Selected files

9 printable files

```
Week_4/4.2/SwinAdventure/GameObject.cs
Week_4/4.2/SwinAdventure/IdentifiableObject.cs
Week_4/4.2/SwinAdventure/Inventory.cs
Week_4/4.2/SwinAdventure/Item.cs
Week_4/4.2/SwinAdventure/Player.cs
Week_4/4.2/SwinAdventure/Program.cs
Week_4/4.2/TestInventory/TestInventory.cs
Week_4/4.2/TestItem/TestItem.cs
Week_4/4.2/TestPlayer/TestPlayer.cs
```

Week_4/4.2/SwinAdventure/GameObject.cs

```csharp
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_4/4.2/SwinAdventure/IdentifiableObject.cs

```csharp
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("104844794");
            }
            else
            {
                _identifiers[0] = "104844794";
            }
        }
    }
}
```

Week_4/4.2/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_4/4.2/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_4/4.2/SwinAdventure/Player.cs

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

namespace SwinAdventure
{
    public class Player : GameObject
    {
        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;
    }
}
```

Week_4/4.2/SwinAdventure/Program.cs

```
namespace SwinAdventure
{
    class Program
    {
        static void Main()
        {
            string expected = "\ta sharp sword (sword)\n" +
                              "\ta blunt axe (axe)";
            System.Console.WriteLine(expected);
        }
    }
}
```

Week_4/4.2/TestInventory/TestInventory.cs

```
using NuGet.Frameworks;
using SwinAdventure;

namespace TestInventory
```

```csharp
{
    public class Tests
    {
        private Inventory _inventory;

        [SetUp]
        public void Setup()
        {
            _inventory = new Inventory();
            _inventory.Put(new Item(new string[] { "sword" }, "a sharp sword", "A sharp sword for cutting things"));
            _inventory.Put(new Item(new string[] { "axe" }, "a blunt axe", "A blunt axe for bludgeoning things"));
        }

        [Test]
        public void TestFindItem()
        {
            Assert.IsTrue(_inventory.HasItem("sword"));
            Assert.IsTrue(_inventory.HasItem("axe"));
        }

        [Test]
        public void TestNoItemFind()
        {
            Assert.IsFalse(_inventory.HasItem("dagger"));
        }

        [Test]
        public void TestFetchItem()
        {
            Item item = _inventory.Fetch("sword")!;
            Assert.IsTrue(item.AreYou("sword"));
            Assert.IsTrue(_inventory.HasItem("sword"));
        }

        [Test]
        public void TestTakeItem()
        {
            Item item = _inventory.Take("axe")!;
            Assert.IsTrue(item.AreYou("axe"));
            Assert.IsFalse(_inventory.HasItem("axe"));
        }

        [Test]
        public void TestItemList()
        {
            string expected = "\ta sharp sword (sword)\n\ta blunt axe (axe)";
            Assert.AreEqual(expected, _inventory.ItemList);
        }
    }
}
```

Week_4/4.2/TestItem/TestItem.cs

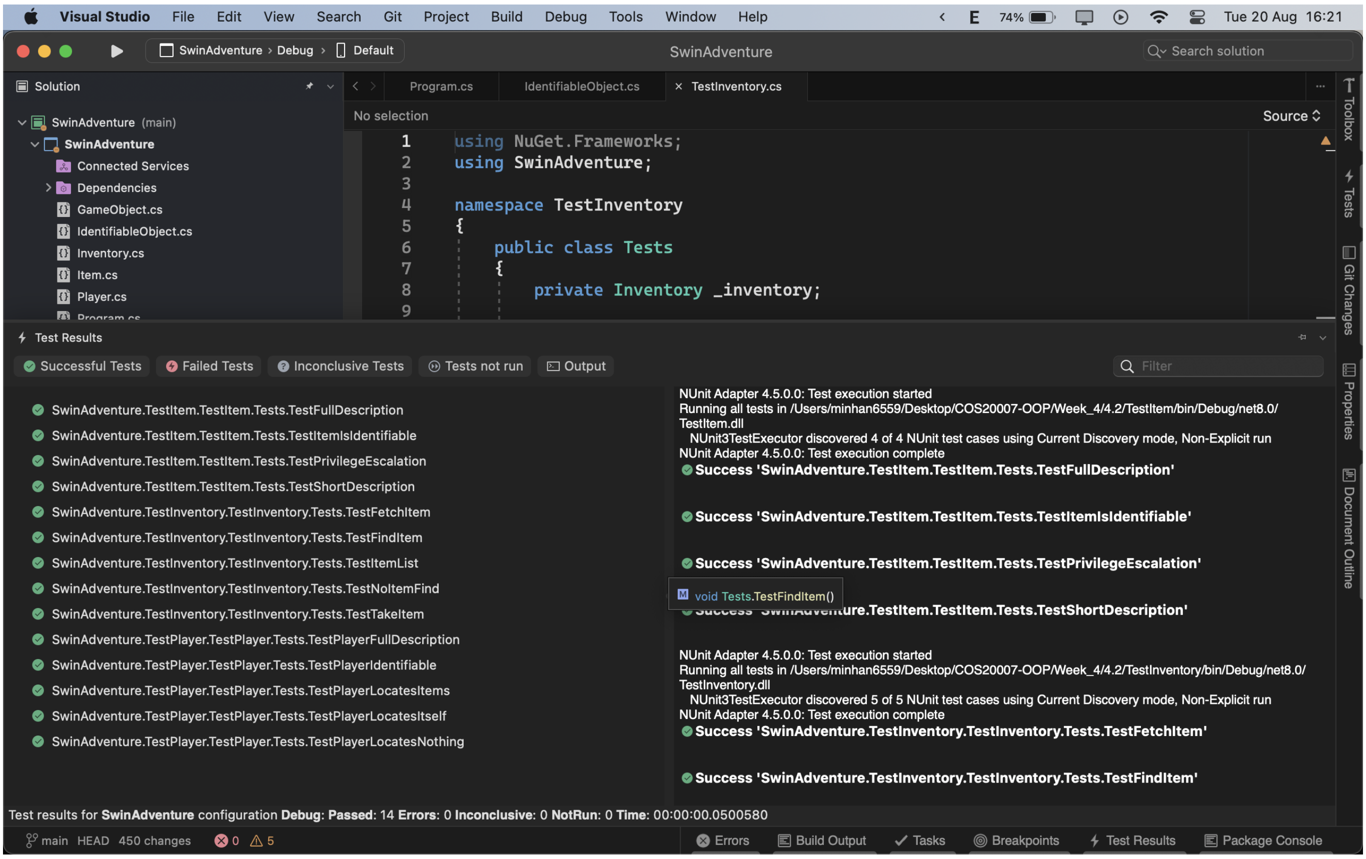
```
using SwinAdventure;

namespace TestItem
{
    public class Tests
    {
        private Item _item;
        [SetUp]
        public void Setup()
        {
            _item = new Item(new string[] { "sword" }, "a sharp sword", "A sharp sword for cutting things");
        }

        [Test]
        public void TestItemIsIdentifiable()
        {
            Assert.IsTrue(_item.AreYou("sword"));
            Assert.IsFalse(_item.AreYou("axe"));
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.AreEqual("a sharp sword (sword)", _item.ShortDescription);
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.AreEqual("A sharp sword for cutting things", _item.FullDescription);
        }

        [Test]
        public void TestPrivilegeEscalation()
        {
            _item.PrivilegeEscalation("4794");
            Assert.AreEqual("104844794", _item.FirstId);
        }
    }
}
```

Week_4/4.2/TestPlayer/TestPlayer.cs

```
using SwinAdventure;

namespace TestPlayer
{
    public class Tests
    {
        private Player _player;

        [SetUp]
        public void Setup()
        {
            _player = new Player("Minh An", "an AI enthusiast");
            _player.Inventory.Put(new Item(new string[] { "sword" }, "a sharp sword", "A sharp sword for cutting things"));
            _player.Inventory.Put(new Item(new string[] { "axe" }, "a blunt axe", "A blunt axe for bludgeoning things"));
        }

        [Test]
        public void TestPlayerIdentifiable()
        {
            Assert.IsTrue(_player.AreYou("me"));
            Assert.IsTrue(_player.AreYou("inventory"));
        }

        [Test]
        public void TestPlayerLocatesItems()
        {
            Assert.IsTrue(_player.Locate("sword")!.AreYou("sword"));
            Assert.IsTrue(_player.Locate("axe")!.AreYou("axe"));
        }

        [Test]
        public void TestPlayerLocatesItself()
        {
            Assert.IsTrue(_player.Locate("me")!.AreYou("me"));
            Assert.IsTrue(_player.Locate("inventory")!.AreYou("inventory"));
        }

        [Test]
        public void TestPlayerLocatesNothing()
        {
            Assert.IsNull(_player.Locate("dagger"));
        }

        [Test]
        public void TestPlayerFullDescription()
        {
            string expected = "You are Minh An, an AI enthusiast\n" +
                                "You are carrying:\n" +
                                "\ta sharp sword (sword)\n" +
                                "\ta blunt axe (axe)";
            Assert.AreEqual(expected, _player.FullDescription);
        }
    }
```

```
}
```

Screenshot of test output:



```
using NuGet.Frameworks;
using SwinAdventure;

namespace TestInventory
{
    public class Tests
    {
        private Inventory _inventory;
```

```
SwinAdventure.TestItem.TestItem.Tests.TestFullDescription
SwinAdventure.TestItem.TestItem.Tests.TestItemIsIdentifiable
SwinAdventure.TestItem.TestItem.Tests.TestPrivilegeEscalation
SwinAdventure.TestItem.TestItem.Tests.TestShortDescription
SwinAdventure.TestInventory.TestInventory.Tests.TestFetchItem
SwinAdventure.TestInventory.TestInventory.Tests.TestFindItem
SwinAdventure.TestInventory.TestInventory.Tests.TestItemList
SwinAdventure.TestInventory.TestInventory.Tests.TestNoItemFind
SwinAdventure.TestInventory.TestInventory.Tests.TestTakeItem
SwinAdventure.TestPlayer.TestPlayer.Tests.TestPlayerFullDescription
SwinAdventure.TestPlayer.TestPlayer.Tests.TestPlayerIdentifiable
SwinAdventure.TestPlayer.TestPlayer.Tests.TestPlayerLocatesItems
SwinAdventure.TestPlayer.TestPlayer.Tests.TestPlayerLocatesItself
SwinAdventure.TestPlayer.TestPlayer.Tests.TestPlayerLocatesNothing
```

```
NUnit Adapter 4.5.0.0: Test execution started
Running all tests in /Users/minhan6559/Desktop/COS20007-OOP/Week_4/4.2/TestItem/bin/Debug/net8.0/
TestItem.dll
  NUnit3TestExecutor discovered 4 of 4 NUnit test cases using Current Discovery mode, Non-Explicit run
NUnit Adapter 4.5.0.0: Test execution complete
Success 'SwinAdventure.TestItem.TestItem.Tests.TestFullDescription'

Success 'SwinAdventure.TestItem.TestItem.Tests.TestItemIsIdentifiable'

Success 'SwinAdventure.TestItem.TestItem.Tests.TestPrivilegeEscalation'

Success 'SwinAdventure.TestItem.TestItem.Tests.TestShortDescription'

NUnit Adapter 4.5.0.0: Test execution started
Running all tests in /Users/minhan6559/Desktop/COS20007-OOP/Week_4/4.2/TestInventory/bin/Debug/net8.0/
TestInventory.dll
  NUnit3TestExecutor discovered 5 of 5 NUnit test cases using Current Discovery mode, Non-Explicit run
NUnit Adapter 4.5.0.0: Test execution complete
Success 'SwinAdventure.TestInventory.TestInventory.Tests.TestFetchItem'

Success 'SwinAdventure.TestInventory.TestInventory.Tests.TestFindItem'
```

Test results for **SwinAdventure** configuration **Debug**: **Passed**: 14 **Errors**: 0 **Inconclusive**: 0 **NotRun**: 0 **Time**: 00:00:00.0500580