COS20007: Object Oriented Programming

Pass Task 6.1: Case Study — Iteration 4: Look Command

Show Wai Yan/105293041

# Command.cs

namespace SwinAdventure

{

public abstract class Command : IndentifiableObject

{

public Command(string[] ids) : base(ids)

{

}

public abstract string Execute(Player p, string[] text);

}

}

# LookCommand.cs

namespace SwinAdventure

{

public class LookCommand : Command

{

public LookCommand()

: base(new string[] { "look", "inventory", "inv" }) { }

public override string Execute(Player p, string[] text)

{

IHaveInventory container = null;

string containerId = null;

string itemId = null;

switch (text.Length)

{

case 1: // inventory case

if (text[0].ToLower() == "inventory" || text[0].ToLower() == "inv")

{

container = p;

itemId = "me";

break;

}

else

return "Error in look input";

case 3: // look at <something>

if (text[0].ToLower() != "look")

return "Error in look input";

if (text[1].ToLower() != "at")

return "What do you want to look at?";

container = p;

itemId = text[2].ToLower();

break;

case 5: // look at <something> in <something>

if (text[0].ToLower() != "look")

return "Error in look input";

if (text[3].ToLower() != "in")

return "What do you want to look in?";

containerId = text[4].ToLower();

container = FetchContainer(p, containerId);

itemId = text[2].ToLower();

break;

default:

return "I don\'t know how to look like that";

}

return LookAtIn(itemId, container);

}

private IHaveInventory FetchContainer(Player p, string containerId)

{

return p.Locate(containerId) as IHaveInventory;

}

private string LookAtIn(string thingId, IHaveInventory container)

{

if (container == null)

return "I cannot find the bag";

if (container.Locate(thingId) == null)

return $"I cannot find the {thingId} in the {container.Name}";

return container.Locate(thingId).FullDescription;

}

}

}

# IHaveInventory.cs

namespace SwinAdventure

{

public interface IHaveInventory

{

public GameObject Locate(string id);

public string Name

{

get;

}

}

}

# Player.cs

namespace SwinAdventure

{

public class Player : GameObject, IHaveInventory

{

// Field

private Inventory \_inventory = new Inventory();

// Constructor

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

{

}

// Properties

public override string FullDescription

{

get { return $"You are {Name} {base.FullDescription}\nYou are carrying\n {Inventory.ItemList}"; }

}

public Inventory Inventory

{

get { return \_inventory; }

}

// Methods

public GameObject Locate(string id)

{

if (AreYou(id)) return this;

return Inventory.Fetch(id);

}

}

}

# Bag.cs

namespace SwinAdventure

{

public class Bag : Item, IHaveInventory

{

// Fields

private Inventory \_inventory;

// Constructor

public Bag(string[] ids, string name, string desc) : base(ids, name, desc)

{

\_inventory = new Inventory();

}

// Methods

public GameObject Locate(string id)

{

if (AreYou(id)) return this;

return \_inventory.Fetch(id);

}

// Properties

public override string FullDescription

{

get { return $"{base.FullDescription}.\nYou look in the {Name.ToLower()} and see:\n{\_inventory.ItemList}"; }

}

public Inventory Inventory

{

get { return \_inventory; }

}

}

}

# TestLookCommand.cs

using System;

using NUnit.Framework;

using NUnit.Framework.Legacy;

using SwinAdventure;

namespace UnitTests

{

[TestFixture]

public class TestLockCommand

{

private LookCommand look;

private Player testPlayer;

private Bag bag;

private Item gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");

private Item shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel");

private Item diamond = new Item(

new string[] { "diamond" },

"a diamond",

"This is a diamond"

);

[SetUp]

public void Setup()

{

look = new LookCommand();

testPlayer = new Player("Show", "The Programmer");

bag = new Bag(

new string[] { "bag", "backpack", "leather bag" },

"Leather Bag",

"A sturdy leather bag to carry items"

);

testPlayer.Inventory.Put(bag);

}

[Test]

public void TestLookAtMe()

{

string excepted = testPlayer.FullDescription;

string testOutPut = look.Execute(

testPlayer,

new string[] { "look", "at", "Inventory" }

);

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void TestLookAtGem()

{

string excepted = gem.FullDescription;

testPlayer.Inventory.Put(gem);

string testOutPut = look.Execute(testPlayer, new string[] { "look", "at", "Gem" });

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void TestLookAtUnk()

{

string excepted = $"I cannot find the gem in the {testPlayer.Name}";

string testOutPut = look.Execute(testPlayer, new string[] { "look", "at", "Gem" });

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void TestLookAtGemInMe()

{

string excepted = gem.FullDescription;

testPlayer.Inventory.Put(gem);

string testOutPut = look.Execute(

testPlayer,

new string[] { "look", "at", "Gem", "in", "me" }

);

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void TestLookAtGemInBag()

{

string excepted = gem.FullDescription;

bag.Inventory.Put(gem);

string testOutPut = look.Execute(

testPlayer,

new string[] { "look", "at", "Gem", "in", "bag" }

);

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void TestLookAtGemInNoBag()

{

string excepted = "I cannot find the bag";

Player noBagPlayer = new Player("Ricky", "I have No Bag bro");

string testOutPut = look.Execute(

noBagPlayer,

new string[] { "look", "at", "Gem", "in", "bag" }

);

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void TestLookAtNoGemInBag()

{

string excepted = $"I cannot find the gem in the {bag.Name}";

string testOutPut = look.Execute(

testPlayer,

new string[] { "look", "at", "Gem", "in", "bag" }

);

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

[Test]

public void InvalidLook()

{

string excepted = "I don\'t know how to look like that";

string testOutPut = look.Execute(testPlayer, new string[] { "look", "around" });

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

excepted = "I don\'t know how to look like that";

testOutPut = look.Execute(testPlayer, new string[] { "hello", "105293041" });

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

excepted = $"I cannot find the show wai yan in the {testPlayer.Name}";

testOutPut = look.Execute(testPlayer, new string[] { "look", "at", "Show Wai Yan" });

ClassicAssert.That(testOutPut, Is.EqualTo(excepted));

}

}

}

# Screenshot of the Test Explorer showing your unit test running

A screenshot of a computer

AI-generated content may be incorrect.