

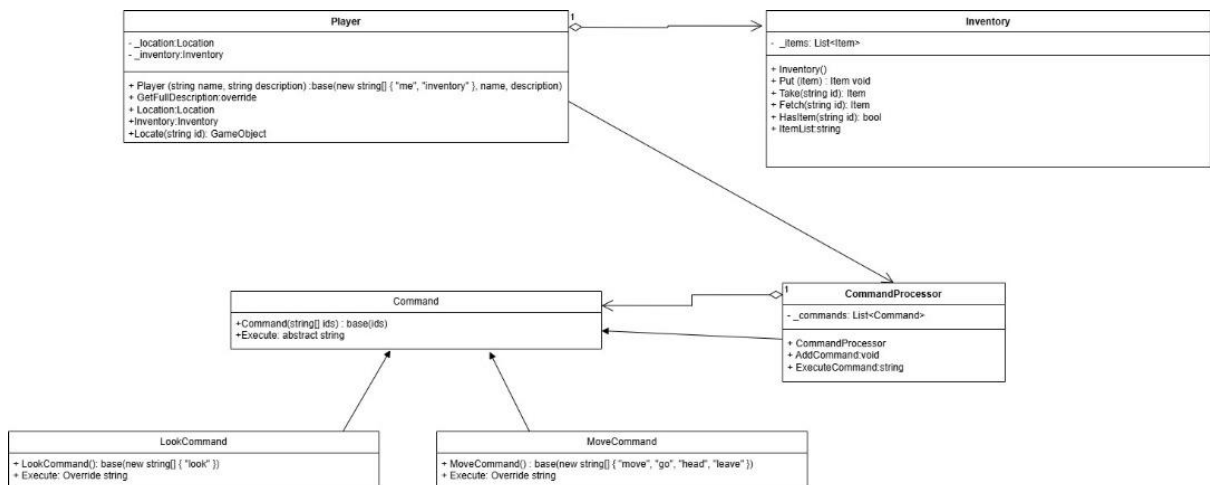
10.1 Correction

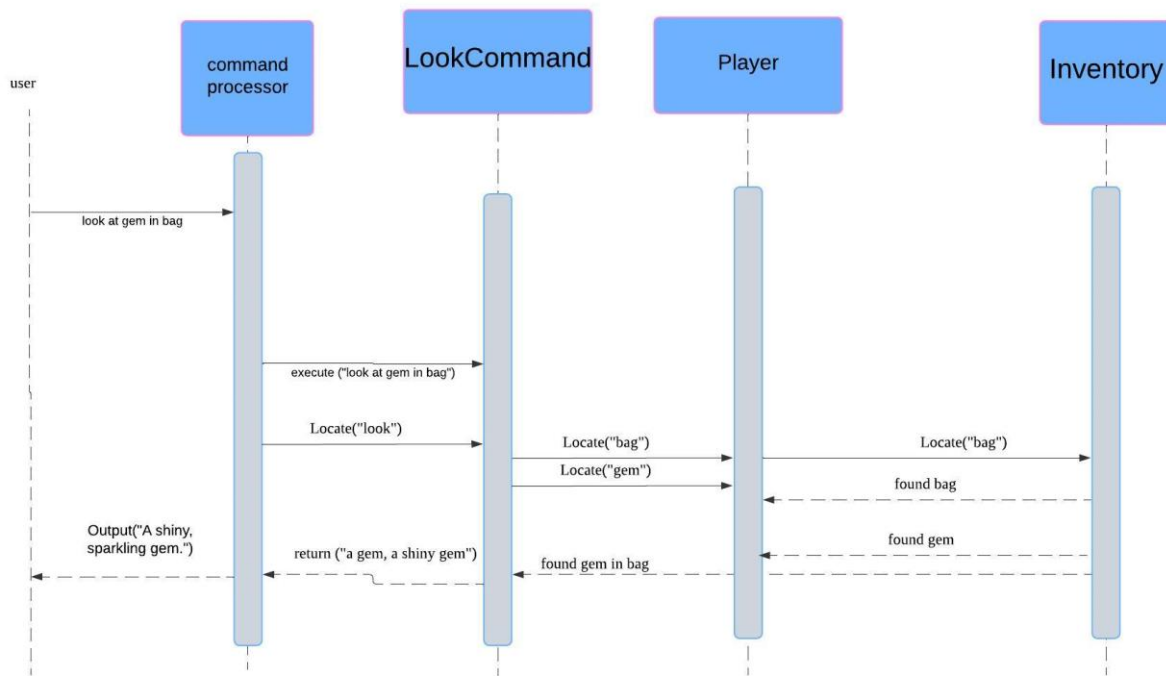
Test run finished: 54 Tests (54 Passed, 0 Failed, 0 Skipped) run in 427 ms

Test	Duration	Traits	Error Message	Run	Debug
✓ BagTesting (5)	154 ms				
✓ CommandProcessorTesting (4)	158 ms				
✓ InventoryUnitTesting (5)	158 ms				
✓ ItemUnitTesting (3)	149 ms				
✓ LocationPathTesting (2)	153 ms				
✓ LocationTesting (3)	189 ms				
✓ LookCommandLocationTesting (5)	164 ms				
✓ LookCommandTesting (8)	151 ms				
✓ MoveCommandTesting (3)	154 ms				
✓ PathValidationTesting (1)	163 ms				
✓ PlayerLocationTesting (4)	187 ms				
✓ PlayerUnitTesting (5)	149 ms				
✓ UnitTesting1 (6)	150 ms				

Group Summary
BagTesting
Tests in group: 5
⌚ Total Duration: 154 ms
Outcomes
✓ 5 Passed

```
C:\Users\Joshua\OneDrive\Des x + v
Enter your player's name: joshua
Enter a description for your player: a king
Welcome, joshua the a king!
Item 'Sword' added to inventory.
Item 'Shield' added to inventory.
Item 'a small bag' added to inventory.
A small bag has been added to your inventory.
Item 'a gem' added to inventory.
Item 'gem' has been placed inside the small bag.
Enter a command (or type 'exit' to quit):
look around
Dark Forest: A dark and ominous forest.
Enter a command (or type 'exit' to quit):
move north
You move north to the Small Village.
Enter a command (or type 'exit' to quit):
move west
You can't go that way.
Enter a command (or type 'exit' to quit):
move east
You move east to the Snowy Mountain.
Enter a command (or type 'exit' to quit):
|
```





Program.cs:

```

using System;
using SwinAdventure;
class Program
{
    static void Main(string[] args)
    {
        // Display available commands at the start
        DisplayHelp();
        // Step 1: Get the player's name and description from the user.
        Console.WriteLine("Enter your player's name: ");
        string playerName = Console.ReadLine();
        Console.WriteLine("Enter a description for your player: ");
        string playerDescription = Console.ReadLine();
    }
}
  
```

```
Player player = new Player(playerName, playerDescription);

Console.WriteLine($"\\nWelcome {playerName}, {playerDescription}!\\n");

// Step 2: Setup locations and paths

Location home = new Location(new string[] { "home" }, "Home", "Your cozy home.");
Location forest = new Location(new string[] { "forest" }, "Forest", "A dark, dense forest.");
Location mountain = new Location(new string[] { "mountain" }, "Mountain", "A tall, imposing
mountain.");

home.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.North, forest));
forest.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.South, home));
forest.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.East, mountain));
mountain.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.West, forest));

player.Location = home;
Console.WriteLine("You are currently at your home.\\n");

// Step 3: Setup player inventory with items

Item sword = new Item(new string[] { "sword" }, "a sword", "A sharp, shining sword.");
Item shield = new Item(new string[] { "shield" }, "a shield", "A sturdy wooden shield.");
player.Inventory.Put(sword);
player.Inventory.Put(shield);

Bag smallBag = new Bag(new string[] { "bag", "small bag" }, "a small bag", "A small leather
bag.");
player.Inventory.Put(smallBag);

Item gem = new Item(new string[] { "gem" }, "a gem", "A shiny, valuable gem.");
smallBag.Inventory.Put(gem);

// Step 4: Initialize the CommandProcessor and add commands
```

```

CommandProcessor commandProcessor = new CommandProcessor();
commandProcessor.AddCommand(new MoveCommand());

// Step 5: Main game loop
while (true)
{
    Console.WriteLine($"
You are in {player.Location.Name}. Enter a command (or type 'exit' to
quit):");

    string command = Console.ReadLine();

    if (command.ToLower() == "exit")
    {
        break;
    }
    else if (command.ToLower() == "help")
    {
        DisplayHelp();
        continue;
    }

    string[] commandWords = command.Split(' ');

    string result = commandProcessor.ExecuteCommand(player, commandWords);
    Console.WriteLine(result);
}

Console.WriteLine("Goodbye!");
}

// Method to display all available commands
static void DisplayHelp()

```

```

{
    Console.WriteLine("\nAvailable Commands:");
    Console.WriteLine(" look - Examine your current surroundings.");
    Console.WriteLine(" inventory - Check the items you have.");
    Console.WriteLine(" move [direction] - Move in a specified direction (e.g., 'move north').");
    Console.WriteLine(" go [direction] - Another way to move in a direction (e.g., 'go east').");
    Console.WriteLine(" head [direction] - Another way to move in a direction (e.g., 'head west').");
    Console.WriteLine(" leave [direction] - Another way to move in a direction (e.g., 'leave
south').");

    Console.WriteLine(" help - Display this list of commands.");
    Console.WriteLine(" exit - Quit the game.");
    Console.WriteLine();
}
}

```

Commandprocessor.cs:

```
using System.Collections.Generic;
```

```
namespace SwinAdventure
```

```

{
    public class CommandProcessor
    {
        private List<Command> _commands;

        public CommandProcessor()
        {
            _commands = new List<Command>
            {
                new MoveCommand(),
                new LookCommand(),

```

```

        };
    }

    public void AddCommand(Command command)
    {
        _commands.Add(command);
    }

    public string ExecuteCommand(Player player, string[] text)
    {
        if (text.Length == 0)
            return "No command provided.";

        string commandId = text[0].ToLower();

        foreach (Command command in _commands)
        {
            if (command.AreYou(commandId))
            {
                return command.Execute(player, text);
            }
        }

        return $"I don't know how to {commandId}.";
    }
}

```

Commandprocessor testing:

using NUnit.Framework;

```

using System.Numerics;

using SwinAdventure;

namespace SwinAdventure.Tests
{
    [TestFixture]
    public class CommandProcessorTests
    {
        private CommandProcessor _commandProcessor;

        private Player _player;

        private Location _startLocation;

        [SetUp]
        public void Setup()
        {
            _startLocation = new Location(new string[] { "start" }, "Starting Location", "The start of your
adventure.");

            _player = new Player("Adventurer", "A brave explorer");

            _player.Location = _startLocation;

            _commandProcessor = new CommandProcessor();
        }

        [Test]
        public void TestExecuteMoveCommand()
        {
            _startLocation.AddPath(new Path(Path.Direction.North, new Location(new string[] { "north" },
"North Location", "You are north.")));

            string result = _commandProcessor.ExecuteCommand(_player, new string[] { "move", "north"
});

            Assert.AreEqual("You move north to the North Location.", result);
        }
    }
}

```

```
[Test]

public void TestUnknownCommand()
{
    string result = _commandProcessor.ExecuteCommand(_player, new string[] { "fly", "upward"
});
    Assert.AreEqual("I don't know how to fly.", result);
}
```

```
[Test]

public void TestIncompleteCommand()
{
    string result = _commandProcessor.ExecuteCommand(_player, new string[] { "move" });
    Assert.AreEqual("Where do you want to go?", result);
}
```

```
[Test]

public void TestExecuteLookCommand()
{
    _player.Inventory.Put(new Item(new string[] { "gem" }, "a shiny gem", "It's a very shiny
gem.));
    string result = _commandProcessor.ExecuteCommand(_player, new string[] { "look", "at",
"gem" });
    Assert.AreEqual("a shiny gem: It's a very shiny gem.", result);
}
}
```