Selected files

7 printable files

Week_9/9.2/SwinAdventure/Location.cs
Week_9/9.2/SwinAdventure/MoveCommand.cs
Week_9/9.2/SwinAdventure/Path.cs
Week_9/9.2/SwinAdventure/Player.cs
Week_9/9.2/SwinAdventure/Program.cs
Week_9/9.2/TestPath/TestMove.cs
Week_9/9.2/TestPath/TestPath.cs

Week_9/9.2/SwinAdventure/Location.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace SwinAdventure
7  {
8      public class Location : GameObject, IHaveInventory
9      {
10         private Inventory _inventory;
11         private List<Path> _paths;
12
13         public Location(string name, string desc) : base(new string[] { "location",
   "room", "here" }, name, desc)
14         {
15             _inventory = new Inventory();
16             _paths = new List<Path>();
17         }
18
19         public Location(string name, string desc, List<Path> paths) : this(name,
   desc)
20         {
21             _paths = paths;
22         }
23
24         public GameObject? Locate(string id)
25         {
26             if (AreYou(id))
27             {
28                 return this;
29             }
30
31             foreach (Path path in _paths)
32             {
33                 if (path.AreYou(id))
34                 {
35                     return path;
36                 }
37             }
38
39             return _inventory.Fetch(id);
```

```
40          }
41
42          public override string FullDescription
43          {
44              get
45              {
46                  return $"You are in the {Name}.\n" +
47                          $"{base.FullDescription}\n" +
48                          $"{PathList}\n" +
49                          $"In this location, you can see:\n{_inventory.ItemList}";
50              }
51          }
52
53          public string PathList
54          {
55              get
56              {
57                  if (_paths.Count == 0)
58                  {
59                      return "There are no paths to other locations";
60                  }
61
62                  string paths = "There are exits to ";
63
64                  for (int i = 0; i < _paths.Count; i++)
65                  {
66                      paths += _paths[i].Name;
67                      if (i < _paths.Count - 1)
68                      {
69                          paths += ", ";
70                      }
71                  }
72
73                  return paths;
74              }
75          }
76
77          public Inventory Inventory
78          {
79              get
80              {
81                  return _inventory;
82              }
83          }
84
85          public void AddPath(Path path)
86          {
87              _paths.Add(path);
88          }
89      }
90  }
```

Week_9/9.2/SwinAdventure/MoveCommand.cs

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

namespace SwinAdventure
{
    public class MoveCommand : Command
    {
        public MoveCommand() : base(new string[] { "move", "go" })
        {
        }

        public override string Execute(Player p, string[] text)
        {
            if (text.Length != 2 && text.Length != 3)
                return "I don't know how to move like that";

            if (text[0] != "move" && text[0] != "go")
                return "Error in move input";

            if (text.Length == 3 && text[1] != "to")
                return "Where do you want to go?";

            string destinationId = "";
            switch (text.Length)
            {
                case 2:
                    destinationId = text[1];
                    break;
                case 3:
                    destinationId = text[2];
                    break;
            }

            Path? path = p!.Locate(destinationId) as Path;
            if (path == null)
                return $"I can't find the path to {destinationId}";

            if (path.IsBlocked)
                return $"The path to {destinationId} is blocked";

            p.Location = path.Destination;
            return $"You have moved to {path.Destination.Name}";
        }
    }
}
```

Week_9/9.2/SwinAdventure/Path.cs

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

```
5
6  namespace SwinAdventure
7  {
8      public class Path : GameObject
9      {
10         private Location _source, _destination;
11         private bool _isBlocked;
12
13         public Path(string[] ids, string name, string desc, Location source, Location
   destination) : base(ids, name, desc)
14         {
15             _source = source;
16             _destination = destination;
17             _isBlocked = false;
18         }
19
20         public bool IsBlocked
21         {
22             get
23             {
24                 return _isBlocked;
25             }
26             set
27             {
28                 _isBlocked = value;
29             }
30         }
31
32         public Location Source
33         {
34             get
35             {
36                 return _source;
37             }
38         }
39
40         public Location Destination
41         {
42             get
43             {
44                 return _destination;
45             }
46         }
47     }
48 }
```

Week_9/9.2/SwinAdventure/Player.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace SwinAdventure
7  {
```

```
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        private Location? _location;

        public Player(string name, string desc) : base(new string[] { "me",
"inventory" }, name, desc)
        {
            _inventory = new Inventory();
        }

        public GameObject? Locate(string id)
        {
            if (AreYou(id))
                return this;

            GameObject? obj = _inventory.Fetch(id);
            if (obj != null)
                return obj;

            if (_location != null)
                return _location.Locate(id);

            return null;
        }

        public override string FullDescription
        {
            get
            {
                return $"You are {Name}, {base.FullDescription}\n" +
                       $"You are carrying:\n{_inventory.ItemList}";
            }
        }

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

        public Location? Location
        {
            get
            {
                return _location;
            }
            set
            {
                _location = value;
            }
        }
```

```
61      }
62  }
```

Week_9/9.2/SwinAdventure/Program.cs

```
1   namespace SwinAdventure
2   {
3       class Program
4       {
5           static void Main()
6           {
7               string playerName, playerDesc;
8               while (true)
9               {
10                  Console.Write("Enter player name: ");
11                  playerName = Console.ReadLine() ?? string.Empty;
12                  Console.Write("Enter player description: ");
13                  playerDesc = Console.ReadLine() ?? string.Empty;
14                  if (string.IsNullOrEmpty(playerName) ||
    string.IsNullOrEmpty(playerDesc))
15                  {
16                      Console.WriteLine("Player name and description cannot be
    empty.");
17                  }
18                  else
19                  {
20                      break;
21                  }
22              }
23              Player player = new Player(playerName, playerDesc);
24
25              // Create items and put them in the player's inventory
26              Item item1 = new Item(new string[] { "shovel" }, "a shovel", "a wooden
    shovel");
27              Item item2 = new Item(new string[] { "sword" }, "a sword", "a steel
    sword");
28              player.Inventory.Put(item1);
29              player.Inventory.Put(item2);
30
31              // Create a bag and put it in the player's inventory
32              Bag bag = new Bag(new string[] { "bag" }, "a bag", "a leather bag");
33              player.Inventory.Put(bag);
34
35              // Create items and put them in the bag's inventory
36              Item item3 = new Item(new string[] { "coin" }, "a coin", "a shiny coin");
37              bag.Inventory.Put(item3);
38
39              // Create location and put some items in its inventory
40              Location location = new Location("forest", "A dark forest with tall
    trees");
41              Item item4 = new Item(new string[] { "rock" }, "a rock", "a big rock");
42              Item item5 = new Item(new string[] { "flower" }, "a flower", "a red
    flower");
43              location.Inventory.Put(item4);
44              location.Inventory.Put(item5);
```

```
45
46              // Create another location and a path between the two locations
47              Location location2 = new Location("cave", "A dark cave with bats");
48              Path path = new Path(new string[] { "north" }, "north", "a path from
    forest to cave", location, location2);
49              Path path2 = new Path(new string[] { "south" }, "south", "a path from
    cave to forest", location2, location);
50              location.AddPath(path);
51              location2.AddPath(path2);
52
53              // Set player's location
54              player.Location = location;
55
56              LookCommand look = new LookCommand();
57              MoveCommand move = new MoveCommand();
58
59              while (true)
60              {
61                  Console.Write("> ");
62                  string command = Console.ReadLine() ?? string.Empty;
63
64                  if (string.IsNullOrEmpty(command))
65                      continue;
66                  if (command == "quit")
67                      break;
68
69                  string response;
70                  if (command.StartsWith("move") || command.StartsWith("go"))
71                  {
72                      response = move.Execute(player, command.Split(" "));
73                      Console.WriteLine(response);
74                      Console.WriteLine();
75                      continue;
76                  }
77
78                  response = look.Execute(player, command.Split(" "));
79                  Console.WriteLine(response);
80                  Console.WriteLine();
81              }
82          }
83      }
84  }
```

Week_9/9.2/TestPath/TestMove.cs

```
1   using NUnit.Framework;
2   using SwinAdventure;
3   using Path = SwinAdventure.Path;
4
5   namespace TestMoveCommand
6   {
7       public class TestMoveCommand
8       {
9           private Player _player;
10          private Location _location1;
```

```
11          private Location _location2;
12          private Path _path;
13          private MoveCommand _moveCommand;
14
15          [SetUp]
16          public void Setup()
17          {
18              _player = new Player("Minh An", "104844794");
19              _location1 = new Location("forest", "A dark forest with tall trees");
20              _location2 = new Location("cave", "A dark cave with bats");
21              _path = new Path(new string[] { "north" }, "north", "a path from forest
    to cave", _location1, _location2);
22              _location1.AddPath(_path);
23              _player.Location = _location1;
24              _moveCommand = new MoveCommand();
25          }
26
27          [Test]
28          public void TestMoveToBlockedPath()
29          {
30              _path.IsBlocked = true;
31              Assert.That(_moveCommand.Execute(_player, new string[] { "go", "north"
    }), Is.EqualTo("The path to north is blocked"));
32          }
33
34          [Test]
35          public void TestMoveToNonExistentPath()
36          {
37              Assert.That(_moveCommand.Execute(_player, new string[] { "go", "south"
    }), Is.EqualTo("I can't find the path to south"));
38          }
39
40          [Test]
41          public void TestMoveToDestination()
42          {
43              Assert.That(_moveCommand.Execute(_player, new string[] { "go", "north"
    }), Is.EqualTo("You have moved to cave"));
44          }
45
46          [Test]
47          public void TestInvalidMoveCommand()
48          {
49              Assert.That(_moveCommand.Execute(_player, new string[] { "move", "north"
    }), Is.EqualTo("You have moved to cave"));
50              Assert.That(_moveCommand.Execute(_player, new string[] { "go", "north",
    "to" }), Is.EqualTo("Where do you want to go?"));
51              Assert.That(_moveCommand.Execute(_player, new string[] { "go", "north",
    "to", "cave" }), Is.EqualTo("I don't know how to move like that"));
52              Assert.That(_moveCommand.Execute(_player, new string[] { "go", "north",
    "to", "cave", "now" }), Is.EqualTo("I don't know how to move like that"));
53          }
54      }
55  }
```

Week_9/9.2/TestPath/TestPath.cs

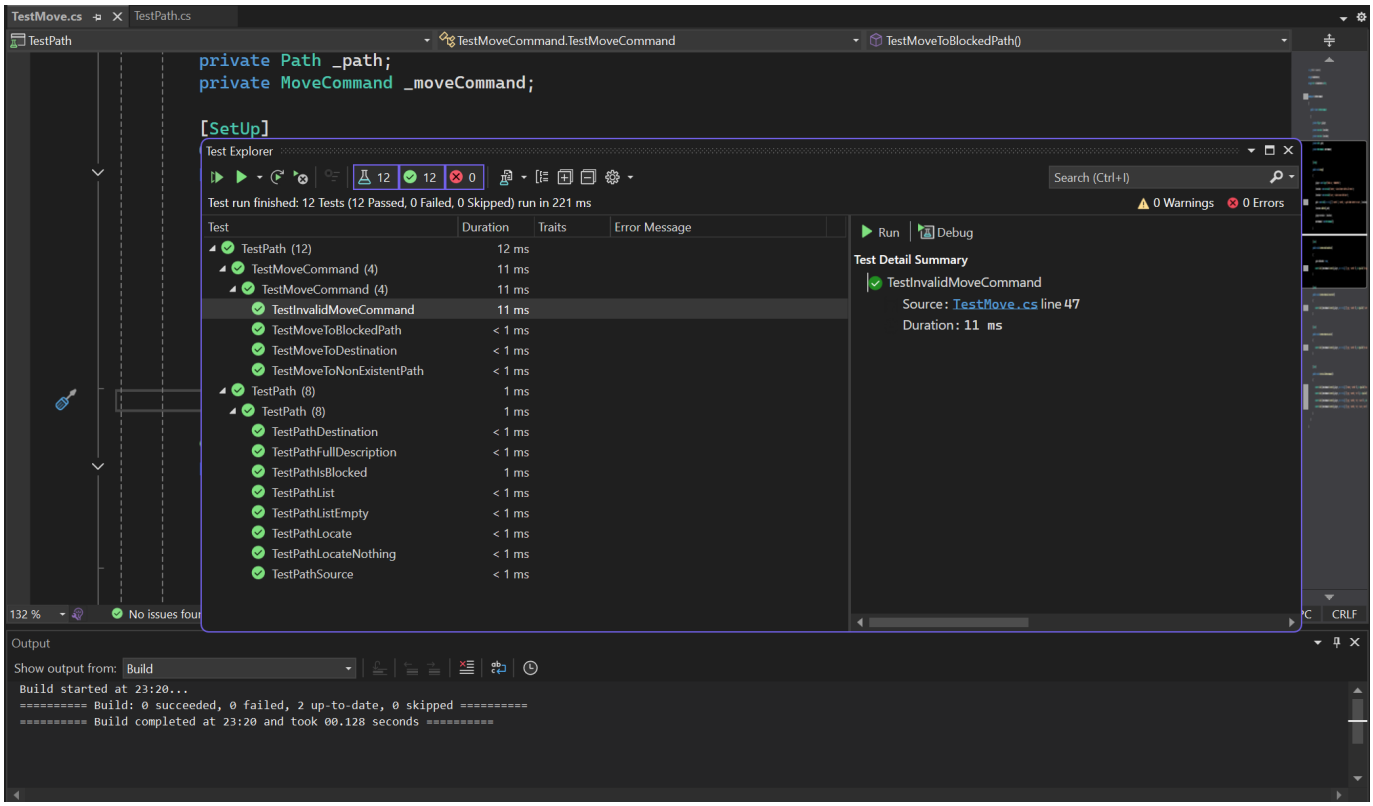
```
using NUnit.Framework;
using SwinAdventure;
using Path = SwinAdventure.Path;

namespace TestPath
{
    public class TestPath
    {
        private Player _player;
        private Location _location1;
        private Location _location2;
        private Path _path;

        [SetUp]
        public void Setup()
        {
            _player = new Player("Minh An", "104844794");
            _location1 = new Location("forest", "A dark forest with tall trees");
            _location2 = new Location("cave", "A dark cave with bats");
            _path = new Path(new string[] { "north" }, "north", "a path from forest to cave", _location1, _location2);
            _location1.AddPath(_path);
            _player.Location = _location1;
        }

        [Test]
        public void TestPathIsBlocked()
        {
            Assert.That(_path.IsBlocked, Is.False);
            _path.IsBlocked = true;
            Assert.That(_path.IsBlocked, Is.True);
        }

        [Test]
        public void TestPathSource()
        {
            Assert.That(_path.Source, Is.EqualTo(_location1));
        }

        [Test]
        public void TestPathDestination()
        {
            Assert.That(_path.Destination, Is.EqualTo(_location2));
        }

        [Test]
        public void TestPathLocate()
        {
            Assert.That(_player.Locate("north"), Is.EqualTo(_path));
        }

        [Test]
        public void TestPathLocateNothing()
        {
```

```
54              Assert.That(_player.Locate("south"), Is.Null);
55          }
56
57          [Test]
58          public void TestPathFullDescription()
59          {
60              Assert.That(_path.FullDescription, Is.EqualTo("a path from forest to
    cave"));
61          }
62
63          [Test]
64          public void TestPathList()
65          {
66              Assert.That(_location1.PathList, Is.EqualTo("There are exits to north"));
67          }
68
69          [Test]
70          public void TestPathListEmpty()
71          {
72              Location location = new Location("desert", "A hot desert with sand
    dunes");
73              Assert.That(location.PathList, Is.EqualTo("There are no paths to other
    locations"));
74          }
75      }
76  }
```

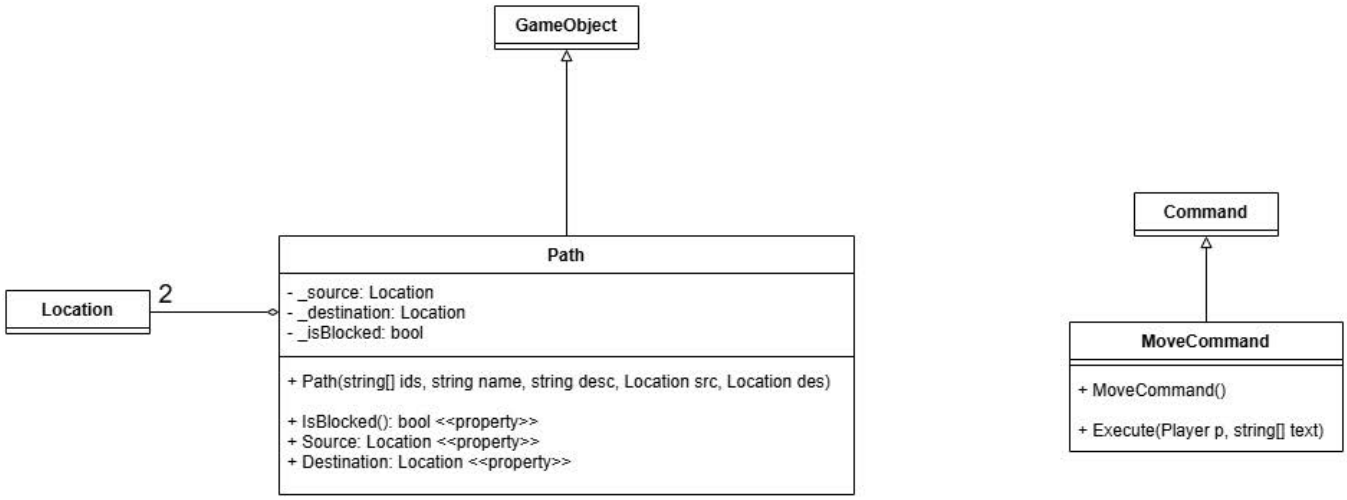
Screenshot of program running:

```
PS C:\Users\Admin\Desktop\COS20007-OOP> cd "c:\Users\Admin\Desktop\COS20007-OOP\Week_9\9.2\SwinAdventure\" ; if ($?) { dotnet run }
Enter player name: Minh An
Enter player description: 104844794
> look
You are in the forest.
A dark forest with tall trees
There are exits to north
In this location, you can see:
        a rock (rock)
        a flower (flower)

> go north
You have moved to cave

> look
You are in the cave.
A dark cave with bats
There are no paths to other locations
In this location, you can see:
        Nothing here!

>
```

Screenshot of test case passed



UML Diagram



Sequence Diagram