Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 7 - Paths

PDF generated at 04:10 on Monday 6th November, 2023

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

namespace SwinAdventure
{
    public class Path : GameObject
    {
        private Location _target;
        private bool _closed;

        public Path(string[] ids, string name, string desc, Location target) :
↪   base(ids, name, desc)
        {
            _target = target;
            _closed = false;

            AddIdentifier("path");
            foreach (string s in name.Split(" "))
            {
                AddIdentifier(s);
            }
        }

        public Location Target
        {
            get
            {
                return _target;
            }
        }

        public override string Description
        {
            get
            {
                return Name;
            }
        }

        public bool Closed
        {
            get
            {
                return _closed;
            }
            set
            {
                _closed = value;
            }
        }
```

```
53      }
54  }
```

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

namespace SwinAdventure
{
    public class PathTesting
    {
        Path pathtestA;
        Path pathtestB;
        Location roomA;
        Location roomB;
        Player playerTest;

        [SetUp]
        public void Setup()
        {
            roomA = new Location("a room", "This is roomA");
            roomB = new Location("a room", "This is roomB");
            pathtestA = new Path(new string[] { "south" }, "south", "A path", roomA);
            Player playerTest = new Player("Binh", "Nepenthes poacher");

            playerTest.Location = roomA;
            pathtestB = new Path(new string[] { "south" }, "south", "B path", roomB);
            roomA.AddPath(pathtestA);
            roomB.AddPath(pathtestB);
        }

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

        [Test]
        public void PathLocationTest()
        {
            Location expected = roomA;
            Location actual = pathtestA.Target;

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

        [Test]
        public void TestLocatePath()
        {

            GameObject expected = roomA.Locate("south");
            GameObject actual = null;
```

```
54
55              Assert.That(actual, Is.EqualTo(expected));
56          }
57      }
58
59  }
```

```
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
    {
        private Inventory _inventory = new Inventory();
        private List<Path> _paths = new List<Path>();

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

        public override string Description
        {
            get
            {
                return Name + _desc + PathList + /*Description +*/ "\n" + "Around you
↪   there are" + "\n" + _inventory.ItemList ;
            }
        }

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

        public Path locatePath(string path)
        {
            foreach (Path p in _paths)
            {
                if (p.AreYou(path))
                {
                    return p;
                }
```

```
52              }
53              return null;
54          }
55
56          public void AddPath(Path path)
57          {
58              _paths.Add(path);
59          }
60
61          public string PathList
62          {
63              get
64              {
65                  string paths = string.Empty + "\n";
66
67                  if (_paths.Count == 1)
68                  {
69                      return "\n" + "There is an exit to Central Hall "; //might change
    ↪  later if there are secret rooms
70                  }
71
72                  paths = paths + "direction/s available are(is): ";
73
74                  foreach (Path path in _paths)
75                  {
76                      if (path == _paths.Last())
77                      {
78                          paths = paths + " and " + path.FirstId + ".";
79                      }
80                      else if (path == _paths.First())
81                      {
82                          paths = paths + " " + path.FirstId;
83                      }
84                      else
85                      {
86                          paths = paths + ", " + path.FirstId;
87                      }
88                  }
89
90                  return paths;
91              }
92          }
93      }
94  }
```

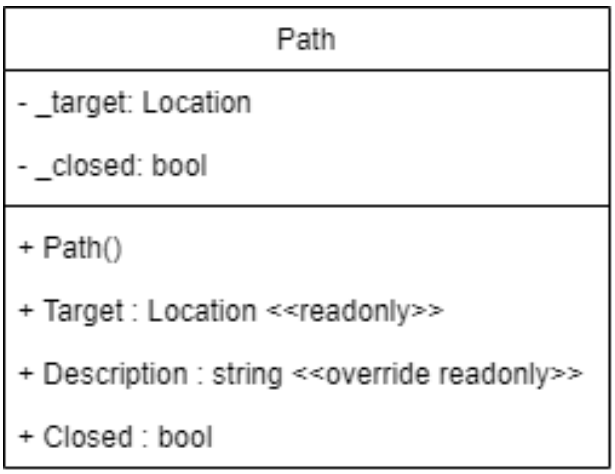
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class LocationTesting
    {
        Player p = new Player("Binh", "A man not a god");
        Location l = new Location("Bed", "Binh old but comfy bed");
        Item ipad = new Item(new string[] { "ipad" }, "ipad pro M1", "A ipad pro released in 2020");

        [Test]
        public void TestNotLocation()
        {
            p.Location = l;
            bool actual = l.AreYou("hi");
            Assert.IsFalse(actual);
        }

        [Test]
        public void TestPLayerHasLocation()
        {
            p.Location = l;
            GameObject expect = l;
            GameObject actual = p.Locate("location");
            Assert.AreEqual(actual, expect);
        }

        [Test]
        public void TestLocationLocateItem()
        {
            l.Inventory.Put(ipad);
            p.Location = l;
            GameObject expect = ipad;
            GameObject actual = l.Locate("ipad");
            Assert.AreEqual(actual, expect);
        }

        [Test]
        public void TestEmptyLocation()
        {
            Assert.That(l.Locate("Vsmart"), Is.EqualTo(null));
        }
    }
}
```

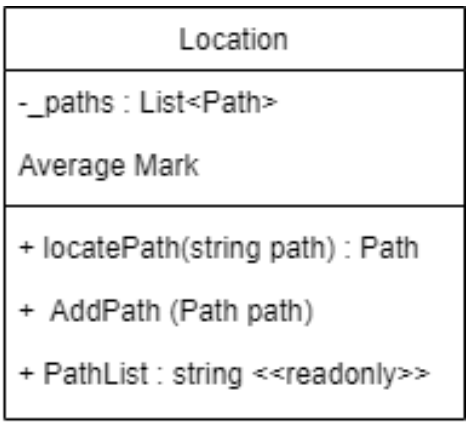
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace SwinAdventure
{
    public class MoveCommand : Command
    {
        public MoveCommand() : base(new string[] { "move", "go", "head", "leave" })
        {
        }
        public override string Execute(Player p, string[] text)
        {
            if (text[0] != "move" && text[0] != "go" && text[0] != "head" && text[0] != "leave")
            {
                return "Error in the first word input: \nFirst word allow: move, go, head, leave";
            }

            string _destination;

            switch (text.Length)
            {
                case 1:
                    return "Specify the destination!!!";

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

                case 3:
                    _destination = text[2].ToLower();
                    break;
                default:
                    return "Only allow 3 words move input";
            }

            Path direction = p.Location.locatePath(_destination);
            if (direction != null)
            {
                p.Move(direction);
                return $"You are now in {direction.Target.Name}";
            }
            else
            {
                return "Something wrong with the direction!!!";
            }
        }
```

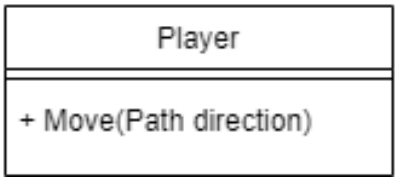
```
52      }
53  }
```

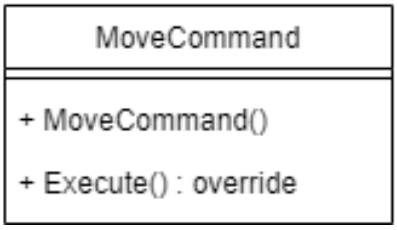
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class Movetesting
    {
        MoveCommand move;
        Location A;
        Location B;
        Path pathtest;
        Player player;

        [SetUp]
        public void Setup()
        {
            move = new MoveCommand();
            A = new Location("A", "pointA");
            B = new Location("B", "pointB");
            pathtest = new Path(new string[] { "north" }, "north", "this is north",
↪   B);
            player = new Player("Binh", "nepenthes collector");

            player.Location = A;
            A.AddPath(pathtest);
        }

        [Test]
        public void TestMove()
        {
            move.Execute(player, new string[] { "move", "north" });
            Assert.That(player.Location, Is.EqualTo(B));
        }

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

        [Test]
        public void TestOnlyMoveInput()
        {
            string command = move.Execute(player, new string[] { "move" });
            string expected = "Specify the destination!!!";

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

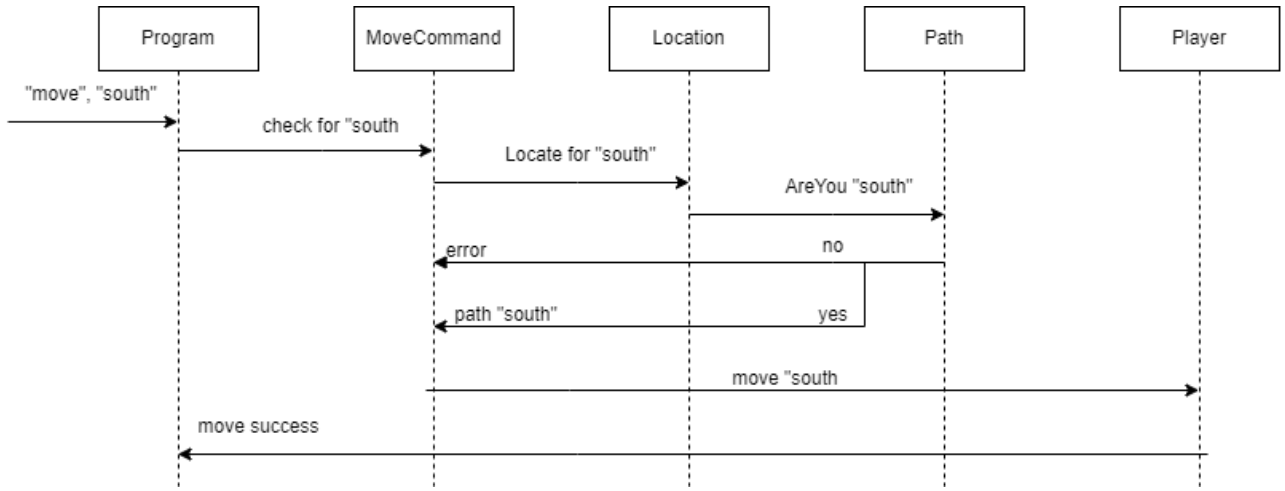
```
53
54          [Test]
55          public void TestInvalidDirection()
56          {
57              string command = move.Execute(player, new string[] { "move", "east" });
58              string expected = "Something wrong with the direction!!!";
59
60              Assert.That(command, Is.EqualTo(expected));
61          }
62      }
63  }
```

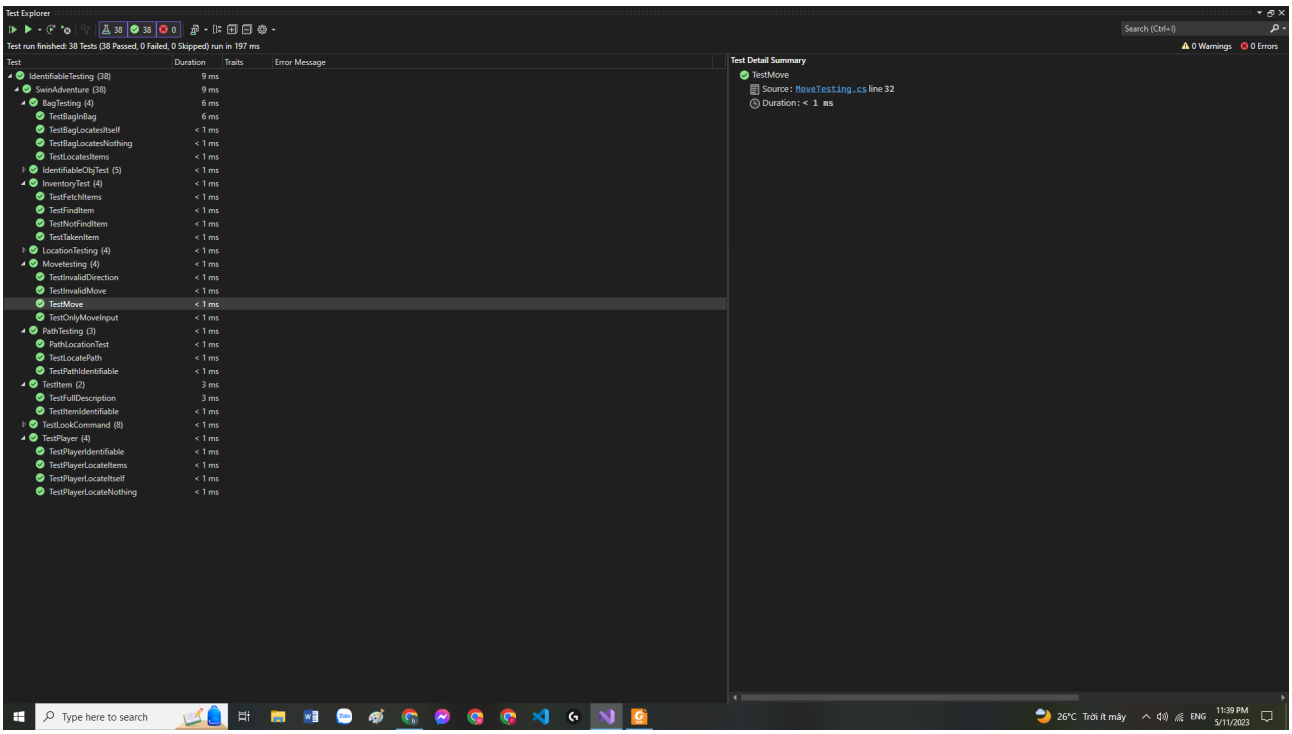

Path
- _target: Location
- _closed: bool
+ Path()
+ Target : Location <<readonly>>
+ Description : string <<override readonly>>
+ Closed : bool

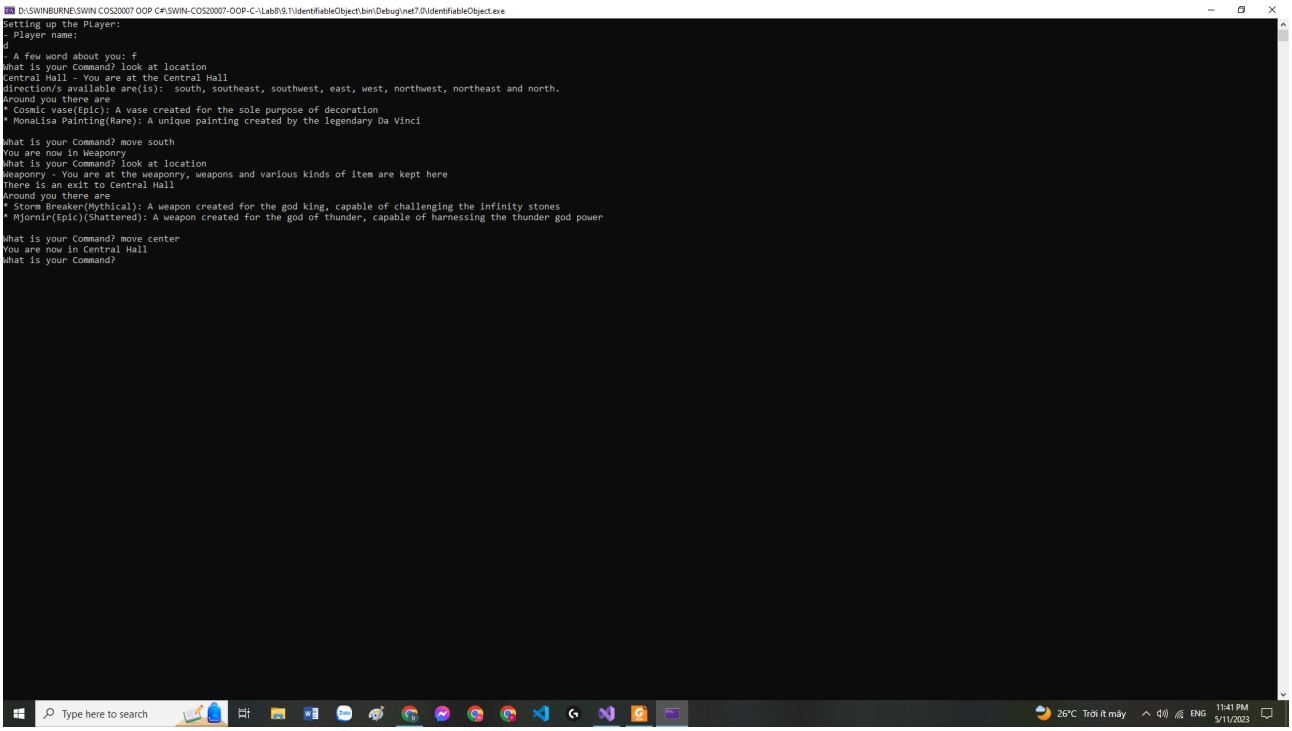

Location
-_paths : List<Path>
Average Mark
+ locatePath(string path) : Path
+ AddPath (Path path)
+ PathList : string <<readonly>>


Player
+ Move(Path direction)


MoveCommand
+ MoveCommand()
+ Execute() : override


Program
MoveCommand
Location
Path
Player
"move", "south"
check for "south
Locate for "south"
AreYou "south"
error
no
path "south"
yes
move "south
move success


Test Explorer
Test run finished: 38 Tests (38 Passed, 0 Failed, 0 Skipped) run in 197 ms
Test
Duration
Traits
Error Message
IdentifiableTesting (38) 9 ms
SwinAdventure (38) 9 ms
BagTesting (4) 6 ms
TestBagInBag 6 ms
TestBagLocatesItself < 1 ms
TestBagLocatesNothing < 1 ms
TestLocatesItems < 1 ms
IdentifiableObjTest (5) < 1 ms
InventoryTest (4) < 1 ms
TestFetchItems < 1 ms
TestFindItem < 1 ms
TestNotFindItem < 1 ms
TestTakenItem < 1 ms
LocationTesting (4) < 1 ms
Movetesting (4) < 1 ms
TestInvalidDirection < 1 ms
TestInvalidMove < 1 ms
TestMove < 1 ms
TestOnlyMoveInput < 1 ms
PathTesting (3) < 1 ms
PathLocationTest < 1 ms
TestLocatePath < 1 ms
TestPathIdentifiable < 1 ms
TestItem (2) 3 ms
TestFullDescription 3 ms
TestItemIdentifiable < 1 ms
TestLookCommand (8) < 1 ms
TestPlayer (4) < 1 ms
TestPlayerIdentifiable < 1 ms
TestPlayerLocateItems < 1 ms
TestPlayerLocateItself < 1 ms
TestPlayerLocateNothing < 1 ms
Test Detail Summary
TestMove
Source: MoveTesting.cs line 32
Duration: < 1 ms
Type here to search
26°C Trời ít mây
ENG
11:39 PM
5/11/2023


D:\SWINBURNE\SWIN COS20007 OOP C#\SWIN-COS20007-OOP-C-\Lab8\9.1\IdentifiableObject\bin\Debug\net7.0\IdentifiableObject.exe
Setting up the PLayer:
- Player name:
d
- A few word about you: f
What is your Command? look at location
Central Hall - You are at the Central Hall
direction/s available are(is): south, southeast, southwest, east, west, northwest, northeast and north.
Around you there are
* Cosmic vase(Epic): A vase created for the sole purpose of decoration
* MonaLisa Painting(Rare): A unique painting created by the legendary Da Vinci
What is your Command? move south
You are now in Weaponry
What is your Command? look at location
Weaponry - You are at the weaponry, weapons and various kinds of item are kept here
There is an exit to Central Hall
Around you there are
* Storm Breaker(Mythical): A weapon created for the god king, capable of challenging the infinity stones
* Mjornir(Epic)(Shattered): A weapon created for the god of thunder, capable of harnessing the thunder god power
What is your Command? move center
You are now in Central Hall
What is your Command?
Type here to search
26°C Trời ít mây
ENG
11:41 PM
5/11/2023