Swinburne University of Technology

COS20007 Object Oriented Programming

4.2P - Case Study - Iteration 2 - Players Items and Inventory

PDF generated at 23:30 on Friday 7th April, 2023

```
using System;

namespace SwinAdventure
{
    public class GameObject : IdentifiableObject
    {
        public GameObject(string[] ids, string name, string desc) : base(ids)
        {
            _description = desc;
            _name = name;
        }

        private string _description;
        private string _name;

        public string Name
        {
            get
            {
                return _name;
            }
        }

        public virtual string FullDescription
        {
            get
            {
                return _description;
            }
        }

        public string ShortDescription
        {
            get
            {
                return $"{Name} ({FirstId})";
            }
        }
    }
}
```

```
using System;

namespace SwinAdventure
{
    public class Player : GameObject
    {
        private Inventory _inventory;

        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;
            }
            return _inventory.Fetch(id);
        }

        public override string FullDescription
        {
            get
            {
                return $"You are {Name}, ({base.FullDescription}), you are
↪   carrying:\n" + _inventory.ItemList;
            }
        }

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

```
namespace SwinAdventure
{
    public class PlayerTest
    {
        private Player _testPlayer;
        private Item _sword;
        private Item _shovel;
        private Item _pc;

        [SetUp]
        public void SetUp()
        {
            _testPlayer = new Player("Trung Kien Nguyen", "I am the player");

            _sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword", "This is a bronze sword");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel");
            _pc = new Item(new string[] { "pc", "computer" }, "a small computer", "This is a small computer");

            _testPlayer.Inventory.Put(_sword);
            _testPlayer.Inventory.Put(_shovel);
            _testPlayer.Inventory.Put(_pc);
        }

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

        [Test]
        public void TestPlayerLocatesItems()
        {
            GameObject locatedItem1 = _testPlayer.Locate("shovel");
            GameObject locatedItem2 = _testPlayer.Locate("pc");

            // Test if player has the located item
            Assert.AreEqual(locatedItem1, _shovel);
            Assert.AreEqual(locatedItem2, _pc);

            // Test if the item remains in the player's inventory
            Assert.IsTrue(_testPlayer.Inventory.HasItem("shovel"));
            Assert.IsTrue(_testPlayer.Inventory.HasItem("pc"));
        }

        [Test]
        public void TestPlayerLocatesItself()
        {
            GameObject playerItself1 = _testPlayer.Locate("me");
            GameObject playerItself2 = _testPlayer.Locate("inventory");
```

```
51
52              // Test if player has the located itself with the keyword "me"
53              Assert.AreEqual(playerItself1, _testPlayer);
54
55              // Test if player has the located itself with the keyword "inventory"
56              Assert.AreEqual(playerItself2, _testPlayer);
57          }
58
59          [Test]
60          public void TestPlayerLocatesNothing()
61          {
62              GameObject nonExistentObject = _testPlayer.Locate("gun");
63
64              Assert.AreEqual(nonExistentObject, null);
65          }
66
67          [Test]
68          public void TestPlayerFullDescription()
69          {
70              string playerFullDesc = $"You are {_testPlayer.Name}, (I am the player),
    ↪  you are carrying:\n{_testPlayer.Inventory.ItemList}";
71
72              Assert.AreEqual(playerFullDesc, _testPlayer.FullDescription);
73          }
74      }
75  }
```

```
using System;

namespace SwinAdventure
{
    public class Item : GameObject
    {
        public Item(string[] idents, string name, string desc) : base(idents, name,
↪   desc)
        {
        }
    }
}
```

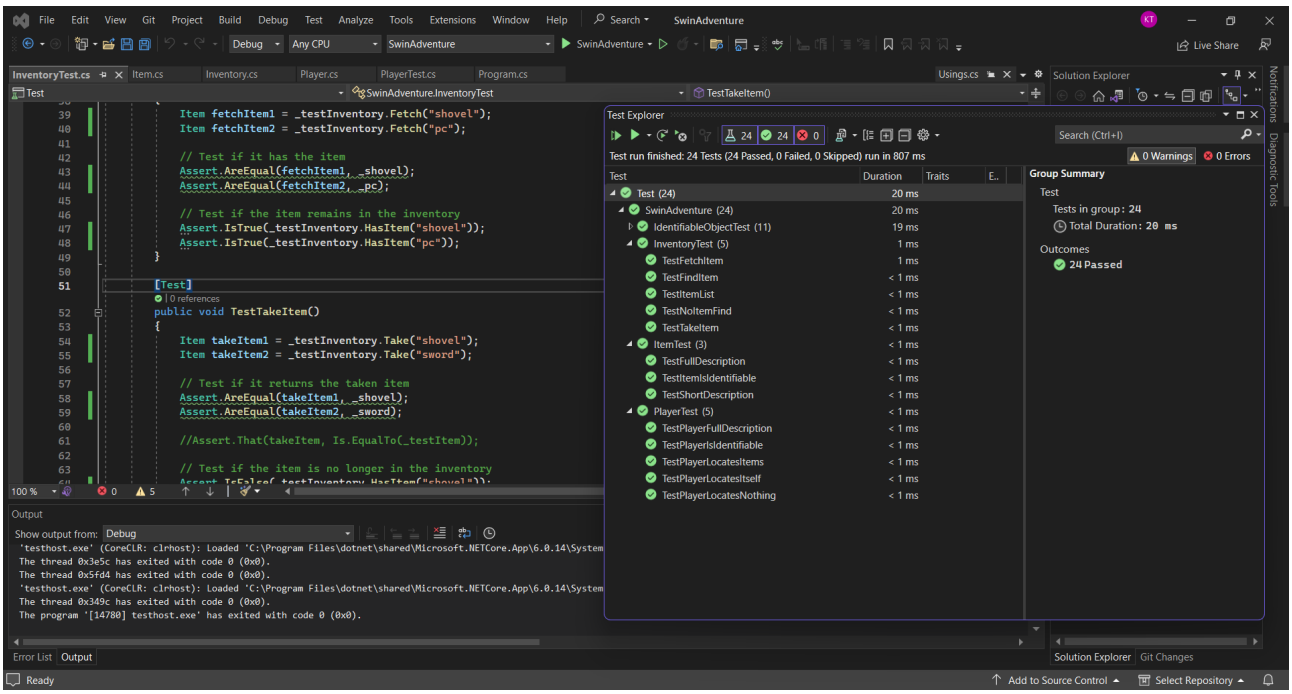
```
namespace SwinAdventure
{
    public class ItemTest
    {
        private Item _sword;

        [SetUp]
        public void SetUp()
        {
            _sword = new Item(new string[] { "sword" }, "a bronze sword", "This is a bronze sword" );
        }

        [Test]
        public void TestItemIsIdentifiable()
        {
            Assert.IsTrue(_sword.AreYou("sword"));
            Assert.IsFalse(_sword.AreYou("gun"));
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.AreEqual(_sword.ShortDescription, "a bronze sword (sword)");
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.AreEqual(_sword.FullDescription, "This is a bronze sword");
        }
    }
}
```

```
using System;

namespace SwinAdventure
{
    public class Inventory
    {
        public Inventory()
        {
            _items = new List<Item>();
        }

        private List<Item> _items;

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

            return false;
        }

        public void Put(Item item)
        {
            _items.Add(item);
        }
        public Item Take(string id)
        {
            Item takenItem = Fetch(id);
            _items.Remove(takenItem);
            return takenItem;
        }
        public Item Fetch(string id)
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id))
                {
                    return item;
                }
            }
            return null;
        }

        public string ItemList
        {
            get
            {
                string result = "";
```

```
54                  foreach (Item item in _items)
55                  {
56                      result += "    - " + item.ShortDescription + "\n";
57                  }
58                  return result;
59              }
60          }
61      }
62  }
```

```
namespace SwinAdventure
{
    public class InventoryTest
    {
        private Inventory _testInventory;
        private Item _sword;
        private Item _shovel;
        private Item _pc;

        [SetUp]
        public void SetUp()
        {
            _testInventory = new Inventory();

            _sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword", "This is a bronze sword");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel");
            _pc = new Item(new string[] { "pc", "computer" }, "a small computer", "This is a small computer");

            _testInventory.Put(_sword);
            _testInventory.Put(_shovel);
            _testInventory.Put(_pc);
        }

        [Test]
        public void TestFindItem()
        {
            Assert.IsTrue(_testInventory.HasItem("sword"));
        }

        [Test]
        public void TestNoItemFind()
        {
            Assert.IsFalse(_testInventory.HasItem("gun"));
        }

        [Test]
        public void TestFetchItem()
        {
            Item fetchItem1 = _testInventory.Fetch("shovel");
            Item fetchItem2 = _testInventory.Fetch("pc");

            // Test if it has the item
            Assert.AreEqual(fetchItem1, _shovel);
            Assert.AreEqual(fetchItem2, _pc);

            // Test if the item remains in the inventory
            Assert.IsTrue(_testInventory.HasItem("shovel"));
            Assert.IsTrue(_testInventory.HasItem("pc"));
        }

```

```
51          [Test]
52          public void TestTakeItem()
53          {
54              Item takeItem1 = _testInventory.Take("shovel");
55              Item takeItem2 = _testInventory.Take("sword");
56
57              // Test if it returns the taken item
58              Assert.AreEqual(takeItem1, _shovel);
59              Assert.AreEqual(takeItem2, _sword);
60
61              //Assert.That(takeItem, Is.EqualTo(_testItem));
62
63              // Test if the item is no longer in the inventory
64              Assert.IsFalse(_testInventory.HasItem("shovel"));
65              Assert.IsFalse(_testInventory.HasItem("sword"));
66          }
67
68          [Test]
69          public void TestItemList()
70          {
71              string expectedResult = "   - a bronze sword (sword)\n   - a shovel
    ↪  (shovel)\n   - a small computer (pc)\n";
72              Assert.AreEqual(expectedResult, _testInventory.ItemList);
73          }
74      }
75  }
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
SwinAdventure
Debug
Any CPU
Live Share
InventoryTest.cs
Item.cs
Inventory.cs
Player.cs
PlayerTest.cs
Program.cs
Usings.cs
Solution Explorer
Test
SwinAdventure.InventoryTest
TestTakeItem()
Item fetchItem1 = _testInventory.Fetch("shovel");
Item fetchItem2 = _testInventory.Fetch("pc");
// Test if it has the item
Assert.AreEqual(fetchItem1, _shovel);
Assert.AreEqual(fetchItem2, _pc);
// Test if the item remains in the inventory
Assert.IsTrue(_testInventory.HasItem("shovel"));
Assert.IsTrue(_testInventory.HasItem("pc"));
}
[Test]
0 references
public void TestTakeItem()
{
Item takeItem1 = _testInventory.Take("shovel");
Item takeItem2 = _testInventory.Take("sword");
// Test if it returns the taken item
Assert.AreEqual(takeItem1, _shovel);
Assert.AreEqual(takeItem2, _sword);
//Assert.That(takeItem, Is.EqualTo(_testItem));
// Test if the item is no longer in the inventory
Test Explorer
Search (Ctrl+I)
Test run finished: 24 Tests (24 Passed, 0 Failed, 0 Skipped) run in 807 ms
0 Warnings
0 Errors
Test
Duration
Traits
Test (24) 20 ms
SwinAdventure (24) 20 ms
IdentifiableObjectTest (11) 19 ms
InventoryTest (5) 1 ms
TestFetchItem 1 ms
TestFindItem < 1 ms
TestItemList < 1 ms
TestNoItemFind < 1 ms
TestTakeItem < 1 ms
ItemTest (3) < 1 ms
TestFullDescription < 1 ms
TestItemIsIdentifiable < 1 ms
TestShortDescription < 1 ms
PlayerTest (5) < 1 ms
TestPlayerFullDescription < 1 ms
TestPlayerIsIdentifiable < 1 ms
TestPlayerLocatesItems < 1 ms
TestPlayerLocatesItself < 1 ms
TestPlayerLocatesNothing < 1 ms
Group Summary
Test
Tests in group: 24
Total Duration: 20 ms
Outcomes
24 Passed
Output
Show output from: Debug
'testhost.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.14\System
The thread 0x3e5c has exited with code 0 (0x0).
The thread 0x5fd4 has exited with code 0 (0x0).
'testhost.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.14\System
The thread 0x349c has exited with code 0 (0x0).
The program '[14780] testhost.exe' has exited with code 0 (0x0).
Error List
Output
Solution Explorer
Git Changes
Ready
Add to Source Control
Select Repository