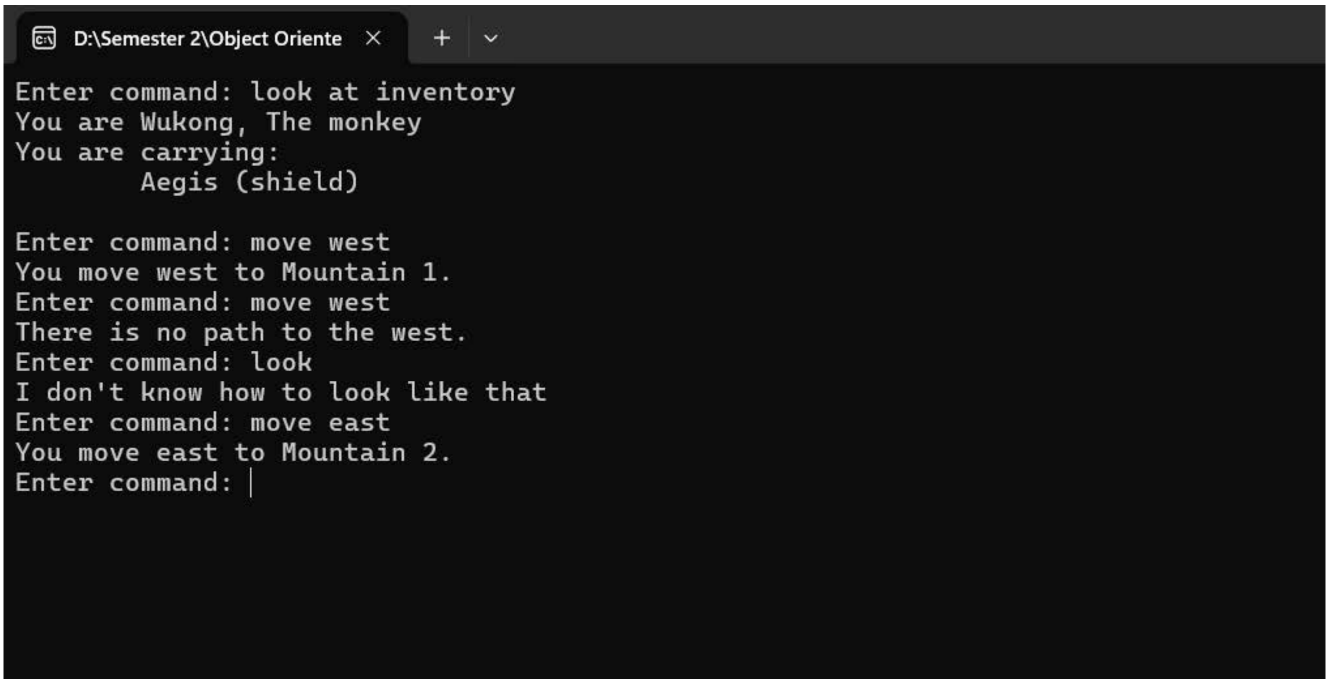
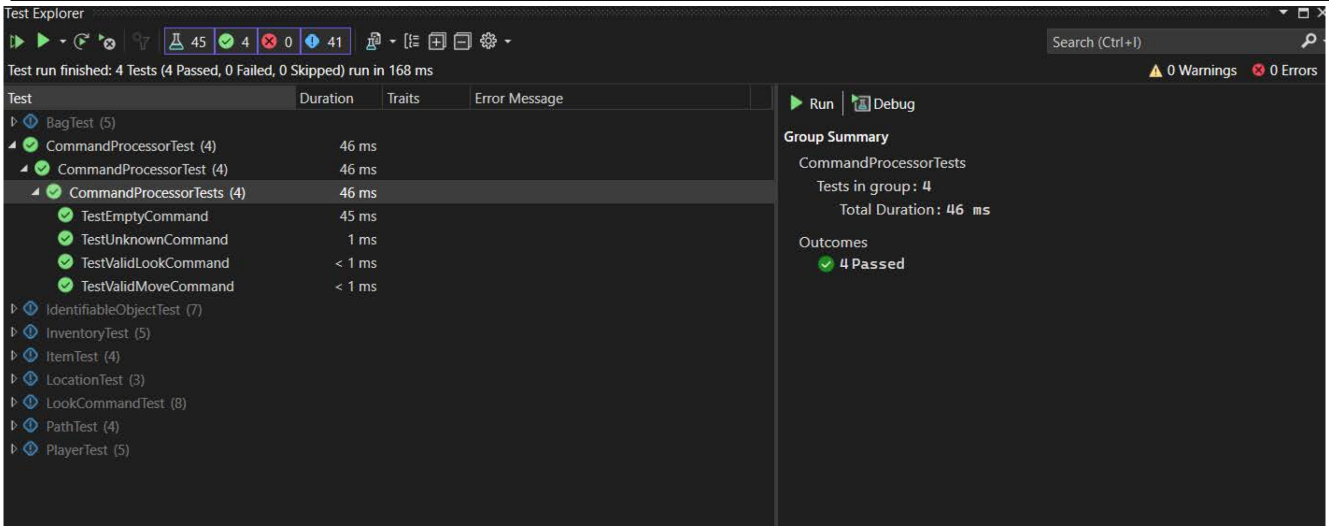
Khoi Nguyen Pham - 104772183

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _10._1C
{
    public class CommandProcessor
    {
        private List<Command> _commands;


        public CommandProcessor()
        {
            _commands = new List<Command>();
            _commands.Add(new LookCommand());
            _commands.Add(new MoveCommand());
        }

        public string ExecuteCommand(string commandText, Player player)
        {
            if (string.IsNullOrWhiteSpace(commandText))
            {
                return "Invalid command.";
            }


            string[] commandWords = commandText.Split(' ', 
              StringSplitOptions.RemoveEmptyEntries); // Split the input 
              into words

            if (commandWords.Length == 0)
            {
                return "Invalid command.";
            }


            string commandKeyword = commandWords[0].ToLower();


            foreach (var command in _commands)
            {
                if (command.AreYou(commandKeyword))
                {
                    return command.Execute(player, commandWords);
                }
            }

            return $"Unknown command: {commandKeyword}";
        }
    }
}
```

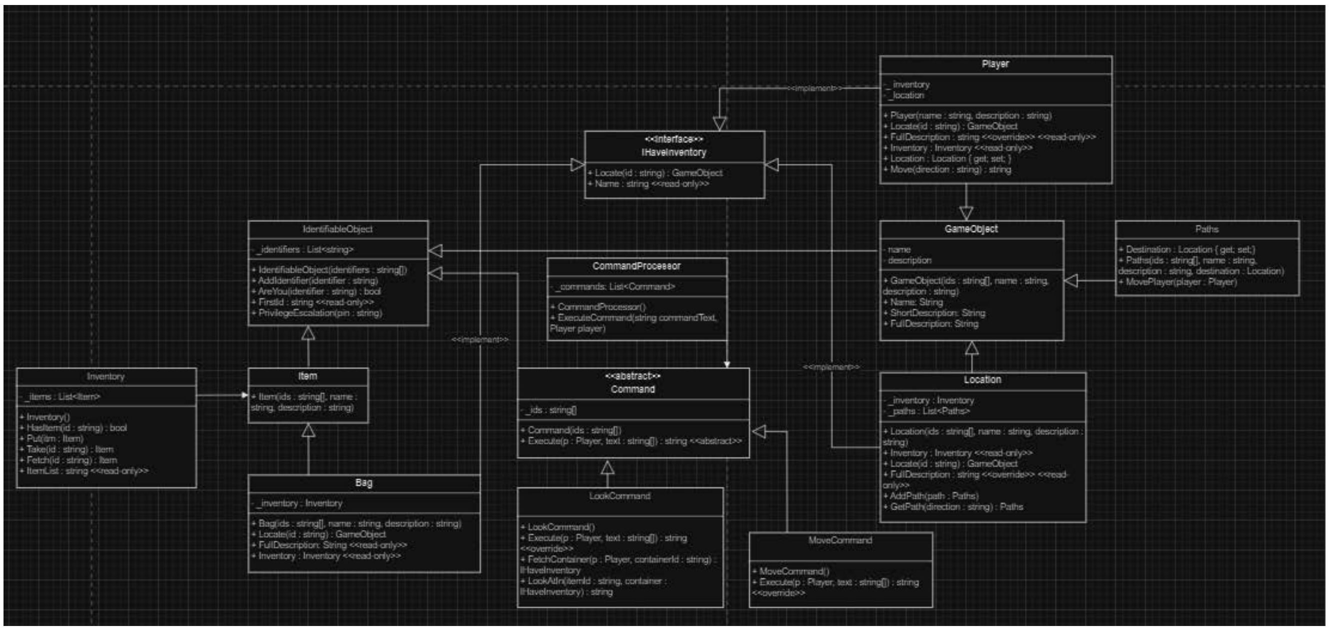
```csharp
using _10._1C;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using NUnit.Framework;
using System.Numerics;


namespace CommandProcessorTest
{
    [TestFixture]
    public class CommandProcessorTests
    {

        private CommandProcessor _commandProcessor;
        private Player _player;


        [SetUp]
        public void SetUp()
        {
            _commandProcessor = new CommandProcessor();

            _player = new Player("Nguyen", "A programmer");

            Location mountain1 = new Location(new string[] { "mountain1" }, "Mountain 1", "first mountain");
            Location mountain2 = new Location(new string[] { "mountain2" }, "Mountain 2", "second mountain");
            Item sword = new Item(new string[] { "sword" }, "Excalibur", "a strong sword");
            _player.Inventory.Put(sword);


            Paths pathToMountain1 = new Paths(new string[] { "west" }, "Journey to the West", "path leading West", mountain1);
            Paths pathToMountain2 = new Paths(new string[] { "east" }, "Journey to the East", "path leading East", mountain2);

            _player.Location = mountain1;
            _player.Location = mountain2;
            mountain2.AddPath(pathToMountain1);
            mountain1.AddPath(pathToMountain2);
        }

        [Test]
        public void TestValidLookCommand()
        {
            // Arrange
            string input = "look at sword in inventory";
            string expectedResponse = "a strong sword";

            // Act
            string response = _commandProcessor.ExecuteCommand(input, _player);

            // Assert
            Assert.AreEqual(expectedResponse, response);
        }

        [Test]
        public void TestValidMoveCommand()
        {
            // Arrange
            string input = "move west";
            string expectedResponse = "You move west to Mountain 1.";

            // Act
            string response = _commandProcessor.ExecuteCommand(input, _player);

            // Assert
            Assert.AreEqual(expectedResponse, response);
            Assert.AreEqual("Mountain 1", _player.Location.Name);
        }

        [Test]
        public void TestUnknownCommand()
        {
            // Arrange
            string input = "fly";
            string expectedResponse = "Unknown command: fly";

            // Act
            string response = _commandProcessor.ExecuteCommand(input, _player);

            // Assert
            Assert.AreEqual(expectedResponse, response);
        }

        [Test]
        public void TestEmptyCommand()
        {
            // Arrange
            string input = ""; // Empty command string
            string expectedResponse = "Invalid command.";

            // Act
            string response = _commandProcessor.ExecuteCommand(input, _player);

            // Assert
            Assert.AreEqual(expectedResponse, response);
        }
        }

}
```



```
D:\Semester 2\Object Oriente

Enter command: look at inventory
You are Wukong, The monkey
You are carrying:
        Aegis (shield)

Enter command: move west
You move west to Mountain 1.
Enter command: move west
There is no path to the west.
Enter command: look
I don't know how to look like that
Enter command: move east
You move east to Mountain 2.
Enter command:
```



```
Test Explorer
45  4  0  41
Test run finished: 4 Tests (4 Passed, 0 Failed, 0 Skipped) run in 168 ms          0 Warnings  0 Errors

Test                                  Duration   Traits   Error Message
▷ BagTest (5)
◢ CommandProcessorTest (4)             46 ms
  ◢ CommandProcessorTest (4)           46 ms
    ◢ CommandProcessorTests (4)        46 ms
        TestEmptyCommand               45 ms
        TestUnknownCommand              1 ms
        TestValidLookCommand          < 1 ms
        TestValidMoveCommand          < 1 ms
▷ IdentifiableObjectTest (7)
▷ InventoryTest (5)
▷ ItemTest (4)
▷ LocationTest (3)
▷ LookCommandTest (8)
▷ PathTest (4)
▷ PlayerTest (5)

Run | Debug
Group Summary
  CommandProcessorTests
    Tests in group: 4
      Total Duration: 46 ms

  Outcomes
    ✓ 4 Passed
```


Player
- _inventory
- _location
+ Player(name : string, description : string)
+ Locate(id : string) : GameObject
+ FullDescription : string <<override>> <<read-only>>
+ Inventory : Inventory <<read-only>>
+ Location : Location { get; set; }
+ Move(direction : string) : string
<<implement>>
<<interface>>
IHaveInventory
+ Locate(id : string) : GameObject
+ Name : string <<read-only>>
IdentifiableObject
- _identifiers : List<string>
+ IdentifiableObject(identifiers : string[])
+ AddIdentifier(identifier : string)
+ AreYou(identifier : string) : bool
+ FirstId : string <<read-only>>
+ PrivilegeEscalation(pin : string)
CommandProcessor
- _commands: List<Command>
+ CommandProcessor()
+ ExecuteCommand(string commandText, Player player)
GameObject
- name
- description
+ GameObject(ids : string[], name : string, description : string)
+ Name: String
+ ShortDescription: String
+ FullDescription: String
Paths
+ Destination : Location { get; set;}
+ Paths(ids : string[], name : string, description : string, destination : Location)
+ MovePlayer(player : Player)
<<implement>>
<<implement>>
Inventory
- _items : List<Item>
+ Inventory()
+ HasItem(id : string) : bool
+ Put(itm : Item)
+ Take(id : string) : Item
+ Fetch(id : string) : Item
+ ItemList : string <<read-only>>
Item
+ Item(ids : string[], name : string, description : string)
<<abstract>>
Command
- _ids : string[]
+ Command(ids : string[])
+ Execute(p : Player, text : string[]) : string <<abstract>>
Location
- _inventory : Inventory
- _paths : List<Paths>
+ Location(ids : string[], name : string, description : string)
+ Inventory : Inventory <<read-only>>
+ Locate(id : string) : GameObject
+ FullDescription : string <<override>> <<read-only>>
+ AddPath(path : Paths)
+ GetPath(direction : string) : Paths
Bag
- _inventory : Inventory
+ Bag(ids : string[], name : string, description : string)
+ Locate(id : string) : GameObject
+ FullDescription: String <<read-only>>
+ Inventory : Inventory <<read-only>>
LookCommand
+ LookCommand()
+ Execute(p : Player, text : string[]) : string <<override>>
+ FetchContainer(p : Player, containerId : string) : IHaveInventory
+ LookAtIn(itemId : string, container : IHaveInventory) : string
MoveCommand
+ MoveCommand()
+ Execute(p : Player, text : string[]) : string <<override>>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _10._1C
{
    public class Program
    {
        public static void Main(string[] args)
        {
            /**
            Console.WriteLine("Enter your name: ");
            string playerName = Console.ReadLine();
            Console.WriteLine("Enter your description: ");
            string playerDescription = Console.ReadLine();
            Player player = new Player(playerName, playerDescription);

            Item sword = new Item(new string[] {"sword"}, "Excalibur", 
              "a strong sword");
            Item shield = new Item(new string[] {"shield"}, "Aegis", "a 
              strong shield");

            player.Inventory.Put(sword);
            player.Inventory.Put(shield);

            Bag backpack = new Bag(new string[] { "backpack" }, 
              "Adidas", "a big backpack");
            player.Inventory.Put(backpack);
            Item gem = new Item(new string[] { "gem" }, "Ruby", "a rare 
              gem");
            backpack.Inventory.Put(gem);

            /////
            LookCommand lookCommand = new LookCommand();
            while (true)
            {
                Console.Write("What do you want to look at?: ");
                string input = Console.ReadLine();
                string[] commandWords = input.Split(' ');
                string result = lookCommand.Execute(player, 
                  commandWords);
                Console.WriteLine(result);
            }
            **/

            Location mountain1 = new Location(new string[] 
              { "mountain1" }, "Mountain 1", "first mountain");
            Location mountain2 = new Location(new string[] 
              { "mountain2" }, "Mountain 2", "second mountain");
            Item sword = new Item(new string[] { "sword" }, "Excalibur", 
               "a strong sword");
            mountain1.Inventory.Put(sword);


            Paths pathToMountain1 = new Paths(new string[] { "west" }, "Journey to the West", "path leading West", mountain1);
            Paths pathToMountain2 = new Paths(new string[] { "east" }, "Journey to the East", "path leading East", mountain2);

            mountain1.AddPath(pathToMountain2);
            mountain2.AddPath(pathToMountain1);

            Item shield = new Item(new string[] { "shield" }, "Aegis", "a strong shield");

            Player player = new Player("Wukong", "The monkey");
            player.Inventory.Put(shield);
            player.Location = mountain1;
            player.Location = mountain2;

            //MoveCommand moveCommand = new MoveCommand();

            CommandProcessor commandProcessor = new CommandProcessor();
            while (true)
            {
                Console.Write("Enter command: ");
                string command = Console.ReadLine();

                // Execute the command and get the response
                string response = commandProcessor.ExecuteCommand(command, player);
                Console.WriteLine(response);
            }

        }
    }
}

```