Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 2 - Players Items and Inventory

PDF generated at 18:42 on Tuesday 10th October, 2023

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

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

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

        public string Name
        {
            get { return _name; }
        }

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

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

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using SwinAdventure;
using System.Xml.Linq;

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}.\n You are
↪   carrying:{_inventory.ItemList}";
            }
        }

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

```

```
using System;
using System.Collections.Generic;
using System.Text;
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventureTests
{
    [TestFixture]
    public class PlayerTest
    {
        private Player _player;
        private Item _item;

        [SetUp]
        public void SetUp()
        {
            _player = new Player("Lorraine", "A test player");
            _item = new Item(new string[] { "sword" }, "Sword", "A sharp sword");
            _player.Inventory.Put(_item);
        }

        [Test]
        public void TestPlayerIsIdentifiable()
        {
            // Test whether the player responds correctly to "Are You" requests
            Assert.IsTrue(_player.AreYou("me"));
            Assert.IsTrue(_player.AreYou("inventory"));
            Assert.IsFalse(_player.AreYou("spade"));
        }

        [Test]
        public void TestPlayerLocatesItems()
        {
            // Test whether the player can locate items in its inventory
            GameObject founditem = _player.Locate("sword");
            Assert.AreEqual(_item, founditem);
        }

        [Test]
        public void TestPlayerLocatesItself()
        {
            // Test whether the player returns itself when asked to locate "me" or
↪   "inventory"
            GameObject founditem1 = _player.Locate("me");
            GameObject founditem2 = _player.Locate("inventory");

            Assert.AreEqual(_player, founditem1);
            Assert.AreEqual(_player, founditem2);
        }

        [Test]
        public void TestPlayerLocatesNothing()
```

```
53          {
54              // Test whether the player returns null/nil when asked to locate
    ↪  something it does not have
55              GameObject founditem = _player.Locate("spade");
56              Assert.IsNull(founditem);
57          }
58
59          [Test]
60          public void TestPlayerFullDescription()
61          {
62              // Test whether the player's full description contains the expected text
63              string expectedFullDescription = "You are Lorraine A test player.\n You
    ↪  are carrying:\n\ta Sword (sword)";
64              Assert.AreEqual(expectedFullDescription, _player.FullDescription);
65          }
66      }
67  }
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

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

        }
    }
}
```

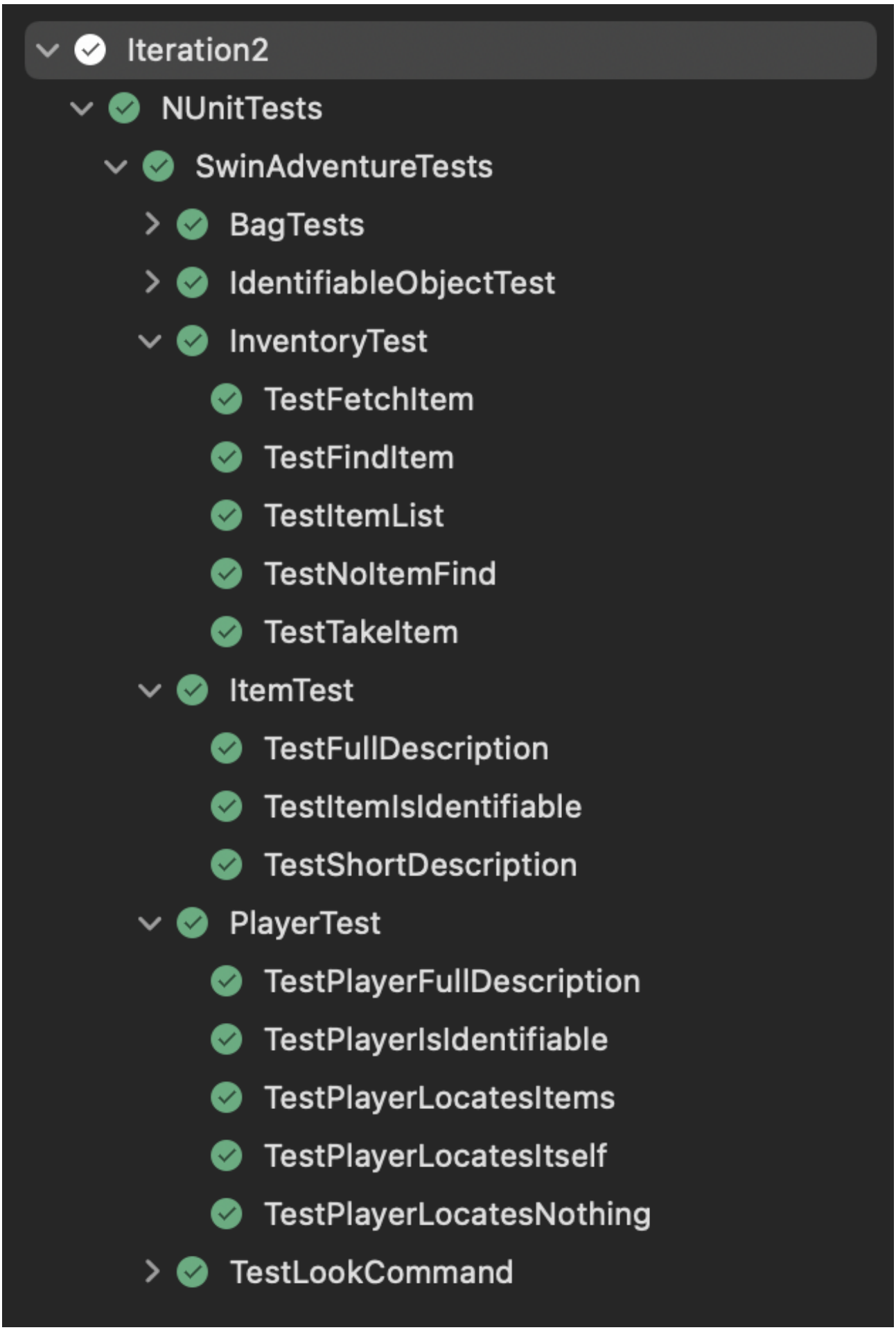
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using NUnit.Framework;
using SwinAdventure;


namespace SwinAdventureTests
{
    [TestFixture]
    public class ItemTest
    {
        private Item _item;

        [SetUp]
        public void SetUp()
        {
            _item = new Item(new string[] { "shovel" }, "Shovel", "This is a shovel");
        }

        [Test]
        public void TestItemIsIdentifiable()
        {
            // Test whether the item is identifiable by its identifiers
            Assert.IsTrue(_item.AreYou("shovel"));
            Assert.IsFalse(_item.AreYou("spade"));
        }

        [Test]
        public void TestShortDescription()
        {
            // Test whether the short description is in the expected format
            string expectedShortDescription = "a Shovel (shovel)";
            Assert.That(_item.ShortDescription, Is.EqualTo(expectedShortDescription));
        }

        [Test]
        public void TestFullDescription()
        {
            // Test whether the full description matches the item's description
            string expectedFullDescription = "This is a shovel";
            Assert.That(_item.FullDescription, Is.EqualTo(expectedFullDescription));
        }
    }
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

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

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

        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 takeItem = Fetch(id);
            if (takeItem != null)
            {
                _items.Remove(takeItem);
            }
            return takeItem;
        }

        public Item Fetch(string id)
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id)) return item;
            }
            return null;
        }

        public string ItemList
        {
            get
            {
```

```
54                  string listItem = "";
55                  foreach (Item item in _items)
56                  {
57                      listItem += "\n\t"+ item.ShortDescription;
58                  }
59                  return listItem;
60              }
61          }
62      }
63  }
64
65
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventureTests
{
    [TestFixture]
    public class InventoryTest
    {
        private Inventory _inventory;
        private Item _item1;
        private Item _item2;

        [SetUp]
        public void SetUp()
        {
            _inventory = new Inventory();
            _item1 = new Item(new string[] { "sword" }, "Sword", "A sharp sword");
            _item2 = new Item(new string[] { "shovel" }, "Shovel", "A metal shovel
↪  for digging holes");

            _inventory.Put(_item1);
            _inventory.Put(_item2);
        }

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

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

        [Test]
        public void TestFetchItem()
        {
            Item fetchedItem1 = _inventory.Fetch("sword");
            Assert.AreEqual(_item1, fetchedItem1);
            Assert.IsTrue(_inventory.HasItem("sword"));

            Item fetchedItem2 = _inventory.Fetch("shovel");
            Assert.AreEqual(_item2, fetchedItem2);
            Assert.IsTrue(_inventory.HasItem("shovel"));
        }

```

```
53          [Test]
54          public void TestTakeItem()
55          {
56              Item takenItem1 = _inventory.Take("sword");
57              Assert.AreEqual(_item1, takenItem1);
58              Assert.IsFalse(_inventory.HasItem("sword"));
59
60
61              Item takenItem2 = _inventory.Take("shovel");
62              Assert.AreEqual(_item2, takenItem2);
63              Assert.IsFalse(_inventory.HasItem("shovel"));
64          }
65
66          [Test]
67          public void TestItemList()
68          {
69              string itemList = _inventory.ItemList;
70              string expectedItemList = "\n\ta Sword (sword)\n\ta Shovel (shovel)";
71
72              Assert.AreEqual(expectedItemList, itemList);
73          }
74      }
75  }
```


Iteration2
NUnitTests
SwinAdventureTests
BagTests
IdentifiableObjectTest
InventoryTest
TestFetchItem
TestFindItem
TestItemList
TestNoItemFind
TestTakeItem
ItemTest
TestFullDescription
TestItemIsIdentifiable
TestShortDescription
PlayerTest
TestPlayerFullDescription
TestPlayerIsIdentifiable
TestPlayerLocatesItems
TestPlayerLocatesItself
TestPlayerLocatesNothing
TestLookCommand