

...iented Programming\Projects\4.2P\4.2P\GameObject.cs

1

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace _4._2P
8 {
9     public class GameObject : IdentifiableObject
10    {
11        private string _name;
12        private string _description;
13
14        public GameObject(string[] ids, string name, string
            description) : base(ids) //call constructor of the base
            class
15        {
16            _name = name;
17            _description = description;
18        }
19
20        public string Name
21        {
22            get { return _name; }
23        }
24
25        public string ShortDescription
26        { get { return $"{_name} ({FirstId})"; } }
27
28        public virtual string FullDescription
29        { get { return _description; } }
30    }
31 }
32 }
33

```

Test	Duration	Traits	Error Message
IdentifiableObjectTest (7)	6 ms		
NUnitTests (7)	6 ms		
TestIdentifiableObject (7)	6 ms		
TestAddId()	4 ms		
TestAreYou()	< 1 ms		
TestCaseSensitivity()	< 1 ms		
TestFirstId()	2 ms		
TestFirstIdWithNoid()	< 1 ms		
TestNotAreYou()	< 1 ms		
TestPrivilegeEscalation()	< 1 ms		
InventoryTest (4)	5 ms		
InventoryTest (4)	5 ms		
Tests (4)	5 ms		
TestFetchItem	5 ms		
TestFindItem	< 1 ms		
TestNotItemFind	< 1 ms		
TestTakeItem	< 1 ms		
ItemTest (4)	5 ms		
ItemTest (4)	5 ms		
Tests (4)	5 ms		
TestFullDescription	4 ms		
TestItemIsIdentifiable	< 1 ms		
TestPrivilegeEscalation	< 1 ms		
TestShortDescription	1 ms		
PlayerTest (5)	28 ms		
PlayerTest (5)	28 ms		
Tests (5)	28 ms		
TestLocatesNothing	25 ms		
TestPlayerFullDescription	3 ms		
TestPlayerIsIdentifiable	< 1 ms		
TestPlayerLocatesItem	< 1 ms		
TestPlayerLocatesItself	< 1 ms		

Run | Debug

Test Detail Summary

TestAddId()

Source: UnitTest1.cs line 55

Duration: 4 ms

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace _4._2P
8 {
9     public class IdentifiableObject
10    {
11        private List<string> _identifiers = new List<string>();
12        public IdentifiableObject(string[] identifiers)
13        {
14
15
16            foreach (string id in identifiers)
17            {
18                AddIdentifier(id);
19            }
20        }
21
22        public void AddIdentifier(string identifier)
23        {
24            _identifiers.Add(identifier.ToLower());
25        }
26
27        public bool AreYou(string identifier)
28        { return _identifiers.Contains(identifier.ToLower()); }
29
30        public string FirstId
31        {
32            get
33            {
34                if (_identifiers.Count > 0)
35                {
36                    return _identifiers[0];
37                }
38                else
39                {
40                    return "";
41                }
42            }
43        }
44
45
46
47        public void PrivilegeEscalation(string pin)
48        {
49            if (pin == "2183" && _identifiers.Count > 0)
50            {
51                _identifiers[0] = "7";
52            }
53        }
54    }
55 }
```

```
54     }  
55 }  
56
```

```
1 using _4._2P;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using NUnit.Framework;
7 using System.Numerics;
8
9
10 namespace NUnitTests
11 {
12     [TestFixture]
13     public class TestIdentifiableObject
14     {
15         private IdentifiableObject _objectIdentifier;
16         private IdentifiableObject _emptyObjectIdentifier;
17         private IdentifiableObject _objectIdentifierNoStuId;
18
19         [SetUp]
20         public void Setup()
21         {
22             _objectIdentifier = new IdentifiableObject(new string[] { "104772183", "Khoi Nguyen", "Pham" });
23             _emptyObjectIdentifier = new IdentifiableObject(new string[] { });
24             _objectIdentifierNoStuId = new IdentifiableObject(new string[] { "1111", "Khoi Nguyen", "Pham" });
25         }
26
27         [TestCase]
28         public void TestAreYou()
29         {
30             Assert.IsTrue(_objectIdentifier.AreYou("104772183"));
31             Assert.IsTrue(_objectIdentifier.AreYou("Khoi Nguyen"));
32         }
33
34         [TestCase]
35         public void TestNotAreYou()
36         {
37             Assert.IsFalse(_objectIdentifier.AreYou("nonexistent identifier"));
38         }
39
40
41
42         [TestCase]
43         public void TestFirstId()
44         {
45             Assert.AreEqual("104772183", _objectIdentifier.FirstId);
46         }
47
48         [TestCase]
49         public void TestFirstIdWithNoId()
```

```
50     {
51         Assert.AreEqual("", _emptyObjectIdentifier.FirstId);
52     }
53
54     [TestCase]
55     public void TestAddId()
56     {
57         _objectIdentifier.AddIdentifier("Nevan");
58         Assert.IsTrue(_objectIdentifier.AreYou("Nevan"));
59     }
60
61     [TestCase]
62     public void TestCaseSensitivity()
63     {
64         Assert.IsTrue(_objectIdentifier.AreYou("khOI NGUYEN"));
65         Assert.IsTrue(_objectIdentifier.AreYou("PhAM"));
66     }
67
68     [TestCase]
69     public void TestPrivilegeEscalation()
70     {
71
72
73         string correctPin = "2183";
74         string expectedFirstId = "7";
75
76
77         _objectIdentifierNoStuId.PrivilegeEscalation(correctPin);
78
79         // Assert
80         Assert.AreEqual(expectedFirstId,
81             _objectIdentifierNoStuId.FirstId);
82     }
83
84
85
86
87 }
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace _4._2P
8 {
9     public class Inventory
10    {
11        private List<Item> _items = new List<Item>();
12        public Inventory() { }
13        public bool HasItem (string id)
14        {
15            return Fetch(id) != null;
16        }
17        public void Put(Item itm)
18        {
19            _items.Add(itm);
20        }
21        public Item Take(string id)
22        {
23            Item item = Fetch(id);
24            if (item != null)
25            {
26                _items.Remove(item);
27            }
28            return item;
29        }
30
31        public Item Fetch(string id)
32        {
33            foreach (Item item in _items)
34            {
35                if (item.AreYou(id))
36                {
37                    return item;
38                }
39            }
40            return null;
41        }
42
43        public string ItemList
44        {
45            get
46            {
47                string itemList = "";
48                foreach (Item item in _items)
49                {
50                    itemList += "\t" + item.ShortDescription + "\n";
51                }
52                return itemList;
53            }
54        }
55    }
56 }
```

54 }

55

56 }


57 }

58

```
1 using _4._2P;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using NUnit.Framework;
7 using System.Numerics;
8
9 namespace InventoryTest
10 {
11     [TestFixture]
12     public class Tests
13     {
14         private Inventory _inventory;
15         private Item _sword;
16         private Item _shield;
17
18         [SetUp]
19         public void Setup()
20         {
21             _inventory = new Inventory();
22             _sword = new Item(new string[] { "sword", "blade" },      ↗
23                             "Excalibur", "A strong sword");
24             _shield = new Item(new string[] { "shield", "safeguard" }, ↗
25                             "Aegis", "A strong shield");
26         }
27
28         [Test]
29         public void TestFindItem()
30         {
31             _inventory.Put(_sword);
32             Assert.IsTrue(_inventory.HasItem("sword"));
33         }
34
35         [Test]
36         public void TestNoItemFind()
37         {
38             Assert.IsFalse(_inventory.HasItem("spear"));
39         }
40
41         [Test]
42         public void TestFetchItem()
43         {
44             _inventory.Put(_sword);
45             Item fetchedItem = _inventory.Fetch("sword");
46             Assert.AreEqual(_sword, fetchedItem);
47             Assert.IsTrue(_inventory.HasItem("sword"));
48         }
49
50         [Test]
51         public void TestTakeItem()
52         {
53             _inventory.Put(_shield);
```



```
52     Item takenItem = _inventory.Take("shield");
53     Assert.AreEqual(takenItem, _shield);
54     Assert.IsFalse(_inventory.HasItem("shield"), "Inventory
    should not contain the shield because it has been taken");
55
56 }
57
58 public void TestItemList()
59 {
60     _inventory.Put(_sword);
61     _inventory.Put(_shield);
62     string expectedList = "\tExcalibur\n\tAegis";
63     Assert.AreEqual(expectedList, _inventory.ItemList);
64 }
65 }
66 }
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace _4._2P
8 {
9     public class Item : GameObject
10    {
11        public Item(string[] ids, string name, string description) : 
12            base(ids, name, description)
13        {
14        }
15    }
16 }
17
```

```
1 using _4._2P;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using NUnit.Framework;
7 using System.Numerics;
8
9 namespace ItemTest
10 {
11     [TestFixture]
12     public class Tests
13     {
14         private Item _sword;
15
16         [SetUp]
17         public void Setup()
18         {
19             _sword = new Item(new string[] { "sword", "blade" },
20                 "Excalibur", "A strong sword");
21         }
22
23         [Test]
24         public void TestItemIsIdentifiable()
25         {
26             Assert.IsTrue(_sword.AreYou("sword"), "Item should be
27                 identifiable as 'sword'");
28             Assert.IsTrue(_sword.AreYou("blade"), "Item should be
29                 identifiable as 'blade'");
30         }
31
32         [Test]
33         public void TestShortDescription()
34         {
35             Assert.AreEqual("Excalibur (sword)",
36                 _sword.ShortDescription);
37         }
38
39         [Test]
40         public void TestFullDescription()
41         {
42             Assert.AreEqual("A strong sword", _sword.FullDescription);
43         }
44
45         [Test]
46         public void TestPrivilegeEscalation()
47         {
48             string correctPin = "2183";
49             string expectedFirstId = "7";
50             _sword.PrivilegeEscalation(correctPin);
51             Assert.AreEqual(expectedFirstId, _sword.FirstId);
52         }
53     }
54 }
```

50 }

51 }

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace _4._2P
8 {
9     public class Player : GameObject
10    {
11        private Inventory _inventory = new Inventory();
12
13        public Player(string name, string description) : base(new
14            string[] { "me", "inventory" }, name, description) { } //name
15            and des gotten from GameObject
16        //help the class identify itself and its item, 3 batteries, 2
17        from GO and 1 from IO
18        public GameObject Locate(string id)
19        {
20            if (AreYou(id))
21            {
22                return this; //return then player object itself
23            }
24            return _inventory.Fetch(id);
25            //searches the inventory for an item with the given
26            identifier and returns it if found. If no item matches, it
27            returns null.
28        }
29
30        public override string FullDescription
31        {
32            get
33            {
34                return $"You are {Name}, {base.FullDescription}\nYou are
35                carrying:\n{_inventory.ItemList}";
36            }
37        }
38
39        public Inventory Inventory { get { return _inventory; } }
40    }
41 }
```

```
1 using _4._2P;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using NUnit.Framework;
7 using System.Numerics;
8
9
10 namespace PlayerTest
11 {
12     [TestFixture] //mark a class that contains tests. It tells NUnit      ↗
13         that this class should be treated as a test suite.
14     public class Tests
15     {
16         private Player _player;
17
18         [SetUp] //mark a method that should be run before each test      ↗
19             method. It's useful for setting up common test data or state.
20         public void Setup()
21         {
22             _player = new Player("Nevan", "a human");
23         }
24
25         [Test] // mark a method inside a [TestFixture] class as a test    ↗
26             method.
27         public void TestPlayerIsIdentifiable()
28         {
29             Assert.IsTrue(_player.AreYou("me"));
30             Assert.IsTrue(_player.AreYou("inventory"));
31         }
32
33         [Test]
34         public void TestPlayerLocatesItem()
35         {
36             Item sword = new Item(new string[] { "sword", "blade" },      ↗
37                 "Excalibur", "A strong sword");
38             _player.Inventory.Put(sword);
39             Assert.AreEqual(sword, _player.Locate("sword"));
40         }
41
42         [Test]
43         public void TestPlayerLocatesItself()
44         {
45             Assert.AreEqual(_player, _player.Locate("me"));
46         }
47
48         [Test]
49         public void TestLocatesNothing()
50         {
51             Assert.IsNull(_player.Locate("sth not exist"));
52         }
53     }
54 }
```

```
50
51     [Test]
52     public void TestPlayerFullDescription()
53     {
54         Item sword = new Item(new string[] { "sword", "blade" },      ↗
55                                "Excalibur", "A strong sword");
56         _player.Inventory.Put(sword);
57         Assert.AreEqual("You are Nevan, a human\nYou are carrying:\n ↗
58                        \tExcalibur (sword)\n", _player.FullDescription);
59     }
60 }
61 }
```