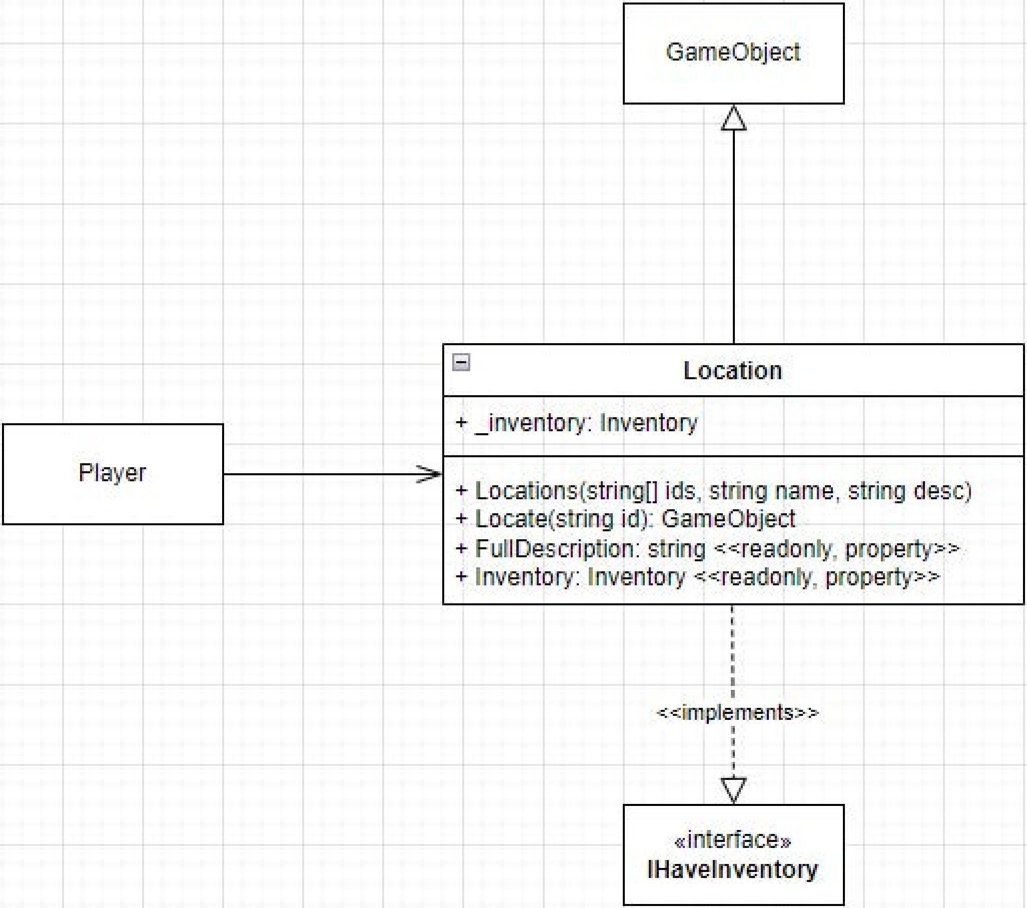
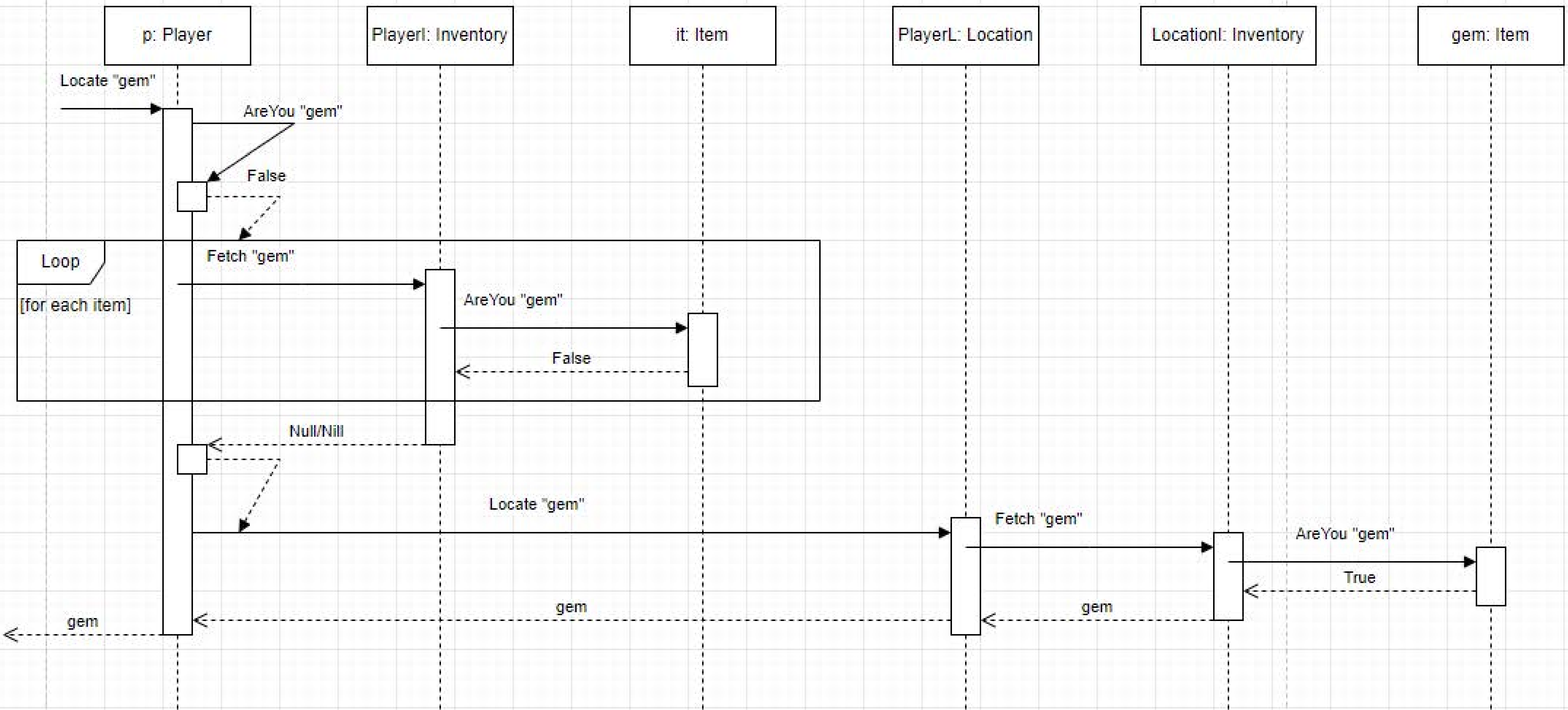
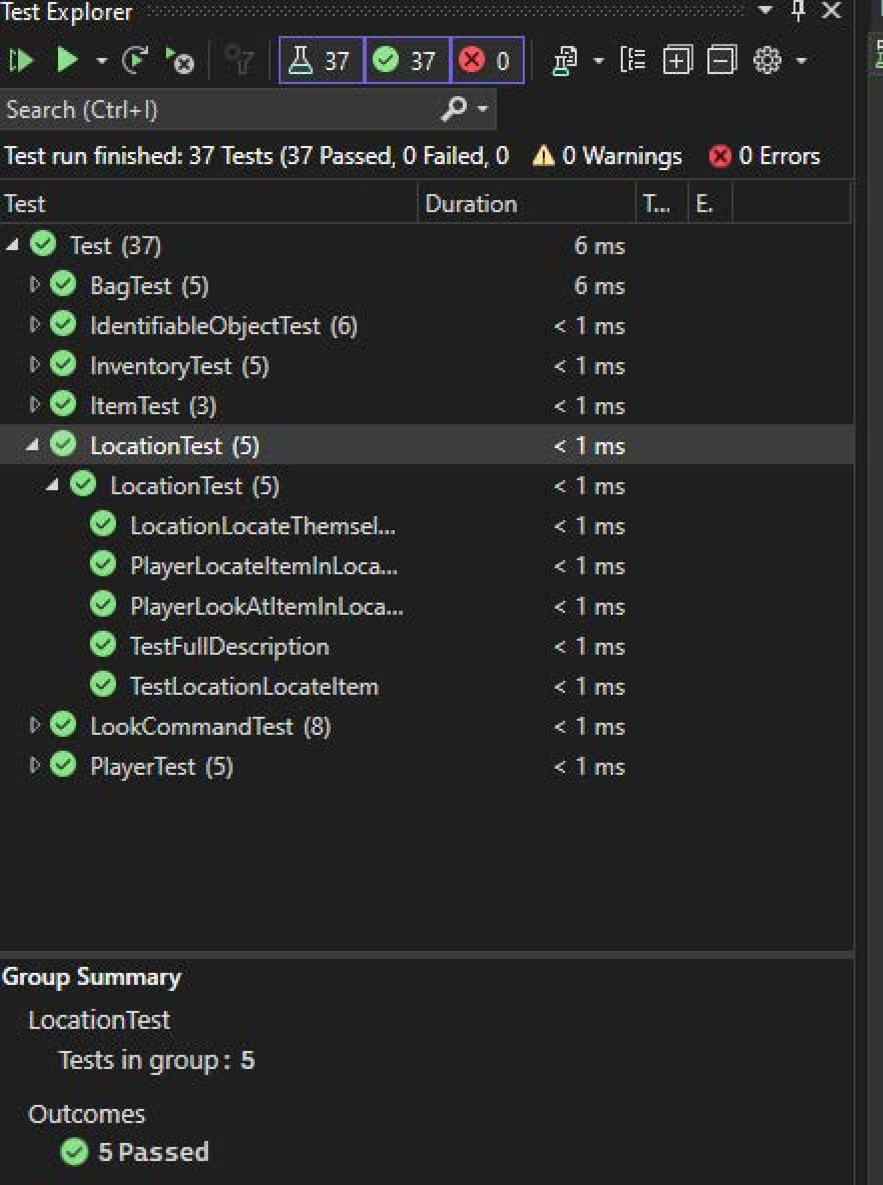

GameObject
Location
+ _inventory: Inventory
+ Locations(string[] ids, string name, string desc)
+ Locate(string id): GameObject
+ FullDescription: string <<readonly, property>>
+ Inventory: Inventory <<readonly, property>>
Player
<<implements>>
«interface»
IHaveInventory


p: Player
PlayerI: Inventory
it: Item
PlayerL: Location
LocationI: Inventory
gem: Item
Locate "gem"
AreYou "gem"
False
Loop
[for each item]
Fetch "gem"
AreYou "gem"
False
Null/Nill
Locate "gem"
Fetch "gem"
AreYou "gem"
True
gem
gem
gem

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

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

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

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

        public override string FullDescription
        {
            get
            {
                return "You are in the " + Name + ".\n" + "You can see:\n" + _inventory.ItemList;
            }
        }

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

```

```csharp
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;
        private Locations _location;

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

        public GameObject? Locate(string id)
        {
            GameObject? res = null;
            if (AreYou(id))
            {
                return this;
            }
            res = _inventory.Fetch(id);
            if(res != null)
            {
                return res;
            }
            if(_location != null)
            {
                res = _location.Locate(id);
                return res;
            }
            return null;
        }

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

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

```



```
Test run finished: 37 Tests (37 Passed, 0 Failed, 0   ⚠ 0 Warnings   ✖ 0 Errors

Test                                      Duration      T...   E.
◢ ✓ Test (37)                                 6 ms
  ▷ ✓ BagTest (5)                             6 ms
  ▷ ✓ IdentifiableObjectTest (6)            < 1 ms
  ▷ ✓ InventoryTest (5)                     < 1 ms
  ▷ ✓ ItemTest (3)                          < 1 ms
  ◢ ✓ LocationTest (5)                      < 1 ms
    ◢ ✓ LocationTest (5)                    < 1 ms
        ✓ LocationLocateThemsel...          < 1 ms
        ✓ PlayerLocateItemInLoca...         < 1 ms
        ✓ PlayerLookAtItemInLoca...         < 1 ms
        ✓ TestFullDescription               < 1 ms
        ✓ TestLocationLocateItem            < 1 ms
  ▷ ✓ LookCommandTest (8)                   < 1 ms
  ▷ ✓ PlayerTest (5)                        < 1 ms

Group Summary
  LocationTest
    Tests in group : 5

  Outcomes
    ✓ 5 Passed
```

```
using SwinAdventure;

namespace LocationTest
{
    public class LocationTest
    {
        Command look;
        Player player;
        Bag bag;
        Item BronzeSword, BronzeAxe, gem;
        Locations home;

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

            player = new Player("Sen", "luv ur mom");

            bag = new Bag(new string[] { "bag", "inventory" }, "Bag", "A simple bag.");

            BronzeSword = new Item(new string[] { "sword", "weapon" }, "Bronze Sword", "A simple bronze sword.");
            BronzeAxe = new Item(new string[] { "axe", "weapon" }, "Bronze Axe", "A simple bronze axe.");
            gem = new Item(new string[] { "gem", "jewel" }, "Gem", "A shiny gem.");

            home = new Locations(new string[] { "home", "house" }, "Home", "Sen's home.");
            player.Location = home;
        }

        [Test]
        public void LocationLocateThemselvesTest()
        {
            player.Locate("home");
            Assert.AreEqual(home, player.Location);
        }

        [Test]
        public void TestLocationLocateItem()
        {
            home.Inventory.Put(BronzeSword);
            home.Locate("sword");
            Assert.AreEqual(BronzeSword, home.Locate("sword"));
        }

        [Test]
        public void TestFullDescription()
        {
            home.Inventory.Put(BronzeSword);
            home.Inventory.Put(BronzeAxe);
            home.Inventory.Put(gem);
            string expected = "You are in the Home.\nYou can see:\nBronze Sword (sword)\nBronze Axe (axe)\nGem (gem)\n";
            Assert.AreEqual(expected, home.FullDescription);
        }

        [Test]
        public void PlayerLocateItemInLocation()
        {
            home.Inventory.Put(gem);
            player.Locate("gem");
            Assert.AreEqual(gem, player.Locate("gem"));
        }

        [Test]
        public void PlayerLookAtItemInLocation()
        {
            home.Inventory.Put(gem);
            string[] text = new string[] { "look", "at", "gem", "in", "home" };
            Assert.AreEqual("Gem (gem): A shiny gem.", look.Execute(player, text));
        }
    }
}

```