Swinburne University of Technology

COS20007 Object Oriented Programming

6.1P - Case Study - Iteration 4 - Look Command

PDF generated at 12:25 on Wednesday 26th April, 2023

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

```

```
using System;

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;
            }
            else
            {
                return _inventory.Fetch(id);
            }
        }

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

    }
}

```

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

```

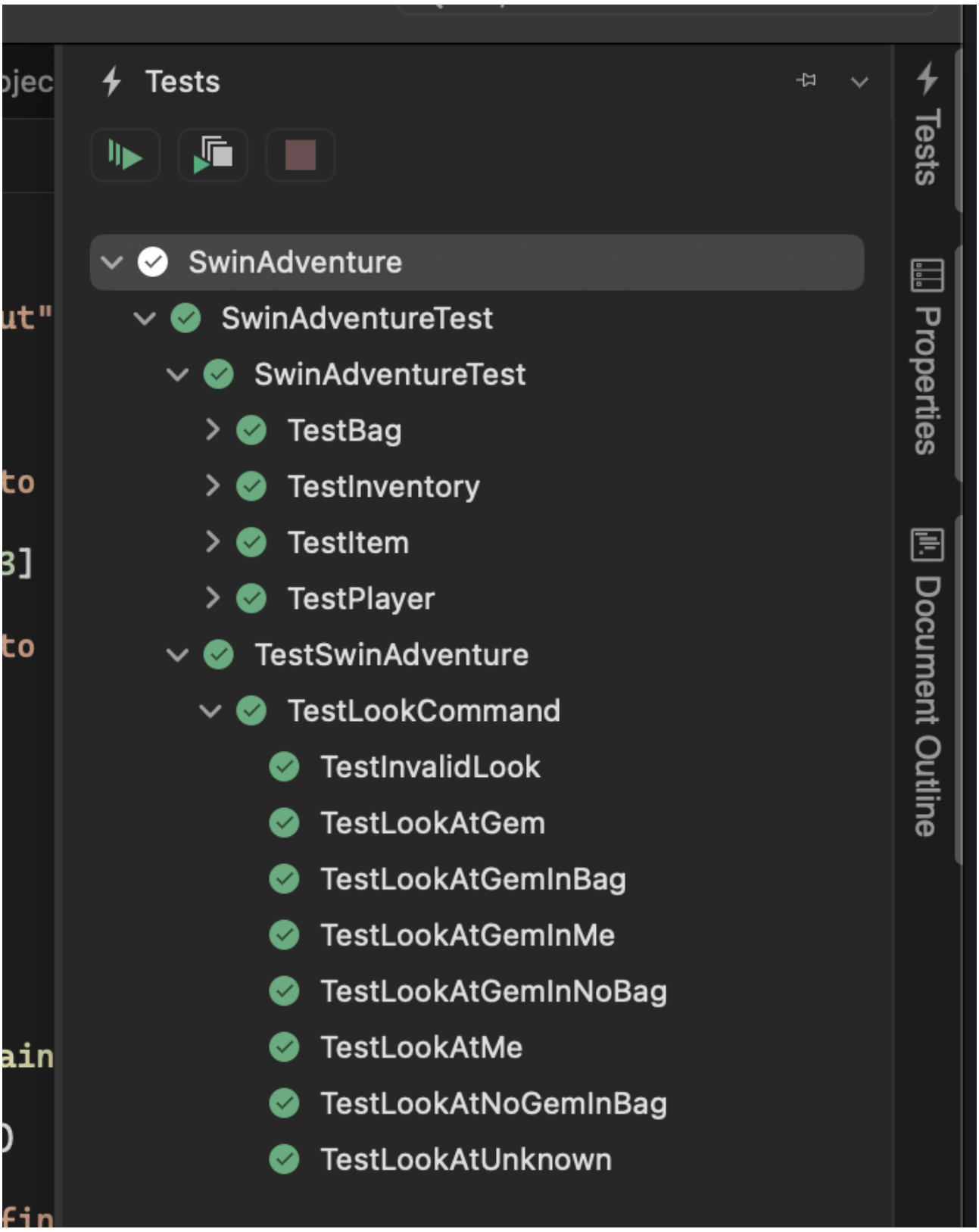
```
namespace SwinAdventure
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" }) { }

        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";
                }
                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);
            }
        }

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

```
54          }
55
56          public string LookAtIn(string thingId, IHaveInventory container)
57          {
58              if (container.Locate(thingId) != null)
59              {
60                  return container.Locate(thingId).FullDescription;
61              }
62              else
63              {
64                  return $"I can't find the {thingId}";
65              }
66          }
67      }
68  }
69
```

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

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

        [SetUp]
        public void Setup()
        {
            look = new LookCommand();
            player = new Player("bob", "the builder");
            bag = new Bag(new string[] { "bag" }, "bag", "This is a bag");
            gem = new Item(new string[] { "gem" }, "big gem", "an expensive item");
        }


        [Test]
        public void TestLookAtMe()
        {
            string actual = look.Execute(player, new string[] { "look", "at",
↪   "inventory" });
            string expected = "You are bob, the builder.\nYou are carrying:\n";

            Assert.That(actual, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtGem()
        {
            //player put gem in inventory
            player.Inventory.Put(gem);

            string actual = look.Execute(player, new string[] { "look", "at", "gem"
↪   });
            string expected = "an expensive item";

            Assert.That(actual, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtUnknown()
        {
```

```
52              string actual = look.Execute(player, new string[] { "look", "at", "gem"
    ↪  });
53              string expected = "I can't find the gem";
54
55              Assert.That(actual, Is.EqualTo(expected));
56          }
57
58          [Test]
59          public void TestLookAtGemInMe()
60          {
61              //look at gem in inventory
62              player.Inventory.Put(gem);
63
64              string actual = look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "inventory" });
65              string expected = "an expensive item";
66
67              Assert.That(actual, Is.EqualTo(expected));
68          }
69
70          [Test]
71          public void TestLookAtGemInBag()
72          {
73              //put gem in bag, then put bag in player's inventory
74              bag.Inventory.Put(gem);
75              player.Inventory.Put(bag);
76
77              string actual = look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "bag" });
78              string expected = "an expensive item";
79
80              Assert.That(actual, Is.EqualTo(expected));
81          }
82
83          [Test]
84          public void TestLookAtGemInNoBag()
85          {
86              bag.Inventory.Put(gem);
87
88              string actual = look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "bag" });
89              string expected = "I can't find the bag";
90
91              Assert.That(actual, Is.EqualTo(expected));
92          }
93
94          // test looking at non existent item in your bag
95          [Test]
96          public void TestLookAtNoGemInBag()
97          {
98              player.Inventory.Put(bag);
99
100             string actual = look.Execute(player, new string[] { "look", "at", "gem",
    ↪  "in", "bag" });
```

```
101             string expected = "I can't find the gem";
102 
103             Assert.That(actual, Is.EqualTo(expected));
104         }
105 
106         [Test]
107         public void TestInvalidLook()
108         {
109             string actual = look.Execute(player, new string[] { "look", "around" });
110             Assert.That(actual, Is.EqualTo("I don't know how to look like that"));
111 
112             string expected = look.Execute(player, new string[] { "hello" });
113             Assert.That(expected, Is.EqualTo("I don't know how to look like that"));
114 
115             string command1 = look.Execute(player, new string[] { "look", "at", "a",
    ↪  "at", "b" });
116             Assert.That(command1, Is.EqualTo("What do you want to look in?"));
117 
118             string command3 = look.Execute(player, new string[] { "hello", "at", "a"
    ↪  });
119             Assert.That(command3, Is.EqualTo("Error in look input"));
120 
121             string command4 = look.Execute(player, new string[] { "look", "by", "a"
    ↪  });
122             Assert.That(command4, Is.EqualTo("What do you want to look at?"));
123         }
124 
125 
126 
127     }
128 }
```


Tests
SwinAdventure
SwinAdventureTest
SwinAdventureTest
TestBag
TestInventory
TestItem
TestPlayer
TestSwinAdventure
TestLookCommand
TestInvalidLook
TestLookAtGem
TestLookAtGemInBag
TestLookAtGemInMe
TestLookAtGemInNoBag
TestLookAtMe
TestLookAtNoGemInBag
TestLookAtUnknown
Tests
Properties
Document Outline