Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 6 - Locations

PDF generated at 18:07 on Thursday 9th November, 2023

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

namespace SwinAdventure
{
    public class Location : GameObject,IHaveInventory
    {
        private Inventory _inventory = new Inventory();

        public Location(string name, string desc) : base(new string[] {"location"},
↪   name, desc)
        {

        }

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

        public override string Description
        {
            get
            {
                return Name + _desc + /*Description +*/ ":" + "\n" +
↪   _inventory.ItemList;
            }
        }

        public GameObject Locate(string id)
        {
            if (AreYou(id) == true)
            {
                return this;
            }
            return _inventory.Fetch(id);
        }
    }
}
```

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

namespace SwinAdventure
{
    public class LocationTesting
    {
        Player p = new Player("Binh", "A man not a god");
        Location l = new Location("Bed", "Binh old but comfy bed");
        Item ipad = new Item(new string[] { "ipad" }, "ipad pro M1", "A ipad pro
    ↪   released in 2020");

        [Test]
        public void TestNotLocation()
        {
            p.Location = l;
            bool actual = l.AreYou("hi");
            Assert.IsFalse(actual);
        }

        [Test]
        public void TestPLayerHasLocation()
        {
            p.Location = l;
            GameObject expect = l;
            GameObject actual = p.Locate("location");
            Assert.AreEqual(actual, expect);
        }

        [Test]
        public void TestLocationLocateItem()
        {
            l.Inventory.Put(ipad);
            p.Location = l;
            GameObject expect = ipad;
            GameObject actual = l.Locate("ipad");
            Assert.AreEqual(actual, expect);
        }

        [Test]
        public void TestEmptyLocation()
        {
            Assert.That(l.Locate("Vsmart"), Is.EqualTo(null));
        }
    }
}
```

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

namespace SwinAdventure
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory = new Inventory();
        private Location _location;
        public Player(string name, string desc) : base(new string[] { "me",
 ↪  "inventory"}, name, desc)
        {
        }
        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

        public override string Description
        {
            get
            {
                return Name + " you have:\n" + _inventory.ItemList;
            }
        }
        public GameObject Locate(string id)
        {
            if (AreYou(id) == true)
            {
                return this;
            }

            GameObject interactable = _inventory.Fetch(id);

            if (interactable != null)
            {
                return interactable;
            }

            if (_location != null)
            {
                interactable = _location.Locate(id);
                return interactable;
            }
            else
            {
                return null;
            }
        }

        public Location Location
        {
            get
            {
                return _location;
            }
            set
            {
                _location = value;
            }
        }
    }
}
```

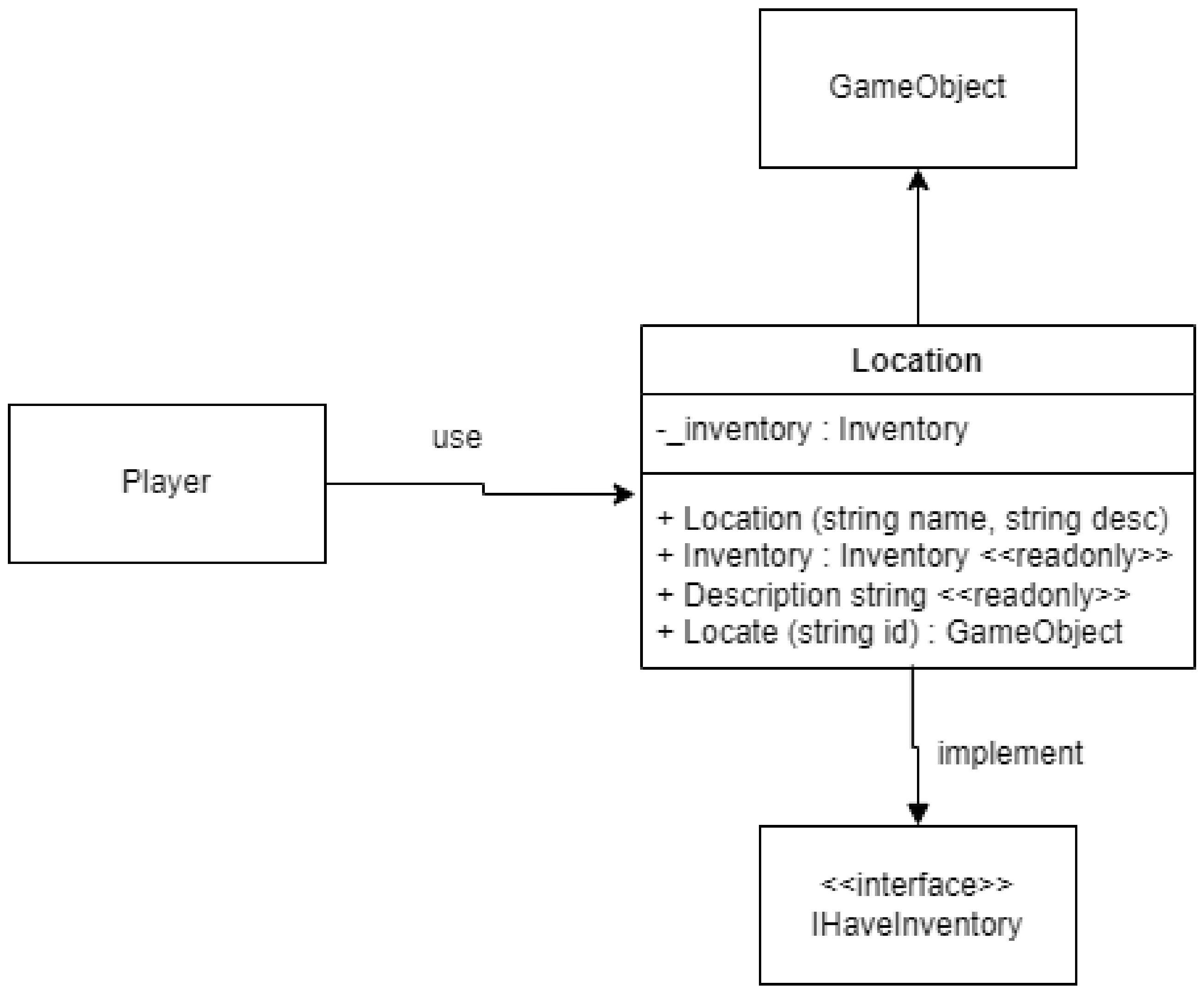
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class TestPlayer
    {
        Player player = new Player("Binh", "Nepenthes poacher");
        Item club = new Item(new string[] { "club" }, "a club", "This is the
    ↪   LEGENDARY BorneBeast iClub");
        Item sword = new Item(new string[] { "sword" }, "a sword", "This is a rusty
    ↪   sword");


        [Test]
        public void TestPlayerIdentifiable()
        {
            Assert.IsTrue(player.AreYou("me") && player.AreYou("inventory"));

        }

        [Test]
        public void TestPlayerLocateItems()
        {
            var result = false;
            player.Inventory.Put(sword);
            var iLocate = player.Locate("sword");
            if (sword == iLocate)
            {
                result = true;
            }
            Assert.IsTrue(result);
        }

        [Test]
        public void TestPlayerLocateItself()
        {
            var me = player.Locate("me");
            var inventory = player.Locate("inventory");
            var result = false;

            if (me == player)
            {
                result = true;
            }
            Assert.IsTrue(result);
            if (inventory == player)
            {
                result = true;
            }
            Assert.IsTrue(result);
        }

        [Test]
        public void TestPlayerLocateNothing()
        {
            var me = player.Locate("Hi");
            Assert.IsNull(me);
        }

        [Test]
        public void TestPlayerFullDescription()
        {
            player.Inventory.Put(sword);
            player.Inventory.Put(club);
            string expected = "Binh you have:\na sword: sword\na club: club\n";
            Assert.AreEqual(player.Description, expected);
        }
    }
}
```

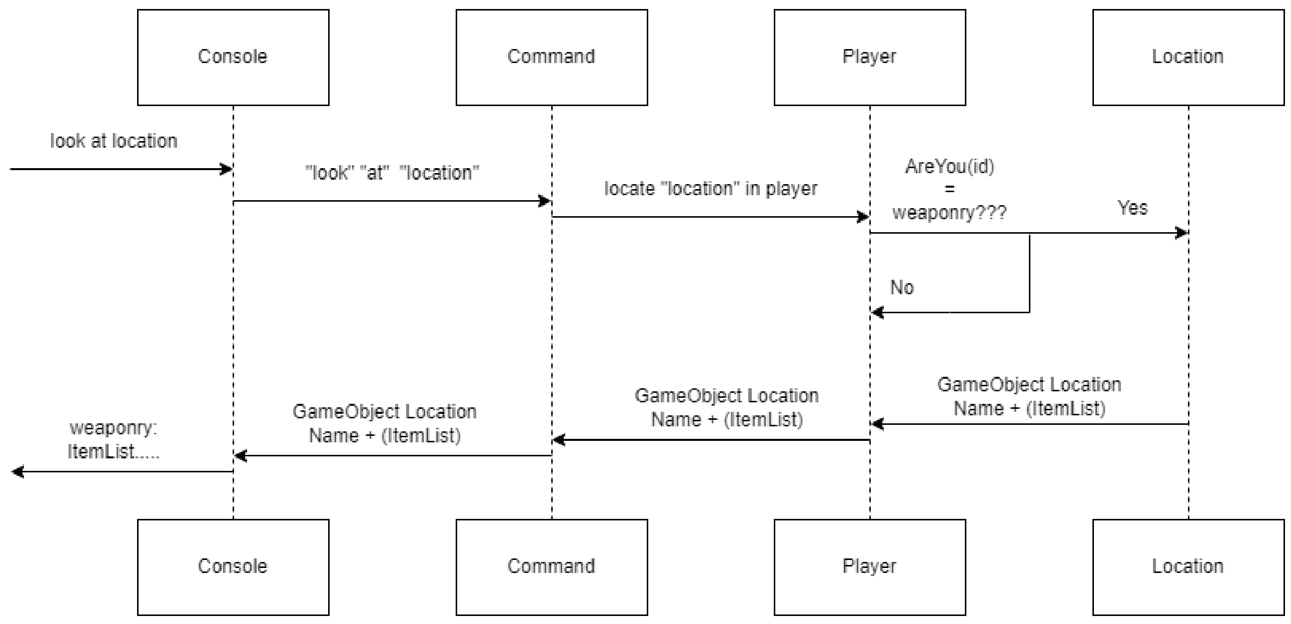
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace SwinAdventure
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {
        }
        public override string Execute(Player p, string[] text)
        {
            IHaveInventory _container;
            string _itemID;

            if (text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look like that";
            }

            if (text[0] != "look")
            {
                return "Error in look input";
            }

            if (text[1] != "at")
            {
                return "What do you want to look at?";
            }

            switch (text.Length)
            {
                case 3:
                    _container = p;
                    break;
                case 5:
                    if (text[3] != "in")
                    {
                        return "What do you want to look in?";
                    }
                    _container = FetchContainer(p, text[4]);
                    if (_container == null)
                    {
                        return $"I can't find the {text[4]}";
                    }
                    break;
                default:
```

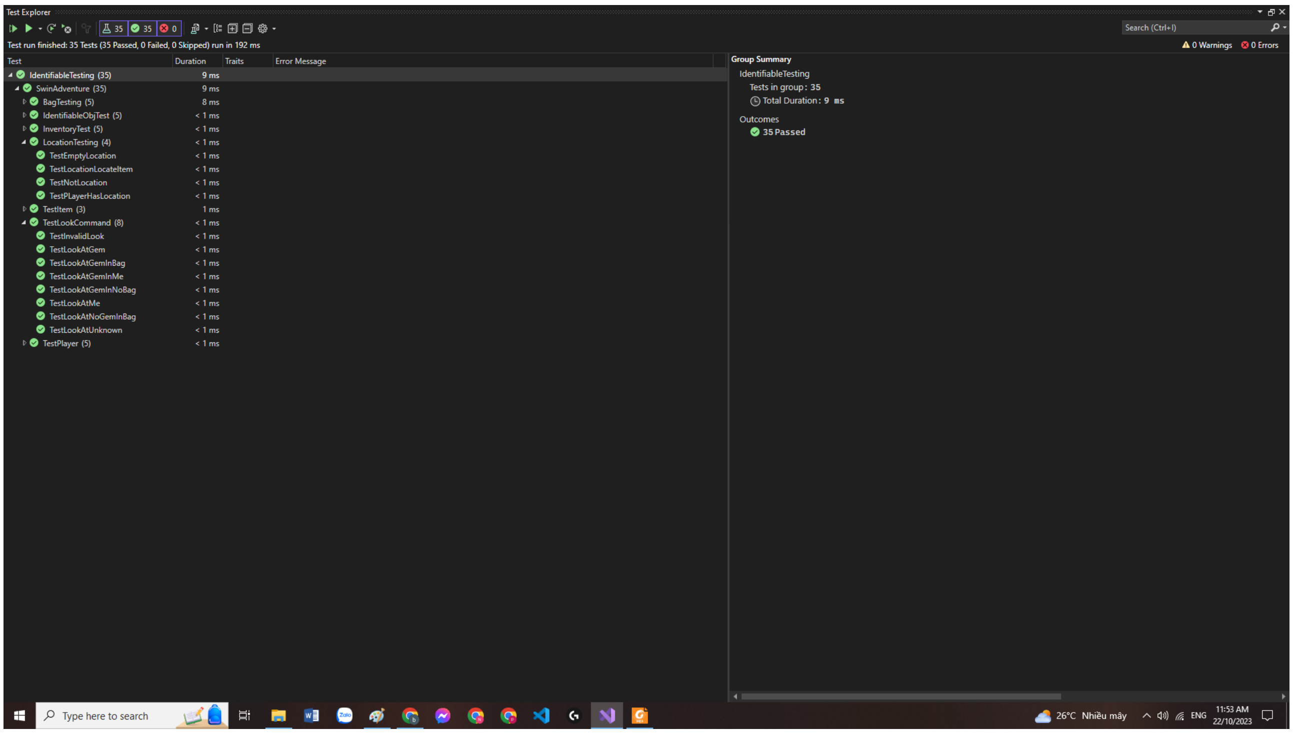
```
                return "Something wrong with the input length";
            }
            _itemID = text[2];
            return LookAtIn(_itemID,  _container);
        }
        private IHaveInventory FetchContainer(Player p, string containerId)
        {
            return p.Locate(containerId) as IHaveInventory;
        }
        private string LookAtIn(string thingId, IHaveInventory container)
        {
            if (container.Locate(thingId) == null)
            {
                return $"I can't find the {thingId}";
            }
            else
            {
                return container.Locate(thingId).Description;
            }
        }
    }
}
```

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

namespace SwinAdventure
{
    [TestFixture]
    public class TestLookCommand
    {
        LookCommand look;
        Player p;
        Bag bag;
        Item gem;

        [SetUp]
        public void Setup()
        {
            look = new LookCommand();
            p = new Player("lickmya707", "the gamer");
            bag = new Bag(new string[] { "bag" }, "bag", "This is the legendary space
    ↪ bag in eastern legend");
            gem = new Item(new string[] { "gem" }, "cool gem", "the reality stone,
    ↪ one of the infinity stones");
        }


        [Test]
        public void TestLookAtMe()
        {
            string command = look.Execute(p, new string[] { "look", "at", "inventory"
    ↪ });
            string expected = "lickmya707 you have:\n";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtGem()
        {
            p.Inventory.Put(gem);
            string command = look.Execute(p, new string[] { "look", "at", "gem" });
            string expected = "the reality stone, one of the infinity stones";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtUnknown()
        {
            string command = look.Execute(p, new string[] { "look", "at", "gem" });
            string expected = "I can't find the gem";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtGemInMe()
        {
            p.Inventory.Put(gem);
            string command = look.Execute(p, new string[] { "look", "at", "gem",
    ↪  "in", "inventory" });
            string expected = "the reality stone, one of the infinity stones";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtGemInBag()
        {
            bag.Inventory.Put(gem);
            p.Inventory.Put(bag);
            string command = look.Execute(p, new string[] { "look", "at", "gem",
    ↪  "in", "bag" });
            string expected = "the reality stone, one of the infinity stones";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtGemInNoBag()
        {
            bag.Inventory.Put(gem);
            string command = look.Execute(p, new string[] { "look", "at", "gem",
    ↪  "in", "bag" });
            string expected = "I can't find the bag";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtNoGemInBag()
        {
            p.Inventory.Put(bag);
            string command = look.Execute(p, new string[] { "look", "at", "gem",
    ↪  "in", "bag" });
            string expected = "I can't find the gem";
            Assert.That(command, Is.EqualTo(expected));
        }

        [Test]
        public void TestInvalidLook()
        {
            string command = look.Execute(p, new string[] { "look", "around" });
            Assert.That(command, Is.EqualTo("I don't know how to look like that"));

            string expected = look.Execute(p, new string[] { "hello" });
            Assert.That(expected, Is.EqualTo("I don't know how to look like that"));

            string command1 = look.Execute(p, new string[] { "look", "at", "a", "at",
    ↪  "b" });
```

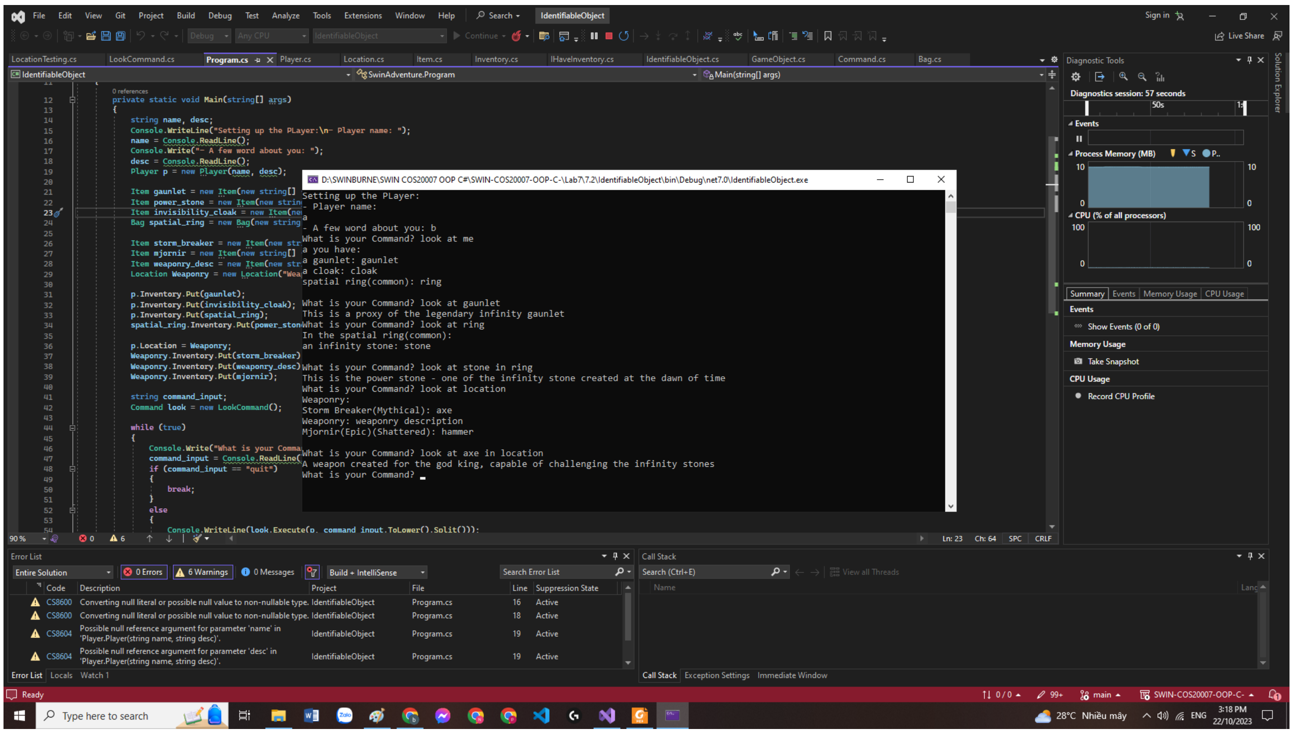
```
            Assert.That(command1, Is.EqualTo("What do you want to look in?"));
        }



    }
}
```


GameObject
Location
-_inventory : Inventory
+ Location (string name, string desc)
+ Inventory : Inventory <<readonly>>
+ Description string <<readonly>>
+ Locate (string id) : GameObject
Player
use
implement
<<interface>>
IHaveInventory


Console
Command
Player
Location
look at location
"look" "at" "location"
locate "location" in player
AreYou(id) = weaponry???
Yes
No
GameObject Location Name + (ItemList)
GameObject Location Name + (ItemList)
GameObject Location Name + (ItemList)
weaponry: ItemList.....
Console
Command
Player
Location



```
Test Explorer
35  35  0
Test run finished: 35 Tests (35 Passed, 0 Failed, 0 Skipped) run in 192 ms                 0 Warnings   0 Errors

Test                                  Duration   Traits   Error Message
IdentifiableTesting (35)                  9 ms
  SwinAdventure (35)                      9 ms
    BagTesting (5)                        8 ms
    IdentifiableObjTest (5)             < 1 ms
    InventoryTest (5)                   < 1 ms
    LocationTesting (4)                 < 1 ms
      TestEmptyLocation                 < 1 ms
      TestLocationLocateItem            < 1 ms
      TestNotLocation                   < 1 ms
      TestPLayerHasLocation             < 1 ms
    TestItem (3)                          1 ms
    TestLookCommand (8)                 < 1 ms
      TestInvalidLook                   < 1 ms
      TestLookAtGem                     < 1 ms
      TestLookAtGemInBag                < 1 ms
      TestLookAtGemInMe                 < 1 ms
      TestLookAtGemInNoBag              < 1 ms
      TestLookAtMe                      < 1 ms
      TestLookAtNoGemInBag              < 1 ms
      TestLookAtUnknown                 < 1 ms
    TestPlayer (5)                      < 1 ms

Group Summary
  IdentifiableTesting
    Tests in group: 35
    Total Duration: 9 ms

  Outcomes
    35 Passed
```



```
private static void Main(string[] args)
{
    string name, desc;
    Console.WriteLine("Setting up the PLayer:\n- Player name: ");
    name = Console.ReadLine();
    Console.Write("- A few word about you: ");
    desc = Console.ReadLine();
    Player p = new Player(name, desc);

    Item gauntlet = new Item(new string[]
    Item power_stone = new Item(new strin
    Item invisibility_cloak = new Item(ne
    Bag spatial_ring = new Bag(new string

    Item storm_breaker = new Item(new str
    Item mjornir = new Item(new string[]
    Item weaponry_desc = new Item(new str
    Location Weaponry = new Location("Wea

    p.Inventory.Put(gauntlet);
    p.Inventory.Put(invisibility_cloak);
    p.Inventory.Put(spatial_ring);
    spatial_ring.Inventory.Put(power_ston

    p.Location = Weaponry;
    Weaponry.Inventory.Put(storm_breaker)
    Weaponry.Inventory.Put(weaponry_desc)
    Weaponry.Inventory.Put(mjornir);

    string command_input;
    Command look = new LookCommand();

    while (true)
    {
        Console.Write("What is your Comma
        command_input = Console.ReadLine(
        if (command_input == "quit")
        {
            break;
        }
        else
        {
            Console.WriteLine(look.Execute(p, command_input.ToLower().Split()));
```

D:\SWINBURNE\SWIN COS20007 OOP C#\SWIN-COS20007-OOP-C-\Lab7\7.2\IdentifiableObject\bin\Debug\net7.0\IdentifiableObject.exe

```
Setting up the PLayer:
- Player name:
a
- A few word about you: b
What is your Command? look at me
a you have:
a gaunlet: gaunlet
a cloak: cloak
spatial ring(common): ring

What is your Command? look at gaunlet
This is a proxy of the legendary infinity gaunlet
What is your Command? look at ring
In the spatial ring(common):
an infinity stone: stone

What is your Command? look at stone in ring
This is the power stone - one of the infinity stone created at the dawn of time
What is your Command? look at location
Weaponry:
Storm Breaker(Mythical): axe
Weaponry: weaponry description
Mjornir(Epic)(Shattered): hammer

What is your Command? look at axe in location
A weapon created for the god king, capable of challenging the infinity stones
What is your Command? _
```

Error List — Entire Solution — 0 Errors — 6 Warnings — 0 Messages — Build + IntelliSense

Code	Description	Project	File	Line	Suppression State
CS8600	Converting null literal or possible null value to non-nullable type.	IdentifiableObject	Program.cs	16	Active
CS8600	Converting null literal or possible null value to non-nullable type.	IdentifiableObject	Program.cs	18	Active
CS8604	Possible null reference argument for parameter 'name' in 'Player.Player(string name, string desc)'.	IdentifiableObject	Program.cs	19	Active
CS8604	Possible null reference argument for parameter 'desc' in 'Player.Player(string name, string desc)'.	IdentifiableObject	Program.cs	19	Active