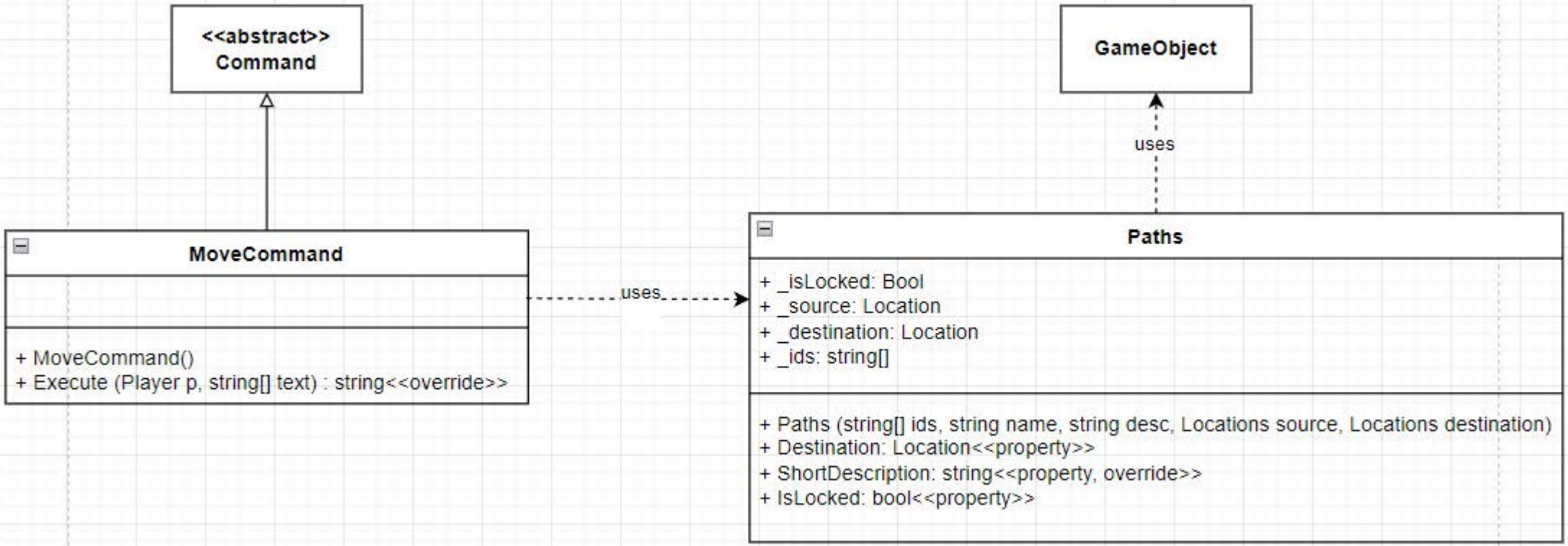
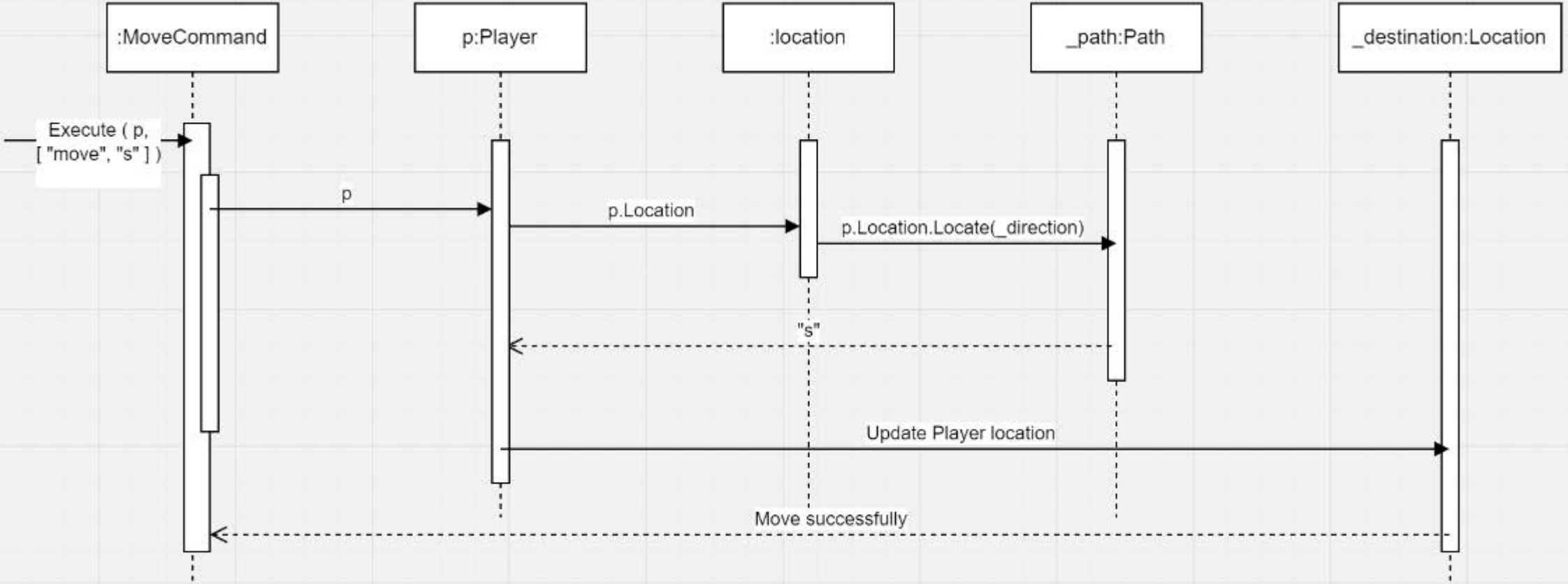

<<abstract>>
Command
GameObject
uses
MoveCommand
+ MoveCommand()
+ Execute (Player p, string[] text) : string<<override>>
uses
Paths
+ _isLocked: Bool
+ _source: Location
+ _destination: Location
+ _ids: string[]
+ Paths (string[] ids, string name, string desc, Locations source, Locations destination)
+ Destination: Location<<property>>
+ ShortDescription: string<<property, override>>
+ IsLocked: bool<<property>>


:MoveCommand
p:Player
:location
_path:Path
_destination:Location
Execute (p, ["move", "s"])
p
p.Location
p.Location.Locate(_direction)
"s"
Update Player location
Move successfully

```
1  using System;
2
3  namespace SwinAdventure
4  {
5      public class Paths : GameObject
6      {
7          bool _isLocked;
8          Locations _source, _destination;
9          string[] _ids;
10
11         public Paths(string[] ids, string name, string desc, Locations
             source, Locations destination) : base(ids, name, desc)
12         {
13             _isLocked = false;
14             _source = source;
15             _destination = destination;
16             _ids = ids;
17
18             AddIdentifier("path");
19             foreach (string s in name.Split(" "))
20             {
21                 AddIdentifier(s);
22             }
23         }
24
25         public Locations Destination
26         {
27             get
28             {
29                 return _destination;
30             }
31         }
32
33         public override string ShortDescription
34         {
35             get
36             {
37                 return "- " + _ids[0] + ": " + _destination.Name + "\n";
38             }
39         }
40
41         public bool IsLocked
42         {
43             get
44             {
45                 return _isLocked;
46             }
47             set
48             {
```

```
49                  _isLocked = value;
50              }
51          }
52      }
53  }
54
```

```
using System;

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

        public override string Execute(Player p, string[] text)
        {
            string error = "I don't know how to move that.";
            string _direction;
            switch (text.Length)
            {
                case 1:
                    return "Move where?";
                case 2:
                    _direction = text[1].ToLower();
                    break;
                case 3:
                    _direction = text[2].ToLower();
                    break;
                default:
                    return error;
            }

            GameObject _path = p.Location.Locate(_direction);
            if (_path == null)
            {
                return error;
            }
            if (_path is Paths)
            {
                Paths _p = (Paths)_path;
                if (_p.IsLocked)
                {
                    return "The path is blocked.";
                }
                p.Location = _p.Destination;
                return "You moved to " + p.Location.Name + ".\n";
            }
            return error;
        }


    }
}
```

```
1  namespace SwinAdventure
2  {
3      internal class Program
4      {
5          static string[] CommandExe(string input)
6          {
7              return input.Split(' ');
8          }
9
10         static void Main(string[] args)
11         {
12             Console.WriteLine("Enter player's name:");
13             string playerName = Console.ReadLine();
14
15             Console.WriteLine("Enter player's description:");
16             string playerDescription = Console.ReadLine();
17
18             Player player = new Player(playerName, playerDescription);
19
20             Locations home = new Locations("Home", playerName + "'s
                 home.");
21             Locations forest = new Locations("Forest", "A dense forest.");
22             Locations cave = new Locations("Cave", "A dark cave.");
23             Paths home2forest = new Paths(new string[] { "north" }, "Path
                 1", "A path from home to the forest.", home, forest);
24             Paths forest2home = new Paths(new string[] { "south" }, "Path
                 2", "A path from the forest to the cave.", forest, cave);
25             home.AddPath(home2forest);
26             forest.AddPath(forest2home);
27
28             Console.WriteLine(home.PathsList);
29
30             player.Location = home;
31
32             Item BronzeSword = new Item(new string[] { "sword", "weapon" },
                  "Bronze Sword", "A simple bronze sword.");
33             Item BronzeAxe = new Item(new string[] { "axe", "weapon" },
                 "Bronze Axe", "A simple bronze axe.");
34
35             player.Inventory.Put(BronzeSword);
36             player.Inventory.Put(BronzeAxe);
37
38             Bag bag = new Bag(new string[] { "bag", "inventory" }, "Bag",
                 "A simple bag.");
39             player.Inventory.Put(bag);
40
41             Item gem = new Item(new string[] { "gem", "jewel" }, "Gem", "A
                 shiny gem.");
42             bag.Inventory.Put(gem);
```

```
43
44              while (true)
45              {
46                  Console.WriteLine("Enter a command: ");
47                  Command command =  new CommandProcessor();
48                  string _input = Console.ReadLine();
49                  string[] _temp = _input.Split(" ");
50
51                  if (_input.ToLower() == "quit")
52                  {
53                      break;
54                  }
55                  else
56                  {
57                      Console.WriteLine(command.Execute(player, _temp));
58                  }
59              }
60          }
61      }
62  }
63
```

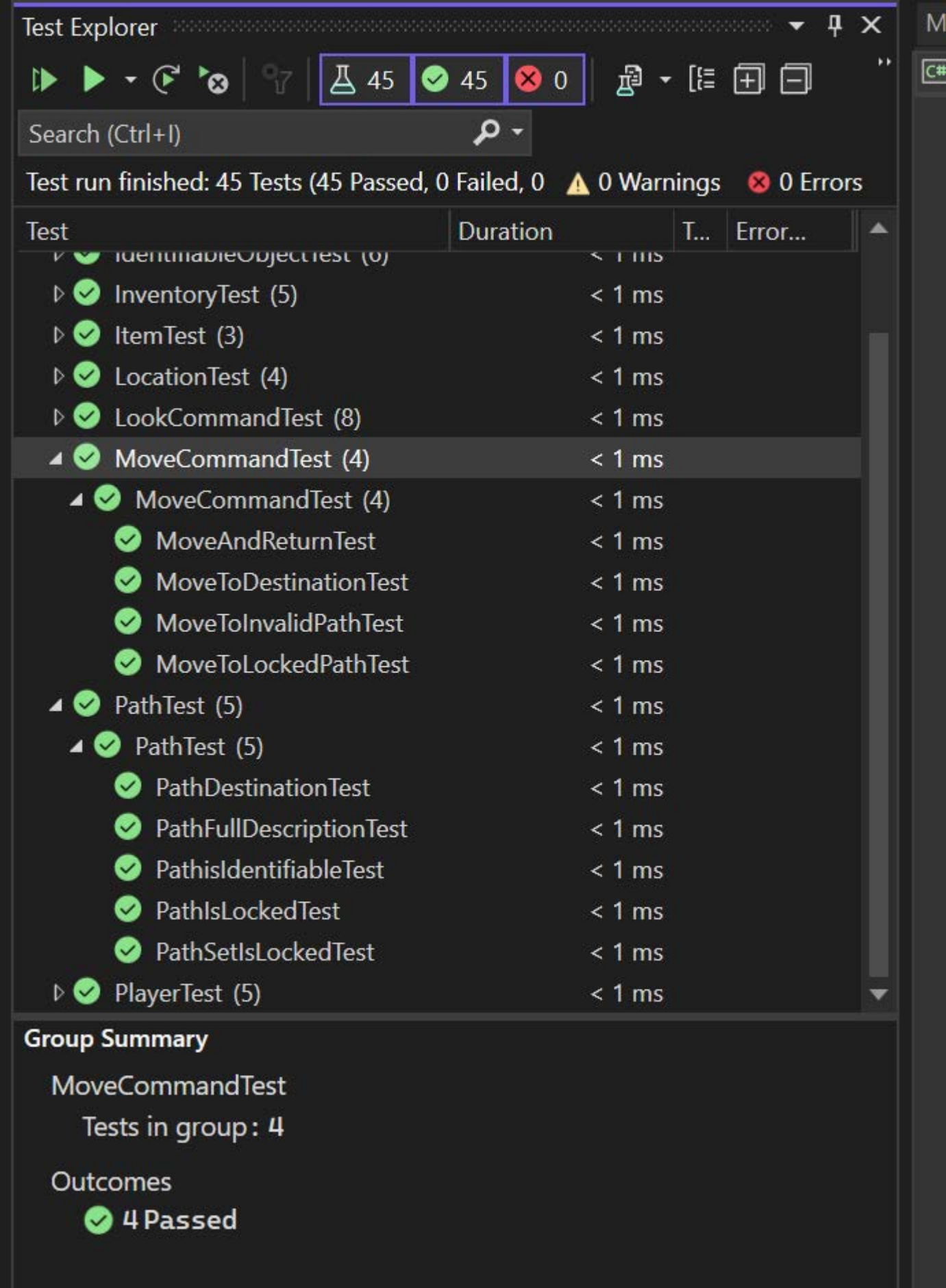

D:\workspace\COS20007\COS
Enter player's name:
jfjwoef
Enter player's description:
njdnjkvnd
You can move to:
- north: Forest
Enter a command:
move north
You moved to Forest.
Enter a command:

```
1  using SwinAdventure;
2
3  namespace PathTest
4  {
5      public class PathTest
6      {
7          Paths _path;
8          Locations _roomA;
9          Locations _roomB;
10
11         [SetUp]
12         public void Setup()
13         {
14             _roomA = new Locations("Room A", "A room.");
15             _roomB = new Locations("Room B", "A room.");
16
17             _path = new Paths(new string[] { "north" }, "Path", "A path
                 from the source to the destination.", _roomA, _roomB);
18         }
19
20         [Test]
21         public void PathisIdentifiableTest()
22         {
23             Assert.IsTrue(_path.AreYou("path"));
24         }
25
26         [Test]
27         public void PathFullDescriptionTest()
28         {
29             string desc = _path.FullDescription;
30
31             Assert.AreEqual("Path (north): A path from the source to the
                 destination.", desc);
32         }
33
34         [Test]
35         public void PathIsLockedTest()
36         {
37             Assert.IsFalse(_path.IsLocked);
38         }
39
40         [Test]
41         public void PathSetIsLockedTest()
42         {
43             _path.IsLocked = true;
44
45             Assert.IsTrue(_path.IsLocked);
46         }
47
```

```
48          [Test]
49          public void PathDestinationTest()
50          {
51              Assert.AreEqual("Room B", _path.Destination.Name);
52          }
53      }
54  }
55
```

```
1  using SwinAdventure;
2
3  namespace MoveCommandTest
4  {
5      public class MoveCommandTest
6      {
7          Command _moveCommand;
8          Player _player;
9          Locations _location, _destination;
10         Paths _path, _returnPath;
11
12         [SetUp]
13         public void Setup()
14         {
15             _moveCommand = new MoveCommand();
16             _player = new Player("Sen", "luv ur mom");
17             _location = new Locations("Location", "A location");
18             _destination = new Locations("Destination", "A destination");
19             _path = new Paths(new string[] { "north" }, "Path", "A path
                 from the source to the destination.", _location,
                 _destination);
20             _returnPath = new Paths(new string[] { "south" }, "Path", "A
                 path from the source to the destination.", _destination,
                 _location);
21             _location.AddPath(_path);
22             _destination.AddPath(_returnPath);
23             _player.Location = _location;
24         }
25
26         [Test]
27         public void MoveToDestinationTest()
28         {
29             string[] text = new string[] { "move", "to", "north" };
30             Assert.AreEqual("You moved to Destination.\n",
                   _moveCommand.Execute(_player, text));
31         }
32
33         [Test]
34         public void MoveToLockedPathTest()
35         {
36             _path.IsLocked = true;
37             string[] text = new string[] { "move", "to", "north" };
38             Assert.AreEqual("The path is blocked.", _moveCommand.Execute
                   (_player, text));
39         }
40
41         [Test]
42         public void MoveToInvalidPathTest()
43         {
```

```
44              string[] text = new string[] { "move", "to", "east" };
45              Assert.AreEqual("I don't know how to move that.",
                  _moveCommand.Execute(_player, text));
46          }
47
48          [Test]
49          public void MoveAndReturnTest()
50          {
51              string[] text = new string[] { "move", "to", "north" };
52              _moveCommand.Execute(_player, text);
53              text = new string[] { "move", "to", "south" };
54              Assert.AreEqual("You moved to Location.\n",
                  _moveCommand.Execute(_player, text));
55          }
56      }
57  }
58
```


Test Explorer
45
45
0
Search (Ctrl+I)
Test run finished: 45 Tests (45 Passed, 0 Failed, 0
0 Warnings
0 Errors
Test
Duration
T...
Error...
InventoryTest (5)
< 1 ms
ItemTest (3)
< 1 ms
LocationTest (4)
< 1 ms
LookCommandTest (8)
< 1 ms
MoveCommandTest (4)
< 1 ms
MoveCommandTest (4)
< 1 ms
MoveAndReturnTest
< 1 ms
MoveToDestinationTest
< 1 ms
MoveToInvalidPathTest
< 1 ms
MoveToLockedPathTest
< 1 ms
PathTest (5)
< 1 ms
PathTest (5)
< 1 ms
PathDestinationTest
< 1 ms
PathFullDescriptionTest
< 1 ms
PathIsIdentifiableTest
< 1 ms
PathIsLockedTest
< 1 ms
PathSetIsLockedTest
< 1 ms
PlayerTest (5)
< 1 ms
Group Summary
MoveCommandTest
Tests in group: 4
Outcomes
4 Passed