Swinburne University of Technology

COS20007 Object Oriented Programming

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

PDF generated at 13:00 on Tuesday 11th April, 2023

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

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

        //constructor
        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})";
            }
        }
        public virtual string FullDescription
        {
            get
            {
                return _description;
            }
        }

    }
}
```

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

namespace SwinAdventure
{
    public class Player : GameObject, IHaveInventory
    {
        //local variables
        // inventory of items of the player
        private Inventory _inventory;
        private Location _location;

        //comstructor
        public Player(string name, string desc) : base(new string[] {"me",
↪   "inventory"}, name, desc)
        {
            _inventory = new Inventory();
        }

        //methods

        //locate method that returns a gameobject based on the id.
        // for now it only returns the player itself if any of the above identifiers
↪   are entereed
        // or returns items that exists in its inventory
        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }
            else if (_location != null)
            {
                return _location.Locate(id);
            }
            else return null;
        }

        public void Move(Path path)
        {
            _location = path.Destination;
        }
        //properties

        // override FullDescription property to include the player's name, and the
↪   shortdescription of themselves and their items in their inventory
```

```
51          public override string FullDescription
52          {
53              get
54              {
55                  return $"You are {Name}, {Description}.\nYou are
    ↪  carrying:\n{_inventory.ItemList}";
56              }
57          }
58
59          public Inventory Inventory
60          {
61              get
62              {
63                  return _inventory;
64              }
65          }
66          public Location Location
67          {
68              get
69              {
70                  return _location;
71              }
72              set
73              {
74                  _location = value;
75              }
76          }
77      }
78  }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Path = SwinAdventure.Path;

namespace TestSwinAdventure
{
    [TestFixture]
    public class TestPlayer
    {
        Player player;
        Item sword;
        Location location;
        Location destination;
        Path path;
        Item gem;

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

            gem = new Item(new string[] { "gem" }, "a gem", "a bright red crystal");
            location = new Location("garden", "This is a garden");
            destination = new Location("a house", "This is a house");
            path = new Path(new string[] { "south" }, "south", "this is south",
↪   destination);
            location.Inventory.Put(gem);
            location.AddPath(path);

            player.Location = location;
        }

        [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]
```

```
52          public void TestLocateItself()
53          {
54              Assert.That(player.Locate("me"), Is.SameAs(player));
55              Assert.That(player.Locate("inventory"), Is.SameAs(player));
56          }
57
58          [Test]
59          public void TestLocateNothing()
60          {
61              Assert.That(player.Locate("scythe"), Is.SameAs(null));
62          }
63
64          [Test]
65          public void TestLocateLocation()
66          {
67              Assert.That(player.Locate("room"), Is.SameAs(location));
68          }
69
70          [Test]
71          public void TestLocateItemInLocation()
72          {
73              Assert.That(player.Locate("gem"), Is.SameAs(gem));
74          }
75          [Test]
76          public void TestFullDescription()
77          {
78              Assert.That(player.FullDescription,
79                  Is.EqualTo("You are shah, the student.\nYou are carrying:\na bronze
    ↪  sword (sword)\n"));
80          }
81
82          [Test]
83          public void TestMove()
84          {
85              player.Move(path);
86              Assert.That(player.Location, Is.SameAs(destination));
87          }
88      }
89  }
```

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

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

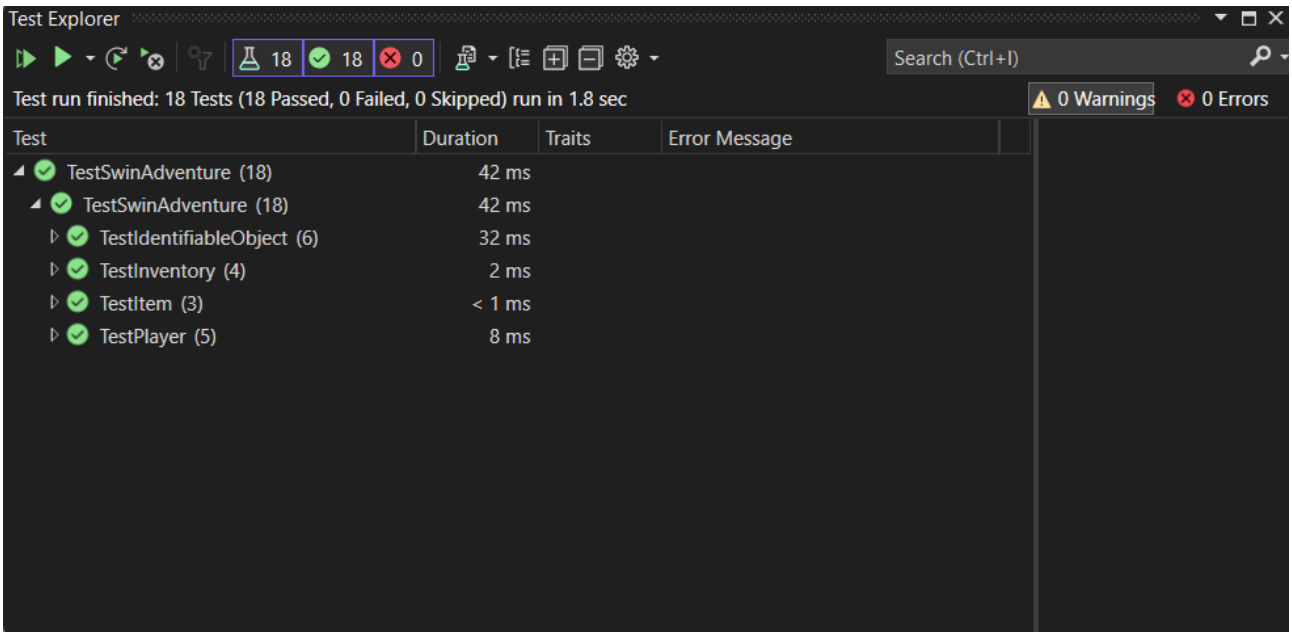
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TestSwinAdventure
{
    [TestFixture]
    public class TestItem
    {
        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.That(sword.AreYou("sword"), Is.True);
            Assert.That(sword.AreYou("knife"), Is.False);
        }

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

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

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

namespace SwinAdventure
{
    public class Inventory
    {
        //local variables
        // collection of items
        private List<Item> _items;

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

        //methods

        // check if the collection has the specific item using AreYou method inherited from IdentifiableObject
        public bool HasItem(string id)
        {
            return _items.Any(item => item.AreYou(id));
        }

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

        // Take item by removing it from the collection and returning it
        public Item Take(string id)
        {
            Item itm = this.Fetch(id);

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

            return itm;
        }

        // find the specific item and return it (collection is not modified)
        public Item Fetch(string id)
        {
            foreach (Item itm in _items)
            {
```

```
53                  if (itm.AreYou(id))
54                  {
55                      return itm;
56                  }
57              }
58              return null;
59          }
60
61          // properties
62
63          //itemlist returns a string of all the short descriptions of the items in
    ↪  the collection
64          public string ItemList
65          {
66              get
67              {
68                  string itemList = "";
69
70                  foreach (Item itm in _items)
71                  {
72                      itemList += $"{itm.ShortDescription}\n";
73                  }
74                  return itemList;
75              }
76          }
77      }
78  }
```

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

namespace TestSwinAdventure
{
    [TestFixture]
    public class TestInventory
    {
        Inventory inventory;
        Item sword;
        Item bat;

        [SetUp]
        public void Setup()
        {
            inventory = new Inventory();
            sword = new Item(new string[] { "Sword" }, "a bronze sword", "This is a bronze sword");
            bat = new Item(new string[] { "Bat" }, "a hard bat", "This is a hard bat");
            inventory.Put(sword);
            inventory.Put(bat);
        }

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

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

        [Test]
        public void TestTake()
        {
            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 bronze sword (sword)\na hard bat (bat)\n"));
```

```
51            }
52        }
53    }
```


Test Explorer
18
18
0
Search (Ctrl+I)
Test run finished: 18 Tests (18 Passed, 0 Failed, 0 Skipped) run in 1.8 sec
0 Warnings
0 Errors
Test
Duration
Traits
Error Message
TestSwinAdventure (18)
42 ms
TestSwinAdventure (18)
42 ms
TestIdentifiableObject (6)
32 ms
TestInventory (4)
2 ms
TestItem (3)
< 1 ms
TestPlayer (5)
8 ms