Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 7 - Paths

PDF generated at 10:50 on Friday 3rd November, 2023

```csharp
using System;

namespace CaseStudy
{
    public class Path : GameObject
    {
        private Location _destination;

        public Path(string[] idents, string name, string desc, Location destination)
            : base(idents, name, desc)
        {
            _destination = destination;
        }

        public string Move(Player p)
        {
            if (Destination != null)
            {
                p.Location = Destination;
                return $"You have moved to {Destination.Name}";
            }
            return "The path leads to nowhere.";
        }

        public Location Destination
        {
            get { return _destination; }
        }
    }
}
```

```
using System;
using CaseStudy;
using NUnit.Framework;

namespace CaseStudyTest
{
    public class PathTest
    {
        private Player _p;
        private Location _loca;
        private Location _locb;
        private CaseStudy.Path _path;
        private MoveCommand _command;

        [SetUp]
        public void Setup()
        {
            _p = new Player("Duc", "This is Vu Duc Tran");
            _loca = new Location(new string[] { "enbuilding" },"classroom",
    "Swinburne University");
            _locb = new Location(new string[] { "enbuilding" },"enbuilding",
    "ENbuilding");
            _path = new CaseStudy.Path(new string[] { "north" }, "ENbuilding",
    "classroom to ENbuilding", _locb);
            _command = new MoveCommand(new string[] { "move" });

            _p.Location = _loca;
            _loca.AddPath(_path);

        }

        [Test]
        public void MovePlayer()
        {
            _command.Execute(_p, new string[] { "Move", "north" }); //Execute method
    will call Move method from Path object
            Assert.That(_p.Location, Is.EqualTo(_locb), "Test Identify Location");
        }

        [Test]
        public void GetPathFromLocation()
        {
            Assert.That(_p.Location.GetPath("north"), Is.EqualTo(_path), "Test Get A
    Path From A Location Given One Of The Path's Identifiers");
        }
    }
}
```

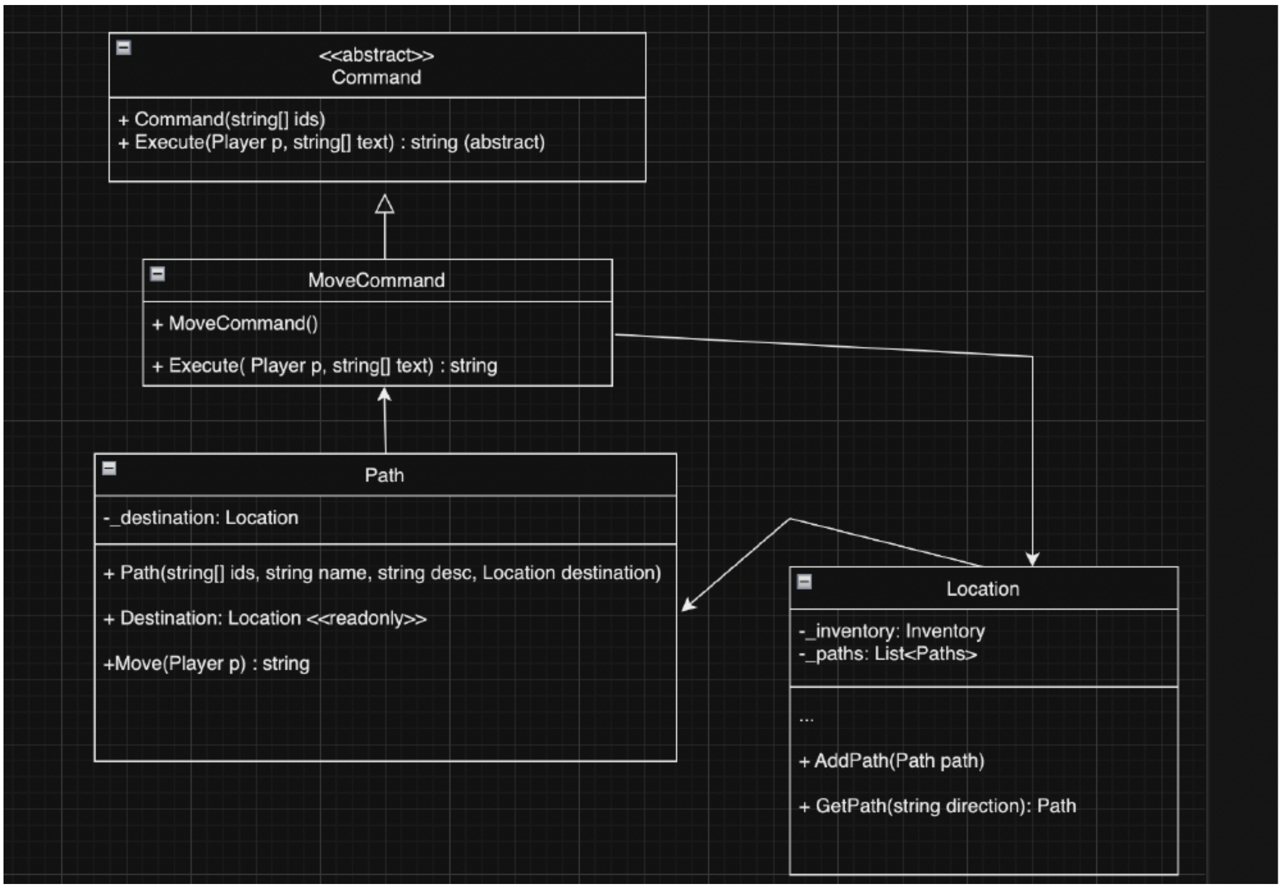
```csharp

namespace CaseStudy
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _container;
        private List<Path> _pathList = new List<Path>() { };
        public Location(string[] idents, string name, string desc) : base(idents,
↪   name, desc)
        {
            _container = new();
        }

        public GameObject Locate(string id)
        {
            if (this.AreYou(id)) return this;
            GameObject locateResult = _container.Fetch(id);
            return locateResult;
        }

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

        public Path? GetPath(string direction)
        {
            foreach (Path path in _pathList)
            {
                if (path.AreYou(direction))
                {
                    return path;
                }
            }
            return null;
        }

        public override string FullDescription
        {
            get
            {
                return "You are at: " + base.ShortDescription + "\nItems at this
↪   location:\n" + _container.ItemList;
            }
        }

        public Inventory Container
        {
            get
            {
                return _container;
            }
        }
    }
}
```

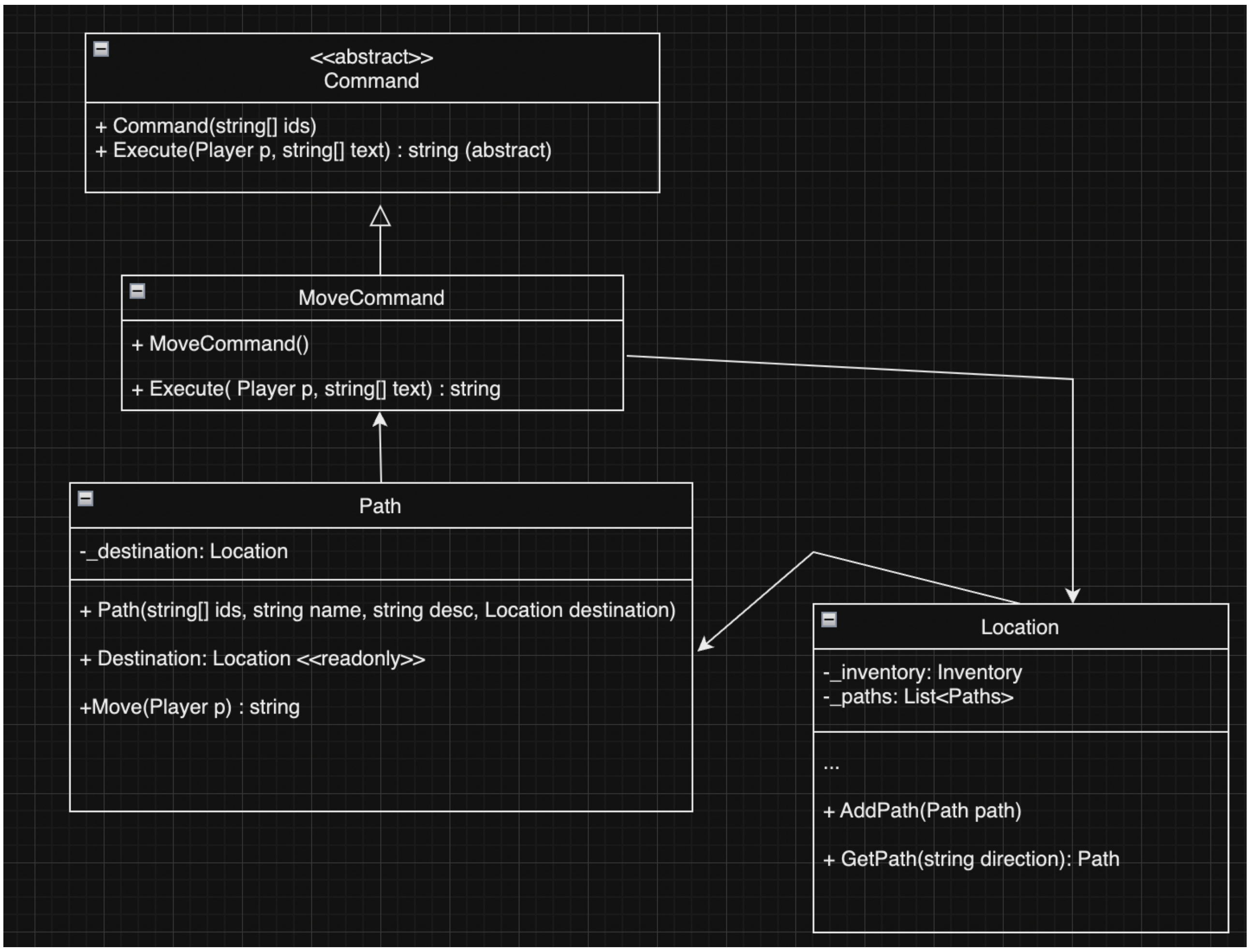
```
using CaseStudy;

namespace CaseStudyTest
{
    public class LocationTest
    {
        private Player _p;
        private Location _loca;
        private Item _sword;

        [SetUp]
        public void Setup()
        {
            _p = new Player("Tran", "This is Vu Duc Tran");
            _loca = new Location(new string[] { "place1" }, "University", "Swinburne
    ↪ University");
            _sword = new Item(new string[] { "sword" }, "a sword", "This is a
    ↪ sword");

            _p.Location = _loca;
            _loca.Container.Put(_sword);
        }

        [Test]
        public void IdentifyLocation()
        {
            Assert.That(_loca.AreYou("place1"), Is.EqualTo(true), "Test Identify
    ↪ Location");
        }

        [Test]
        public void TestLocationLocateItem()
        {
            Assert.That(_loca.Locate("sword"), Is.EqualTo(_sword), "Test Identify
    ↪ Location");
        }

        [Test]
        public void PlayerLocateItemsInLocation()
        {
            Assert.That(_p.Location.Locate("sword"), Is.EqualTo(_sword), "Test Player
    ↪ Locate Items In Location");
        }
    }
}
```

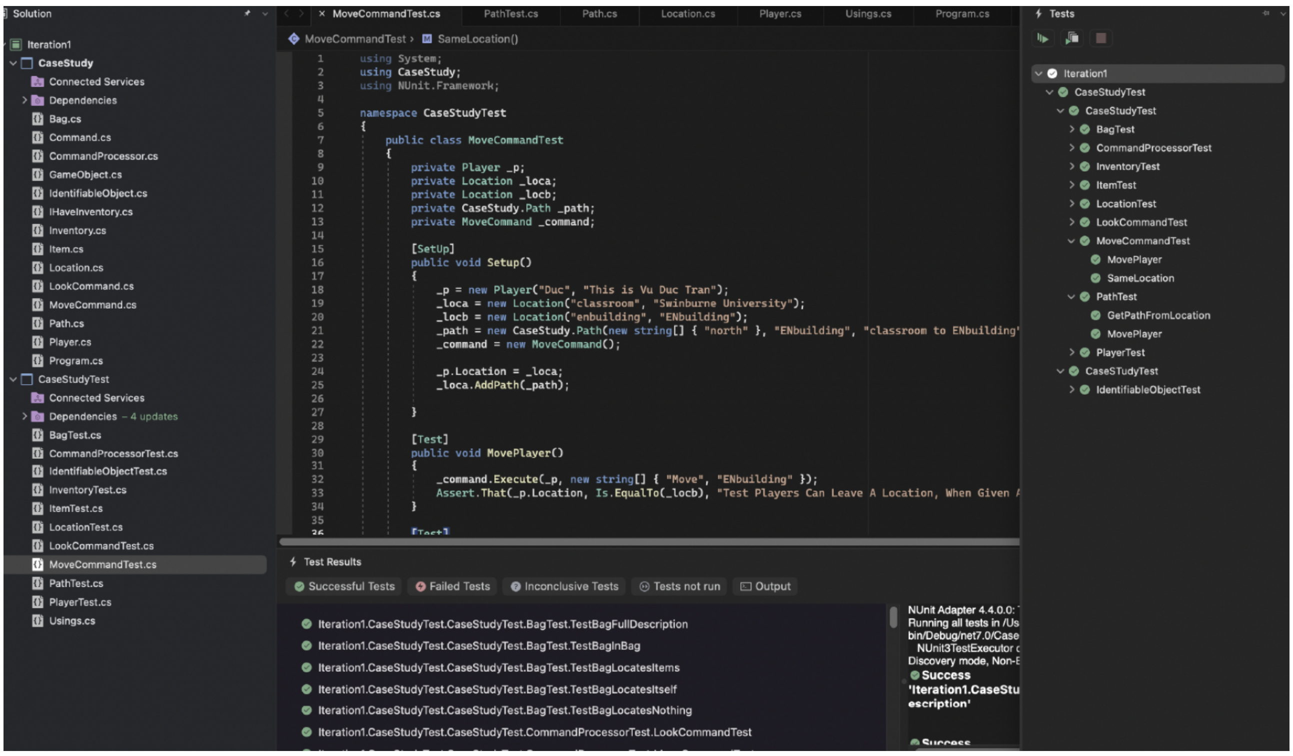
```
using CaseStudy;

namespace CaseStudy
{
    public class MoveCommand : Command
    {
        public MoveCommand(string[] ids) : base(ids)
        {
            AddIdentifier("move");
            AddIdentifier("go");
            AddIdentifier("leave");
            AddIdentifier("head");

        }

        public override string Execute(Player p, string[] text)
        {
            string message = "Error in move input.";
            string _moveDirection;

            switch (text.Length)
            {
                case 1:
                    return "Where do you want to move?";

                case 2:
                    _moveDirection = text[1].ToLower();
                    break;

                case 3:
                    _moveDirection = text[2].ToLower();
                    break;

                default:
                    return message;
            }

            if (!AreYou(text[0])) return message;

            GameObject? _path = p.Location.GetPath(_moveDirection);
            if (_path != null)
            {
                try
                {
                    (_path as Path).Move(p);
                }
                catch
                {
                    return "Can not find the " + _path.Name;
                }

                return "You have moved to the " + p.Location.Name + ".\r\n\n" +
↪   p.Location.FullDescription;
```

```
            }
            else
            {
                return message;
            }
        }
    }
}
```

```
using System;
using CaseStudy;
using NUnit.Framework;

namespace CaseStudyTest
{
    public class MoveCommandTest
    {
        private Player _p;
        private Location _loca;
        private Location _locb;
        private CaseStudy.Path _path;
        private MoveCommand _command;

        [SetUp]
        public void Setup()
        {
            _p = new Player("Duc", "This is Vu Duc Tran");
            _loca = new Location(new string[] { "north" }, "classroom", "Swinburne
↪   University");
            _locb = new Location(new string[] { "north" }, "ENbuilding",
↪   "ENbuilding");
            _path = new CaseStudy.Path(new string[] { "north", "ENbuilding" },
↪   "ENbuilding", "classroom to ENbuilding", _locb);
            _command = new MoveCommand(new string[] { "" });

            _p.Location = _loca;
            _loca.AddPath(_path);

        }
        [Test]
        public void InvalidMove()
        {
            _command.Execute(_p, new string[] { "walk", "ENbuilding" });
            Assert.That(_p.Location, Is.EqualTo(_loca), "Test Players Cannot Leave A
↪   Location, When Given An InValid Path Identifier");
        }

        [Test]
        public void MovePlayer()
        {
            _command.Execute(_p, new string[] { "Move", "ENbuilding" });
            Assert.That(_p.Location, Is.EqualTo(_locb), "Test Players Can Leave A
↪   Location, When Given A Valid Path Identifier");
        }

        [Test]
        public void SameLocation()
        {
            _command.Execute(_p, new string[] { "Move", "classroom" });
            Assert.That(_p.Location, Is.EqualTo(_loca), "Test Players Remain In The
↪   Same Location When They Leave With An Invalid Path Identifier");
        }
    }
}
```


<<abstract>>
Command
+ Command(string[] ids)
+ Execute(Player p, string[] text) : string (abstract)
MoveCommand
+ MoveCommand()
+ Execute(Player p, string[] text) : string
Path
-_destination: Location
+ Path(string[] ids, string name, string desc, Location destination)
+ Destination: Location <<readonly>>
+Move(Player p) : string
Location
-_inventory: Inventory
-_paths: List<Paths>
...
+ AddPath(Path path)
+ GetPath(string direction): Path

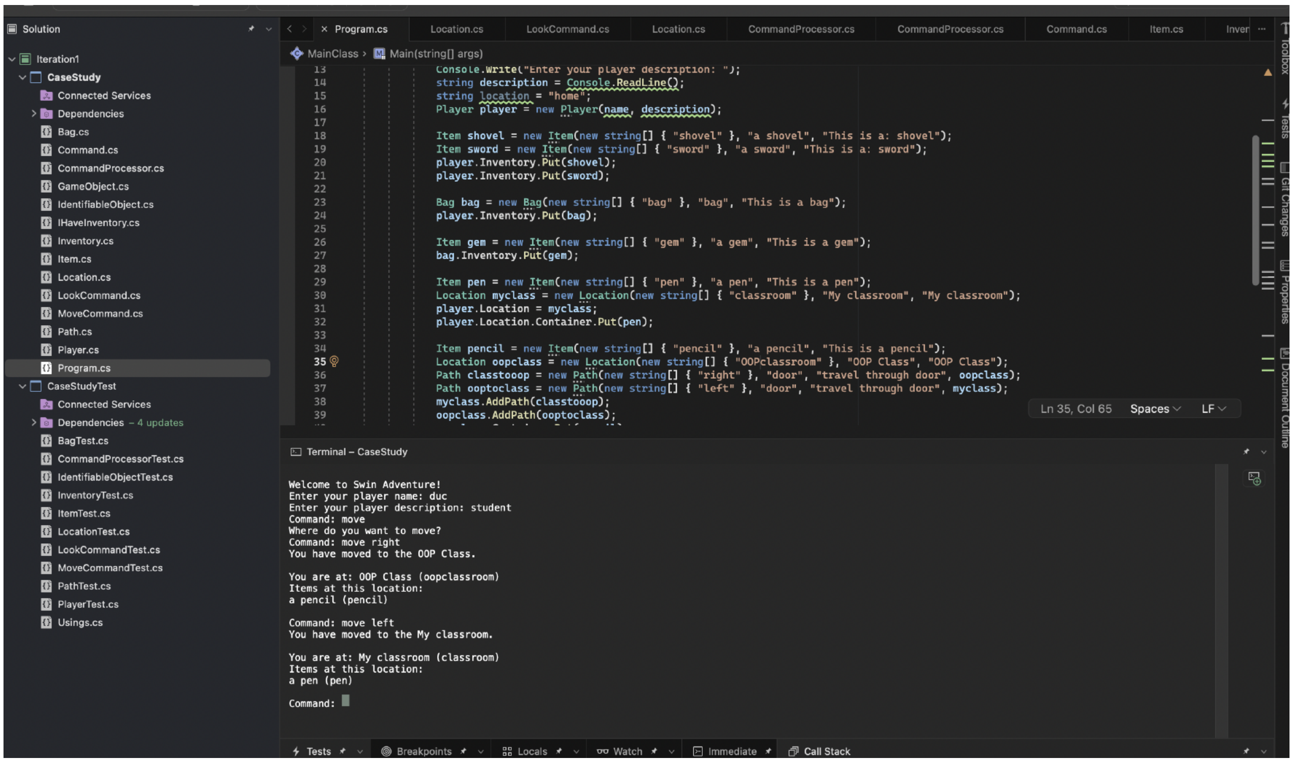

<<abstract>>
Command
+ Command(string[] ids)
+ Execute(Player p, string[] text) : string (abstract)
MoveCommand
+ MoveCommand()
+ Execute(Player p, string[] text) : string
Path
-_destination: Location
+ Path(string[] ids, string name, string desc, Location destination)
+ Destination: Location <<readonly>>
+Move(Player p) : string
Location
-_inventory: Inventory
-_paths: List<Paths>
...
+ AddPath(Path path)
+ GetPath(string direction): Path



```
using System;
using CaseStudy;
using NUnit.Framework;

namespace CaseStudyTest
{
    public class MoveCommandTest
    {
        private Player _p;
        private Location _loca;
        private Location _locb;
        private CaseStudy.Path _path;
        private MoveCommand _command;

        [SetUp]
        public void Setup()
        {
            _p = new Player("Duc", "This is Vu Duc Tran");
            _loca = new Location("classroom", "Swinburne University");
            _locb = new Location("enbuilding", "ENbuilding");
            _path = new CaseStudy.Path(new string[] { "north" }, "ENbuilding", "classroom to ENbuilding'
            _command = new MoveCommand();

            _p.Location = _loca;
            _loca.AddPath(_path);

        }

        [Test]
        public void MovePlayer()
        {
            _command.Execute(_p, new string[] { "Move", "ENbuilding" });
            Assert.That(_p.Location, Is.EqualTo(_locb), "Test Players Can Leave A Location, When Given A
        }

        [Test]
```

```
Iteration1.CaseStudyTest.CaseStudyTest.BagTest.TestBagFullDescription
Iteration1.CaseStudyTest.CaseStudyTest.BagTest.TestBagInBag
Iteration1.CaseStudyTest.CaseStudyTest.BagTest.TestBagLocatesItems
Iteration1.CaseStudyTest.CaseStudyTest.BagTest.TestBagLocatesItself
Iteration1.CaseStudyTest.CaseStudyTest.BagTest.TestBagLocatesNothing
Iteration1.CaseStudyTest.CaseStudyTest.CommandProcessorTest.LookCommandTest
```

```
NUnit Adapter 4.4.0.0:
Running all tests in /Us
bin/Debug/net7.0/Case
  NUnit3TestExecutor c
Discovery mode, Non-E
Success
'Iteration1.CaseStu
escription'

Success
```



```csharp
            Console.Write("Enter your player description: ");
            string description = Console.ReadLine();
            string location = "home";
            Player player = new Player(name, description);

            Item shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a: shovel");
            Item sword = new Item(new string[] { "sword" }, "a sword", "This is a: sword");
            player.Inventory.Put(shovel);
            player.Inventory.Put(sword);

            Bag bag = new Bag(new string[] { "bag" }, "bag", "This is a bag");
            player.Inventory.Put(bag);

            Item gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");
            bag.Inventory.Put(gem);

            Item pen = new Item(new string[] { "pen" }, "a pen", "This is a pen");
            Location myclass = new Location(new string[] { "classroom" }, "My classroom", "My classroom");
            player.Location = myclass;
            player.Location.Container.Put(pen);

            Item pencil = new Item(new string[] { "pencil" }, "a pencil", "This is a pencil");
            Location oopclass = new Location(new string[] { "OOPclassroom" }, "OOP Class", "OOP Class");
            Path classtooop = new Path(new string[] { "right" }, "door", "travel through door", oopclass);
            Path ooptoclass = new Path(new string[] { "left" }, "door", "travel through door", myclass);
            myclass.AddPath(classtooop);
            oopclass.AddPath(ooptoclass);
```

```
Welcome to Swin Adventure!
Enter your player name: duc
Enter your player description: student
Command: move
Where do you want to move?
Command: move right
You have moved to the OOP Class.

You are at: OOP Class (oopclassroom)
Items at this location:
a pencil (pencil)

Command: move left
You have moved to the My classroom.

You are at: My classroom (classroom)
Items at this location:
a pen (pen)

Command: 
```