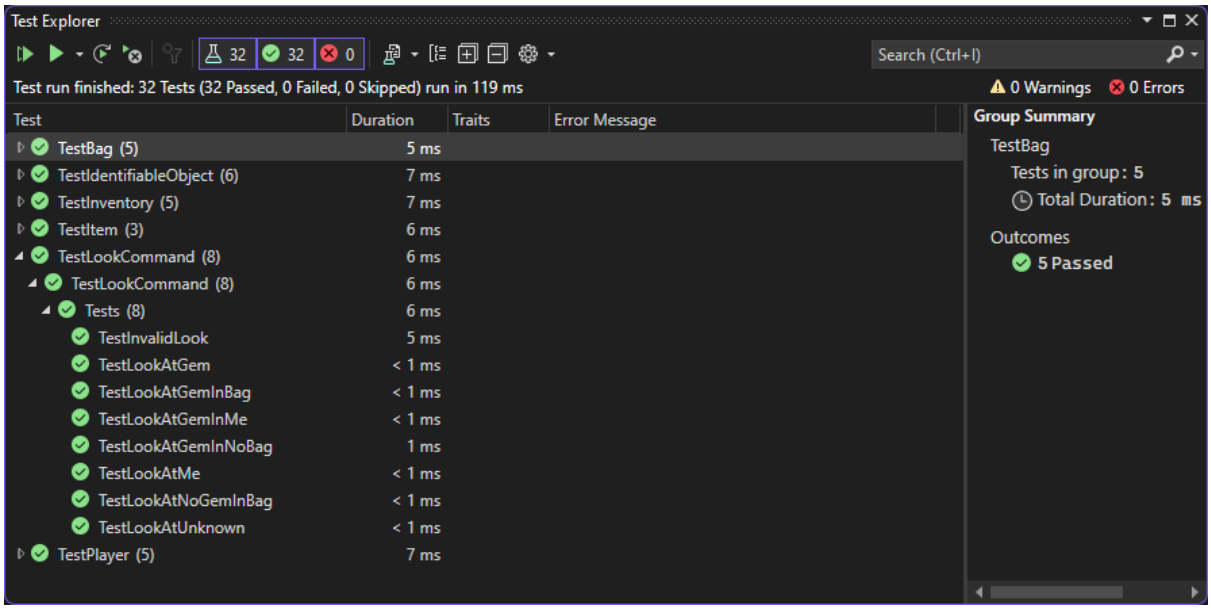
6.1P - Case Study - Iteration 4 - Look Command

Jayden Kong, 104547242

Tests Passing:



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public abstract class Command : IdentifiableObject
10     {
11         public Command(string[] ids) : base(ids) { }
12
13         public abstract string Execute(Player p, string[] text);
14     }
15 }
16
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Linq;
 5  using System.Text;
 6  using System.Threading.Tasks;
 7
 8  namespace SwinAdventure
 9  {
10      public class LookCommand : Command
11      {
12          public LookCommand() : base(new string[] {"look"}) { }
13
14          public override string Execute(Player p, string[] text)
15          {
16              if (!(text.Length == 3 || text.Length == 5))
17              {
18                  return "I don't know how to look like that";
19              }
20
21              if (text[0] != "look")
22              {
23                  return "Error in look input";
24              }
25
26              if (text[1] != "at")
27              {
28                  return "What do you want to look at?";
29              }
30
31              if (text.Length == 5 && text[3] != "in")
32              {
33                  return "What do you want to look in?";
34              }
35
36              if (text.Length == 3)
37              {
38                  string? itemDescription3 = LookAtIn(text[2], p);
39                  if (itemDescription3 == null)
40                  {
41                      return string.Format("I cannot find the {0}", text
                        [2]);
42                  }
43                  return itemDescription3;
44              }
45
46              // By this point the 3 element look command is done
47              IHaveInventory? container = FetchContainer(p, text[4]);
48              if (container == null)
49              {
50                  return string.Format("I cannot find the {0}", text[4]);
51              }
52
```

```
53              string? itemDescription5 = LookAtIn(text[2], container);
54              if (itemDescription5 == null)
55              {
56                  return string.Format("I cannot find the {0} in the {1}", ↪
                       text[2], text[4]);
57              }
58              return itemDescription5;
59          }
60
61          private IHaveInventory? FetchContainer(Player p, string          ↪
              containerId)
62          {
63              IHaveInventory? container = p.Locate(containerId) as         ↪
                  IHaveInventory;
64              return container;
65          }
66
67          private string? LookAtIn(string thingId, IHaveInventory          ↪
              container)
68          {
69              GameObject? item = container.Locate(thingId);
70              if (item == null)
71              {
72                  return null;
73              }
74              return item.FullDescription;
75          }
76      }
77  }
78
```

```
1  using SwinAdventure;
2
3
4  namespace TestLookCommand
5  {
6      public class Tests
7      {
8          private LookCommand look;
9          private Player testPlayer;
10         private Item gem;
11         private Bag bag;
12
13         [SetUp]
14         public void Setup()
15         {
16             look = new LookCommand();
17             testPlayer = new Player("testPlayer", "test player
                   description");
18             gem = new Item(new string[] { "gem" }, "a gem", "gem's
                   description");
19             bag = new Bag(new string[] { "bag" }, "a bag", "bag's
                   description");
20         }
21
22         [Test]
23         public void TestLookAtMe()
24         {
25             string testLookAtInventory = look.Execute(testPlayer, new
                   string[] { "look", "at", "inventory" });
26             Assert.That(testLookAtInventory, Is.EqualTo("You are
                   testPlayer, test player description.\nYou are carrying:
                   "));
27         }
28
29         [Test]
30         public void TestLookAtGem()
31         {
32             testPlayer.Inventory.Put(gem);
33             string testLookAtGem = look.Execute(testPlayer, new string[]
                   { "look", "at", "gem" });
34             Assert.That(testLookAtGem, Is.EqualTo("gem's description"));
35         }
36
37         [Test]
38         public void TestLookAtUnknown()
39         {
40             string testLookAtUnknown = look.Execute(testPlayer, new
                   string[] { "look", "at", "gem" });
41             Assert.That(testLookAtUnknown, Is.EqualTo("I cannot find the
                   gem"));
42         }
43
44         [Test]
```

```
45          public void TestLookAtGemInMe()
46          {
47              testPlayer.Inventory.Put(gem);
48              string testLookAtGem = look.Execute(testPlayer, new string[]
                    { "look", "at", "gem", "in", "inventory" });
49              Assert.That(testLookAtGem, Is.EqualTo("gem's description"));
50          }
51
52          [Test]
53          public void TestLookAtGemInBag()
54          {
55              bag.Inventory.Put(gem);
56              testPlayer.Inventory.Put(bag);
57              string testLookAtGemInBag = look.Execute(testPlayer, new
                    string[] { "look", "at", "gem", "in", "bag" });
58              Assert.That(testLookAtGemInBag, Is.EqualTo("gem's
                    description"));
59          }
60
61          [Test]
62          public void TestLookAtGemInNoBag()
63          {
64              string testLookAtGemInNoBag = look.Execute(testPlayer, new
                    string[] { "look", "at", "gem", "in", "bag" });
65              Assert.That(testLookAtGemInNoBag, Is.EqualTo("I cannot find
                    the bag"));
66          }
67
68          [Test]
69          public void TestLookAtNoGemInBag()
70          {
71              testPlayer.Inventory.Put(bag);
72              string testLookAtNoGemInBag = look.Execute(testPlayer, new
                    string[] { "look", "at", "gem", "in", "bag" });
73              Assert.That(testLookAtNoGemInBag, Is.EqualTo("I cannot find
                    the gem in the bag"));
74          }
75
76          [Test]
77          public void TestInvalidLook()
78          {
79              string testIncorrectTextLength = look.Execute(testPlayer,
                    new string[] { "testing", "incorrect", "text", "length" });
80              string testLookNotFirstWord = look.Execute(testPlayer, new
                    string[] { "testing", "look", "is", "not", "first" });
81              string testAtNotSecondWord = look.Execute(testPlayer, new
                    string[] { "look", "test", "at", "not", "second" });
82              string testInNotFourthWord = look.Execute(testPlayer, new
                    string[] { "look", "at", "in", "not", "fourth" });
83              Assert.That(testIncorrectTextLength, Is.EqualTo("I don't
                    know how to look like that"));
84              Assert.That(testLookNotFirstWord, Is.EqualTo("Error in look
                    input"));
```

```
85              Assert.That(testAtNotSecondWord, Is.EqualTo("What do you
                  want to look at?"));
86              Assert.That(testInNotFourthWord, Is.EqualTo("What do you
                  want to look in?"));
87          }
88      }
89  }
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace SwinAdventure
 8  {
 9      public interface IHaveInventory
10      {
11          public string Name { get; }
12
13          public GameObject? Locate(string id);
14      }
15  }
16
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public class Player : GameObject, IHaveInventory
10     {
11         private Inventory _inventory;
12
13         public override string FullDescription
14         {
15             get
16             {
17                 return string.Format("You are {0}, {1}.\nYou are
                       carrying: {2}", Name, base.FullDescription,
                       Inventory.ItemList);
18             }
19         }
20
21         public Inventory Inventory
22         {
23             get
24             {
25                 return _inventory;
26             }
27         }
28
29         public Player(string name, string desc) : base(new string[]
             {"me", "inventory"}, name, desc)
30         {
31             _inventory = new Inventory();
32         }
33
34         public GameObject? Locate(string id)
35         {
36             if (AreYou(id))
37             {
38                 return this;
39             }
40             return Inventory.Fetch(id);
41         }
42     }
43 }
44
```

```
1  using SwinAdventure;
2
3  namespace TestPlayer
4  {
5      public class Tests
6      {
7          [Test]
8          public void TestPlayerIsIdentifiable()
9          {
10             Player p = new Player("Tester", "the mighty test player");
11
12             bool testPMe = p.AreYou("me");
13             bool testPInventory = p.AreYou("inventory");
14             Assert.That(testPMe, Is.EqualTo(true));
15             Assert.That(testPInventory, Is.EqualTo(true));
16         }
17
18         [Test]
19         public void TestPlayerLocatesItems()
20         {
21             Player p = new Player("Tester", "the mighty test player");
22             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
23             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
24             p.Inventory.Put(shovel);
25             p.Inventory.Put(bronzeSword);
26
27             GameObject? testLocateShovel = p.Locate("shovel");
28             GameObject? testLocateBronzeSword = p.Locate("sword");
29             Assert.That(testLocateShovel, Is.EqualTo(shovel));
30             Assert.That(testLocateBronzeSword, Is.EqualTo(bronzeSword));
31         }
32
33         [Test]
34         public void TestPlayerLocatesItself()
35         {
36             Player p = new Player("Tester", "the mighty test player");
37
38             GameObject? testLocatePMe = p.Locate("me");
39             GameObject? testLocatePInventory = p.Locate("inventory");
40             Assert.That(testLocatePMe, Is.EqualTo(p));
41             Assert.That(testLocatePInventory, Is.EqualTo(p));
42         }
43
44         [Test]
45         public void TestPlayerLocatesNothing()
46         {
47             Player p = new Player("Tester", "the mighty test player");
48             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
49             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
```

```
50              p.Inventory.Put(shovel);
51              p.Inventory.Put(bronzeSword);
52
53              GameObject? testLocateNothing = p.Locate("nothing");
54              Assert.That(testLocateNothing, Is.EqualTo(null));
55          }
56
57          [Test]
58          public void TestPlayerFullDescription()
59          {
60              Player p = new Player("Tester", "the mighty test player");
61              Item shovel = new Item(new string[] { "shovel", "spade" },
                  "a shovel", "shovel description");
62              Item bronzeSword = new Item(new string[] { "sword", "bronze
                  sword" }, "a bronze sword", "bronze sword description");
63              p.Inventory.Put(shovel);
64              p.Inventory.Put(bronzeSword);
65
66              string testFullDescription = p.FullDescription;
67              Assert.That(testFullDescription, Is.EqualTo("You are Tester,
                   the mighty test player.\nYou are carrying: \n  a shovel
                  (shovel)\n  a bronze sword (sword)"));
68
69          }
70      }
71  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public class Bag : Item, IHaveInventory
10     {
11         private Inventory _inventory;
12
13         public override string FullDescription
14         {
15             get
16             {
17                 return string.Format("In the {0} you can see: {1}",
                       Name, Inventory.ItemList);
18             }
19         }
20
21         public Inventory Inventory
22         {
23             get
24             {
25                 return _inventory;
26             }
27         }
28
29         public Bag(string[] ids, string name, string desc): base(ids,
             name, desc)
30         {
31             _inventory = new Inventory();
32         }
33
34         public GameObject? Locate(string id)
35         {
36             if (AreYou(id))
37             {
38                 return this;
39             }
40             return Inventory.Fetch(id);
41         }
42     }
43 }
44
```

```
1  using SwinAdventure;
2
3  namespace TestBag
4  {
5      public class Tests
6      {
7          [Test]
8          public void TestBagLocatesItems()
9          {
10             Bag testBag = new Bag(new string[] { "bag", "testingBag"},
                 "test bag", "this is the test bag's description");
11             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
12             testBag.Inventory.Put(shovel);
13
14             GameObject? testLocateShovel = testBag.Locate("shovel");
15             GameObject? testShovelRemainsInBag = testBag.Locate
                 ("shovel");
16             Assert.That(testLocateShovel, Is.EqualTo(shovel));
17             Assert.That(testShovelRemainsInBag, Is.EqualTo(shovel));
18         }
19
20         [Test]
21         public void TestBagLocatesItself()
22         {
23             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
                 "test bag", "this is the test bag's description");
24
25             GameObject? testLocateBagID1 = testBag.Locate("bag");
26             GameObject? testLocateBagID2 = testBag.Locate("testingBag");
27             Assert.That(testLocateBagID1, Is.EqualTo(testBag));
28             Assert.That(testLocateBagID2, Is.EqualTo(testBag));
29         }
30
31         [Test]
32         public void TestBagLocatesNothing()
33         {
34             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
                 "test bag", "this is the test bag's description");
35
36             GameObject? testLocateShovel = testBag.Locate("shovel");
37             Assert.That(testLocateShovel, Is.EqualTo(null));
38         }
39
40         [Test]
41         public void TestBagFullDescription()
42         {
43             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
                 "test bag", "this is the test bag's description");
44             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
45             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
```

```
46              testBag.Inventory.Put(shovel);
47              testBag.Inventory.Put(bronzeSword);
48
49              string testBagFullDescription = testBag.FullDescription;
50              Assert.That(testBagFullDescription, Is.EqualTo("In the test bag you can see: \n  a shovel (shovel)\n  a bronze sword (sword)"));
51          }
52
53          [Test]
54          public void TestBagInBag()
55          {
56              Bag b1 = new Bag(new string[] { "bag", "testingBag1" }, "test bag 1", "this is test bag 1's description");
57              Bag b2 = new Bag(new string[] { "bag", "testingBag2" }, "test bag 2", "this is test bag 2's description");
58              Item shovel = new Item(new string[] { "shovel", "spade" }, "a shovel", "shovel description");
59              Item bronzeSword = new Item(new string[] { "sword", "bronze sword" }, "a bronze sword", "bronze sword description");
60              b1.Inventory.Put(shovel);
61              b2.Inventory.Put(bronzeSword);
62              b1.Inventory.Put(b2);
63
64              GameObject? testB1LocatesB2 = b1.Locate("testingBag2");
65              GameObject? testB1LocatesShovel = b1.Locate("shovel");
66              GameObject? testB1LocatesBronzeSword = b1.Locate("sword");
67              Assert.That(testB1LocatesB2, Is.EqualTo(b2));
68              Assert.That(testB1LocatesShovel, Is.EqualTo(shovel));
69              Assert.That(testB1LocatesBronzeSword, Is.EqualTo(null));
70          }
71      }
72  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.Design;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace SwinAdventure
9  {
10     public abstract class GameObject : IdentifiableObject
11     {
12         private string _description;
13         private string _name;
14
15         public string Name
16         {
17             get
18             {
19                 return _name;
20             }
21         }
22
23         public string ShortDescription
24         {
25             get
26             {
27                 return string.Format("{0} ({1})", Name, base.FirstID);
28             }
29         }
30
31         public virtual string FullDescription
32         {
33             get
34             {
35                 return _description;
36             }
37         }
38
39         public GameObject(string[] ids, string name, string desc) : base ↵
             (ids)
40         {
41             _name = name;
42             _description = desc;
43         }
44     }
45 }
46
```

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

```
1  using SwinAdventure;
2  using System.Reflection.Metadata;
3
4  namespace TestItem
5  {
6      public class Tests
7      {
8          [Test]
9          public void TestItemIsIdentifiable()
10         {
11             // testing identifiers of Item object
12             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
13             bool testBronzeSwordID1 = bronzeSword.AreYou("sword");
14             bool testBronzeSwordID2 = bronzeSword.AreYou("bronze
                 sword");
15             Assert.That(testBronzeSwordID1, Is.EqualTo(true));
16             Assert.That(testBronzeSwordID2, Is.EqualTo(true));
17         }
18
19         [Test]
20
21         public void TestItemShortDescription()
22         {
23             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
24             string testBronzeSword = bronzeSword.ShortDescription;
25             Assert.That(testBronzeSword, Is.EqualTo("a bronze sword
                 (sword)"));
26         }
27
28         [Test]
29
30         public void TestItemFullDescription()
31         {
32             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
33             string testBronzeSword = bronzeSword.FullDescription;
34             Assert.That(testBronzeSword, Is.EqualTo("bronze sword
                 description"));
35         }
36     }
37 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public class Inventory
10     {
11         private List<Item> _items;
12
13         public string ItemList
14         {
15             get
16             {
17                 string itemList = "";
18                 foreach (Item item in _items)
19                 {
20                     itemList += (string.Format("\n  {0}",
                       item.ShortDescription));
21                 }
22
23                 return itemList;
24             }
25         }
26
27         public Inventory()
28         {
29             _items = new List<Item>();
30         }
31
32         public bool HasItem(string id)
33         {
34             return Fetch(id) != null;
35         }
36
37         public void Put(Item itm)
38         {
39             _items.Add(itm);
40         }
41
42         public Item? Take(string id)
43         {
44             foreach (Item item in _items)
45             {
46                 if (item.AreYou(id))
47                 {
48                     _items.Remove(item);
49                     return item;
50                 }
51             }
52             return null;
```

```
53          }
54
55          public Item? Fetch(string id)
56          {
57              foreach (Item item in _items)
58              {
59                  if (item.AreYou(id))
60                  {
61                      return item;
62                  }
63              }
64              return null;
65          }
66
67      }
68  }
69
```

```
1  using SwinAdventure;
2
3  namespace TestInventory
4  {
5      public class Tests
6      {
7
8          [Test]
9          public void TestFindItem()
10         {
11             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
12             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
13             Inventory testInventory = new Inventory();
14             testInventory.Put(shovel);
15             testInventory.Put(bronzeSword);
16
17             bool testShovel = testInventory.HasItem("shovel");
18             bool testBronzeSword = testInventory.HasItem("sword");
19             Assert.That(testShovel, Is.EqualTo(true));
20             Assert.That(testBronzeSword, Is.EqualTo(true));
21         }
22
23         [Test]
24         public void TestNoItemFind()
25         {
26             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
27             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
28             Inventory testInventory = new Inventory();
29             testInventory.Put(shovel);
30             testInventory.Put(bronzeSword);
31
32             bool testSmallComputer = testInventory.HasItem("pc");
33             Assert.That(testSmallComputer, Is.EqualTo(false));
34         }
35
36         [Test]
37         public void TestFetchItem()
38         {
39             Item shovel = new Item(new string[] { "shovel", "spade" },
                 "a shovel", "shovel description");
40             Item bronzeSword = new Item(new string[] { "sword", "bronze
                 sword" }, "a bronze sword", "bronze sword description");
41             Inventory testInventory = new Inventory();
42             testInventory.Put(shovel);
43             testInventory.Put(bronzeSword);
44
45             Item? testShovel = testInventory.Fetch("shovel");
46             Item? testBronzeSword = testInventory.Fetch("sword");
47             Assert.That(testShovel, Is.EqualTo(shovel));
```

```
48              Assert.That(testBronzeSword, Is.EqualTo(bronzeSword));
49          }
50
51          [Test]
52          public void TestTakeItem()
53          {
54              Item shovel = new Item(new string[] { "shovel", "spade" },
                  "a shovel", "shovel description");
55              Item bronzeSword = new Item(new string[] { "sword", "bronze
                  sword" }, "a bronze sword", "bronze sword description");
56              Inventory testInventory = new Inventory();
57              testInventory.Put(shovel);
58              testInventory.Put(bronzeSword);
59
60              Item? testFetchShovel = testInventory.Take("shovel");
61              bool testShovelInInventory = testInventory.HasItem
                  ("shovel");
62              Assert.That(testFetchShovel, Is.EqualTo(shovel));
63              Assert.That(testShovelInInventory, Is.EqualTo(false));
64          }
65
66          [Test]
67          public void TestItemList()
68          {
69              Item shovel = new Item(new string[] { "shovel", "spade" },
                  "a shovel", "shovel description");
70              Item bronzeSword = new Item(new string[] { "sword", "bronze
                  sword" }, "a bronze sword", "bronze sword description");
71              Inventory testInventory = new Inventory();
72              testInventory.Put(shovel);
73              testInventory.Put(bronzeSword);
74
75              string testInventoryList = testInventory.ItemList;
76              Assert.That(testInventoryList, Is.EqualTo("\n  a shovel
                  (shovel)\n  a bronze sword (sword)"));
77          }
78      }
79  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SwinAdventure
8  {
9      public class IdentifiableObject
10     {
11         private List<string> _identifiers;
12
13         public string FirstID
14         {
15             get
16             {
17                 if (_identifiers.Count > 0)
18                 {
19                     return _identifiers[0];
20                 }
21                 return "";
22             }
23         }
24
25         public IdentifiableObject(string[] idents)
26         {
27             _identifiers = new List<string>();
28             for (int i = 0; i < idents.Length; i++)
29             {
30                 _identifiers.Add(idents[i].ToLower());
31             }
32         }
33
34         public bool AreYou(string id)
35         {
36             bool result = false;
37
38             foreach (string ident in _identifiers)
39             {
40                 if (ident == id.ToLower())
41                 {
42                     result = true;
43                     break;
44                 }
45             }
46
47             return result;
48         }
49
50         public void AddIdentifier(string id)
51         {
52             _identifiers.Add(id.ToLower());
53         }
```

```
54      }
55  }
56
```

```
1  using SwinAdventure;
2
3  namespace TestIdentifiableObject
4  {
5      public class Tests
6      {
7
8          [Test]
9          public void TestAreYou()
10         {
11             IdentifiableObject myIdents = new IdentifiableObject(new
                 string[] { "fred", "bob" });
12
13             bool fred = myIdents.AreYou("fred");
14             Assert.That(fred, Is.EqualTo(true));
15             bool bob = myIdents.AreYou("bob");
16             Assert.That(bob, Is.EqualTo(true));
17         }
18
19         [Test]
20
21         public void TestNotAreYou()
22         {
23             IdentifiableObject myIdents = new IdentifiableObject(new
                 string[] { "fred", "bob" });
24
25             bool wilma = myIdents.AreYou("wilma");
26             Assert.That(wilma, Is.EqualTo(false));
27             bool boby = myIdents.AreYou("boby");
28             Assert.That(boby, Is.EqualTo(false));
29         }
30
31         [Test]
32
33         public void TestCaseSensitive()
34         {
35             IdentifiableObject myIdents = new IdentifiableObject(new
                 string[] { "fred", "bob" });
36
37             bool fred = myIdents.AreYou("FRED");
38             Assert.That(fred, Is.EqualTo(true));
39             bool bob = myIdents.AreYou("bOB");
40             Assert.That(bob, Is.EqualTo(true));
41         }
42
43         [Test]
44
45         public void TestFirstID()
46         {
47             IdentifiableObject myIdents = new IdentifiableObject(new
                 string[] { "fred", "bob" });
48
49             string firstID = myIdents.FirstID;
```

```
50              Assert.That(firstID, Is.EqualTo("fred"));
51          }
52
53          [Test]
54
55          public void TestFirstIDNoIDs()
56          {
57              IdentifiableObject myIdents = new IdentifiableObject(new
                  string[] {});
58
59              string firstID = myIdents.FirstID;
60              Assert.That(firstID, Is.EqualTo(""));
61          }
62
63          [Test]
64
65          public void TestAddIDs()
66          {
67              IdentifiableObject myIdents = new IdentifiableObject(new
                  string[] { "fred", "bob" });
68              myIdents.AddIdentifier("wilma");
69
70              bool fred = myIdents.AreYou("fred");
71              Assert.That(fred, Is.EqualTo(true));
72              bool bob = myIdents.AreYou("bob");
73              Assert.That(bob, Is.EqualTo(true));
74              bool wilma = myIdents.AreYou("wilma");
75              Assert.That(wilma, Is.EqualTo(true));
76          }
77      }
78  }
```