Selected files

12 printable files

Week_7/7.2/SwinAdventure/Bag.cs
Week_7/7.2/SwinAdventure/Command.cs
Week_7/7.2/SwinAdventure/GameObject.cs
Week_7/7.2/SwinAdventure/IdentifiableObject.cs
Week_7/7.2/SwinAdventure/IHaveInventory.cs
Week_7/7.2/SwinAdventure/Inventory.cs
Week_7/7.2/SwinAdventure/Item.cs
Week_7/7.2/SwinAdventure/Location.cs
Week_7/7.2/SwinAdventure/LookCommand.cs
Week_7/7.2/SwinAdventure/Program.cs
Week_7/7.2/SwinAdventure/Player.cs
Week_7/7.2/TestLocation/TestLocation.cs

Week_7/7.2/SwinAdventure/Bag.cs

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class Bag : Item, IHaveInventory
    {
        private Inventory _inventory;
        public Bag(string[] idents, string name, string desc) : base(idents, name,
desc)
        {
            _inventory = new Inventory();
        }

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

            if (_inventory.HasItem(id))
                return _inventory.Fetch(id);

            return null;
        }

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

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

Week_7/7.2/SwinAdventure/Command.cs

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

Week_7/7.2/SwinAdventure/GameObject.cs

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

namespace SwinAdventure
{
    public abstract class GameObject : IdentifiableObject
    {
        private string _description, _name;

        public GameObject(string[] idents, string name, string desc) : base(idents)
        {
            _name = name;
            _description = desc;
        }

        public string Name
        {
            get
```

```
            {
                return _name;
            }
        }

        public string ShortDescription
        {
            get
            {
                return $"{Name} ({FirstId})";
            }
        }

        public virtual string FullDescription
        {
            get
            {
                return _description;
            }
        }
    }
}
```

Week_7/7.2/SwinAdventure/IdentifiableObject.cs

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

namespace SwinAdventure
{

    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();

        public IdentifiableObject(string[] idents)
        {
            foreach (string id in idents)
            {
                AddIdentifier(id);
            }
        }

        public bool AreYou(string id)
        {
            return _identifiers.Contains(id.ToLower());
        }

        public string FirstId
        {
```

```
            get
            {
                if (_identifiers.Count > 0)
                {
                    return _identifiers[0];
                }

                return "";
            }
        }

        public void AddIdentifier(string id)
        {
            _identifiers.Add(id.ToLower());
        }

        public void PrivilegeEscalation(string pin)
        {
            if (pin != "4794")
                return;

            if (_identifiers.Count == 0)
            {
                AddIdentifier("12");
            }
            else
            {
                _identifiers[0] = "12";
            }
        }
    }
}
```

Week_7/7.2/SwinAdventure/IHaveInventory.cs

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

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

Week_7/7.2/SwinAdventure/Inventory.cs

```
using System;
```

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

namespace SwinAdventure
{
    public class Inventory : GameObject
    {
        private List<Item> _items;

        public Inventory() : base(new string[] { "inventory" }, "inventory", "The player's inventory")
        {
            _items = new List<Item>();
        }

        public string ItemList
        {
            get
            {
                List<string> itemsDesc = new List<string>();
                foreach (Item item in _items)
                {
                    itemsDesc.Add("\t" + item.ShortDescription);
                }
                return string.Join("\n", itemsDesc);
            }
        }

        public bool HasItem(string id)
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id))
                {
                    return true;
                }
            }
            return false;
        }

        public void Put(Item itm)
        {
            _items.Add(itm);
        }

        public Item? Take(string id)
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id))
```

```
                {
                    _items.Remove(item);
                    return item;
                }
            }
            return null;
        }

        public Item? Fetch(string id)
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id))
                {
                    return item;
                }
            }
            return null;
        }
    }
}
```

Week_7/7.2/SwinAdventure/Item.cs

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

namespace SwinAdventure
{
    public class Item : GameObject
    {
        public Item(string[] idents, string name, string desc) : base(idents, name,
desc)
        {
        }
    }
}
```

Week_7/7.2/SwinAdventure/Location.cs

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

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

```

```
        public Location(string name, string desc) : base(new string[] { "location" }, name, desc)
        {
            _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 in the {Name}.\n{base.FullDescription}\nIn this location, you can see:\n{_inventory.ItemList}";
            }
        }

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

Week_7/7.2/SwinAdventure/LookCommand.cs

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

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

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

```
            // If text length is not 1,3,5
            if (text.Length != 1 && 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.Length != 1 && text[1] != "at")
                return "What do you want to look at?";

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

            string containerId = "";
            string itemId = "";
            switch (text.Length)
            {
                case 1:
                    containerId = "location";
                    itemId = "location";
                    break;
                case 3:
                    containerId = p.FirstId;
                    itemId = text[2];
                    break;
                case 5:
                    containerId = text[4];
                    itemId = text[2];
                    break;
            }

            IHaveInventory? container = FetchContainer(p, containerId);
            if (container == null)
                return $"I can't find the {containerId}";

            return LookAtIn(itemId, container);
        }

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

        public string LookAtIn(string thingId, IHaveInventory container)
        {
            GameObject? thing = container.Locate(thingId);
            if (thing == null)
                return $"I can't find the {thingId}";

            return thing.FullDescription;
        }
```

```
    }
}
```

Week_7/7.2/SwinAdventure/Program.cs

```
namespace SwinAdventure
{
    class Program
    {
        static void Main()
        {
            string playerName, playerDesc;
            while (true)
            {
                Console.Write("Enter player name: ");
                playerName = Console.ReadLine();
                if (playerName == null)
                {
                    playerName = string.Empty;
                }

                Console.Write("Enter player description: ");
                playerDesc = Console.ReadLine();
                if (playerDesc == null)
                {
                    playerDesc = string.Empty;
                }
                if (string.IsNullOrEmpty(playerName) || string.IsNullOrEmpty(playerDesc))
                {
                    Console.WriteLine("Player name and description cannot be empty.");
                }
                else
                {
                    break;
                }
            }
            Player player = new Player(playerName, playerDesc);

            // Create items and put them in the player's inventory
            Item item1 = new Item(new string[] { "shovel" }, "a shovel", "a wooden shovel");
            Item item2 = new Item(new string[] { "sword" }, "a sword", "a steel sword");
            player.Inventory.Put(item1);
            player.Inventory.Put(item2);

            // Create a bag and put it in the player's inventory
            Bag bag = new Bag(new string[] { "bag" }, "a bag", "a leather bag");
            player.Inventory.Put(bag);

```

```
            // Create items and put them in the bag's inventory
            Item item3 = new Item(new string[] { "coin" }, "a coin", "a shiny coin");
            bag.Inventory.Put(item3);

            // Create location and put some items in its inventory
            Location location = new Location("forest", "A dark forest with tall
trees");
            Item item4 = new Item(new string[] { "rock" }, "a rock", "a big rock");
            Item item5 = new Item(new string[] { "flower" }, "a flower", "a red
flower");
            location.Inventory.Put(item4);
            location.Inventory.Put(item5);

            // Set player's location
            player.Location = location;

            LookCommand look = new LookCommand();

            while (true)
            {
                Console.WriteLine(player.FullDescription);
                Console.Write("> ");
                string command = Console.ReadLine();

                if (string.IsNullOrEmpty(command))
                    continue;
                if (command == "quit")
                    break;

                string response = look.Execute(player, command.Split(" "));
                Console.WriteLine(response);
                Console.WriteLine();
            }
        }
    }
}
```

Week_7/7.2/SwinAdventure/Player.cs

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

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

        public Player(string name, string desc) : base(new string[] { "me",
```

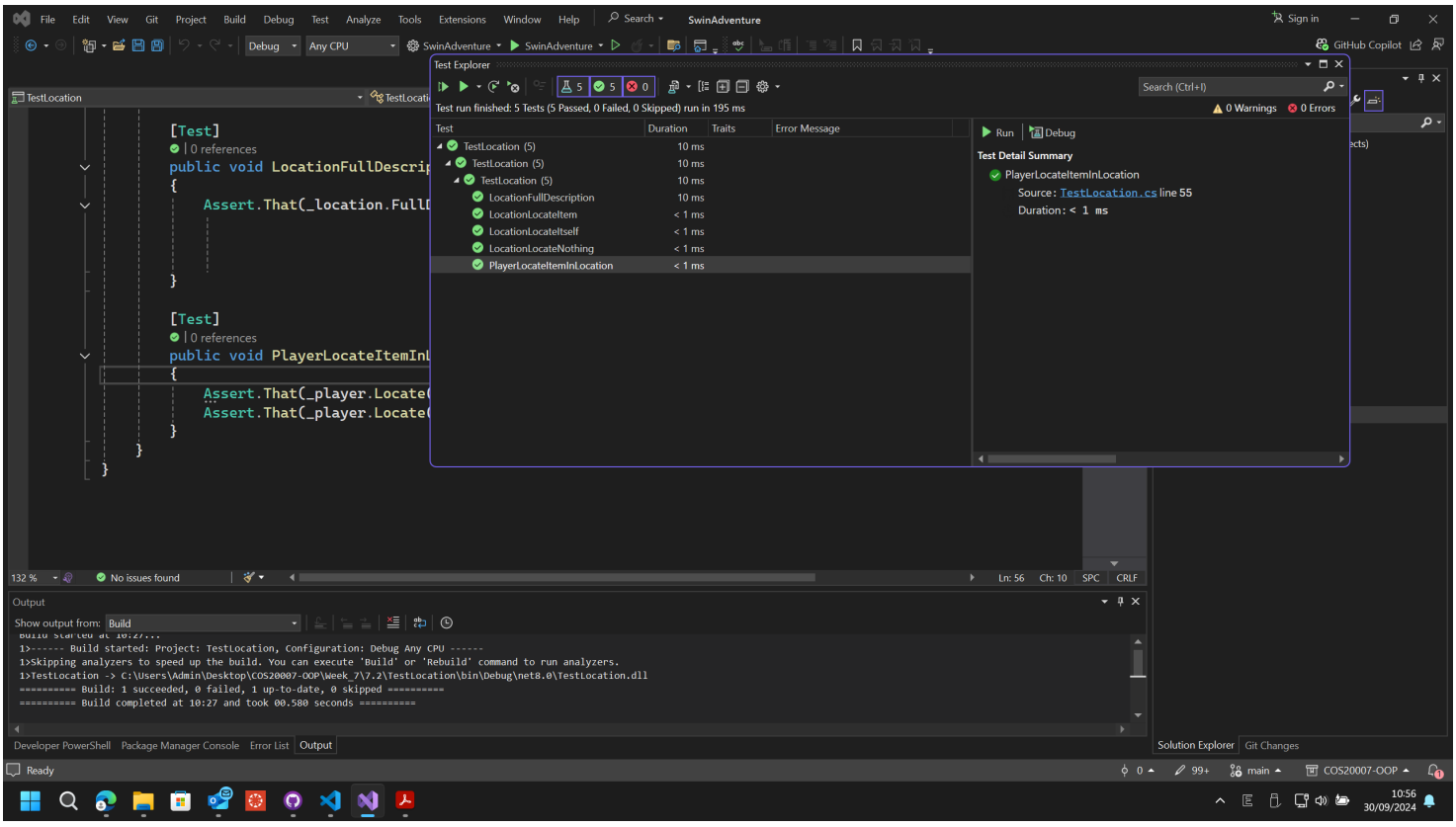
```
"inventory" }, name, desc)
        {
            _inventory = new Inventory();
        }

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

            GameObject? obj = _inventory.Fetch(id);
            if (obj != null)
                return obj;

            if (_location != null)
                return _location.Locate(id);

            return null;
        }

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }
        public Location? Location
        {
            get
            {
                return _location;
            }
            set
            {
                _location = value;
            }
        }
    }
```

Week_7/7.2/TestLocation/TestLocation.cs

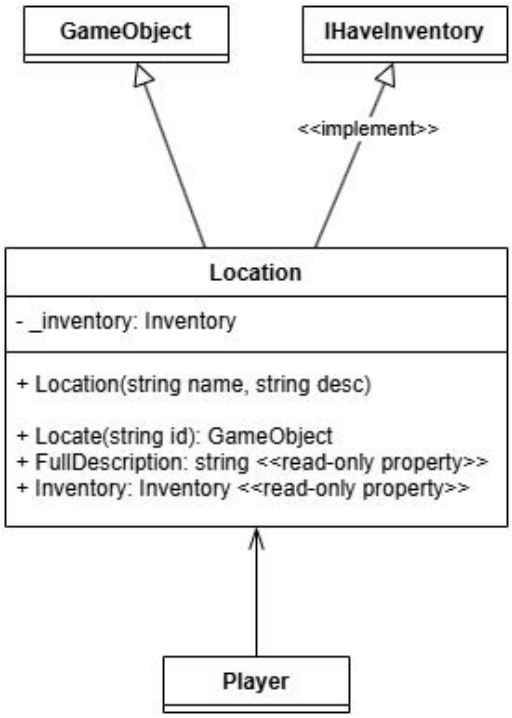
```
using SwinAdventure;
using NUnit.Framework;

namespace TestLocation
{
    public class TestLocation
    {
        private Location _location;
        private Item _item1;
        private Item _item2;
        private Player _player;

        [SetUp]
        public void Setup()
        {
            _player = new Player("Minh An", "104844794");
            _location = new Location("forest", "A dark forest with tall trees");
            _player.Location = _location;

            _item1 = new Item(new string[] { "rock" }, "a rock", "a big rock");
            _item2 = new Item(new string[] { "flower" }, "a flower", "a red flower");
            _location.Inventory.Put(_item1);
            _location.Inventory.Put(_item2);
        }

        [Test]
        public void LocationLocateItself()
        {
            Assert.That(_location.Locate("location"), Is.EqualTo(_location));
        }

        [Test]
        public void LocationLocateItem()
        {
            Assert.That(_location.Locate("rock"), Is.EqualTo(_item1));
            Assert.That(_location.Locate("flower"), Is.EqualTo(_item2));
        }

        [Test]
        public void LocationLocateNothing()
        {
            Assert.That(_location.Locate("nothing"), Is.Null);
        }

        [Test]
        public void LocationFullDescription()
        {
            Assert.That(_location.FullDescription, Is.EqualTo("You are in the
forest.\nA dark forest with tall trees\n" +
                                                    "In this location, you
can see:\n" +
```

```
50                                                          "\ta rock (rock)\n" +
51                                                          "\ta flower
    (flower)"));
52          }
53
54          [Test]
55          public void PlayerLocateItemInLocation()
56          {
57              Assert.That(_player.Locate("rock"), Is.EqualTo(_item1));
58              Assert.That(_player.Locate("flower"), Is.EqualTo(_item2));
59          }
60      }
61  }
```

Screenshot of test



UML diagram



Sequence diagram