

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Case Study - Iteration 4 - Look Command

PDF generated at 01:44 on Tuesday 5th September, 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 Iteration4
8  {
9      public interface IHaveInventory
10     {
11         GameObject Locate(string id);
12
13         string Name
14         {
15             get;
16         }
17     }
18 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Iteration4
8  {
9      public class Player : GameObject, IHaveInventory
10     {
11         private Inventory _inventory;
12
13         public Player(string name, string desc) : base(new string[] { "Adze", "a
14             ↪ Gamer" }, 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             else
26             {
27                 return _inventory.Fetch(id);
28             }
29         }
30
31         public override string FullDescription
32         {
33             get
34             {
35                 return $"You are {this.Name}, \nYou are
36             ↪ carrying:\n{Inventory.ItemList}";
37             }
38         }
39
40         public Inventory Inventory
41         {
42             get
43             {
44                 return _inventory;
45             }
46         }
47     }
48 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Iteration4
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 {this.Name} \nYou can see:\n{_inventory.ItemList}";
35             }
36         }
37
38         public Inventory Inventory
39         {
40             get
41             {
42                 return _inventory;
43             }
44         }
45     }
46 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Iteration4
8  {
9      public abstract class Command : IdentifiableObject
10     {
11         public Command(string[] ids) : base(ids)
12         {
13
14         }
15
16         public abstract string Execute(Player p, string[] text);
17     }
18 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Iteration4
8  {
9      public class LookCommand : Command
10     {
11         public LookCommand() : base(new string[] { "look" })
12         {
13
14         }
15
16         public override string Execute(Player p, string[] text)
17         {
18             IHaveInventory container;
19             string itemId;
20             string error = "Look input error";
21
22             if (text[0].ToLower() != "look")
23             {
24                 return error;
25             }
26
27             if (text.Length == 3)
28             {
29                 if (text[1].ToLower() != "at")
30                 {
31                     return "What do you want to look at?";
32                 }
33                 container = p;
34                 itemId = text[2];
35
36             }
37             else if (text.Length == 5)
38             {
39                 container = FetchContainer(p, text[4]);
40                 if (container == null)
41                 {
42                     return "Couldn't find " + text[4];
43                 }
44                 itemId = text[2];
45
46             }
47             else
48             {
49                 return error;
50             }
51
52             return LookAtIn(itemId, container);
53         }
54     }
55 }
```

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

```
1  using Iteration4;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace IterationTest4
9  {
10     public class LookCommandTest
11     {
12         Command look;
13         Player player, player2;
14         Bag bag;
15
16         Item gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");
17
18         [SetUp()]
19         public void SetUp()
20         {
21             look = new LookCommand();
22             player = new Player("Adze", "Adze is a player");
23             bag = new Bag(new string[] { "bag" }, $"Adze's bag", $"This is Adze's
24             ↵ bag"); // player containing a bag
25
26             player.Inventory.Put(bag);
27             player2 = new Player("Ducky", "Ducky is a player"); // player with no bag
28         }
29
30         [Test()]
31         public void TestLookAtMe()
32         {
33             string Output = look.Execute(player, new string[] { "look", "at", "Adze"
34             ↵ });
35             string exp = $"You are {player.Name}, \nYou are
36             ↵ carrying:\n{player.Inventory.ItemList}";
37             Assert.AreEqual(exp, Output);
38         }
39
40         [Test()]
41         public void TestLookAtGem()
42         {
43             player.Inventory.Put(gem);
44
45             string Output = look.Execute(player, new string[] { "look", "at", "gem"
46             ↵ });
47             string exp = $"{gem.FullDescription}";
48             Assert.AreEqual(exp, Output);
49         }
50
51         [Test()]
52         public void TestLookAtUnk()
53         {
```

```
50             string Output = look.Execute(player, new string[] { "look", "at", "gem" });
51         });
52         string exp = $"Couldn't find gem";
53         Assert.AreEqual(exp, Output);
54     }
55
56     [Test]
57     public void TestLookAtGemInMe()
58     {
59         player.Inventory.Put(gem);
60         string Output = look.Execute(player, new string[] { "look", "at", "gem",
61             "in", "adze" });
62         string exp = $"{gem.FullDescription}";
63         Assert.AreEqual(exp, Output);
64     }
65
66     [Test()]
67     public void TestLookAtGemInBag()
68     {
69         bag.Inventory.Put(gem);
70         string Output = look.Execute(player, new string[] { "look", "at", "gem",
71             "in", $"bag" });
72         string exp = $"{gem.FullDescription}";
73         Assert.AreEqual(exp, Output);
74     }
75
76     [Test()]
77     public void TestLookAtNoGemInBag()
78     {
79         bag.Inventory.Put(gem);
80         string Output = look.Execute(player, new string[] { "look", "at", "iron",
81             "in", $"bag" });
82         string exp = $"Couldn't find iron";
83         Assert.AreEqual(exp, Output);
84     }
85
86     [Test()]
87     public void TestLookAtGemInNoBag()
88     {
89         bag.Inventory.Put(gem);
90         player2.Inventory.Put(bag);
91         string Output = look.Execute(player2, new string[] { "look", "at", "gem",
92             "in", $"{player.FirstId}" });
93         string exp = $"Couldn't find gem";
94         Assert.AreEqual(exp, Output);
95     }
96
97     [Test()]
98     public void TestInvalidLook()
99     {
100         Assert.AreEqual(look.Execute(player2, new string[] { "look", "around" }),
101             "Look input error");
102         Assert.AreEqual(look.Execute(player2, new string[] { "find", "gem" }),
103             "Look input error");
104     }
105 }
```

```
97      }
98
99
100
101     }
102 }
```

The screenshot shows the Visual Studio Test Explorer window. The status bar at the top right indicates "0 Warnings" and "0 Errors". The main area displays a list of 33 passed tests under the "Test" column. The "D" and "T" columns show the duration and test name respectively. A "Test Detail Summary" pane on the right shows a single expanded test entry: "BagLocatesNothing" with a duration of 1 ms. The Solution Explorer on the right shows files like LookCommand.cs, Player.cs, Program.cs, IterationTest4, Dependencies, and BagTest.cs.

Test	D.	T.	Error Message
BagLocatesNothing	<...		
FullDescription	1...		
IdentifiableObjectTest (6)	<...		
TestAddID	<...		
TestAreYou	<...		
TestCaseSensitive	<...		
TestFirstID	<...		
TestFirstIDNoID	<...		
TestNotAreYou	<...		
InventoryTest (5)	<...		
ItemTest (3)	<...		
FullDesc	<...		
ItemIdentifiable	<...		
ShortDesc	<...		
LookCommandTest (8)	<...		
TestInvalidLook	<...		
TestLookAtGem	<...		
TestLookAtGemInBag	<...		
TestLookAtGemInMe	<...		
TestLookAtGemInNoBag	<...		