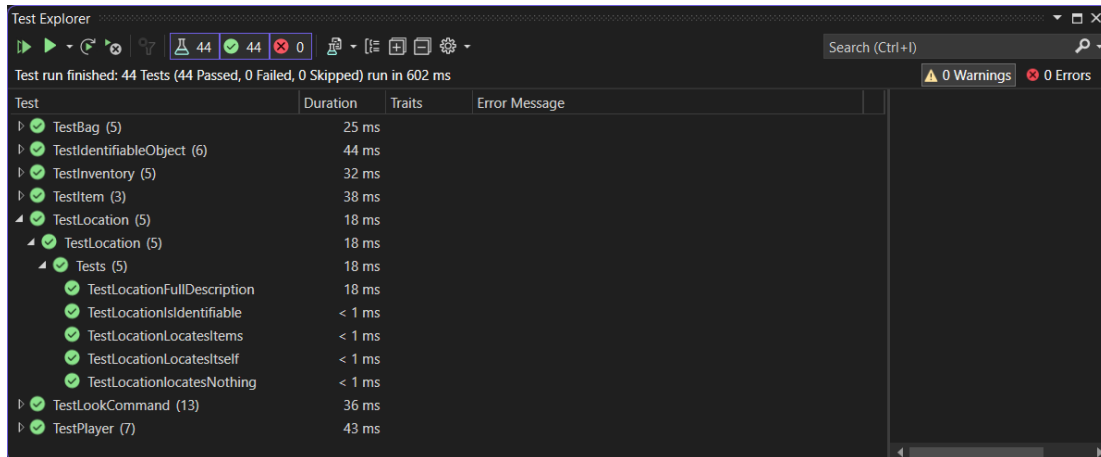


7.2C - Case Study - Iteration 6 – Locations

Jayden Kong, 104547242

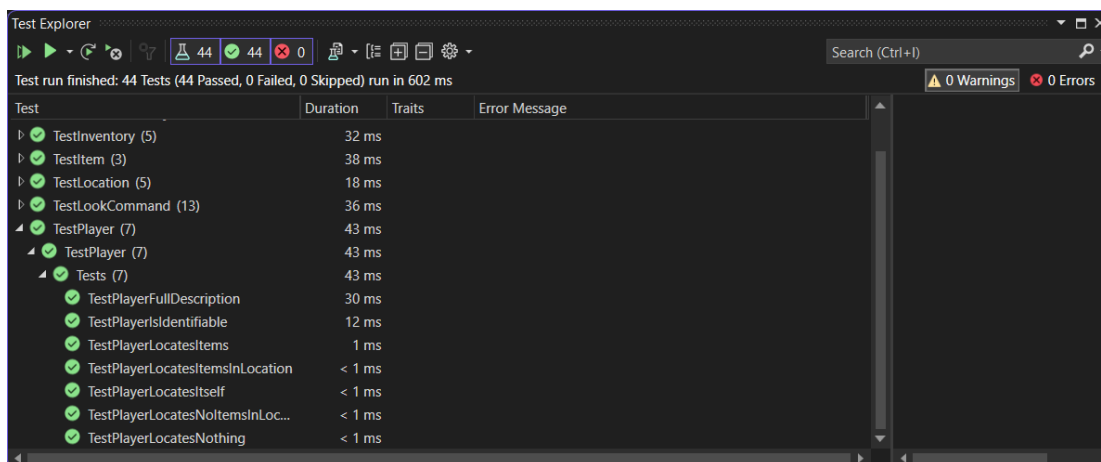
Location tests passing:



The screenshot shows the Test Explorer window with a test run summary at the top: "Test run finished: 44 Tests (44 Passed, 0 Failed, 0 Skipped) run in 602 ms". Below this, a table lists the tests and their durations. The tests are grouped by category, with expandable icons (plus and minus) next to the category names. The tests are all marked with a green checkmark, indicating they passed.

Test	Duration	Traits	Error Message
TestBag (5)	25 ms		
TestIdentifiableObject (6)	44 ms		
TestInventory (5)	32 ms		
TestItem (3)	38 ms		
TestLocation (5)	18 ms		
TestLocation (5)	18 ms		
Tests (5)	18 ms		
TestLocationFullDescription	18 ms		
TestLocationIsIdentifiable	< 1 ms		
TestLocationLocatesItems	< 1 ms		
TestLocationLocatesItself	< 1 ms		
TestLocationLocatesNothing	< 1 ms		
TestLookCommand (13)	36 ms		
TestPlayer (7)	43 ms		

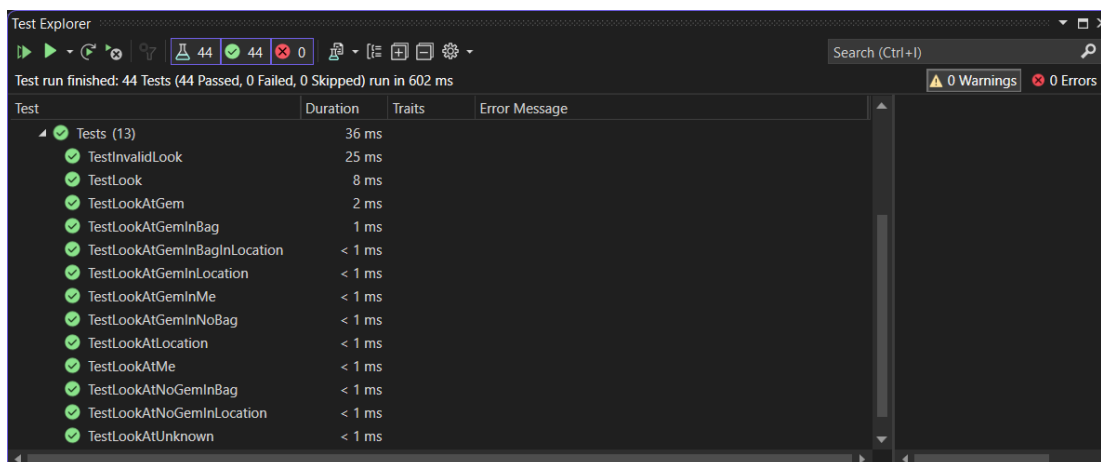
Player tests passing:



The screenshot shows the Test Explorer window with a test run summary at the top: "Test run finished: 44 Tests (44 Passed, 0 Failed, 0 Skipped) run in 602 ms". Below this, a table lists the tests and their durations. The tests are grouped by category, with expandable icons (plus and minus) next to the category names. The tests are all marked with a green checkmark, indicating they passed.

Test	Duration	Traits	Error Message
TestInventory (5)	32 ms		
TestItem (3)	38 ms		
TestLocation (5)	18 ms		
TestLookCommand (13)	36 ms		
TestPlayer (7)	43 ms		
TestPlayer (7)	43 ms		
Tests (7)	43 ms		
TestPlayerFullDescription	30 ms		
TestPlayerIsIdentifiable	12 ms		
TestPlayerLocatesItems	1 ms		
TestPlayerLocatesItemsInLocation	< 1 ms		
TestPlayerLocatesItself	< 1 ms		
TestPlayerLocatesNoItemsInLoc...	< 1 ms		
TestPlayerLocatesNothing	< 1 ms		

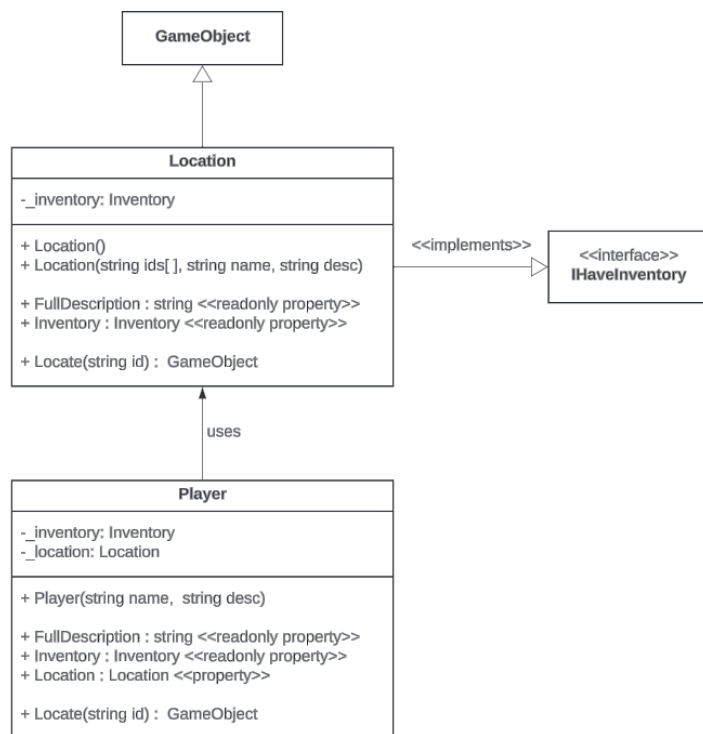
Look command tests passing:



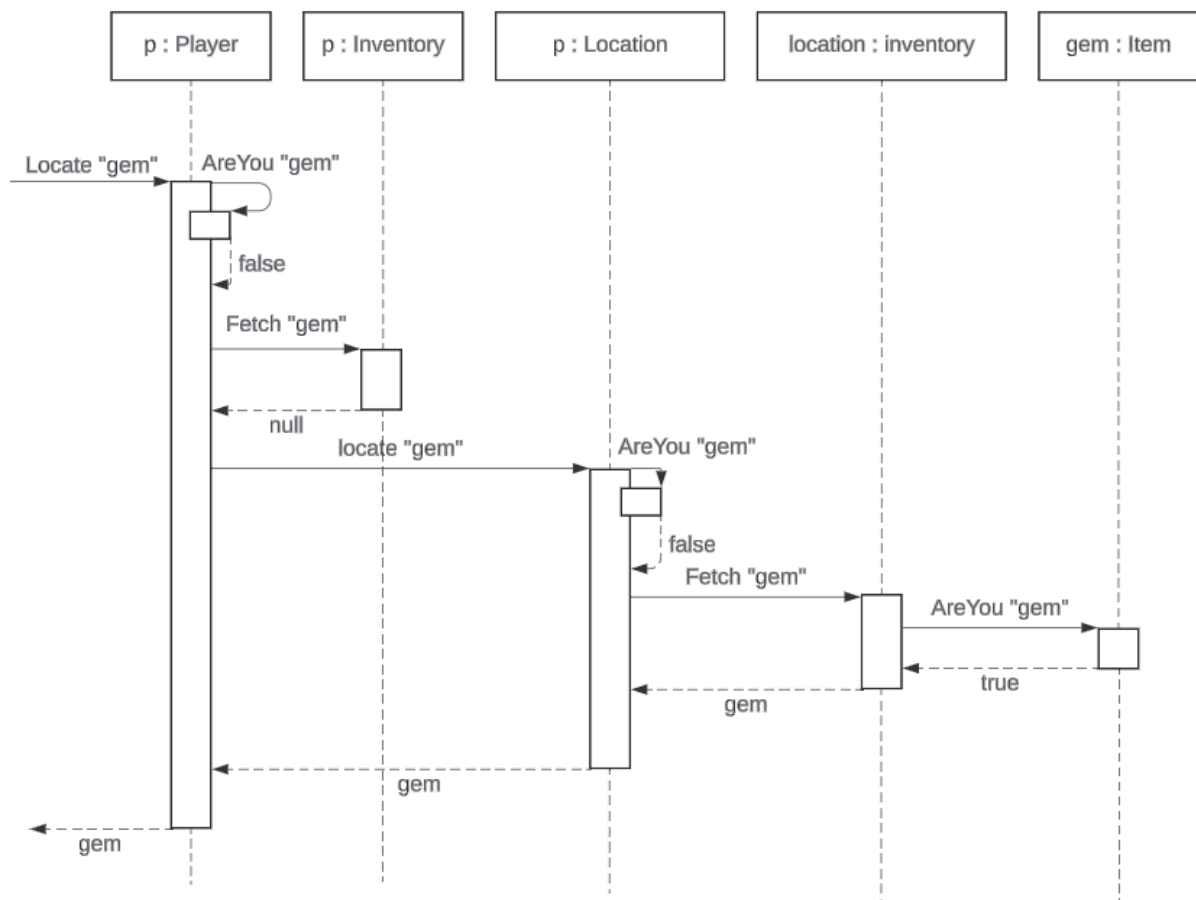
The screenshot shows the Test Explorer window with a test run summary at the top: "Test run finished: 44 Tests (44 Passed, 0 Failed, 0 Skipped) run in 602 ms". Below this, a table lists the tests and their durations. The tests are grouped by category, with expandable icons (plus and minus) next to the category names. The tests are all marked with a green checkmark, indicating they passed.

Test	Duration	Traits	Error Message
Tests (13)	36 ms		
TestInvalidLook	25 ms		
TestLook	8 ms		
TestLookAtGem	2 ms		
TestLookAtGemInBag	1 ms		
TestLookAtGemInBagInLocation	< 1 ms		
TestLookAtGemInLocation	< 1 ms		
TestLookAtGemInMe	< 1 ms		
TestLookAtGemInNoBag	< 1 ms		
TestLookAtLocation	< 1 ms		
TestLookAtMe	< 1 ms		
TestLookAtNoGemInBag	< 1 ms		
TestLookAtNoGemInLocation	< 1 ms		
TestLookAtUnknown	< 1 ms		

UML Diagram:



UML Sequence Diagram:



Program running:

```
C:\Users\Jayden Kong\OneDri x + v
Please enter your name -> Jayden
How would you describe yourself? -> a human
You are Jayden, a human.
Is this correct? (yes/no) -> yes
-----
Welcome to Swin Adventure!
You have arrived in the Classroom
Command -> look

You are in the Classroom
This is a dimly lit classroom
In this room you can see:
  a small computer (pc)
  laptop bag (laptop_bag)
Command -> look at classroom

You are in the Classroom
This is a dimly lit classroom
In this room you can see:
  a small computer (pc)
  laptop bag (laptop_bag)
Command -> look at pc

A dusty PC with a flickering screen
Command -> look at laptop_bag

In the laptop bag you can see:
  a laptop (laptop)
Command -> look at laptop in laptop_bag

A compact, modern laptop with a matte black finish
Command -> |
```

```
1 namespace SwinAdventure
2 {
3     internal class Program
4     {
5         static void Main(string[] args)
6         {
7             Player? player = null;
8             Command lookCommand = new LookCommand();
9             while (player == null)
10            {
11                Console.Write("Please enter your name -> ");
12                string? playerName = Console.ReadLine();
13                Console.Write("How would you describe yourself? -> ");
14                string? playerDescription = Console.ReadLine();
15                Console.Write("You are {0}, {1}.\nIs this correct? (yes/no) ↗
16                    -> ", playerName, playerDescription);
17                bool confirmationMenuLoop = true;
18                while (confirmationMenuLoop)
19                {
20                    string? decision = Console.ReadLine().ToLower();
21                    switch (decision)
22                    {
23                        case "yes":
24                            player = new Player(playerName,
25                                playerDescription); ↗
26                            confirmationMenuLoop = false;
27                            break;
28                        case "no":
29                            confirmationMenuLoop = false;
30                            break;
31                        default:
32                            Console.Write("Invalid option: please enter yes ↗
33                                or no. -> ");
34                            break;
35                    }
36                }
37            }
38
39            Item shovel = new Item(new string[] { "shovel", "spade" }, "a ↗
40                shovel", "A sturdy shovel, the perfect tool for digging");
41            Item bronzeSword = new Item(new string[] { "sword" }, "a bronze ↗
42                sword", "A short sword forged from bronze");
43            Item ruby = new Item(new string[] { "gem", "ruby" }, "a red ↗
44                gem", "A brilliant ruby, glows with a fiery red hue");
45            Bag bag = new Bag(new string[] { "bag" }, "leather bag", ↗
46                "Crafted from supple brown leather, this small bag is perfect ↗
47                for carrying items");
48            Item computer = new Item(new string[] { "pc" }, "a small ↗
49                computer", "A dusty PC with a flickering screen");
```

```

...versity\Year 2\COS20007\7.2C\SwinAdventure\Program.cs 2
41      Bag laptopBag = new Bag(new string[] { "laptop_bag", "bag" },  ↗
      "laptop bag", "A sleek, black laptop bag. Its fabric is  ↗
      slightly worn from use");
42      Item laptop = new Item(new string[] { "laptop" }, "a laptop",  ↗
      "A compact, modern laptop with a matte black finish");
43      Location classroom = new Location(new string[] { "classroom",  ↗
      "location" }, "the Classroom", "This is a dimly lit  ↗
      classroom");          // Player will initially be in the  ↗
      classroom
44
45      player.Location = classroom;
46      laptopBag.Inventory.Put(laptop);
47      player.Location.Inventory.Put(computer);
48      player.Location.Inventory.Put(laptopBag);
49      bag.Inventory.Put(ruby);
50      player.Inventory.Put(shovel);
51      player.Inventory.Put(bronzeSword);
52      player.Inventory.Put(bag);
53
54      Console.WriteLine("-----");
55      Console.WriteLine("Welcome to Swin Adventure!");
56      Console.WriteLine("You have arrived in {0}",  ↗
      player.Location.Name);
57
58      bool gameLoop = true;
59      while (gameLoop)
60      {
61          Console.Write("Command -> ");
62          string? playerInput = Console.ReadLine();
63          string[] inputToPass = playerInput.Split(new char[] { },  ↗
      StringSplitOptions.RemoveEmptyEntries);
64          Console.WriteLine("");
65          Console.WriteLine(lookCommand.Execute(player,  ↗
      inputToPass));
66      }
67  }
68 }
69 }
70

```

```
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 class Location : GameObject, IHaveInventory
11     {
12         private Inventory _inventory;
13
14         public override string FullDescription
15         {
16             get
17             {
18                 return string.Format("You are in {0}\n{1}\nIn this room you can see:{2}", Name, base.FullDescription,
19                                     Inventory.ItemList);
20             }
21         }
22         public Inventory Inventory
23         {
24             get
25             {
26                 return _inventory;
27             }
28         }
29         public Location() : this(new string[] { "location", "unknown" },
30                                 "an unknown location", "This is a mysterious location")
31         { } // Default constructor, to make sure the player has a location if not allocated one
32         public Location(string[] ids, string name, string desc) : base(ids, name, desc)
33         {
34             _inventory = new Inventory();
35         }
36         public GameObject? Locate(string id)
37         {
38             if (AreYou(id))
39             {
40                 return this;
41             }
42             return Inventory.Fetch(id);
43         }
44     }
45 }
```

44 }

45

```
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        private Location _location;
13
14        public override string FullDescription
15        {
16            get
17            {
18                return string.Format("You are {0}, {1}.\nYou are carrying: {2}", Name, base.FullDescription, Inventory.ItemList);
19            }
20        }
21
22        public Inventory Inventory
23        {
24            get
25            {
26                return _inventory;
27            }
28        }
29
30        public Location Location
31        {
32            get
33            {
34                return _location;
35            }
36            set
37            {
38                _location = value;
39            }
40        }
41
42        public Player(string name, string desc) : base(new string[] { "me", "inventory" }, name, desc)
43        {
44            _inventory = new Inventory();
45            _location = new Location();
46        }
47    }
```



```
48     public GameObject? Locate(string id)
49     {
50         if (AreYou(id))
51         {
52             return this;
53         }
54         GameObject? item = Inventory.Fetch(id);
55         if (item != null)
56         {
57             return item;
58         }
59         item = Location.Locate(id);
60         return item;
61     }
62 }
63 }
64
```

```
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 == 1 || 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.Length == 1)
27             {
28                 string locationDescription = p.Location.FullDescription;
29                 return locationDescription;
30             }
31
32             if (text[1] != "at")
33             {
34                 return "What do you want to look at?";
35             }
36
37             if (text.Length == 5 && text[3] != "in")
38             {
39                 return "What do you want to look in?";
40             }
41
42             if (text.Length == 3)
43             {
44                 string? itemDescription3 = LookAtIn(text[2], p);
45                 if (itemDescription3 == null)
46                 {
47                     return string.Format("I cannot find the {0}", text[2]);
48                 }
49             }
50         }
51     }
52 }
```

```
49         return itemDescription3;
50     }
51
52     // By this point the 1 and 3 element look commands are done
53     IHaveInventory? container = FetchContainer(p, text[4]);
54     if (container == null)
55     {
56         return string.Format("I cannot find the {0}", text[4]);
57     }
58
59     string? itemDescription5 = LookAtIn(text[2], container);
60     if (itemDescription5 == null)
61     {
62         return string.Format("I cannot find the {0} in the {1}",
63                               text[2], text[4]);
64     }
65     return itemDescription5;
66 }
67 private IHaveInventory? FetchContainer(Player p, string
68     containerId)
69 {
70     IHaveInventory? container = p.Locate(containerId) as
71     IHaveInventory;
72     return container;
73 }
74 private string? LookAtIn(string thingId, IHaveInventory container)
75 {
76     GameObject? item = container.Locate(thingId);
77     if (item == null)
78     {
79         return null;
80     }
81     return item.FullDescription;
82 }
83 }
84
```

```
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,
18                                     Inventory.ItemList);
19            }
20        }
21
22        public Inventory Inventory
23        {
24            get
25            {
26                return _inventory;
27            }
28        }
29
30        public Bag(string[] ids, string name, string desc): base(ids, name,
31                                                                desc)
32        {
33            _inventory = new Inventory();
34        }
35
36        public GameObject? Locate(string id)
37        {
38            if (AreYou(id))
39            {
40                return this;
41            }
42            return Inventory.Fetch(id);
43        }
44    }
```

```
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 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.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 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 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}",
21                                                item.ShortDescription));
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 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
12            (idents, name, desc) { }
13    }
14 }
```

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

```
46     public void TestPlayerLocatesItself()
47     {
48         GameObject? testLocatePMe = p.Locate("me");
49         GameObject? testLocatePInventory = p.Locate("inventory");
50         Assert.That(testLocatePMe, Is.EqualTo(p));
51         Assert.That(testLocatePInventory, Is.EqualTo(p));
52     }
53
54     [Test]
55     public void TestPlayerLocatesNothing()
56     {
57         p.Inventory.Put(shovel);
58         p.Inventory.Put(bronzeSword);
59
60         GameObject? testLocateNothing = p.Locate("nothing");
61         Assert.That(testLocateNothing, Is.EqualTo(null));
62     }
63
64     [Test]
65     public void TestPlayerFullDescription()
66     {
67         p.Inventory.Put(shovel);
68         p.Inventory.Put(bronzeSword);
69
70         string testFullDescription = p.FullDescription;
71         Assert.That(testFullDescription, Is.EqualTo("You are Tester,
72             the mighty test player.\nYou are carrying: \n  a shovel
73             (shovel)\n  a bronze sword (sword)"));
74     }
75
76     [Test]
77     public void TestPlayerLocatesItemsInLocation()
78     {
79         location.Inventory.Put(ruby);
80         p.Location = location;
81
82         GameObject? testPlayerLocatesRubyInLocation = p.Locate("ruby");
83         Assert.That(testPlayerLocatesRubyInLocation, Is.EqualTo(ruby));
84     }
85
86     [Test]
87     public void TestPlayerLocatesNoItemsInLocation()
88     {
89         p.Location = location;
90
91         GameObject? testPlayerLocatesNothingInLocation = p.Locate
92             ("nothing");
93         Assert.That(testPlayerLocatesNothingInLocation, Is.EqualTo
94             (null));
```

```
91         }  
92     }  
93 }
```

```
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 myIdsents = new IdentifiableObject(new string [] { "fred", "bob" });
48
49     string firstID = myIdsents.FirstID;
50     Assert.That(firstID, Is.EqualTo("fred"));
51 }
52
53 [Test]
54
55 public void TestFirstIDNoIDs()
56 {
57     IdentifiableObject myIdsents = new IdentifiableObject(new string [] {});
58
59     string firstID = myIdsents.FirstID;
60     Assert.That(firstID, Is.EqualTo(""));
61 }
62
63 [Test]
64
65 public void TestAddIDs()
66 {
67     IdentifiableObject myIdsents = new IdentifiableObject(new string [] { "fred", "bob" });
68     myIdsents.AddIdentifier("wilma");
69
70     bool fred = myIdsents.AreYou("fred");
71     Assert.That(fred, Is.EqualTo(true));
72     bool bob = myIdsents.AreYou("bob");
73     Assert.That(bob, Is.EqualTo(true));
74     bool wilma = myIdsents.AreYou("wilma");
75     Assert.That(wilma, Is.EqualTo(true));
76 }
77 }
78 }
```

```
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
12                shovel", "shovel description");
13            Item bronzeSword = new Item(new string[] { "sword", "bronze
14                sword" }, "a bronze sword", "bronze sword description");
15            Inventory testInventory = new Inventory();
16            testInventory.Put(shovel);
17            testInventory.Put(bronzeSword);
18
19            bool testShovel = testInventory.HasItem("shovel");
20            bool testBronzeSword = testInventory.HasItem("sword");
21            Assert.That(testShovel, Is.EqualTo(true));
22            Assert.That(testBronzeSword, Is.EqualTo(true));
23        }
24
25        [Test]
26        public void TestNoItemFind()
27        {
28            Item shovel = new Item(new string[] { "shovel", "spade" }, "a
29                shovel", "shovel description");
30            Item bronzeSword = new Item(new string[] { "sword", "bronze
31                sword" }, "a bronze sword", "bronze sword description");
32            Inventory testInventory = new Inventory();
33            testInventory.Put(shovel);
34            testInventory.Put(bronzeSword);
35
36            bool testSmallComputer = testInventory.HasItem("pc");
37            Assert.That(testSmallComputer, Is.EqualTo(false));
38        }
39
40        [Test]
41        public void TestFetchItem()
42        {
43            Item shovel = new Item(new string[] { "shovel", "spade" }, "a
44                shovel", "shovel description");
45            Item bronzeSword = new Item(new string[] { "sword", "bronze
46                sword" }, "a bronze sword", "bronze sword description");
47            Inventory testInventory = new Inventory();
48            testInventory.Put(shovel);
49            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
55 shovel", "shovel description");
56     Item bronzeSword = new Item(new string[] { "sword", "bronze
57 sword" }, "a bronze sword", "bronze sword description");
58     Inventory testInventory = new Inventory();
59     testInventory.Put(shovel);
60     testInventory.Put(bronzeSword);
61
62     Item? testFetchShovel = testInventory.Take("shovel");
63     bool testShovelInInventory = testInventory.HasItem("shovel");
64     Assert.That(testFetchShovel, Is.EqualTo(shovel));
65     Assert.That(testShovelInInventory, Is.EqualTo(false));
66 }
67
68 [Test]
69 public void TestItemList()
70 {
71     Item shovel = new Item(new string[] { "shovel", "spade" }, "a
72 shovel", "shovel description");
73     Item bronzeSword = new Item(new string[] { "sword", "bronze
74 sword" }, "a bronze sword", "bronze sword description");
75     Inventory testInventory = new Inventory();
76     testInventory.Put(shovel);
77     testInventory.Put(bronzeSword);
78
79     string testInventoryList = testInventory.ItemList;
80     Assert.That(testInventoryList, Is.EqualTo("\n a shovel
81 (shovel)\n a bronze sword (sword)"));
82 }
83 }
```

```
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"},
11                                     "test bag", "this is the test bag's description");
12             Item shovel = new Item(new string[] { "shovel", "spade" }, "a
13                                     shovel", "shovel description");
14             testBag.Inventory.Put(shovel);
15
16             GameObject? testLocateShovel = testBag.Locate("shovel");
17             GameObject? testShovelRemainsInBag = testBag.Locate("shovel");
18             Assert.That(testLocateShovel, Is.EqualTo(shovel));
19             Assert.That(testShovelRemainsInBag, Is.EqualTo(shovel));
20         }
21
22         [Test]
23         public void TestBagLocatesItself()
24         {
25             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
26                                     "test bag", "this is the test bag's description");
27
28             GameObject? testLocateBagID1 = testBag.Locate("bag");
29             GameObject? testLocateBagID2 = testBag.Locate("testingBag");
30             Assert.That(testLocateBagID1, Is.EqualTo(testBag));
31             Assert.That(testLocateBagID2, Is.EqualTo(testBag));
32         }
33
34         [Test]
35         public void TestBagLocatesNothing()
36         {
37             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
38                                     "test bag", "this is the test bag's description");
39
40             GameObject? testLocateShovel = testBag.Locate("shovel");
41             Assert.That(testLocateShovel, Is.EqualTo(null));
42         }
43
44         [Test]
45         public void TestBagFullDescription()
46         {
47             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
48                                     "test bag", "this is the test bag's description");
49             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 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
13                sword" }, "a bronze sword", "bronze sword description");
14            bool testBronzeSwordID1 = bronzeSword.AreYou("sword");
15            bool testBronzeSwordID2 = bronzeSword.AreYou("bronze sword");
16            Assert.That(testBronzeSwordID1, Is.EqualTo(true));
17            Assert.That(testBronzeSwordID2, Is.EqualTo(true));
18        }
19
20        [Test]
21        public void TestItemShortDescription()
22        {
23            Item bronzeSword = new Item(new string[] { "sword", "bronze
24                sword" }, "a bronze sword", "bronze sword description");
25            string testBronzeSword = bronzeSword.ShortDescription;
26            Assert.That(testBronzeSword, Is.EqualTo("a bronze sword
27                (sword)"));
28        }
29
30        [Test]
31        public void TestItemFullDescription()
32        {
33            Item bronzeSword = new Item(new string[] { "sword", "bronze
34                sword" }, "a bronze sword", "bronze sword description");
35            string testBronzeSword = bronzeSword.FullDescription;
36            Assert.That(testBronzeSword, Is.EqualTo("bronze sword
37                description"));
38        }
39    }
40 }
```

```
1 using SwinAdventure;
2
3 namespace TestLocation
4 {
5     public class Tests
6     {
7         Location location;
8         Item ruby;
9
10        [SetUp]
11        public void Setup()
12        {
13            location = new Location(new string[] { "location" }, "the
14                Location", "This is a test location");
15            ruby = new Item(new string[] { "gem", "ruby" }, "a red gem", "A
16                brilliant ruby, glows with a fiery red hue");
17            location.Inventory.Put(ruby);
18        }
19
20        [Test]
21        public void TestLocationIsIdentifiable()
22        {
23            GameObject? testLocationId = location.Locate("location");
24            Assert.That(testLocationId, Is.EqualTo(location));
25        }
26
27        [Test]
28        public void TestLocationLocatesItems()
29        {
30            GameObject? testLocationLocatesRuby = location.Locate("ruby");
31            Assert.That(testLocationLocatesRuby, Is.EqualTo(ruby));
32        }
33
34        [Test]
35        public void TestLocationLocatesItself()
36        {
37            GameObject? testLocationLocatesItself = location.Locate
38                ("location");
39            Assert.That(testLocationLocatesItself, Is.EqualTo(location));
40        }
41
42        [Test]
43        public void TestLocationlocatesNothing()
44        {
45            GameObject? testLocationLocatesItself = location.Locate
46                ("nothing");
47            Assert.That(testLocationLocatesItself, Is.EqualTo(null));
48        }
49    }
50 }
```

```
46     [Test]
47     public void TestLocationFullDescription()
48     {
49         string testLocationFullDescription = location.FullDescription;
50         Assert.That(testLocationFullDescription, Is.EqualTo("You are in ↵
        the Location\nThis is a test location\nIn this room you can ↵
        see:\n  a red gem (gem)"));
51     }
52 }
53 }
```



```
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        Location location;
13
14        [SetUp]
15        public void Setup()
16        {
17            look = new LookCommand();
18            testPlayer = new Player("testPlayer", "test player" ➤
19                description");
20            gem = new Item(new string[] { "gem" }, "a gem", "gem's" ➤
21                description");
22            bag = new Bag(new string[] { "bag" }, "a bag", "bag's" ➤
23                description");
24            location = new Location(new string[] { "location" }, "the" ➤
25                Location, "This is a test location");
26            testPlayer.Location = location;
27        }
28
29        [Test]
30        public void TestLookAtMe()
31        {
32            string testLookAtInventory = look.Execute(testPlayer, new ➤
33                string[] { "look", "at", "inventory" });
34            Assert.That(testLookAtInventory, Is.EqualTo("You are" ➤
35                testPlayer, test player description.\nYou are carrying: "));
36        }
37
38        [Test]
39        public void TestLookAtGem()
40        {
41            testPlayer.Inventory.Put(gem);
42            string testLookAtGem = look.Execute(testPlayer, new string[] ➤
43                { "look", "at", "gem" });
44            Assert.That(testLookAtGem, Is.EqualTo("gem's description"));
45        }
46
47        [Test]
48        public void TestLookAtUnknown()
49        {
50        }
```

```
43         string testLookAtUnknown = look.Execute(testPlayer, new string[]
    [] { "look", "at", "gem" });
44         Assert.That(testLookAtUnknown, Is.EqualTo("I cannot find the
    gem"));
45     }
46
47     [Test]
48     public void TestLookAtGemInMe()
49     {
50         testPlayer.Inventory.Put(gem);
51         string testLookAtGem = look.Execute(testPlayer, new string[]
    { "look", "at", "gem", "in", "inventory" });
52         Assert.That(testLookAtGem, Is.EqualTo("gem's description"));
53     }
54
55     [Test]
56     public void TestLookAtGemInBag()
57     {
58         bag.Inventory.Put(gem);
59         testPlayer.Inventory.Put(bag);
60         string testLookAtGemInBag = look.Execute(testPlayer, new
    string[] { "look", "at", "gem", "in", "bag" });
61         Assert.That(testLookAtGemInBag, Is.EqualTo("gem's
    description"));
62     }
63
64     [Test]
65     public void TestLookAtGemInNoBag()
66     {
67         string testLookAtGemInNoBag = look.Execute(testPlayer, new
    string[] { "look", "at", "gem", "in", "bag" });
68         Assert.That(testLookAtGemInNoBag, Is.EqualTo("I cannot find
    the bag"));
69     }
70
71     [Test]
72     public void TestLookAtNoGemInBag()
73     {
74         testPlayer.Inventory.Put(bag);
75         string testLookAtNoGemInBag = look.Execute(testPlayer, new
    string[] { "look", "at", "gem", "in", "bag" });
76         Assert.That(testLookAtNoGemInBag, Is.EqualTo("I cannot find
    the gem in the bag"));
77     }
78
79     [Test]
80     public void TestInvalidLook()
81     {
82         string testIncorrectTextLength = look.Execute(testPlayer, new
```

```
        string[] { "testing", "incorrect", "text", "length" });
83     string testLookNotFirstWord = look.Execute(testPlayer, new string[] { "testing", "look", "is", "not", "first" });
84     string testAtNotSecondWord = look.Execute(testPlayer, new string[] { "look", "test", "at", "not", "second" });
85     string testInNotFourthWord = look.Execute(testPlayer, new string[] { "look", "at", "in", "not", "fourth" });
86     Assert.That(testIncorrectTextLength, Is.EqualTo("I don't know how to look like that"));
87     Assert.That(testLookNotFirstWord, Is.EqualTo("Error in look input"));
88     Assert.That(testAtNotSecondWord, Is.EqualTo("What do you want to look at?"));
89     Assert.That(testInNotFourthWord, Is.EqualTo("What do you want to look in?"));
90 }
91
92 [Test]
93 public void TestLook()
94 {
95     location.Inventory.Put(gem);
96     string testLook = look.Execute(testPlayer, new string[] { "look" });
97     Assert.That(testLook, Is.EqualTo("You are in the Location\nThis is a test location\nIn this room you can see:\n a gem (gem)"));
98 }
99
100 [Test]
101 public void TestLookAtLocation()
102 {
103     location.Inventory.Put(gem);
104     string testLookAtLocation = look.Execute(testPlayer, new string[] { "look", "at", "location" });
105     Assert.That(testLookAtLocation, Is.EqualTo("You are in the Location\nThis is a test location\nIn this room you can see:\n a gem (gem)"));
106 }
107
108 [Test]
109 public void TestLookAtGemInLocation()
110 {
111     location.Inventory.Put(gem);
112     string testLookAtGem = look.Execute(testPlayer, new string[] { "look", "at", "gem" });
113     string testLookAtGemInLocation = look.Execute(testPlayer, new string[] { "look", "at", "gem", "in", "location" });
114     Assert.That(testLookAtGem, Is.EqualTo("gem's description"));
115     Assert.That(testLookAtGemInLocation, Is.EqualTo("gem's
```

```
        description"));
116     }
117
118     [Test]
119     public void TestLookAtNoGemInLocation()
120     {
121         string testLookAtGemInLocation = look.Execute(testPlayer, new
122             string[] { "look", "at", "gem", "in", "location" });
123         Assert.That(testLookAtGemInLocation, Is.EqualTo("I cannot find
124             the gem in the location"));
125     }
126
127     [Test]
128     public void TestLookAtGemInBagInLocation()
129     {
130         bag.Inventory.Put(gem);
131         location.Inventory.Put(bag);
132         string testLookAtGemInBag = look.Execute(testPlayer, new
133             string[] { "look", "at", "gem", "in", "bag" });
134         Assert.That(testLookAtGemInBag, Is.EqualTo("gem's
135             description"));
136     }
137 }
138 }
```