

# CommandProcessor.cs

```
..07\Week_10\Iteration 7\Iteration1\CommandProcessor.cs 1
1  using static PowerArgs.Ansi.Cursor;
2
3  namespace Iteration1
4  {
5      public class CommandProcessor : Command
6      {
7          List<Command> _commands;
8
9          public CommandProcessor() : base(new string[] { "command" })
10         {
11             _commands = new List<Command>();
12             _commands.Add(new LookCommand());
13             _commands.Add(new MoveCommand());
14         }
15
16         public override string Execute(Player p, string[] text)
17         {
18             string[] array = text[0].Split(' ');
19             foreach (Command command in _commands)
20             {
21                 if (command.AreYou(array[0]))
22                 {
23                     return command.Execute(p, new string[] { text[0] });
24                 }
25             }
26             return "Wrong command!!!";
27         }
28     }
29 }
30 }
```

# Program.cs

```
...op\COS20007\Week_10\Iteration 7\Iteration1\Program.cs 1
1  using System.Runtime.InteropServices;
2
3  namespace Iteration1
4  {
5      internal class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Player name: ");
10             string name = Console.ReadLine();
11             Console.WriteLine("Player description: ");
12             string desc_ = Console.ReadLine();
13             Player player = new Player(name, desc_);
14
15             Item sword = new Item(new string[] { "sword" }, "sword", "Short ↵
16                                         range weapon!");
16             Item ak47 = new Item(new string[] { "ak47" }, "ak47", "Average ↵
17                                         range weapon with high damage!");
18
18             player.Inventory.Put(sword);
19             player.Inventory.Put(ak47);
20
21             Item grenade = new Item(new string[] { "grenade" }, "grenade", ↵
22                                         "Extreme damage and short range weapon!");
22             Bag bag1 = new Bag(new string[] { "bag1" }, "bag1", "");
23             bag1.Inventory.Put(grenade);
24             player.Inventory.Put(bag1);
25
26             Location location = new Location(new string[] { "military ↵
27                                         base" }, "military base", "large area");
27             Path _northpath = new Path(new string[] { "north" }, ↵
28                                         "hospital", "this is a hospital");
28
29             location.AddPath(_northpath);
30             player.CurrentLocation = location;
31
32             while (true)
33             {
34                 Console.WriteLine("Command -> ");
35                 string command = Console.ReadLine();
36                 CommandProcessor selected_command = new CommandProcessor();
37                 string message = selected_command.Execute(player, new ↵
38                                         string[] {command});
38                 Console.WriteLine(message);
39             }
40         }
41     }
42 }
```

# CommandProcessorUnitTest.cs

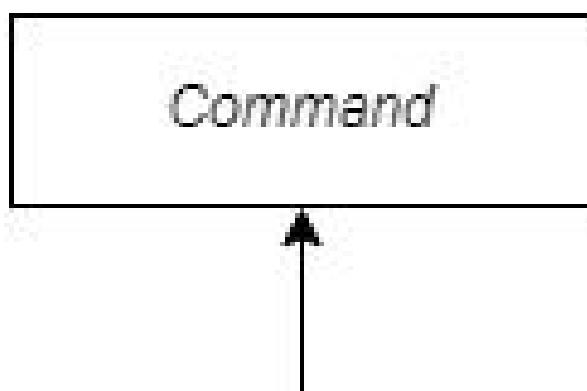
...\_10\Iteration 7\CommandProcessorUnitTest\UnitTest1.cs

1

```
1 using Iteration1;
2 using static PowerArgs.Ansi.Cursor;
3 using Path = Iteration1.Path;
4
5 namespace CommandProcessorUnitTest
6 {
7     public class Tests
8     {
9         private CommandProcessor _command;
10        private Player _player;
11        private Location _location;
12        private Item _sword;
13        private Item _ak47;
14        private Item _grenade;
15        private Path _northpath;
16
17        [SetUp]
18        public void Setup()
19        {
20            _command = new CommandProcessor();
21            _player = new Player("Chien", "A boy with high curiosity");
22            _location = new Location(new string[] { "military base", "military base", "large area" });
23            _sword = new Item(new string[] { "sword", "melee" }, "sword", "Short range weapon");
24            _ak47 = new Item(new string[] { "ak47" }, "ak47", "Long range weapon");
25            _grenade = new Item(new string[] { "grenade" }, "grenade", "Very high damage weapon!");
26            _location.Inventory.Put(_sword);
27            _location.Inventory.Put(_ak47);
28            _player.CurrentLocation = _location;
29            _northpath = new Path(new string[] { "north" }, "hospital", "this is a hospital");
30            _location.AddPath(_northpath);
31        }
32
33        [Test]
34        public void TestMoveCommand()
35        {
36            Assert.That(_command.Execute(_player, ["move north"]),
37                        Is.EqualTo("You have moved to hospital\n"));
38            Assert.Pass();
39        }
40
41        [Test]
42        public void TestLookCommand()
43        {
44            Assert.That(_command.Execute(_player, ["look at me"]),
45                        Is.EqualTo("You are looking at yourself\n"));
46        }
47    }
48}
```

```
        Is.EqualTo(_player.FullDescription + "\n"));
44    }
45}
46
47[TestMethod]
48public void TestWrongCommand()
49{
50    Assert.That(_command.Execute(_player, new string[] { "look      ↵
51        around" })), Is.EqualTo("What do you want to look at?\n"));
52    Assert.That(_command.Execute(_player, new string[] { "run      ↵
53        north" })), Is.EqualTo("Wrong command!!!"));
54    Assert.That(_command.Execute(_player, new string[]
55        { "hello" })), Is.EqualTo("Wrong command!!!"));
56    Assert.Pass();
57}
58}
```

## UML Class Diagram



CommandProcessor

---

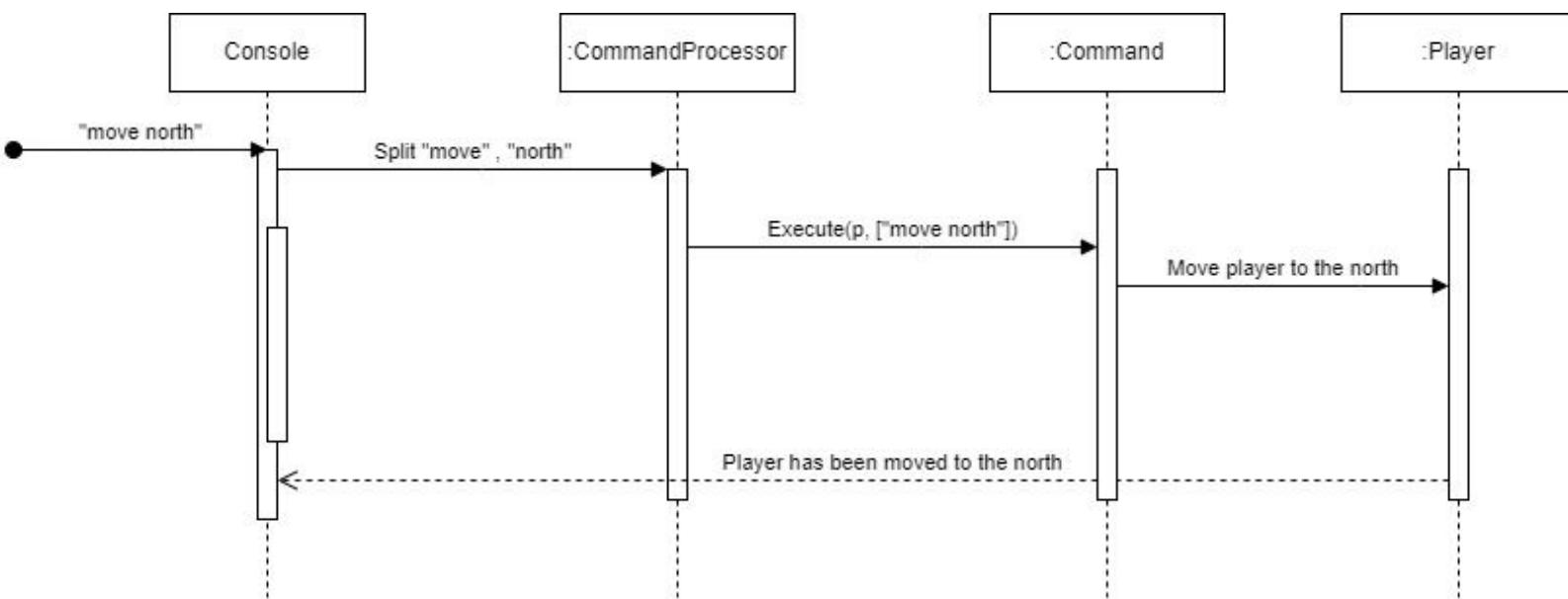
- \_commands : List<Command>

+ CommandProcessor()

+ Execute (Player p, string[] text) : string <<override>>

---

## UML Sequence Diagram



## Program Running

The screenshot shows a Microsoft Visual Studio interface with the following details:

- File**, **Edit**, **View**, **Git**, **Project**, **Build**, **Debug**, **Test**, **Analyze**, **Tools**, **Extensions**, **Window**, **Help**, **Search**, and **Iteration1** in the top menu.
- Sign in** button in the top right corner.
- Units** and **Items** buttons in the top left.
- C:\Users\PC\Desktop\COS200** is the active project in the title bar.
- Output** window at the bottom left showing command-line output:

```
Player name: Chien
Player description: hii
Command -> move north
You have moved to hospital

Command -> look at inventory
I cannot find the inventory

Command -> look at inventory
You are Chien, hii
You are carrying
    a sword (sword)
    a ak47 (ak47)
    a bag1 (bag1)

Command -> look at sword
Short range weapon!

Command -> |
```

- Code Editor** window showing C# code:

```
35
36     string command = Console.ReadLine();
37     CommandProcessor selected_command = new CommandProcessor();
38     string message = selected_command.Execute(player, new string[] { command });
Console.WriteLine(message);
```

- Output** window at the bottom left showing build status:

```
Show output from: Build
```

- CommandProcessorUnitTest** tab is active in the top right.
- Solution Explorer** window on the right showing the project structure:

  - Solution 'Iteration1' (10 of 1)
  - BagUnitTests
  - CommandProcessorUnitTests
  - InventoryUnitTests
  - ItemUnitTests
  - Iteration1
    - Dependencies
    - Bag.cs
    - Command.cs
    - CommandProcessor.cs
    - GameObject.cs
    - IdentifiableObject.cs
    - IHaveInventory.cs
    - Inventory.cs
    - Item.cs
    - Location.cs
    - LookCommand.cs
    - MoveCommand.cs
    - Path.cs
    - Player.cs
    - Program.cs
  - LookCommandUnitTests
  - PlayerUnitTests
  - TestIdentifiableObject
  - TestLocations
  - TestMoveCommand
    - Dependencies
    - UnitTest1.cs

- Task List** window on the far right.
- Ready** status in the bottom left.
- Add to Source Control** and **Select Repository** buttons in the bottom right.
- System tray icons in the bottom right.

# UnitTest Passing

The screenshot shows the Microsoft Visual Studio Test Explorer window. At the top, there are buttons for running tests (play/pause, stop, refresh) and a status bar indicating "Test run finished: 45 Tests (45 Passed, 0 Failed, 0 Skipped) run in 282 ms". The main area displays a table of test results:

Test	Duration	Traits	Error Message
TestMoveCommand (5)	149 ms		
TestLocations (5)	163 ms		
TestIdentifiableObject (6)	167 ms		
PlayerUnitTests (5)	97 ms		
LookCommandUnitTests (8)	183 ms		
ItemUnitTests (3)	106 ms		
InventoryUnitTests (5)	189 ms		
CommandProcessorUnit... (14 ms)	14 ms		
CommandProcessorUnit... (14 ms)	14 ms		
Tests (3) (14 ms)	14 ms		
TestWrongCommand (< 1 ms)	< 1 ms		
TestMoveCommand (< 1 ms)	< 1 ms		
TestLookCommand (14 ms)	14 ms		
BagUnitTests (5)	193 ms		

On the right side, a "Test Detail Summary" pane is open, showing details for the selected test "TestWrongCommand":

- Result: ✓ TestWrongCommand
- Source: [UnitTest1.cs](#) line 48
- Duration: < 1 ms

The taskbar at the bottom shows various pinned application icons, and the system tray indicates the date and time as 7/25/2024 9:42 PM.