Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 4 - Look Command

PDF generated at 11:01 on Friday 13th October, 2023

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

namespace Iteration2
{
    public interface IHaveInventory
    {
        public GameObject Locate(string id);

        public string Name
        {
            get;
        }

    }
}
```

```
using Iteration2;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace Iteration2
{


    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;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }
            else
            {
                return null;
            }


           // return _inventory.Fetch(id);
        }



        public override string FullDescription
        {
            get
            {


```

```
54                 return $"You are {Name}, {base.FullDescription} \nyou are
   ↪  carrying:\n\t" + _inventory.ItemList;
55             }
56         }
57
58
59
60         public Inventory Inventory
61         {
62             get
63
64             { return _inventory; }
65         }
66
67     }
68 }
```

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

namespace Iteration2
{


    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 if (_inventory.HasItem(id))
            {


                return _inventory.Fetch(id);
            }


            else return null;
        }



        public override string FullDescription
        {
            get
            {
```

```
54
55                  return $"In the bag you can see:\n" + _inventory.ItemList;
56              }
57          }
58
59
60          public Inventory Inventory
61          {
62              get
63              {
64                  return _inventory;
65              }
66          }
67
68      }
69  }
```

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

namespace Iteration2
{
    public abstract class Command : IdentifiableObject
    {
        public Command(string[] ids) :
            base(ids)
        {
        }
        public abstract string Execute(Player player, string[] text);

    }
}
```

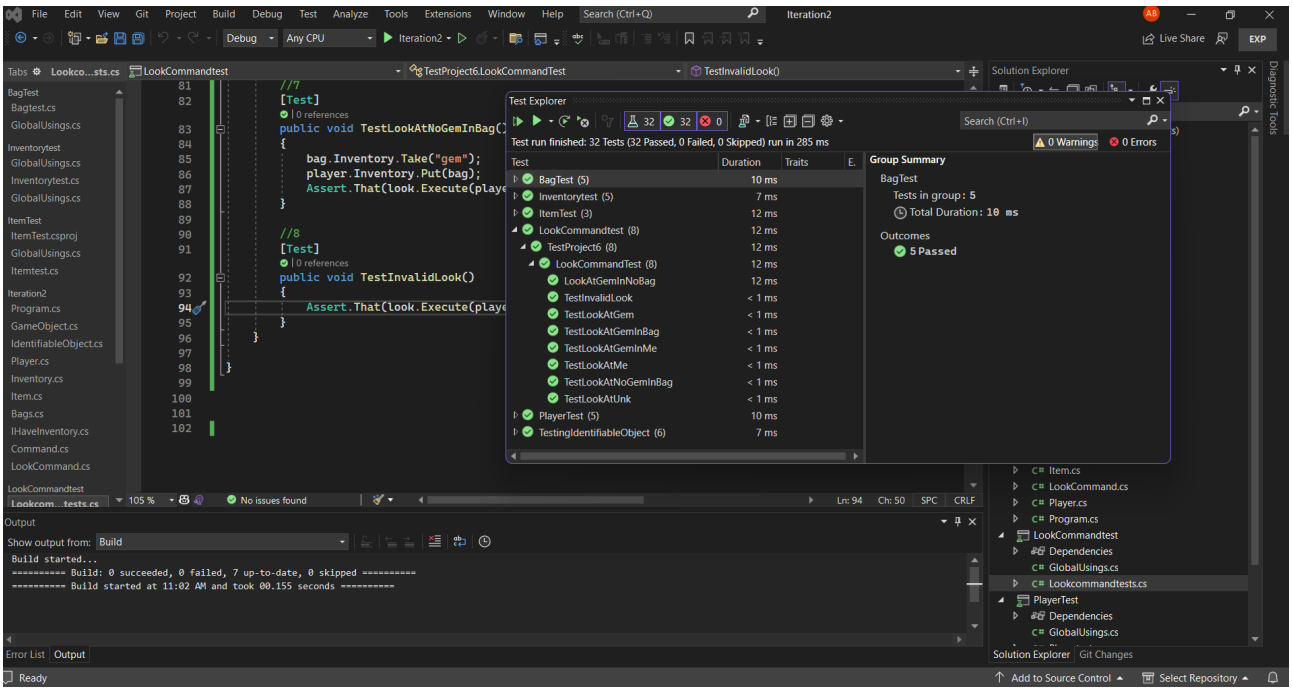
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iteration2
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {

        }
        public override string Execute(Player p, string[] text)
        {
            IHaveInventory container = null;
            string itemId;
            if (text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look like that";
            }
            else
            {
                if (text[0] != "look")
                {
                    return "Error in look input";
                }
                else if (text[1] != "at")
                {
                    return "What do you want to look at?";
                }
                if (text.Length == 5 && text[3] != "in")
                {
                    return "What do you want to look in?";
                }
                switch (text.Length)
                {
                    case 3:
                        container = p;
                        break;
                    case 5:
                        container = FetchContainer(p, text[4]);
                        if (container == null)
                        {
                            return "I can't find the " + text[4];
                        }
                        break;
                }
                itemId = text[2];
                return LookAtIn(itemId, container);
            }
        }
```

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

```
using Iteration2;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection.Metadata;


namespace TestProject6
{
    [TestFixture]
    public class LookCommandTest
    {
        LookCommand look;
        Player player;
        Bag bag;
        Item gem;


        [SetUp]
        public void SetUp()
        {
            look = new();
            player = new Player("Aaryan", "the student");
            bag = new Bag(new string[] { "bag" }, "bag", "This is a expensive bag");
            gem = new Item(new string[] { "gem" }, "a gem", "a bright red crystal");

            player.Inventory.Put(gem);

        }

        //1
        [Test]
        public void TestLookAtMe()
        {
            string expected = player.FullDescription;
            Assert.That(look.Execute(player, new string[] { "look", "at", "me" }),
↪   Is.EqualTo(expected));
        }

        //2
        [Test]
        public void TestLookAtGem()
        {
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem" }),
↪   Is.EqualTo("a bright red crystal"));
        }

        //3
        [Test]
        public void TestLookAtUnk()
        {
```

```
52              player.Inventory.Take("gem");
53              Assert.That(look.Execute(player, new string[] { "look", "at", "gem" }),
    ↪  Is.EqualTo("I can't find the gem"));
54          }
55
56
57          //4
58          [Test]
59          public void TestLookAtGemInMe()
60          {
61              Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "me" }), Is.EqualTo("a bright red crystal"));
62          }
63
64          //5
65          [Test]
66          public void TestLookAtGemInBag()
67          {
68              bag.Inventory.Put(gem);
69              player.Inventory.Put(bag);
70              Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "bag" }), Is.EqualTo("a bright red crystal"));
71          }
72
73          //6
74          [Test]
75          public void LookAtGemInNoBag()
76          {
77              player.Inventory.Take("bag");
78              Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "bag" }), Is.EqualTo("I can't find the bag"));
79          }
80
81          //7
82          [Test]
83          public void TestLookAtNoGemInBag()
84          {
85              bag.Inventory.Take("gem");
86              player.Inventory.Put(bag);
87              Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "bag" }), Is.EqualTo("I can't find the gem"));
88          }
89
90          //8
91          [Test]
92          public void TestInvalidLook()
93          {
94              Assert.That(look.Execute(player, new string[] { "find", "the", "gem" }),
    ↪  Is.EqualTo("Error in look input"));
95          }
96      }
97
98  }
```

99
100
101


Test Explorer
Test run finished: 32 Tests (32 Passed, 0 Failed, 0 Skipped) run in 285 ms
Group Summary
BagTest
Tests in group: 5
Total Duration: 10 ms
Outcomes
5 Passed
Build started...
========== Build: 0 succeeded, 0 failed, 7 up-to-date, 0 skipped ==========
========== Build started at 11:02 AM and took 00.155 seconds ==========