Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 4 - Look Command

PDF generated at 08:17 on Sunday 15th October, 2023

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

namespace SwinAdventure
{
    public interface IHaveInventory
    {
        public GameObject Locate(string id);
        string Name
        {
            get;
        }
    }
}

```

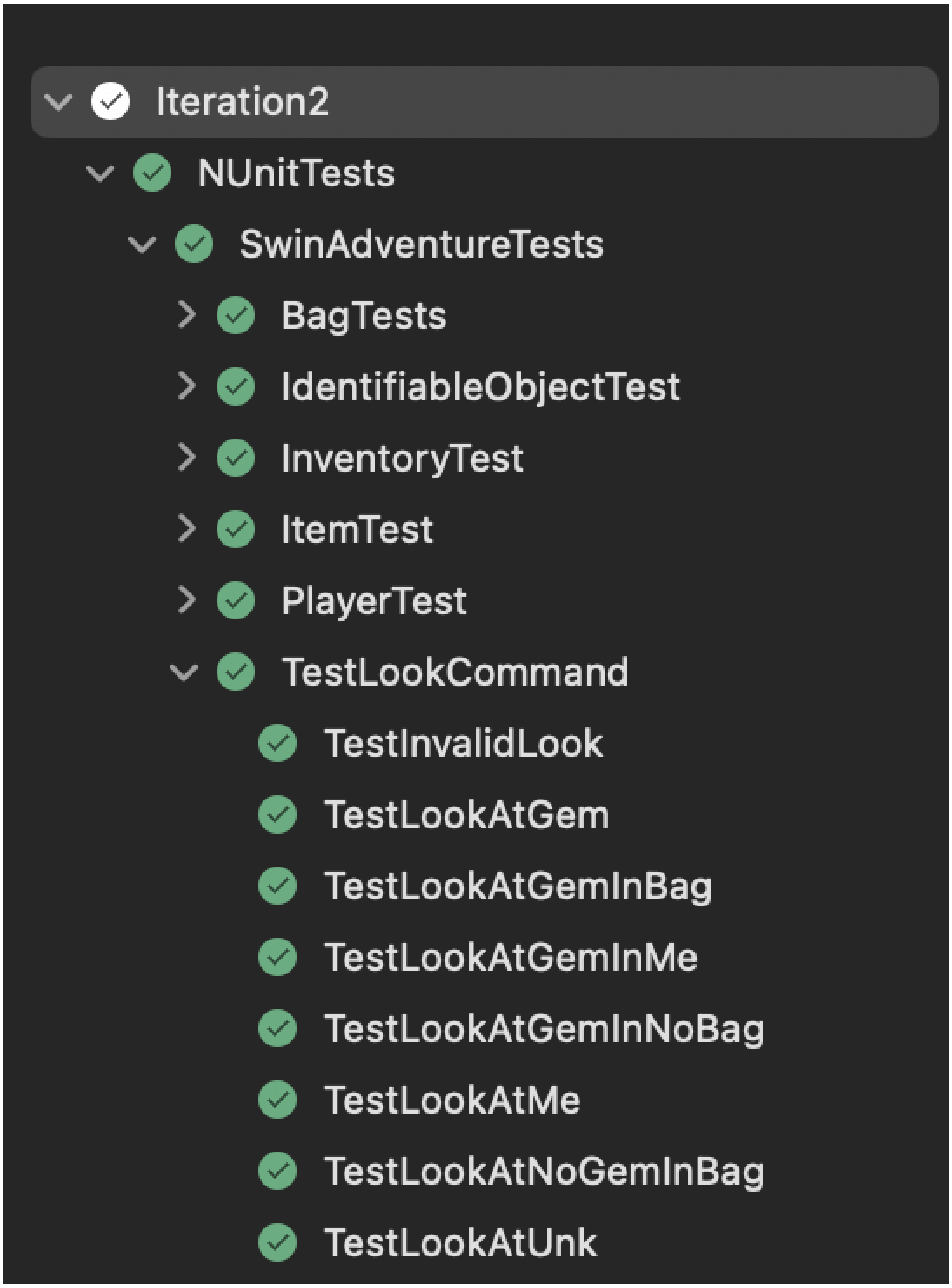
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using SwinAdventure;
using System.Xml.Linq;

namespace SwinAdventure
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        public Player(string name, string desc) : base(new string[] { "me",
 ↪  "inventory" }, name, desc)
        {
            _inventory = new Inventory();
        }

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

        public override string FullDescription
        {
            get
            {
                return $"You are {Name} {base.FullDescription}.\n You are
 ↪  carrying:{_inventory.ItemList}";
            }
        }

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

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Linq;

namespace SwinAdventure;

public class Bag : Item, IHaveInventory
{
    private Inventory _inventory;

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

    public GameObject 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:{_inventory.ItemList}";
        }
    }

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

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

namespace SwinAdventure
{
    public abstract class Command : IdentifiableObject
    {
        public Command(string[] ids) : base(ids) { }
        public abstract string Execute(Player p, string[] text);
    }
}
```

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

namespace SwinAdventure
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {
        }

        public override string Execute(Player p, string[] text)
        {
            IHaveInventory container;
            string itemId;
            string error = "Error in Look Output";

            if (text[0].ToLower() != "look")
                return error;

            switch (text.Length)
            {
                case 3:
                    if (text[1].ToLower() != "at")
                        return "What do you want to look at?";
                    container = p;
                    itemId = text[2];
                    break;

                case 5:
                    container = FetchContainer(p, text[4]);
                    if (container == null)
                        return "I can't find the " + text[4];
                    itemId = text[2];
                    break;

                default:
                    return error;
            }
            return LookAtIn(itemId, container);
        }

        public IHaveInventory FetchContainer(Player p, string containerId)
        {
            return p.Locate(containerId) as IHaveInventory;
        }

        public string LookAtIn(string thingId, IHaveInventory container)
        {
            if (container.Locate(thingId) != null)
            {
                return container.Locate(thingId).FullDescription;
            }
            else
            {
                return $"I can't find the {thingId}";
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;
using NUnit.Framework;
using NUnit.Framework.Internal;
using SwinAdventure;

namespace SwinAdventureTests
{
    [TestFixture]
    public class TestLookCommand
    {
        LookCommand look;
        private Player player;
        private Bag bag;
        private Item gem;


        [SetUp]
        public void SetUp()
        {
            look = new LookCommand();
            player = new Player("Lorraine", "the player");
            bag = new Bag(new string[] { "bag" }, "Bag", "This is a bag");
            gem = new Item(new string[] { "gem" }, "Gem", "A big gem");

            player.Inventory.Put(gem);
            player.Inventory.Put(bag);
            bag.Inventory.Put(gem);
        }

        [Test]
        public void TestLookAtMe()
        {
            string[] input = { "look", "at", "inventory" };
            string actual = look.Execute(player, input);
            string expected = "You are Lorraine the player.\n You are carrying:\n\ta
↪   Gem (gem)\n\ta Bag (bag)";
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestLookAtGem()
        {
            string[] input = { "look", "at", "gem" };
            string expected = "A big gem";
            string actual = look.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestLookAtUnk()
        {
            string[] input = { "look", "at", "unknown" };
            string expected = "I can't find the unknown";
            string actual = look.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestLookAtGemInMe()
        {
            string[] input = { "look", "at", "gem", "in", "inventory" };
            string expected = "A big gem";
            string actual = look.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestLookAtGemInBag()
        {
            string[] input = { "look", "at", "gem", "in", "bag" };
            string expected = "A big gem";
            string actual = look.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestLookAtGemInNoBag()
        {
            string[] input = { "look", "at", "gem", "in", "unknown" };
            string expected = "I can't find the unknown";
            string actual = look.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestLookAtNoGemInBag()
        {
            bag.Inventory.Take("gem");

            string[] input = { "look", "at", "gem", "in", "bag" };
            string expected = "I can't find the gem";
            string actual = look.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }

        [Test]
        public void TestInvalidLook()
        {
            LookCommand lookCmd = new LookCommand();
            string[] input = { "look", "around" };
            string expected = "Error in Look Output";
            string actual = lookCmd.Execute(player, input);
            Assert.AreEqual(expected, actual);
        }
    }
}

```



```
✓ Iteration2
  ✓ NUnitTests
    ✓ SwinAdventureTests
      ✓ BagTests
      ✓ IdentifiableObjectTest
      ✓ InventoryTest
      ✓ ItemTest
      ✓ PlayerTest
      ✓ TestLookCommand
        ✓ TestInvalidLook
        ✓ TestLookAtGem
        ✓ TestLookAtGemInBag
        ✓ TestLookAtGemInMe
        ✓ TestLookAtGemInNoBag
        ✓ TestLookAtMe
        ✓ TestLookAtNoGemInBag
        ✓ TestLookAtUnk
```