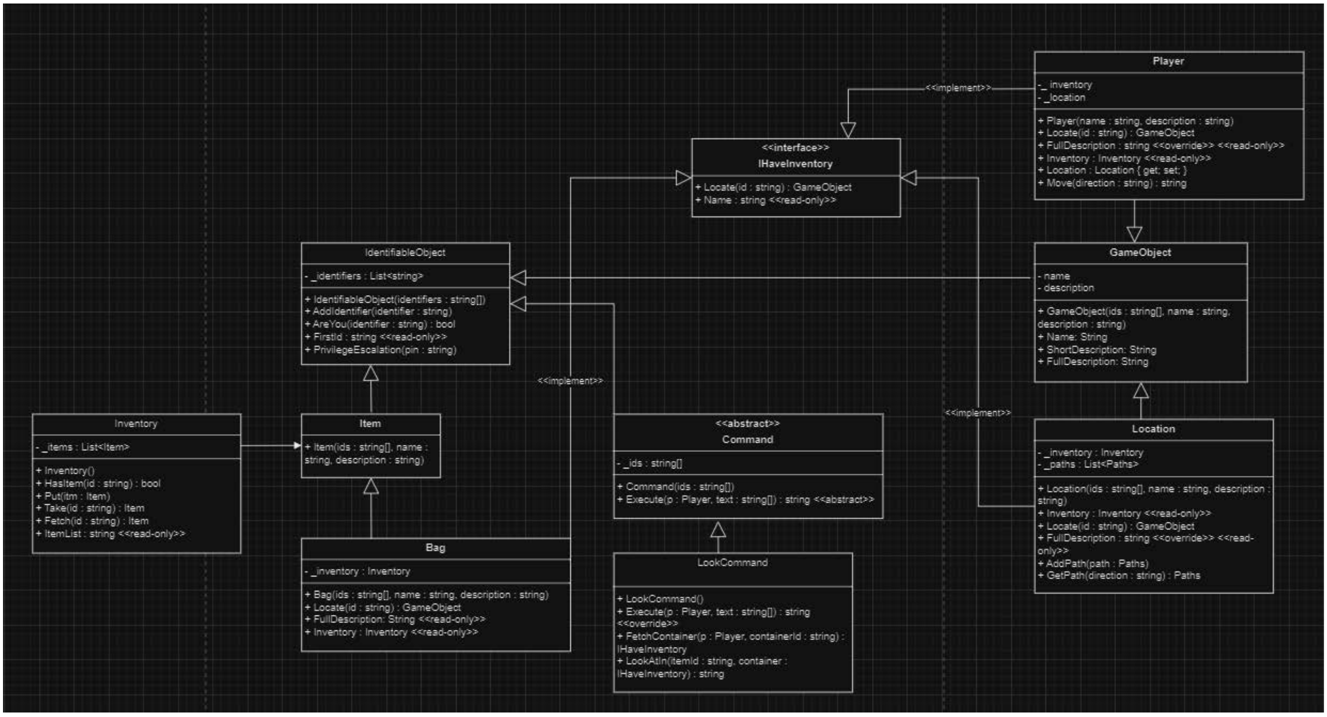
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _7._2C
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _inventory;

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

        public Inventory Inventory
        { get { return _inventory; } }

        public GameObject Locate(string id) //the purpose is to return the gameobject itself
        {
            if (AreYou(id))
                {  return this; }
            return _inventory.Fetch(id);

        }

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

```

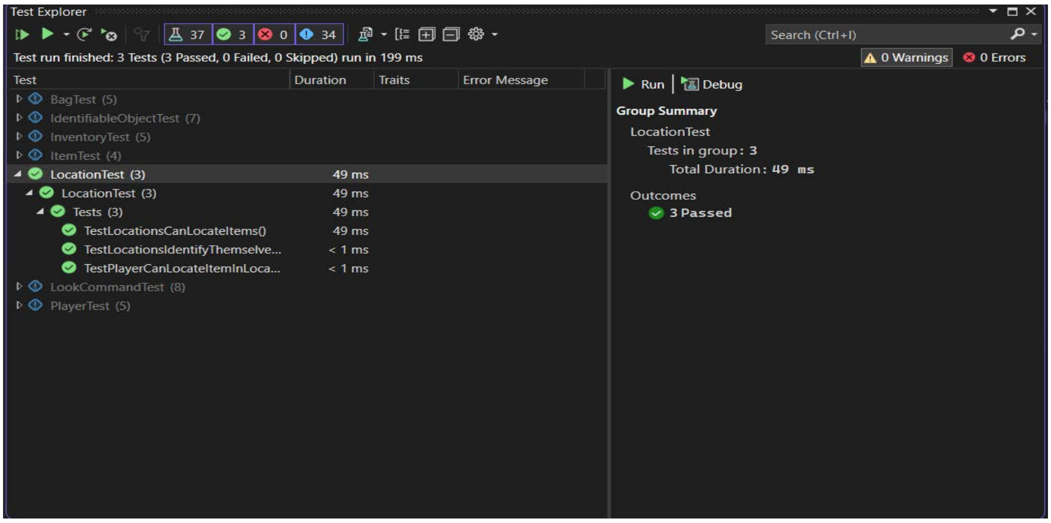
```csharp
using _9._2C;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using NUnit.Framework;
using System.Numerics;

namespace LocationTest
{
    [TestFixture]
    public class Tests
    {
        private Location _location;
        private Item _sword;
        private Player _player;

        [SetUp]
        public void Setup()
        {
            _sword = new Item(new string[] { "sword" }, "Excalibur", "a strong sword");
            _player = new Player("Nevan", "a human");
            _location = new Location(new string[] { "classroom" }, "EN310", "Swinburne's classroom");
            _location.Inventory.Put(_sword);

        }

        [TestCase]
        public void TestLocationsIdentifyThemselves()
        {
            Assert.IsTrue(_location.AreYou("classroom"));
        }

        [TestCase]
        public void TestLocationsCanLocateItems()
        {
            Assert.AreEqual(_sword, _location.Locate("sword"));
        }

        [TestCase]
        public void TestPlayerCanLocateItemInLocation()
        {
            _player.Location = _location; //player in a location that has a sword, so don't necessary need to have sword in inventory
            //_player.Inventory.Put(_sword);
            Assert.AreEqual(_sword, _player.Locate("sword"));
        }
    }
}
```

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

namespace _7._2C
{
    public class LookCommand : Command
    {
        public LookCommand () : base(new string[] {"look"})
        {
        }

        public override string Execute(Player p, string[] text)
        {
            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?";
            }

            if (text.Length == 5 && text[3] != "in")
            {
                return "What do you want to look in?";
            }

            IHaveInventory container;
            if (text.Length == 3)
            {

                container = FetchContainer(p, "inventory");
            }

            else
            {

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

            // Step 7: The item id is the 3rd word
            string itemId = text[2];
            return LookAtIn(itemId, container);

        }
        public IHaveInventory FetchContainer(Player p, string containerId)
        {
            if (containerId.ToLower() == "inventory")
            {
                return p;
            }

            GameObject obj = p.Locate(containerId);
            if (obj is IHaveInventory)
            {
                return (IHaveInventory)obj; //container is bag(?)
            }
            return null;
        }

        public string LookAtIn(string itemId, IHaveInventory container)
        {
            // Try to locate the item within the specified container
            GameObject item = container.Locate(itemId);
            if (item == null)
            {
                return $"I cannot find the {itemId} in {container.Name}";
            }
            ////

            // Return the item's full description if found
            return item.FullDescription;
        }
    }
}
```


Player
- _inventory
- _location
+ Player(name : string, description : string)
+ Locate(id : string) : GameObject
+ FullDescription : string <<override>> <<read-only>>
+ Inventory : Inventory <<read-only>>
+ Location : Location { get; set; }
+ Move(direction : string) : string
<<implement>>
<<interface>>
IHaveInventory
+ Locate(id : string) : GameObject
+ Name : string <<read-only>>
IdentifiableObject
- _identifiers : List<string>
+ IdentifiableObject(identifiers : string[])
+ AddIdentifier(identifier : string)
+ AreYou(identifier : string) : bool
+ FirstId : string <<read-only>>
+ PrivilegeEscalation(pin : string)
GameObject
- name
- description
+ GameObject(ids : string[], name : string, description : string)
+ Name: String
+ ShortDescription: String
+ FullDescription: String
<<implement>>
<<implement>>
Inventory
- _items : List<Item>
+ Inventory()
+ HasItem(id : string) : bool
+ Put(itm : Item)
+ Take(id : string) : Item
+ Fetch(id : string) : Item
+ ItemList : string <<read-only>>
Item
+ Item(ids : string[], name : string, description : string)
<<abstract>>
Command
- _ids : string[]
+ Command(ids : string[])
+ Execute(p : Player, text : string[]) : string <<abstract>>
Location
- _inventory : Inventory
- _paths : List<Paths>
+ Location(ids : string[], name : string, description : string)
+ Inventory : Inventory <<read-only>>
+ Locate(id : string) : GameObject
+ FullDescription : string <<override>> <<read-only>>
+ AddPath(path : Paths)
+ GetPath(direction : string) : Paths
Bag
- _inventory : Inventory
+ Bag(ids : string[], name : string, description : string)
+ Locate(id : string) : GameObject
+ FullDescription: String <<read-only>>
+ Inventory : Inventory <<read-only>>
LookCommand
+ LookCommand()
+ Execute(p : Player, text : string[]) : string <<override>>
+ FetchContainer(p : Player, containerId : string) : IHaveInventory
+ LookAtIn(itemId : string, container : IHaveInventory) : string

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

namespace _7._2C
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory = new Inventory();
        private Location _location;

        public Player(string name, string description) : base(new string [] { "me", "inventory" }, name, description) { } //name and des gotten from GameObject
        //help the class identify itself and its item, 3 batteries, 2 from GO and 1 from IO
        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this; //return then player object itself
            }
            GameObject item = _inventory.Fetch(id); // Fetch the item from the inventory if it exists.
            if (item != null)
            {
                return item; // Return the item if found in the inventory.
            }
            //Check for location if not found in inventory
            if (_location != null)
            {
                return _location.Locate(id); //instead of returning null like the first time, this time it will look for the location
            }
            return null;
        }

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

        public Inventory Inventory { get { return _inventory; } }

        public Location Location
```

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


  }
}

```



Test Explorer

Search (Ctrl+I)

37 3 0 34

Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 199 ms

0 Warnings 0 Errors

Test	Duration	Traits	Error Message
BagTest (5)			
IdentifiableObjectTest (7)			
InventoryTest (5)			
ItemTest (4)			
LocationTest (3)	49 ms		
LocationTest (3)	49 ms		
Tests (3)	49 ms		
TestLocationsCanLocateItems()	49 ms		
TestLocationsIdentifyThemselve...	< 1 ms		
TestPlayerCanLocateItemInLoca...	< 1 ms		
LookCommandTest (8)			
PlayerTest (5)			

Run | Debug

Group Summary

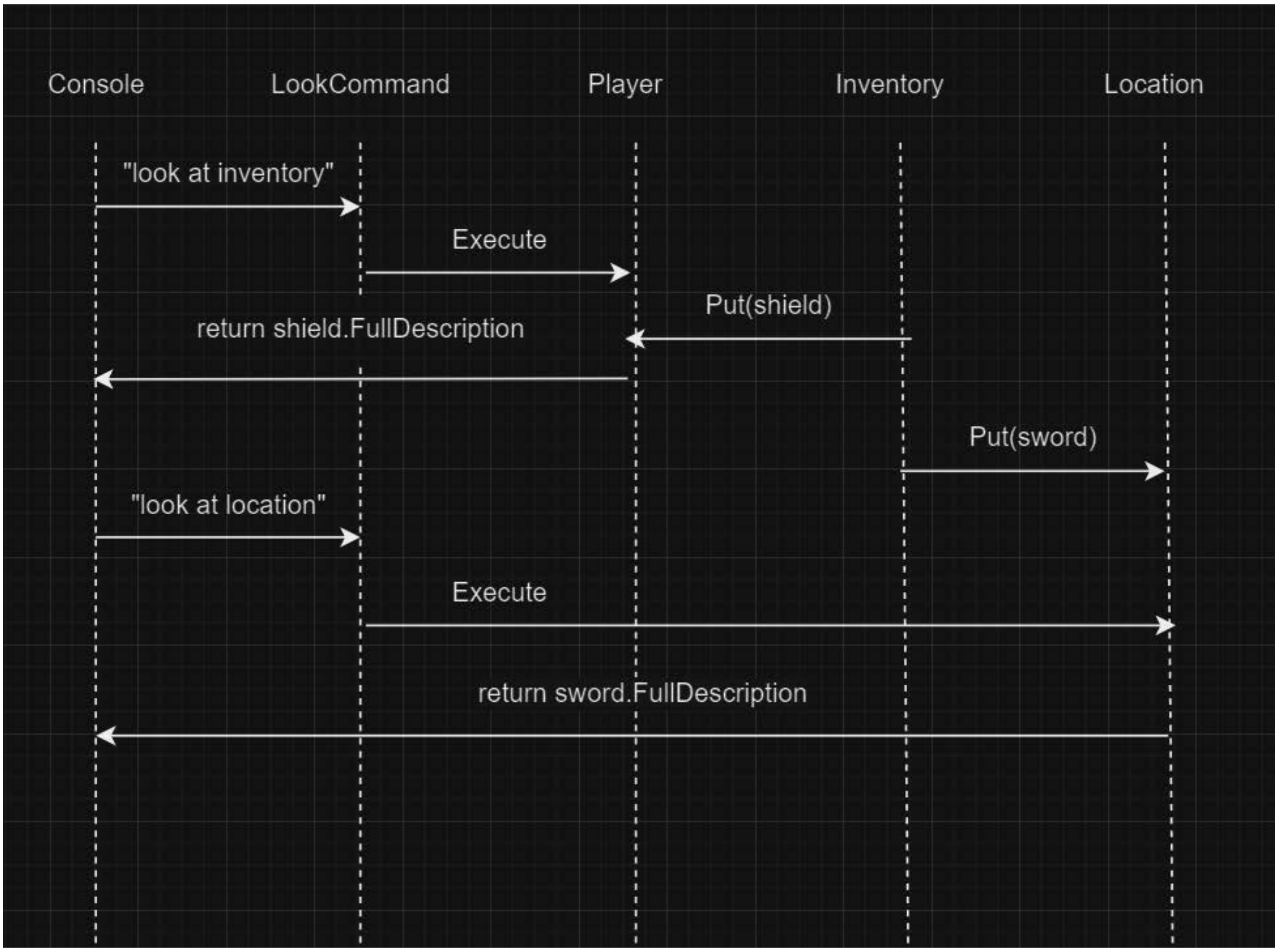
LocationTest

Tests in group: 3

Total Duration: 49 ms

Outcomes

3 Passed


Console
LookCommand
Player
Inventory
Location
"look at inventory"
Execute
Put(shield)
return shield.FullDescription
Put(sword)
"look at location"
Execute
return sword.FullDescription