Selected files

11 printable files

Week_6\6.1\SwinAdventure\Bag.cs
Week_6\6.1\SwinAdventure\Command.cs
Week_6\6.1\SwinAdventure\GameObject.cs
Week_6\6.1\SwinAdventure\IdentifiableObject.cs
Week_6\6.1\SwinAdventure\IHaveInventory.cs
Week_6\6.1\SwinAdventure\Inventory.cs
Week_6\6.1\SwinAdventure\Item.cs
Week_6\6.1\SwinAdventure\LookCommand.cs
Week_6\6.1\SwinAdventure\Player.cs
Week_6\6.1\SwinAdventure\Program.cs
Week_6\6.1\TestLookCommand\TestLookCommand.cs

Week_6\6.1\SwinAdventure\Bag.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Security.Cryptography;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public class Bag : Item, IHaveInventory
10     {
11         private Inventory _inventory;
12         public Bag(string[] idents, string name, string desc) : base(idents, name, desc)
13         {
14             _inventory = new Inventory();
15         }
16
17         public GameObject? Locate(string id)
18         {
19             if (AreYou(id))
20                 return this;
21
22             if (_inventory.HasItem(id))
23                 return _inventory.Fetch(id);
24
25             return null;
26         }
27
28         public Inventory Inventory => _inventory;
29         public override string FullDescription
30         {
31             get
32             {
33                 return $"In the {Name} you can see:\n{_inventory.ItemList}";
34             }
```

```
        }
    }
}
```

Week_6\6.1\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_6\6.1\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 => _name;

        public string ShortDescription => $"{Name} ({FirstId})";

        public virtual string FullDescription => _description;
    }
}
```

Week_6\6.1\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_6\6.1\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_6\6.1\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_6\6.1\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_6\6.1\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 != 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?";

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

            string containerId = "";
            if (text.Length == 3)
                containerId = p.FirstId;
            else if (text.Length == 5)
                containerId = text[4];

            IHaveInventory? container = FetchContainer(p, containerId);
```

```
            if (container == null)
                return $"I can't find the {containerId}";

            return LookAtIn(text[2], 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_6\6.1\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;

        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;

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

            return null;
        }
```

```
27
28          public override string FullDescription
29          {
30              get
31              {
32                  return $"You are {Name}, {base.FullDescription}\n" +
33                         $"You are carrying:\n{_inventory.ItemList}";
34              }
35          }
36
37          public Inventory Inventory => _inventory;
38      }
39  }
```

Week_6\6.1\SwinAdventure\Program.cs

```
1   namespace SwinAdventure
2   {
3       class Program
4       {
5           static void Main()
6           {
7
8           }
9       }
10  }
```

Week_6\6.1\TestLookCommand\TestLookCommand.cs

```
1   using SwinAdventure;
2
3   namespace TestLookCommand
4   {
5       public class TestLookCommand
6       {
7           private Player _player;
8           private LookCommand _look;
9           private Item _gem;
10          private Bag _bag;
11
12          [SetUp]
13          public void Setup()
14          {
15              _player = new Player("Minh An Nguyen", "Student ID: 104844794");
16              _look = new LookCommand();
17              _gem = new Item(new string[] { "gem" }, "a gem", "a shiny gem");
18              _bag = new Bag(new string[] { "bag" }, "a bag", "a small bag");
19          }
20
21          [Test]
22          public void TestLookAtMe()
23          {
```

```
24              _bag.Inventory.Put(_gem);
25              _player.Inventory.Put(_bag);
26              string expected = "You are Minh An Nguyen, Student ID: 104844794\n" +
27                                "You are carrying:\n" +
28                                "\ta bag (bag)";
29              Assert.That(_look.Execute(_player, new string[] { "look", "at", "inventory" }),
Is.EqualTo(expected));
30          }
31
32          [Test]
33          public void TestLookAtGem()
34          {
35              _player.Inventory.Put(_gem);
36
37              string expected = "a shiny gem";
38              Assert.That(_look.Execute(_player, new string[] { "look", "at", "gem" }),
Is.EqualTo(expected));
39          }
40
41          [Test]
42          public void TestLookAtUnkown()
43          {
44              string expected = "I can't find the gem";
45              Assert.That(_look.Execute(_player, new string[] { "look", "at", "gem" }),
Is.EqualTo(expected));
46          }
47
48          [Test]
49          public void TestLookAtGemInMe()
50          {
51              _player.Inventory.Put(_gem);
52
53              string expected = "a shiny gem";
54              Assert.That(_look.Execute(_player, new string[] { "look", "at", "gem", "in",
"inventory" }), Is.EqualTo(expected));
55          }
56
57          [Test]
58          public void TestLookAtGemInBag()
59          {
60              _bag.Inventory.Put(_gem);
61              _player.Inventory.Put(_bag);
62
63              string expected = "a shiny gem";
64              Assert.That(_look.Execute(_player, new string[] { "look", "at", "gem", "in", "bag"
}), Is.EqualTo(expected));
65          }
66
67          [Test]
68          public void TestLookAtGemInNoBag()
69          {
```

```
70              string expected = "I can't find the bag";
71              Assert.That(_look.Execute(_player, new string[] { "look", "at", "gem", "in", "bag" }), Is.EqualTo(expected));
72          }
73
74          [Test]
75          public void TestLookAtNoGemInBag()
76          {
77              _player.Inventory.Put(_bag);
78
79              string expected = "I can't find the gem";
80              Assert.That(_look.Execute(_player, new string[] { "look", "at", "gem", "in", "bag" }), Is.EqualTo(expected));
81          }
82
83          [Test]
84          public void TestInvalidLook()
85          {
86              Assert.That(_look.Execute(_player, new string[] { "look", "around" }), Is.EqualTo("I don't know how to look like that"));
87              Assert.That(_look.Execute(_player, new string[] { "hello", "104844794" }), Is.EqualTo("I don't know how to look like that"));
88              Assert.That(_look.Execute(_player, new string[] { "hello", "Minh", "An" }), Is.EqualTo("Error in look input"));
89              Assert.That(_look.Execute(_player, new string[] { "look", "this", "bag" }), Is.EqualTo("What do you want to look at?"));
90              Assert.That(_look.Execute(_player, new string[] { "look", "at", "bag", "inside", "inventory" }), Is.EqualTo("What do you want to look in?"));
91          }
92      }
93  }
```

Screenshot of running test: