Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 2 - Players Items and Inventory

PDF generated at 18:57 on Sunday 21st May, 2023

```
using System;

namespace SwinAdventure
{
    public class GameObject : IdentifiableObject
    {
        string _description;
        string _name;

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

        public string Name
        {
            get { return _name; }
        }

        public string Description
        {
            get { return _description; }
        }

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

        }

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

```

```
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 override string FullDescription
        {
            get { return $"{Name}, {base.FullDescription}, you are carrying:
↪   {_inventory.ItemList}"; }

        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }


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


    }
}

```

```
using System;
using System.Numerics;
using System.Xml.Linq;
using NUnit.Framework;

namespace SwinAdventure
{
    [TestFixture]

    public class TestPlayer
    {
        Player player;
        Item shield;
        Item axe;
        Inventory inventory;


        [SetUp]
        public void SetUp()
        {
            player = new Player("andy", "music producer");
            shield = new Item(new string[] { "shield" }, "a shield", "this is a shield");
            axe = new Item(new string[] { "axe" }, "an axe", "this is an axe");
            inventory = new();

            player.Inventory.Put(shield);
            player.Inventory.Put(axe);

        }

        [Test]
        public void TestPlayerIsIdentifiable()
        {
            Assert.IsTrue(player.AreYou("me"));
            Assert.IsTrue(player.AreYou("Inventory"));

        }

        [Test]
        public void TestPlayerLocatesItems()
        {


            Assert.That(player.Inventory.HasItem("shield"), Is.True);
            Assert.That(player.Inventory.HasItem("axe"), Is.True);

            Assert.That(player.Inventory.HasItem("shield"), Is.True);
            Assert.That(player.Inventory.HasItem("axe"), Is.True);

        }

        [Test]
```

```
53          public void TestPlayerLocatesItself()
54          {
55              Assert.That(player, Is.EqualTo(player.Locate("me")));
56              Assert.That(player, Is.EqualTo(player.Locate("inventory")));
57
58
59          }
60
61          [Test]
62          public void TestPlayerLocatesNothing()
63          {
64              Assert.That(player.Locate("macbookpro"), Is.EqualTo(null));
65
66          }
67
68          [Test]
69          public void TestPlayerFullDescriotion()
70          {
71              Assert.That(player.FullDescription, Is.EqualTo($"andy, music producer,
    ↪  you are carrying: {player.Inventory.ItemList}"));
72
73          }
74      }
75  }
```

```
using System;

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

        }
    }
}

```

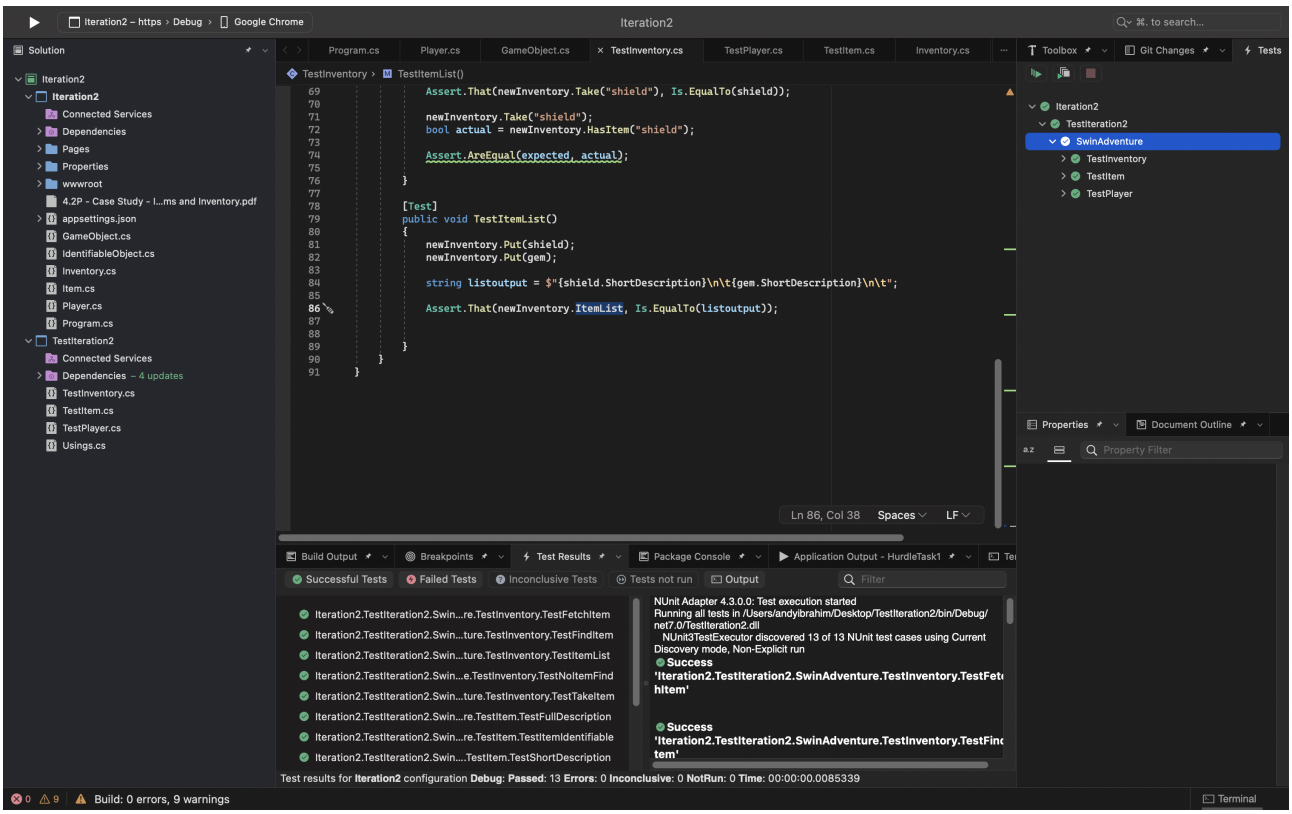
```
using System;
using NUnit.Framework;

namespace SwinAdventure
{
    [TestFixture]

    public class TestItem
    {
        Item axe;

        [SetUp]
        public void SetUp()
        {

            axe = new Item(new string[] { "axe" }, "an axe", "this is an axe");

        }

        [Test]
        public void TestItemIdentifiable()
        {
            Assert.IsTrue(axe.AreYou("axe"));

        }

        [Test]
        public void TestShortDescription()
        {
            Assert.AreEqual(axe.ShortDescription, "an axe (axe)");


        }

        [Test]
        public void TestFullDescription()
        {
            Assert.AreEqual(axe.FullDescription, "this is an axe");

        }


    }
}
```

```
using System;

namespace SwinAdventure
{
    public class Inventory
    {
        private List<Item> _items;

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

        public string ItemList
        {
            get
            {
                string list = "";
                foreach (Item i in _items)
                {
                    list += i.ShortDescription + "\n\t";
                }

                return list;
            }
        }

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

                }
            }

            return false;

        }

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

        }

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

            _items.Remove(take_item);

```

```
54              return take_item;
55          }
56
57          public Item Fetch(string id)
58          {
59              foreach (Item i in _items)
60              {
61                  if (i.AreYou(id))
62                  {
63                      return i;
64                  }
65              }
66              return null;
67
68          }
69
70      }
71  }
72
```

```
using System;
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventure
{
    [TestFixture]

    public class TestInventory
    {
        Item shield;
        Item gem;
        Inventory newInventory;

        [SetUp]
        public void SetUp()
        {
            shield = new Item(new string[] { "shield" }, "a shield", "this is a shield");
            gem = new Item(new string[] { "gem" }, "a gem", "this is a gem");
            newInventory = new Inventory();

        }

        [Test]
        public void TestFindItem()
        {
            newInventory.Put(shield);
            bool expected = true;
            bool actual = newInventory.HasItem("shield");

            Assert.AreEqual(expected, actual);


        }

        [Test]
        public void TestNoItemFind()
        {
            newInventory.Put(shield);

            bool expected = false;
            bool actual = newInventory.HasItem("randomitem");

            Assert.AreEqual(expected, actual);

        }

        [Test]
        public void TestFetchItem()
        {
            newInventory.Put(shield);
            bool expected = true;
```

```
53
54              Assert.That(newInventory.Fetch("shield"), Is.EqualTo(shield));
55
56              newInventory.Fetch("shield");
57              bool actual = newInventory.HasItem("shield");
58
59              Assert.AreEqual(expected, actual);
60
61          }
62
63          [Test]
64          public void TestTakeItem()
65          {
66              newInventory.Put(shield);
67              bool expected = false;
68
69              Assert.That(newInventory.Take("shield"), Is.EqualTo(shield));
70
71              newInventory.Take("shield");
72              bool actual = newInventory.HasItem("shield");
73
74              Assert.AreEqual(expected, actual);
75
76          }
77
78          [Test]
79          public void TestItemList()
80          {
81              newInventory.Put(shield);
82              newInventory.Put(gem);
83
84              string listoutput =
    ↪  $"{shield.ShortDescription}\n\t{gem.ShortDescription}\n\t";
85
86              Assert.That(newInventory.ItemList, Is.EqualTo(listoutput));
87
88
89          }
90      }
91  }
```


Iteration2 – https › Debug › Google Chrome
Iteration2
Solution
Iteration2
Iteration2
Connected Services
Dependencies
Pages
Properties
wwwroot
4.2P - Case Study - I...ms and Inventory.pdf
appsettings.json
GameObject.cs
IdentifiableObject.cs
Inventory.cs
Item.cs
Player.cs
Program.cs
TestIteration2
Connected Services
Dependencies – 4 updates
TestInventory.cs
TestItem.cs
TestPlayer.cs
Usings.cs
Program.cs
Player.cs
GameObject.cs
TestInventory.cs
TestPlayer.cs
TestItem.cs
Inventory.cs
TestInventory › TestItemList()
Assert.That(newInventory.Take("shield"), Is.EqualTo(shield));
newInventory.Take("shield");
bool actual = newInventory.HasItem("shield");
Assert.AreEqual(expected, actual);
}
[Test]
public void TestItemList()
{
newInventory.Put(shield);
newInventory.Put(gem);
string listoutput = $"{shield.ShortDescription}\n\t{gem.ShortDescription}\n\t";
Assert.That(newInventory.ItemList, Is.EqualTo(listoutput));
}
}
}
Ln 86, Col 38
Spaces
LF
Toolbox
Git Changes
Tests
Iteration2
TestIteration2
SwinAdventure
TestInventory
TestItem
TestPlayer
Properties
Document Outline
Property Filter
Build Output
Breakpoints
Test Results
Package Console
Application Output - HurdleTask1
Successful Tests
Failed Tests
Inconclusive Tests
Tests not run
Output
Filter
Iteration2.TestIteration2.Swin...re.TestInventory.TestFetchItem
Iteration2.TestIteration2.Swin...ture.TestInventory.TestFindItem
Iteration2.TestIteration2.Swin...ture.TestInventory.TestItemList
Iteration2.TestIteration2.Swin...e.TestInventory.TestNoItemFind
Iteration2.TestIteration2.Swin...ture.TestInventory.TestTakeItem
Iteration2.TestIteration2.Swin...re.TestItem.TestFullDescription
Iteration2.TestIteration2.Swin...re.TestItem.TestItemIdentifiable
Iteration2.TestIteration2.Swin....TestItem.TestShortDescription
NUnit Adapter 4.3.0.0: Test execution started
Running all tests in /Users/andyibrahim/Desktop/TestIteration2/bin/Debug/net7.0/TestIteration2.dll
NUnit3TestExecutor discovered 13 of 13 NUnit test cases using Current Discovery mode, Non-Explicit run
Success
'Iteration2.TestIteration2.SwinAdventure.TestInventory.TestFetchItem'
Success
'Iteration2.TestIteration2.SwinAdventure.TestInventory.TestFindItem'
Test results for Iteration2 configuration Debug: Passed: 13 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0085339
0 9
Build: 0 errors, 9 warnings
Terminal