Selected files

3 printable files

Week_10/10.1/SwinAdventure/CommandProcessor.cs
Week_10/10.1/SwinAdventure/Program.cs
Week_10/10.1/TestCommandProcessor/TestCommandProcessor.cs

Week_10/10.1/SwinAdventure/CommandProcessor.cs

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

namespace SwinAdventure
{
    public class CommandProcessor : IdentifiableObject
    {
        private List<Command> _commands;

        public CommandProcessor() : base(new string[] { "commands" })
        {
            _commands = new List<Command>();
            _commands.Add(new LookCommand());
            _commands.Add(new MoveCommand());
        }

        public string Execute(Player p, string[] text)
        {
            foreach (Command c in _commands)
            {
                if (c.AreYou(text[0]))
                {
                    return c.Execute(p, text);
                }
            }
            return "I don't know how to do that";
        }

    }
}
```

Week_10/10.1/SwinAdventure/Program.cs

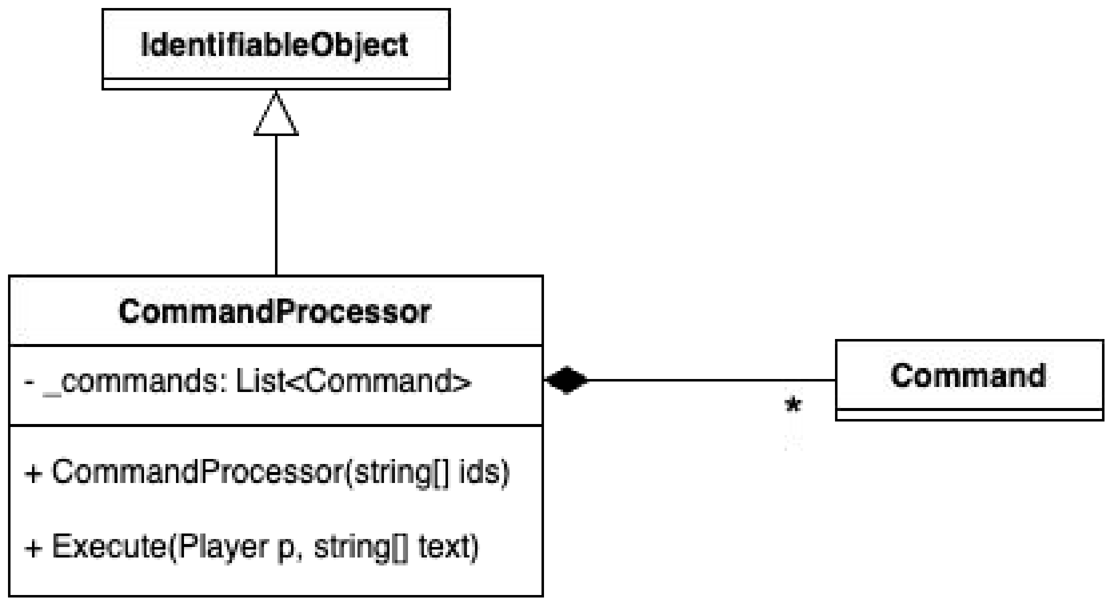
```
namespace SwinAdventure
{
    class Program
    {
        static void Main()
        {
            string? playerName, playerDesc;
            while (true)
            {
                Console.Write("Enter player name: ");
```

```csharp
                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);

            // Create another location and a path between the two locations
            Location location2 = new Location("cave", "A dark cave with bats");
            Path path = new Path(new string[] { "north" }, "north", "a path from forest to cave", location, location2);
            Path path2 = new Path(new string[] { "south" }, "south", "a path from cave to forest", location2, location);
            location.AddPath(path);
            location2.AddPath(path2);

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

            CommandProcessor command = new CommandProcessor();

            while (true)
            {
                Console.Write("> ");
                string? input = Console.ReadLine();

                if (string.IsNullOrEmpty(input))
                    continue;

                if (input == "quit")
                    break;

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

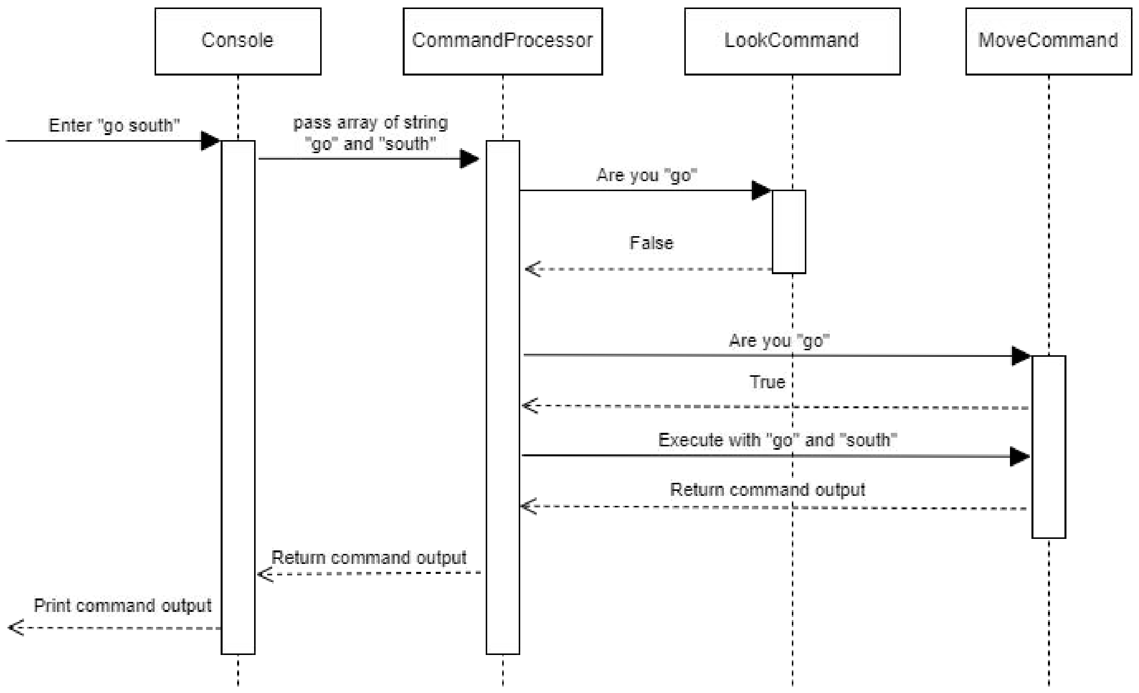
Week_10/10.1/TestCommandProcessor/TestCommandProcessor.cs

```
using SwinAdventure;
using Path = SwinAdventure.Path;

namespace TestCommandProcessor
{
    public class TestCommandProcessor
    {
        private CommandProcessor _cmdProcessor;
        private Player _player;
        private Location _location1;
        private Location _location2;
        private Path _path;
        private Item _sword;
        private Bag _bag;


        [SetUp]
        public void Setup()
        {
            _cmdProcessor = new CommandProcessor();
            _player = new Player("Minh An", "104844794");
            _location1 = new Location("forest", "A dark forest with tall trees");
            _location2 = new Location("cave", "A dark cave with bats");
            _path = new Path(new string[] { "north" }, "north", "a path from forest to cave", _location1, _location2);
            _location1.AddPath(_path);
            _player.Location = _location1;
            _sword = new Item(new string[] { "sword" }, "a sword", "a steel sword");
            _bag = new Bag(new string[] { "bag" }, "a bag", "a leather bag");
            _player.Inventory.Put(_sword);
            _player.Inventory.Put(_bag);
        }

        [Test]
        public void TestLookAtMe()
        {
            string expected = "You are Minh An, 104844794\n" +
                              "You are carrying:\n" +
                              "\ta sword (sword)\n\ta bag (bag)";
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "inventory" }), Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtSword()
        {
            string expected = "a steel sword";
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "sword" }), Is.EqualTo(expected));
        }

        [Test]
        public void TestLookAtUnkown()
        {
            string expected = "I can't find the gem";
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "gem" }), Is.EqualTo(expected));
        }

        [Test]
        public void TestMoveToNonExistentPath()
        {
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "south" }), Is.EqualTo("I can't find the path to south"));
        }

        [Test]
        public void TestMoveToDestination()
        {
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north" }), Is.EqualTo("You have moved to cave"));
        }

        [Test]
        public void TestInvalidMoveCommand()
        {
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "move", "north" }), Is.EqualTo("You have moved to cave"));
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north", "to" }), Is.EqualTo("Where do you want to go?"));
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north", "to", "cave" }), Is.EqualTo("I don't know how to move like that"));
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north", "to", "cave", "now" }), Is.EqualTo("I don't know how to move like that"));
        }

        [Test]
        public void TestInvalidLook()
        {
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "around" }), Is.EqualTo("I don't know how to look like that"));
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "this", "bag" }), Is.EqualTo("What do you want to look at?"));
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "bag", "inside", "inventory" }), Is.EqualTo("What do you want to look in?"));
        }

        [Test]
        public void TestInvalidCommand()
        {
            Assert.That(_cmdProcessor.Execute(_player, new string[] { "hi", "hey" }), Is.EqualTo("I don't know how to do that"));
        }
    }
}
```

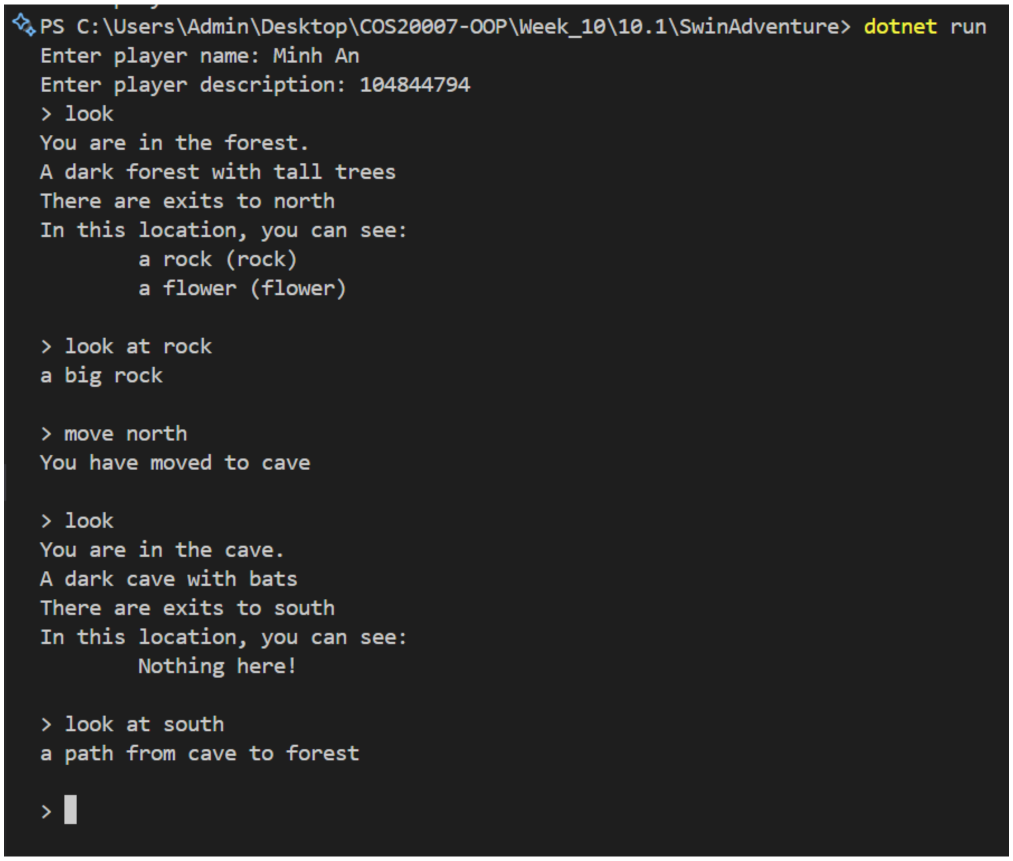
UML diagram:


IdentifiableObject
CommandProcessor
- _commands: List<Command>
+ CommandProcessor(string[] ids)
+ Execute(Player p, string[] text)
*
Command

Sequence diagram:

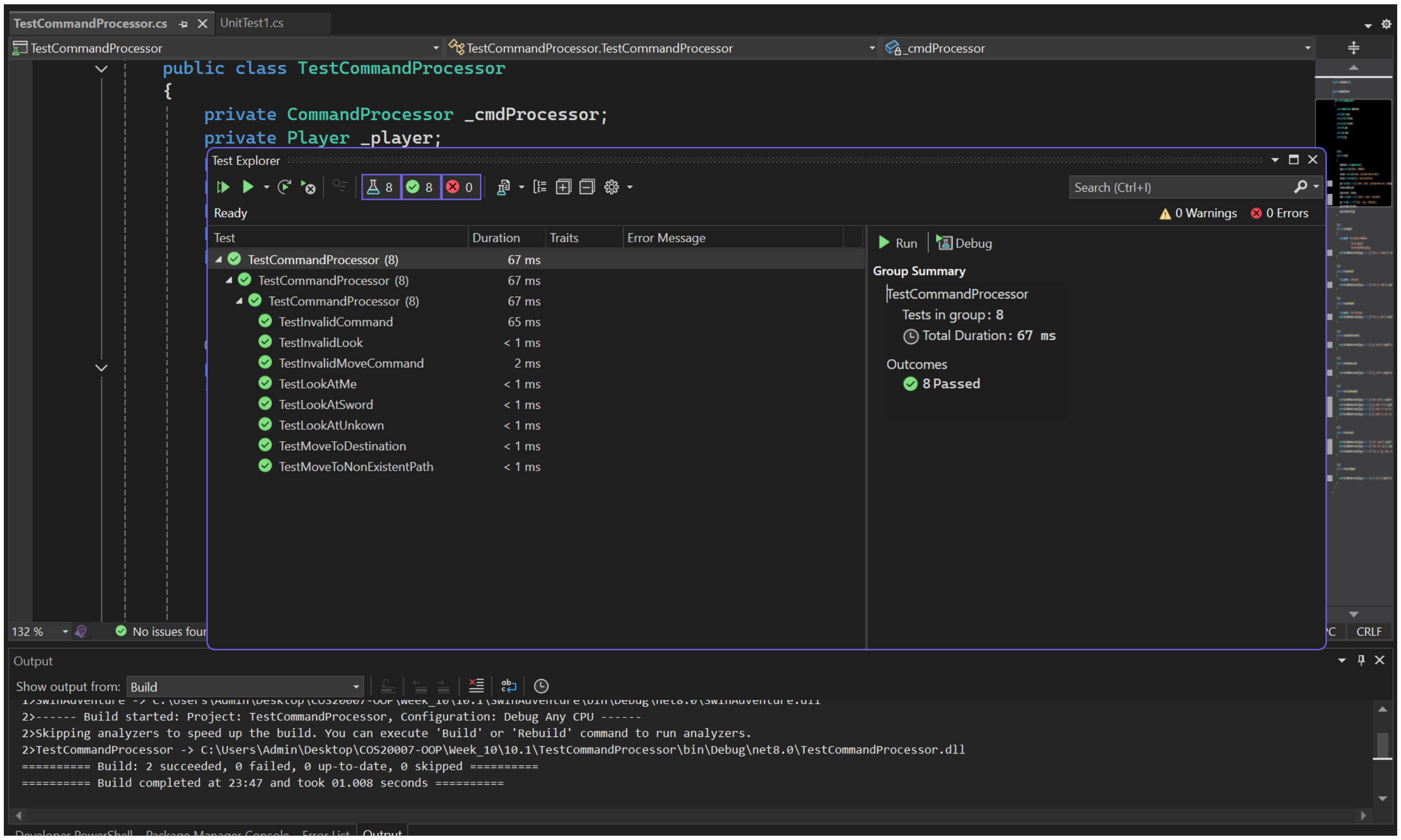


Screenshot of program running:



```
PS C:\Users\Admin\Desktop\COS20007-OOP\Week_10\10.1\SwinAdventure> dotnet run
 Enter player name: Minh An
 Enter player description: 104844794
 > look
 You are in the forest.
 A dark forest with tall trees
 There are exits to north
 In this location, you can see:
         a rock (rock)
         a flower (flower)

 > look at rock
 a big rock

 > move north
 You have moved to cave

 > look
 You are in the cave.
 A dark cave with bats
 There are exits to south
 In this location, you can see:
         Nothing here!

 > look at south
 a path from cave to forest

 >
```

Screenshot of unit test passing:



```
public class TestCommandProcessor
{
    private CommandProcessor _cmdProcessor;
    private Player _player;
```

Test Explorer

Ready — 8 tests, 8 passed, 0 failed — 0 Warnings, 0 Errors

Test	Duration	Traits	Error Message
TestCommandProcessor (8)	67 ms		
TestCommandProcessor (8)	67 ms		
TestCommandProcessor (8)	67 ms		
TestInvalidCommand	65 ms		
TestInvalidLook	< 1 ms		
TestInvalidMoveCommand	2 ms		
TestLookAtMe	< 1 ms		
TestLookAtSword	< 1 ms		
TestLookAtUnkown	< 1 ms		
TestMoveToDestination	< 1 ms		
TestMoveToNonExistentPath	< 1 ms		

Group Summary

TestCommandProcessor
Tests in group: 8
Total Duration: 67 ms

Outcomes
8 Passed

Output — Show output from: Build

```
1>SwinAdventure -> C:\Users\Admin\Desktop\COS20007-OOP\Week_10\10.1\SwinAdventure\bin\Debug\net8.0\SwinAdventure.dll
2>------ Build started: Project: TestCommandProcessor, Configuration: Debug Any CPU ------
2>Skipping analyzers to speed up the build. You can execute 'Build' or 'Rebuild' command to run analyzers.
2>TestCommandProcessor -> C:\Users\Admin\Desktop\COS20007-OOP\Week_10\10.1\TestCommandProcessor\bin\Debug\net8.0\TestCommandProcessor.dll
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 23:47 and took 01.008 seconds ==========
```