Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 2 - Players Items and Inventory

PDF generated at 19:16 on Friday 29th September, 2023

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

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

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

        public string Name
        {
            get { return _name; }
        }

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

        public virtual 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 GameObject? Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }
            else
            {
                return null;
            }
        }

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

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

```

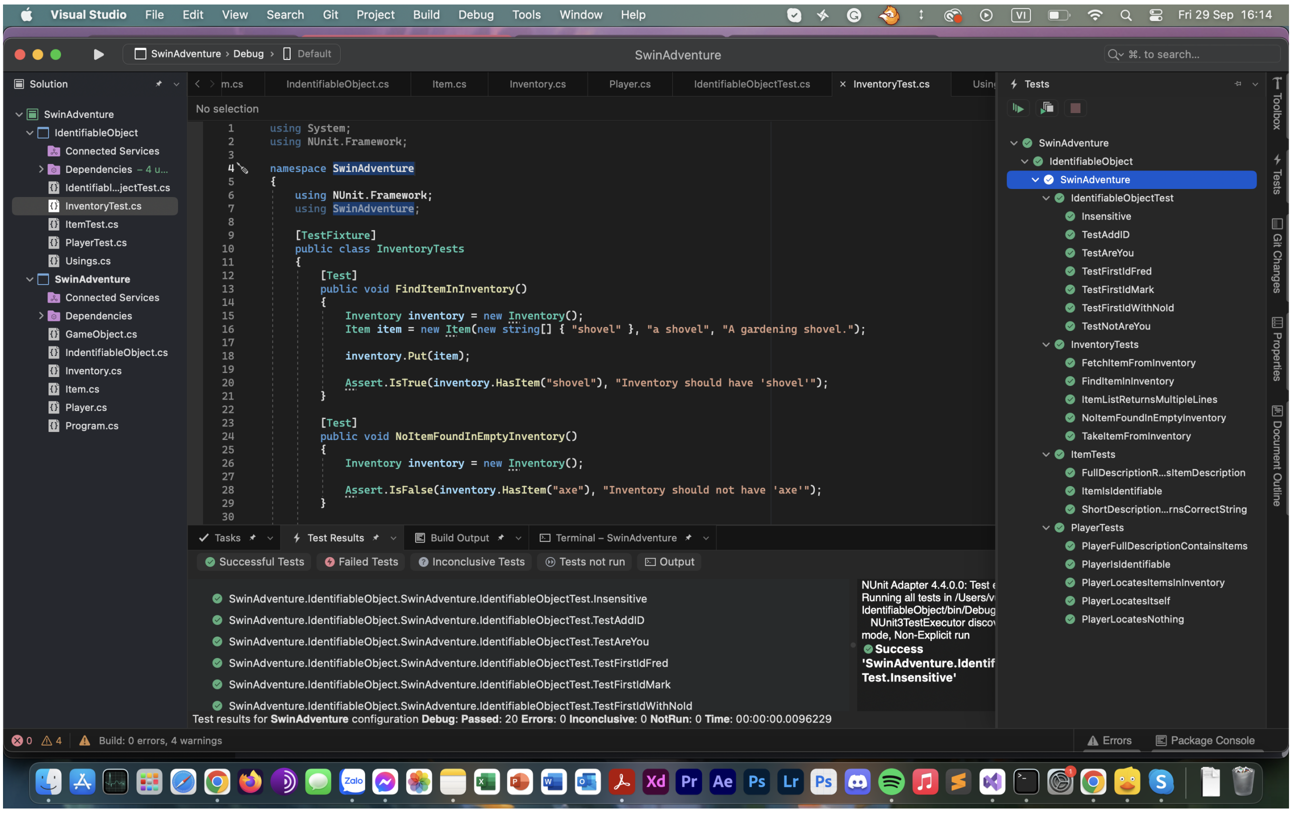
```
using System;
using NUnit.Framework;

namespace SwinAdventure
{
    using NUnit.Framework;
    using SwinAdventure;

    [TestFixture]
    public class PlayerTests
    {
        [Test]
        public void PlayerIsIdentifiable()
        {
            Player player = new Player("Fred", "the mighty programmer");

            Assert.IsTrue(player.AreYou("me"), "Player should respond to 'me'");
            Assert.IsTrue(player.AreYou("inventory"), "Player should respond to 'inventory'");
            Assert.IsFalse(player.AreYou("player"), "Player should not respond to 'player'");
        }

        [Test]
        public void PlayerLocatesItemsInInventory()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item item = new Item(new string[] { "sword" }, "bronze sword", "This is a mighty fine sword.");
            player.Inventory.Put(item);

            GameObject locatedItem = player.Locate("sword");

            Assert.IsNotNull(locatedItem, "Player should locate 'sword'");
            Assert.IsTrue(locatedItem.AreYou("sword"), "Located item should be 'sword'");
            Assert.IsTrue(player.Inventory.HasItem("sword"), "Item should remain in player's inventory");
        }

        [Test]
        public void PlayerLocatesItself()
        {
            Player player = new Player("Fred", "the mighty programmer");

            GameObject locatedPlayer1 = player.Locate("me");
            GameObject locatedPlayer2 = player.Locate("inventory");

            Assert.IsNotNull(locatedPlayer1, "Player should locate 'me'");
            Assert.IsTrue(locatedPlayer1.AreYou("me"), "Located object should be 'me'");
            Assert.IsNotNull(locatedPlayer2, "Player should locate 'inventory'");
            Assert.IsTrue(locatedPlayer2.AreYou("inventory"), "Located object should be 'inventory'");
        }

        [Test]
        public void PlayerLocatesNothing()
        {
            Player player = new Player("Fred", "the mighty programmer");

            GameObject locatedObject = player.Locate("axe");

            Assert.IsNull(locatedObject, "Player should not locate 'axe'");
        }

        [Test]
        public void PlayerFullDescriptionContainsItems()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item item1 = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");
            Item item2 = new Item(new string[] { "sword" }, "a sword", "A sharp sword.");
            player.Inventory.Put(item1);
            player.Inventory.Put(item2);

            string fullDescription = player.FullDescription;

            Assert.IsTrue(fullDescription.Contains("You are Fred, the mighty programmer."), "FullDescription should contain player's name and description");
            Assert.IsTrue(fullDescription.Contains("You are carrying:"), "FullDescription should contain 'You are carrying:'");
            Assert.IsTrue(fullDescription.Contains("a shovel (shovel)"), "FullDescription should contain 'a shovel'");
            Assert.IsTrue(fullDescription.Contains("a sword (sword)"), "FullDescription should contain 'a sword'");
        }
    }
}
```

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

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

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

namespace SwinAdventure
{
    public class ItemTests
    {
        [Test]
        public void ItemIsIdentifiable()
        {
            Item item = new Item(new string[] { "sword" }, "bronze sword", "This is a mighty fine sword.");

            Assert.IsTrue(item.AreYou("sword"), "Item should respond to 'sword'");
            Assert.IsFalse(item.AreYou("axe"), "Item should not respond to 'axe'");
        }

        [Test]
        public void ShortDescriptionReturnsCorrectString()
        {
            Item item = new Item(new string[] { "sword" }, "bronze sword", "This is a mighty fine sword.");

            Assert.That(item.ShortDescription, Is.EqualTo("a bronze sword (sword)"));
        }

        [Test]
        public void FullDescriptionReturnsItemDescription()
        {
            Item item = new Item(new string[] { "sword" }, "bronze sword", "This is a mighty fine sword.");

            Assert.That(item.FullDescription, Is.EqualTo("This is a mighty fine sword."));
        }
    }

}

```

```
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)
        {
            return _items.Any(item => item.AreYou(id));
        }

        public void Put(Item item)
        {
            _items.Add(item);
        }

        public Item Take(string id)
        {
            Item item = _items.FirstOrDefault(i => i.AreYou(id));
            if (item != null)
            {
                _items.Remove(item);
            }
            return item;
        }

        public Item Fetch(string id)
        {
            return _items.FirstOrDefault(i => i.AreYou(id));
        }

        public string ItemList
        {
            get
            {
                return string.Join("\n", _items.Select(item =>
↪   item.ShortDescription));
            }
        }
    }
}
```

```
using System;
using NUnit.Framework;

namespace SwinAdventure
{
    using NUnit.Framework;
    using SwinAdventure;

    [TestFixture]
    public class InventoryTests
    {
        [Test]
        public void FindItemInInventory()
        {
            Inventory inventory = new Inventory();
            Item item = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");

            inventory.Put(item);

            Assert.IsTrue(inventory.HasItem("shovel"), "Inventory should have 'shovel'");
        }

        [Test]
        public void NoItemFoundInEmptyInventory()
        {
            Inventory inventory = new Inventory();

            Assert.IsFalse(inventory.HasItem("axe"), "Inventory should not have 'axe'");
        }

        [Test]
        public void FetchItemFromInventory()
        {
            Inventory inventory = new Inventory();
            Item item = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");
            inventory.Put(item);

            Item fetchedItem = inventory.Fetch("shovel");

            Assert.IsNotNull(fetchedItem, "Inventory should fetch 'shovel'");
            Assert.IsTrue(fetchedItem.AreYou("shovel"), "Fetched item should be 'shovel'");
            Assert.IsTrue(inventory.HasItem("shovel"), "Item should remain in inventory");
        }

        [Test]
        public void TakeItemFromInventory()
        {
            Inventory inventory = new Inventory();
            Item item = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");
            inventory.Put(item);

            Item takenItem = inventory.Take("shovel");

            Assert.IsNotNull(takenItem, "Inventory should take 'shovel'");
            Assert.IsTrue(takenItem.AreYou("shovel"), "Taken item should be 'shovel'");
            Assert.IsFalse(inventory.HasItem("shovel"), "Item should be removed from inventory");
        }

        [Test]
        public void ItemListReturnsMultipleLines()
        {
            Inventory inventory = new Inventory();
            Item item1 = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");
            Item item2 = new Item(new string[] { "sword" }, "a sword", "A sharp sword.");
            inventory.Put(item1);
            inventory.Put(item2);

            string itemList = inventory.ItemList;

            Assert.IsTrue(itemList.Contains("a shovel (shovel)"), "Item list should contain 'a shovel'");
            Assert.IsTrue(itemList.Contains("a sword (sword)"), "Item list should contain 'a sword'");
        }
    }
}
```



```
using System;
using NUnit.Framework;

namespace SwinAdventure
{
    using NUnit.Framework;
    using SwinAdventure;

    [TestFixture]
    public class InventoryTests
    {
        [Test]
        public void FindItemInInventory()
        {
            Inventory inventory = new Inventory();
            Item item = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");

            inventory.Put(item);

            Assert.IsTrue(inventory.HasItem("shovel"), "Inventory should have 'shovel'");
        }

        [Test]
        public void NoItemFoundInEmptyInventory()
        {
            Inventory inventory = new Inventory();

            Assert.IsFalse(inventory.HasItem("axe"), "Inventory should not have 'axe'");
        }

```

```
✓ SwinAdventure.IdentifiableObject.SwinAdventure.IdentifiableObjectTest.Insensitive
✓ SwinAdventure.IdentifiableObject.SwinAdventure.IdentifiableObjectTest.TestAddID
✓ SwinAdventure.IdentifiableObject.SwinAdventure.IdentifiableObjectTest.TestAreYou
✓ SwinAdventure.IdentifiableObject.SwinAdventure.IdentifiableObjectTest.TestFirstIdFred
✓ SwinAdventure.IdentifiableObject.SwinAdventure.IdentifiableObjectTest.TestFirstIdMark
✓ SwinAdventure.IdentifiableObject.SwinAdventure.IdentifiableObjectTest.TestFirstIdWithNoId
Test results for SwinAdventure configuration Debug: Passed: 20 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0096229
```

```
NUnit Adapter 4.4.0.0: Test e
Running all tests in /Users/v
IdentifiableObject/bin/Debug
  NUnit3TestExecutor discov
mode, Non-Explicit run
✓ Success
'SwinAdventure.Identif
Test.Insensitive'
```

```
✓ SwinAdventure
  ✓ IdentifiableObject
    ✓ SwinAdventure
      ✓ IdentifiableObjectTest
        ✓ Insensitive
        ✓ TestAddID
        ✓ TestAreYou
        ✓ TestFirstIdFred
        ✓ TestFirstIdMark
        ✓ TestFirstIdWithNoId
        ✓ TestNotAreYou
      ✓ InventoryTests
        ✓ FetchItemFromInventory
        ✓ FindItemInInventory
        ✓ ItemListReturnsMultipleLines
        ✓ NoItemFoundInEmptyInventory
        ✓ TakeItemFromInventory
      ✓ ItemTests
        ✓ FullDescriptionR...sItemDescription
        ✓ ItemIsIdentifiable
        ✓ ShortDescription...rnsCorrectString
      ✓ PlayerTests
        ✓ PlayerFullDescriptionContainsItems
        ✓ PlayerIsIdentifiable
        ✓ PlayerLocatesItemsInInventory
        ✓ PlayerLocatesItself
        ✓ PlayerLocatesNothing
```

Build: 0 errors, 4 warnings