

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          {
```

```
7     string? playerName, playerDesc;
8     while (true)
9     {
10         Console.Write("Enter player name: ");
11         playerName = Console.ReadLine();
12         if (playerName == null)
13         {
14             playerName = string.Empty;
15         }
16
17         Console.Write("Enter player description: ");
18         playerDesc = Console.ReadLine();
19         if (playerDesc == null)
20         {
21             playerDesc = string.Empty;
22         }
23         if (string.IsNullOrEmpty(playerName) || string.IsNullOrEmpty(playerDesc))
24         {
25             Console.WriteLine("Player name and description cannot be empty.");
26         }
27         else
28         {
29             break;
30         }
31     }
32     Player player = new Player(playerName, playerDesc);
33
34     // Create items and put them in the player's inventory
35     Item item1 = new Item(new string[] { "shovel" }, "a shovel", "a wooden shovel");
36     Item item2 = new Item(new string[] { "sword" }, "a sword", "a steel sword");
37     player.Inventory.Put(item1);
38     player.Inventory.Put(item2);
39
40     // Create a bag and put it in the player's inventory
41     Bag bag = new Bag(new string[] { "bag" }, "a bag", "a leather bag");
42     player.Inventory.Put(bag);
43
44     // Create items and put them in the bag's inventory
45     Item item3 = new Item(new string[] { "coin" }, "a coin", "a shiny coin");
46     bag.Inventory.Put(item3);
47
48     // Create location and put some items in its inventory
49     Location location = new Location("forest", "A dark forest with tall trees");
50     Item item4 = new Item(new string[] { "rock" }, "a rock", "a big rock");
51     Item item5 = new Item(new string[] { "flower" }, "a flower", "a red flower");
52     location.Inventory.Put(item4);
53     location.Inventory.Put(item5);
54
55     // Create another location and a path between the two locations
56     Location location2 = new Location("cave", "A dark cave with bats");
```

```

57         Path path = new Path(new string[] { "north" }, "north", "a path from forest to
cave", location, location2);
58         Path path2 = new Path(new string[] { "south" }, "south", "a path from cave to
forest", location2, location);
59         location.AddPath(path);
60         location2.AddPath(path2);
61
62         // Set player's location
63         player.Location = location;
64
65         CommandProcessor command = new CommandProcessor();
66
67         while (true)
68         {
69             Console.Write("> ");
70             string? input = Console.ReadLine();
71
72             if (string.IsNullOrEmpty(input))
73                 continue;
74
75             if (input == "quit")
76                 break;
77
78             string response = command.Execute(player, input.Split(" "));
79             Console.WriteLine(response);
80             Console.WriteLine();
81         }
82     }
83 }
84 }

```

Week_10\10.1\TestCommandProcessor\TestCommandProcessor.cs

```

1  using SwinAdventure;
2  using Path = SwinAdventure.Path;
3
4  namespace TestCommandProcessor
5  {
6      public class TestCommandProcessor
7      {
8          private CommandProcessor _cmdProcessor;
9          private Player _player;
10         private Location _location1;
11         private Location _location2;
12         private Path _path;
13         private Item _sword;
14         private Bag _bag;
15
16
17         [SetUp]
18         public void Setup()

```

```

19     {
20         _cmdProcessor = new CommandProcessor();
21         _player = new Player("Minh An", "104844794");
22         _location1 = new Location("forest", "A dark forest with tall trees");
23         _location2 = new Location("cave", "A dark cave with bats");
24         _path = new Path(new string[] { "north" }, "north", "a path from forest to cave",
_location1, _location2);
25         _location1.AddPath(_path);
26         _player.Location = _location1;
27         _sword = new Item(new string[] { "sword" }, "a sword", "a steel sword");
28         _bag = new Bag(new string[] { "bag" }, "a bag", "a leather bag");
29         _player.Inventory.Put(_sword);
30         _player.Inventory.Put(_bag);
31     }
32
33     [Test]
34     public void TestLookAtMe()
35     {
36         string expected = "You are Minh An, 104844794\n" +
37             "You are carrying:\n" +
38             "\ta sword (sword)\n\ta bag (bag)";
39         Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "inventory"
}), Is.EqualTo(expected));
40     }
41
42     [Test]
43     public void TestLookAtSword()
44     {
45         string expected = "a steel sword";
46         Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "sword" }),
Is.EqualTo(expected));
47     }
48
49     [Test]
50     public void TestLookAtUnkown()
51     {
52         string expected = "I can't find the gem";
53         Assert.That(_cmdProcessor.Execute(_player, new string[] { "look", "at", "gem" }),
Is.EqualTo(expected));
54     }
55
56     [Test]
57     public void TestMoveToNonExistentPath()
58     {
59         Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "south" }),
Is.EqualTo("I can't find the path to south"));
60     }
61
62     [Test]
63     public void TestMoveToDestination()
64     {

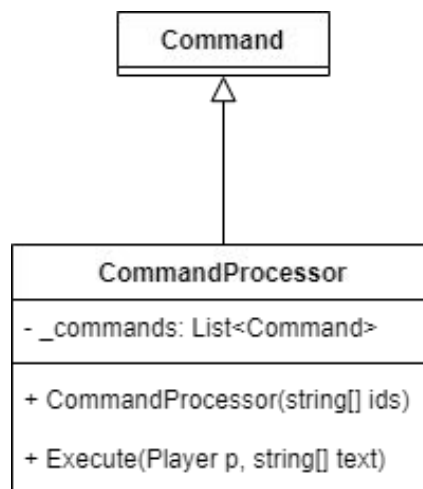
```

```

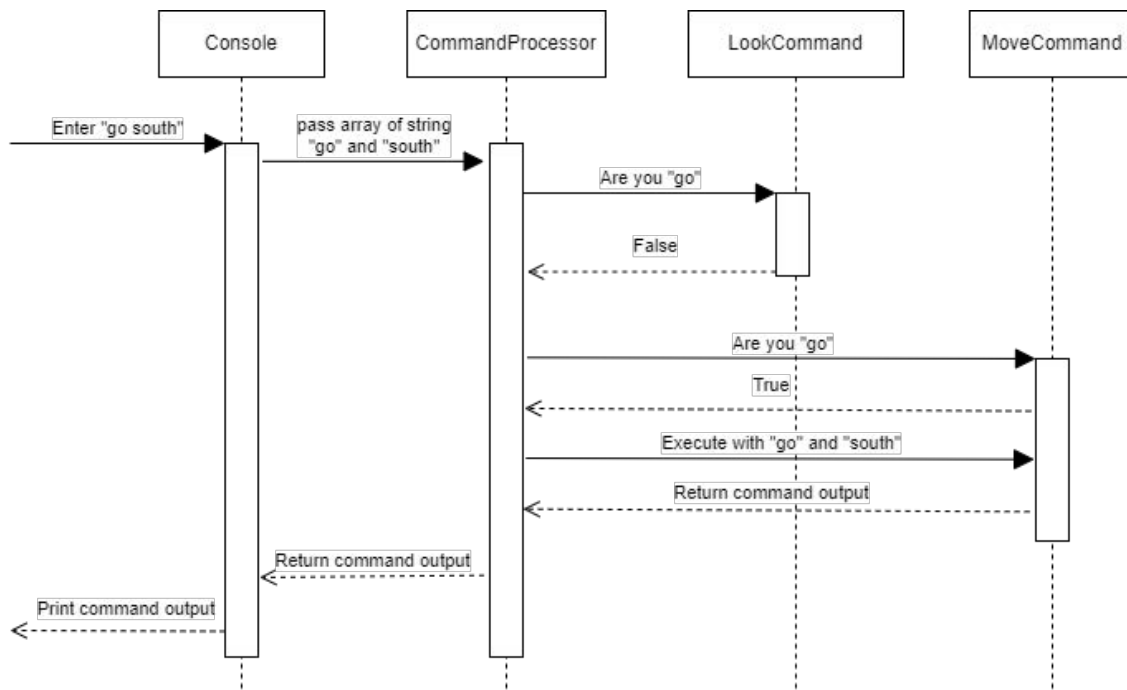
65         Assert.That(_cmdProcessor.Execute(_player, new string[] { "go", "north" }),
Is.EqualTo("You have moved to cave"));
66     }
67
68     [Test]
69     public void TestInvalidMoveCommand()
70     {
71         Assert.That(_cmdProcessor.Execute(_player, new string[] { "move", "north" }),
Is.EqualTo("You 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:

The screenshot displays the Visual Studio IDE with the `TestCommandProcessor.cs` file open. The `Test Explorer` window is active, showing a list of tests that have all passed. The `Group Summary` on the right indicates that 8 tests passed with a total duration of 67 ms. The `Output` window at the bottom shows the build process, confirming that the build succeeded.

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

    // ... (test methods) ...
}
```

Test	Duration	Traits	Error Message
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
Build started: Project: TestCommandProcessor, Configuration: Debug Any CPU
Skipping analyzers to speed up the build. You can execute 'Build' or 'Rebuild' command to run analyzers.
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