Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 4 - Look Command

PDF generated at 19:23 on Wednesday 11th October, 2023

```
using SwinAdventure;

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

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

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

```

```
using System;
using System.Collections.Generic;

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 Inventory Inventory
        {
            get { return _inventory; }
        }

        public override string FullDescription
        {
            get
            {
                string bagDesc = "In the " + Name + " you can see:" +
    ↪ Environment.NewLine;
                bagDesc += _inventory.ItemList;
                return bagDesc;
            }
        }

        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }
            else
            {
                return null;
            }
        }
    }
}

```

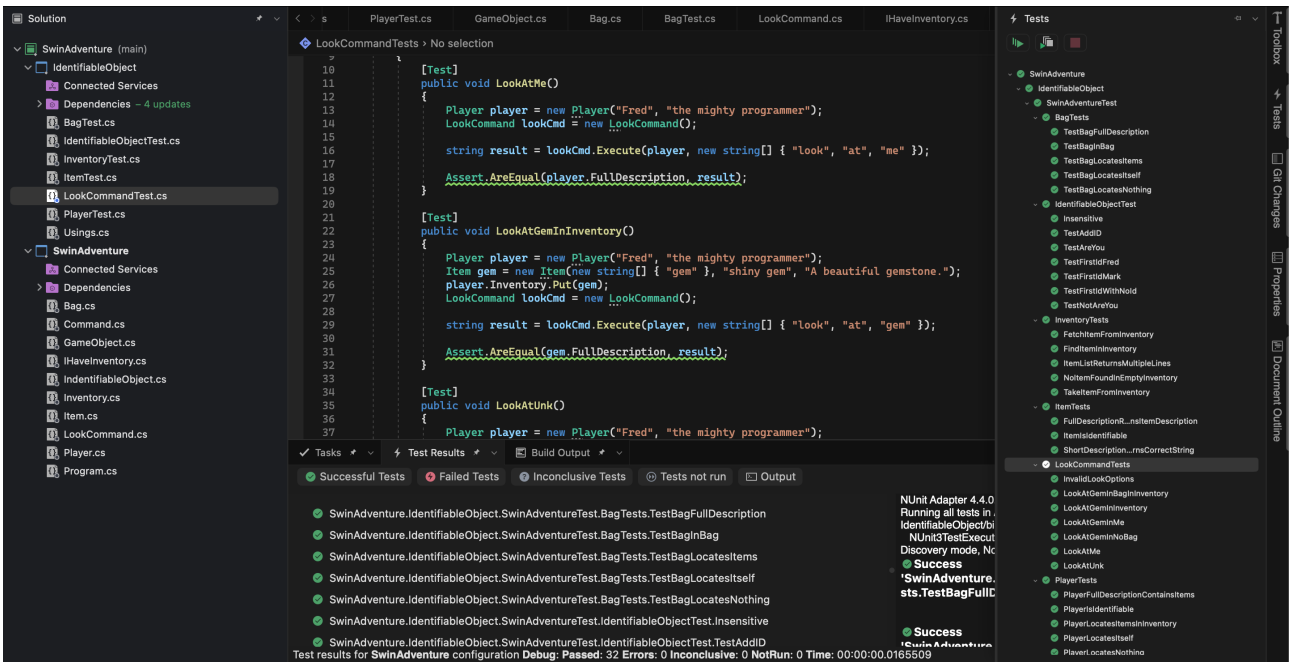
```
using System;
using System.Collections.Generic;

namespace SwinAdventure
{
    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;

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

        public override string Execute(Player player, string[] text)
        {
            if (text.Length == 1)
            {
                return player.FullDescription;
            }
            else if (text.Length == 3 && text[1] == "at")
            {
                string itemToLookAt = text[2];
                GameObject item = player.Locate(itemToLookAt);

                if (item != null)
                {
                    return item.FullDescription;
                }
                else
                {
                    return $"I cannot find the {itemToLookAt}.";
                }
            }
            else if (text.Length == 5 && text[1] == "at" && text[3] == "in")
            {
                string itemToLookAt = text[2];
                string containerId = text[4];

                GameObject container = player.Locate(containerId);

                if (container != null && container is IHaveInventory)
                {
                    IHaveInventory containerWithInventory = container as
↪   IHaveInventory;
                    GameObject item = containerWithInventory.Locate(itemToLookAt);

                    if (item != null)
                    {
                        return item.FullDescription;
                    }
                    else
                    {
                        return $"I cannot find the {itemToLookAt} in the
↪   {container.Name}.";
                    }
                }
```

```
52                  else
53                  {
54                      return $"I cannot find the {containerId}.";
55                  }
56              }
57              else
58              {
59                  return "Look at what?";
60              }
61          }
62      }
63  }
```

```
using System;
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class LookCommandTests
    {
        [Test]
        public void LookAtMe()
        {
            Player player = new Player("Fred", "the mighty programmer");
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "me"
↪   });

            Assert.AreEqual(player.FullDescription, result);
        }

        [Test]
        public void LookAtGemInInventory()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful
↪   gemstone.");
            player.Inventory.Put(gem);
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at",
↪   "gem" });

            Assert.AreEqual(gem.FullDescription, result);
        }

        [Test]
        public void LookAtUnk()
        {
            Player player = new Player("Fred", "the mighty programmer");
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at",
↪   "unknown" });

            Assert.AreEqual("I cannot find the unknown.", result);
        }

        [Test]
        public void LookAtGemInMe()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful
↪   gemstone.");
```

```
50              player.Inventory.Put(gem);
51              LookCommand lookCmd = new LookCommand();
52
53              string result = lookCmd.Execute(player, new string[] { "look", "at",
    ↪  "gem", "in", "me" });
54
55              Assert.AreEqual(gem.FullDescription, result);
56          }
57
58          [Test]
59          public void LookAtGemInBagInInventory()
60          {
61              Player player = new Player("Fred", "the mighty programmer");
62              Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful
    ↪  gemstone.");
63              Bag bag = new Bag(new string[] { "bag" }, "small bag", "A small bag.");
64              bag.Inventory.Put(gem);
65              player.Inventory.Put(bag);
66              LookCommand lookCmd = new LookCommand();
67
68              string result = lookCmd.Execute(player, new string[] { "look", "at",
    ↪  "gem", "in", "bag" });
69
70              Assert.AreEqual(gem.FullDescription, result);
71          }
72
73          [Test]
74          public void LookAtGemInNoBag()
75          {
76              Player player = new Player("Fred", "the mighty programmer");
77              Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful
    ↪  gemstone.");
78              LookCommand lookCmd = new LookCommand();
79
80              string result = lookCmd.Execute(player, new string[] { "look", "at",
    ↪  "gem", "in", "bag" });
81
82              Assert.AreEqual("I cannot find the bag.", result);
83          }
84
85          [Test]
86          public void InvalidLookOptions()
87          {
88              Player player = new Player("Fred", "the mighty programmer");
89              LookCommand lookCmd = new LookCommand();
90
91              string result1 = lookCmd.Execute(player, new string[] { "look", "around"
    ↪  });
92          }
93      }
94  }
```


Solution
SwinAdventure (main)
IdentifiableObject
Connected Services
Dependencies – 4 updates
BagTest.cs
IdentifiableObjectTest.cs
InventoryTest.cs
ItemTest.cs
LookCommandTest.cs
PlayerTest.cs
Usings.cs
SwinAdventure
Connected Services
Dependencies
Bag.cs
Command.cs
GameObject.cs
IHaveInventory.cs
IndentifiableObject.cs
Inventory.cs
Item.cs
LookCommand.cs
Player.cs
Program.cs
PlayerTest.cs
GameObject.cs
Bag.cs
BagTest.cs
LookCommand.cs
IHaveInventory.cs
LookCommandTests › No selection
[Test]
public void LookAtMe()
{
Player player = new Player("Fred", "the mighty programmer");
LookCommand lookCmd = new LookCommand();
string result = lookCmd.Execute(player, new string[] { "look", "at", "me" });
Assert.AreEqual(player.FullDescription, result);
}
[Test]
public void LookAtGemInInventory()
{
Player player = new Player("Fred", "the mighty programmer");
Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful gemstone.");
player.Inventory.Put(gem);
LookCommand lookCmd = new LookCommand();
string result = lookCmd.Execute(player, new string[] { "look", "at", "gem" });
Assert.AreEqual(gem.FullDescription, result);
}
[Test]
public void LookAtUnk()
{
Player player = new Player("Fred", "the mighty programmer");
Tasks
Test Results
Build Output
Successful Tests
Failed Tests
Inconclusive Tests
Tests not run
Output
SwinAdventure.IdentifiableObject.SwinAdventureTest.BagTests.TestBagFullDescription
SwinAdventure.IdentifiableObject.SwinAdventureTest.BagTests.TestBagInBag
SwinAdventure.IdentifiableObject.SwinAdventureTest.BagTests.TestBagLocatesItems
SwinAdventure.IdentifiableObject.SwinAdventureTest.BagTests.TestBagLocatesItself
SwinAdventure.IdentifiableObject.SwinAdventureTest.BagTests.TestBagLocatesNothing
SwinAdventure.IdentifiableObject.SwinAdventureTest.IdentifiableObjectTest.Insensitive
SwinAdventure.IdentifiableObject.SwinAdventureTest.IdentifiableObjectTest.TestAddID
Test results for SwinAdventure configuration Debug: Passed: 32 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0165509
NUnit Adapter 4.4.0
Running all tests in
IdentifiableObject\bi
NUnit3TestExecut
Discovery mode, Nc
Success
'SwinAdventure.
sts.TestBagFullD
Success
Tests
SwinAdventure
IdentifiableObject
SwinAdventureTest
BagTests
TestBagFullDescription
TestBagInBag
TestBagLocatesItems
TestBagLocatesItself
TestBagLocatesNothing
IdentifiableObjectTest
InsensiItive
TestAddID
TestAreYou
TestFirstIdFred
TestFirstIdMark
TestFirstIdWithNoId
TestNotAreYou
InventoryTests
FetchItemFromInventory
FindItemInInventory
ItemListReturnsMultipleLines
NoItemFoundInEmptyInventory
TakeItemFromInventory
ItemTests
FullDescriptionR...nsItemDescription
ItemIsIdentifiable
ShortDescription...rnsCorrectString
LookCommandTests
InvalidLookOptions
LookAtGemInBagInInventory
LookAtGemInInventory
LookAtGemInMe
LookAtGemInNoBag
LookAtMe
LookAtUnk
PlayerTests
PlayerFullDescriptionContainsItems
PlayerIsIdentifiable
PlayerLocatesItemsInInventory
PlayerLocatesItself
Toolbox
Tests
Git Changes
Properties
Document Outline