Selected files

22 printable files

IdentifiableObjectTest/IdentifiableObjectTest.cs
IdentifiableObjectTest/ItemTest.cs
IdentifiableObjectTest/InventoryTest.cs
IdentifiableObjectTest/LocationTest.cs
IdentifiableObjectTest/LookCommandTest.cs
IdentifiableObjectTest/MoveCommandTest.cs
IdentifiableObjectTest/PathTest.cs
IdentifiableObjectTest/PlayerTest.cs
IdentifiableObjectTest/BagTest.cs
Swin-Adventure/Bag.cs
Swin-Adventure/Command.cs
Swin-Adventure/IdentifiableObjectClass.cs
Swin-Adventure/GameObject.cs
Swin-Adventure/IHaveInventory.cs
Swin-Adventure/Inventory.cs
Swin-Adventure/Item.cs
Swin-Adventure/Location.cs
Swin-Adventure/LookCommand.cs
Swin-Adventure/MoveCommand.cs
Swin-Adventure/Path.cs
Swin-Adventure/Player.cs
Swin-Adventure/Program.cs

IdentifiableObjectTest/IdentifiableObjectTest.cs

```
1  using NUnit.Framework;
2  using Swin_Adventure;
3
4  namespace IdentifiableObjectTest
5  {
6
7      internal class Tests
8      {
9          private IdentifiableObject _test1;
10         private IdentifiableObject _test2;
11         private IdentifiableObject _test3;
12         private IdentifiableObject _test4;
13         private IdentifiableObject _test5;
14         private IdentifiableObject _test6;
15
16         [SetUp]
17         public void Setup()
18         {
19             _test1 = new IdentifiableObject(new string[] { "fred", "bob" });
20             _test2 = new IdentifiableObject(new string[] { "fred", "bob" });
21             _test3 = new IdentifiableObject(new string[] { "fred", "bob" });
22             _test4 = new IdentifiableObject(new string[] { "fred", "bob" });
23             _test5 = new IdentifiableObject(new string[] { });
24             _test6 = new IdentifiableObject(new string[] { "fred", "bob" });
25         }
26
27         [Test]
28         public void TestAreYou()
29         {
30             Assert.IsTrue(_test1.AreYou("fred"));
31             Assert.IsTrue(_test1.AreYou("bob"));
32         }
33
```

```
34          [Test]
35          public void TestNotAreYou()
36          {
37              Assert.IsFalse(_test2.AreYou("wilma"));
38              Assert.IsFalse(_test2.AreYou("boby"));
39          }
40
41          [Test]
42          public void TestCaseSensitive()
43          {
44              Assert.IsTrue(_test3.AreYou("FRED"));
45              Assert.IsTrue(_test3.AreYou("bOB"));
46          }
47
48          [Test]
49          public void TestFirstID()
50          {
51              Assert.AreEqual("fred", _test4.FirstId);
52          }
53
54          [Test]
55          public void TestFirstIdWithNoIDs()
56          {
57              Assert.AreEqual("", _test5.FirstId);
58          }
59
60          [Test]
61          public void TestAddID()
62          {
63              _test6.AddIdentifier("wilma");
64              Assert.IsTrue(_test6.AreYou("fred"));
65              Assert.IsTrue(_test6.AreYou("bob"));
66              Assert.IsTrue(_test6.AreYou("wilma"));
67          }
68      }
69  }
70
71
```

IdentifiableObjectTest/ItemTest.cs

```
1   using System;
2   using Swin_Adventure;
3
4   namespace SwinAdventureTest
5   {
6       [TestFixture]
7       public class ItemTest
8        {
9             private Item _itemTest;
10
11
12          [SetUp]
13          public void Setup()
14          {
15              _itemTest = new Item(new string[] { "weapon" }, "sword", "This is an
    Excalibur");
16
17          }
18
19          [Test]
```

```
        public void TestItemIsIdentifiable()
        {
            Assert.IsTrue(_itemTest.AreYou("weapon"));
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.IsTrue(_itemTest.ShortDescription == "a sword (weapon)");
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.IsTrue(_itemTest.FullDescription == "This is an Excalibur");
        }
    }
}


```

IdentifiableObjectTest/InventoryTest.cs

```
using System;
using Swin_Adventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class InventoryTest
     {
          private Inventory _inventoryTest;
          private Item _weaponTest;
          private Item _armorTest;

          [SetUp]
          public void SetUp()
          {
               _inventoryTest = new Inventory();
               _weaponTest = new Item(new string[] { "weapon" }, "sword", "this is an Excalibur");
            _armorTest = new Item(new string[] { "armor" }, "shield", "this is a shield");

            _inventoryTest.Put(_weaponTest);
            _inventoryTest.Put(_armorTest);
        }

        [Test]
        public void TestFindItem()
        {
            Assert.IsTrue(_inventoryTest.HasItem("weapon"));
            Assert.IsTrue(_inventoryTest.HasItem("armor"));
        }

        [Test]
        public void TestNoItemFind()
        {
            Assert.IsFalse(_inventoryTest.HasItem("axe"));
            Assert.IsFalse(_inventoryTest.HasItem("helmet"));
        }
```

```
37
38          [Test]
39          public void TestFetchItem()
40          {
41              Assert.IsTrue(_weaponTest == _inventoryTest.Fetch("weapon"));
42              Assert.IsTrue(_inventoryTest.HasItem("weapon"));
43
44              Assert.IsTrue(_armorTest == _inventoryTest.Fetch("armor"));
45              Assert.IsTrue(_inventoryTest.HasItem("armor"));
46          }
47
48          [Test]
49          public void TestTakeItem()
50          {
51              Assert.IsTrue(_weaponTest == _inventoryTest.Take("weapon"));
52              Assert.IsFalse(_inventoryTest.HasItem("weapon"));
53
54              Assert.IsTrue(_armorTest == _inventoryTest.Take("armor"));
55              Assert.IsFalse(_inventoryTest.HasItem("armor"));
56          }
57
58          [Test]
59          public void TestItemList()
60          {
61              Assert.IsTrue(_inventoryTest.ItemList.Replace("\t", "") == "a sword
    (weapon)\na shield (armor)\n");
62          }
63      }
64  }
65
66
```

IdentifiableObjectTest/LocationTest.cs

```
1   using NUnit.Framework;
2   using Swin_Adventure;
3   using System;
4
5   namespace SwinAdventureTest
6   {
7       [TestFixture]
8       public class LocationTest
9       {
10          public Location _locationTest;
11          public Item _itemTest;
12          public Player _playerTest;
13          Location _roomBTest;
14          Swin_Adventure.Path _pathTest;
15          Move _commandTest;
16
17          [SetUp]
18          public void Setup()
19          {
20              _locationTest = new Location(new string[] { "location" }, "forest", "a
    rain forest");
21              _itemTest = new Item(new string[] { "gem" }, "ruby", "a bright red ruby")
    ;
22              _playerTest = new Player("thuan", "dan choi", _locationTest);
23
24              _roomBTest = new Location(new string[] { "roomB" }, "Room B", "Room B");
25              _pathTest = new Swin_Adventure.Path(new string[] { "north" }, "Door", "A
```

```
test door", _locationTest, _roomBTest);
            _locationTest.AddPath(_pathTest);
            _commandTest = new Move();


            _locationTest.Inventory.Put(_itemTest);
        }

        [Test]
        public void TestLocationIsIdentifiable()
        {
            Assert.IsTrue(_locationTest.AreYou("location"));
        }

        [Test]
        public void TestLocationCanLocateItems()
        {
            Assert.IsTrue(_locationTest.Locate("gem") == _itemTest);
            Assert.IsTrue(_locationTest.Inventory.HasItem("gem"));
        }

        [Test]
        public void TestLocationLocateItself()
        {
            Assert.IsTrue(_locationTest == _locationTest.Locate("location"));
        }

        [Test]
        public void TestLocationLocateNothing()
        {
            Assert.IsNull(_locationTest.Locate("sword"));
        }

        [Test]
        public void TestLocationFullDescription()
        {
            Assert.IsTrue(_locationTest.FullDescription.Contains("a rain forest"));
            Assert.IsTrue(_locationTest.FullDescription.Contains("a ruby (gem)"));
        }

        [Test]
        public void TestLocationHasPath()
        {
            Assert.AreEqual(_pathTest, _playerTest.Location.Locate("north"));
        }

        [Test]
        public void TestLocationHasNoPath()
        {
            Assert.IsNull(_locationTest.Locate("south"));
        }

        [Test]
        public void TestMovePlayer()
        {
            _commandTest.Execute(_playerTest, new string[] { "move", "north" });
            Assert.AreEqual(_playerTest.Location, _roomBTest);
        }

        [Test]
```

```
85          public void TestSameLocation()
86          {
87              _commandTest.Execute(_playerTest, new string[] { "Move", "south" });
88              Assert.AreEqual(_playerTest.Location, _locationTest);
89          }
90      }
91  }
92
93
94
```

IdentifiableObjectTest/LookCommandTest.cs

```
1   using NUnit.Framework;
2   using System.Numerics;
3   using Swin_Adventure;
4
5   namespace SwinAdventureTest
6   {
7       [TestFixture]
8       public class TestLookCommand
9       {
10          private LookCommand _lookCommandTest;
11          private Player _playerTest;
12          private Bag _bagTest;
13          private Item _gemTest;
14          public Location _locationTest;
15
16          [SetUp]
17          public void Setup()
18          {
19              _lookCommandTest = new LookCommand();
20              _playerTest = new Player("thuan", "dan choi", _locationTest);
21              _bagTest = new Bag(new string[] { "duffelbag" }, "duffelbag", "it's
    small-sized");
22              _gemTest = new Item(new string[] { "gem" }, "gem", "a beautiful gem");
23          }
24
25          [Test]
26          public void TestLookAtMe()
27          {
28              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "inventory" }), Is.EqualTo("You are thuan, dan choi.\nYou are carrying:\n"));
29          }
30
31          [Test]
32          public void TestLookAtGem()
33          {
34              _playerTest.Inventory.Put(_gemTest);
35
36              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "gem" }), Is.EqualTo("a beautiful gem"));
37          }
38
39          [Test]
40          public void TestLookAtUnk()
41          {
42              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "unknown" }), Is.EqualTo("I can't find the unknown"));
43          }
44
```

```
45          [Test]
46          public void TestLookAtGemInMe()
47          {
48              _playerTest.Inventory.Put(_gemTest);
49
50              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "gem", "in", "inventory" }), Is.EqualTo("a beautiful gem"));
51          }
52
53          [Test]
54          public void TestLookAtGemInBag()
55          {
56              _bagTest.Inventory.Put(_gemTest);
57              _playerTest.Inventory.Put(_bagTest);
58
59              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "gem", "in", "duffelbag" }), Is.EqualTo("a beautiful gem"));
60          }
61
62          [Test]
63          public void TestLookAtGemInNoBag()
64          {
65              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "gem", "in", "duffelbag" }), Is.EqualTo("I can't find the duffelbag"));
66          }
67
68          [Test]
69          public void TestLookAtNoGemInBag()
70          {
71              _playerTest.Inventory.Put(_bagTest);
72
73              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "gem", "in", "duffelbag" }), Is.EqualTo("I can't find the gem"));
74          }
75
76          [Test]
77          public void TestInvalidLook()
78          {
79              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "around" }), Is.EqualTo("I don't know how to look like that"));
80              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "hello"
    }), Is.EqualTo("I don't know how to look like that"));
81              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "at", "a", "at", "b" }), Is.EqualTo("What do you want to look in?"));
82              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "hello",
    "at", "a" }), Is.EqualTo("Error in look input"));
83              Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
    "by", "a" }), Is.EqualTo("What do you want to look at?"));
84          }
85      }
86  }
87
```

IdentifiableObjectTest/MoveCommandTest.cs

```
1  using NUnit.Framework;
2  using System.Collections.Generic;
3  using System;
4  using Swin_Adventure;
5
6  namespace SwinAdventureTest
7  {
8      public class MoveTests
```

```
    {
        private Location _location1;
        private Location _location2;
        private Swin_Adventure.Path _path;
        private Player _player;
        private Move _moveCommand;

        [SetUp]
        public void Setup()
        {
            _location1 = new Location(new string[] { "location1" }, "Location1", "Location1");
            _location2 = new Location(new string[] { "location2" }, "Location2", "Location2");
            _path = new Swin_Adventure.Path(new string[] { "path1" }, "a path", "long path", _location1, _location2);
            _player = new Player("TestPlayer", "Player 1", _location1);
            _location1.AddPath(_path);
            _moveCommand = new Move();
        }

        [Test]
        public void GetPathFromLocation()
        {
            Assert.AreEqual(_path, _player.Location.Locate("path1"));
        }

        [Test]
        public void PathCanMovePlayerToDestination()
        {
            string expected = $"You have moved to {_location2.Name}.\r\n\n{_location2.FullDescription}";
            string result = _moveCommand.Execute(_player, new string[] { "move", "path1" });
            Assert.AreEqual(expected, result);
        }

        [Test]
        public void PlayersCanLeaveLocationWithValidPathIdentifier()
        {
            string expected = $"You have moved to {_location2.Name}.\r\n\n{_location2.FullDescription}";
            string result = _moveCommand.Execute(_player, new string[] { "move", "path1" });
            Assert.AreEqual(expected, result);
        }

        [Test]
        public void PlayersInSameLocationWithInvalidPathIdentifier()
        {
            Move moveCommand = new Move();
            string result = moveCommand.Execute(_player, new string[] { "move", "invalidpath" });
            Assert.AreEqual("Invalid path identifier. You remain in the same location.", result);
            Assert.AreEqual(_location1, _player.Location);
        }
    }
}


```

IdentifiableObjectTest/PathTest.cs

```
1  using System;
2  using System.Numerics;
3  using System.Reflection.Metadata;
4  using NUnit.Framework;
5  using Swin_Adventure;
6
7  namespace SwinAdventureTest
8  {
9      public class PathTest
10     {
11         Player _playerTest;
12         Location _roomATest;
13         Location _roomBTest;
14         Swin_Adventure.Path _pathTest;
15         Move _commandTest;
16
17         [SetUp]
18         public void Setup()
19         {
20             _roomATest = new Location(new string[] { "roomA" }, "Room A", "Room A");
21             _roomBTest = new Location(new string[] { "roomB" }, "Room B", "Room B");
22
23
24             _playerTest = new Player("thuan", "dan choi", _roomATest);
25             _pathTest = new Swin_Adventure.Path(new string[] { "north" }, "Door", "A
   test door", _roomATest, _roomBTest);
26             _roomATest.AddPath(_pathTest);
27             _commandTest = new Move();
28         }
29
30         [Test]
31         public void TestPathLocation()
32         {
33             Location _expected = _roomBTest;
34             Location _actual = _pathTest.Destination;
35             Assert.AreEqual(_expected, _actual);
36         }
37
38         [Test]
39         public void TestPathNameFullDescription()
40         {
41             string _expected = "A test door";
42             string _actual = _pathTest.FullDescription;
43             Assert.AreEqual(_expected, _actual);
44         }
45
46         [Test]
47         public void TestPathNameShortDescription()
48         {
49
50             string _expected = "a door (north)";
51             string _actual = _pathTest.ShortDescription;
52             Assert.AreEqual(_expected, _actual);
53         }
54
55
56         [Test]
57         public void TestPathNotBlocked()
```

```
58          {
59              bool _actual = _pathTest.IsBlocked;
60              Assert.IsFalse(_actual);
61          }
62
63          [Test]
64          public void TestPathBlocked()
65          {
66              _pathTest.IsBlocked = true;
67              bool _actual = _pathTest.IsBlocked;
68              Assert.IsTrue(_actual);
69          }
70
71          [Test]
72          public void TestLocatePathfromLocation()
73          {
74              Assert.AreEqual(_pathTest, _playerTest.Location.Locate("north"));
75          }
76
77          [Test]
78          public void TestLocateNoPath()
79          {
80              Assert.AreNotEqual(_pathTest, _roomATest.Locate("south"));
81          }
82
83          [Test]
84          public void TestMovePlayer()
85          {
86              _commandTest.Execute(_playerTest, new string[] { "Move", "north" });
87              Assert.AreEqual(_playerTest.Location, _roomBTest);
88          }
89      }
90  }
91
```

IdentifiableObjectTest/PlayerTest.cs

```
1   using System;
2   using Swin_Adventure;
3
4   namespace SwinAdventureTest
5   {
6       [TestFixture]
7       public class PlayerTest
8       {
9           private Player _playerTest;
10          private Item _weaponTest;
11          private Item _armorTest;
12          public Location _locationTest;
13
14          [SetUp]
15          public void Setup()
16          {
17              _playerTest = new Player("thuan", "dan choi", _locationTest);
18              _weaponTest = new Item(new string[] { "weapon" }, "sword", "this is an
    Excalibur");
19              _armorTest = new Item(new string[] { "armor" }, "shield", "this is a
    shield");
20
21
22
```

```
            _playerTest.Inventory.Put(_weaponTest);
            _playerTest.Inventory.Put(_armorTest);
        }

        [Test]
        public void TestPlayerIsIdentifiable()
        {
            Assert.IsTrue(_playerTest.AreYou("me"));
            Assert.IsTrue(_playerTest.AreYou("inventory"));
        }

        [Test]
        public void TestPlayerLocateItems()
        {
            Assert.IsTrue(_playerTest.Locate("weapon") == _weaponTest);
            Assert.IsTrue(_playerTest.Locate("armor") == _armorTest);

            Assert.IsTrue(_playerTest.Inventory.HasItem("weapon"));
            Assert.IsTrue(_playerTest.Inventory.HasItem("armor"));
        }

        [Test]
        public void TestPlayerLocateItself()
        {
            Assert.IsTrue(_playerTest == _playerTest.Locate("me"));
            Assert.IsTrue(_playerTest == _playerTest.Locate("inventory"));
        }

        [Test]
        public void TestPlayerLocateNothing()
        {
            Assert.IsTrue(_playerTest.Locate("helmet") == null);
        }

        [Test]
        public void TestPlayerFullDescription()
        {
            Assert.IsTrue(_playerTest.FullDescription == "You are thuan, dan
choi.\nYou are carrying:\n\ta sword (weapon)\n\ta shield (armor)\n");
        }
    }
}


```

IdentifiableObjectTest/BagTest.cs

```
using System;
namespace Swin_Adventure
{
     [TestFixture]
     public class BagTest
     {
          private Bag _bagTest1;
        private Bag _bagTest2;
        private Item _weaponTest;
        private Item _armorTest;

        [SetUp]
        public void SetUp()
          {
```

```
15                _bagTest1 = new Bag(new string[] { "bag1" }, "backpack", "It's
spacious");
16            _bagTest2 = new Bag(new string[] { "bag2" }, "suitcase", "It's compact");
17            _weaponTest = new Item(new string[] { "weapon" }, "sword", "this is an
Excalibur");
18            _armorTest = new Item(new string[] { "armor" }, "shield", "this is a
shield");
19
20            _bagTest1.Inventory.Put(_bagTest2);
21            _bagTest1.Inventory.Put(_weaponTest);
22            _bagTest2.Inventory.Put(_armorTest);
23        }
24
25        [Test]
26        public void TestBagLocatesItems()
27        {
28            Assert.AreSame(_weaponTest, _bagTest1.Locate("weapon"));
29        }
30
31        [Test]
32        public void TestBagLocatesitself()
33        {
34            Assert.AreSame(_bagTest1, _bagTest1.Locate("bag1"));
35        }
36
37        [Test]
38        public void TestBagLocatesnothing()
39        {
40            Assert.IsNull(_bagTest1.Locate("bag3"));
41        }
42
43        [Test]
44        public void TestBagFullDescription()
45        {
46            Assert.AreEqual("In the backpack you can see:\n\ta suitcase (bag2)\n\ta
sword (weapon)\n", _bagTest1.FullDescription);
47        }
48
49        [Test]
50        public void TestBaginBag()
51        {
52            Assert.AreSame(_bagTest2, _bagTest1.Locate("bag2"));
53            Assert.AreSame(_weaponTest, _bagTest1.Locate("weapon"));
54            Assert.IsNull(_bagTest1.Locate("armor"));
55        }
56    }
57 }
58
59
```

Swin-Adventure/Bag.cs

```
1  using System;
2  namespace Swin_Adventure
3  {
4      public class Bag : Item, IHaveInventory
5      {
6          private Inventory _inventory;
7
8          public Bag(string[] ids, string name, string description) : base(ids, name,
description)
9          {
```

```
            _inventory = new Inventory();
        }

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

        public override string FullDescription
        {
            get { return $"In the {Name} you can see:\n" + _inventory.ItemList; }
        }

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


```

Swin-Adventure/Command.cs

```
using System;
namespace Swin_Adventure
{
    public abstract class Command : IdentifiableObject
    {
        public Command(string[] ids) : base(ids)
        {
        }
         public abstract string Execute(Player player, string[] text);
    }
}


```

Swin-Adventure/IdentifiableObjectClass.cs

```
using System;
namespace Swin_Adventure
{
     public class IdentifiableObject
     {
          private List<string> _identifiers;

          public IdentifiableObject(string[] idents)
          {
               _identifiers = new List<string>(idents);
               _identifiers.AddRange(idents);
          }

          public bool AreYou(string id)
          {
               return _identifiers.Contains(id.ToLower());
          }
```

```
18
19            public string FirstId
20            {
21                 get
22                 {
23                      if (_identifiers.Count == 0)
24                      {
25                           return "";
26                      }
27                      return _identifiers[0];
28                 }
29            }
30
31            public void AddIdentifier(string id)
32            {
33                 _identifiers.Add(id.ToLower());
34            }
35       }
36  }
37
38
```

Swin-Adventure/GameObject.cs

```
1   using System;
2   using System.Xml.Linq;
3
4   namespace Swin_Adventure
5   {
6        public class GameObject : IdentifiableObject
7        {
8           private string _description;
9           private string _name;
10
11          public GameObject(string[] ids, string name, string description) : base(ids)
12          {
13              _description = description;
14              _name = name;
15          }
16
17          public string Name
18            {
19                 get { return _name.ToLower(); }
20            }
21
22            public string ShortDescription
23            {
24              get { return $"a {_name.ToLower()} ({FirstId.ToLower()})"; }
25          }
26
27          public virtual string FullDescription
28            {
29                 get { return _description; }
30            }
31       }
32  }
33
34
```

Swin-Adventure/IHaveInventory.cs

```
using System;
namespace Swin_Adventure
{
    public interface IHaveInventory
    {
        GameObject Locate(string id);
        string Name { get; }
    }
}


```

Swin-Adventure/Inventory.cs

```
using System;
namespace Swin_Adventure
{
     public class Inventory
     {
          private List<Item> _items;
          public Inventory()
          {
               _items = new List<Item>();

          }

          public bool HasItem(string id)
          {
               foreach (Item itm in _items)
               {
                    if (itm.AreYou(id))
                    {
                         return true;
                    }
               }
               return false;
          }

          public void Put(Item itm)
          {
               _items.Add(itm);
          }

        public Item Take(string id)
        {
            Item itm = Fetch(id);

            if (itm != null)
            {
                _items.Remove(itm);
            }

            return itm;
        }

        public Item Fetch(string id)
          {
               foreach (Item itm in _items)
               {
                    if (itm.AreYou(id))
```

```
                    {
                        return itm;
                    }
                }
                return null;
            }

            public string ItemList
            {
                get
                {
                    string list = "";
                 foreach (Item item in _items)
                 {
                    list += "\t" + "a " + item.Name + " (" + item.FirstId + ")\n";
                 }
                 return list;
              }
            }
      }
}
```

Swin-Adventure/Item.cs

```
using System;

namespace Swin_Adventure
{
    public class Item : GameObject
    {
        public Item(string[] idents, string name, string description) : base(idents, name, description)
        {

        }
    }
}
```

Swin-Adventure/Location.cs

```
using System;
using System.Collections.Generic;
using Swin_Adventure;

namespace Swin_Adventure
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        List<Path> _paths;

        public Location(string [] idents,string name, string description) : base(idents, name, description)
        {
            _inventory = new Inventory();
            _paths = new List<Path>();
        }
```

```
17
18          public Inventory Inventory
19          {
20              get { return _inventory; }
21          }
22
23          public GameObject Locate(string id)
24          {
25              if (AreYou(id))
26              {
27                  return this;
28              }
29              foreach (Path path in _paths)
30              {
31                  if (path.AreYou(id))
32                  {
33                      return path;
34                  }
35              }
36              return _inventory.Fetch(id);
37          }
38
39          public override string FullDescription
40          {
41              get
42              {
43                  return "You are at the " + Name + ". " + base.FullDescription + "\n"
    +
44                         "You can see:\n" + Inventory.ItemList;
45              }
46          }
47          public string PathList
48          {
49              get
50              {
51                  string _pathList = "";
52                  foreach (Path path in _paths)
53                  {
54                      _pathList += "\t" + path.ShortDescription + "\n\t" +
    path.FullDescription;
55                  }
56                  return _pathList;
57              }
58          }
59
60          public void AddPath(Path path)
61          {
62              _paths.Add(path);
63          }
64      }
65  }
66
67
```

Swin-Adventure/LookCommand.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using Swin_Adventure;
5
```

```
namespace Swin_Adventure
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" }) { }

        public override string Execute(Player player, string[] text)
        {
            IHaveInventory container = null;
            string itemId;

            if (text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look like that";
            }
            else
            {
                if (text[0] != "look")
                {
                    return "Error in look input";
                }
                if (text[1] != "at")
                {
                    return "What do you want to look at?";
                }
                if (text.Length == 5 && text[3] != "in")
                {
                    return "What do you want to look in?";
                }

                switch (text.Length)
                {
                    case 3:
                        container = player;
                        break;

                    case 5:
                        container = FetchContainer(player, text[4]);

                        if (container == null)
                        {
                            return $"I can't find the {text[4]}";
                        }
                        break;
                }
                itemId = text[2];
                return LookAtIn(itemId, container);
            }
        }

        private IHaveInventory FetchContainer(Player player, string containerId)
        {
            return player.Locate(containerId) as IHaveInventory;
        }

        private string LookAtIn(string thingId, IHaveInventory container)
        {
            GameObject locatedObject = container.Locate(thingId);

            if (locatedObject != null)
```

```
            {
                return locatedObject.FullDescription;
            }
            else
            {
                return $"I can't find the {thingId}";
            }
        }
    }
}

```

Swin-Adventure/MoveCommand.cs

```
using System;
using Swin_Adventure;


namespace Swin_Adventure
{
    public class Move : Command
    {
        public Move() : base(new string[] { "move", "go", "head", "leave" })
        {
        }

        public override string Execute(Player player, string[] text)
        {
            string _moveDirection;

            if (text.Length > 0 && !AreYou(text[0].ToLower()))
            {
                return "Invalid move command. You remain in the same location.";
            }

            switch (text.Length)
            {
                case 1:
                    return "Go: ???";
                case 2:
                    _moveDirection = text[1].ToLower();
                    break;
                case 3:
                    _moveDirection = text[2].ToLower();
                    break;
                default:
                    return "Error in move input.";
            }

            GameObject _path = player.Location.Locate(_moveDirection);
            if (_path != null)
            {
                if (_path.GetType() == typeof(Path))
                {
                    return (_path as Path).Move(player) + ".\r\n\n" +
player.Location.FullDescription;
                }
                else
                {
                    return "Could not find the " + _path.Name;
                }
```

```
47              }
48              else
49              {
50                  return "Invalid path identifier. You remain in the same location.";
51              }
52          }
53      }
54  }
55
```

Swin-Adventure/Path.cs

```
1   using Swin_Adventure;
2   using System;
3   using System.Xml.Linq;
4
5   namespace Swin_Adventure
6   {
7       public class Path : GameObject
8       {
9           private bool _isBlocked;
10          private Location _source, _destination;
11
12          public Path(string[] idents, string name, string desc, Location source,
    Location destination) : base(idents, name, desc)
13          {
14              _source = source;
15              _destination = destination;
16              _isBlocked = false;
17              AddIdentifier("path");
18              foreach (string s in name.Split(' '))
19              {
20                  AddIdentifier(s);
21              }
22          }
23
24          public string Move(Player player)
25          {
26              if (_destination == null)
27              {
28                  return "Cannot move in that direction";
29              }
30              else
31              {
32                  player.Location = _destination;
33                  return $"You have moved to {_destination.Name}";
34              }
35          }
36
37
38          public Location Source { get { return _source; } }
39          public Location Destination { get { return _destination; } }
40          public bool IsBlocked
41          {
42              get { return _isBlocked; }
43              set { _isBlocked = value; }
44          }
45      }
46  }
47
48
```

Swin-Adventure/Player.cs

```
using System;

namespace Swin_Adventure
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        private Location _location;

        public Player(string name, string description, Location location) : base(new string[] {"me", "inventory" }, name, description)
        {
            _inventory = new Inventory();
            _location = location;
        }

        public override string FullDescription
        {
            get
            {
                return "You are " + Name + ", " + base.FullDescription + ".\n" +
                       "You are carrying:\n" + Inventory.ItemList;
            }
        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }

        public Location Location
        {
            get { return _location; }
            set { _location = value; }
        }

        public GameObject Locate(string id)
        {

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


            GameObject found = _inventory.Fetch(id);
            if (found != null)
            {
                return found;
            }


            if (_location != null)
            {
                found = _location.Locate(id);
                if (found != null)
                {
                    return found;
```

```
                }
            }

            return null;
        }
    }
}


```

Swin-Adventure/Program.cs

```
using System;
using Swin_Adventure;

namespace Swin_Adventure;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Welcome to SwinAdventure, designed by Thuan!");

        Console.WriteLine("Enter Player Name: ");
        string playerName = Console.ReadLine();

        Console.WriteLine("Enter your description: ");
        string playerDescription = Console.ReadLine();

        Item item1 = new Item(new string[] { "weapon" }, "sword", "this is an Excalibur");
        Item item2 = new Item(new string[] { "armor" }, "shield", "this is a shield");
        Bag bag = new Bag(new string[] { "bag" }, "bag", "This is a bag.");
        Item itemInBag = new Item(new string[] { "gem" }, "ruby", "This is a beautiful gem");
        Location location1 = new Location(new string[] { "roomA" }, "Room A", "You are in Room A.");
        Location location2 = new Location(new string[] { "roomB" }, "Room B", "You are in Room B.");
        Path path = new Path(new string[] { "north" }, "Door", "A test door", location1, location2);
        Player player = new Player(playerName, playerDescription, location1);

        player.Inventory.Put(item1);
        player.Inventory.Put(item2);
        player.Inventory.Put(bag);
        bag.Inventory.Put(itemInBag);
        location1.Inventory.Put(itemInBag);
        location1.AddPath(path);

        Move moveCommand = new Move();
        LookCommand lookCommand = new LookCommand();

        while (true)
        {
            Console.WriteLine("\nItem at this location:");
            Console.WriteLine($"\n{player.Location.Inventory.ItemList}");

            Console.WriteLine("\nItem in Inventory:");
            Console.WriteLine($"{player.Inventory.ItemList}");

```

```
            Console.WriteLine("\nCommands: look at <item>, look at <item> in <container>, quit");
            Console.WriteLine("Commands: move <path>, exit");
            Console.Write("Enter a command: ");

            string command = Console.ReadLine().ToLower();
            string[] commandParts = command.Split(' ');

            switch (commandParts[0])
            {
                case "look":
                    string result = lookCommand.Execute(player, commandParts);
                    Console.WriteLine("\n" + result);
                    break;
                case "move":
                    if (commandParts.Length >= 2)
                    {
                        string direction = commandParts[1];
                        Console.WriteLine(moveCommand.Execute(player, new string[] { "move", direction }));
                    }
                    else
                    {
                        Console.WriteLine(moveCommand.Execute(player, new string[] { "move" }));
                    }
                    break;
                case "exit":
                    return;
                default:
                    Console.WriteLine("Invalid command. Type 'move <direction>' or 'exit' to quit.");
                    break;
            }
        }
    }
}
```