Swinburne University of Technology

COS20007 Object Oriented Programming

7.2C - Case Study - Iteration 6 - Locations

PDF generated at 16:07 on Monday 1st May, 2023

```
using System;

namespace SwinAdventure
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _inventory;

        public Location(string name, string desc) : base(new string[] { "room",
↪   "here" }, name, desc)
        {
            _inventory = new Inventory();
        }

        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }

            return _inventory.Fetch(id);
        }

        // Locations will need to be identifiable and have a name, and description.
        public override string FullDescription
        {
            get
            {
                return $"You are in {Name}\n{base.FullDescription}\nIn this room you
↪   can see:\n{Inventory.ItemList}";
            }
        }

        // Location can contain items
        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }
    }
}
```

```
using System.Security.Principal;

namespace SwinAdventure
{
    public class LocationTest
    {
        private Player _testPlayer;
        private Location _testLocation;

        private Item _sword;
        private Item _shovel;
        private Item _pc;

        [SetUp]
        public void SetUp()
        {
            _testPlayer = new Player("Trung Kien Nguyen", "I am the player");
            _testLocation = new Location("a studio", "A small, beautiful and
    ↪ fully-furnished studio.");

            _sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword",
    ↪ "This is a bronze sword");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a
    ↪ shovel");
            _pc = new Item(new string[] { "pc", "computer" }, "a small computer",
    ↪ "This is a small computer");

            _testPlayer.Inventory.Put(_sword);

            _testLocation.Inventory.Put(_shovel);
            _testLocation.Inventory.Put(_pc);
        }

        // Locations can identify themselves
        [Test]
        public void TestLocationLocateItself()
        {
            Assert.AreEqual(_testLocation.Locate("room"), _testLocation);
            Assert.AreEqual(_testLocation.Locate("here" +
                ""), _testLocation);
        }

        // Locations can locate items they have
        [Test]
        public void TestLocationLocateItems()
        {
            // Locate items that are in the location inventory
            Assert.AreEqual(_testLocation.Locate("shovel"), _shovel);
            Assert.AreEqual(_testLocation.Locate("pc"), _pc);

            // Locate item that is not in the location inventory
            Assert.AreEqual(_testLocation.Locate("sword"), null);
        }
```

```
50
51          [Test]
52          public void TestLocationFullDescription()
53          {
54              Assert.AreEqual(_testLocation.FullDescription, $"You are in
    ↪ {_testLocation.Name}\nA small, beautiful and fully-furnished studio.\nIn this
    ↪ room you can see:\n{_testLocation.Inventory.ItemList}");
55          }
56      }
57  }
```

```
using System;
using System.Net.NetworkInformation;
using System.Text.RegularExpressions;

namespace SwinAdventure
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        private Location _location;

        public Player(string name, string desc) : base( new string[] { "me",
↪   "inventory"}, name, desc)
        {
            _inventory = new Inventory();
        }

        // Players "locate" items by checking three things (in order):
        public GameObject Locate(string id)
        {
            // First checking if they are what is to be located (locate inventory)
            if (AreYou(id))
            {
                return this;
            }

            // Second, checking if they have what is being located (_inventory fetch
↪   gem)
            if (_inventory.Fetch(id) != null)
            {
                return _inventory.Fetch(id);
            }

            // Lastly, checking if the item can be located where they are (
↪   _location, locate gem)
            if (_location != null)
            {
                return _location.Locate(id);
            }

            return null;
        }

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

        public Inventory Inventory
```

```
50          {
51              get
52              {
53                  return _inventory;
54              }
55          }
56
57          // Players have a location.
58          public Location Location
59          {
60              get
61              {
62                  return _location;
63              }
64              set
65              {
66                  _location = value;
67              }
68          }
69      }
70  }
```

```
namespace SwinAdventure
{
    public class PlayerTest
    {
        private Player _testPlayer;
        private Location _testLocation;
        private Item _sword;
        private Item _shovel;
        private Item _pc;

        [SetUp]
        public void SetUp()
        {
            _testPlayer = new Player("Trung Kien Nguyen", "I am the player");
            _testLocation = new Location("a studio", "A small, beautiful and
    ↪ fully-furnished studio.");

            _sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword",
    ↪ "This is a bronze sword");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a
    ↪ shovel");
            _pc = new Item(new string[] { "pc", "computer" }, "a small computer",
    ↪ "This is a small computer");
        }

        [Test]
        public void TestPlayerIsIdentifiable()
        {
            _testPlayer.Inventory.Put(_sword);
            _testPlayer.Inventory.Put(_shovel);
            _testPlayer.Inventory.Put(_pc);

            Assert.IsTrue(_testPlayer.AreYou("me"));
            Assert.IsTrue(_testPlayer.AreYou("inventory"));
        }

        [Test]
        public void TestPlayerLocatesItems()
        {
            _testPlayer.Inventory.Put(_sword);
            _testPlayer.Inventory.Put(_shovel);
            _testPlayer.Inventory.Put(_pc);

            GameObject locatedItem1 = _testPlayer.Locate("shovel");
            GameObject locatedItem2 = _testPlayer.Locate("pc");

            // Test if player has the located item
            Assert.AreEqual(locatedItem1, _shovel);
            Assert.AreEqual(locatedItem2, _pc);

            // Test if the item remains in the player's inventory
            Assert.IsTrue(_testPlayer.Inventory.HasItem("shovel"));
            Assert.IsTrue(_testPlayer.Inventory.HasItem("pc"));
```

```
50          }
51
52          [Test]
53          public void TestPlayerLocatesItself()
54          {
55              _testPlayer.Inventory.Put(_sword);
56              _testPlayer.Inventory.Put(_shovel);
57              _testPlayer.Inventory.Put(_pc);
58
59              GameObject playerItself1 = _testPlayer.Locate("me");
60              GameObject playerItself2 = _testPlayer.Locate("inventory");
61
62              // Test if player has the located itself with the keyword "me"
63              Assert.AreEqual(playerItself1, _testPlayer);
64
65              // Test if player has the located itself with the keyword "inventory"
66              Assert.AreEqual(playerItself2, _testPlayer);
67          }
68
69          [Test]
70          public void TestPlayerLocatesNothing()
71          {
72              _testPlayer.Inventory.Put(_sword);
73              _testPlayer.Inventory.Put(_shovel);
74              _testPlayer.Inventory.Put(_pc);
75
76              GameObject nonExistentObject = _testPlayer.Locate("gun");
77
78              Assert.AreEqual(nonExistentObject, null);
79          }
80
81          [Test]
82          public void TestPlayerFullDescription()
83          {
84              _testPlayer.Inventory.Put(_sword);
85              _testPlayer.Inventory.Put(_shovel);
86              _testPlayer.Inventory.Put(_pc);
87
88              string playerFullDesc = $"You are {_testPlayer.Name}, (I am the player),
    ↪  you are carrying:\n{_testPlayer.Inventory.ItemList}";
89
90              Assert.AreEqual(playerFullDesc, _testPlayer.FullDescription);
91          }
92
93          [Test]
94          public void TestPlayersCanLocateTheirLocation()
95          {
96              _testPlayer.Inventory.Put(_sword);
97
98              _testLocation.Inventory.Put(_shovel);
99              _testLocation.Inventory.Put(_pc);
100
101             _testPlayer.Location = _testLocation;
```

```
102
103             Assert.AreEqual(_testPlayer.Locate("room"), _testLocation);
104             Assert.AreEqual(_testPlayer.Locate("here"), _testLocation);
105         }
106
107         // Players can locate items in their location
108         [Test]
109         public void TestPlayerCanLocateItemsInTheirLocation()
110         {
111             _testPlayer.Inventory.Put(_sword);
112
113             _testLocation.Inventory.Put(_shovel);
114             _testLocation.Inventory.Put(_pc);
115
116             _testPlayer.Location = _testLocation;
117
118             // Locate items that are in the player location inventory, but not in
    ↪  the player inventory
119             Assert.AreEqual(_testPlayer.Locate("shovel"), _shovel);
120             Assert.AreEqual(_testPlayer.Locate("pc"), _pc);
121
122             Assert.IsFalse(_testPlayer.Inventory.HasItem("shovel"));
123             Assert.IsFalse(_testPlayer.Inventory.HasItem("pc"));
124         }
125     }
126 }
```

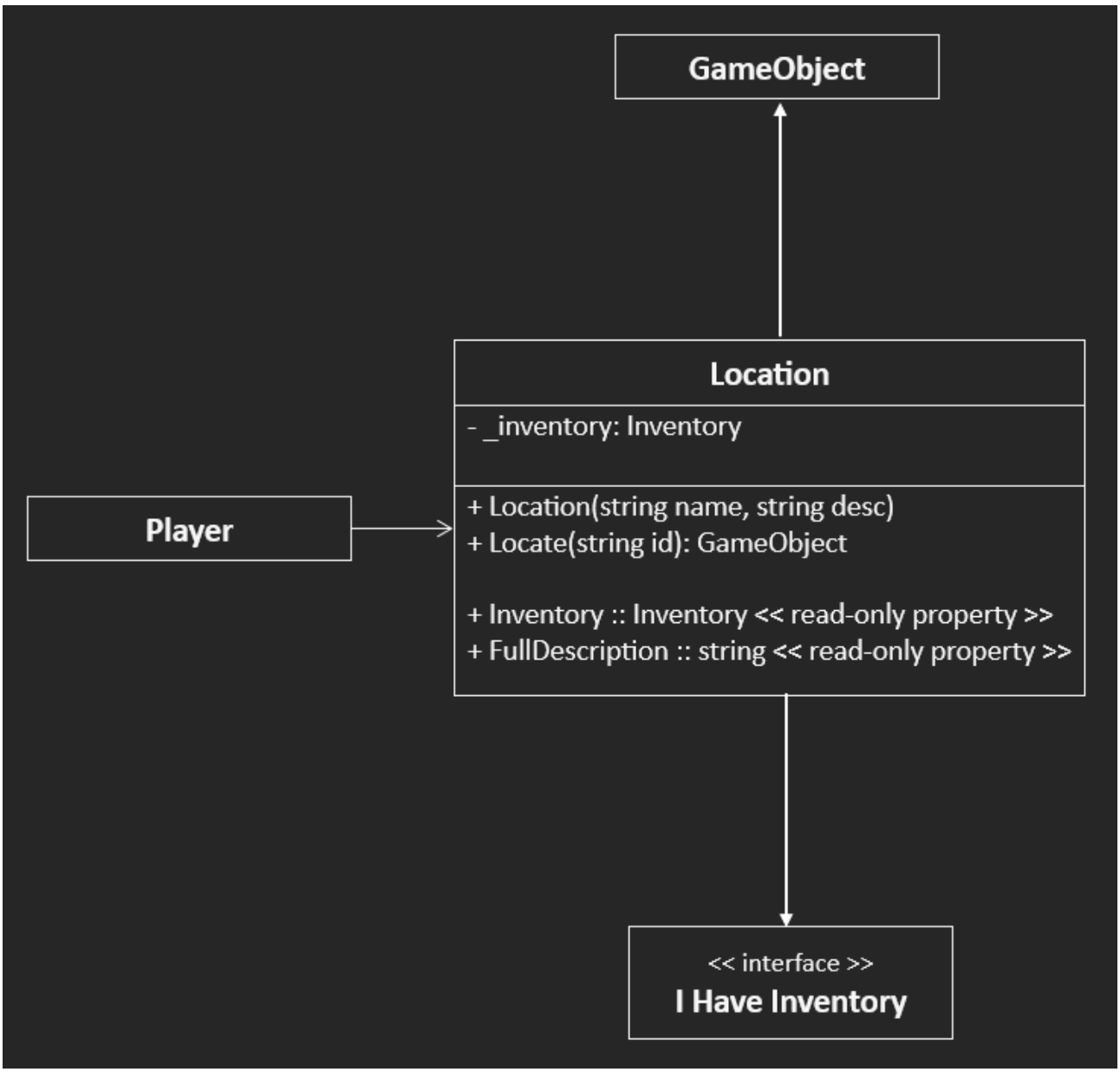
```
using System;

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

        public override string Execute(Player p, string[] text)
        {
            if ((text.Length == 1) || (text.Length == 3) || (text.Length == 5))
            {
                if (text[0].ToLower() != "look")
                {
                    return "Error in look input";
                }

                IHaveInventory container;
                string itemId;

                // This will change the look command to also include "look" to look
↪  at the player's location.
                if (text.Length == 1)
                {
                    container = p as IHaveInventory;
                    itemId = "room";
                }
                else
                {
                    if (text[1].ToLower() != "at")
                    {
                        return "What do you want to look at?";
                    }

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

                        container = FetchContainer(p, text[4]);
                        if (container == null)
                        {
                            return $"I cannot find the {text[4]}";
                        }
```

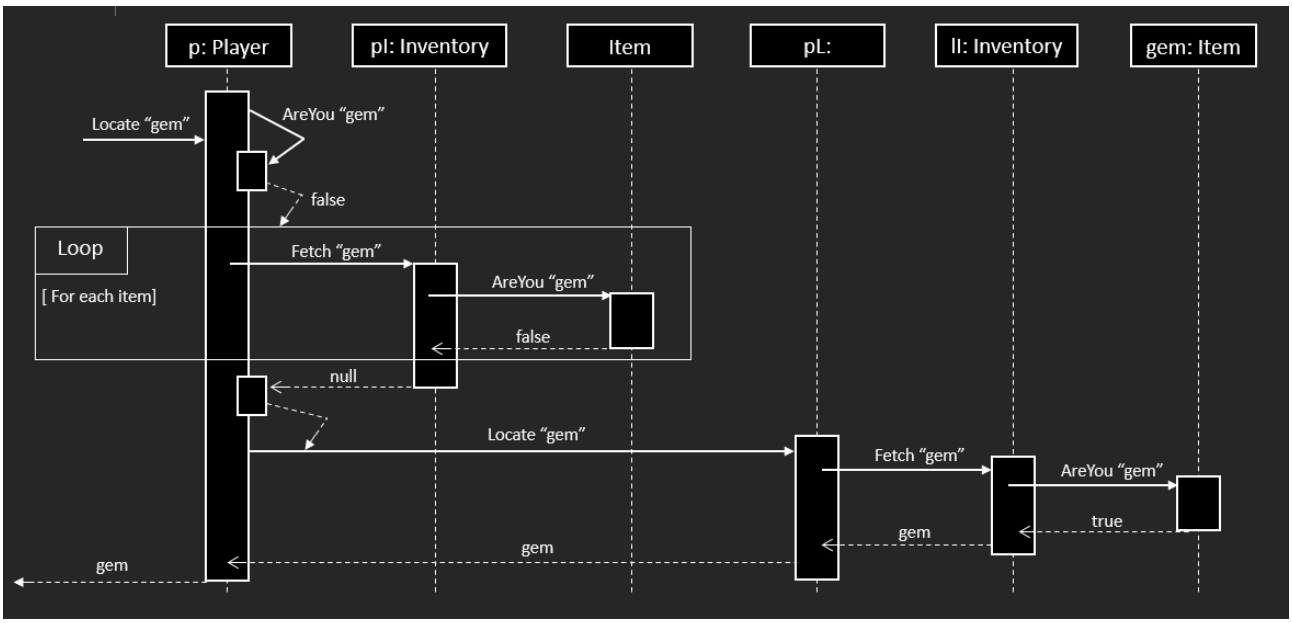
```
53                          itemId = text[2];
54                      }
55                  }
56
57                  return LookAtIn(itemId, container);
58              }
59
60              return "I don't know how to look like that";
61          }
62
63          private IHaveInventory FetchContainer(Player p, string containerId)
64          {
65              return p.Locate(containerId) as IHaveInventory;
66          }
67
68          private string LookAtIn(string thingId, IHaveInventory container)
69          {
70              if (container.Locate(thingId) != null)
71              {
72                  return container.Locate(thingId).FullDescription;
73              }
74
75              return $"I cannot find the {thingId} in the {container.Name}";
76          }
77      }
78  }
```

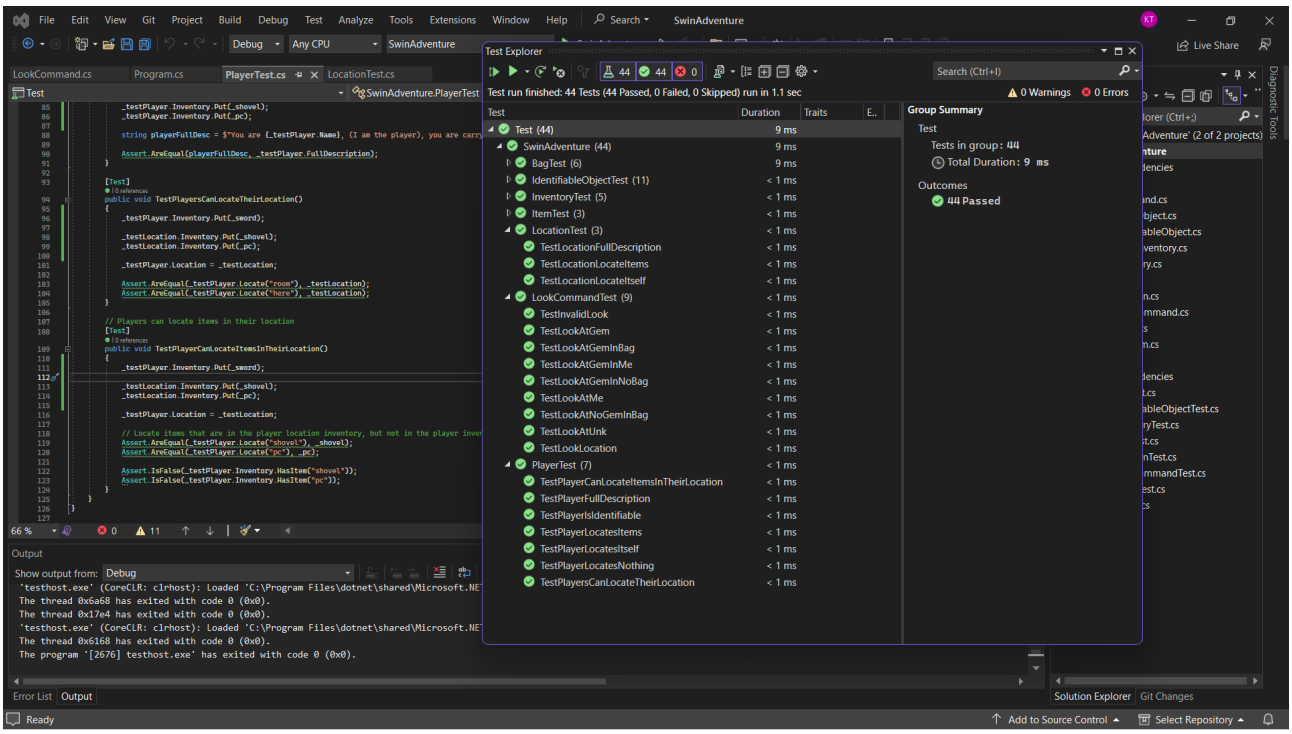
```
using System.Xml.Linq;

namespace SwinAdventure
{
    public class LookCommandTest
    {
        private LookCommand _testLookCommand;
        private Player _testPlayer;
        private Bag _testBag;

        private Item _gem;
        private Item _sword;

        [SetUp]
        public void SetUp()
        {
            _testLookCommand = new LookCommand();

            _gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");
            _sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword",
↪   "This is a bronze sword");

            _testPlayer = new Player("Trung Kien Nguyen", "I am the player");

            _testBag = new Bag(new string[] { "bag" }, "small bag", "This is a small
↪   bag");
        }

        [Test]
        public void TestLookAtMe()
        {
            string[] testCommand = new string[] { "look", "at", "inventory" };
            Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
↪   $"You are {_testPlayer.Name}, (I am the player), you are
↪   carrying:\n{_testPlayer.Inventory.ItemList}");
        }

        [Test]
        public void TestLookAtGem()
        {
            _testPlayer.Inventory.Put(_gem);

            string[] testCommand = new string[] { "look", "at", "gem" };
            Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
↪   "This is a gem");
            Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
↪   _gem.FullDescription);
        }

        [Test]
        public void TestLookAtUnk()
        {
            string[] testCommand = new string[] { "look", "at", "gem" };
```

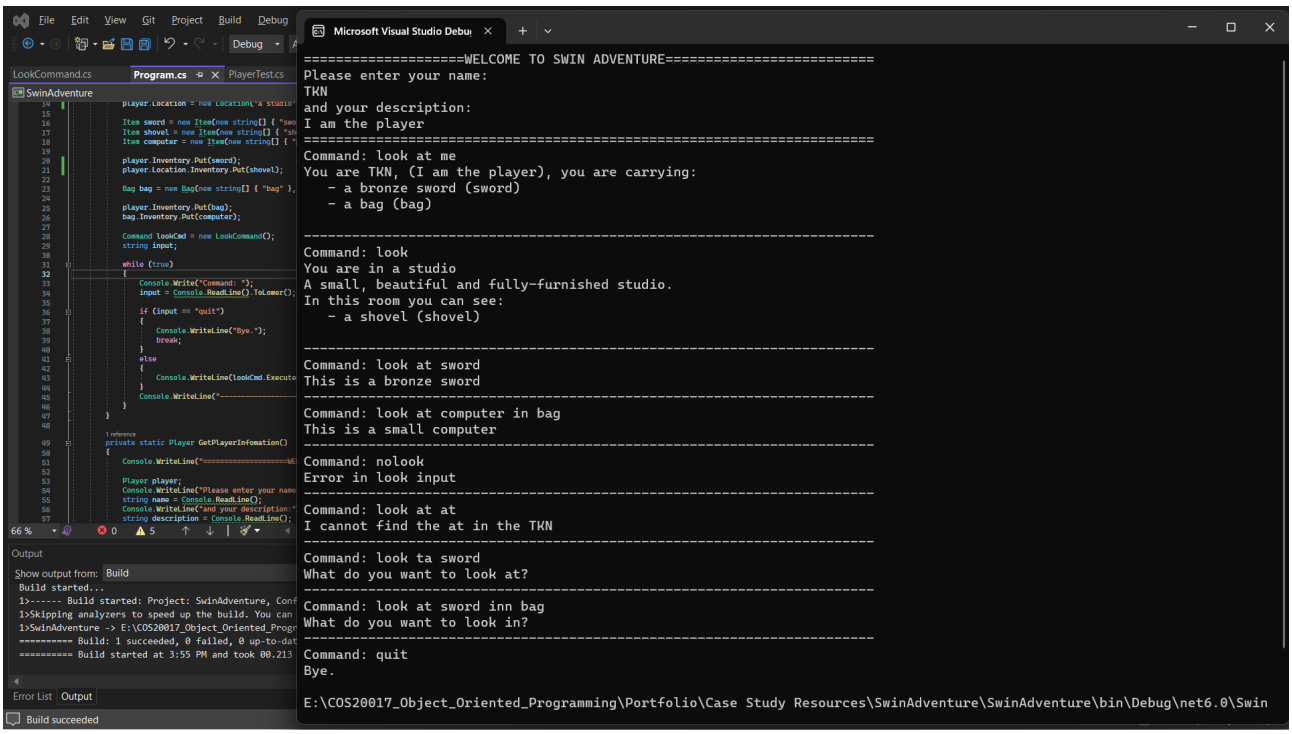
```
48              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand), $"I
    ↪ cannot find the {testCommand[2]} in the {_testPlayer.Name}");
49          }
50
51          [Test]
52          public void TestLookAtGemInMe()
53          {
54              _testPlayer.Inventory.Put(_gem);
55
56              string[] testCommand = new string[] { "look", "at", "gem", "in",
    ↪ "inventory" };
57              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
    ↪ "This is a gem");
58              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
    ↪ _gem.FullDescription);
59          }
60
61          [Test]
62          public void TestLookAtGemInBag()
63          {
64              _testBag.Inventory.Put(_gem);
65              _testPlayer.Inventory.Put(_testBag);
66
67              string[] testCommand = new string[] { "look", "at", "gem", "in", "bag" };
68              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
    ↪ "This is a gem");
69              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand),
    ↪ _gem.FullDescription);
70          }
71
72          [Test]
73          public void TestLookAtGemInNoBag()
74          {
75              _testBag.Inventory.Put(_gem);
76              string[] testCommand = new string[] { "look", "at", "gem", "in", "bag" };
77              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand), $"I
    ↪ cannot find the {testCommand[4]}");
78          }
79
80          [Test]
81          public void TestLookAtNoGemInBag()
82          {
83              _testPlayer.Inventory.Put(_testBag);
84              string[] testCommand = new string[] { "look", "at", "gem", "in", "bag" };
85              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand), $"I
    ↪ cannot find the {testCommand[2]} in the {_testBag.Name}");
86          }
87
88          [Test]
89          public void TestInvalidLook()
90          {
91              string[] testCommand1 = new string[] { "hello" , "hi", "howareyou"};
92              string[] testCommand2 = new string[] { "no", "look", "at" };
```

```
93
94              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand1),
    ↪ "Error in look input");
95              Assert.AreEqual(_testLookCommand.Execute(_testPlayer, testCommand2),
    ↪ "Error in look input");
96          }
97
98          [Test]
99          public void TestLookLocation()
100         {
101             _testPlayer.Location = new Location("a studio", "A small, beautiful and
    ↪ fully-furnished studio.");
102             _testPlayer.Location.Inventory.Put(_sword);
103
104             Assert.AreEqual(_testLookCommand.Execute(_testPlayer, new string[] {
    ↪ "look" }),
105                 $"You are in {_testPlayer.Location.Name}\nA small, beautiful and
    ↪ fully-furnished studio.\nIn this room you can
    ↪ see:\n{_testPlayer.Location.Inventory.ItemList}");
106         }
107     }
108 }
```


GameObject
Location
- _inventory: Inventory
Player
+ Location(string name, string desc)
+ Locate(string id): GameObject
+ Inventory :: Inventory << read-only property >>
+ FullDescription :: string << read-only property >>
<< interface >>
I Have Inventory


p: Player
pI: Inventory
Item
pL:
lI: Inventory
gem: Item
Locate "gem"
AreYou "gem"
false
Loop
[For each item]
Fetch "gem"
AreYou "gem"
false
null
Locate "gem"
Fetch "gem"
AreYou "gem"
true
gem
gem
gem


Test Explorer
Test run finished: 44 Tests (44 Passed, 0 Failed, 0 Skipped) run in 1.1 sec
Group Summary
Tests in group: 44
Total Duration: 9 ms
Outcomes
44 Passed
Test (44)
SwinAdventure (44)
BagTest (6)
IdentifiableObjectTest (11)
InventoryTest (5)
ItemTest (3)
LocationTest (3)
TestLocationFullDescription
TestLocationLocateItems
TestLocationLocateItself
LookCommandTest (9)
TestInvalidLook
TestLookAtGem
TestLookAtGemInBag
TestLookAtGemInMe
TestLookAtGemInNoBag
TestLookAtMe
TestLookAtNoGemInBag
TestLookAtUnk
TestLookLocation
PlayerTest (7)
TestPlayerCanLocateItemsInTheirLocation
TestPlayerFullDescription
TestPlayerIsIdentifiable
TestPlayerLocatesItems
TestPlayerLocatesItself
TestPlayerLocatesNothing
TestPlayersCanLocateTheirLocation


File Edit View Git Project Build Debug
Debug
LookCommand.cs
Program.cs
PlayerTest.cs
SwinAdventure
Output
Show output from: Build
Build started...
Error List
Output
Build succeeded
Microsoft Visual Studio Debu
====================WELCOME TO SWIN ADVENTURE==========================
Please enter your name:
TKN
and your description:
I am the player
==
Command: look at me
You are TKN, (I am the player), you are carrying:
- a bronze sword (sword)
- a bag (bag)

Command: look
You are in a studio
A small, beautiful and fully-furnished studio.
In this room you can see:
- a shovel (shovel)

Command: look at sword
This is a bronze sword

Command: look at computer in bag
This is a small computer

Command: nolook
Error in look input

Command: look at at
I cannot find the at in the TKN

Command: look ta sword
What do you want to look at?

Command: look at sword inn bag
What do you want to look in?

Command: quit
Bye.
E:\COS20017_Object_Oriented_Programming\Portfolio\Case Study Resources\SwinAdventure\SwinAdventure\bin\Debug\net6.0\Swin