel" }, "satchel", "a  
satchel");  
  
        backpack.Inventory.Put(satchel);  
  
        Assert.That(satchel, Is.EqualTo(backpack.Locate("satchel")));  
    }
```

All Tests Passed:

