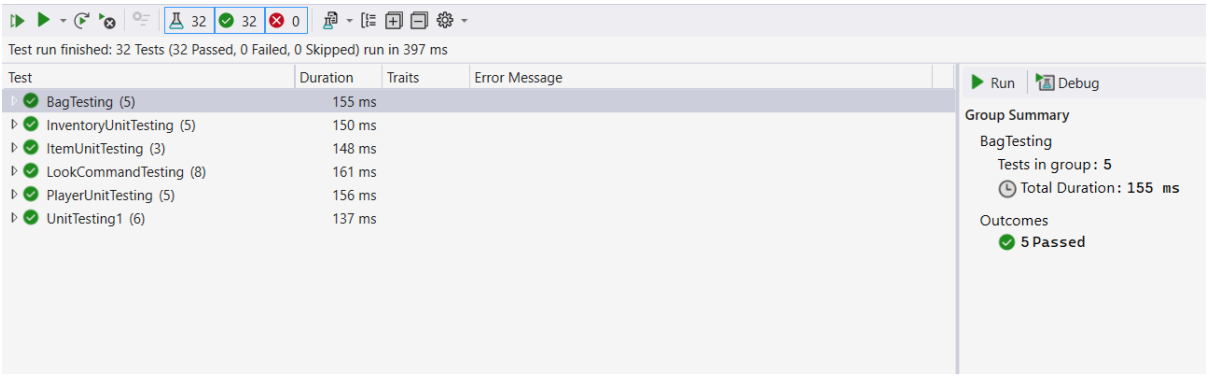

Command.cs:

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

namespace SwinAdventure
{
    public abstract class Command:IdentifiableObject
    {
        public Command(string[] ids) : base(ids) { }

        public abstract string Execute(Player p, string[] text);
    }
}
```

Lookcommand.cs:

```
using SwinAdventure;

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

    public override string Execute(Player player, string[] text)
```

```
{
    // Error if the input is not in the correct format
    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?";
    }

    string itemId = text[2];
    IHaveInventory container;

    // Handle "look at item in 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);
```

```
            // Container not found
            if (container == null)
            {
                return $"I cannot find the {containerId}.";
            }
        }
        else
        {
            // If no container specified, use player inventory
            container = player;
        }

        // Try to locate the item in the container
        GameObject item = container.Locate(itemId);

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

        // Return full description of the located item
        return item.GetFullDescription();
    }

    // Fetch the container (like a bag) where the player is looking
    private IHaveInventory FetchContainer(Player player, string containerId)
    {
        return player.Locate(containerId) as IHaveInventory;
    }
}
```

IHaveInventory:

```
using SwinAdventure;

public interface IHaveInventory
{
    GameObject Locate(string id);
    string Name { get; }
}
```

Player.cs:

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

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


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


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

Bag.cs:

```
using MiNET.Utils.Skins;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public  class Bag : Item, IHaveInventory
    {
```

```
        private Inventory _inventory;

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

        public override string GetFullDescription()
        {

            return $" You are {Name}, a brave adventurer. You are holding" + _inventory.ItemList;
        }
        public Inventory Inventory
        {
            get { return _inventory; }
        }

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

}
```

Testlookcommand.cs:

```
using NUnit.Framework;
```

```
namespace SwinAdventure.Tests
{
    [TestFixture]
    public class LookCommandTests
    {
        private Player _player;
        private LookCommand _lookCommand;
        private Item _gem;
        private Bag _bag;

        [SetUp]
        public void Setup()
        {
            // Initialize LookCommand
            _lookCommand = new LookCommand();

            // Initialize player, gem, and bag
            _player = new Player("Player", "A brave adventurer");
            _gem = new Item(new string[] { "gem" }, "a gem", "A shiny red gem");
            _bag = new Bag(new string[] { "bag" }, "a small bag", "A small leather bag");

            // Add gem to the player's inventory
            _player.Inventory.Put(_gem);

            // Add the bag to the player's inventory
            _player.Inventory.Put(_bag);

            // Add another gem to the bag
            Item anotherGem = new Item(new string[] { "gem" }, "another gem", "Another shiny gem");
            _bag.Inventory.Put(anotherGem);
```

```
    }

    [Test]
    public void TestLookAtMe()
    {
        string result = _lookCommand.Execute(_player, new string[] { "look", "at", "inventory" });
        Assert.AreEqual("You are Player, A brave adventurer. You are holding \tA shiny red gem (gem)\n\tA small leather bag (bag)\n", result);
    }

    [Test]
    public void TestLookAtGem()
    {
        string result = _lookCommand.Execute(_player, new string[] { "look", "at", "gem" });
        Assert.AreEqual("a gem: A shiny red gem", result);
    }

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

    [Test]
    public void TestLookAtGemInMe()
    {
        string result = _lookCommand.Execute(_player, new string[] { "look", "at", "gem", "in", "inventory" });
        Assert.AreEqual("a gem: A shiny red gem", result);
    }
```

```
    [Test]
    public void TestLookAtGemInBag()
    {
        string result = _lookCommand.Execute(_player, new string[] { "look", "at", "gem", "in", "bag" });
        Assert.AreEqual("another gem: Another shiny gem", result);
    }

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

    [Test]
    public void TestLookAtNoGemInBag()
    {
        Player player = new Player("Player", "A brave adventurer");
        Bag smallBag = new Bag(new string[] { "small bag" }, "small bag", "A small leather bag");

        Item anotherGem = new Item(new string[] { "another gem" }, "another gem", "Another shiny gem");
        player.Inventory.Put(smallBag);
        smallBag.Inventory.Put(anotherGem);

        LookCommand lookCommand = new LookCommand();
        string result = lookCommand.Execute(player, new string[] { "look", "at", "gem", "in", "small bag" });

        Assert.AreEqual("I cannot find the gem in the small bag.", result);
```

```
        }



        [Test]
        public void TestInvalidLook()
        {
            string result = _lookCommand.Execute(_player, new string[] { "hello" });
            Assert.AreEqual("I don't know how to look like that", result);
        }
    }
}
```