Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 4 - Look Command

PDF generated at 17:41 on Thursday 9th November, 2023

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

namespace MazeGame
{
    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;
using System.Xml.Linq;

namespace MazeGame
{
    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)
        {
            List<GameObject> gameOBJ = new List<GameObject>();

            if (id == "me" || id == "inventory")
            {
                gameOBJ.Add(this);
            }
            else if (_inventory.HasItem(id))
            {
                var item = _inventory.Fetch(id);
                gameOBJ.Add(item);
            }
            else
            {
                Item nullObj = null;
                gameOBJ.Add(nullObj);
            }

            var result = gameOBJ.ElementAt(0);
            gameOBJ.Clear();
            return result;
        }

        public override string FullDescription
        {
            get
            {
                return
                    $"You are {Name} {Description}\n" +
                    $"You are carrying: \n{_inventory.ItemList}";
            }
        }

```

```
53          public Inventory Inventory
54          {
55              get { return _inventory; }
56          }
57      }
58  }
```

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

namespace MazeGame
{
    public class Bag : Item, IHaveInventory
    {
        private Inventory _inventory;
        public Bag(string[] ids, string name, string desc) : base(ids, name, desc)
        {
            _inventory = new Inventory();
        }

        public override string FullDescription
        {
            get { return $"In the {Name}, you can see:\n{_inventory.ItemList}"; }
        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }

        public GameObject Locate(string id)
        {
            List<GameObject> list = new List<GameObject>();

            if(id == FirstId)
            {
                list.Add(this);
            }
            else if (_inventory.HasItem(id))
            {
                list.Add(_inventory.Fetch(id));
            }
            else
            {
                list.Add(null);
            }

            var result = list.ElementAt(0);
            list.Clear();
            return result;
        }
    }
}
```

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

namespace MazeGame
{
    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 MazeGame
{
    public class LookCommand : Command
    {
        public LookCommand(string[] ids) : base(ids)
        {

        }

        public override string Execute(Player p, string[] text)
        {
            IHaveInventory container;
            string itemID;
            if (text[0] == "look")
            {
                if (text.Length == 3 || text.Length == 5)
                {
                    if (text[1] == "at")
                    {
                        if (text.Length == 3)
                        {
                            container = p;
                            itemID = text[2];

                            return LookAtIn(itemID, container);
                        }
                        else if (text.Length == 5 && text[3] == "in")
                        {
                            string containerID = text[4];
                            container = FetchContainer(p, containerID);
                            string itmReturn;
                            if(container != null)
                            {
                                itemID = text[2];
                                itmReturn = LookAtIn(itemID, container);

                                if(itmReturn == ("I can't find the " + itemID))
                                {
                                    return $"I can't find the {itemID} in
↪  {container.Name}";
                                }
                                else
                                {
                                    return itmReturn;
                                }
                            }
                            else
```

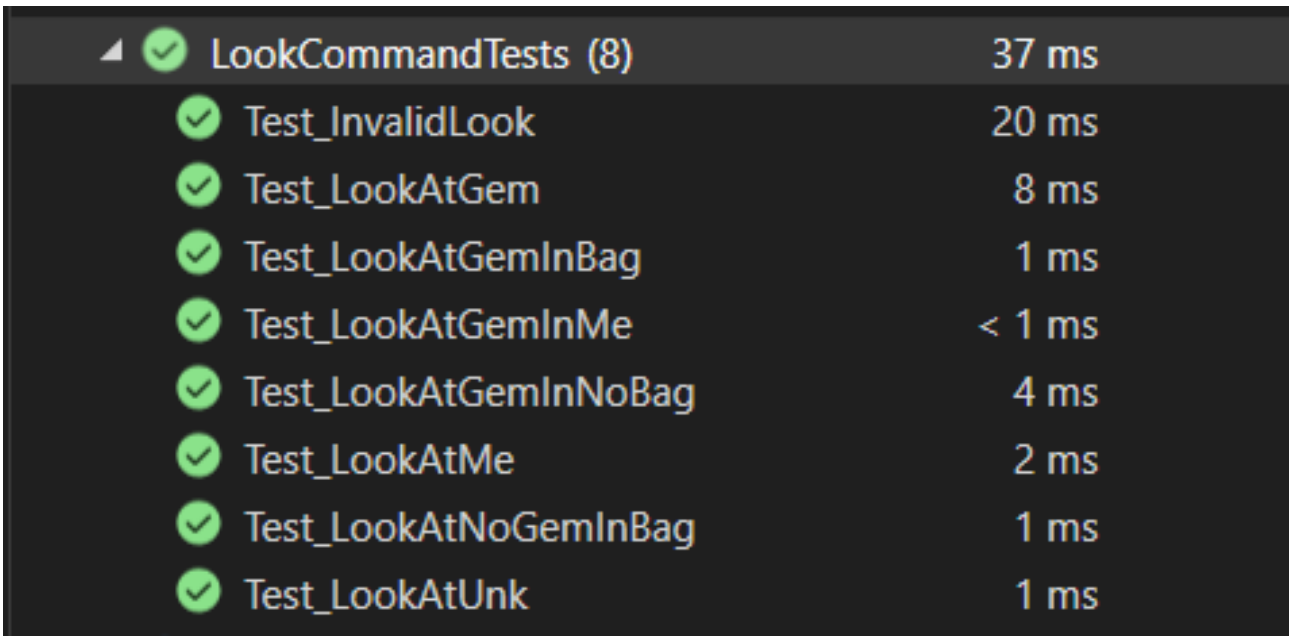
```
                            {
                                return "I can't find the " + containerID;
                            }
                        }
                        else
                        {
                            return "What do you want to looking in?";
                        }
                    }
                    else
                    {
                        return "What do you want to look at?";
                    }
                }
                else
                {
                    return "I don't know how to look like that";
                }
            }
            else
            {
                return "Error in look input";
            }
        }

        private IHaveInventory FetchContainer (Player p, string containerID)
        {
            //IHaveInventory container;
            var result = p.Locate(containerID);
/*              container = result as IHaveInventory;
            return container;*/
            return (IHaveInventory)result;
        }

        private string LookAtIn(string thingId, IHaveInventory container)
        {
            var result = container.Locate(thingId);

            if(result != null)
            {
                return result.FullDescription;
            }
            else
            {
                return "I can't find the " + thingId;
            }
        }
    }
}
```

```
namespace MazeGame.nUnitTests
{
    public class LookCommandTests
    {
        private Player _player { get; set; } = null!;
        private Item sword { get; set; } = null!;
        private Item shovel { get; set; } = null!;
        private Item knife { get; set; } = null!;
        private Item gem { get; set; } = null!;
        private Bag _bag1 { get; set; } = null!;
        private LookCommand look { get; set; } = null!;
        [SetUp]
        public void SetUp()
        {
            _player = new Player("Hoang An", "the comtemplator of infinity");
            sword = new Item(new string[] { "sword" }, "a bronze sword", "A short sword cast from bronze");
            shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable shovel borrowed from the village");
            knife = new Item(new string[] { "knife" }, "an obsidian knife", "A knife made of obsidian");
            _player.Inventory.Put(sword);
            _player.Inventory.Put(shovel);
            _player.Inventory.Put(knife);
            look = new LookCommand(new string[] {"look"});
        }

        [Test]
        public void Test_LookAtMe()
        {

            string command = "look at me";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
            {
                Assert.IsNotNull(sut);
                Assert.That(sut, Is.EqualTo(_player.FullDescription));
            });

            Console.WriteLine(sut.ToString());
        }

        [Test]
        public void Test_LookAtGem()
        {
            gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem that can only be obtained through trade");
            _player.Inventory.Put(gem);
            string command = "look at gem";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
```

```
50              {
51                  Assert.IsNotNull(sut);
52                  Assert.That(sut, Is.EqualTo(gem.FullDescription));
53              });
54              Console.WriteLine();
55              Console.WriteLine(sut);
56          }
57
58          [Test]
59          public void Test_LookAtUnk()
60          {
61              string command = "look at gem";
62              string[] array = command.Split(' ');
63              var sut = look.Execute(_player, array);
64              Assert.That(sut, Is.EqualTo("I can't find the gem"));
65              Console.WriteLine();
66              Console.WriteLine(sut);
67          }
68
69          [Test]
70          public void Test_LookAtGemInMe()
71          {
72              gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem
    ↪  that can only be obtained through trade");
73              _player.Inventory.Put(gem);
74
75              string command = "look at gem in inventory";
76              string[] array = command.Split(' ');
77              var sut = look.Execute(_player, array);
78              Assert.Multiple(() =>
79              {
80                  Assert.IsNotNull(sut);
81                  Assert.That(sut, Is.EqualTo(gem.FullDescription));
82              });
83              Console.WriteLine();
84              Console.WriteLine(sut);
85          }
86
87          [Test]
88          public void Test_LookAtGemInBag()
89          {
90              _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and
    ↪  stiched with leather.");
91              gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem
    ↪  that can only be obtained through trade");
92              _bag1.Inventory.Put(gem);
93              _player.Inventory.Put(_bag1);
94
95              string command = "look at gem in bag1";
96              string[] array = command.Split(' ');
97              var sut = look.Execute(_player, array);
98              Assert.Multiple(() =>
99              {
```

```
100                 Assert.IsNotNull(sut);
101                 Assert.That(sut, Is.EqualTo(gem.FullDescription));
102             });
103             Console.WriteLine();
104             Console.WriteLine(sut);
105         }
106 
107         [Test]
108         public void Test_LookAtGemInNoBag()
109         {
110             _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and
    ↪  stiched with leather.");
111             gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem
    ↪  that can only be obtained through trade");
112             _bag1.Inventory.Put(gem);
113             _player.Inventory.Put(_bag1);
114 
115             string command = "look at gem in bag2";
116             string[] array = command.Split(' ');
117             var sut = look.Execute(_player, array);
118             Assert.Multiple(() =>
119             {
120                 Assert.IsNotNull(sut);
121                 Assert.That(sut, Is.EqualTo("I can't find the bag2"));
122             });
123             Console.WriteLine();
124             Console.WriteLine(sut);
125         }
126 
127         [Test]
128         public void Test_LookAtNoGemInBag()
129         {
130             _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and
    ↪  stiched with leather.");
131             _player.Inventory.Put(_bag1);
132 
133             string command = "look at gem in bag1";
134             string[] array = command.Split(' ');
135             var sut = look.Execute(_player, array);
136             Assert.Multiple(() =>
137             {
138                 Assert.IsNotNull(sut);
139                 Assert.That(sut, Is.EqualTo("I can't find the gem in brown bag"));
140             });
141             Console.WriteLine();
142             Console.WriteLine(sut);
143         }
144 
145         [Test]
146         public void Test_InvalidLook()
147         {
148             string command = "hi there";
149             string command2 = "look";
```

```
150             string command3 = "look in here";
151             string command4 = "look at this at here";
152 
153             string[] array = command.Split(' ');
154             string[] array2 = command2.Split(' ');
155             string[] array3 = command3.Split(' ');
156             string[] array4 = command4.Split(' ');
157 
158             var sut = look.Execute(_player, array);
159             var sut2 = look.Execute(_player, array2);
160             var sut3 = look.Execute(_player, array3);
161             var sut4 = look.Execute(_player, array4);
162 
163             Assert.Multiple(() =>
164             {
165                 Assert.That(sut, Is.EqualTo("Error in look input"));
166                 Assert.That(sut2, Is.EqualTo("I don't know how to look like that"));
167                 Assert.That(sut3, Is.EqualTo("What do you want to look at?"));
168                 Assert.That(sut4, Is.EqualTo("What do you want to looking in?"));
169             });
170 
171             Console.WriteLine();
172             Console.WriteLine("Lists of possible errors with wrong input");
173             Console.WriteLine(sut);
174             Console.WriteLine(sut2);
175             Console.WriteLine(sut3);
176             Console.WriteLine(sut4);
177         }
178     }
179 }
```


LookCommandTests (8) 37 ms
Test_InvalidLook 20 ms
Test_LookAtGem 8 ms
Test_LookAtGemInBag 1 ms
Test_LookAtGemInMe < 1 ms
Test_LookAtGemInNoBag 4 ms
Test_LookAtMe 2 ms
Test_LookAtNoGemInBag 1 ms
Test_LookAtUnk 1 ms