Iteration 7
Source Code
path.cs

```
using SwinAdventure4;

using System;

using System.Collections.Generic;

using System.Data.Common;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;


namespace SwinAdventure4

{

  public class Path : GameObject

  {

    private bool _isblocked;


     Location _source, _destination;


     public Path(string[] idents, string name, string desc, Location source, Location destination) : base(idents, name, desc)

     {

       _source = source;

       _destination = destination;

       _isblocked = false;


       AddIdentifier("path");

       foreach (string s in name.Split(" "))
```

```
        {
            AddIdentifier(s);
        }

    }

    public Location Destination
    {
        get { return _destination; }
    }

    public override string ShortDescription
    {
        get { return Name; }
    }

    public bool IsBlocked
    {
        get { return _isblocked; }
        set { _isblocked = value; }
    }

  }
}
```

Move.cs

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

```
using System.Text;
using System.Threading.Tasks;

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

      }
      public override string Execute(Player p, string[] Text)
      {
         string error = "Error in move input";
         if (Text.Length > 2)
         {
            return "Move Where?";
         }

         String direction = Text[1].ToLower();
         GameObject path = p.Location.Locate(direction);

         if (path is Path targetPath)
         {
            if (!targetPath.IsBlocked)
            {
               p.Move(targetPath);
```

```
                return $"You have moved {direction} through a {targetPath.Name} to the
{p.Location.Name}.\n\n{p.Location.FullDescription}";
            }

            else
            {
                return $"The Path to the {targetPath.Name} is blocked.";
            }
        }
        return error;
    }
  }
}
```

Player.cs

```
using SwinAdventure4;
using System;
using System.Collections.Generic;
using System.Data.Common;
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;
    private Location _location;
```

```
    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)) //first checking locate
        {
            return this;
        }
        GameObject obj = _inventory.Fetch(id);
        if (obj != null) //second checking for object
        {
            return obj;
        }
        if (_location != null) //third checking for location
        {
            obj = _location.Locate(id);
            return obj;
        }
        else
        {
            return null;
```

```
        }



    }
    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;
    }
    public Location Location
    {
        get => _location;
        set => _location = value; //get and set value into _location
    }


    public void Move(Path path)
    {
        if (path.Destination != null)
        {
            _location = path.Destination;
```

```
        }
      }
    }

}
```

Location.cs

```
using SwinAdventure4;
using System;
using System.Collections.Generic;
using System.Data.Common;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace SwinAdventure4
{
   public class Location : GameObject, IhaveInv //inheritance from bag and player
   {
      private Inventory _inventory;
      private string _name, _desc;
      List<Path> _paths;

      public Location(string name, string desc) : base(new string[] { "location" }, name,
desc)
      {
         _inventory = new Inventory();
         _paths = new List<Path>();
      }
```

```
public Location(string name, string desc, List<Path> paths) : this(name, desc)
{
   _paths = paths;
}

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

   foreach (Path p in _paths)
   {
      if (p.AreYou(id))
      {
         return p;
      }
   }

   return _inventory.Fetch(id);
}

public string PathList
{
   get
   {
```

```
        string list = string.Empty + "\n";

        if (_paths.Count == 1)
        {
            return "There is an exit " + _paths[0].FirstId + ".";
        }

        list = list + "There are exits to the ";

        for (int i = 0; i < _paths.Count; i++)
        {
            if (i == _paths.Count - 1)
            {
                list = list + "and " + _paths[i].FirstId + ".";
            }
            else
            {
                list = list + _paths[i].FirstId + ", ";
            }
        }

        return list;
    }
}

public string ItemList
{
    get
```

```
        {
            if (_inventory.Count == 0)
            {
                return string.Empty;
            }
            return "In the room you see:\n" + Inventory.ItemList;
        }
    }

    public override string ShortDescription
    {
        get
        {
            return "You are in a " + Name;
        }
    }

    public override string FullDescription
    {
        get
        {
            return $"Room Description: {base.FullDescription}\n\nItems at this
location:\n{ItemList}{PathList}\n";
            //return base.FullDesciption + "\n" + ItemList + PathList;

        }
    }
```

```
        public Inventory Inventory
        {
            get => _inventory;
        }

        public void AddPath(Path path)
        {
            _paths.Add(path);
        }




    }

}
```

Program.cs

```
using SwinAdventure4;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using System.Xml.Linq;

namespace SwinAdventure4
{
```

```
public class Program
{
    static void LookCommandExe(Command l, string Input, Player player)
    {
        Console.WriteLine(l.Execute(player, Input.Split()));
    }

    static void Main(string[] args)
    {
        //Greeting + info
        string name, desc;
        string help = "-look\n\nGetting list of item:\n-look at me\n-look at bag\n\nGetting item description:\nlook at {item}\nlook at {item} in me\nlook at {item} in bag\n\n";
        Console.WriteLine(help);



        //Setting up player

        Console.Write("Setting up player:\nPlayer Name: ");
        name = Console.ReadLine();
        Console.Write("Player Description: ");
        desc = Console.ReadLine();
        Player player = new Player(name, desc);

        //setting a location
        Location Myroom = new Location("MyRoom", $"This is my Room");
        player.Location = Myroom;
```

```
Location GamingRoom = new Location("GamingRoom", "Gaming Room");

Path MyroomtoGamingRoom = new Path(new string[] { "north" }, "Door", "Travel through door", Myroom, GamingRoom); //create a link from Myroom to GamingRoom (north of MyRoom)

Path GamingRoomtoMyroom = new Path(new string[] { "south" }, "Door", "Travel through door", GamingRoom, Myroom);

Myroom.AddPath(MyroomtoGamingRoom);// add the path

GamingRoom.AddPath(GamingRoomtoMyroom);


Location Kitchen = new Location("Kitchen", "Kitchen");

Path MyRoomToKitchen = new Path(new string[] { "east" }, "Door", "Travel through door", Myroom, Kitchen); //create a link from Myroom to GamingRoom (north of MyRoom)

Path KitchenToMyRoom = new Path(new string[] { "west" }, "Door", "Travel through door", Kitchen, Myroom);

Myroom.AddPath(MyRoomToKitchen);

Kitchen.AddPath(KitchenToMyRoom);// add the path


Location Porch = new Location("Porch", "Car Porch");

Path KitchenToPorch = new Path(new string[] { "north" }, "Door", "Travel through door", Kitchen, Porch); //create a link from Myroom to GamingRoom (north of MyRoom)

Path PorchToKitchen = new Path(new string[] { "south" }, "Door", "Travel through door", Porch, Kitchen); //way back to kitchen

Kitchen.AddPath(KitchenToPorch);

Porch.AddPath(PorchToKitchen);// add the path




//Setting up list of items
```

```
Item bed = new Item(new string[] { "Bed" }, "a Bed", "This is a Bed");

Item PC = new Item(new string[] { "PC" }, "a PC", "This is a PC");

Item Nintendo = new Item(new string[] { "Nintendo" }, "a Nintendo", "This is a
Nintendo");

Item closet = new Item(new string[] { "closet" }, "a closet", "This is a closet");

Item Dishwasher = new Item(new string[] { "Dishwasher" }, "a Dishwasher", "This is
a Dishwasher");

Item Stove = new Item(new string[] { "Stove" }, "a stove", "This is a Stove");

Item plants = new Item(new string[] { "plants" }, "some plants", "This is some
plants");

Item ShoeRack = new Item(new string[] { "Shoe Rack" }, "a Shoe Rack", "This is a
Shoe Rack");


Myroom.Inventory.Put(bed);//Myroom

Myroom.Inventory.Put(closet);


GamingRoom.Inventory.Put(PC);//gamingroom

GamingRoom.Inventory.Put(Nintendo);


Kitchen.Inventory.Put(Dishwasher);//kitchen

Kitchen.Inventory.Put(Stove);


Porch.Inventory.Put(plants); //porch

Porch.Inventory.Put(ShoeRack);
```

```
        Item shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel"); //
declare two items
        Item sword = new Item(new string[] { "sword" }, "a sword", "This is a sword");
        player.Inventory.Put(shovel); //put 2 item in iventory
        player.Inventory.Put(sword);

        Bag bag = new Bag(new string[] { $"bag" }, $"{player.Name}'s bag", $"This is
{player.Name}'s bag"); //create a bag
        player.Inventory.Put(bag); //place item in bag
        Item diamond = new Item(new string[] { "diamond" }, "a diamond", "This is a
diamond");
        Item Phone = new Item(new string[] { "Phone" }, "a phone", "This is a phone");
        bag.Inventory.Put(Phone);
        bag.Inventory.Put(diamond);

        string _input;
       LookCommand Look= (new LookCommand());
       Move Move=(new Move());


        while (true)
        {
           Console.Write("Command: ");
           _input = Console.ReadLine();


           string[] split;
           split = _input.Split(' ');
```

```
        if (Look.AreYou(split[0].ToLower()))
        {
            Console.WriteLine(Look.Execute(player, split));
        }
        else if (Move.AreYou(split[0].ToLower()))
        {
            Console.WriteLine(Move.Execute(player, split));
        }



        else if (_input == "Inventory")
        {
            Console.WriteLine(player.Inventory.ItemList);
        }

        else
        {
            Console.WriteLine("Bye");
            Console.ReadLine();
            break;
        }

    }


}
```

```
    }
}
```

TestPath.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using NUnit.Framework;
using SwinAdventure4;

namespace IdentifiableObjectTestingPath

{
    public class PathTest
    {
        Player _testPlayer;
        Location _testRoomA;
        Location _testRoomB;
        SwinAdventure4.Path _testPath;

        [Test]
        public void TestPathLocation()
        {
            _testPlayer = new Player("Danny", "The Player!");

            _testRoomA = new Location("Room A", "Room A");
            _testRoomB = new Location("Room B", "Room B");
```

```
        _testPlayer.Location = _testRoomA;
        _testPath = new SwinAdventure4.Path(new string[] { "north" }, "Door", "A test door",
_testRoomA, _testRoomB);
        _testRoomA.AddPath(_testPath);

        Location _expected = _testRoomB;
        Location _actual = _testPath.Destination;

        Assert.AreEqual(_expected, _actual);
    }

    [Test]
    public void TestPathName()
    {
        _testPlayer = new Player("Danny", "The Player!");

        _testRoomA = new Location("Room A", "Room A");
        _testRoomB = new Location("Room B", "Room B");

        _testPlayer.Location = _testRoomA;
        _testPath = new SwinAdventure4.Path(new string[] { "north" }, "Door", "A test door",
_testRoomA, _testRoomB);
        _testRoomA.AddPath(_testPath);

        string _expected = "A test door";
        string _actual = _testPath.FullDescription;

        Assert.AreEqual(_expected, _actual);
```

```
    }

    [Test]
    public void TestLocatePath()
    {
        _testPlayer = new Player("Danny", "The Player!");

        _testRoomA = new Location("Room A", "Room A");
        _testRoomB = new Location("Room B", "Room B");

        _testPlayer.Location = _testRoomA;
        _testPath = new SwinAdventure4.Path(new string[] { "north" }, "Door", "A test door",
_testRoomA, _testRoomB);
        _testRoomA.AddPath(_testPath);

        GameObject _expected = _testRoomA.Locate("north");
        GameObject _actual = _testPath;

        Assert.AreEqual(_expected, _actual);
    }
  }
}
```

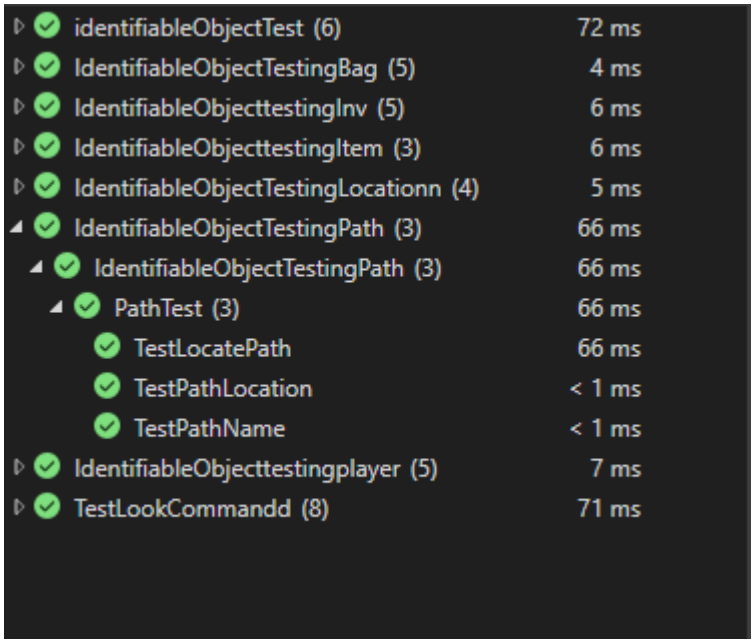
TestLocation

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

```
using NUnit.Framework;
using SwinAdventure4;



namespace IdentifiableObjectTestingLocation
{

    public class TestLocation
    {
        Player p = new Player("Anh", "This is Anh");
        Location l = new Location("MyRoom", "This is my room");
        Item sword = new Item(new string[] { "sword" }, "a sword", "this is a sword");

        [SetUp]
        public void Setup()
        {

        }

        [Test]
        public void TestLookCommand()
        {
            p.Location = l;
            bool actual = l.AreYou("Location");
            Assert.IsTrue(actual);
        }
        [Test]
        public void TestNotLookCommand()
```

```
    {
        p.Location = l;
        bool actual = l.AreYou("hi");
        Assert.IsFalse(actual);
    }

    [Test]
    public void TestPlayerHasLocation()
    {
        p.Location = l;
        GameObject expect = l;
        GameObject actual = p.Locate("location");
        Assert.AreEqual(expect, actual);
    }

    [Test]
    public void TestLocationLocateTest()
    {
        l.Inventory.Put(sword);
        GameObject expect = sword;
        GameObject actual = l.Locate("sword");
        Assert.AreEqual (expect, actual);
    }

  }
}
```

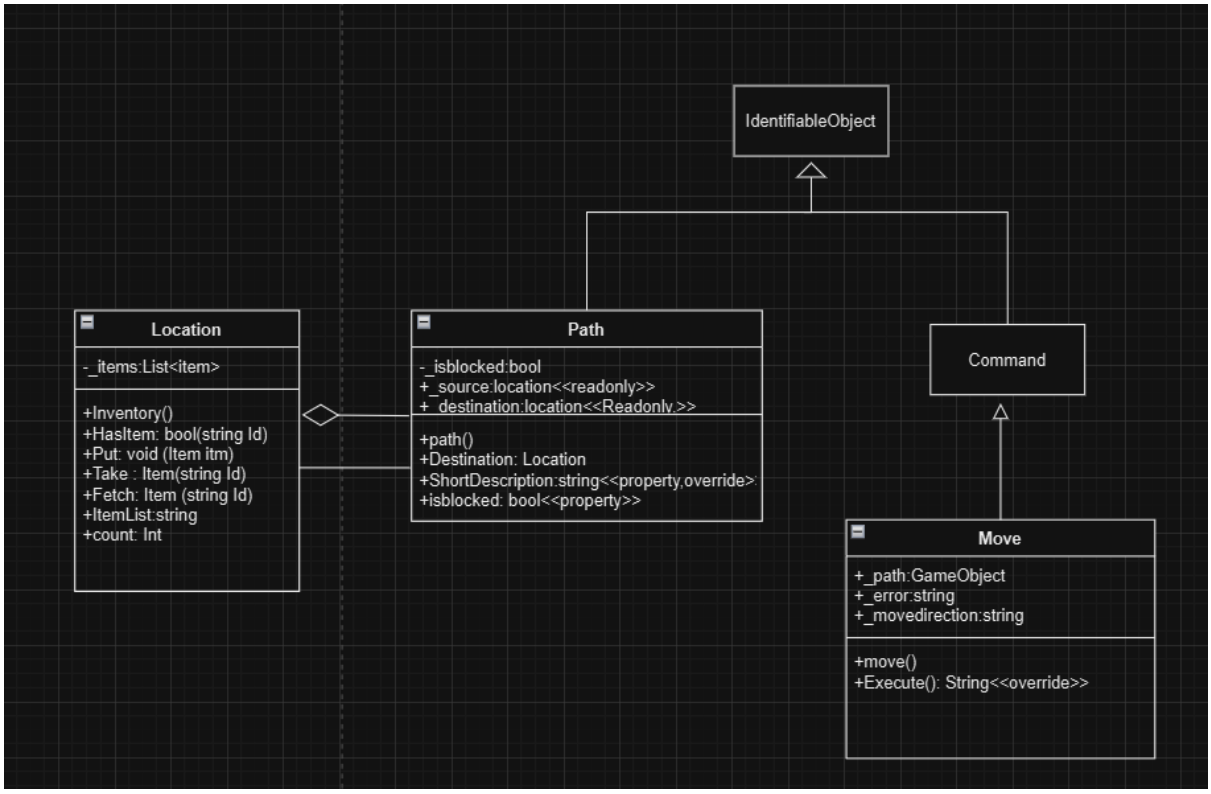
TestUnit



Output

```
Command: move north
Move Where?
Command: move east
Move Where?
Command: Move north
You have moved north through a Door to the GamingRoom.

Room Description: Gaming Room

Items at this location:
In the room you see:
a PC (pc)a Nintendo (nintendo) There is an exit south.

Command: look at pc
This is a PC
Command: look
Room Description: Gaming Room

Items at this location:
In the room you see:
a PC (pc)a Nintendo (nintendo) There is an exit south.

Command: Move south
You have moved south through a Door to the MyRoom.

Room Description: This is my Room

Items at this location:
In the room you see:
a Bed (bed)a closet (closet)
There are exits to the north, and east.

Command: Move east
You have moved east through a Door to the Kitchen.

Room Description: Kitchen

Items at this location:
In the room you see:
a Dishwasher (dishwasher)a stove (stove)
There are exits to the west, and north.

Command: Move north
You have moved north through a Door to the Porch.

Room Description: Car Porch

Items at this location:
In the room you see:
some plants (plants)a Shoe Rack (shoe rack) There is an exit south.
```

UML



sequence diagram