Swinburne University of Technology

COS20007 Object Oriented Programming

7.2C - Case Study - Iteration 6 - Locations

PDF generated at 01:59 on Thursday 4th May, 2023

```
using System;

namespace SwinAdventure
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _inventory;

        public Location(string name, string desc) : base(new string[] { "house", "here" }, name, desc)
        {
            _inventory = new Inventory();
        }

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

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

    }
}

```

```
using System;
using SwinAdventure;

namespace SwinAdventureTest
{
    public class TestLocation
    {
        Location location;
        Player player;
        Item knife;

        [SetUp]
        public void Setup()
        {
            location = new Location("a jungle", "This is a jungle");
            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 TestLocationFullDesc()
        {
            string expected = "You are in a jungle\nThis is a jungle\nHere you can see:\na sharp knife (knife)\n";
            Assert.That(location.FullDescription, Is.EqualTo(expected));
        }
    }
}

```

```
using System;

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


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

        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            GameObject obj = _inventory.Fetch(id);
            if (obj != null)
            {
                return obj;
            }
            if (_location != null)
            {
                obj = _location.Locate(id);
                return obj;
            }
            else
            {
                return null;
            }
        }

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

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

```
        }

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

```

```
using SwinAdventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class TestPlayer
    {
        Player player;
        Item sword;
        Location location;
        Item stone;


        [SetUp]
        public void Setup()
        {
            player = new Player("bob", "the builder");
            sword = new Item(new string[] { "Sword" }, "a golden sword", "This is a golden sword");
            player.Inventory.Put(sword);

            location = new Location("jungle", "This is a creepy jungle");
            stone = new Item(new string[] { "stone" }, "a stone", "a nice shaped stone");
            location.Inventory.Put(stone);

            player.Location = location;
        }

        [Test]
        public void TestIsIdentifiable()
        {
            Assert.That(player.AreYou("me"), Is.True);
            Assert.That(player.AreYou("inventory"), Is.True);
        }

        [Test]
        public void TestLocateItems()
        {
            Assert.That(player.Locate("sword"), Is.SameAs(sword));
            Assert.That(player.Inventory.HasItem("sword"), Is.True);
        }

        [Test]
        public void TestLocateItself()
        {
            Assert.That(player.Locate("me"), Is.SameAs(player));
            Assert.That(player.Locate("inventory"), Is.SameAs(player));
        }

        [Test]
        public void TestLocateNothing()
        {
```

```
            Assert.That(player.Locate("cucumber"), Is.SameAs(null));
        }

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

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

        [Test]
        public void TestFullDescription()
        {
            Assert.That(player.FullDescription,
                Is.EqualTo("You are bob, the builder.\nYou are carrying:\na golden sword (sword)\n"));
        }
    }
}
```

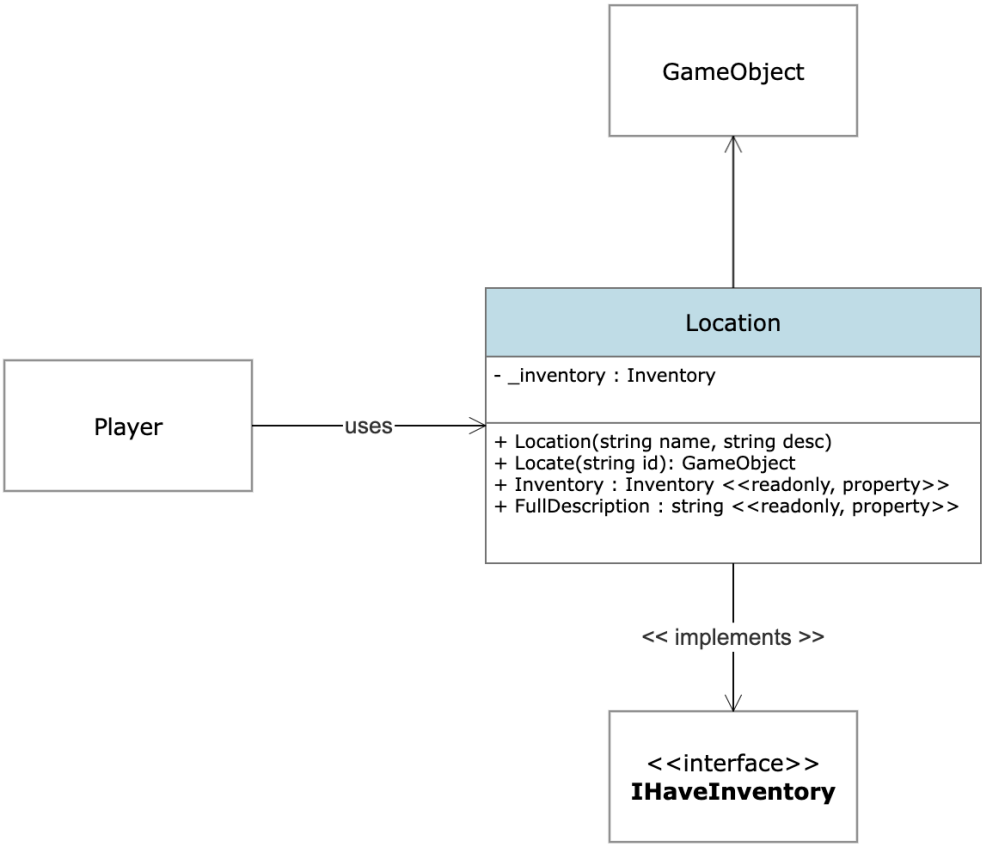
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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

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

            if (text.Length != 1 && text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look like that";
            }
            else
            {
                // The first word must be "look",
                if (text[0] != "look")
                {
                    return "Error in look input";
                }
                if (text[0] == "look" && text.Length == 1)
                {
                    return p.Location.FullDescription;
                }
                // The second word must be "at"
                if (text.Length != 1 && text[1] != "at")
                {
                    return "What do you want to look at?";
                }
                // The fourth word must be "in"
                if (text.Length == 5 && text[3] != "in")
                {
                    return "What do you want to look in?";
                }

                switch (text.Length)
                {
                    case 1:
                        container = p;
                        itemId = "location";
                        break;


                    case 3:
                        container = p;
```

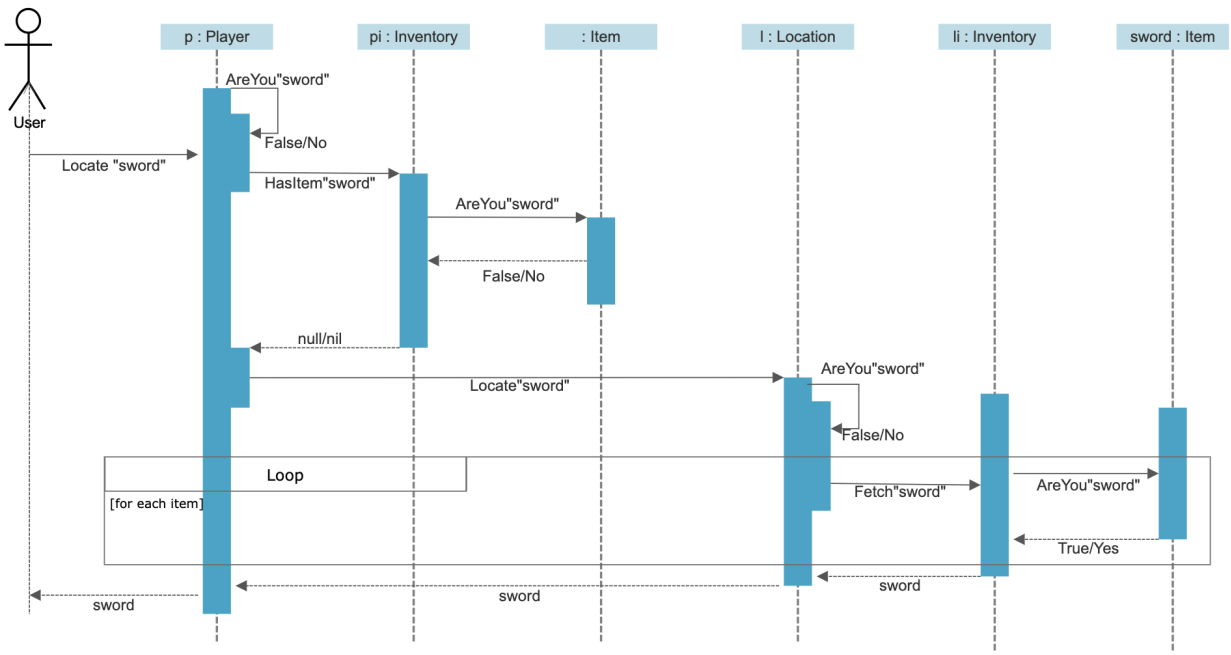
```
                        itemId = text[2];
                        break;

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

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

        public IHaveInventory FetchContainer(Player p, string containerId)
        {
            return p.Locate(containerId) as IHaveInventory;
        }

        public string LookAtIn(string thingId, IHaveInventory container)
        {
            if (container.Locate(thingId) != null)
            {
                return container.Locate(thingId).FullDescription;
            }
            else
            {
                return $"I can't find the {thingId}";
            }
        }
    }
}
```

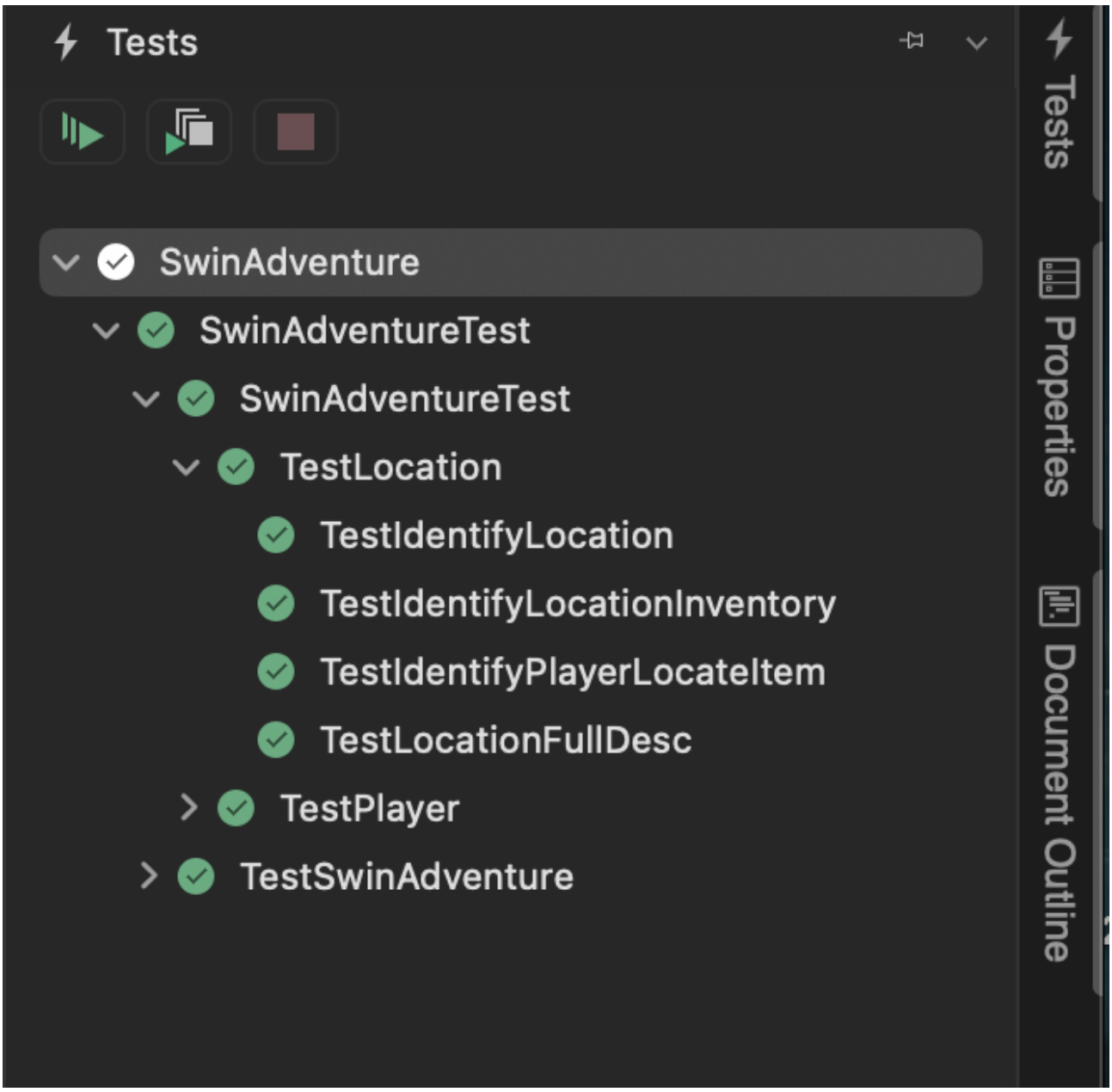
```
using SwinAdventure;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NUnit.Framework;

namespace SwinAdventureTest
{
    [TestFixture]
    public class TestLookCommand
    {
        LookCommand look;
        Player player;
        Bag bag;
        Item gem;
        Location location;

        [SetUp]
        public void Setup()
        {
            look = new LookCommand();
            player = new Player("bob", "the builder");
            bag = new Bag(new string[] { "bag" }, "bag", "This is a bag");
            gem = new Item(new string[] { "gem" }, "big gem", "an expensive item");
            location = new Location("jungle", "This is a creepy jungle");
        }


        [Test]
        public void TestLookAtMe()
        {
            string actual = look.Execute(player, new string[] { "look", "at", "inventory" });
            string expected = "You are bob, the builder.\nYou are carrying:\n";

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

        [Test]
        public void TestLookAtGem()
        {
            //player put gem in inventory
            player.Inventory.Put(gem);

            string actual = look.Execute(player, new string[] { "look", "at", "gem" });
            string expected = "an expensive item";

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

```

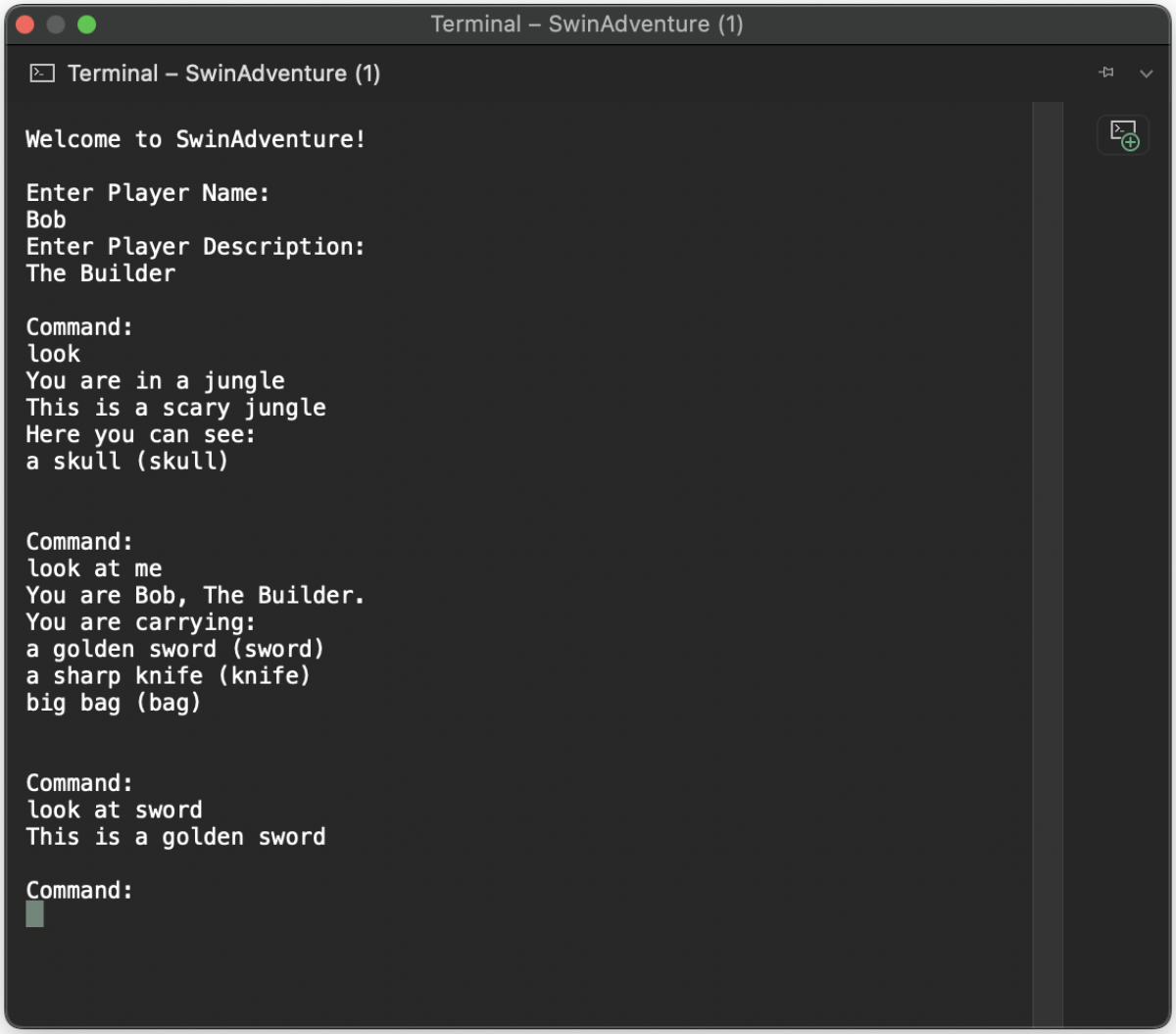
```
        [Test]
        public void TestLookAtUnknown()
        {
            string actual = look.Execute(player, new string[] { "look", "at", "gem" });
            string expected = "I can't find the gem";

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

        [Test]
        public void TestLookAtGemInMe()
        {
            //look at gem in inventory
            player.Inventory.Put(gem);

            string actual = look.Execute(player, new string[] { "look", "at", "gem", "in", "inventory" });
            string expected = "an expensive item";

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

        [Test]
        public void TestLookAtGemInBag()
        {
            //put gem in bag, then put bag in player's inventory
            bag.Inventory.Put(gem);
            player.Inventory.Put(bag);

            string actual = look.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });
            string expected = "an expensive item";

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

        [Test]
        public void TestLookAtGemInNoBag()
        {
            bag.Inventory.Put(gem);

            string actual = look.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });
            string expected = "I can't find the bag";

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

        // test looking at non existent item in your bag
        [Test]
        public void TestLookAtNoGemInBag()
        {
```

```
            player.Inventory.Put(bag);

            string actual = look.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });
            string expected = "I can't find the gem";

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

        [Test]
        public void TestPlayerLocation()
        {
            player.Location = location;
            string actual = look.Execute(player, new string[] { "look" });
            Assert.That(actual, Is.EqualTo(location.FullDescription));
        }

        [Test]
        public void TestInvalidLook()
        {
            string actual = look.Execute(player, new string[] { "hello" });
            Assert.That(actual, Is.EqualTo("Error in look input"));
        }

        [Test]
        public void TestInvalidAt()
        {
            string actual = look.Execute(player, new string[] { "look", "in", "gem" });
            Assert.That(actual, Is.EqualTo("What do you want to look at?"));
        }

        [Test]
        public void TestInvalidIn()
        {
            string actual = look.Execute(player, new string[] { "look", "at", "gem", "at", "bag" });
            Assert.That(actual, Is.EqualTo("What do you want to look in?"));
        }

    }
}
```


GameObject
Location
- _inventory : Inventory
+ Location(string name, string desc)
+ Locate(string id): GameObject
+ Inventory : Inventory <<readonly, property>>
+ FullDescription : string <<readonly, property>>
Player
uses
<< implements >>
<<interface>>
IHaveInventory


User
p : Player
pi : Inventory
: Item
l : Location
li : Inventory
sword : Item
AreYou"sword"
False/No
Locate "sword"
HasItem"sword"
AreYou"sword"
False/No
null/nil
Locate"sword"
AreYou"sword"
False/No
Loop
[for each item]
Fetch"sword"
AreYou"sword"
True/Yes
sword
sword
sword


Tests
SwinAdventure
SwinAdventureTest
SwinAdventureTest
TestLocation
TestIdentifyLocation
TestIdentifyLocationInventory
TestIdentifyPlayerLocateItem
TestLocationFullDesc
TestPlayer
TestSwinAdventure
Tests
Properties
Document Outline


Terminal – SwinAdventure (1)
Terminal – SwinAdventure (1)
Welcome to SwinAdventure!
Enter Player Name:
Bob
Enter Player Description:
The Builder
Command:
look
You are in a jungle
This is a scary jungle
Here you can see:
a skull (skull)
Command:
look at me
You are Bob, The Builder.
You are carrying:
a golden sword (sword)
a sharp knife (knife)
big bag (bag)
Command:
look at sword
This is a golden sword
Command: