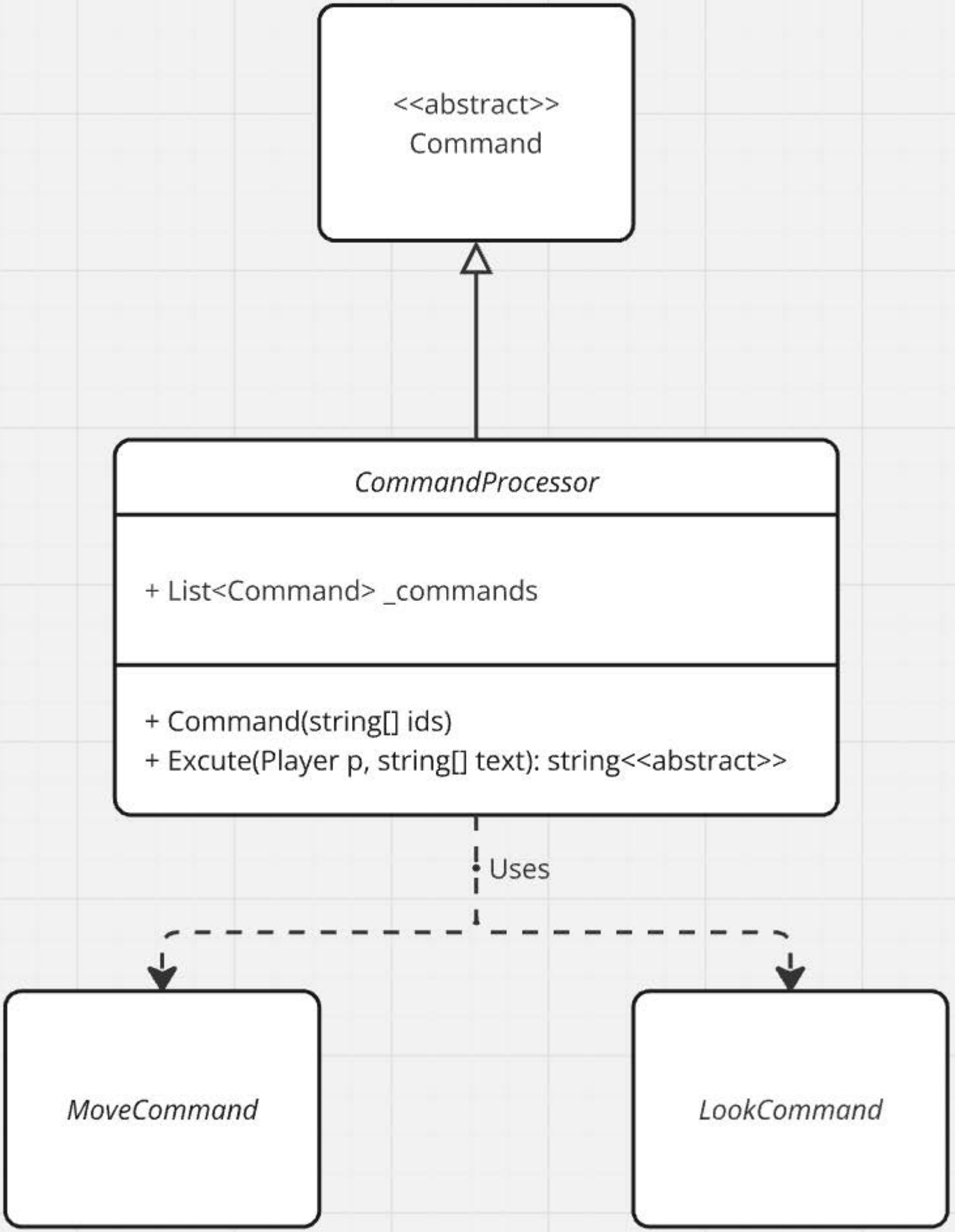
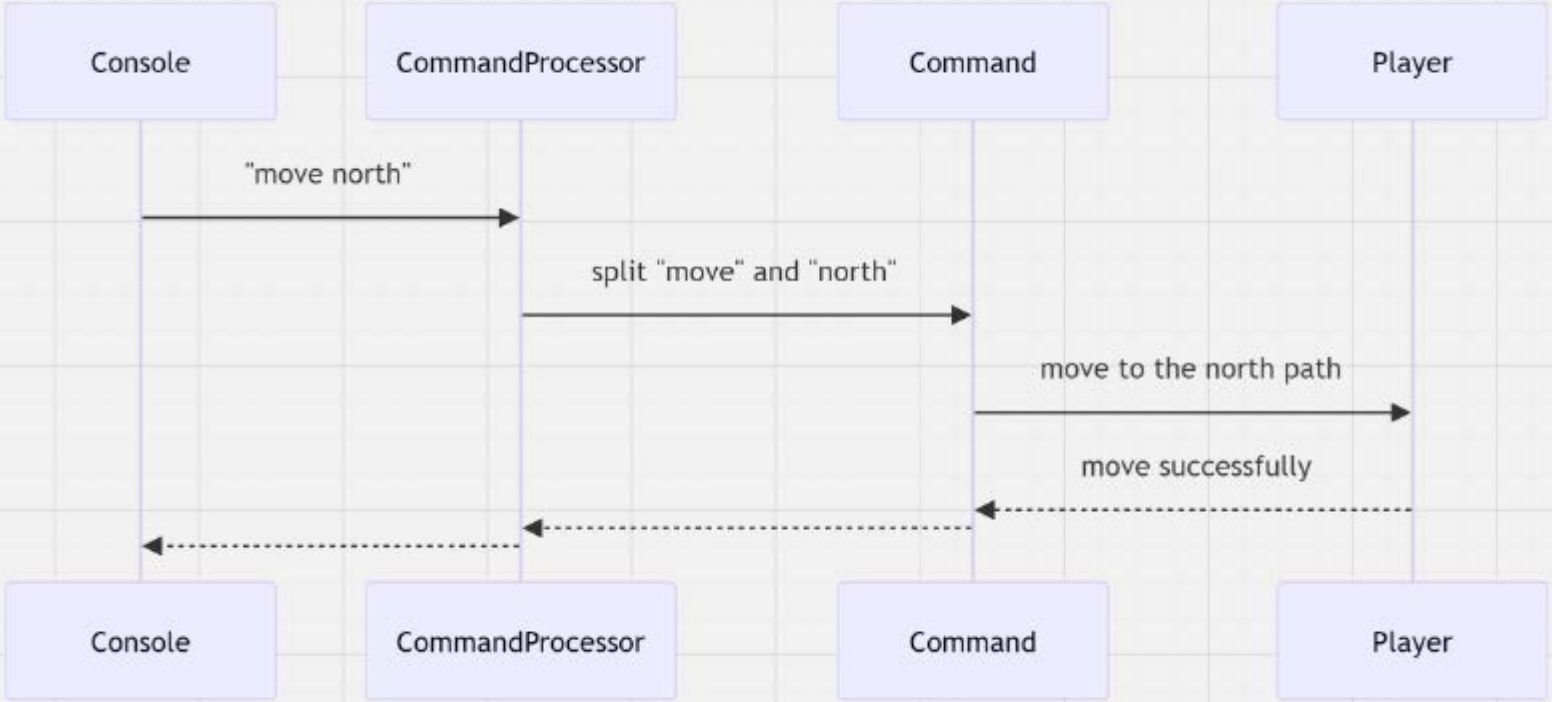

<<abstract>>
Command
CommandProcessor
+ List<Command> _commands
+ Command(string[] ids)
+ Excute(Player p, string[] text): string<<abstract>>
Uses
MoveCommand
LookCommand


Console
CommandProcessor
Command
Player
"move north"
split "move" and "north"
move to the north path
move successfully
Console
CommandProcessor
Command
Player

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public class CommandProcessor : Command
10     {
11         List<Command> _commands;
12
13         public CommandProcessor() : base(new string[]
             { "commandprocessor" })
14         {
15             _commands = new List<Command>();
16             _commands.Add(new LookCommand());
17             _commands.Add(new MoveCommand());
18         }
19
20         public override string Execute(Player p, string[] text)
21         {
22             foreach (Command c in _commands)
23             {
24                 if (c.AreYou(text[0]))
25                 {
26                     return c.Execute(p, text);
27                 }
28             }
29             return "This command is not availble.";
30         }
31     }
32 }
33
```

D:\workspace\COS20007\COS

```
Enter player's name:
ijfoise
Enter player's description:
fnefow
You can move to:
- north: Forest

Enter a command:
look at me
You are ijfoise, you are carrying:
Bronze Sword (sword)
Bronze Axe (axe)
Bag (bag)

Enter a command:
move north
You moved to Forest.

Enter a command:
```

```
1  using SwinAdventure;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Test
9  {
10     public class CommandProcessorTest
11     {
12         private CommandProcessor _commandProcessor;
13         private Player _player;
14         private Locations _location;
15
16         [SetUp]
17         public void Setup()
18         {
19             _commandProcessor = new CommandProcessor();
20             _location = new Locations( "Start Location", "This is the
                 starting location.");
21             _player = new Player("player", "player description") { Location
                  = _location };
22
23             // Adding a path to test MoveCommand
24             var destination = new Locations( "Destination", "This is the
                 destination.");
25             var path = new Paths(new string[] { "north" }, "North Path", "A
                  path to the north.", _location, destination);
26             _location.AddPath(path);
27         }
28
29         [Test]
30         public void TestMoveCommand()
31         {
32             string result = _commandProcessor.Execute(_player, new string[]
                  { "move", "north" });
33             Assert.AreEqual("You moved to Destination.\n", result);
34             Assert.AreEqual("Destination", _player.Location.Name);
35         }
36
37         [Test]
38         public void TestLookCommand()
39         {
40             // Add an item to the player's inventory to test LookCommand
41             var item = new Item(new string[] { "shovel" }, "a shovel",
                 "This is a rusty old shovel.");
42             _player.Inventory.Put(item);
43
```

```
44              string result = _commandProcessor.Execute(_player, new string[] ↪
                    { "look", "at", "shovel" });
45              Assert.AreEqual("a shovel (shovel): This is a rusty old         ↪
                  shovel.", result);
46          }
47
48          [Test]
49          public void TestInvalidCommand()
50          {
51              string result = _commandProcessor.Execute(_player, new string[] ↪
                    { "fly" });
52              Assert.AreEqual("This command is not availble.", result);
53          }
54
55          [Test]
56          public void TestMoveCommandInvalidDirection()
57          {
58              string result = _commandProcessor.Execute(_player, new string[] ↪
                    { "move", "south" });
59              Assert.AreEqual("I don't know how to move that.", result);
60          }
61
62          [Test]
63          public void TestLookCommandInvalidSyntax()
64          {
65              string result = _commandProcessor.Execute(_player, new string[] ↪
                    { "look", "shovel" });
66              Assert.AreEqual("I don't know how to look like that", result);
67          }
68      }
69  }
70
```

Test run finished: 50 Tests (50 Passed, 0 Failed, 0 ⚠ 0 Warnings ✖ 0 Errors

Test	Duration	T...	Error...
Test (50)	10 ms		
BagTest (5)	10 ms		
IdentifiableObjectTest (6)	< 1 ms		
InventoryTest (5)	< 1 ms		
ItemTest (3)	< 1 ms		
LocationTest (4)	< 1 ms		
LookCommandTest (8)	< 1 ms		
MoveCommandTest (4)	< 1 ms		
PathTest (5)	< 1 ms		
PlayerTest (5)	< 1 ms		
Test (5)	< 1 ms		
CommandProcessorTest (5)	< 1 ms		
TestInvalidCommand	< 1 ms		
TestLookCommand	< 1 ms		
TestLookCommandInval...	< 1 ms		
TestMoveCommand	< 1 ms		
TestMoveCommandInva...	< 1 ms		