6.1 ilteration 4
Source code
Command.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace SwinAdventure4

{

   public abstract class Command : Identifiable_Object //inherit directly from
identifiableObject

   {


       public Command(string[] idents) : base(idents)

       {


       }

       public abstract string Execute(Player p, string[] Text); // abstract to override in
LookCommand






   }

}
```

LookCommand.cs

```
using System;

using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace SwinAdventure4
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "Look" })
        {

        }

        public override string Execute(Player p, string[] text)
        {
            // Check "look" first, convert to lowercase
            if (text[0].ToLower() != "look")
            {
                return "Error in look input";
            }

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

```
}

if (text[1].ToLower() != "at")
{
    return "What do you want to look at?";
}

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

// Determine the container (Player or another object)
IhaveInv container;
if (text.Length == 3) // "Look at [thing]"
{
    // Handle "look at me" or "look at inventory"
    if (text[2].ToLower() == "me" || text[2].ToLower() == "inventory")
    {
        return p.FullDescription; // Return the player's full description
    }
    container = p;
}
else // "Look at [thing] in [container]"
{
    container = FetchContainer(p, text[4]);
    if (container == null)
    {
```

```
                return $"I can't find the {text[4]}";

            }

        }

        return LookAtLn(text[2], container);

    }



    private IhaveInv FetchContainer(Player p, string ContainerId) //use ihaveInv fetch
from thr

    {

        return p.Locate(ContainerId) as IhaveInv;

    }


  private string LookAtLn(string thingId, IhaveInv container)

    {

        GameObject item = container.Locate(thingId) as GameObject; // uses depency
from GameObject

        if (item != null)

        {

            return item.FullDescription;

        }

        return "Couldn't Find";


    }

  }

}
```
IhaveInv.cs

```
using System;

using System.Collections.Generic;
using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace SwinAdventure4

{

  public interface IhaveInv

  {

    GameObject Locate(string id);  //From GameObject to be used by LookCommand
by creating dependency

    string Name //read only name

    {

      get;

    }


  }

}
```

Player.cs

```
using SwinAdventure4;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;


namespace SwinAdventure4

{

  public class Player : GameObject , IhaveInv //inheritance from bag and player
  {
      private Inventory _inventory;

      public Player(string name, string desc) : base(new string[] { "Me", "Inventory " },
name, desc) //overide new info for name and description

      {
          _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 {Name}, " + base.FullDescription + ".\nYou are carrying\n" +
_inventory.ItemList; //display our name is carrying itemlist which it varries between total
list length
          }
      }
```

```
        public Inventory Inventory

        {

            get => _inventory;

        }


    }


}
```

Bag.cs

```
using SwinAdventure4;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace SwinAdventure4

{

    public class Bag : Item , IhaveInv // inherintance from Item and IhaveInv

    {

        Inventory _inventory;


        public Bag(string[] idents, string name, string description) : base(idents, name,
description)

        {

            _inventory = new Inventory();// taking the the list from inventory then initilize it


        }

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


            }
            else if (_inventory.HasItem(id))
            {
                return (_inventory.Fetch(id));
            }
            return null;


        }


        public Inventory Inventory //read only property
        {
            get


            { return _inventory; }
        }


    }


}
```

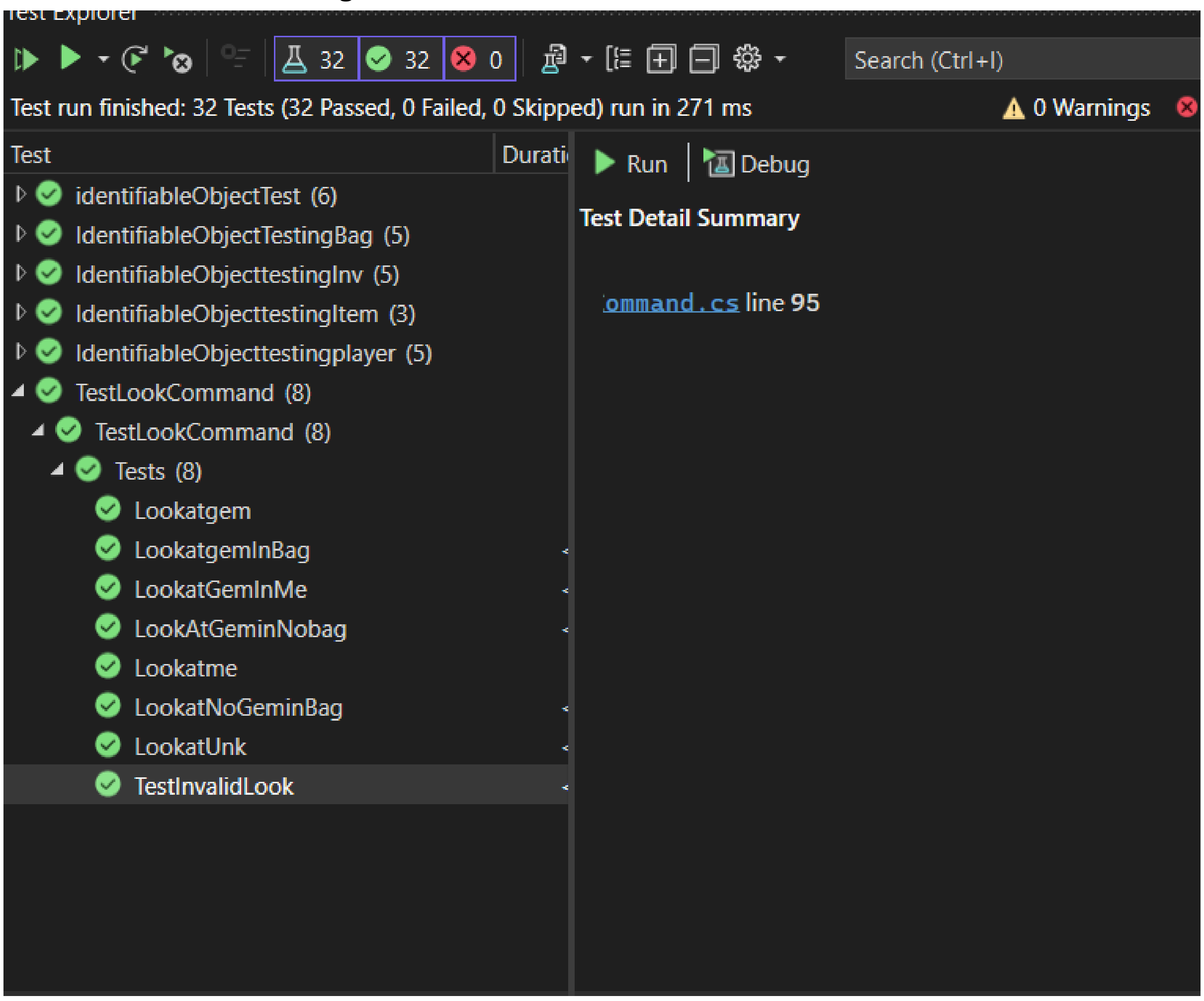
TestLookCommand.cs

```
using System;

using System.Collections.Generic;

using System.Runtime.CompilerServices;
using NUnit.Framework;
using SwinAdventure4;

namespace TestLookCommand
{
  public class Tests
  {
    private LookCommand Look;
    private Player player;
    private Bag bag;

    Item gem;
    Item gun;
    Item katana;

    [SetUp]
    public void Setup()
    {
      Look = new LookCommand();
      player = new Player("Bryan", "Bryan's player");
      gem = new Item(new string[] { "gem" }, "Bryan's gem", "This is a huge gem");
      gun = new Item(new string[] { "gun" }, "Bryan's gun", "This is a powerful gun");
      katana = new Item(new string[] { "katana" }, "Bryan's katana", "This is a sharp
katana");
      bag = new Bag(new string[] { "bag" }, "Bryan's bag", "This is a big bag");

      // Add items to the player's inventory
      player.Inventory.Put(gem);
    player.Inventory.Put(bag);

    // Add items to the bag
    bag.Inventory.Put(gun);
    bag.Inventory.Put(katana);
}

[Test]
public void Lookatme()
{
    // Test looking at the player's inventory
    string Output = Look.Execute(player, new string[] { "look", "at", "inventory" });
    Assert.AreEqual(player.FullDescription, Output);
}

[Test]
public void Lookatgem()
{
    string Output = Look.Execute(player, new string[] { "look", "at", "gem" });
    Assert.AreEqual(gem.FullDescription, Output);
}

[Test]
public void LookatUnk()
{
    player.Inventory.take("gem");
    string Output = Look.Execute(player, new string[] { "look", "at", "gem" });
    Assert.AreEqual("Couldn't Find", Output);
```

```
}


[Test]
public void LookatGemInMe()
{


   string Output = Look.Execute(player, new string[] { "look", "at", "gem", "in", "me" });
   Assert.AreEqual (gem.FullDescription, Output);
}


[Test]
public void LookatgemInBag()
{
   bag.Inventory.Put(gem);
   string Output = Look.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });
   Assert.AreEqual(gem.FullDescription, Output);
}


[Test]
public void LookAtGeminNobag()
{
   String Output = Look.Execute(player, new string[] { "look", "at", "gem", "in","bag" });
   String expected = $"Couldn't Find";
   Assert.AreEqual(expected, Output);
}


[Test]
public void LookatNoGeminBag()
```

```
        {
          string Output = Look.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });
          string expected = $"Couldn't Find";
          Assert.AreEqual(expected , Output);


        }


        [Test]
        public void TestInvalidLook()
        {
          Assert.AreEqual("I don't know how to look like that", Look.Execute(player, new
string[] { "look", "around" }));
          Assert.AreEqual("What do you want to look at?", Look.Execute(player, new string[]
{ "look", "for", "gem" }));
          Assert.AreEqual("Error in look input",  Look.Execute(player, new string[] { "find",
"gem" }));
        }
    }
}
```

Screenshot of unit testing



```
Test run finished: 32 Tests (32 Passed, 0 Failed, 0 Skipped) run in 271 ms
```

Test
✓ identifiableObjectTest (6)
✓ IdentifiableObjectTestingBag (5)
✓ IdentifiableObjecttestingInv (5)
✓ IdentifiableObjecttestingItem (3)
✓ IdentifiableObjecttestingplayer (5)
✓ TestLookCommand (8)
✓ TestLookCommand (8)
✓ Tests (8)
✓ Lookatgem
✓ LookatgemInBag
✓ LookatGemInMe
✓ LookAtGeminNobag
✓ Lookatme
✓ LookatNoGeminBag
✓ LookatUnk
✓ TestInvalidLook

Test Detail Summary

ommand.cs line 95