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

namespace IdentifiableObject
{
    public class Bag:Item, IHaveInventory
    {
        private Inventory _inventory;

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

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

        public string FullDescription
        {
            get
            {
                return $"In the {this.Name} you can see:\n{_inventory.ItemList}";
            }
        }

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

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

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

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

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

        }

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

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

```

...P\Swin_Adventure\IdentifiableObject\IHaveInventory.cs 1

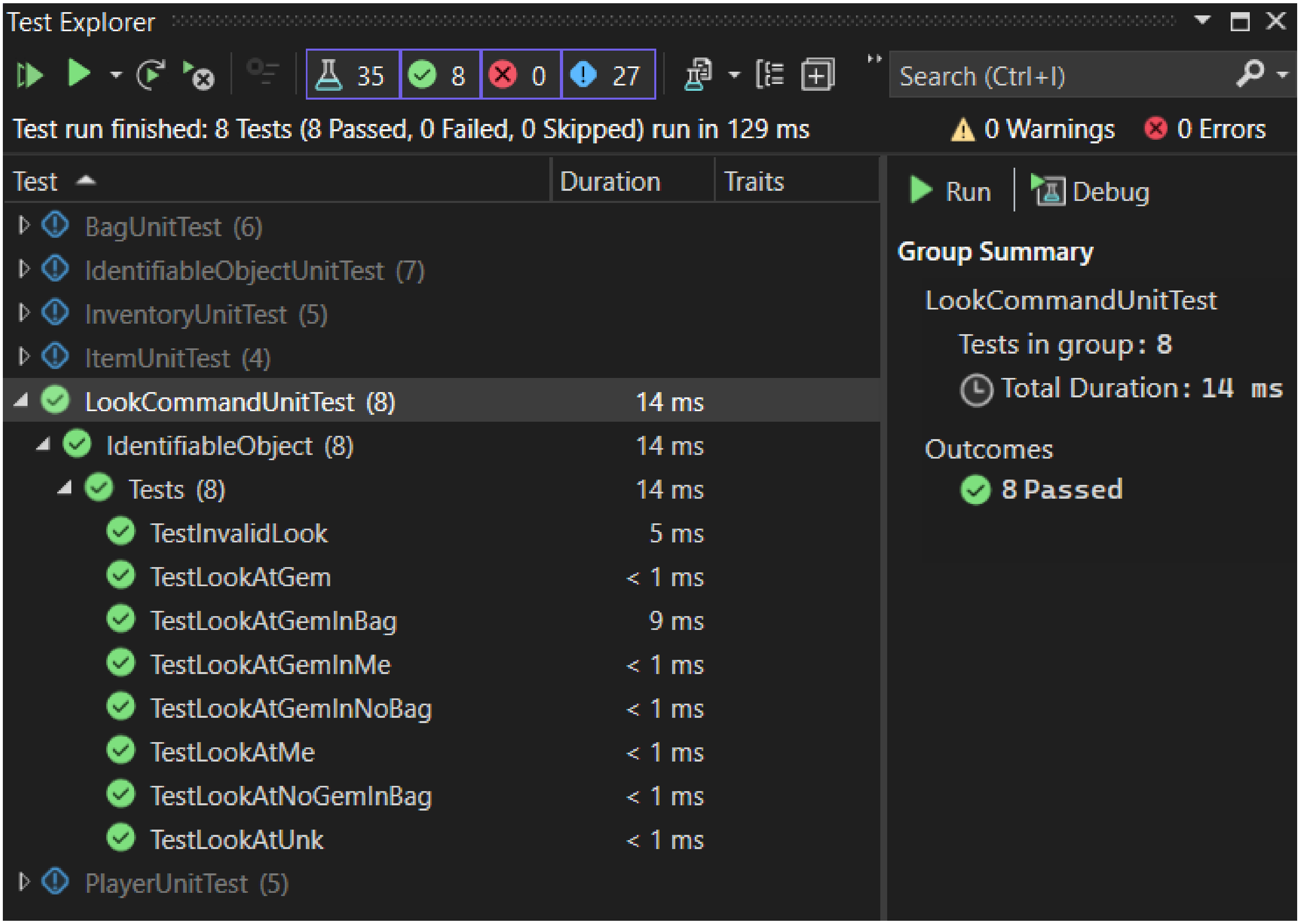
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IdentifiableObject
{
    public interface IHaveInventory
    {
        public GameObject Locate(string id);

        public string Name
        {
            get;
        }
    }
}

```

...ek6\6.1P\Swin_Adventure\IdentifiableObject\Command.cs 1

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

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

        }

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

```

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

namespace IdentifiableObject
{
    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)
            {
                if (text[0].ToLower() == "look")
                {
                    if (text[1].ToLower() == "at")
                    {
                        if(text.Length==3)
                        {
                            return LookAtIn(text[2], p);
                        }
                        else if(text.Length==5)
                        {
                            if (text[3].ToLower() == "in")
                            {
                                return LookAtIn(text[2], FetchContainer(p, text[4]));
                            }
                            else
                            {
                                return "What do you want to look in?";
                            }
                        }
                        else
                        {
                            return "I don't know how to look like that";
                        }
                    }
                    else return "What do you want to look at?";
                }
                else
                {
                    return "Error in look input";
                }
            }
            else
            {
                return "I don't know how to look like that";
            }
        }

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

        private string LookAtIn(string thingId, IHaveInventory container)
        {
            if(container.Locate(thingId)!=null)
            {
                return container.Locate(thingId).FullDescription;
            }
            return "I can't find the " + thingId;
        }
    }
}

```

```
namespace IdentifiableObject
{
    public class Tests
    {
        private Player _player;
        private Player _player_no_bag;
        private Item _gem;
        private Bag _bag;
        private Command _look;

        [SetUp]
        public void Setup()
        {
            _look = new LookCommand();
            _player = new Player("Duc Thang", "Student");
            _player_no_bag = new Player("player", "participant");
            _gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");
            _bag = new Bag(new string[] { "bag" }, "Thang's bag", "student");
            _player.Inventory.Put(_bag);
        }

        [Test]
        public void TestLookAtMe()
        {
            string look_execution = _look.Execute(_player, new string[] { "look", "at", "inventory" });
            string output = _player.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGem()
        {
            _player.Inventory.Put(_gem);
            string look_execution = _look.Execute(_player, new string[] {"look", "at", "gem"});
            string output = _gem.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtUnk()
        {
            string look_execution = _look.Execute(_player, new string[] { "look", "at", "gem" });
            string output = "I can't find the gem";
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGemInMe()
        {
            _player.Inventory.Put(_gem);
            string look_execution = _look.Execute(_player, new string[] { "look", "at", "gem", "in", "me" });
            string output = _gem.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGemInBag()
        {
            _bag.Inventory.Put(_gem);
            string look_execution = _look.Execute(_player, new string[] { "look", "at", "gem", "in", "bag" });
            string output = _gem.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGemInNoBag()
        {
            string look_execution = _look.Execute(_player_no_bag, new string[] { "look", "at", "bag" });
            string output = "I can't find the bag";
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtNoGemInBag()
        {
            string look_execution = _look.Execute(_player, new string[] { "look", "at", "gem", "in", "bag" });
            string output = "I can't find the gem";
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestInvalidLook()
        {
            Assert.AreEqual(_look.Execute(_player, new string[] { "look", "around" }), "I don't know how to look like that");
            Assert.AreEqual(_look.Execute(_player, new string[] { "hello", "104776473" }), "I don't know how to look like that");
            Assert.AreEqual(_look.Execute(_player, new string[] { "look", "at", "Nguyen Duc Thang" }), "I can't find the Nguyen Duc
              Thang");
        }
    }
}
```



Test Explorer

35 | 8 | 0 | 27

Search (Ctrl+I)

Test run finished: 8 Tests (8 Passed, 0 Failed, 0 Skipped) run in 129 ms

0 Warnings 0 Errors

```
Test                               Duration    Traits
▷ BagUnitTest (6)
▷ IdentifiableObjectUnitTest (7)
▷ InventoryUnitTest (5)
▷ ItemUnitTest (4)
◢ LookCommandUnitTest (8)           14 ms
  ◢ IdentifiableObject (8)          14 ms
    ◢ Tests (8)                     14 ms
        TestInvalidLook              5 ms
        TestLookAtGem              < 1 ms
        TestLookAtGemInBag           9 ms
        TestLookAtGemInMe          < 1 ms
        TestLookAtGemInNoBag       < 1 ms
        TestLookAtMe               < 1 ms
        TestLookAtNoGemInBag       < 1 ms
        TestLookAtUnk              < 1 ms
▷ PlayerUnitTest (5)
```

Group Summary

LookCommandUnitTest

Tests in group: 8

Total Duration: 14 ms

Outcomes

8 Passed