Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 2 - Players Items and Inventory

PDF generated at 01:51 on Friday 10th November, 2023

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

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

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

        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }

        public string ShortDescription
        {
            get { return $"{Name} ({FirstId})"; }
            set { _description = value; }
        }

        public virtual string FullDescription
        {
            get { return _description; }
            set { _description = value; }
        }
    }
}
```

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

namespace MazeGame
{
    public class Player : GameObject
    {
        Inventory _inventory;
        string _name;
        string _description;

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

        public GameObject Locate(string id)
        {
            List<GameObject> gameOBJ = new List<GameObject>();

            if (id == "me" || id == "inventory")
            {
                gameOBJ.Add(this);
            }
            else if (_inventory.HasItem(id))
            {
                var item = _inventory.Fetch(id);
                gameOBJ.Add(item);
            }
            else
            {
                Item nullObj = null;
                gameOBJ.Add(nullObj);
            }

            var result = gameOBJ.ElementAt(0);
            gameOBJ.Clear();
            return result;
        }

        public override string FullDescription
        {
            get
            {
                return
                    $"You are {_name} {_description}\n" +
```

```
53                      $"You are carrying: \n{_inventory.ItemList}";
54              }
55
56              set
57              {
58                  _name = value;
59                  _description = value;
60              }
61          }
62
63          public Inventory Inventory
64          {
65              get { return _inventory; }
66              set { _inventory = value; }
67          }
68      }
69  }
```

```
namespace MazeGame.nUnitTests
{
    public class PlayerTests
    {
        private Player _player { get; set; } = null!;
        private Item sword { get; set; } = null!;
        private Item shovel { get; set; } = null!;
        private Item pickaxe { get; set; } = null!;

        [SetUp]
        public void SetUp()
        {
            _player = new Player("Hoang An", "the comtemplator of infinity");
            sword = new Item(new string[] { "sword" }, "a bronze sword", "A short
↪   sword cast from bronze");
            shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable
↪   shovel borrowed from the village");
            pickaxe = new Item(new string[] { "pickaxe" }, "an obsidian pickaxe", "A
↪   pickaxe made of obsidian");
            _player.Inventory.Put(sword);
            _player.Inventory.Put(shovel);
            _player.Inventory.Put(pickaxe);
        }

        [Test]
        public void Test_Identifiable()
        {
            string id_ME = "me";
            var sut1 = _player.AreYou(id_ME);
            string id_INV = "inventory";
            var sut2 = _player.AreYou(id_INV);
            Assert.That(sut1, Is.EqualTo(true));
            Assert.That(sut2, Is.EqualTo(true));
        }

        [Test]
        public void Test_LocateItems()
        {
            string sampleID = "sword";
            var sut = _player.Locate(sampleID);
            Assert.That(sut.FirstId, Is.EqualTo(sampleID));
            Console.WriteLine(sut.FullDescription);
        }

        [Test]
        public void Test_LocateItself()
        {
            string id_ME = "me";
            var sut1 = _player.Locate(id_ME);
            string id_INV = "inventory";
            var sut2 = _player.Locate(id_INV);
            Assert.IsNotNull(sut1);
            Assert.IsNotNull(sut2);
```

```
51              Console.WriteLine($"Player (me): {sut1.ShortDescription}");
52              Console.WriteLine();
53              Console.WriteLine("Inventory (inventory): \n" + sut2.FullDescription);
54          }
55
56          [Test]
57          public void Test_LocateNothing()
58          {
59              string sampleID = "shoe";
60              var sut = _player.Locate(sampleID);
61              Assert.IsNull(sut);
62          }
63
64          [Test]
65          public void Test_FullDescription()
66          {
67              string sample = "You are Hoang An the comtemplator of infinity\n" +
68                  "You are carrying: \n" +
69                  "a bronze sword (sword)\n" +
70                  "a shovel (shovel)\n" +
71                  "an obsidian pickaxe (pickaxe)";
72              var sut = _player.FullDescription;
73              Assert.IsNotNull(sut);
74              Assert.That(sut, Is.EqualTo(sample));
75              Console.WriteLine($"Sample return: \n" +
76                  $"{sample}");
77              Console.WriteLine();
78              Console.WriteLine($"Test return: \n" +
79                  $"{sut}");
80          }
81      }
82  }
```

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

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

        }
    }
}
```

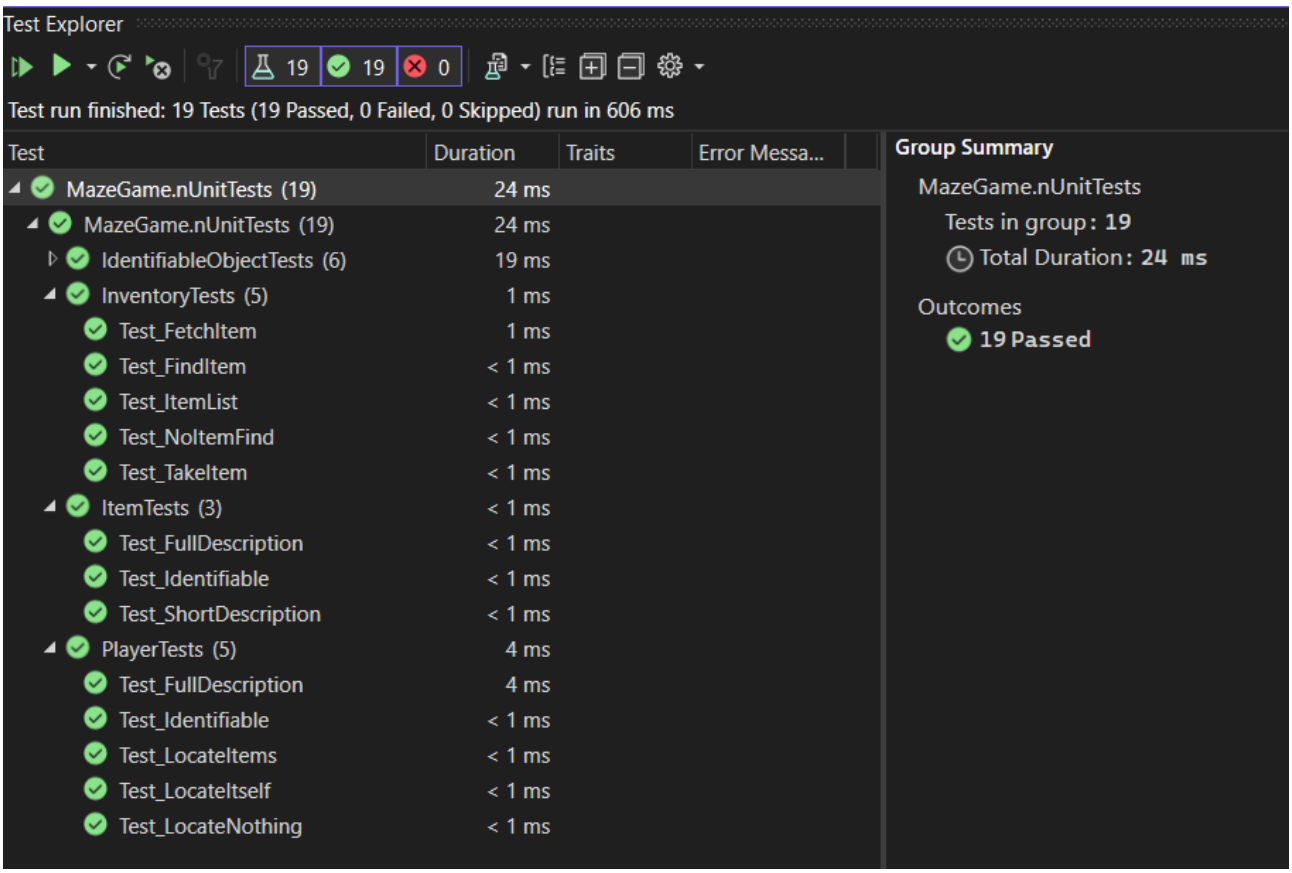
```
namespace MazeGame.nUnitTests
{
    public class ItemTests
    {
        private Item _item { get; set; } = null!;

        [SetUp]
        public void Setup()
        {
            _item = new Item(new string[] { "sword" }, "a bronze sword", "A short sword cast from bronze");
        }

        [Test]
        public void Test_Identifiable()
        {
            var sut = _item.AreYou(_item.FirstId);
            Assert.That(sut, Is.EqualTo(true));
        }

        [Test]
        public void Test_ShortDescription()
        {
            string sample = "a bronze sword (sword)";
            var sut = _item.ShortDescription;
            Assert.That(sut, Is.EqualTo(sample));
            Console.WriteLine(sut);
        }

        [Test]
        public void Test_FullDescription()
        {
            string sample = "A short sword cast from bronze";
            var sut = _item.FullDescription;
            Assert.That(sut, Is.EqualTo(sample));
            Console.WriteLine(sut);
        }

    }
}
```

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

namespace MazeGame
{
    public class Inventory
    {
        List<Item> _items;

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

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

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

        public Item Take(string id)
        {
            List<Item> items = new List<Item>();
            foreach(Item itm in _items)
            {
                if(id == itm.FirstId)
                {
                    items.Add(itm);
                }
            }
            var result = items.ElementAt(0);
            items.Clear();
            _items.Remove(result);
            return result;
        }

        public Item Fetch(string id)
        {
            List<Item> items = new List<Item>();
```

```
54              foreach(Item itm in _items)
55              {
56                  if (itm.AreYou(id))
57                  {
58                      items.Add(itm);
59                  }
60              }
61              var result = items.ElementAt(0);
62              items.Clear();
63              return result;
64          }
65
66          public string ItemList
67          {
68              get
69              {
70                  List<string> list = new List<string>();
71
72                  foreach (Item itm in _items)
73                  {
74                      list.Add($"{itm.Name} ({itm.FirstId})");
75                  }
76
77                  string itemList = string.Join("\n", list);
78                  return itemList;
79              }
80
81              set
82              {
83                  foreach (Item itm in _items)
84                  {
85                      itm.Name = value;
86                      itm.ShortDescription = value;
87                  }
88              }
89          }
90      }
91  }
```

```
namespace MazeGame.nUnitTests
{
    public class InventoryTests
    {
        private Inventory _inventory { get; set; } = null!;
        private Item sword { get; set; } = null!;
        private Item shovel { get; set; } = null!;
        private Item pickaxe { get; set; } = null!;

        [SetUp]
        public void SetUp()
        {
            _inventory = new Inventory();
            sword = new Item(new string[] { "sword" }, "a bronze sword", "A short sword cast from bronze");
            shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable shovel borrowed from the village");
            pickaxe = new Item(new string[] { "pickaxe" }, "an obsidian pickaxe", "A pickaxe made of obsidian");
            _inventory.Put(sword);
            _inventory.Put(shovel);
            _inventory.Put(pickaxe);
        }

        [Test]
        public void Test_FindItem()
        {
            string sampleID = "shovel";
            var sut = _inventory.HasItem(sampleID);
            Assert.That(sut, Is.EqualTo(true));
            Console.WriteLine("Item is in the inventory: " + sut);
        }

        [Test]
        public void Test_NoItemFind()
        {
            string sampleID = "hat";
            var sut = _inventory.HasItem(sampleID);
            Assert.That(sut, Is.EqualTo(false));
            Console.WriteLine("Item is in the inventory: " + sut);
        }

        [Test]
        public void Test_FetchItem()
        {
            string sampleID = "shovel";
            var sut = _inventory.Fetch(sampleID);
            Assert.That(sut, Is.EqualTo(shovel));
            Console.WriteLine("Item fetched from inventory: " + sut.ShortDescription);
        }

        [Test]
```

```
50          public void Test_TakeItem()
51          {
52              string sampleID = "pickaxe";
53              var sut = _inventory.Take(sampleID);
54              Assert.That(sut, Is.EqualTo(pickaxe));
55              Console.WriteLine(sut.ShortDescription);
56              var itemExist = _inventory.HasItem(sampleID);
57              Assert.That(itemExist, Is.EqualTo(false));
58              Console.WriteLine("Item still in inventory after Test_TakeItem: " +
    ↪  itemExist);
59          }
60
61          [Test]
62          public void Test_ItemList()
63          {
64              string sampleReturn =
65                  $"{sword.Name} ({sword.FirstId})\n" +
66                  $"{shovel.Name} ({shovel.FirstId})\n" +
67                  $"{pickaxe.Name} ({pickaxe.FirstId})";
68              var sut = _inventory.ItemList;
69              Console.WriteLine("Test return: \n" + sut);
70              Console.WriteLine();
71              Console.WriteLine("Sample return: \n" + sampleReturn);
72              Assert.That(sut, Is.EqualTo(sampleReturn));
73          }
74      }
75  }
```


Test Explorer
19
19
0
Test run finished: 19 Tests (19 Passed, 0 Failed, 0 Skipped) run in 606 ms
Test
Duration
Traits
Error Messa...
MazeGame.nUnitTests (19) 24 ms
MazeGame.nUnitTests (19) 24 ms
IdentifiableObjectTests (6) 19 ms
InventoryTests (5) 1 ms
Test_FetchItem 1 ms
Test_FindItem < 1 ms
Test_ItemList < 1 ms
Test_NoItemFind < 1 ms
Test_TakeItem < 1 ms
ItemTests (3) < 1 ms
Test_FullDescription < 1 ms
Test_Identifiable < 1 ms
Test_ShortDescription < 1 ms
PlayerTests (5) 4 ms
Test_FullDescription 4 ms
Test_Identifiable < 1 ms
Test_LocateItems < 1 ms
Test_LocateItself < 1 ms
Test_LocateNothing < 1 ms
Group Summary
MazeGame.nUnitTests
Tests in group: 19
Total Duration: 24 ms
Outcomes
19 Passed