Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 6 - Locations

PDF generated at 20:33 on Thursday 26th October, 2023

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

namespace Iteration2
{
    public class Location : GameObject, IHaveInventory
    {

        private Inventory _inventory;

        public Location(string name, string description) : base(new string[] {
↪   "room", "here" }, name, description)
        {
            _inventory = new Inventory();
        }

        public GameObject Locate(string id)
        {

            if (AreYou(id))
            {
                return this;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.FetchItem(id);
            }
          // might be here
            else
            {
               return null;
            }
        }
        public override string FullDescription
        {
            get
            {
                return $"You are in {Name}\n{Description}\nIn this room you can
↪   see:\n{_inventory.ItemList}";
            }
        }
        public Inventory Inventory

        {
            get
            {
                return _inventory;
            }
        }

        public Item FetchItem (string id)
        {
            return _inventory.FetchItem(id);
        }

    }
}
```

```
using System;
using NUnit.Framework;
using Iteration2;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace TestProject7
{
    [TestFixture]
    public class TestLocation
    {
        Location location;
        Player player;
        Item sword;

        Item pistol;

        [SetUp]
        public void Setup ()
        {
            location = new Location("a conflict", "In World");
            player = new Player("Aaryan", "the student");
            sword = new Item(new string[] { "Sword" }, "a Sword", "a sharp Sword");
            location.Inventory.Put(sword);
            player.Location = location;
        }


        //1
        [Test]
        public void TestIdentifyLocation ()
        {
            Assert.That(location.Locate("room"), Is.SameAs(location));

        }

        //2
        [Test]
        public void TestIdentifyItemsInLocation ()
        {
            Assert.That(location.Locate("pistol"), Is.SameAs(pistol));
        }

        //3
        [Test]
        public void TestIdentifyPlayerInLocation ()
        {
            Assert.That(location.Locate("pistol"), Is.SameAs(pistol));
        }

        //4
        [Test]
        public void TestLocationFullDescription ()
        {
            string actual = location.FullDescription;
            string expected = "You are in a conflict\nIn World\nIn this room you can
    ↪   see:\n\n\ta Sword (Sword)";
            Assert.That(actual, Is.EqualTo(expected));
        }






    }
}
```

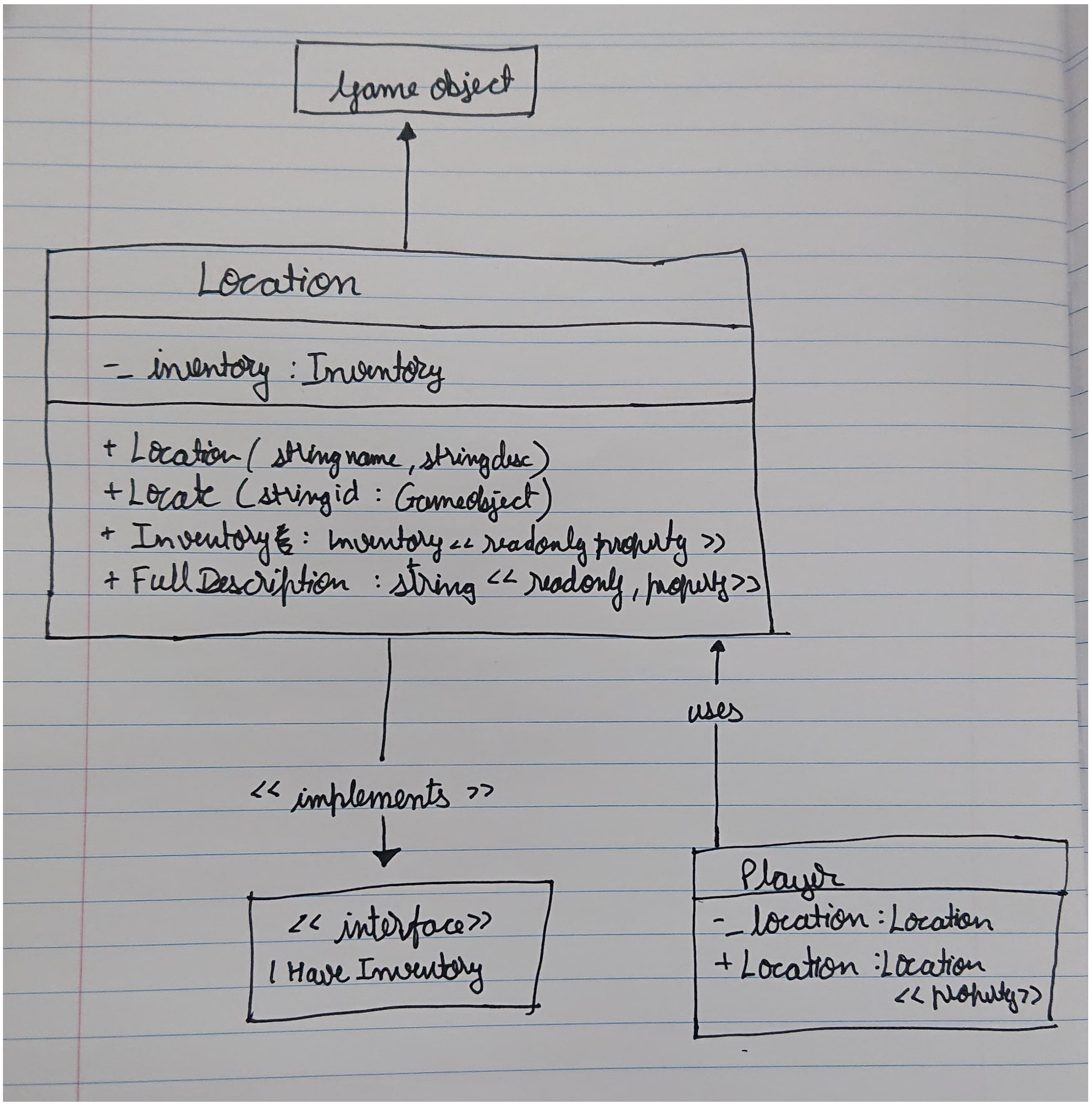
```
using Iteration2;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace Iteration2
{


    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)
        {
            if (AreYou(id))
            {
                return this;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.FetchItem(id);
            }
            else if (_location != null)
            {
                return _location.Locate(id);
            }
            else
            {
                return null;
            }


           // return _inventory.Fetch(id);
        }


        public override string FullDescription
        {
            get
            {


                return $"You are {Name}, {base.FullDescription} \nyou are
  ↪   carrying:\n\t" + _inventory.ItemList;
            }
        }


        public Inventory Inventory
        {
            get

            { return _inventory; }
        }
        public Location Location
        {
            get
            {
                return _location;

            }


            set
            {
                _location = value;
            }

        }

    }
}
```

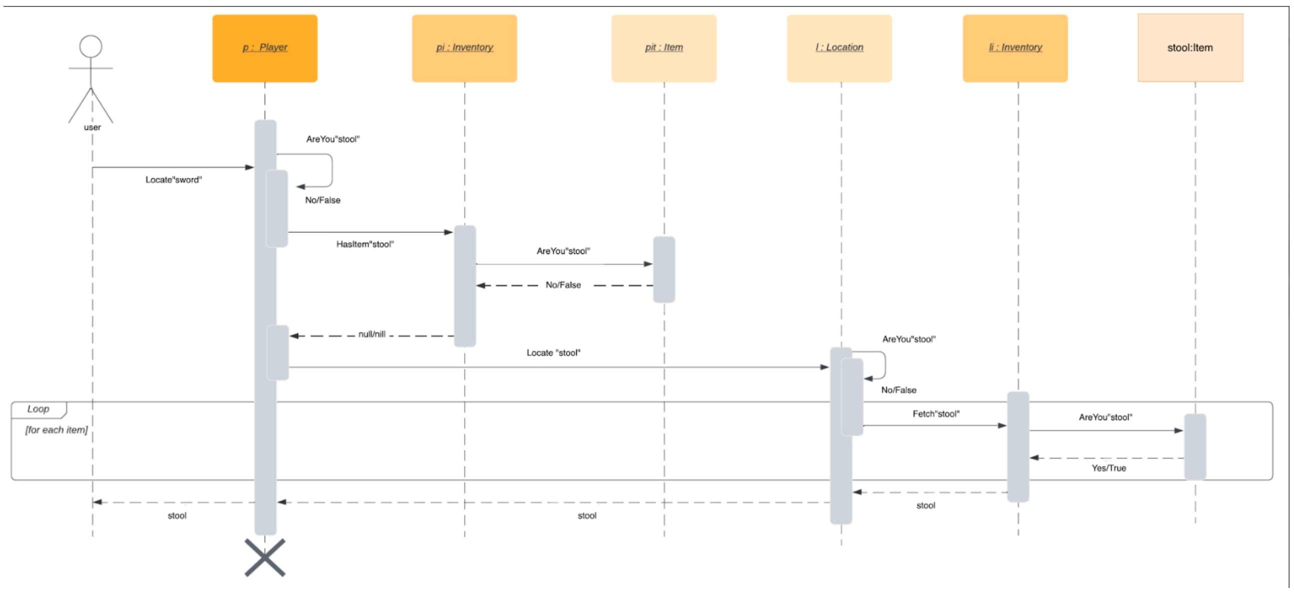
```
using Iteration2;

namespace TestProject3
{
    [TestFixture]
    public class TestPlayer
    {
        Player player;
        Item stool;
        Location location;
        Item gem;

        [SetUp]
        public void Setup()
        {
            player = new Player("Aaryan", "the student");
            stool = new Item(new string[] { "stool" }, "50cm stool", "This is a
↪   stool");
            player.Inventory.Put(stool);
            gem = new Item(new string[] { "gem" }, "a gem", "a bright red crystal");
            location = new Location("a conflict", "In world");
            location.Inventory.Put(gem);
            player.Location = location;

        }

        [Test]
        public void TestLocateItems()
        {
            Assert.That(player.Locate("stool"), Is.SameAs(stool));
            Assert.That(player.Inventory.HasItem("stool"), Is.True);


        }

        [Test]
        public void TestIsIdentifiable()
        {
            Assert.That(player.AreYou("me"), Is.True);
            Assert.That(player.AreYou("inventory"), Is.True);
        }

        [Test]
        public void TestLocateItself()
        {
            Assert.That(player.Locate("me"), Is.SameAs(player));
            Assert.That(player.Locate("inventory"), Is.SameAs(player));
        }
        [Test]
        public void TestLocateNothing()
        {
            Assert.That(player.Locate("ssswor"), Is.SameAs(null));

        }

        [Test]
        public void TestFullDescription()
        {
            Assert.That(player.FullDescription, Is.EqualTo("You are Aaryan, the
↪   student \nyou are carrying:\n\t\n\t50cm stool (stool)"));
        }
    }
}
```

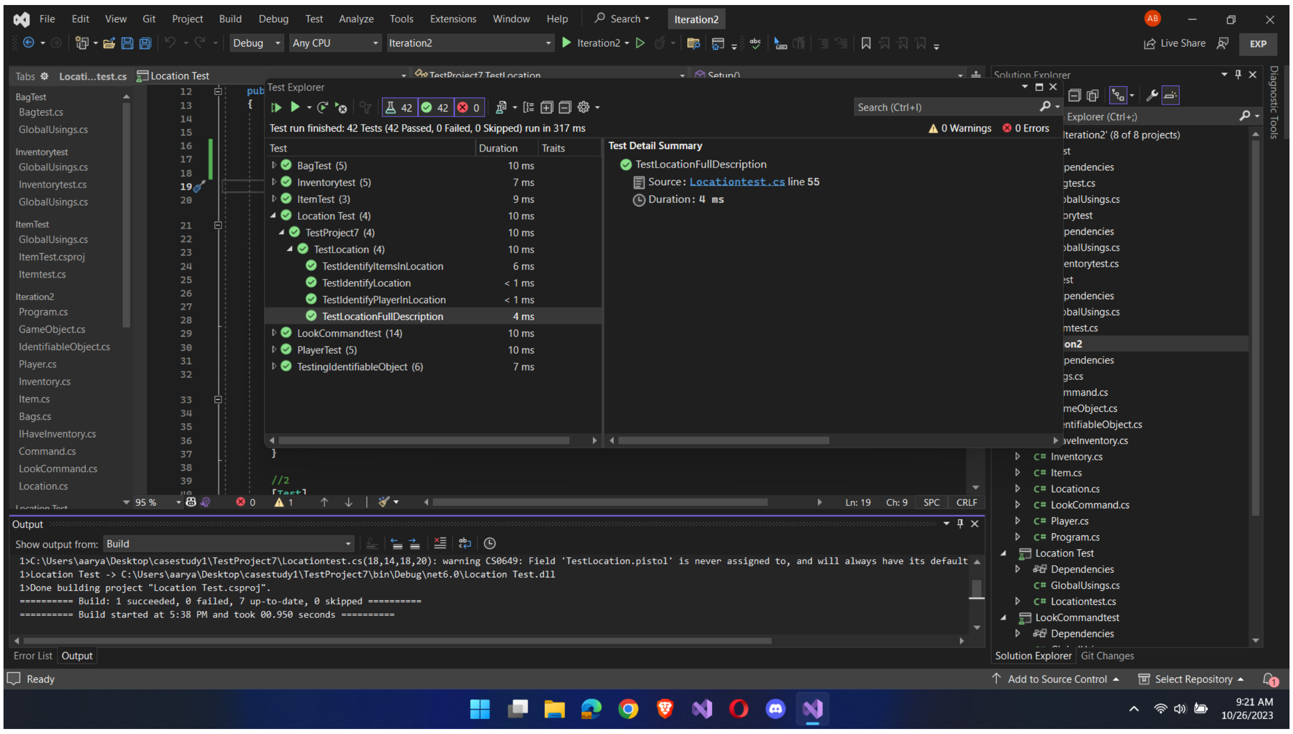
```
using System;


namespace Iteration2
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {

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

            IHaveInventory container = null;
            string itemId = null;
            if (text.Length != 1  && text.Length != 3 && text.Length != 5)

            {
                return "I don't know how to look like that";
            }

            else
            {
                if (text[0] != "look")
                {
                    return "Error in look input";
                }
                else if ( text.Length !=1 &&   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?";
                }

                switch (text.Length)

                {
                    case 1:
                        container = p;
                        itemId = "room";
                        break;
                    case 3:
                        container = p;
                        itemId = text[2];
                        break;
                    case 5:
                        container = FetchContainer(p, text[4]);
                        if (container == null)
                        {
                            return "I can't find the " + text[4];
                        }
                        itemId = text[2];
                        break;

                }
                return LookAtIn(itemId, container);
            }
        }
        private IHaveInventory FetchContainer(Player p, string containerId)
        {
            return p.Locate(containerId) as IHaveInventory;
        }
        private string LookAtIn(string thingId, IHaveInventory container)
        {
            GameObject thing = container.Locate(thingId);
            if (thing == null)
            {
                return $"I can't find the {thingId}";
            }
            else
            {
                return thing.FullDescription;
            }
        }
    }
}
```

```
using Iteration2;
using System;


namespace TestProject6
{
    [TestFixture]
    public class LookCommandTest
    {
        LookCommand look;
        Bag bag;
        Player player;
        Location location;
        Item gem;



        [SetUp]
        public void Setup()
        {
            look = new();
            player = new Player("Aaryan", "student");
            bag = new Bag(new string[] { "bag" }, "bag", "This is a expensive bag");
            gem = new Item(new string[] { "gem" }, "a gem", "a bright red crystal");
            location = new Location("a conflict", "In world");

        }

        //1
        [Test]
        public void TestLookAtMe()
        {
            string expected = player.FullDescription;
            Assert.That(look.Execute(player, new string[] { "look", "at", "me" }),
↪   Is.EqualTo(expected));
        }

        //2
        [Test]
        public void TestLookAtGem()
        {
            player.Inventory.Put(gem);
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem" }),
↪   Is.EqualTo("a bright red crystal"));
        }

        //3
        [Test]
        public void TestLookAtUnk()
        {
            player.Inventory.Take("gem");
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem" }),
↪   Is.EqualTo("I can't find the gem"));
```

```
        }


        //4
        [Test]
        public void TestLookAtGemInMe()
        {
            player.Inventory.Put(gem);
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
↪   "in", "me" }), Is.EqualTo("a bright red crystal"));
        }

        //5
        [Test]
        public void TestLookAtGemInBag()
        {
            bag.Inventory.Put(gem);
            player.Inventory.Put(bag);
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
↪   "in", "bag" }), Is.EqualTo("a bright red crystal"));
        }

        //6
        [Test]
        public void LookAtGemInNoBag()
        {
            player.Inventory.Take("bag");
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
↪   "in", "bag" }), Is.EqualTo("I can't find the bag"));
        }

        //7
        [Test]
        public void TestLookAtNoGemInBag()
        {
            bag.Inventory.Take("gem");
            player.Inventory.Put(bag);
            Assert.That(look.Execute(player, new string[] { "look", "at", "gem",
↪   "in", "bag" }), Is.EqualTo("I can't find the gem"));
        }

        //8
        [Test]
        public void TestInvalidLook()
        {
            Assert.That(look.Execute(player, new string[] { "find", "the", "gem" }),
↪   Is.EqualTo("Error in look input"));
        }

        //1.1
        [Test]
        public void TestPlayerLocation()
        {
            player.Location = location;
            string actual = look.Execute(player, new string[] { "look" });
            Assert.That(actual, Is.EqualTo(location.FullDescription));
        }

        [TestCaseSource(nameof(InvalidLenghtTestCases))]
        public void TestInvalidLenght(string[] toTest)
        {
            Assert.That(look.Execute(player, toTest), Is.EqualTo("I don't know how to
    ↪   look like that"));
        }

        private static IEnumerable<string[]> InvalidLenghtTestCases
        {
            get
            {
                yield return new string[] { "look", "at" };
                yield return new string[] { "look", "at", "gem", "in", "the", "bag"
    ↪   };
                yield return new string[] { "look", "at", "big", "back" };
            }
        }

        [Test]
        public void TestInvalidAt()
        {
            string actual = look.Execute(player, new string[] { "look", "in", "gem"
    ↪   });
            string expected = "What do you want to look at?";
            Assert.That(actual, Is.EqualTo(expected));
        }

        [Test]
        public void TestInvalidIn()
        {
            player.Inventory.Put(gem);
            player.Inventory.Put(bag);
            string actual = look.Execute(player, new string[] { "look", "at","in",
    ↪   "bag" , "bag" });
            string expected = "What do you want to look in?";
            Assert.That(actual, Is.EqualTo(expected));
        }

    }

}
```


Game object
Location
-_inventory : Inventory
+ Location (string name, string desc)
+ Locate (string id : Gameobject)
+ Inventory {} : inventory << readonly property >>
+ FullDescription : string << readonly, property >>
uses
<< implements >>
<< interface>>
I Have Inventory
Player
-_location : Location
+ Location : Location
<< property>>

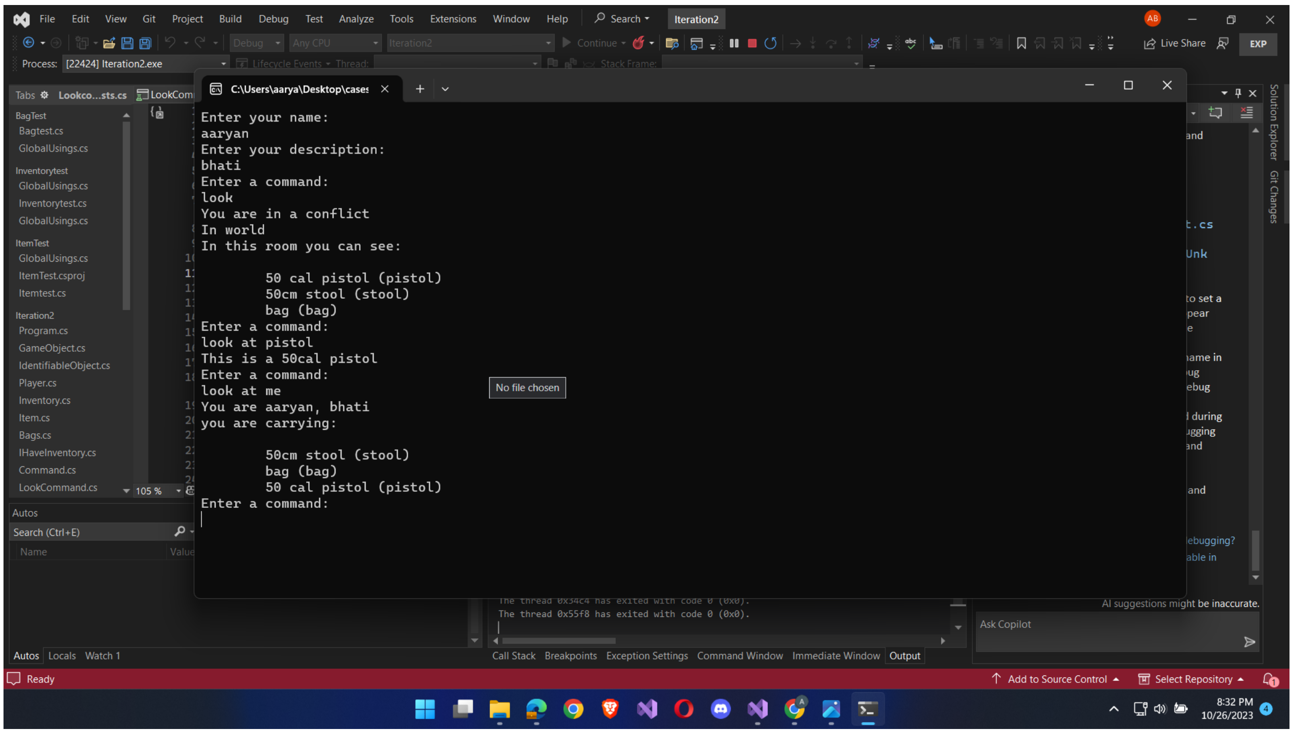

p : Player
pi : Inventory
pit : Item
l : Location
li : Inventory
stool:Item
user
AreYou"stool"
Locate"sword"
No/False
HasItem"stool"
AreYou"stool"
No/False
null/nill
AreYou"stool"
Locate "stool"
No/False
Loop
[for each item]
Fetch"stool"
AreYou"stool"
Yes/True
stool
stool
stool



```
Test run finished: 42 Tests (42 Passed, 0 Failed, 0 Skipped) run in 317 ms        0 Warnings   0 Errors

Test                                   Duration   Traits
▷ ✓ BagTest (5)                          10 ms
▷ ✓ Inventorytest (5)                     7 ms
▷ ✓ ItemTest (3)                          9 ms
◢ ✓ Location Test (4)                    10 ms
  ◢ ✓ TestProject7 (4)                   10 ms
    ◢ ✓ TestLocation (4)                 10 ms
        ✓ TestIdentifyItemsInLocation     6 ms
        ✓ TestIdentifyLocation          < 1 ms
        ✓ TestIdentifyPlayerInLocation  < 1 ms
        ✓ TestLocationFullDescription     4 ms
▷ ✓ LookCommandtest (14)                 10 ms
▷ ✓ PlayerTest (5)                       10 ms
▷ ✓ TestingIdentifiableObject (6)         7 ms

Test Detail Summary
✓ TestLocationFullDescription
  Source: Locationtest.cs line 55
  Duration: 4 ms
```

Output — Show output from: Build

```
1>C:\Users\aarya\Desktop\casestudy1\TestProject7\Locationtest.cs(18,14,18,20): warning CS0649: Field 'TestLocation.pistol' is never assigned to, and will always have its default
1>Location Test -> C:\Users\aarya\Desktop\casestudy1\TestProject7\bin\Debug\net6.0\Location Test.dll
1>Done building project "Location Test.csproj".
========== Build: 1 succeeded, 0 failed, 7 up-to-date, 0 skipped ==========
========== Build started at 5:38 PM and took 00.950 seconds ==========
```



```
Enter your name:
aaryan
Enter your description:
bhati
Enter a command:
look
You are in a conflict
In world
In this room you can see:

        50 cal pistol (pistol)
        50cm stool (stool)
        bag (bag)
Enter a command:
look at pistol
This is a 50cal pistol
Enter a command:
look at me
You are aaryan, bhati
you are carrying:

        50cm stool (stool)
        bag (bag)
        50 cal pistol (pistol)
Enter a command:
```