Swinburne University of Technology

COS20007 Object Oriented Programming

# Case Study - Iteration 8 - Command Processor

PDF generated at 11:33 on Wednesday 11th October, 2023

```
namespace SwinAdventure
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            string name;
            string description;
            Player player;

            //Direct paths
            Path path1;
            Path path2;
            Path path3;
            Path path4;

            // Indirect paths
            Path path5a;
            Path path6a;
            Path path7a;
            Path path8a;

            Path path5b;
            Path path6b;
            Path path7b;
            Path path8b;

            Console.WriteLine("Welcome to SwinAdventure!");

            Console.WriteLine("Enter Player Name: ");
            name = Console.ReadLine();
            Console.WriteLine("Enter Player Description:");
            description = Console.ReadLine();

            player = new Player(name, description);

            Item sword = new Item(new string[] { "sword" }, "sword", "a bronze sword");
            Item gem = new Item(new string[] { "gem" }, "ruby", "a shining gem");
            Item pc = new Item(new string[] { "pc" }, "computer", "a small computer");
            Item book = new Item(new string[] { "book" }, "novel", "a lifetime autobiography");
            Item fruit = new Item(new string[] { "fruit" }, "orange", "sweet and fresh");
            Item fish = new Item(new string[] { "fish" }, "salmon", "very fresh seafood");
            Item ticket = new Item(new string[] { "ticket" }, "ticket", "horror movie ticket");
            Item popcorn = new Item(new string[] { "popcorn" }, "popcorn", "cheesy or caramel");

            Bag bag = new Bag(new string[] { "bag" }, "backpack", "a big bag");
```

```
47
48              Location home = new Location(new string[] { "home" }, "home", "home sweet
    ↪ home", "west, northwest and southwest");
49              Location school = new Location(new string[] { "school" }, "school", "a
    ↪ study time", "east, northeast and southeast");
50              Location market = new Location(new string[] { "market" }, "market",
    ↪ "fresh fruits here", "north, northwest and northeast");
51              Location cinema = new Location(new string[] { "cinema" }, "cinema", "late
    ↪ service tonight", "south, southwest and southeast");
52
53              // Direct paths
54              path1 = new Path(new string[] { "east" }, "east direction", "bus stop",
    ↪ school, home);
55              path2 = new Path(new string[] { "west" }, "west direction", "bus stop",
    ↪ home, school);
56              path3 = new Path(new string[] { "north" }, "north direction", "grocery
    ↪ store", market, cinema);
57              path4 = new Path(new string[] { "south" }, "south direction", "grocery
    ↪ store", cinema, market);
58
59              // Indirect paths
60              path5a = new Path(new string[] { "northeast" }, "north-east direction",
    ↪ "greeny park", school, cinema);
61              path5b = new Path(new string[] { "northeast" }, "north-east direction",
    ↪ "asian restaurant", market, home);
62
63              path6a = new Path(new string[] { "northwest" }, "north-west direction",
    ↪ "fortress arcade", home, cinema);
64              path6b = new Path(new string[] { "northwest" }, "north-west direction",
    ↪ "public library", market, school);
65
66              path7a = new Path(new string[] { "southeast" }, "south-east direction",
    ↪ "public library", school, market);
67              path7b = new Path(new string[] { "southeast" }, "south-east direction",
    ↪ "fortress arcade", cinema, home);
68
69              path8a = new Path(new string[] { "southwest" }, "south-west direction",
    ↪ "asian restaurant", home, market);
70              path8b = new Path(new string[] { "southwest" }, "south-west direction",
    ↪ "greeny park", cinema, school);
71
72              player.Inventory.Put(bag);
73              player.Inventory.Put(gem);
74              player.Inventory.Put(sword);
75              player.Inventory.Put(pc);
76
77              bag.Inventory.Put(sword);
78              bag.Inventory.Put(pc);
79
80              player.Location = home;
81
82              home.Inventory.Put(sword);
83              home.Inventory.Put(gem);
```

```
84
85              school.Inventory.Put(book);
86              school.Inventory.Put(pc);
87
88              cinema.Inventory.Put(popcorn);
89              cinema.Inventory.Put(ticket);
90
91              market.Inventory.Put(fish);
92              market.Inventory.Put(fruit);
93
94              school.AddPath(path1);
95              school.AddPath(path5a);
96              school.AddPath(path7a);
97
98              home.AddPath(path2);
99              home.AddPath(path6a);
100             home.AddPath(path8a);
101
102             market.AddPath(path3);
103             market.AddPath(path5b);
104             market.AddPath(path6b);
105
106             cinema.AddPath(path4);
107             cinema.AddPath(path7b);
108             cinema.AddPath(path8b);
109
110
111             string cmd;
112
113             LookCommand look = new LookCommand();
114             MoveCommand move = new MoveCommand();
115
116             while (true)
117             {
118                 Console.WriteLine("Enter a Command: ");
119                 cmd = Console.ReadLine();
120
121                 //Using for 9.2C only
122                 /*if (cmd == "quit")
123                 {
124                     quit = true;
125                     moved = false;
126                     looked = false;
127                 }
128
129                 else if (cmd == "move")
130                 {
131                     moved = true;
132                     looked = false;
133                     quit = false;
134                 }
135                 else if (cmd == "look")
136                 {
```

```
137                     looked = true;
138                     moved = false;
139                     quit = false;
140                 }*/
141
142                 string[] exeCommand = cmd.ToLower().Split(' ');
143                 if (exeCommand[0] == "quit")
144                 {
145                     break;
146                 }
147                 else
148                 {
149                     Console.WriteLine(new CommandProcessor().ExecuteCommand(player,
    ↪ exeCommand));
150                 }
151             }
152         }
153     }
154 }
```

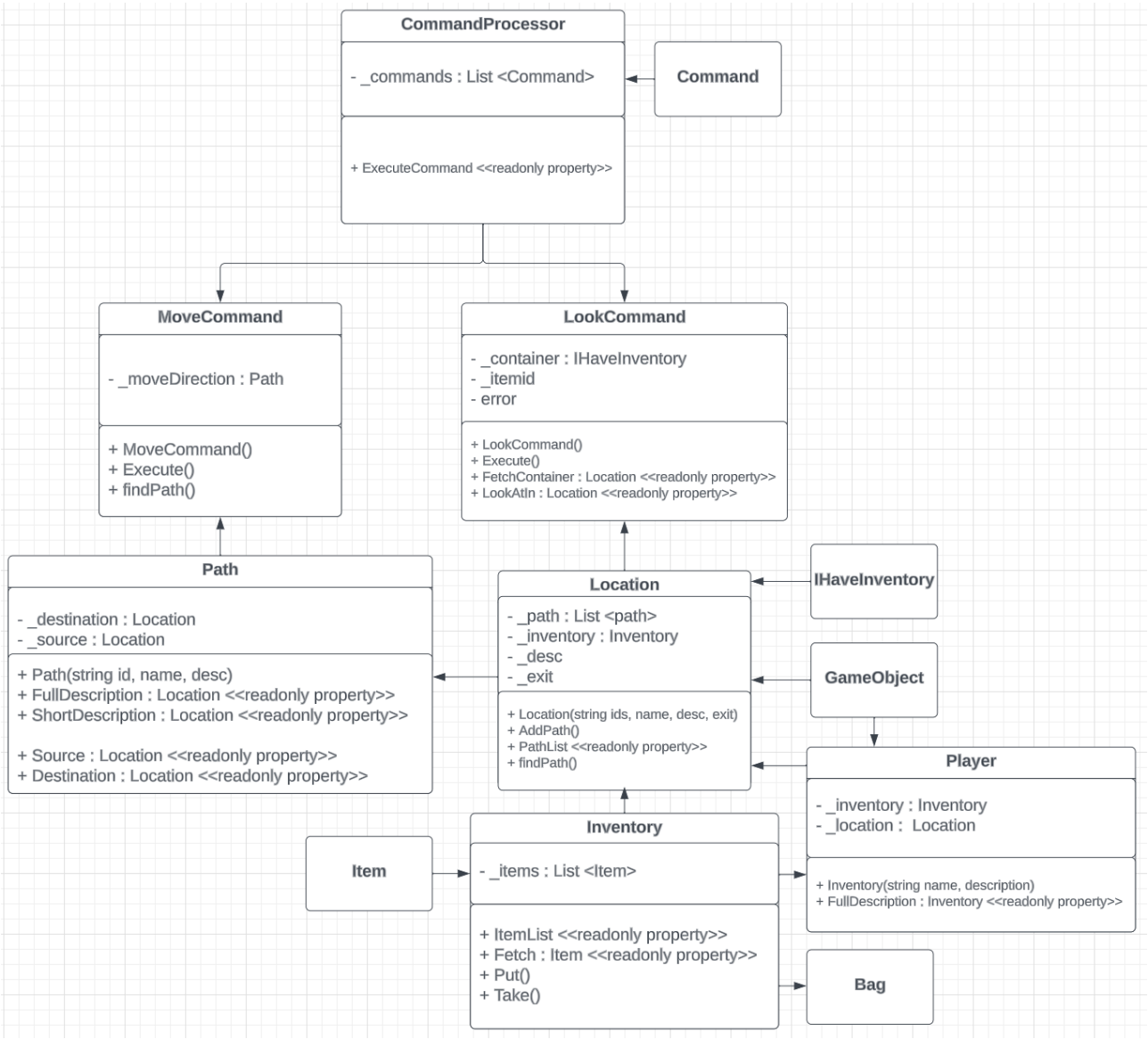
```
using System;
using System.Numerics;

namespace SwinAdventure
{
    public class CommandProcessor
    {
        List<Command> _commands;

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

        public string ExecuteCommand(Player p, string[] text)
        {
            string input = text[0].ToLower();
            Command exeCommand = null;
            // loop to find the most suitable command
            foreach (Command command in _commands)
            {
                if (command.AreYou(input))
                {
                    exeCommand = command;
                    break;
                }
            }
            // if can't find the suitable command
            if (exeCommand == null)
            {
                return "I don't know how to " + input + ".";
            }
            return exeCommand.Execute(p, text);
        }
    }
}
```

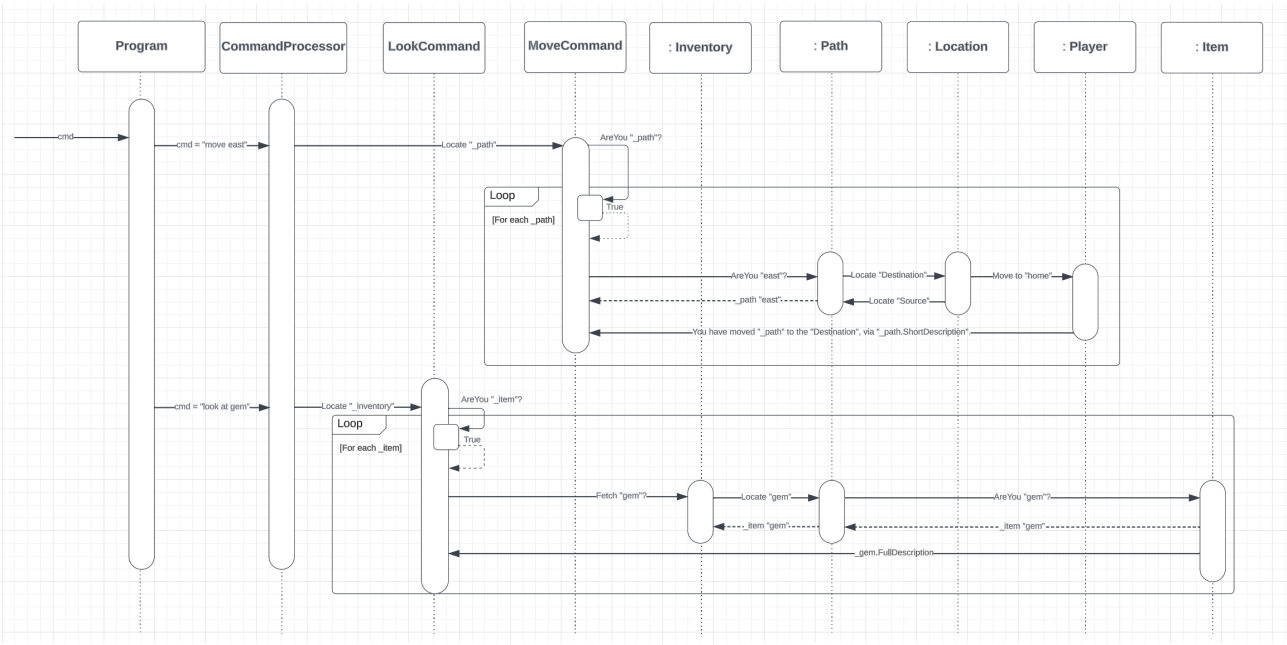
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;
using SwinAdventure;
using Path = SwinAdventure.Path;

namespace CommandProcessorTest
{
    [TestFixture]
    public class TestCommandProcessor
    {
        CommandProcessor command;

        Path path1;
        Path path2;

        Location home;
        Location school;

        Item sword;
        Item pc;
        Item gem;
        Item book;

        Player player;

        Bag bag;

        [SetUp]
        public void Setup()
        {
            command = new();
            player = new Player("Fred", "the mighty programmer");

            //Move setup
            home = new Location(new string[] { "home" }, "home", "home sweet home",
↪   "west");
            school = new Location(new string[] { "school" }, "school", "a study
↪   time", "east");

            path1 = new Path(new string[] { "east" }, "east", "crowded village",
↪   school, home);
            path2 = new Path(new string[] { "west" }, "west", "empty highway", home,
↪   school);

            school.AddPath(path1);
            home.AddPath(path2);

            //Look setup
            sword = new Item(new string[] { "sword" }, "sword", "a bronze sword");
```

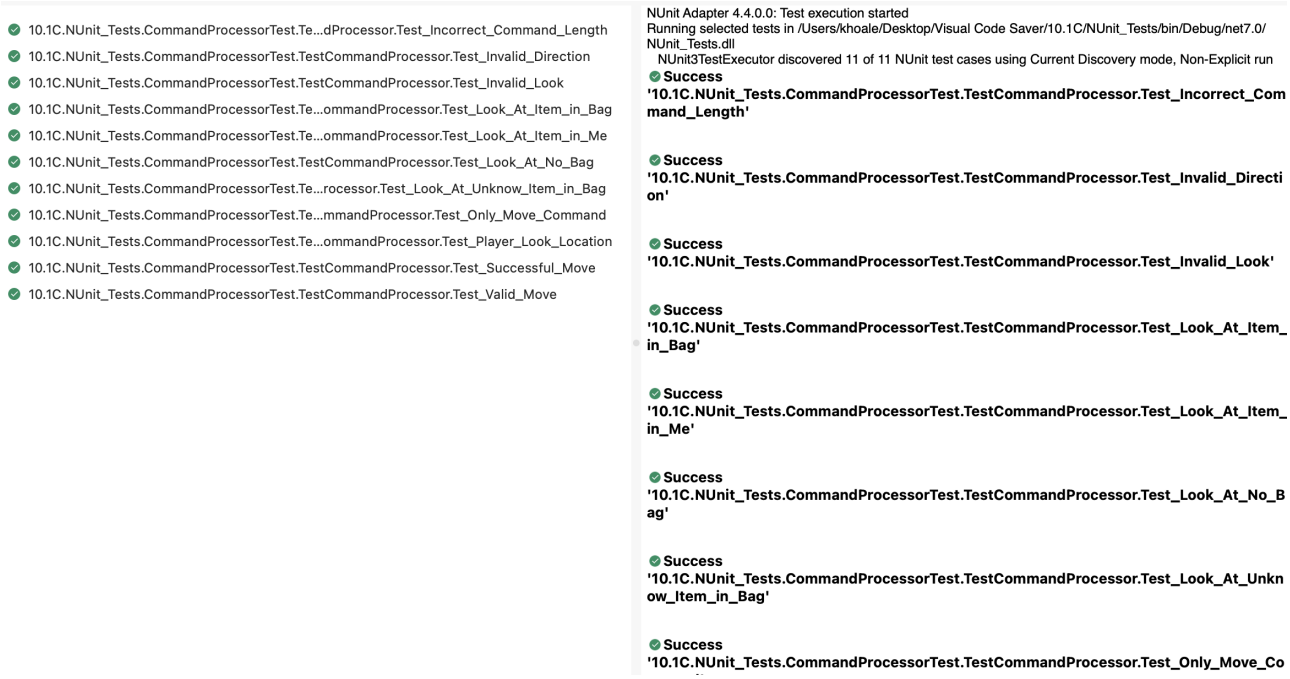
```
50              pc = new Item(new string[] { "pc" }, "computer", "a small computer");
51              gem = new Item(new string[] { "gem" }, "ruby", "a shining gem");
52              book = new Item(new string[] { "book" }, "novel", "a lifetime
    ↪  autobiography");
53
54              bag = new Bag(new string[] { "backpack" }, "backpack", "a big bag");
55          }
56
57          // LookCommand Test
58          [Test]
59          public void Test_Look_At_Item_in_Bag()
60          {
61              player.Inventory.Put(bag);
62              bag.Inventory.Put(gem);
63              string actual = command.ExecuteCommand(player, new string[] { "look",
    ↪  "at", "gem", "in", $"backpack" });
64              string expect = $"{gem.FullDescription}";
65              Assert.AreEqual(actual, expect);
66          }
67
68          [Test]
69          public void Test_Look_At_Unknow_Item_in_Bag()
70          {
71              player.Inventory.Put(bag);
72              string actual = command.ExecuteCommand(player, new string[] { "look",
    ↪  "at", "sword", "in", $"backpack" });
73              string expect = $"I can't find the sword";
74              Assert.AreEqual(actual, expect);
75          }
76
77          [Test]
78          public void Test_Look_At_No_Bag()
79          {
80              string Output = command.ExecuteCommand(player, new string[] { "look",
    ↪  "at", "backpack", "in", $"{player.FirstID}" });
81              string exp = $"I can't find the backpack";
82              Assert.AreEqual(exp, Output);
83          }
84
85          [Test]
86          public void Test_Look_At_Item_in_Me()
87          {
88              player.Inventory.Put(book);
89              player.Inventory.Put(pc);
90              string actual1 = command.ExecuteCommand(player, new string[] { "look",
    ↪  "at", "book", "in", "me" });
91              string actual2 = command.ExecuteCommand(player, new string[] { "look",
    ↪  "at", "pc", "in", "me" });
92              string expect1 = $"{book.FullDescription}";
93              string expect2 = $"{pc.FullDescription}";
94              Assert.AreEqual(actual1, expect1);
95              Assert.AreEqual(actual2, expect2);
96          }
```

```
97
98          [Test]
99          public void Test_Invalid_Look()
100         {
101             Assert.AreEqual(command.ExecuteCommand(player, new string[] { "look",
    ↪ "around" }), "I don't know how to look like that.");
102             Assert.AreEqual(command.ExecuteCommand(player, new string[] { "look",
    ↪ "by", "none" }), "What do you want to look at?");
103         }
104
105         [Test]
106         public void Test_Player_Look_Location()
107         {
108             player.Location = home;
109             home.Inventory.Put(gem);
110             Assert.AreEqual(command.ExecuteCommand(player, new string[] { "look" }),
    ↪ $"You are at the home\nhome sweet home\nYou can see these
    ↪ items:\n{home.Inventory.ItemList}\n{home.ShortDescription}.");
111         }
112
113         // MoveCommand Test
114         [Test]
115         public void Test_Valid_Move()
116         {
117             player.Location = home;
118             command.ExecuteCommand(player, new string[] { "move", "west" });
119             Assert.AreEqual(school.Name, player.Location.Name);
120
121             player.Location = school;
122             command.ExecuteCommand(player, new string[] { "go", "east" });
123             Assert.AreEqual(home.Name, player.Location.Name);
124         }
125
126         [Test]
127         public void Test_Successful_Move()
128         {
129             player.Location = school;
130             string actual1 = command.ExecuteCommand(player, new string[] { "move",
    ↪ "east" });
131             string expected1 = "You have moved east, via a crowded village to the
    ↪ home";
132             Assert.That(actual1, Is.EqualTo(expected1));
133
134             player.Location = home;
135             string actual2 = command.ExecuteCommand(player, new string[] { "go",
    ↪ "west" });
136             string expected2 = "You have moved west, via a empty highway to the
    ↪ school";
137             Assert.That(actual2, Is.EqualTo(expected2));
138         }
139
140         [Test]
141         public void Test_Incorrect_Command_Length()
```

```
142         {
143             string actual = command.ExecuteCommand(player, new string[] { "move",
    ↪  "to", "north" });
144             string expected = "Error in move input.";
145             Assert.That(actual, Does.Contain(expected));
146         }
147 
148         [Test]
149         public void Test_Only_Move_Command()
150         {
151             string actual1 = command.ExecuteCommand(player, new string[] { "move" });
152             string expected1 = "Where do you want to move?";
153             Assert.That(actual1, Is.EqualTo(expected1));
154 
155             string actual2 = command.ExecuteCommand(player, new string[] { "go" });
156             string expected2 = "Where do you want to move?";
157             Assert.That(actual2, Is.EqualTo(expected2));
158         }
159 
160         [Test]
161         public void Test_Invalid_Direction()
162         {
163             player.Location = home;
164             string actual1 = command.ExecuteCommand(player, new string[] { "move",
    ↪  "south" });
165             string expected1 = "Invalid pathway";
166             Assert.That(actual1, Is.EqualTo(expected1));
167 
168             player.Location = home;
169             string actual2 = command.ExecuteCommand(player, new string[] { "go",
    ↪  "north" });
170             string expected2 = "Invalid pathway";
171             Assert.That(actual2, Is.EqualTo(expected2));
172         }
173     }
174 }
```


CommandProcessor
- _commands : List <Command>
+ ExecuteCommand <<readonly property>>
Command
MoveCommand
- _moveDirection : Path
+ MoveCommand()
+ Execute()
+ findPath()
LookCommand
- _container : IHaveInventory
- _itemid
- error
+ LookCommand()
+ Execute()
+ FetchContainer : Location <<readonly property>>
+ LookAtIn : Location <<readonly property>>
Path
- _destination : Location
- _source : Location
+ Path(string id, name, desc)
+ FullDescription : Location <<readonly property>>
+ ShortDescription : Location <<readonly property>>
+ Source : Location <<readonly property>>
+ Destination : Location <<readonly property>>
Location
- _path : List <path>
- _inventory : Inventory
- _desc
- _exit
+ Location(string ids, name, desc, exit)
+ AddPath()
+ PathList <<readonly property>>
+ findPath()
IHaveInventory
GameObject
Player
- _inventory : Inventory
- _location : Location
+ Inventory(string name, description)
+ FullDescription : Inventory <<readonly property>>
Inventory
- _items : List <Item>
+ ItemList <<readonly property>>
+ Fetch : Item <<readonly property>>
+ Put()
+ Take()
Item
Bag


Program
CommandProcessor
LookCommand
MoveCommand
: Inventory
: Path
: Location
: Player
: Item
cmd
cmd = "move east"
Locate "_path"
AreYou "_path"?
Loop
[For each _path]
True
AreYou "east"?
Locate "Destination"
Move to "home"
_path "east"
Locate "Source"
You have moved "_path" to the "Destination", via "_path.ShortDescription".
cmd = "look at gem"
Locate "_inventory"
AreYou "_item"?
Loop
[For each _item]
True
Fetch "gem"?
Locate "gem"
AreYou "gem"?
_item "gem"
_item "gem"
_gem.FullDescription


10.1C.NUnit_Tests.CommandProcessorTest.Te...dProcessor.Test_Incorrect_Command_Length
10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Invalid_Direction
10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Invalid_Look
10.1C.NUnit_Tests.CommandProcessorTest.Te...ommandProcessor.Test_Look_At_Item_in_Bag
10.1C.NUnit_Tests.CommandProcessorTest.Te...ommandProcessor.Test_Look_At_Item_in_Me
10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Look_At_No_Bag
10.1C.NUnit_Tests.CommandProcessorTest.Te...rocessor.Test_Look_At_Unknow_Item_in_Bag
10.1C.NUnit_Tests.CommandProcessorTest.Te...mmandProcessor.Test_Only_Move_Command
10.1C.NUnit_Tests.CommandProcessorTest.Te...ommandProcessor.Test_Player_Look_Location
10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Successful_Move
10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Valid_Move
NUnit Adapter 4.4.0.0: Test execution started
Running selected tests in /Users/khoale/Desktop/Visual Code Saver/10.1C/NUnit_Tests/bin/Debug/net7.0/NUnit_Tests.dll
NUnit3TestExecutor discovered 11 of 11 NUnit test cases using Current Discovery mode, Non-Explicit run
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Incorrect_Command_Length'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Invalid_Direction'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Invalid_Look'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Look_At_Item_in_Bag'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Look_At_Item_in_Me'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Look_At_No_Bag'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Look_At_Unknow_Item_in_Bag'
Success '10.1C.NUnit_Tests.CommandProcessorTest.TestCommandProcessor.Test_Only_Move_Co

```
Welcome to SwinAdventure!
Enter Player Name:
Khoa
Enter Player Description:
a random guy
Enter a Command:
look
You are at the home
home sweet home
You can see these items:
        a sword (sword)
        a ruby (gem)

There are exits via the west, northwest and southwest.
Enter a Command:
move northwest
You have moved north-west direction, via a fortress arcade to the cinema
Enter a Command:
look
You are at the cinema
late service tonight
You can see these items:
        a popcorn (popcorn)
        a ticket (ticket)

There are exits via the south, southwest and southeast.
Enter a Command:
move south
You have moved south direction, via a grocery store to the market
Enter a Command:
look
You are at the market
fresh fruits here
You can see these items:
        a salmon (fish)
        a orange (fruit)

There are exits via the north, northwest and northeast.
Enter a Command:
move northwest
You have moved north-west direction, via a public library to the school
Enter a Command:
move east
You have moved east direction, via a bus stop to the home
Enter a Command:
look
You are at the home
home sweet home
You can see these items:
        a sword (sword)
        a ruby (gem)

There are exits via the west, northwest and southwest.
```