Selected files

10 printable files

Week_7\7.1\SwinAdventure\Program.cs
Week_7\7.1\SwinAdventure\Bag.cs
Week_7\7.1\SwinAdventure\Command.cs
Week_7\7.1\SwinAdventure\GameObject.cs
Week_7\7.1\SwinAdventure\IdentifiableObject.cs
Week_7\7.1\SwinAdventure\IHaveInventory.cs
Week_7\7.1\SwinAdventure\Inventory.cs
Week_7\7.1\SwinAdventure\Item.cs
Week_7\7.1\SwinAdventure\LookCommand.cs
Week_7\7.1\SwinAdventure\Player.cs

Week_7\7.1\SwinAdventure\Program.cs

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

            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);

            Bag bag = new Bag(new string[] { "bag" }, "a bag", "a leather bag");
            player.Inventory.Put(bag);

            Item item3 = new Item(new string[] { "coin" }, "a coin", "a shiny coin");
            bag.Inventory.Put(item3);
            LookCommand look = new LookCommand();
```

```
36
37              while (true)
38              {
39                  Console.WriteLine(player.FullDescription);
40                  Console.Write("> ");
41                  string command = Console.ReadLine() ?? string.Empty;
42
43                  if (string.IsNullOrEmpty(command))
44                      continue;
45                  if (command == "quit")
46                      break;
47
48                  string response = look.Execute(player, command.Split(" "));
49                  Console.WriteLine(response);
50                  Console.WriteLine();
51              }
52          }
53      }
54  }
```

Week_7\7.1\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
```

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

Week_7\7.1\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_7\7.1\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_7\7.1\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_7\7.1\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_7\7.1\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_7\7.1\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_7\7.1\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_7\7.1\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;
        }

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

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

Screenshot of running program: