Swinburne University of Technology

COS20007 Object Oriented Programming

6.1P - Case Study - Iteration 4 - Look Command

PDF generated at 11:14 on Monday 24th April, 2023

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

namespace Iteration1
{
    public interface IHaveInventory
    {
        public GameObject Locate(string id);

        public string Name
        {
            get;
        }
    }
}
```

```
using Iteration1;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iteration1
{
    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} You are carrying:
↪  {_inventory.ItemList}";
            }
        }

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

```
using Iteration1;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace Iteration1
{
    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;
            }
            return _inventory.Fetch(id);

        }

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

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

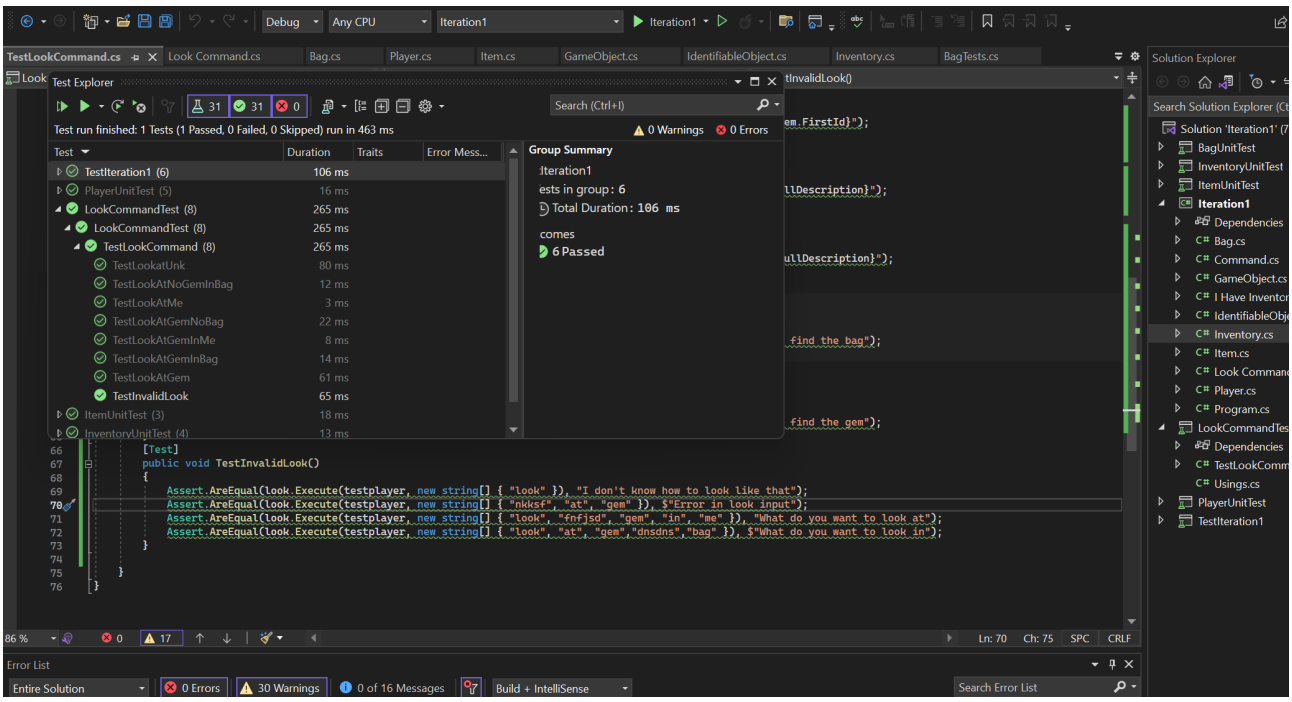
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iteration1
{
    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.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iteration1
{
    public class Look_Command : Command
    {
        public Look_Command() : base(new string[] { })
        {

        }

        public override string Execute(Player p, string[] text)
        {
            IHaveInventory container;
            string itemId;

            if (text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look like that";
            }
            if (text[0] != "look")
            {
                return "Error in look input";
            }
            if (text[1] != "at")
            {
                return "What do you want to look at";
            }
            if (text.Length == 5)
            {
                if (text[3] != "in")
                {
                    return "What do you want to look in";
                }
            }
            if (text.Length == 3)
            {
                container = p;
            }
            else
            {
                container = FetchContainer(p, text[4]);
                if (container == null)
                {
                    return $"I can't find the {text[4]}";
                }
            }
            itemId = text[2];
```

```
54              return LookAtIn(itemId, container);
55          }
56
57          public IHaveInventory FetchContainer (Player p, string containerId)
58          {
59              return p.Locate(containerId) as IHaveInventory;
60          }
61
62          public string LookAtIn (string thingId, IHaveInventory container)
63          {
64              if (container.Locate(thingId) == null)
65              {
66                  return $"I can't find the {thingId}";
67              }
68              else return container.Locate(thingId).FullDescription;
69
70          }
71
72
73      }
74  }
```

```
using Iteration1;
using System.Collections.Concurrent;

namespace LookCommandTest
{
    public class TestLookCommand
    {
        private Look_Command look;
        private Player testplayer;
        private Bag testBag;
        private Item gem;


        [SetUp]
        public void Setup()
        {
            look = new Look_Command();
            testplayer = new Player( "Bob", "the player");
            testBag = new Bag(new string[] {"bag"}, "bag", "this is a bag");
            gem = new Item(new string[] { "gem" }, "red gem", "this is a gem");

            testplayer.Inventory.Put(gem);
            testBag.Inventory.Put(gem);
            testplayer.Inventory.Put(testBag);

        }

        [Test]
        public void TestLookAtMe()
        {
            Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
↪  "inventory" }), testplayer.FullDescription);
        }
        [Test]
        public void TestLookAtGem()
        {
            Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
↪  "gem" }), gem.FullDescription);
        }
        [Test]
        public void TestLookatUnk()
        {
            testplayer.Inventory.Take("gem");
            Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
↪  "gem" }), $"I can't find the {gem.FirstId}");
        }
        [Test]
        public void TestLookAtGemInMe()
        {
            Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
↪  "gem", "in", "me" }), $"{gem.FullDescription}");
        }
        [Test]
```

```
50          public void TestLookAtGemInBag()
51          {
52              Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
    ↪  "gem", "in", "bag" }), $"{gem.FullDescription}");
53          }
54          [Test]
55          public void TestLookAtGemNoBag()
56          {
57              testplayer.Inventory.Take("bag");
58              Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
    ↪  "gem", "in", "bag" }), "I can't find the bag");
59          }
60          [Test]
61          public void TestLookAtNoGemInBag()
62          {
63              testBag.Inventory.Take("gem");
64              Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
    ↪  "gem", "in", "bag" }), "I can't find the gem");
65          }
66          [Test]
67          public void TestInvalidLook()
68          {
69              Assert.AreEqual(look.Execute(testplayer, new string[] { "look" }), "I
    ↪  don't know how to look like that");
70              Assert.AreEqual(look.Execute(testplayer, new string[] { "nkksf", "at",
    ↪  "gem" }), "Error in look input");
71              Assert.AreEqual(look.Execute(testplayer, new string[] { "look",
    ↪  "fnfjsd", "gem", "in", "me" }), "What do you want to look at");
72              Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at",
    ↪  "gem","dnsdns","bag" }), "What do you want to look in");
73          }
74
75      }
76  }
```


Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 463 ms
TestIteration1 (6) 106 ms
PlayerUnitTest (5) 16 ms
LookCommandTest (8) 265 ms
TestLookCommand (8) 265 ms
TestLookatUnk 80 ms
TestLookAtNoGemInBag 12 ms
TestLookAtMe 3 ms
TestLookAtGemNoBag 22 ms
TestLookAtGemInMe 8 ms
TestLookAtGemInBag 14 ms
TestLookAtGem 61 ms
TestInvalidLook 65 ms
ItemUnitTest (3) 18 ms
InventoryUnitTest (4) 13 ms
Group Summary
Iteration1
ests in group: 6
Total Duration: 106 ms
comes
6 Passed
[Test]
public void TestInvalidLook()
{
Assert.AreEqual(look.Execute(testplayer, new string[] { "look" }), "I don't know how to look like that");
Assert.AreEqual(look.Execute(testplayer, new string[] { "nkksf", "at", "gem" }), $"Error in look input");
Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "fnfjsd", "gem", "in", "me" }), "What do you want to look at");
Assert.AreEqual(look.Execute(testplayer, new string[] { "look", "at", "gem","dnsdns","bag" }), $"What do you want to look in");
}
}
}
Solution Explorer
Solution 'Iteration1'
BagUnitTest
InventoryUnitTest
ItemUnitTest
Iteration1
Dependencies
Bag.cs
Command.cs
GameObject.cs
Inventory.cs
Item.cs
Player.cs
Program.cs
PlayerUnitTest
TestIteration1
Error List
0 Errors
30 Warnings
0 of 16 Messages