Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 4 - Look Command

PDF generated at 01:44 on Tuesday 5th September, 2023

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

namespace Iteration4
{
    public interface IHaveInventory
    {
        GameObject Locate(string id);

        string Name
        {
            get;
        }
    }
}
```

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

namespace Iteration4
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;

        public Player(string name, string desc) : base(new string[] { "Adze", "a
↪  Gamer" }, 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 {this.Name}, \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 Iteration4
{
    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 {this.Name} \nYou can see:\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 Iteration4
{
    public abstract class Command : IdentifiableObject
    {
        public Command(string[] ids) : base(ids)
        {

        }

        public abstract string Execute(Player p, string[] text);
    }
}
```

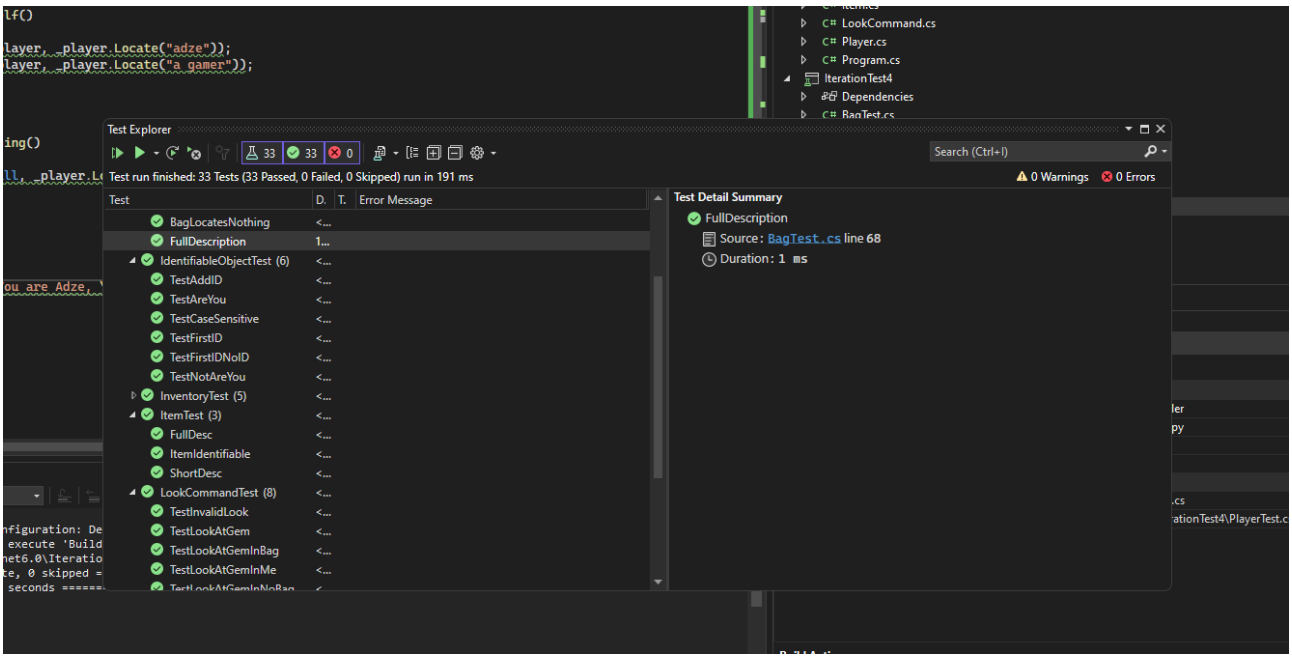
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iteration4
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {

        }

        public override string Execute(Player p, string[] text)
        {
            IHaveInventory container;
            string itemId;
            string error = "Look input error";

            if (text[0].ToLower() != "look")
            {
                return error;
            }

            if (text.Length == 3)
            {
                if (text[1].ToLower() != "at")
                {
                    return "What do you want to look at?";
                }
                container = p;
                itemId = text[2];

            }
            else if (text.Length == 5)
            {
                container = FetchContainer(p, text[4]);
                if (container == null)
                {
                    return "Couldn't find " + text[4];
                }
                itemId = text[2];

            }
            else
            {
                return error;
            }

            return LookAtIn(itemId, container);

```

```
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  }
```

```
using Iteration4;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IterationTest4
{
    public class LookCommandTest
    {
        Command look;
        Player player, player2;
        Bag bag;

        Item gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");

        [SetUp()]
        public void SetUp()
        {
            look = new LookCommand();
            player = new Player("Adze", "Adze is a player");
            bag = new Bag(new string[] { "bag" }, $"Adze's bag", $"This is Adze's bag"); // player containing a bag

            player.Inventory.Put(bag);
            player2 = new Player("Ducky", "Ducky is a player"); // player with no bag
        }

        [Test()]
        public void TestLookAtMe()
        {
            string Output = look.Execute(player, new string[] { "look", "at", "Adze" });
            string exp = $"You are {player.Name}, \nYou are carrying:\n{player.Inventory.ItemList}";
            Assert.AreEqual(exp, Output);
        }

        [Test()]
        public void TestLookAtGem()
        {
            player.Inventory.Put(gem);

            string Output = look.Execute(player, new string[] { "look", "at", "gem" });
            string exp = $"{gem.FullDescription}";
            Assert.AreEqual(exp, Output);
        }

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

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

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


Test Explorer
Test run finished: 33 Tests (33 Passed, 0 Failed, 0 Skipped) run in 191 ms
Search (Ctrl+I)
0 Warnings
0 Errors
Test
D.
T.
Error Message
BagLocatesNothing
FullDescription
IdentifiableObjectTest (6)
TestAddID
TestAreYou
TestCaseSensitive
TestFirstID
TestFirstIDNoID
TestNotAreYou
InventoryTest (5)
ItemTest (3)
FullDesc
ItemIdentifiable
ShortDesc
LookCommandTest (8)
TestInvalidLook
TestLookAtGem
TestLookAtGemInBag
TestLookAtGemInMe
Test Detail Summary
FullDescription
Source: BagTest.cs line 68
Duration: 1 ms
LookCommand.cs
Player.cs
Program.cs
IterationTest4
Dependencies