Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 7 - Paths

PDF generated at 00:00 on Thursday 2nd November, 2023

```
using System;
namespace Iteration5

{
    public class Path:Game_Object
    {
        Location _l;
        bool _locked;

        public Path(string[] ids, string name, string desc, Location l) : base(ids, name, desc)
        {
            _l = l;
        }

        public Location Loc
        {
            get
            {
                return _l;
            }
        }
        public bool Locked
        {
            get
            {
                return _locked;
            }
            set
            {
                _locked = value;
            }
        }

        public override string FullDescription
        {
            get
            {
                return "At " + Name + "there is" + Loc.Name;
            }
        }
        public void Move_to_path(Path p)
        {
            _l = p.Loc;
        }
    }
}
```

```
using System;
namespace Iteration5
{
    [TestFixture()]
    public class PathTest
    {
        Location l;
        Path p;


        [SetUp()]
        public void Setup()
        {
            l = new Location("a rich country", "This is my beautiful country");
            p = new Path(new string[] { "west", "w" }, "west", "moving towards west", l);
        }
        [Test()]
        public void IdentifyPaths()
        {
            Assert.IsTrue(p.AreYou("west"));
            Assert.IsFalse(p.AreYou("there"));
        }
        [Test()]
        public void PathFullDescription()
        {
            Assert.AreEqual("On moving west there is a rich country", p.FullDescription);
        }
    }
}

```

```
using System;
using System.IO;
using Iteration6;
namespace Iteration5
{
    public class Location : Game_Object, IHaveInventory

    {
        Inventory _inv = new Inventory();
        List<Path> _path = new List<Path>();

        public Location(string name, string desc) : base(new string[] { "room" }, name, desc)
        {

        }
        public Game_Object Locate(string id)
        {

            if (AreYou(id) == true)
            {
                return this;
            }

            else if (_path.Count >= 1)
            {
                foreach (Path pt in _path)
                {
                    if (pt.AreYou(id))
                    {
                        return pt;
                    }
                }
                return null;
            }
            else
            {
                return _inv.Fetch(id);
            }
        }
        public override string FullDescription
        {
            get
            {
                return ("You are in the " + Name + "\n" + base.FullDescription + "\n" + ListofPaths + ".\nIn this room you can see:\n" + _inv.ItemList);
            }
        }
        public Inventory Inv
        {
            get
            {
                return _inv;
```

```
            }
        }
        public void AddPathInList(Path p)
        {
            _path.Add(p);
        }

        public string ListofPaths
        {
            get
            {
                if (_path.Count == 0)
                {
                    return "There are no exits.";
                }
                else
                {
                    string multiplepaths = "";
                    bool isFirst = true;
                    foreach (Path path in _path)
                    {
                        if (!isFirst)
                        {
                            multiplepaths += ", ";
                        }
                        multiplepaths += path.Name;
                        isFirst = false;
                    }
                    return "There are exits to the " + multiplepaths;
                }


            }

        }

    }
}

```

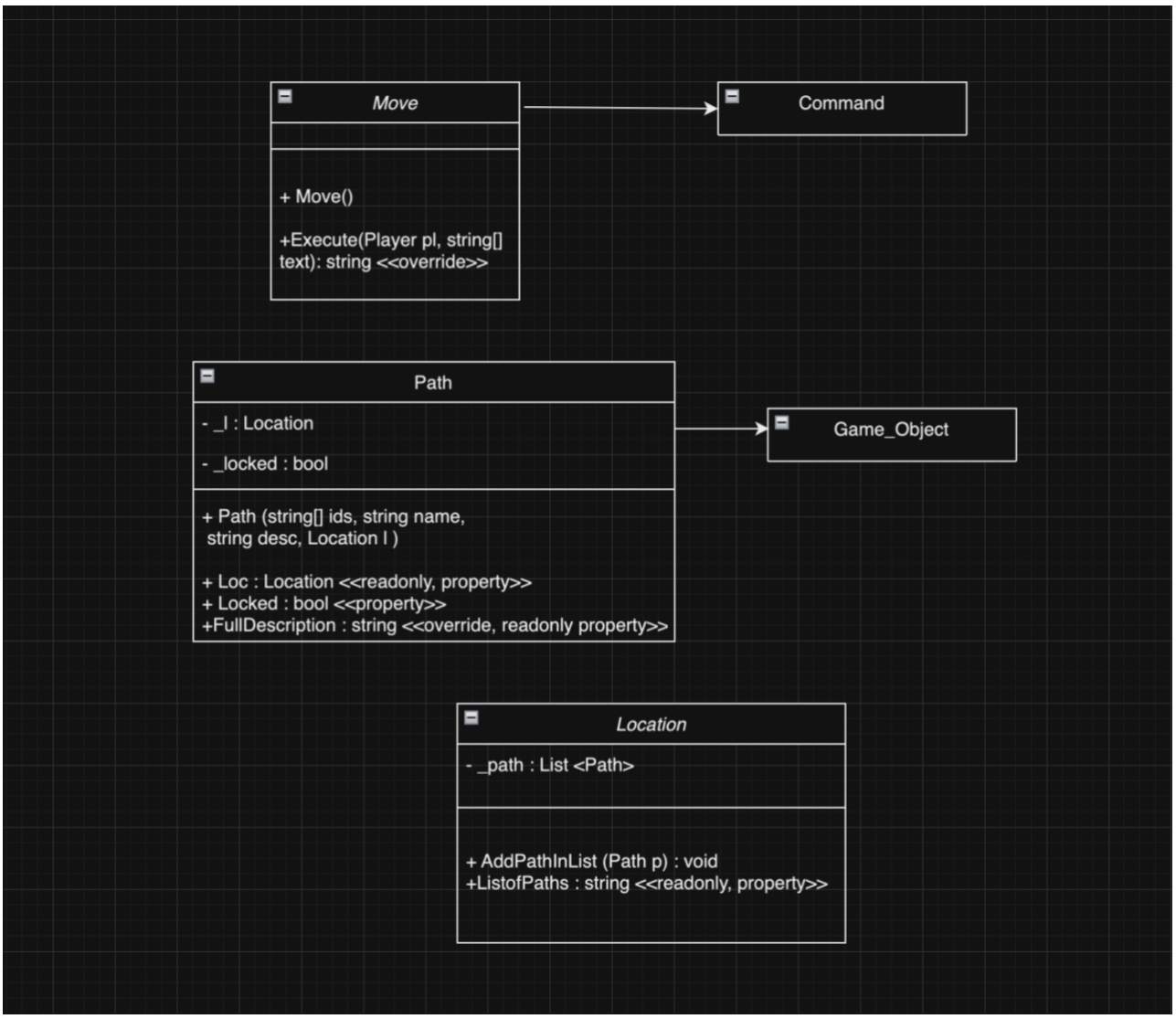
```
using System;
using System.ComponentModel;
using System.Xml.Linq;
using Iteration5;
using NUnit;
namespace Iteration5
{
    [TestFixture()]
    public class LocationTest
    {
        Player p;
        Location l;
        Item gem;
        Path path;
        Location finallocation;

        [SetUp()]
        public void SetUp()
        {
            p = new Player("Fred", "the mighty programmer");
            l = new Location("room", "a big room");
            gem = new Item(new string[] { "gem" }, "gem", "a bright red gem");
            path = new Path(new string[] { "west", "w" }, "west", "going to the west", finallocation);
            finallocation = new Location("University", "Swinburne Uni");


            p.Loc = l;
            l.Inv.Put(gem);
        }

        [Test()]
        public void Locations_identify_themselves()
        {
            Assert.AreSame(l, l.Locate("room"));
        }
        [Test()]
        public void Locations_locate_items()
        {
            Assert.AreSame(gem, l.Locate("gem"));
        }
        [Test()]
        public void Get_Path_From_location()
        {
            l.AddPathInList(path);
            Assert.AreSame(path, p.Locate("w"));
        }
        [Test()]
        public void FullDescriptionTest()
        {
            l.AddPathInList(path);
            Assert.AreEqual("You are in the room\na big room\nThere are exits to the west.\nIn this room you can see:\n\ta gem (gem)\n", l.FullDescription);
```

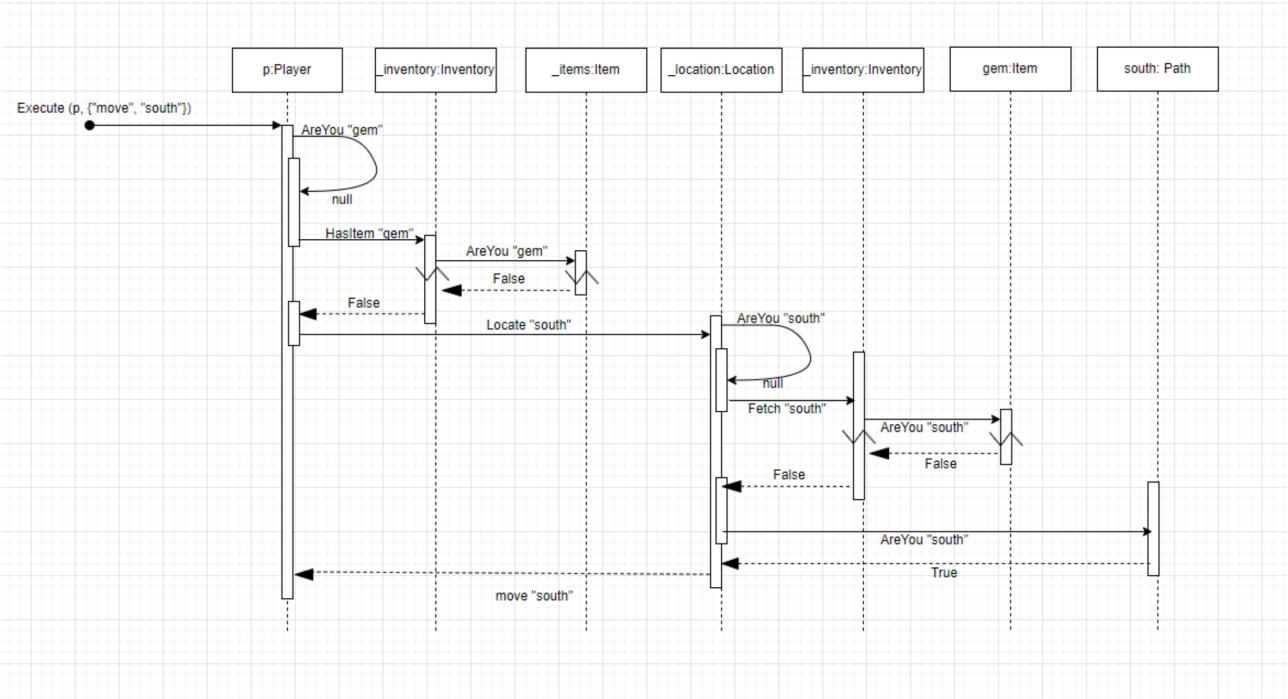
```

            }
        }
}
```

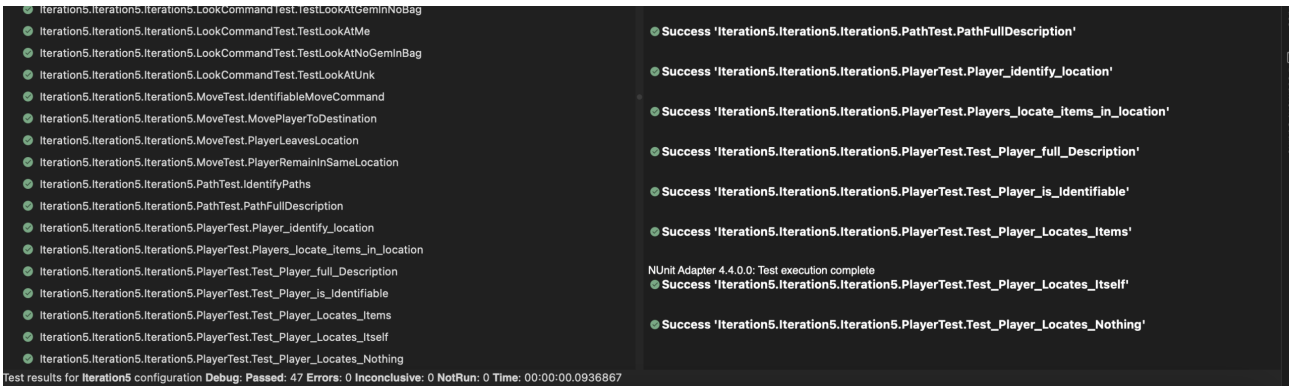
```
using System;
namespace Iteration5
{
    public class Move:Command
    {
        public Move(): base(new string[] {"leave", "head", "go", "move"})
        {
        }
        public override string Execute(Player pl, string[] text)
        {

            if (text.Length > 2)
            {
                return "How to move like that?";
            }
            else if (text[0] != "leave" && text[0] != "go" && text[0] != "head" && text[0] != "move")
            {
                return "Error";
            }
            else if (text.Length == 1)
            {
                return "Where to move?";
            }
            else
            {
                string direction = text[1];
                if (pl.Locate(direction) is Path p)
                {
                    pl.Move_to_path(p);
                    return ("You head towards " + p.Name + ". You have arrived in " + p.Loc.Name);
                }
                return "Error! ";
            }
        }
    }
}

```

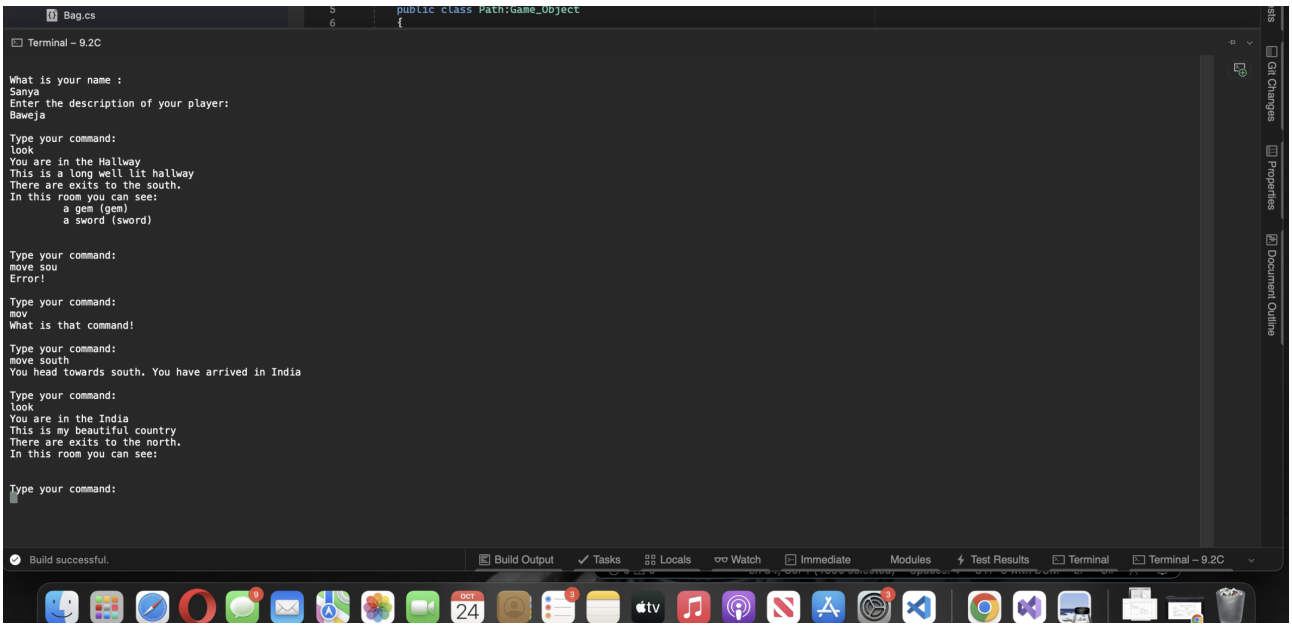
```
using NUnit.Framework.Internal;

namespace Iteration5
{
    [TestFixture()]
    public class MoveTest
    {
        Player p;
        Move m;
        Location l;
        Location dest;
        Path path;
        [SetUp()]
        public void SetUp()
        {
            p = new Player("Sanya", "I am a student at Swinburne.");
            m = new Move();
            l = new Location("Ambala", "My hometown");
            dest = new Location("Phagwara", "My mum's hometown");
            path = new Path(new string[] { "north", "n" }, "north", "State of Punjab", dest);

            p.Loc = l;
            l.AddPathInList(path);
        }
        [Test()]
        public void IdentifiableMoveCommand()
        {
            Assert.IsTrue(m.AreYou("move"));
            Assert.IsTrue(m.AreYou("go"));
            Assert.IsFalse(m.AreYou("chalo"));
        }
        [Test()]
        public void MovePlayerToDestination()
        {
            Assert.AreSame(l, p.Loc);
            m.Execute(p, new string[] { "move", "n" });
            Assert.AreSame(dest, p.Loc);
        }
        [Test()]
        public void PlayerLeavesLocation()
        {
            Assert.AreSame(l, p.Loc);
            m.Execute(p, new string[] { "move", "n" });
            Assert.AreNotSame(l, p.Loc);
        }
        [Test()]
        public void PlayerRemainInSameLocation()
        {
            Assert.AreSame(l, p.Loc);
            m.Execute(p, new string[] { "move", "s" });
            Assert.AreSame(l, p.Loc);
        }
```

```
    }
}
```


Move
+ Move()
+Execute(Player pl, string[] text): string <<override>>
Command
Path
- _l : Location
- _locked : bool
+ Path (string[] ids, string name, string desc, Location l)
+ Loc : Location <<readonly, property>>
+ Locked : bool <<property>>
+FullDescription : string <<override, readonly property>>
Game_Object
Location
- _path : List <Path>
+ AddPathInList (Path p) : void
+ListofPaths : string <<readonly, property>>


p:Player
_inventory:Inventory
_items:Item
_location:Location
_inventory:Inventory
gem:Item
south: Path
Execute (p, {"move", "south"})
AreYou "gem"
null
HasItem "gem"
AreYou "gem"
False
False
Locate "south"
AreYou "south"
null
Fetch "south"
AreYou "south"
False
False
AreYou "south"
True
move "south"


Iteration5.Iteration5.Iteration5.LookCommandTest.TestLookAtGemInNoBag
Iteration5.Iteration5.Iteration5.LookCommandTest.TestLookAtMe
Iteration5.Iteration5.Iteration5.LookCommandTest.TestLookAtNoGemInBag
Iteration5.Iteration5.Iteration5.LookCommandTest.TestLookAtUnk
Iteration5.Iteration5.Iteration5.MoveTest.IdentifiableMoveCommand
Iteration5.Iteration5.Iteration5.MoveTest.MovePlayerToDestination
Iteration5.Iteration5.Iteration5.MoveTest.PlayerLeavesLocation
Iteration5.Iteration5.Iteration5.MoveTest.PlayerRemainInSameLocation
Iteration5.Iteration5.Iteration5.PathTest.IdentifyPaths
Iteration5.Iteration5.Iteration5.PathTest.PathFullDescription
Iteration5.Iteration5.Iteration5.PlayerTest.Player_identify_location
Iteration5.Iteration5.Iteration5.PlayerTest.Players_locate_items_in_location
Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_full_Description
Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_is_Identifiable
Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_Locates_Items
Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_Locates_Itself
Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_Locates_Nothing
Success 'Iteration5.Iteration5.Iteration5.PathTest.PathFullDescription'
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Player_identify_location'
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Players_locate_items_in_location'
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_full_Description'
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_is_Identifiable'
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_Locates_Items'
NUnit Adapter 4.4.0.0: Test execution complete
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_Locates_Itself'
Success 'Iteration5.Iteration5.Iteration5.PlayerTest.Test_Player_Locates_Nothing'
Test results for Iteration5 configuration Debug: Passed: 47 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0936867


Bag.cs
public class Path:Game_Object
{
Terminal – 9.2C
What is your name :
Sanya
Enter the description of your player:
Baweja
Type your command:
look
You are in the Hallway
This is a long well lit hallway
There are exits to the south.
In this room you can see:
a gem (gem)
a sword (sword)
Type your command:
move sou
Error!
Type your command:
mov
What is that command!
Type your command:
move south
You head towards south. You have arrived in India
Type your command:
look
You are in the India
This is my beautiful country
There are exits to the north.
In this room you can see:
Type your command:
Build successful.
Build Output
Tasks
Locals
Watch
Immediate
Modules
Test Results
Terminal
Terminal – 9.2C
Git Changes
Properties
Document Outline