

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

---

## Case Study - Iteration 6 - Locations

---

PDF generated at 02:46 on Friday 10<sup>th</sup> November, 2023

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MazeGame
8  {
9      public class Location : GameObject, IHaveInventory
10     {
11         private Inventory _inventory;
12
13         public Location(string[] ids, string name, string desc) : base(ids, name,
↪ desc)
14         {
15             _inventory = new Inventory();
16         }
17
18         public GameObject Locate(string id)
19         {
20             List<GameObject> items = new List<GameObject>();
21
22             if(_inventory.HasItem(id))
23             {
24                 var itm = _inventory.Fetch(id);
25                 items.Add(itm);
26             }
27             else
28             {
29                 Item nullObj = null;
30                 items.Add(nullObj);
31             }
32
33             var result = items.ElementAt(0);
34             items.Clear();
35             return result;
36         }
37
38         public override string FullDescription
39         {
40             get
41             {
42                 return $"You are in a {Name}\n" +
43                     $"{Description}\n"+
44                     $"In this place, you can see:\n"+
45                     $"{Inventory.ItemList}";
46             }
47         }
48
49         public Inventory Inventory
50         {
51             get { return _inventory; }
52         }
53     }
```

53        }  
54    }

```
1 using System.ComponentModel;
2 using System.Diagnostics.Metrics;
3 using System.Security.Cryptography;
4
5 namespace MazeGame.nUnitTests
6 {
7     public class LocationTests
8     {
9         private Location garden { get; set; } = null!;
10        private Item water { get; set; } = null!;
11        private Item pearl { get; set; } = null!;
12        private Player _player { get; set; } = null!;
13        private Item sword { get; set; } = null!;
14        private Item shovel { get; set; } = null!;
15        private Item pickaxe { get; set; } = null!;
16        private LookCommand look { get; set; } = null!;
17        private Bag _bag1 { get; set; } = null!;
18        private Item gem { get; set; } = null!;
19
20        [SetUp]
21        public void Setup()
22        {
23            garden = new Location(new string[] { "garden" }, "green garden", "A
↪ garden blooming with natural plants, trees, and flowers");
24            water = new Item(new string[] { "water" }, "a bottled water", "A 1 Litres
↪ bottle of spring water to keep you hydrated");
25            pearl = new Item(new string[] { "pearl" }, "a pearl", "A pearl picked
↪ from pearl tree. A fruit great for snack");
26            garden.Inventory.Put(water);
27            garden.Inventory.Put(pearl);
28            _player = new Player("Hoang An", "the comtemplator of infinity");
29            sword = new Item(new string[] { "sword" }, "a bronze sword", "A short
↪ sword cast from bronze");
30            shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable
↪ shovel borrowed from the village");
31            pickaxe = new Item(new string[] { "pickaxe" }, "an obsidian pickaxe", "A
↪ pickaxe made of obsidian");
32            _player.ChangeLocation(garden);
33            _player.Inventory.Put(sword);
34            _player.Inventory.Put(shovel);
35            _player.Inventory.Put(pickaxe);
36            look = new LookCommand(new string[] { "look", "Look" });
37        }
38
39        [Test]
40        public void Test_LocationIsIdentifiable()
41        {
42            var sut = garden.AreYou("garden");
43            Assert.That(sut, Is.True);
44            Console.WriteLine(sut);
45        }
46
47        [Test]
```

```
48     public void Test_LocationSelfLocate()
49     {
50         string command = "Look";
51         string[] array = command.Split(' ');
52         var sut = look.Execute(_player, array);
53         Assert.Multiple(() =>
54         {
55             Assert.That(sut, Is.Not.Null);
56             Assert.That(sut, Is.EqualTo(garden.FullDescription));
57         });
58         Console.WriteLine(sut);
59     }
60
61     [Test]
62     public void Test_LocationLocatesItem()
63     {
64         var sut1 = garden.Locate("water");
65         var sut2 = garden.Locate("pearl");
66         Assert.Multiple(() =>
67         {
68             Assert.That(sut1.Description, Is.EqualTo(water.Description));
69             Assert.That(sut1.Description, Is.Not.Null);
70             Assert.That(sut2.Description, Is.EqualTo(pearl.Description));
71             Assert.That(sut2.Description, Is.Not.Null);
72         });
73         Console.WriteLine($"Water: {sut1.Description}");
74         Console.WriteLine($"Pearl: {sut2.Description}");
75     }
76
77     [Test]
78     public void Test_LocationLocatesUnkItem()
79     {
80         var sut = garden.Locate("chair");
81         Assert.That(sut, Is.Null);
82         if(sut == null)
83         {
84             Console.WriteLine("Item was not found");
85         }
86     }
87
88     [Test]
89     public void Test_PlayerLocatesItemInArea()
90     {
91         string command1 = "look at water";
92         string command2 = "look at pearl";
93         string[] array1 = command1.Split(' ');
94         string[] array2 = command2.Split(' ');
95         var sut1 = look.Execute(_player, array1);
96         var sut2 = look.Execute(_player, array2);
97         Assert.Multiple(() =>
98         {
99             Assert.That(sut1, Is.EqualTo(water.Description));
100             Assert.That(sut1, Is.Not.Null);
```

```
101         Assert.That(sut2, Is.EqualTo(pearl.Description));
102         Assert.That(sut2, Is.Not.Null);
103     });
104     Console.WriteLine($"Water: {sut1}");
105     Console.WriteLine($"Pearl: {sut2}");
106 }
107
108 [Test]
109 public void Test_LocationLocatesPlayerItem()
110 {
111     var sut1 = garden.Locate("sword");
112     var sut2 = garden.Locate("shovel");
113     var sut3 = garden.Locate("pickaxe");
114
115     Assert.Multiple(() =>
116     {
117         Assert.That(sut1, Is.Null);
118         Assert.That(sut2, Is.Null);
119         Assert.That(sut3, Is.Null);
120     });
121     if(sut1 == null && sut2 == null && sut3 == null)
122     {
123         Console.WriteLine("Sword, shovel, and pickaxe is not found");
124     }
125 }
126
127 [Test]
128 public void Test_LocatesLocationItemInBag()
129 {
130     _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and
↪ stiched with leather.");
131     gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem
↪ that can only be obtained through trade");
132     _bag1.Inventory.Put(gem);
133     _player.Inventory.Put(_bag1);
134
135     string command1 = "look at water in bag1";
136     string command2 = "look at pearl in bag1";
137     string[] array1 = command1.Split(' ');
138     string[] array2 = command2.Split(' ');
139
140     var sut1 = look.Execute(_player, array1);
141     var sut2 = look.Execute(_player, array2);
142
143     Assert.Multiple(() =>
144     {
145         Assert.That(sut1, Is.EqualTo($"I can't find the {water.FirstId} in
↪ {_bag1.Name}"));
146         Assert.That(sut2, Is.EqualTo($"I can't find the {pearl.FirstId} in
↪ {_bag1.Name}"));
147     });
148
149     Console.WriteLine(sut1);
```

```
150         Console.WriteLine(sut2);
151     }
152
153     [Test]
154     public void Test_LocatesBagItemInLocation()
155     {
156         _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and
↵ stiched with leather.");
157         gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem
↵ that can only be obtained through trade");
158         _bag1.Inventory.Put(gem);
159         _player.Inventory.Put(_bag1);
160
161         var sut = garden.Locate("gem");
162         Assert.That(sut, Is.Null);
163         if (sut == null)
164         {
165             Console.WriteLine("Item was not found");
166         }
167     }
168 }
169 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Xml.Linq;
7
8  namespace MazeGame
9  {
10     public class Player : GameObject, IHaveInventory
11     {
12         private Inventory _inventory;
13         private Location _location;
14
15         public Player(string name, string desc) : base(new string[] { "me",
↵ "inventory" }, name, desc )
16         {
17             _inventory = new Inventory();
18             _location = null;
19         }
20
21         public GameObject Locate(string id)
22         {
23             List<GameObject> gameOBJ = new List<GameObject>();
24
25             if (id == "me" || id == "inventory")
26             {
27                 gameOBJ.Add(this);
28             }
29             else if (_inventory.HasItem(id))
30             {
31                 var item = _inventory.Fetch(id);
32                 gameOBJ.Add(item);
33             }
34             else if (!_inventory.HasItem(id))
35             {
36                 var item = _location.Locate(id);
37                 gameOBJ.Add(item);
38             }
39             else
40             {
41                 Item nullObj = null;
42                 gameOBJ.Add(nullObj);
43             }
44
45             var result = gameOBJ.ElementAt(0);
46             gameOBJ.Clear();
47             return result;
48         }
49
50         public string ChangeLocation(Location place)
51         {
52             if (place.AreYou(place.FirstId))
```



```
53         {
54             _location = place;
55             return $"You have arrived at {_location.Name}";
56         }
57         else
58         {
59             return "I don't know where that is";
60         }
61     }
62
63     public override string FullDescription
64     {
65         get
66         {
67             return
68                 $"You are {Name} {Description}\n" +
69                 $"You are carrying: \n{_inventory.ItemList}";
70         }
71     }
72
73     public Inventory Inventory
74     {
75         get { return _inventory; }
76     }
77
78     public Location Location
79     {
80         get { return _location; }
81     }
82
83 }
84 }
```

```
1 namespace MazeGame.nUnitTests
2 {
3     public class PlayerTests
4     {
5         private Player _player { get; set; } = null!;
6         private Item sword { get; set; } = null!;
7         private Item shovel { get; set; } = null!;
8         private Item pickaxe { get; set; } = null!;
9         private Location garden { get; set; } = null!;
10
11         [SetUp]
12         public void SetUp()
13         {
14             _player = new Player("Hoang An", "the comtemplator of infinity");
15             sword = new Item(new string[] { "sword" }, "a bronze sword", "A short
↪ sword cast from bronze");
16             shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable
↪ shovel borrowed from the village");
17             pickaxe = new Item(new string[] { "pickaxe" }, "an obsidian pickaxe", "A
↪ pickaxe made of obsidian");
18             _player.Inventory.Put(sword);
19             _player.Inventory.Put(shovel);
20             _player.Inventory.Put(pickaxe);
21             garden = new Location(new string[] { "garden" }, "green garden", "A
↪ garden blooming with natural plants, trees, and flowers");
22             _player.ChangeLocation(garden);
23         }
24
25         [Test]
26         public void Test_Identifiable()
27         {
28             string id_ME = "me";
29             var sut1 = _player.AreYou(id_ME);
30             string id_INV = "inventory";
31             var sut2 = _player.AreYou(id_INV);
32             Assert.That(sut1, Is.EqualTo(true));
33             Assert.That(sut2, Is.EqualTo(true));
34         }
35
36         [Test]
37         public void Test_LocateItems()
38         {
39             string sampleID = "sword";
40             var sut = _player.Locate(sampleID);
41             Assert.That(sut.FirstId, Is.EqualTo(sampleID));
42             Console.WriteLine(sut.FullDescription);
43         }
44
45         [Test]
46         public void Test_LocateItself()
47         {
48             string id_ME = "me";
49             var sut1 = _player.Locate(id_ME);
```

```
50         string id_INV = "inventory";
51         var sut2 = _player.Locate(id_INV);
52         Assert.IsNotNull(sut1);
53         Assert.IsNotNull(sut2);
54         Console.WriteLine($"Player (me): {sut1.ShortDescription}");
55         Console.WriteLine();
56         Console.WriteLine("Inventory (inventory): \n" + sut2.FullDescription);
57     }
58
59     [Test]
60     public void Test_LocateNothing()
61     {
62         string sampleID = "shoe";
63         var sut = _player.Locate(sampleID);
64         Assert.IsNull(sut);
65     }
66
67     [Test]
68     public void Test_FullDescription()
69     {
70         string sample = "You are Hoang An the comtemplator of infinity\n" +
71             "You are carrying: \n" +
72             "a bronze sword (sword)\n" +
73             "a shovel (shovel)\n" +
74             "an obsidian pickaxe (pickaxe)";
75         var sut = _player.FullDescription;
76         Assert.IsNotNull(sut);
77         Assert.That(sut, Is.EqualTo(sample));
78         Console.WriteLine($"Sample return: \n" +
79             $"{sample}");
80         Console.WriteLine();
81         Console.WriteLine($"Test return: \n" +
82             $"{sut}");
83     }
84 }
85 }
```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MazeGame
8  {
9      public class LookCommand : Command
10     {
11         public LookCommand(string[] ids) : base(ids)
12         {
13
14         }
15
16         public override string Execute(Player p, string[] text)
17         {
18             IHaveInventory container;
19             string itemID;
20
21             if (text[0] == "look")
22             {
23                 if (text.Length == 3 || text.Length == 5)
24                 {
25                     if (text[1] == "at")
26                     {
27                         if (text.Length == 3)
28                         {
29                             container = p;
30                             itemID = text[2];
31
32                             return LookAtIn(itemID, container);
33                         }
34                         else if (text.Length == 5 && text[3] == "in")
35                         {
36                             string containerID = text[4];
37                             container = FetchContainer(p, containerID);
38                             string itmReturn;
39                             if (container != null)
40                             {
41                                 itemID = text[2];
42                                 itmReturn = LookAtIn(itemID, container);
43
44                                 if (itmReturn == ("I can't find the " + itemID))
45                                 {
46                                     return $"I can't find the {itemID} in
↪ {container.Name}";
47                                 }
48                                 else
49                                 {
50                                     return itmReturn;
51                                 }
52                             }
53                         }
54                     }
55                 }
56             }
57         }
58     }
59 }

```

```
53         else
54         {
55             return "I can't find the " + containerID;
56         }
57     }
58     else
59     {
60         return "What do you want to looking in?";
61     }
62 }
63 else
64 {
65     return "What do you want to look at?";
66 }
67 }
68 else
69 {
70     return "I don't know how to look like that";
71 }
72 }
73 else if (text[0] == "Look")
74 {
75     return p.Location.FullDescription;
76 }
77 else
78 {
79     return "Error in look input";
80 }
81 }
82
83 private IHaveInventory FetchContainer (Player p, string containerID)
84 {
85     var result = p.Locate(containerID);
86     return (IHaveInventory)result;
87 }
88
89 private string LookAtIn(string thingId, IHaveInventory container)
90 {
91     var result = container.Locate(thingId);
92
93     if(result != null)
94     {
95         return result.FullDescription;
96     }
97     else
98     {
99         return "I can't find the " + thingId;
100     }
101 }
102 }
103 }
```

```

1 namespace MazeGame.nUnitTests
2 {
3     public class LookCommandTests
4     {
5         private Player _player { get; set; } = null!;
6         private Item sword { get; set; } = null!;
7         private Item shovel { get; set; } = null!;
8         private Item knife { get; set; } = null!;
9         private Item gem { get; set; } = null!;
10        private Bag _bag1 { get; set; } = null!;
11        private LookCommand look { get; set; } = null!;
12        private Location garden { get; set; } = null!;
13
14        [SetUp]
15        public void SetUp()
16        {
17            _player = new Player("Hoang An", "the comtemplator of infinity");
18            sword = new Item(new string[] { "sword" }, "a bronze sword", "A short
↪ sword cast from bronze");
19            shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable
↪ shovel borrowed from the village");
20            knife = new Item(new string[] { "knife" }, "an obsidian knife", "A knife
↪ made of obsidian");
21            _player.Inventory.Put(sword);
22            _player.Inventory.Put(shovel);
23            _player.Inventory.Put(knife);
24            look = new LookCommand(new string[] { "look" });
25            garden = new Location(new string[] { "garden" }, "green garden", "A
↪ garden blooming with natural plants, trees, and flowers");
26            _player.ChangeLocation(garden);
27        }
28
29        [Test]
30        public void Test_LookAtMe()
31        {
32
33            string command = "look at me";
34            string[] array = command.Split(' ');
35            var sut = look.Execute(_player, array);
36            Assert.Multiple(() =>
37            {
38                Assert.IsNotNull(sut);
39                Assert.That(sut, Is.EqualTo(_player.FullDescription));
40            });
41
42            Console.WriteLine(sut.ToString());
43        }
44
45        [Test]
46        public void Test_LookAtGem()
47        {
48            gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem
↪ that can only be obtained through trade");

```

```

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

```

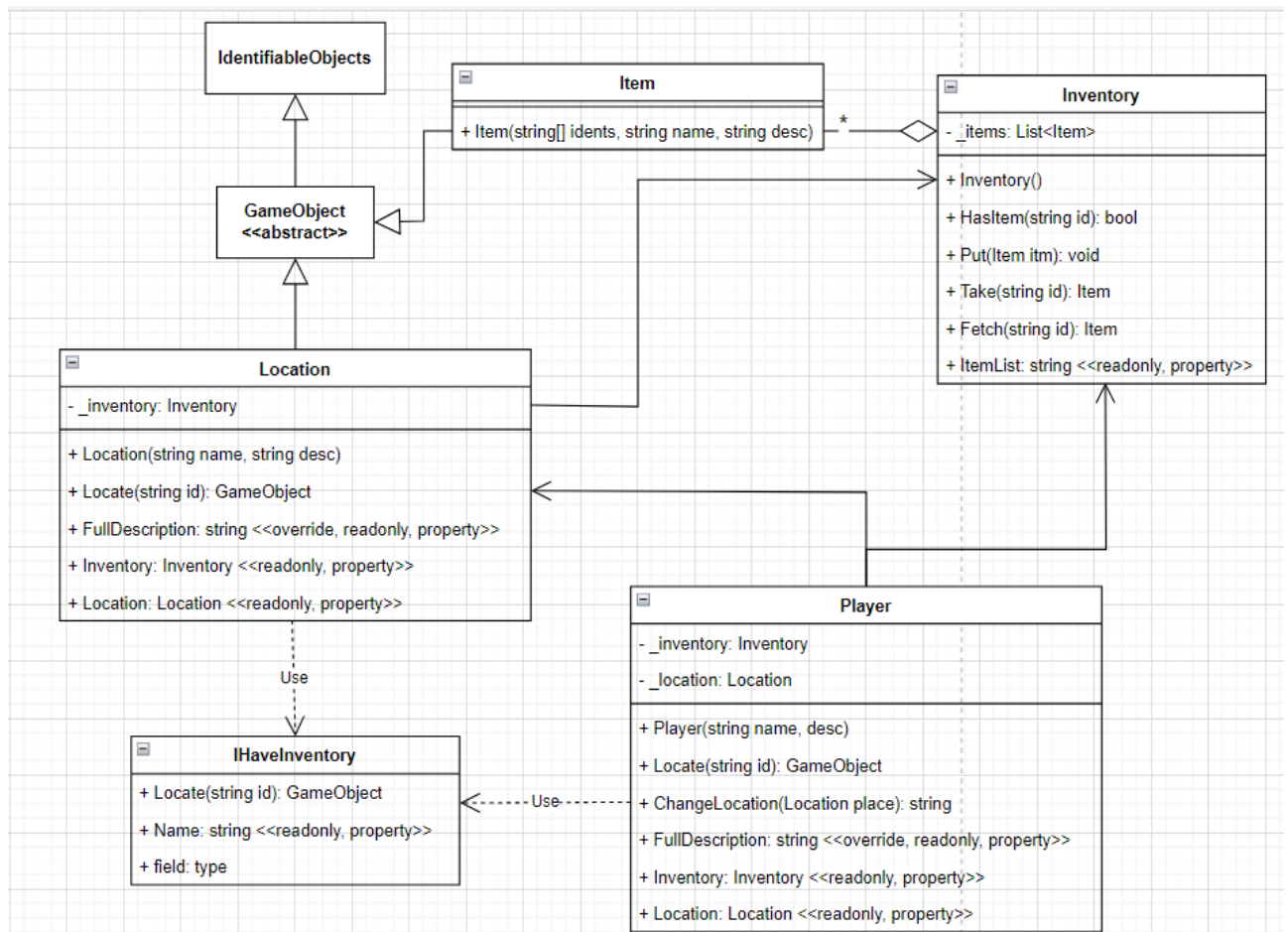
```

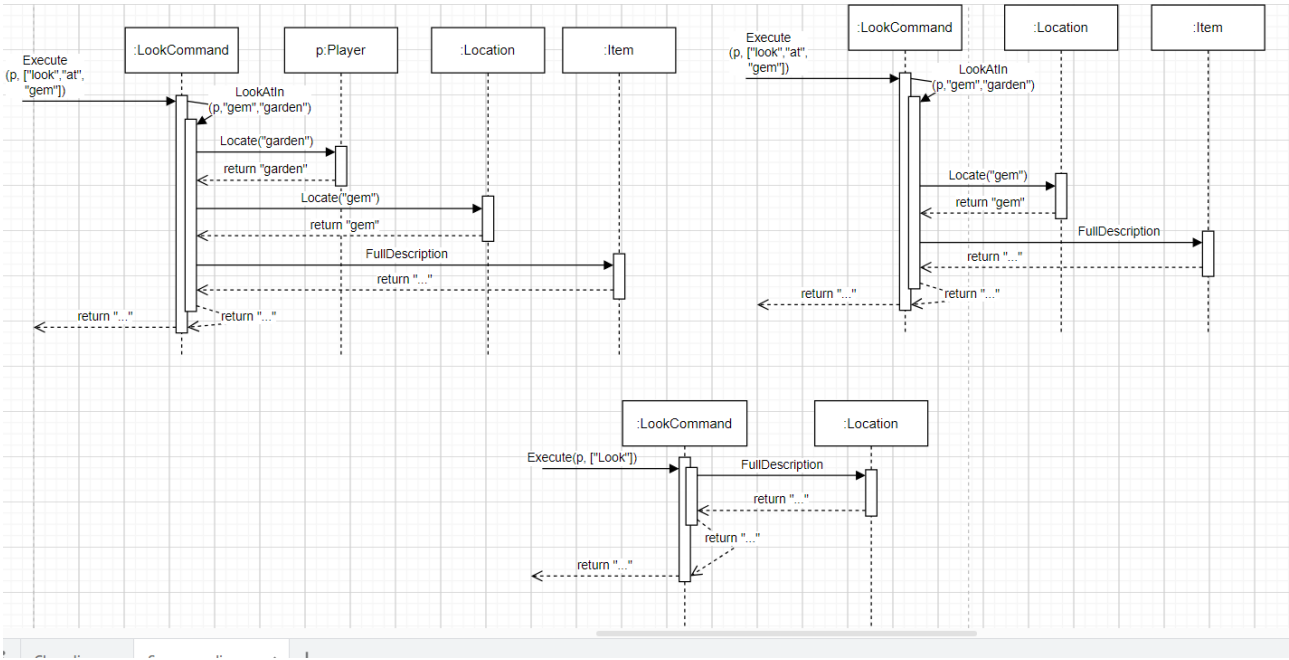
99         string command = "look at gem in bag1";
100         string[] array = command.Split(' ');
101         var sut = look.Execute(_player, array);
102         Assert.Multiple(() =>
103         {
104             Assert.IsNotNull(sut);
105             Assert.That(sut, Is.EqualTo(gem.FullDescription));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

```

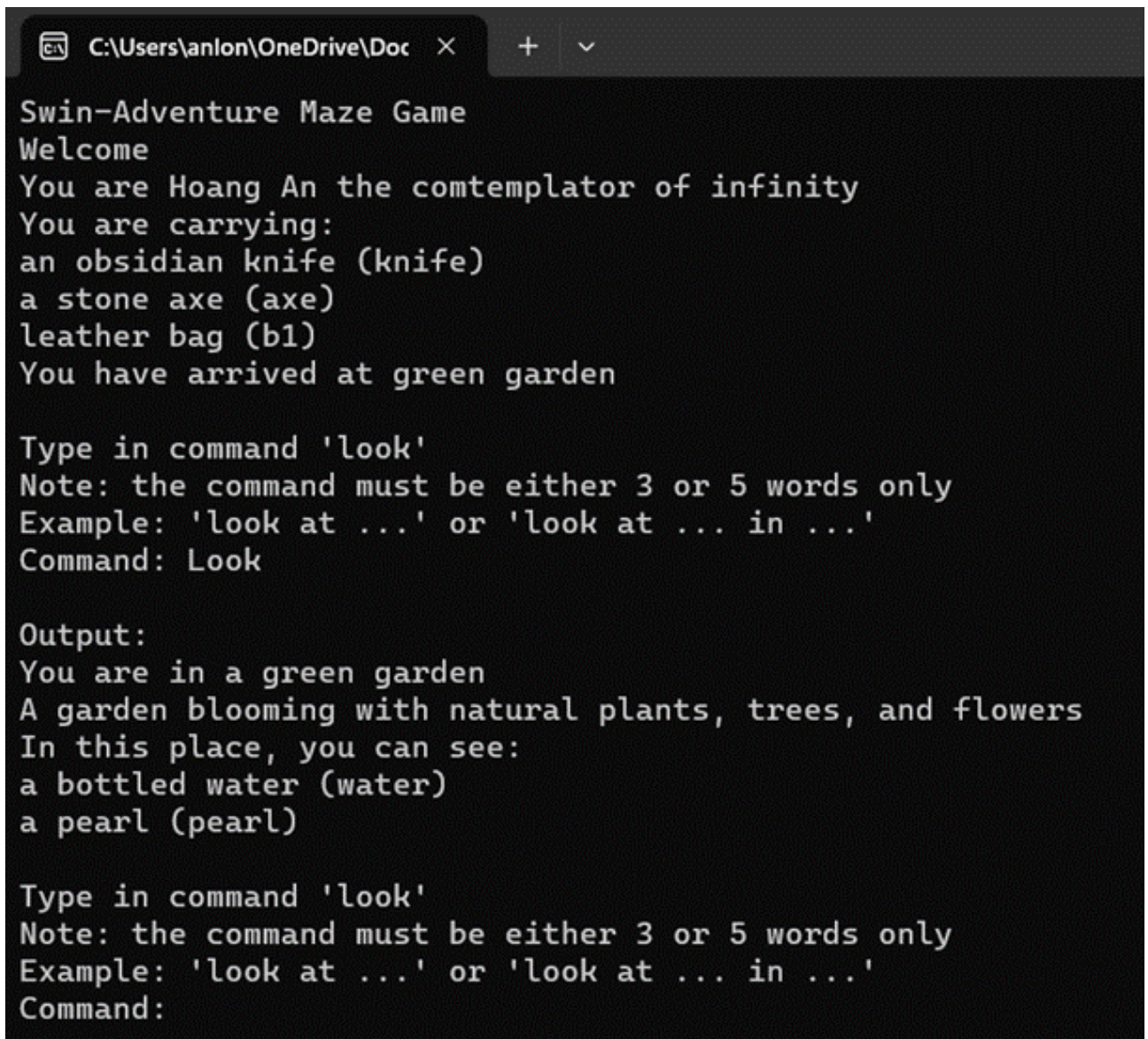


```
149     [Test]
150     public void Test_InvalidLook()
151     {
152         string command = "hi there";
153         string command2 = "look in";
154         string command3 = "look in here";
155         string command4 = "look at this at here";
156
157         string[] array = command.Split(' ');
158         string[] array2 = command2.Split(' ');
159         string[] array3 = command3.Split(' ');
160         string[] array4 = command4.Split(' ');
161
162         var sut = look.Execute(_player, array);
163         var sut2 = look.Execute(_player, array2);
164         var sut3 = look.Execute(_player, array3);
165         var sut4 = look.Execute(_player, array4);
166
167         Assert.Multiple(() =>
168         {
169             Assert.That(sut, Is.EqualTo("Error in look input"));
170             Assert.That(sut2, Is.EqualTo("I don't know how to look like that"));
171             Assert.That(sut3, Is.EqualTo("What do you want to look at?"));
172             Assert.That(sut4, Is.EqualTo("What do you want to looking in?"));
173         });
174
175         Console.WriteLine();
176         Console.WriteLine("Lists of possible errors with wrong input");
177         Console.WriteLine(sut);
178         Console.WriteLine(sut2);
179         Console.WriteLine(sut3);
180         Console.WriteLine(sut4);
181     }
182 }
183 }
```





Test	Duration	Traits	Error Message	Group Summary
✔ MazeGame.nUnitTests (40)	146 ms			<div>LocationTests</div> <div>Tests in group: 8</div> <div>⌚ Total Duration: 61 ms</div> <div>Outcomes</div> <div>✔ 8 Passed</div>
✔ MazeGame.nUnitTests (40)	146 ms			
✔ BagTests (5)	29 ms			
✔ IdentifiableObjectTests (6)	13 ms			
✔ InventoryTests (5)	2 ms			
✔ ItemTests (3)	< 1 ms			
✔ LocationTests (8)	61 ms			
✔ Test_LocatesBagItemInLocation	13 ms			
✔ Test_LocatesLocationItemInBag	19 ms			
✔ Test_LocationIsIdentifiable	< 1 ms			
✔ Test_LocationLocatesItem	3 ms			
✔ Test_LocationLocatesPlayerItem	14 ms			
✔ Test_LocationLocatesUnkItem	12 ms			
✔ Test_LocationSelfLocate	< 1 ms			
✔ Test_PlayerLocatesItemInArea	< 1 ms			
✔ LookCommandTests (8)	26 ms			
✔ PlayerTests (5)	15 ms			



```
C:\Users\anlon\OneDrive\Doc × + v
Swin-Adventure Maze Game
Welcome
You are Hoang An the comtemplator of infinity
You are carrying:
an obsidian knife (knife)
a stone axe (axe)
leather bag (b1)
You have arrived at green garden

Type in command 'look'
Note: the command must be either 3 or 5 words only
Example: 'look at ...' or 'look at ... in ...'
Command: Look

Output:
You are in a green garden
A garden blooming with natural plants, trees, and flowers
In this place, you can see:
a bottled water (water)
a pearl (pearl)

Type in command 'look'
Note: the command must be either 3 or 5 words only
Example: 'look at ...' or 'look at ... in ...'
Command:
```