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

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace SwinAdventure
7  {
8      public class CommandProcessor : Command
9      {
10         private List<Command> _commands;
11
12         public CommandProcessor() : base(new string[] { "commands" })
13         {
14             _commands = new List<Command>();
15             _commands.Add(new LookCommand());
16             _commands.Add(new MoveCommand());
17         }
18
19         public override string Execute(Player p, string[] text)
20         {
21             foreach (Command c in _commands)
22             {
23                 if (c.AreYou(text[0]))
24                 {
25                     return c.Execute(p, text);
26                 }
27             }
28             return "I don't know how to do that";
29         }
30
31     }
32 }
```

Week_10\10.1\SwinAdventure\Program.cs

```
1  namespace SwinAdventure
2  {
3      class Program
4      {
5          static void Main()
6          {
```

```
            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);

            // 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();
            }
        }
    }
}
```

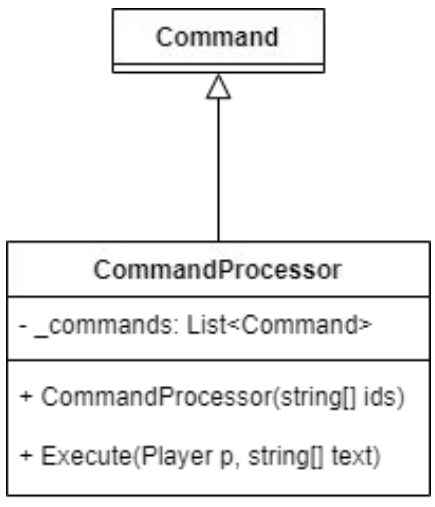
Week_10\10.1\TestCommandProcessor\TestCommandProcessor.cs

```
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()
```

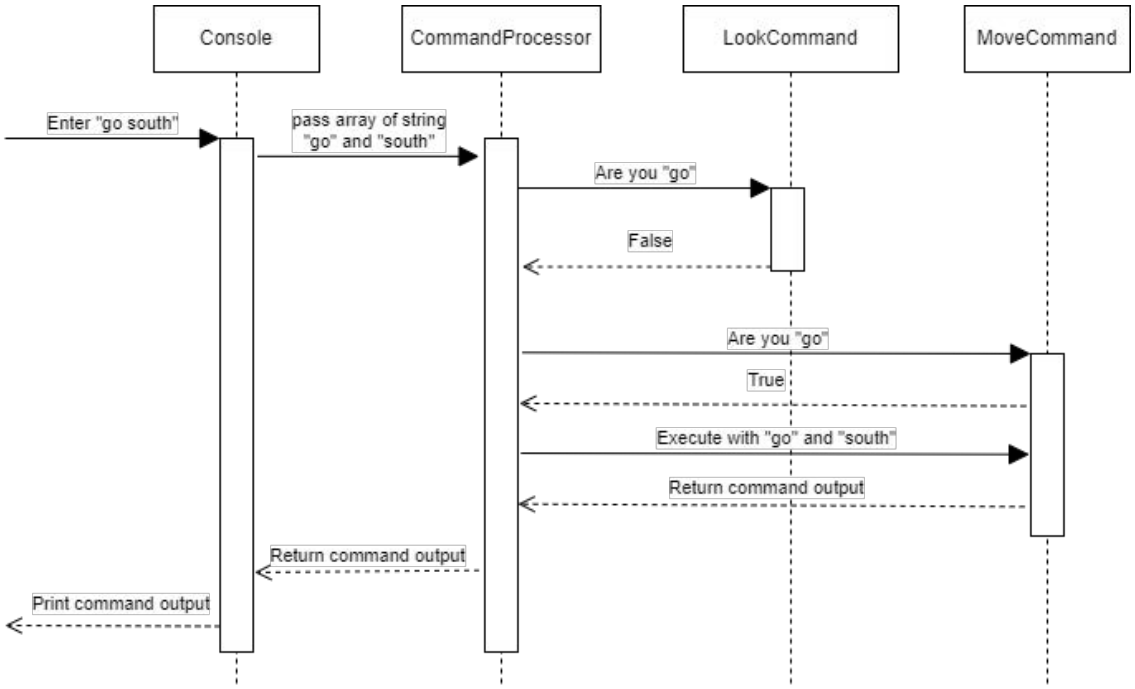
```
        {
            _cmdProcessor = new CommandProcessor();
            _player = new Player("Minh An", "104844794");
            _location1 = new Location("forest", "A dark forest with tall trees");
            _location2 = new Location("cave", "A dark cave with bats");
            _path = new 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()
        {
```

```
65              Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north" }),
Is.EqualTo("You have moved to cave"));
66          }
67
68          [Test]
69          public void TestInvalidMoveCommand()
70          {
71              Assert.That(_cmdProcessor.Execute(_player, new string[] { "move", "north" }),
Is.EqualTo("You have moved to cave"));
72              Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north", "to" }),
Is.EqualTo("Where do you want to go?"));
73              Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north", "to",
"cave" }), Is.EqualTo("I don't know how to move like that"));
74              Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north", "to",
"cave", "now" }), Is.EqualTo("I don't know how to move like that"));
75          }
76
77          [Test]
78          public void TestInvalidLook()
79          {
80              Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "around" }),
Is.EqualTo("I don't know how to look like that"));
81              Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "this", "bag" }),
Is.EqualTo("What do you want to look at?"));
82              Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "bag",
"inside", "inventory" }), Is.EqualTo("What do you want to look in?"));
83          }
84
85          [Test]
86          public void TestInvalidCommand()
87          {
88              Assert.That(_cmdProcessor.Execute(_player, new string[] { "hi", "hey" }),
Is.EqualTo("I don't know how to do that"));
89          }
90      }
91  }
```

UML diagram:



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: