Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 6 - Locations

PDF generated at 02:46 on Friday 10th November, 2023

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

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

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

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

            if(_inventory.HasItem(id))
            {
                var itm = _inventory.Fetch(id);
                items.Add(itm);
            }
            else
            {
                Item nullObj = null;
                items.Add(nullObj);
            }

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

        public override string FullDescription
        {
            get
            {
                return $"You are in a {Name}\n" +
                    $"{Description}\n"+
                    $"In this place, you can see:\n"+
                    $"{Inventory.ItemList}";
            }
        }

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

```
    }
}
```

```
using System.ComponentModel;
using System.Diagnostics.Metrics;
using System.Security.Cryptography;

namespace MazeGame.nUnitTests
{
    public class LocationTests
    {
        private Location garden { get; set; } = null!;
        private Item water { get; set; } = null!;
        private Item pearl { get; set; } = null!;
        private Player _player { get; set; } = null!;
        private Item sword { get; set; } = null!;
        private Item shovel { get; set; } = null!;
        private Item pickaxe { get; set; } = null!;
        private LookCommand look { get; set; } = null!;
        private Bag _bag1 { get; set; } = null!;
        private Item gem { get; set; } = null!;

        [SetUp]
        public void SetUp()
        {
            garden = new Location(new string[] { "garden" }, "green garden", "A garden blooming with natural plants, trees, and flowers");
            water = new Item(new string[] { "water" }, "a bottled water", "A 1 Litres bottle of spring water to keep you hydrated");
            pearl = new Item(new string[] { "pearl" }, "a pearl", "A pearl picked from pearl tree. A fruit great for snack");
            garden.Inventory.Put(water);
            garden.Inventory.Put(pearl);
            _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");
            pickaxe = new Item(new string[] { "pickaxe" }, "an obsidian pickaxe", "A pickaxe made of obsidian");
            _player.ChangeLocation(garden);
            _player.Inventory.Put(sword);
            _player.Inventory.Put(shovel);
            _player.Inventory.Put(pickaxe);
            look = new LookCommand(new string[] { "look", "Look" });
        }

        [Test]
        public void Test_LocationIsIdentifiable()
        {
            var sut = garden.AreYou("garden");
            Assert.That(sut, Is.True);
            Console.WriteLine(sut);
        }

        [Test]
```

```
        public void Test_LocationSelfLocate()
        {
            string command = "Look";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
            {
                Assert.That(sut, Is.Not.Null);
                Assert.That(sut, Is.EqualTo(garden.FullDescription));
            });
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_LocationLocatesItem()
        {
            var sut1 = garden.Locate("water");
            var sut2 = garden.Locate("pearl");
            Assert.Multiple(() =>
            {
                Assert.That(sut1.Description, Is.EqualTo(water.Description));
                Assert.That(sut1.Description, Is.Not.Null);
                Assert.That(sut2.Description, Is.EqualTo(pearl.Description));
                Assert.That(sut2.Description, Is.Not.Null);
            });
            Console.WriteLine($"Water: {sut1.Description}");
            Console.WriteLine($"Pearl: {sut2.Description}");
        }

        [Test]
        public void Test_LocationLocatesUnkItem()
        {
            var sut = garden.Locate("chair");
            Assert.That(sut, Is.Null);
            if(sut == null)
            {
                Console.WriteLine("Item was not found");
            }
        }

        [Test]
        public void Test_PlayerLocatesItemInArea()
        {
            string command1 = "look at water";
            string command2 = "look at pearl";
            string[] array1 = command1.Split(' ');
            string[] array2 = command2.Split(' ');
            var sut1 = look.Execute(_player, array1);
            var sut2 = look.Execute(_player, array2);
            Assert.Multiple(() =>
            {
                Assert.That(sut1, Is.EqualTo(water.Description));
                Assert.That(sut1, Is.Not.Null);
```

```
                Assert.That(sut2, Is.EqualTo(pearl.Description));
                Assert.That(sut2, Is.Not.Null);
            });
            Console.WriteLine($"Water: {sut1}");
            Console.WriteLine($"Pearl: {sut2}");
        }

        [Test]
        public void Test_LocationLocatesPlayerItem()
        {
            var sut1 = garden.Locate("sword");
            var sut2 = garden.Locate("shovel");
            var sut3 = garden.Locate("pickaxe");

            Assert.Multiple(() =>
            {
                Assert.That(sut1, Is.Null);
                Assert.That(sut2, Is.Null);
                Assert.That(sut3, Is.Null);
            });
            if(sut1 == null && sut2 == null &&  sut3 == null)
            {
                Console.WriteLine("Sword, shovel, and pickaxe is not found");
            }
        }

        [Test]
        public void Test_LocatesLocationItemInBag()
        {
            _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and stiched with leather.");
            gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem that can only be obtained through trade");
            _bag1.Inventory.Put(gem);
            _player.Inventory.Put(_bag1);

            string command1 = "look at water in bag1";
            string command2 = "look at pearl in bag1";
            string[] array1 = command1.Split(' ');
            string[] array2 = command2.Split(' ');

            var sut1 = look.Execute(_player, array1);
            var sut2 = look.Execute(_player, array2);

            Assert.Multiple(() =>
            {
                Assert.That(sut1, Is.EqualTo($"I can't find the {water.FirstId} in {_bag1.Name}"));
                Assert.That(sut2, Is.EqualTo($"I can't find the {pearl.FirstId} in {_bag1.Name}"));
            });

            Console.WriteLine(sut1);
```

```
            Console.WriteLine(sut2);
        }

        [Test]
        public void Test_LocatesBagItemInLocation()
        {
            _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and stiched with leather.");
            gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem that can only be obtained through trade");
            _bag1.Inventory.Put(gem);
            _player.Inventory.Put(_bag1);

            var sut = garden.Locate("gem");
            Assert.That(sut, Is.Null);
            if (sut == null)
            {
                Console.WriteLine("Item was not found");
            }
        }
    }
}
```

```
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;
        private Location _location;

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

        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 if (!_inventory.HasItem(id))
            {
                var item = _location.Locate(id);
                gameOBJ.Add(item);
            }
            else
            {
                Item nullObj = null;
                gameOBJ.Add(nullObj);
            }

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

        public string ChangeLocation(Location place)
        {
            if (place.AreYou(place.FirstId))
```

```
            {
                _location = place;
                return $"You have arrived at {_location.Name}";
            }
            else
            {
                return "I don't know where that is";
            }
        }

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

        public Inventory Inventory
        {
            get { return _inventory; }
        }

        public Location Location
        {
            get { return _location; }
        }

    }
}
```

```
namespace MazeGame.nUnitTests
{
    public class PlayerTests
    {
        private Player _player { get; set; } = null!;
        private Item sword { get; set; } = null!;
        private Item shovel { get; set; } = null!;
        private Item pickaxe { get; set; } = null!;
        private Location garden { 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");
            pickaxe = new Item(new string[] { "pickaxe" }, "an obsidian pickaxe", "A pickaxe made of obsidian");
            _player.Inventory.Put(sword);
            _player.Inventory.Put(shovel);
            _player.Inventory.Put(pickaxe);
            garden = new Location(new string[] { "garden" }, "green garden", "A garden blooming with natural plants, trees, and flowers");
            _player.ChangeLocation(garden);
        }

        [Test]
        public void Test_Identifiable()
        {
            string id_ME = "me";
            var sut1 = _player.AreYou(id_ME);
            string id_INV = "inventory";
            var sut2 = _player.AreYou(id_INV);
            Assert.That(sut1, Is.EqualTo(true));
            Assert.That(sut2, Is.EqualTo(true));
        }

        [Test]
        public void Test_LocateItems()
        {
            string sampleID = "sword";
            var sut = _player.Locate(sampleID);
            Assert.That(sut.FirstId, Is.EqualTo(sampleID));
            Console.WriteLine(sut.FullDescription);
        }

        [Test]
        public void Test_LocateItself()
        {
            string id_ME = "me";
            var sut1 = _player.Locate(id_ME);
```

```
            string id_INV = "inventory";
            var sut2 = _player.Locate(id_INV);
            Assert.IsNotNull(sut1);
            Assert.IsNotNull(sut2);
            Console.WriteLine($"Player (me): {sut1.ShortDescription}");
            Console.WriteLine();
            Console.WriteLine("Inventory (inventory): \n" + sut2.FullDescription);
        }

        [Test]
        public void Test_LocateNothing()
        {
            string sampleID = "shoe";
            var sut = _player.Locate(sampleID);
            Assert.IsNull(sut);
        }

        [Test]
        public void Test_FullDescription()
        {
            string sample = "You are Hoang An the comtemplator of infinity\n" +
                "You are carrying: \n" +
                "a bronze sword (sword)\n" +
                "a shovel (shovel)\n" +
                "an obsidian pickaxe (pickaxe)";
            var sut = _player.FullDescription;
            Assert.IsNotNull(sut);
            Assert.That(sut, Is.EqualTo(sample));
            Console.WriteLine($"Sample return: \n" +
                $"{sample}");
            Console.WriteLine();
            Console.WriteLine($"Test return: \n" +
                $"{sut}");
        }
    }
}
```

```
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;
                                }
                            }
```

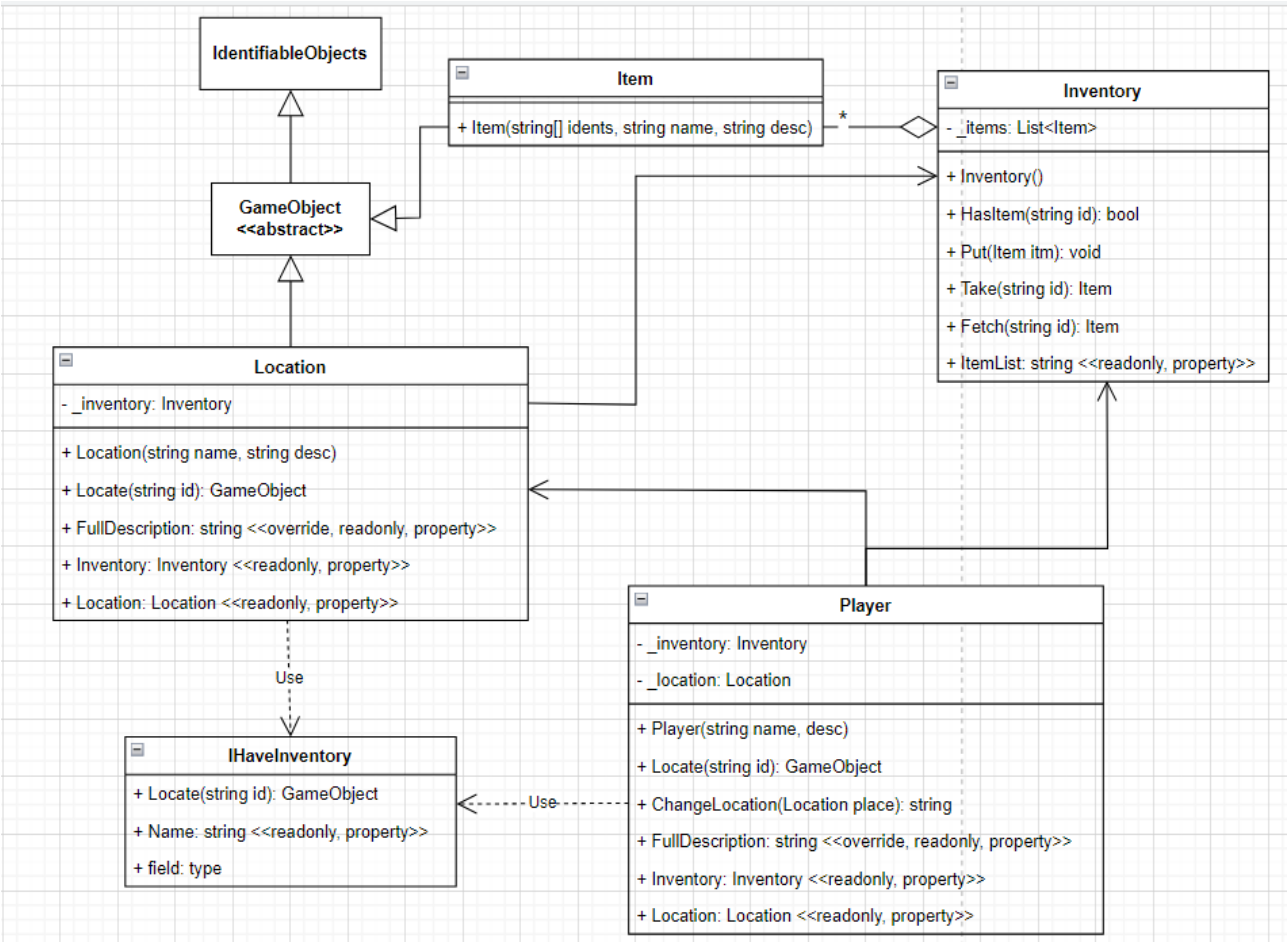
```
                                else
                                {
                                    return "I can't find the " + containerID;
                                }
                            }
                            else
                            {
                                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 if (text[0] == "Look")
                {
                    return p.Location.FullDescription;
                }
                else
                {
                    return "Error in look input";
                }
            }

            private IHaveInventory FetchContainer (Player p, string containerID)
            {
                var result = p.Locate(containerID);
                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;
                }
            }
        }
}
```

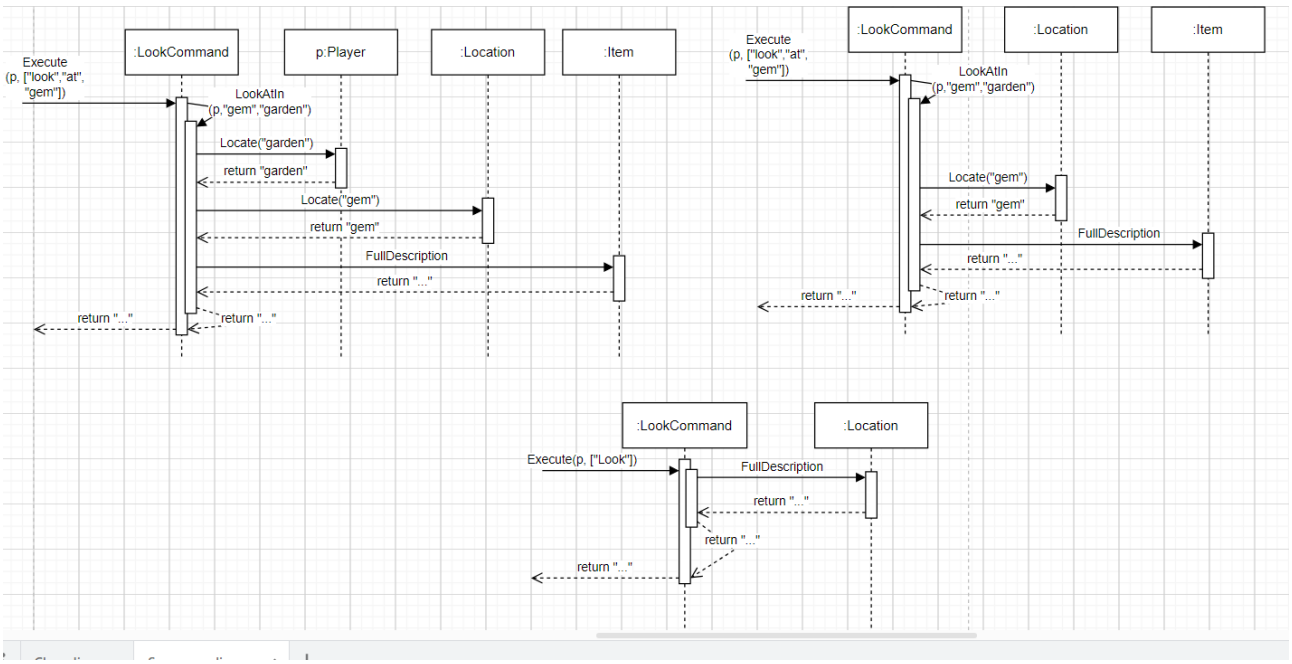
```
namespace MazeGame.nUnitTests
{
    public class LookCommandTests
    {
        private Player _player { get; set; } = null!;
        private Item sword { get; set; } = null!;
        private Item shovel { get; set; } = null!;
        private Item knife { get; set; } = null!;
        private Item gem { get; set; } = null!;
        private Bag _bag1 { get; set; } = null!;
        private LookCommand look { get; set; } = null!;
        private Location garden { 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"});
            garden = new Location(new string[] { "garden" }, "green garden", "A garden blooming with natural plants, trees, and flowers");
            _player.ChangeLocation(garden);
        }

        [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(() =>
            {
                Assert.IsNotNull(sut);
                Assert.That(sut, Is.EqualTo(gem.FullDescription));
            });
            Console.WriteLine();
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_LookAtUnk()
        {
            string command = "look at gem";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.That(sut, Is.EqualTo("I can't find the gem"));
            Console.WriteLine();
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_LookAtGemInMe()
        {
            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 in inventory";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
            {
                Assert.IsNotNull(sut);
                Assert.That(sut, Is.EqualTo(gem.FullDescription));
            });
            Console.WriteLine();
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_LookAtGemInBag()
        {
            _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and stiched with leather.");
            gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem that can only be obtained through trade");
            _bag1.Inventory.Put(gem);
            _player.Inventory.Put(_bag1);

```

```
            string command = "look at gem in bag1";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
            {
                Assert.IsNotNull(sut);
                Assert.That(sut, Is.EqualTo(gem.FullDescription));
            });
            Console.WriteLine();
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_LookAtGemInNoBag()
        {
            _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and stiched with leather.");
            gem = new Item(new string[] { "gem" }, "a green gem", "A rare type of gem that can only be obtained through trade");
            _bag1.Inventory.Put(gem);
            _player.Inventory.Put(_bag1);

            string command = "look at gem in bag2";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
            {
                Assert.IsNotNull(sut);
                Assert.That(sut, Is.EqualTo("I can't find the bag2"));
            });
            Console.WriteLine();
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_LookAtNoGemInBag()
        {
            _bag1 = new Bag(new string[] { "bag1" }, "brown bag", "a bag made and stiched with leather.");
            _player.Inventory.Put(_bag1);

            string command = "look at gem in bag1";
            string[] array = command.Split(' ');
            var sut = look.Execute(_player, array);
            Assert.Multiple(() =>
            {
                Assert.IsNotNull(sut);
                Assert.That(sut, Is.EqualTo("I can't find the gem in brown bag"));
            });
            Console.WriteLine();
            Console.WriteLine(sut);
        }

```

```
        [Test]
        public void Test_InvalidLook()
        {
            string command = "hi there";
            string command2 = "look in";
            string command3 = "look in here";
            string command4 = "look at this at here";

            string[] array = command.Split(' ');
            string[] array2 = command2.Split(' ');
            string[] array3 = command3.Split(' ');
            string[] array4 = command4.Split(' ');

            var sut = look.Execute(_player, array);
            var sut2 = look.Execute(_player, array2);
            var sut3 = look.Execute(_player, array3);
            var sut4 = look.Execute(_player, array4);

            Assert.Multiple(() =>
            {
                Assert.That(sut, Is.EqualTo("Error in look input"));
                Assert.That(sut2, Is.EqualTo("I don't know how to look like that"));
                Assert.That(sut3, Is.EqualTo("What do you want to look at?"));
                Assert.That(sut4, Is.EqualTo("What do you want to looking in?"));
            });

            Console.WriteLine();
            Console.WriteLine("Lists of possible errors with wrong input");
            Console.WriteLine(sut);
            Console.WriteLine(sut2);
            Console.WriteLine(sut3);
            Console.WriteLine(sut4);
        }
    }
}
```


IdentifiableObjects
Item
+ Item(string[] idents, string name, string desc)
*
Inventory
- _items: List<Item>
+ Inventory()
+ HasItem(string id): bool
+ Put(Item itm): void
+ Take(string id): Item
+ Fetch(string id): Item
+ ItemList: string <<readonly, property>>
GameObject
<<abstract>>
Location
- _inventory: Inventory
+ Location(string name, string desc)
+ Locate(string id): GameObject
+ FullDescription: string <<override, readonly, property>>
+ Inventory: Inventory <<readonly, property>>
+ Location: Location <<readonly, property>>
Use
IHaveInventory
+ Locate(string id): GameObject
+ Name: string <<readonly, property>>
+ field: type
Use
Player
- _inventory: Inventory
- _location: Location
+ Player(string name, desc)
+ Locate(string id): GameObject
+ ChangeLocation(Location place): string
+ FullDescription: string <<override, readonly, property>>
+ Inventory: Inventory <<readonly, property>>
+ Location: Location <<readonly, property>>


:LookCommand
p:Player
:Location
:Item
Execute
(p, ["look","at",
"gem"])
LookAtIn
(p,"gem","garden")
Locate("garden")
return "garden"
Locate("gem")
return "gem"
FullDescription
return "..."
return "..."
return "..."
:LookCommand
:Location
:Item
Execute
(p, ["look","at",
"gem"])
LookAtIn
(p,"gem","garden")
Locate("gem")
return "gem"
FullDescription
return "..."
return "..."
return "..."
:LookCommand
:Location
Execute(p, ["Look"])
FullDescription
return "..."
return "..."
return "..."

Test	Duration	Traits	Error Message
MazeGame.nUnitTests (40)	146 ms		
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		

Group Summary

LocationTests

Tests in group: 8

Total Duration: 61 ms

Outcomes

8 Passed

```
C:\Users\anlon\OneDrive\Doc

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:
```