

# Location.cs

...op\COS20007\Week\_9\Iteration 7\Iteration1\Location.cs

1

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Linq;
5 using System.Reflection.Metadata.Ecma335;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace Iteration1
10 {
11     public class Location : GameObject, IHaveInventory
12     {
13         private Inventory _inventory;
14         private Dictionary<string, Path> _paths;
15
16         public Location(string[] ids, string name, string desc) : base(ids, ↵
            name, desc)
17         {
18             _inventory = new Inventory();
19             _paths = new Dictionary<string, Path>();
20         }
21
22         public GameObject Locate(string id)
23         {
24             if (AreYou(id))
25             {
26                 return this;
27             }
28             else if (_inventory.HasItem(id))
29             {
30                 return _inventory.Fetch(id);
31             }
32
33             foreach (Path p in _paths.Values)
34             {
35                 if (p.AreYou(id))
36                 {
37                     return p;
38                 }
39             }
40
41             return null;
42         }
43
44         public void AddPath(Path path)
45         {
46             string direction = path.Direction;
47             _paths.Add(direction, path);
48         }
49     }
50 }
```

```
49
50     public Dictionary<string, Path> Paths
51     {
52         get => _paths;
53     }
54
55     public Inventory Inventory
56     {
57         get => _inventory;
58     }
59
60     public string Description
61     {
62         get => base.FullDescription;
63     }
64 }
65 }
66
```

# Path.cs

...esktop\COS20007\Week\_9\Iteration 7\Iteration1\Path.cs

1

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Iteration1
8 {
9     public class Path : GameObject
10    {
11        public Path(string[] ids, string destination, string desc) : base
12            (ids, destination, desc)
13        {
14        }
15
16        public string Move(Player player)
17        {
18            string id = this.Destination;
19            string destination = this.Destination;
20            string desc = this.Description;
21            Location newLocation = new Location(new string[] { id },
22                destination, desc);
23            player.CurrentLocation = newLocation;
24            return $"You have moved to {destination}\n";
25        }
26
27        public string Description
28        {
29            get => base.FullDescription;
30        }
31
32        public string Direction
33        {
34            get => base.FirstId();
35        }
36
37        public string Destination
38        {
39            get => base.Name;
40        }
41    }
42 }
```

# MoveCommand.cs

...COS20007\Week\_9\Iteration 7\Iteration1\MoveCommand.cs

1

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Iteration1
8 {
9     public class MoveCommand : Command
10    {
11        private List<string> _directions;
12        private List<string> _identifiers;
13
14        public MoveCommand() : base(new string[] { "move", "go", "head",
15            "leave" })
16        {
17            _directions = new List<string> { "north", "west", "south",
18                "east"};
19            _identifiers = new List<string> { "move", "head", "go",
20                "leave" };
21        }
22
23        public override string Execute(Player p, string[] text)
24        {
25            string[] array = text[0].Split(" ");
26            if (array.Length != 2)
27            {
28                return "Wrong command input!\n";
29            }
30            else if (!_directions.Contains(array[1]) || !
31                _identifiers.Contains(array[0]))
32            {
33                return "Wrong command input!\n";
34            }
35            else
36            {
37                if (p.CurrentLocation.Locate(array[1]) == null)
38                {
39                    return "There is not that path in this location!\n";
40                }
41                else
42                {
43                    Path selected_path = p.CurrentLocation.Paths[array[1]];
44                    return selected_path.Move(p);
45                }
46            }
47        }
48    }
49 }
```

# MoveCommandUnitTest

...20007\Week\_9\Iteration 7\TestMoveCommand\UnitTest1.cs

1

```
1 using Iteration1;
2 using Path = Iteration1.Path;
3
4 namespace TestMoveCommand
5 {
6     public class Tests
7     {
8         private Player _player;
9         private Location _location;
10        private Item _sword;
11        private Item _ak47;
12        private Item _grenade;
13        private Path _northpath;
14        private MoveCommand _move;
15
16        [SetUp]
17        public void Setup()
18        {
19            _player = new Player("Chien", "A boy with high curiosity");
20            _location = new Location(new string[] { "military base" },
21                                     "military base", "large area");
22            _sword = new Item(new string[] { "sword", "melee" }, "sword",
23                              "Short range weapon");
24            _ak47 = new Item(new string[] { "ak47" }, "ak47", "Long range
25                              weapon");
26            _grenade = new Item(new string[] { "grenade" }, "grenade",
27                                "Very high damage weapon!");
28            _location.Inventory.Put(_sword);
29            _location.Inventory.Put(_ak47);
30            _player.CurrentLocation = _location;
31            _northpath = new Path(new string[] { "north" }, "hospital",
32                                   "this is a hospital");
33            _location.AddPath(_northpath);
34            _move = new MoveCommand();
35        }
36
37        [Test]
38        public void TestPathMovesPlayer()
39        {
40            _northpath.Move(_player);
41            Assert.That(_player.CurrentLocation.Name, Is.EqualTo
42                        (_northpath.Destination));
43            Assert.Pass();
44        }
45
46        [Test]
47        public void TestGetPathFromLocation()
48        {
49            string id = "north";
```

```
44     Assert.That(_location.Locate(id), Is.EqualTo(_northpath));
45     Assert.Pass();
46 }
47
48 [Test]
49 public void TestPlayerLeaveLocation()
50 {
51     Assert.That(_move.Execute(_player, new string[] { "move
52         north" }), Is.EqualTo($"You have moved to
53         {_northpath.Destination}\n"));
54     _player.CurrentLocation = _location;
55     Assert.That(_move.Execute(_player, new string[] { "leave
56         north" }), Is.EqualTo($"You have moved to
57         {_northpath.Destination}\n"));
58     _player.CurrentLocation = _location;
59     Assert.That(_move.Execute(_player, new string[] { "go
60         north" }), Is.EqualTo($"You have moved to
61         {_northpath.Destination}\n"));
62     _player.CurrentLocation = _location;
63     Assert.That(_move.Execute(_player, new string[] { "head
64         north" }), Is.EqualTo($"You have moved to
65         {_northpath.Destination}\n"));
66     Assert.Pass();
67 }
68
69 [Test]
70 public void TestWrongCommand()
71 {
72     Assert.That(_move.Execute(_player, new string[] { "run
73         north" }), Is.EqualTo($"Wrong command input!\n"));
74     Assert.That(_move.Execute(_player, new string[] { "go home" }),
75         Is.EqualTo($"Wrong command input!\n"));
76     Assert.That(_move.Execute(_player, new string[] { "move
77         south" }), Is.EqualTo("There is not that path in this
78         location!\n"));
79     Assert.Pass();
80 }
81
82 [Test]
83 public void TestNotMoveWithUnvalidPath()
84 {
85     _move.Execute(_player, new string[] { "go home" });
86     _move.Execute(_player, new string[] { "move south" });
87     Assert.That(_player.CurrentLocation, Is.EqualTo(_location));
88     Assert.Pass();
89 }
90 }
```

# Program.cs

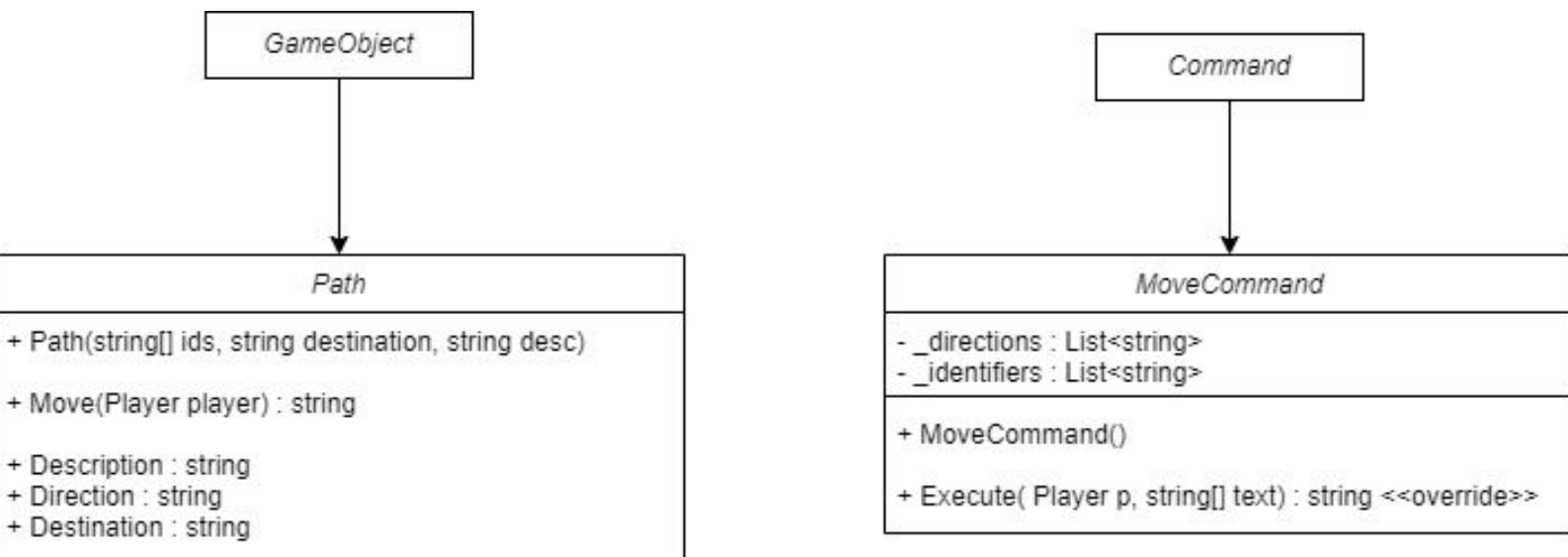
...eek\_9\9.2C - Answer\Iteration 7\Iteration1\Program.cs

1

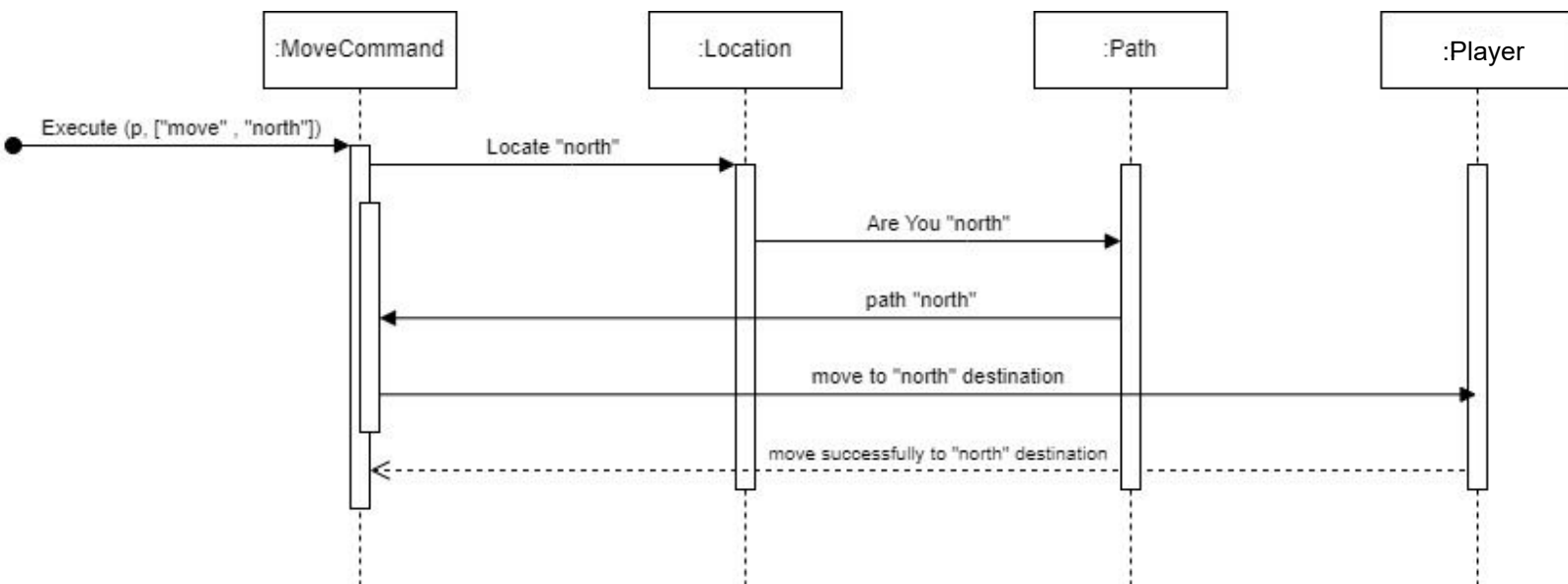
```
1 using System.Runtime.InteropServices;
2
3 namespace Iteration1
4 {
5     internal class Program
6     {
7         static void Main(string[] args)
8         {
9             string name;
10            string desc_;
11            string command;
12            Item sword;
13            Item ak47;
14            Item grenade;
15
16            Console.Write("Player name: ");
17            name = Console.ReadLine();
18            Console.Write("Player description: ");
19            desc_ = Console.ReadLine();
20            Player player = new Player(name, desc_);
21
22            sword = new Item(new string[] { "sword" }, "sword", "Short
23                range weapon!");
24            ak47 = new Item(new string[] { "ak47" }, "ak47", "Average range
25                weapon with high damage!");
26
27            player.Inventory.Put(sword);
28            player.Inventory.Put(ak47);
29
30            grenade = new Item(new string[] { "grenade" }, "grenade",
31                "Extreme damage and short range weapon!");
32            Bag bag1 = new Bag(new string[] { "bag1" }, "bag1", "");
33            bag1.Inventory.Put(grenade);
34            player.Inventory.Put(bag1);
35
36            Location _location = new Location(new string[] { "military
37                base" }, "military base", "large area");
38            player.CurrentLocation = _location;
39            Path _northpath = new Path(new string[] { "north" },
40                "hospital", "this is a hospital");
41            _location.AddPath(_northpath);
42
43            LookCommand look_command = new LookCommand();
44            MoveCommand move_command = new MoveCommand();
45            while (true)
46            {
47                Console.Write("Which look command do you want to execute:
48                    ");
49                command = Console.ReadLine();
```

```
44         string message1 = look_command.Execute(player, new string[] { command });
45         Console.WriteLine(message1);
46
47         Console.Write("Which move command do you want to execute: ");
48         command = Console.ReadLine();
49         string message2 = move_command.Execute(player, new string[] { command });
50         Console.WriteLine(message2);
51     }
52 }
53 }
54 }
```

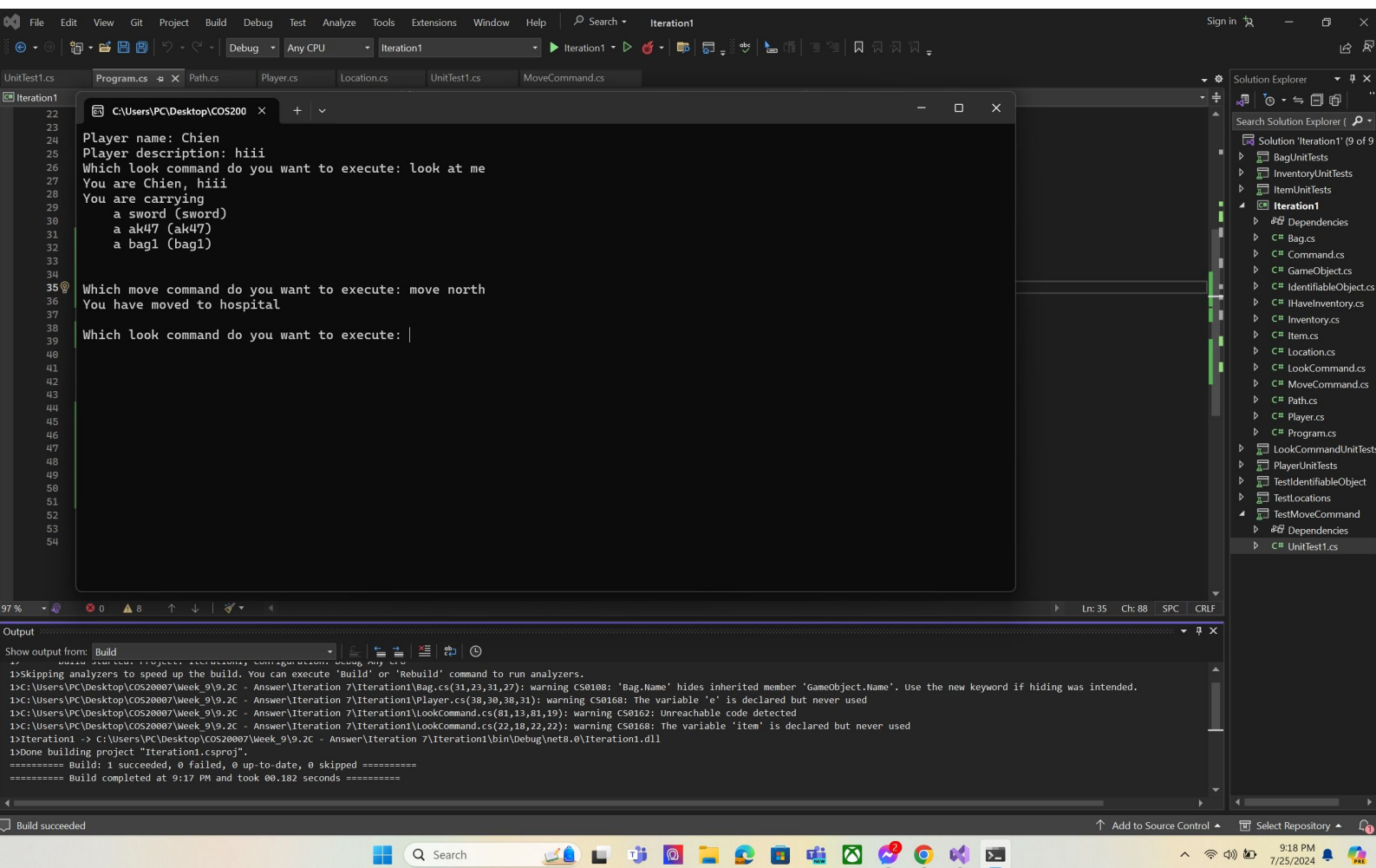
## UML Diagram



# UML Sequence Diagram



# Program Running



```
22 Player name: Chien
23
24 Player description: hiii
25
26 Which look command do you want to execute: look at me
27 You are Chien, hiii
28
29 You are carrying
30     a sword (sword)
31     a ak47 (ak47)
32     a bag1 (bag1)
33
34
35 Which move command do you want to execute: move north
36 You have moved to hospital
37
38
39 Which look command do you want to execute: |
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```

```
1>Skipping analyzers to speed up the build. You can execute 'Build' or 'Rebuild' command to run analyzers.
1>C:\Users\PC\Desktop\COS20007\Week_9\9.2C - Answer\Iteration 7\Iteration1\Bag.cs(31,23,31,27): warning CS0108: 'Bag.Name' hides inherited member 'GameObject.Name'. Use the new keyword if hiding was intended.
1>C:\Users\PC\Desktop\COS20007\Week_9\9.2C - Answer\Iteration 7\Iteration1\Player.cs(38,30,38,31): warning CS0168: The variable 'e' is declared but never used
1>C:\Users\PC\Desktop\COS20007\Week_9\9.2C - Answer\Iteration 7\Iteration1\LookCommand.cs(81,13,81,19): warning CS0162: Unreachable code detected
1>C:\Users\PC\Desktop\COS20007\Week_9\9.2C - Answer\Iteration 7\Iteration1\LookCommand.cs(22,18,22,22): warning CS0168: The variable 'item' is declared but never used
1>Iteration1 -> C:\Users\PC\Desktop\COS20007\Week_9\9.2C - Answer\Iteration 7\Iteration1\bin\Debug\net8.0\Iteration1.dll
1>Done building project "Iteration1.csproj".
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
===== Build completed at 9:17 PM and took 00.182 seconds =====
```

# Unit Test Passing

Test Explorer

Test discovery skipped: All test containers are up to date

42 42 0

Search (Ctrl+I)

0 Warnings 0 Errors

Test	Duration	Traits	Error Message
TestMoveCommand (5)	13 ms		
TestMoveCommand (5)	13 ms		
Tests (5)	13 ms		
TestWrongCommand	< 1 ms		
TestPlayerLeaveLocat...	1 ms		
TestPathMovesPlayer	< 1 ms		
TestNotMoveWithUn...	< 1 ms		
TestGetPathFromLoc...	12 ms		
TestLocations (5)	95 ms		
TestIdentifiableObject (6)	159 ms		
PlayerUnitTests (5)	114 ms		
LookCommandUnitTests (8)	67 ms		
ItemUnitTests (3)	187 ms		
InventoryUnitTests (5)	158 ms		
BagUnitTests (5)	115 ms		

**Group Summary**

TestMoveCommand

Tests in group : 5

Total Duration: 13 ms

**Outcomes**

5 Passed

Windows taskbar: Search, 9:13 PM, 7/25/2024