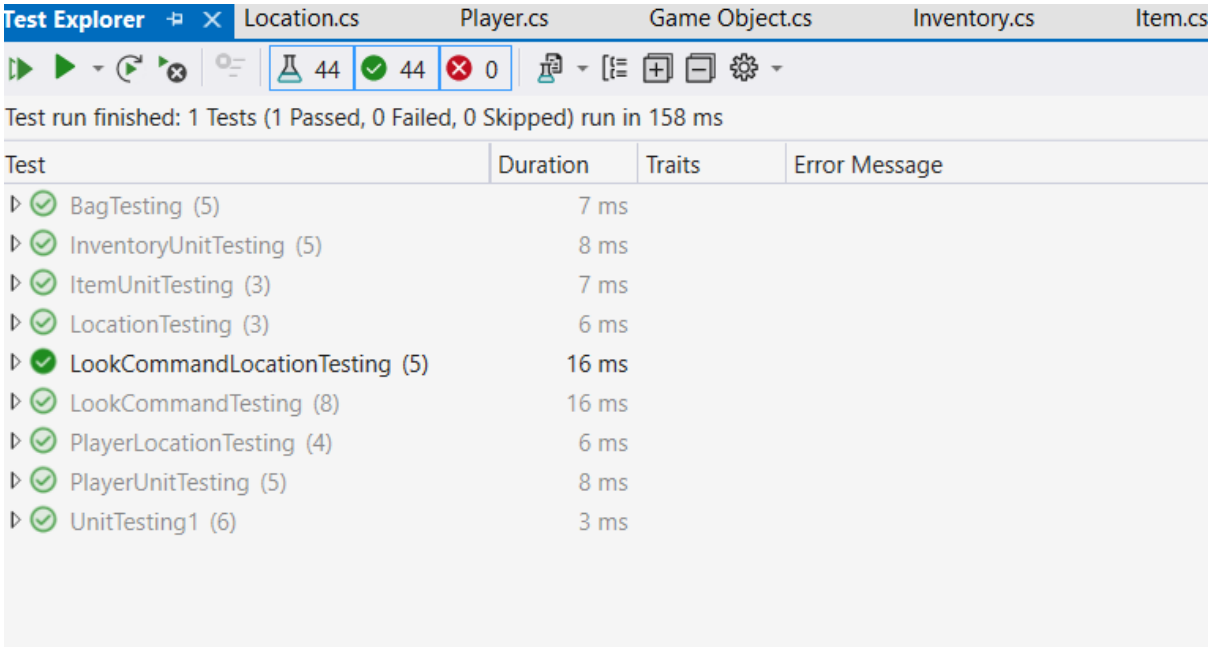
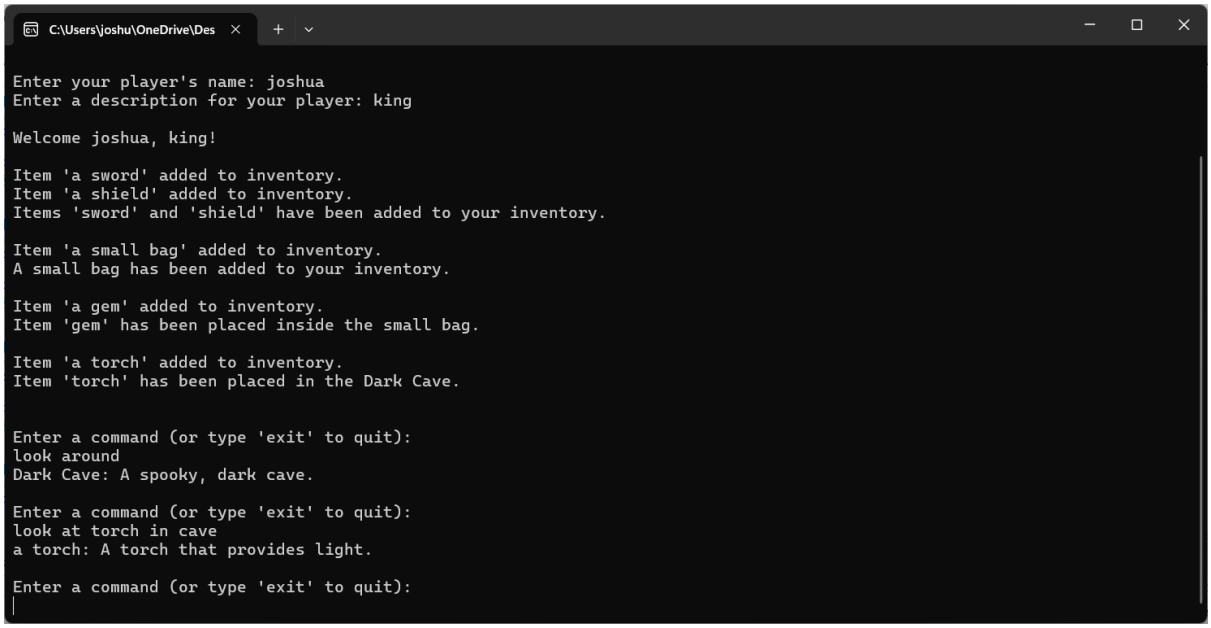
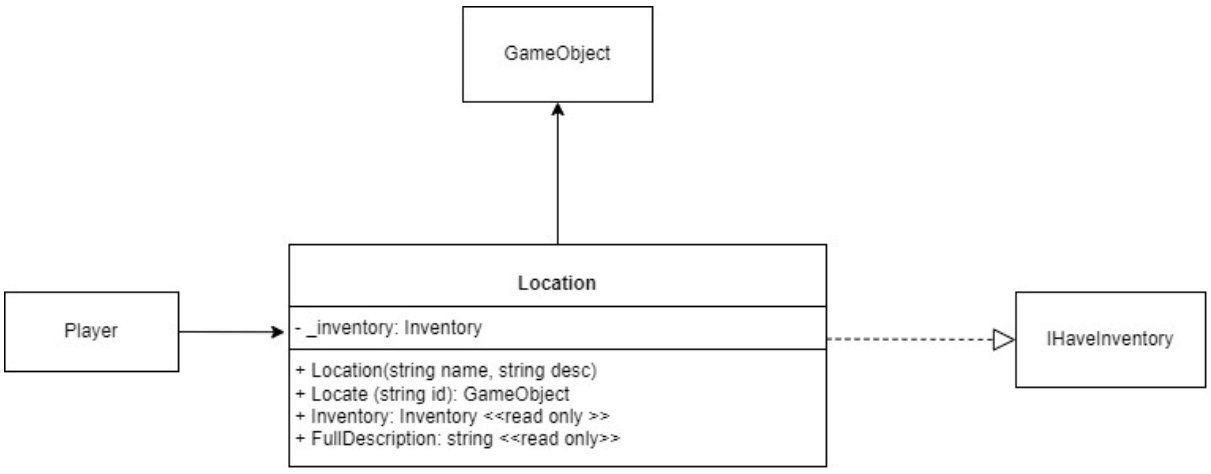
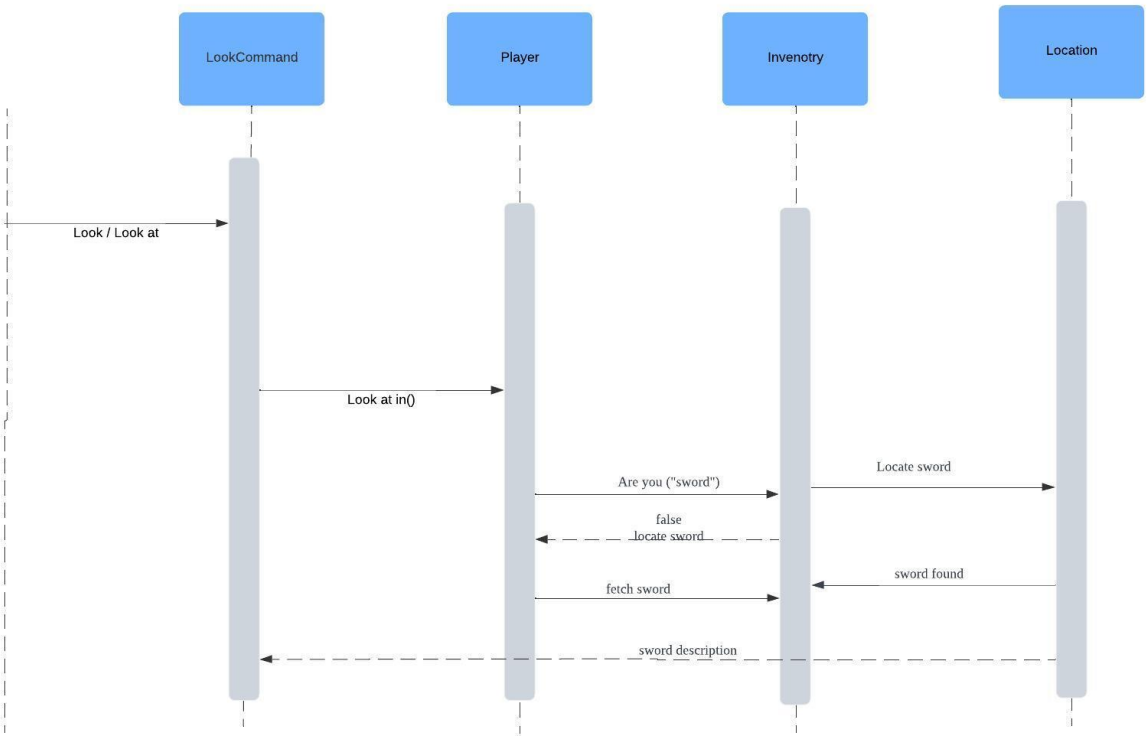
7.2C


Test Explorer
Location.cs
Player.cs
Game Object.cs
Inventory.cs
Item.cs
44
44
0
Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 158 ms
Test
Duration
Traits
Error Message
BagTesting (5) 7 ms
InventoryUnitTesting (5) 8 ms
ItemUnitTesting (3) 7 ms
LocationTesting (3) 6 ms
LookCommandLocationTesting (5) 16 ms
LookCommandTesting (8) 16 ms
PlayerLocationTesting (4) 6 ms
PlayerUnitTesting (5) 8 ms
UnitTesting1 (6) 3 ms


C:\Users\joshu\OneDrive\Des
Enter your player's name: joshua
Enter a description for your player: king
Welcome joshua, king!
Item 'a sword' added to inventory.
Item 'a shield' added to inventory.
Items 'sword' and 'shield' have been added to your inventory.
Item 'a small bag' added to inventory.
A small bag has been added to your inventory.
Item 'a gem' added to inventory.
Item 'gem' has been placed inside the small bag.
Item 'a torch' added to inventory.
Item 'torch' has been placed in the Dark Cave.
Enter a command (or type 'exit' to quit):
look around
Dark Cave: A spooky, dark cave.
Enter a command (or type 'exit' to quit):
look at torch in cave
a torch: A torch that provides light.
Enter a command (or type 'exit' to quit):


GameObject
Location
- _inventory: Inventory
+ Location(string name, string desc)
+ Locate (string id): GameObject
+ Inventory: Inventory <<read only >>
+ FullDescription: string <<read only>>
Player
IHaveInventory



Location.cs:

```
namespace SwinAdventure
{
    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)
        {
            // Check if the location itself is being looked for
            if (AreYou(id))
            {
                return this;
            }

            // Check if the item is in the location's inventory
            return _inventory.Fetch(id);
        }

        public override string GetFullDescription()
        {
            return $"{Name}: {ShortDescription}";
        }
    }
}
```

LookCommand.cs:

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

        public override string Execute(Player player, string[] text)
        {
            // Allow for "look", "look at <item>", "look at <item> in <container>", and "look around"
```

```
if (text.Length != 3 && text.Length != 5 && !(text.Length == 2 && text[1] == "around"))
{
    return "I don't know how to look like that";
}

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

if (text[1] == "around" && text.Length == 2)
{
    // Handle "look around"
    return player.Location?.GetFullDescription() ?? "You are nowhere.";
}

if (text[1] == "at")
{
    string itemId = text[2];
    IHaveInventory container;

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

        string containerId = text[4];
        container = FetchContainer(player, containerId);
```

```
            if (container == null)
            {
                return $"I cannot find the {containerId}";
            }
        }
        else
        {
            container = player;
        }

        GameObject item = container.Locate(itemId);

        if (item == null)
        {
            return $"I cannot find the {itemId} in  {container.Name}";
        }

        return item.GetFullDescription();
      }

      return "I don't understand what you want to do.";
    }

    private IHaveInventory FetchContainer(Player player, string containerId)
    {
      return player.Locate(containerId) as IHaveInventory;
    }
  }
}
```

Player.cs:

```
using MiNET.Utils.Skins;
using System;

namespace SwinAdventure
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        private Location _location;

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

        public override string GetFullDescription()
        {

            return $"You are {Name}, {ShortDescription}. You are holding " + _inventory.ItemList;
        }
        public Location Location
        {
            get { return _location; }
            set { _location = value; }
        }

        public GameObject Locate(string id)
        {
            if (AreYou(id)) return this;
```

```
            GameObject item = Inventory.Fetch(id);
            if (item != null) return item;

            return _location?.Locate(id);
        }

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

Program.cs:

```
using System;
using SwinAdventure;

class Program
{
    static void Main(string[] args)
    {
        // Display list of available commands at the beginning
        Console.WriteLine("Welcome to Swin-Adventure!");
        Console.WriteLine("Here are the commands you can use:");
        Console.WriteLine(" - 'look at [item]': Look at an item in your inventory or in the location.");
        Console.WriteLine(" - 'look at [item] in [container]': Look at an item inside a container (e.g., a bag).");
        Console.WriteLine(" - 'look': Look around your current location and see items there.");
        Console.WriteLine(" - 'exit': Exit the game.\n");
```

```
// Step 1: Get the player's name and description from the user.
Console.Write("Enter your player's name: ");
string playerName = Console.ReadLine();

Console.Write("Enter a description for your player: ");
string playerDescription = Console.ReadLine();

// Create a Player object with the user's input.
Player player = new Player(playerName, playerDescription);
Console.WriteLine($"\nWelcome {playerName}, {playerDescription}!\n");

// Step 2: Create items and add them to the player's inventory.
Item sword = new Item(new string[] { "sword" }, "a sword", "A sharp, shining sword.");
Item shield = new Item(new string[] { "shield" }, "a shield", "A sturdy wooden shield.");
player.Inventory.Put(sword);
player.Inventory.Put(shield);
Console.WriteLine("Items 'sword' and 'shield' have been added to your inventory.\n");

// Step 3: Create a bag and add it to the player's inventory.
Bag smallBag = new Bag(new string[] { "bag", "small bag" }, "a small bag", "A small leather bag.");
player.Inventory.Put(smallBag);
Console.WriteLine("A small bag has been added to your inventory.\n");

// Step 4: Create another item and add it to the bag.
Item gem = new Item(new string[] { "gem" }, "a gem", "A shiny, valuable gem.");
smallBag.Inventory.Put(gem);
Console.WriteLine("Item 'gem' has been placed inside the small bag.\n");

// Step 5: Set up the location and add items to it
```

```
Location darkCave = new Location(new string[] { "cave" }, "Dark Cave", "A spooky, dark cave.");
Item torch = new Item(new string[] { "torch" }, "a torch", "A torch that provides light.");
darkCave.Inventory.Put(torch);  // Place the torch in the dark cave
Console.WriteLine("Item 'torch' has been placed in the Dark Cave.\n");

// Set the player's initial location
player.Location = darkCave;

// Step 6: Create the LookCommand
LookCommand lookCommand = new LookCommand();

// Step 7: Main command loop
while (true)
{
    Console.WriteLine("\nEnter a command (or type 'exit' to quit):");
    string command = Console.ReadLine();

    if (command.ToLower() == "exit")
    {
        break; // Exit the loop and end the program.
    }

    // Split the command into an array of words
    string[] commandWords = command.Split(' ');

    // Execute the LookCommand with the player and command words
    string result = lookCommand.Execute(player, commandWords);
    Console.WriteLine(result);
}

Console.WriteLine("Goodbye!");
```

```
    }
}
```

Player Location test:

```
using NUnit.Framework;
using System.Numerics;
using SwinAdventure;
namespace SwinAdventure.Tests
{
    [TestFixture]
    public class PlayerTests
    {
        private Player _player;
        private Location _location;
        private Item _inventoryItem;
        private Item _locationItem;

        [SetUp]
        public void SetUp()
        {
            _player = new Player("Hero", "A brave adventurer.");
            _location = new Location(new string[] { "forest" }, "Dark Forest", "A spooky dark forest.");
            _inventoryItem = new Item(new string[] { "sword" }, "Sword", "A sharp blade.");
            _locationItem = new Item(new string[] { "shield" }, "Shield", "A sturdy shield.");

            _player.Inventory.Put(_inventoryItem);
            _location.Inventory.Put(_locationItem);
            _player.Location = _location;
        }

        [Test]
```

```
        public void PlayerCanLocateItself()
        {
            Assert.AreEqual(_player, _player.Locate("inventory"));
        }

        [Test]
        public void PlayerCanLocateItemInInventory()
        {
            Assert.AreEqual(_inventoryItem, _player.Locate("sword"));
        }

        [Test]
        public void PlayerCanLocateItemInLocation()
        {
            Assert.AreEqual(_locationItem, _player.Locate("shield"));
        }

        [Test]
        public void PlayerCannotLocateNonexistentItem()
        {
            Assert.IsNull(_player.Locate("potion"));
        }
    }
}
```

LookCommand Location testing:

```
using NUnit.Framework;
using System.Numerics;
using SwinAdventure;

namespace SwinAdventure.Tests
```

```
{
    [TestFixture]
    public class LookCommandTests
    {
        private Player _player;
        private Location _location;
        private Item _inventoryItem;
        private Item _locationItem;
        private LookCommand _lookCommand;

        [SetUp]
        public void SetUp()
        {
            _player = new Player("Hero", "A brave adventurer.");
            _location = new Location(new string[] { "cave" }, "Dark Cave", "A spooky dark cave.");
            _inventoryItem = new Item(new string[] { "sword" }, "Sword", "A sharp blade.");
            _locationItem = new Item(new string[] { "shield" }, "Shield", "A sturdy shield.");

            _player.Inventory.Put(_inventoryItem);
            _location.Inventory.Put(_locationItem);
            _player.Location = _location;

            _lookCommand = new LookCommand();
        }

        [Test]
        public void LookAtItemInInventory()
        {
            string result = _lookCommand.Execute(_player, new string[] { "look", "at", "sword" });
            Assert.AreEqual("Sword: A sharp blade.", result);
        }
```

```
        [Test]
        public void LookAtItemInLocation()
        {
            string result = _lookCommand.Execute(_player, new string[] { "look", "at", "shield" });
            Assert.AreEqual("Shield: A sturdy shield.", result);
        }

        [Test]
        public void LookAroundLocation()
        {
            string result = _lookCommand.Execute(_player, new string[] { "look", "around" });
            Assert.AreEqual("Dark Cave: A spooky dark cave.", result);
        }

        [Test]
        public void LookAtNonexistentItem()
        {
            string result = _lookCommand.Execute(_player, new string[] { "look", "at", "potion" });
            Assert.AreEqual("I cannot find the potion in  Hero", result);
        }

        [Test]
        public void LookInNonexistentLocation()
        {
            string result = _lookCommand.Execute(_player, new string[] { "look", "in", "lake" });
            Assert.AreEqual("I don't understand what you want to do.", result);
        }
    }
}
```

Location testing:

```
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventure.Tests
{
    [TestFixture]
    public class LocationTests
    {
        private Location _location;
        private Item _item;

        [SetUp]
        public void SetUp()
        {
            _location = new Location(new string[] { "cave" }, "Dark Cave", "A dimly lit cave.");
            _item = new Item(new string[] { "gem" }, "Shiny Gem", "A sparkling gem lies here.");
            _location.Inventory.Put(_item);
        }

        [Test]
        public void LocationCanIdentifyItself()
        {
            Assert.AreEqual(_location, _location.Locate("cave"));
        }

        [Test]
        public void LocationCanLocateItemInInventory()
        {
            Assert.AreEqual(_item, _location.Locate("gem"));
        }
```

```
        [Test]
        public void LocationCannotLocateNonexistentItem()
        {
            Assert.IsNull(_location.Locate("sword"));
        }
    }
}
```