Swinburne University of Technology

COS20007 Object Oriented Programming

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

PDF generated at 01:35 on Thursday 30th March, 2023

Produced by OOP Submission Formatter

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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;
        }

        //properties
        public string Name
        {
            get
            {
                return _name;
            }
        }

        public string Description
        {
            get
            {
                return _description;
            }
        }

        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;
            }
            else
            {
                return _inventory.Fetch(id);
            }
        }

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

        public Inventory Inventory => _inventory;
    }
}

```

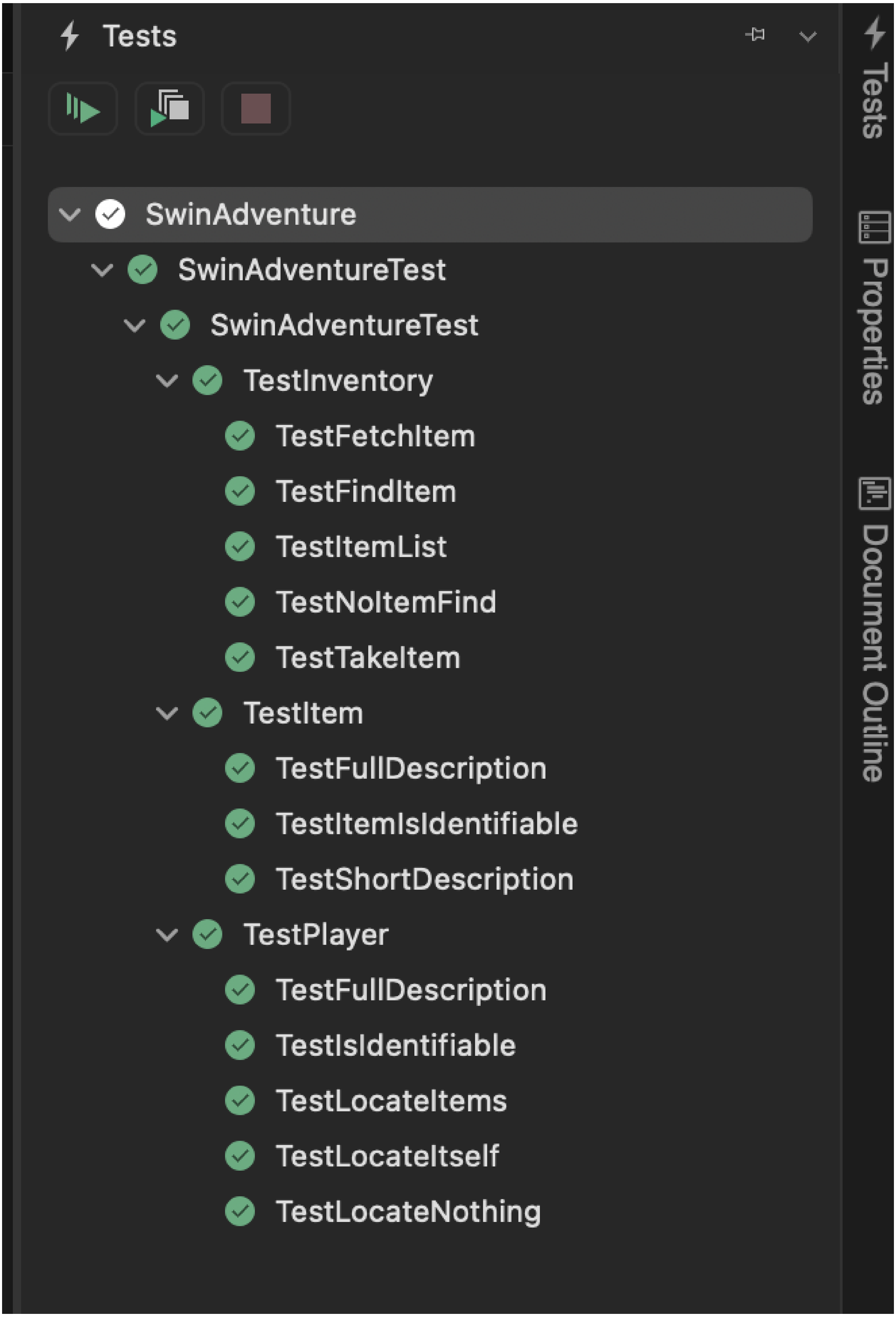
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventureTest
{
    [TestFixture]
    public class TestPlayer
    {
        Player player;
        Item sword;

        [SetUp]
        public void Setup()
        {
            player = new Player("bob", "the builder");
            sword = new Item(new string[] { "Sword" }, "a golden sword", "This is a golden sword");
            player.Inventory.Put(sword);
        }

        [Test]
        public void TestIsIdentifiable()
        {
            Assert.That(player.AreYou("me"), Is.True);
            Assert.That(player.AreYou("inventory"), Is.True);
        }

        [Test]
        public void TestLocateItems()
        {
            Assert.That(player.Locate("sword"), Is.SameAs(sword));
            Assert.That(player.Inventory.HasItem("sword"), Is.True);
        }

        [Test]
        public void TestLocateItself()
        {
            Assert.That(player.Locate("me"), Is.SameAs(player));
            Assert.That(player.Locate("inventory"), Is.SameAs(player));
        }

        [Test]
        public void TestLocateNothing()
        {
            Assert.That(player.Locate("cucumber"), Is.SameAs(null));
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.That(player.FullDescription,
                Is.EqualTo("You are bob, the builder.\nYou are carrying:\na golden
↪   sword (sword)\n"));
        }
    }
}
```

```
using System;

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

    }
}


```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventureTest
{
    [TestFixture]
    public class TestItem
    {
        Item sword;

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

        [Test]
        public void TestItemIsIdentifiable()
        {
            Assert.That(sword.AreYou("sword"), Is.True);
            Assert.That(sword.AreYou("knife"), Is.False);
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.That(sword.ShortDescription, Is.EqualTo("a golden sword (sword)"));
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.That(sword.FullDescription, Is.EqualTo("This is a golden sword"));
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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

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

        //methods
        public bool HasItem(string id)
        {
            return _items.Any(item => item.AreYou(id));
        }

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

        //removing and returning
        public Item Take(string id)
        {
            Item itm = this.Fetch(id);

            if (itm != null)
            {
                _items.Remove(itm);
            }
            return itm;
        }

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

        public string ItemList
        {
            get
            {
                string itemList = "";

                foreach (Item itm in _items)
                {
                    itemList += $"{itm.ShortDescription}\n";
                }
                return itemList;
            }
        }
    }
}

```

```
using System;

namespace SwinAdventureTest

{
    [TestFixture]
    public class TestInventory
    {
        Inventory inventory;
        Item sword;
        Item knife;

        [SetUp]
        public void SetUp()
        {
            inventory = new Inventory();
            sword = new Item(new string[] { "Sword" }, "a gold sword", "This is a
 ↪  gold sword");
            knife = new Item(new string[] { "Knife" }, "a sharp knife", "This is a
 ↪  sharp knife");
            inventory.Put(sword);
            inventory.Put(knife);
        }

        [Test]
        public void TestFindItem()
        {
            Assert.That(inventory.HasItem("sword"), Is.True);
            Assert.That(inventory.HasItem("shovel"), Is.False);
        }

        [Test]
        public void TestNoItemFind()
        {
            Assert.That(inventory.HasItem("wrench"), Is.False);
        }

        [Test]
        public void TestFetchItem()
        {
            Assert.That(inventory.Fetch("sword"), Is.SameAs(sword));
            Assert.That(inventory.HasItem("sword"), Is.True);
        }

        [Test]
        public void TestTakeItem()
        {
            Assert.That(inventory.Take("sword"), Is.SameAs(sword));
            Assert.That(inventory.HasItem("sword"), Is.False);
        }

        [Test]
        public void TestItemList()
        {
            Assert.That(inventory.ItemList, Is.EqualTo("a gold sword (sword)\na sharp knife (knife)\n"));
        }
    }
}
```



```
✓ SwinAdventure
  ✓ SwinAdventureTest
    ✓ SwinAdventureTest
      ✓ TestInventory
        ✓ TestFetchItem
        ✓ TestFindItem
        ✓ TestItemList
        ✓ TestNoItemFind
        ✓ TestTakeItem
      ✓ TestItem
        ✓ TestFullDescription
        ✓ TestItemIsIdentifiable
        ✓ TestShortDescription
      ✓ TestPlayer
        ✓ TestFullDescription
        ✓ TestIsIdentifiable
        ✓ TestLocateItems
        ✓ TestLocateItself
        ✓ TestLocateNothing
```