

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Case Study - Iteration 4 - Look Command

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

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public interface IHaveInventory
10     {
11         public GameObject Locate(string id);
12         string Name
13         {
14             get;
15         }
16     }
17 }
18
```

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using SwinAdventure;
6  using System.Xml.Linq;
7
8  namespace SwinAdventure
9  {
10     public class Player : GameObject, IHaveInventory
11     {
12         private Inventory _inventory;
13         public Player(string name, string desc) : base(new string[] { "me",
14             "inventory" }, name, desc)
15         {
16             _inventory = new Inventory();
17         }
18
19         public GameObject Locate(string id)
20         {
21             if (AreYou(id))
22             {
23                 return this;
24             }
25             return _inventory.Fetch(id);
26         }
27
28         public override string FullDescription
29         {
30             get
31             {
32                 return $"You are {Name} {base.FullDescription}.\n You are
33             carrying:{_inventory.ItemList}";
34             }
35         }
36
37         public Inventory Inventory {get { return _inventory; } }
38     }
39 }
```

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Xml.Linq;
6
7  namespace SwinAdventure;
8
9  public class Bag : Item, IHaveInventory
10 {
11     private Inventory _inventory;
12
13     public Bag(string[] ids, string name, string desc) : base(ids, name, desc)
14     {
15         _inventory = new Inventory();
16     }
17
18     public GameObject Locate(string id)
19     {
20         if (AreYou(id))
21         {
22             return this;
23         }
24         else
25         {
26             return _inventory.Fetch(id);
27         }
28     }
29
30     public override string FullDescription
31     {
32         get
33         {
34             return $"In the {Name}, you can see:{_inventory.ItemList}";
35         }
36     }
37
38     public Inventory Inventory
39     {
40         get { return _inventory; }
41     }
42 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public abstract class Command : IdentifiableObject
10     {
11         public Command(string[] ids) : base(ids) { }
12         public abstract string Execute(Player p, string[] text);
13     }
14 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
{
8
9      public class LookCommand : Command
10     {
11         public LookCommand() : base(new string[] { "look" })
12         {
13         }
14
15         public override string Execute(Player p, string[] text)
16         {
17             IHaveInventory container;
18             string itemId;
19             string error = "Error in Look Output";
20
21             if (text[0].ToLower() != "look")
22                 return error;
23
24             switch (text.Length)
25             {
26                 case 3:
27                     if (text[1].ToLower() != "at")
28                         return "What do you want to look at?";
29                     container = p;
30                     itemId = text[2];
31                     break;
32
33                 case 5:
34                     container = FetchContainer(p, text[4]);
35                     if (container == null)
36                         return "I can't find the " + text[4];
37                     itemId = text[2];
38                     break;
39
40                 default:
41                     return error;
42             }
43             return LookAtIn(itemId, container);
44         }
45
46         public IHaveInventory FetchContainer(Player p, string containerId)
47         {
48             return p.Locate(containerId) as IHaveInventory;
49         }
50
51         public string LookAtIn(string thingId, IHaveInventory container)
52         {
53             if (container.Locate(thingId) != null)
```

```
54         {
55             return container.Locate(thingId).FullDescription;
56         }
57     else
58     {
59         return $"I can't find the {thingId}";
60     }
61 }
62 }
63 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Numerics;
5  using System.Text;
6  using System.Threading.Tasks;
7  using NUnit.Framework;
8  using NUnit.Framework.Internal;
9  using SwinAdventure;

10
11 namespace SwinAdventureTests
12 {
13     [TestFixture]
14     public class TestLookCommand
15     {
16         LookCommand look;
17         private Player player;
18         private Bag bag;
19         private Item gem;

20
21
22         [SetUp]
23         public void SetUp()
24         {
25             look = new LookCommand();
26             player = new Player("Lorraine", "the player");
27             bag = new Bag(new string[] { "bag" }, "Bag", "This is a bag");
28             gem = new Item(new string[] { "gem" }, "Gem", "A big gem");

29
30             player.Inventory.Put(gem);
31             player.Inventory.Put(bag);
32             bag.Inventory.Put(gem);
33         }

34
35         [Test]
36         public void TestLookAtMe()
37         {
38             string[] input = { "look", "at", "inventory" };
39             string actual = look.Execute(player, input);
40             string expected = "You are Lorraine the player.\n You are carrying:\n\tta
→ Gem (gem)\n\tBag (bag)";
41             Assert.AreEqual(expected, actual);
42         }

43
44         [Test]
45         public void TestLookAtGem()
46         {
47             string[] input = { "look", "at", "gem" };
48             string expected = "A big gem";
49             string actual = look.Execute(player, input);
50             Assert.AreEqual(expected, actual);
51         }

52 }
```

```
53     [Test]
54     public void TestLookAtUnk()
55     {
56         string[] input = { "look", "at", "unknown" };
57         string expected = "I can't find the unknown";
58         string actual = look.Execute(player, input);
59         Assert.AreEqual(expected, actual);
60     }
61
62     [Test]
63     public void TestLookAtGemInMe()
64     {
65         string[] input = { "look", "at", "gem", "in", "inventory" };
66         string expected = "A big gem";
67         string actual = look.Execute(player, input);
68         Assert.AreEqual(expected, actual);
69     }
70
71     [Test]
72     public void TestLookAtGemInBag()
73     {
74         string[] input = { "look", "at", "gem", "in", "bag" };
75         string expected = "A big gem";
76         string actual = look.Execute(player, input);
77         Assert.AreEqual(expected, actual);
78     }
79
80     [Test]
81     public void TestLookAtGemInNoBag()
82     {
83         string[] input = { "look", "at", "gem", "in", "unknown" };
84         string expected = "I can't find the unknown";
85         string actual = look.Execute(player, input);
86         Assert.AreEqual(expected, actual);
87     }
88
89     [Test]
90     public void TestLookAtNoGemInBag()
91     {
92         bag.Inventory.Take("gem");
93
94         string[] input = { "look", "at", "gem", "in", "bag" };
95         string expected = "I can't find the gem";
96         string actual = look.Execute(player, input);
97         Assert.AreEqual(expected, actual);
98     }
99
100    [Test]
101    public void TestInvalidLook()
102    {
103        LookCommand lookCmd = new LookCommand();
104        string[] input = { "look", "around" };
105        string expected = "Error in Look Output";
```

```
106         string actual = lookCmd.Execute(player, input);
107         Assert.AreEqual(expected, actual);
108     }
109 }
110 }
111
```

