Swinburne University of Technology

COS20007 Object Oriented Programming

9.2C - Case Study - Iteration 7 - Paths

PDF generated at 00:07 on Monday 8th May, 2023

```
using System;

namespace SwinAdventure
{
    public class Path : GameObject
    {
        private Location _destination;
        private bool _isLocked;
        public Path(string[] ids, string name, string desc, Location destination) :
    base(ids, name, desc)
        {
            _destination = destination;
        }

        public Location Destination
        {
            get
            {
                return _destination;
            }
        }

        public override string FullDescription
        {
            get
            {
                return $"{Destination.Name} is in the {Name}";
            }
        }

        public bool IsLocked
        {
            get
            {
                return _isLocked;
            }
            set
            {
                _isLocked = value;
            }
        }
    }
}

```

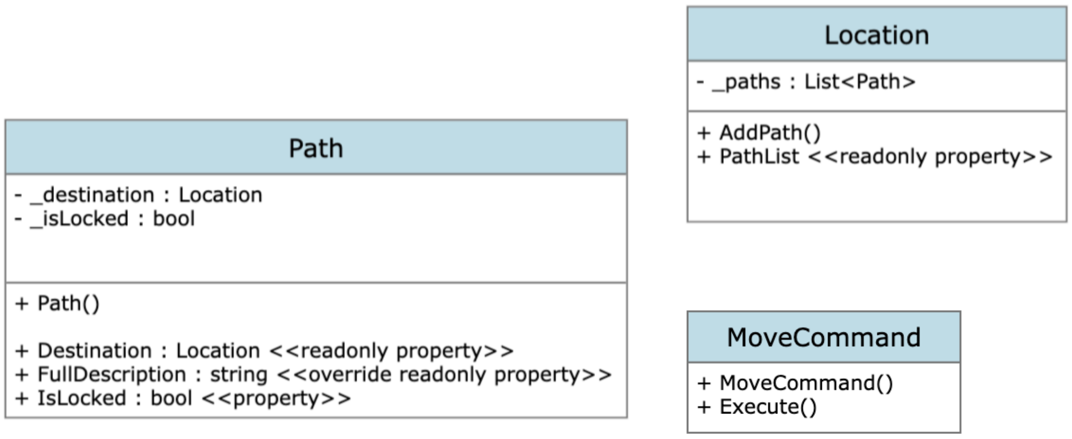
```
using Path = SwinAdventure.Path;

namespace SwinAdventureTest
{
    [TestFixture]
    public class TestPath
    {
        Path path;
        Location jungle;

        [SetUp]
        public void Setup()
        {
            jungle = new Location("a jungle", "This is a scary  jungle");
            path = new Path(new string[] { "south", "s" }, "south", "this is south",
    ↪   jungle);
        }

        [Test]
        public void TestPathIsIdentifiable()
        {
            Assert.That(path.AreYou("south"), Is.True);
            Assert.That(path.AreYou("s"), Is.True);
            Assert.That(path.AreYou("north"), Is.False);
        }

        [Test]
        public void TestFullDesc()
        {
            string actual = path.FullDescription;
            string expected = "a jungle is in the south";
            Assert.That(actual, Is.EqualTo(expected));
        }
    }
}
```

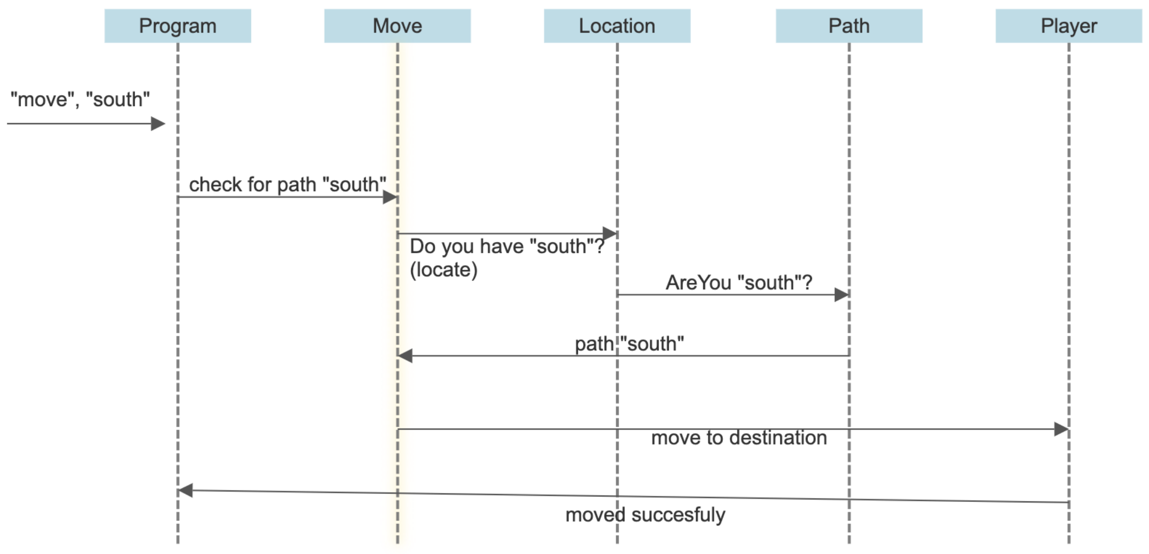
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class Location : GameObject, IHaveInventory
    {
        //local variables
        private Inventory _inventory;
        private List<Path> _paths;

        //constructor
        public Location(string name, string desc) : base(new string[] { "house",
↪   "here" }, name, desc)
        {
            _inventory = new Inventory();
            _paths = new List<Path>();
        }

        //methods
        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }

            foreach (Path p in _paths)
            {
                if (p.AreYou(id))
                {
                    return p;
                }
            }

            return _inventory.Fetch(id);
        }

        public void AddPath(Path path)
        {
            _paths.Add(path);
        }

        //properties
        public string PathList
        {
            get
            {
                string pathList = "";

                if (_paths.Count > 0)
                {
                    for (int i = 0; i < _paths.Count; i++)
                    {
                        if (_paths.Count == 1)
                        {
                            pathList += $"{_paths[i].Name}";
                        }
                        else if (i == _paths.Count - 1)
                        {
                            pathList += $"and {_paths[i].Name}";
                        }
                        else
                        {
                            if (_paths.Count == 1)
                            {
                                pathList += $"{_paths[i].Name}, ";
                            }
                        }
                    }
                    return $"You can go to the {pathList}.";
                }
                else
                {
                    return "There are no exits.";
                }

            }
        }

        public override string FullDescription
        {
            get
            {
                return $"You are in {Name}\n{Description}\n{PathList}\nHere you can see:\n{_inventory.ItemList}";
            }
        }

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }
    }
}
```

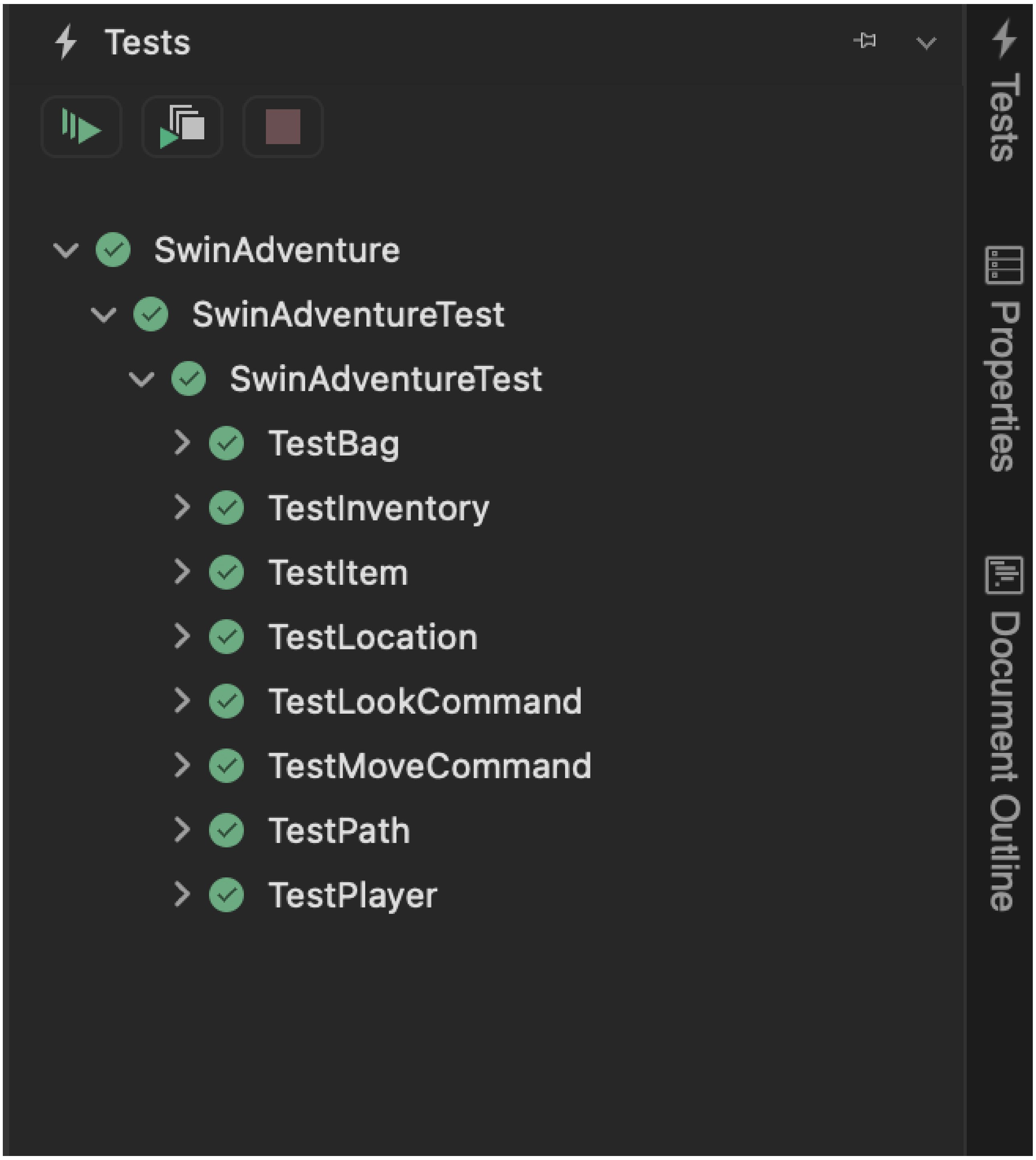
```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;



namespace SwinAdventureTest
{
    public class TestLocation
    {
        Location location;
        Location destination;
        SwinAdventure.Path path;
        Player player;
        Item knife;

        [SetUp]
        public void Setup()
        {
            location = new Location("a jungle", "This is a jungle");
            destination = new Location("a tower", "This is a tilted tower");
            path = new SwinAdventure.Path(new string[] { "south" }, "south", "this
    ↪ is south", destination);
            player = new Player("bob", "the builder");
            knife = new Item(new string[] { "Knife" }, "a sharp knife", "This is a
    ↪ sharp knife");

            location.Inventory.Put(knife);
            player.Location = location;
        }

        [Test]
        public void TestIdentifyLocation()
        {
            Assert.That(location.Locate("house"), Is.SameAs(location));
        }

        [Test]
        public void TestIdentifyLocationInventory()
        {
            Assert.That(location.Locate("knife"), Is.SameAs(knife));
        }

        [Test]
        public void TestIdentifyPlayerLocateItem()
        {
            Assert.That(player.Locate("knife"), Is.SameAs(knife));
        }

        [Test]
        public void TestIdentifyPath()
        {
            location.AddPath(path);
            Assert.That(player.Locate("south"), Is.SameAs(path));
        }

        [Test]
        public void TestLocationFullDesc()
        {
            location.AddPath(path);
            string expected = "You are in a jungle\nThis is a jungle\nYou can go to the south.\nHere you can see:\na sharp knife (knife)\n";
            Assert.That(location.FullDescription, Is.EqualTo(expected));
        }
    }
}

```

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

        }

        public override string Execute(Player p, string[] text)
        {
            string moveTo;
            string[] moveIds = new string[] { "move", "go", "head", "leave" };

            if (text.Length >= 3)
            {
                return "I don't know how to move like that";
            }
            else if (!moveIds.Contains(text[0]))
            {
                return "Error in move input";
            }
            else if (text.Length == 1)
            {
                return "Where do you want to move?";
            }
            else
            {

                moveTo = text[1];

                if (p.Locate(moveTo) is Path path)
                {
                    p.Move(path);

                    return $"You went {path.Name}\nYou have arrived in
↪   {path.Destination.Name}";
                }
                else
                {
                    return "Error in direction!";
                }
            }


        }
    }
}
```

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

namespace SwinAdventureTest
{
    public class TestMoveCommand
    {
        MoveCommand move;
        Location location;
        Location destination;
        Path path;
        Player player;

        [SetUp]
        public void Setup()
        {
            move = new MoveCommand();
            location = new Location("a jungle", "This is a creepy jungle");
            destination = new Location("a tower", "This is a tilted tower");
            path = new Path(new string[] { "south" }, "south", "this is south",
    ↪  destination);
            player = new Player("bob", "the builder");

            player.Location = location;
            location.AddPath(path);
        }

        [Test]
        public void TestMove()
        {
            Assert.That(player.Location, Is.SameAs(location));
            move.Execute(player, new string[] { "move", "south" });
            Assert.That(player.Location, Is.SameAs(destination));
        }

        [Test]
        public void TestInvalidMove()
        {
            Assert.That(player.Location, Is.SameAs(location));
            move.Execute(player, new string[] { "move", "east" });
            Assert.That(player.Location, Is.SameAs(location));
        }

        [Test]
        public void TestSuccessfulMoveOutput()
        {
            string actual = move.Execute(player, new string[] { "move", "south" });
            string expected = "You went south\nYou have arrived in a tower";

            Assert.That(actual, Is.EqualTo(expected));
        }

        //errors

        [Test]
        public void TestIncorrectLength()
        {
            string actual = move.Execute(player, new string[] { "move", "to", "north" });
            string expected = "I don't know how to move like that";

            Assert.That(actual, Is.EqualTo(expected));
        }

        [Test]
        public void TestInvalidCommand()
        {
            string actual = move.Execute(player, new string[] { "run", "south" });
            string expected = "Error in move input";

            Assert.That(actual, Is.EqualTo(expected));

        }

        [Test]
        public void TestOnlyMoveInput()
        {
            string actual = move.Execute(player, new string[] { "move" });
            string expected = "Where do you want to move?";

            Assert.That(actual, Is.EqualTo(expected));
        }

        [Test]
        public void TestInvalidDirection()
        {
            string actual = move.Execute(player, new string[] { "move", "east" });
            string expected = "Error in direction!";

            Assert.That(actual, Is.EqualTo(expected));
        }
    }
}
```

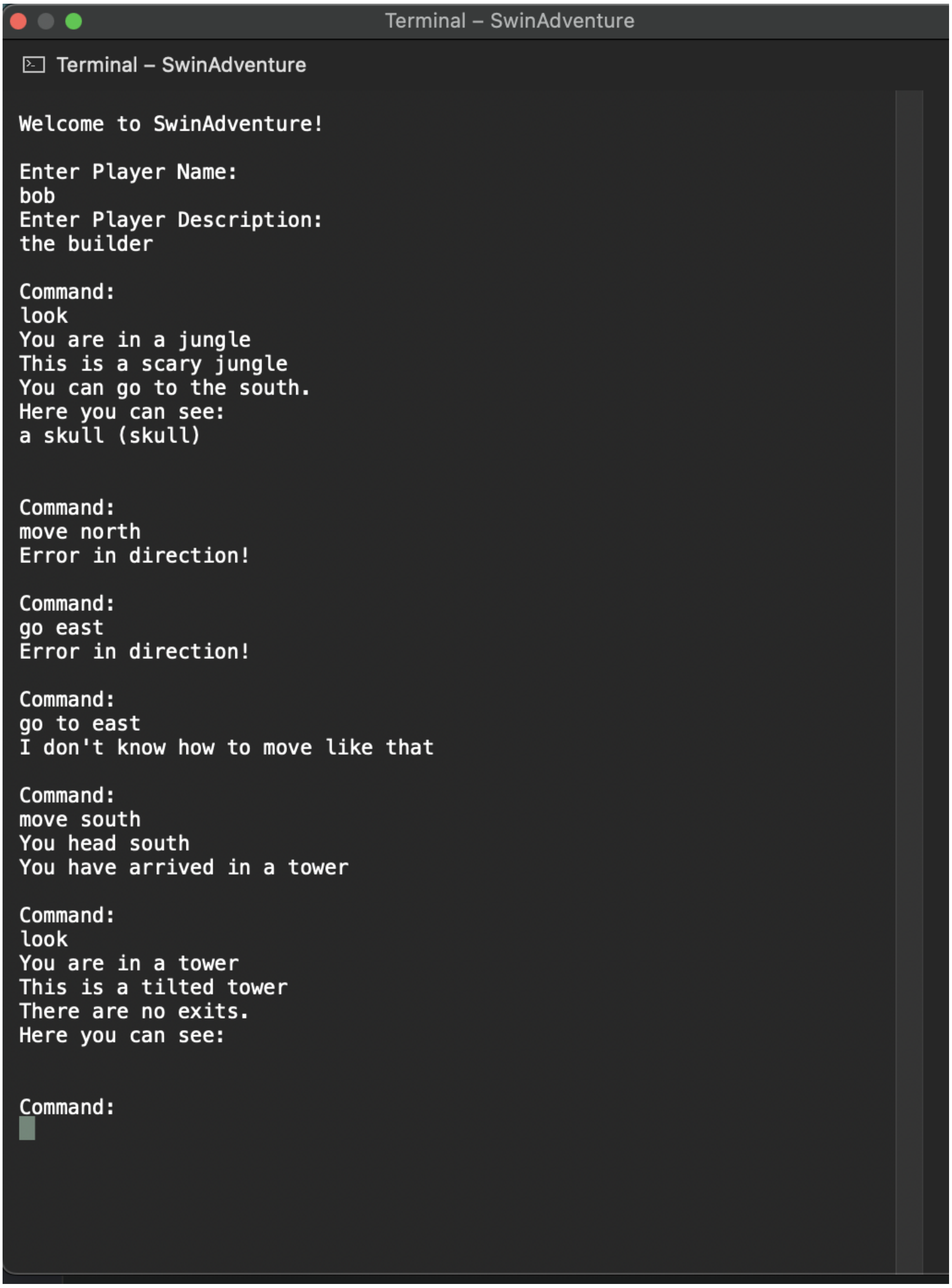

Location
- _paths : List<Path>
+ AddPath()
+ PathList <<readonly property>>
Path
- _destination : Location
- _isLocked : bool
+ Path()
+ Destination : Location <<readonly property>>
+ FullDescription : string <<override readonly property>>
+ IsLocked : bool <<property>>
MoveCommand
+ MoveCommand()
+ Execute()


Program
Move
Location
Path
Player
"move", "south"
check for path "south"
Do you have "south"? (locate)
AreYou "south"?
path "south"
move to destination
moved succesfuly



Tests

- ✓ SwinAdventure
  - ✓ SwinAdventureTest
    - ✓ SwinAdventureTest
      - ✓ TestBag
      - ✓ TestInventory
      - ✓ TestItem
      - ✓ TestLocation
      - ✓ TestLookCommand
      - ✓ TestMoveCommand
      - ✓ TestPath
      - ✓ TestPlayer



```
Welcome to SwinAdventure!

Enter Player Name:
bob
Enter Player Description:
the builder

Command:
look
You are in a jungle
This is a scary jungle
You can go to the south.
Here you can see:
a skull (skull)


Command:
move north
Error in direction!

Command:
go east
Error in direction!

Command:
go to east
I don't know how to move like that

Command:
move south
You head south
You have arrived in a tower

Command:
look
You are in a tower
This is a tilted tower
There are no exits.
Here you can see:


Command:
```