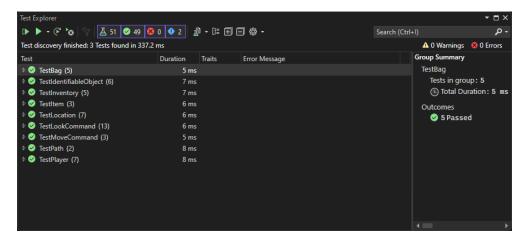
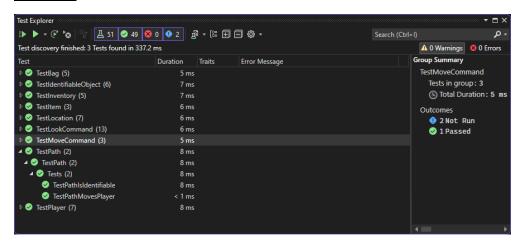
9.2C - Case Study - Iteration 7 - Paths

Jayden Kong, 104547242

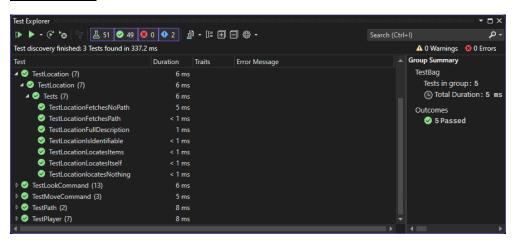
All tests passing:



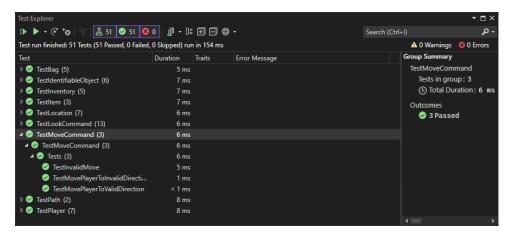
Path tests:



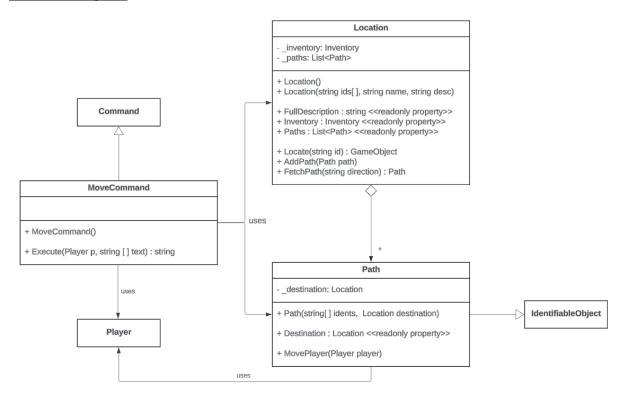
Location tests:



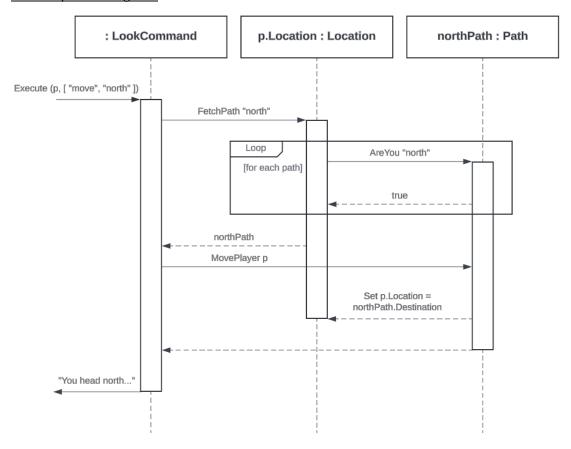
Move command tests:



UML Class Diagram:



UML Sequence Diagram:



Program running:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
7 namespace SwinAdventure
8 {
       public class Path : IdentifiableObject
9
10
       {
11
           private Location _destination;
12
13
           public Location Destination
14
15
                get
16
                {
17
                    return _destination;
18
                }
           }
19
20
21
           public Path(string[] idents, Location destination) : base
             (idents)
            {
22
23
                _destination = destination;
           }
24
25
           public void MovePlayer(Player player)
26
27
28
                player.Location = Destination;
29
           }
30
       }
31 }
32
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
7 namespace SwinAdventure
8 {
9
       public class MoveCommand : Command
10
       {
            public MoveCommand() : base(new string[] { "move", "go", "head", →
11
               "leave" }) { }
12
           public override string Execute(Player p, string[] text)
13
14
                if (text.Length != 2)
15
                {
16
17
                    return "I don't know how to move like that";
                }
18
19
                if (!AreYou(text[0]))
20
21
22
                    return "Error in move input";
23
                }
24
25
                Path? path;
26
                string direction;
27
                switch (text[1])
28
                {
29
                    case string north when north == "north" || north == "n":
30
                        direction = "north";
31
                        break;
32
                    case string northEast when northEast == "north_east" || >
                      northEast == "ne":
33
                        direction = "north_east";
34
                        break;
                    case string northWest when northWest == "north_west" || >
35
                      northWest == "nw":
36
                        direction = "north_west";
37
                        break;
38
                    case string south when south == "south" || south == "s":
39
                        direction = "south";
40
                        break;
                    case string southEast when southEast == "south_east" | | →
41
                      southEast == "se":
42
                        direction = "south_east";
43
                        break;
44
                    case string southWest when southWest == "south_west" || >
                      southWest == "sw":
45
                        direction = "south_west";
46
                    case string east when east == "east" || east == "e":
47
48
                        direction = "east";
```

50 51

52

53 54

55

56 57

58

59

60

61

62 63

64 65

66

67 68 69

70

71

{

}

}

}

```
break;
    case string west when west == "west" || west == "w":
        direction = "west";
        break;
    case "up":
        direction = text[1];
        break:
    case "down":
        direction = text[1];
        break;
    default:
        return "I don't know that direction";
path = p.Location.FetchPath(direction);
if (path == null)
    return string.Format("I cannot move {0}", text[1]);
path.MovePlayer(p);
```

return string.Format("You head {0}\nYou have arrived in

{1}", path.FirstID, path.Destination.Name);

```
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
            private List<Path> _paths;
14
15
            public override string FullDescription
16
17
                get
18
                {
                    string description = string.Format("You are in {0}\n
19
                      {1}", Name, base.FullDescription);
                    if (Paths.Count > 0)
20
21
                        description += "\nThere are exits to the ";
22
                        int lastIndex = Paths.Count - 1;
23
24
                        int i = 0;
                        foreach (Path path in Paths)
25
26
27
                            if (Paths.Count > 1)
28
29
                                 if (i < lastIndex)</pre>
30
                                     description += string.Format("{0}, ", →
31
                      path.FirstID);
32
                                 }
33
                                 else
34
                                     description += string.Format("and
35
                      {0}.", path.FirstID);
36
                                 }
37
                                 i++;
38
                            }
39
                            else
40
                                 description += string.Format("{0}.",
41
                      path.FirstID);
42
                            }
                        }
43
44
                    description += string.Format("\nIn this room you can
45
                      see:{0}", Inventory.ItemList);
46
                    return description;
47
                }
            }
48
```

```
...sity\Year 2\COS20007\9.2C\SwinAdventure\Location.cs
49
             public Inventory Inventory
50
51
                 get
                 {
52
53
                     return _inventory;
54
                 }
55
            }
56
             public List<Path> Paths
57
58
59
                 get
                 {
60
61
                     return _paths;
                 }
62
            }
63
64
65
             public Location() : this(new string[] { "location",
               "unknown" }, "an unknown location", "This is a mysterious
               location") { }
                                        // Default constructor, to make sure >
                the player has a location if not allocated one
             public Location(string[] ids, string name, string desc) : base →
66
               (ids, name, desc)
67
                 AddIdentifier("room");
68
                 AddIdentifier("here");
69
                 AddIdentifier("location");
70
                 _inventory = new Inventory();
71
                 _paths = new List<Path>();
72
            }
73
74
             public GameObject? Locate(string id)
75
76
77
                 if (AreYou(id))
                 {
78
79
                     return this;
80
81
                 return Inventory.Fetch(id);
            }
82
83
84
            public void AddPath(Path path)
             {
85
                 _paths.Add(path);
86
87
            }
88
89
             public Path? FetchPath(string direction)
90
                 foreach (Path path in Paths)
91
92
                 {
                     if (path.AreYou(direction))
93
94
                     {
95
                         return path;
```

97

}

```
98 return null;
99 }
100 }
101 }
```

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

```
...rsity\Year 2\COS20007\9.2C\SwinAdventure\Program.cs
                Item computer = new Item(new string[] { "pc", "computer" },
47
                   "a small computer", "A dusty PC with a flickering
                  screen");
                Item laptop = new Item(new string[] { "laptop" }, "a
48
                  laptop", "A compact, modern laptop with a matte black
                  finish");
49
                Item bulletinBoard = new Item(new string[]
                  { "bulletin_board" }, "a bulletin board", "A small
                  bulletin board filled with announcements, flyers, and
                  posters");
                Item server = new Item(new string[] { "server" }, "a
50
                  server", "A sleek server that hums with activity, the
                  heart of Swin Adventure");
51
52
                // Create Bags
                Bag laptopBag = new Bag(new string[] { "laptop_bag",
53
                  "bag" }, "laptop bag", "A sleek, black laptop bag. Its
                  fabric is slightly worn from use");
                Bag bag = new Bag(new string[] { "bag" }, "leather bag", "A >
54
                   small bag crafted from supple brown leather, perfect for >
                  carrying items");
55
                // Create Locations
56
                Location classroom = new Location(new string[]
57
                  { "classroom" }, "the Classroom", "This is a dimly lit
                                       // Player will initially be in the
                  classroom.");
                  classroom
                Location hallway = new Location(new string[] { "hallway" },
58
                   "the Hallway", "This is a long well lit hallway, many
                  Swin Adventurers are roaming around.");
59
                Location serverRoom = new Location(new string[]
                                                                              P
                  { "server_room" }, "the Server Room", "This is a dark
                                                                              P
                  server room. Rows of humming servers stand within sleek
                  cabinets, bathed in dim light.");
60
                // Create Paths
61
                Path classroomToHallway = new Path(new string[] { "east",
62
                  "hallway" }, hallway);
                Path hallwayToClassroom = new Path(new string[] { "west",
63
                  "classroom" }, classroom);
64
                Path classroomToServerRoom = new Path(new string[]
65
                  { "west", "server_room" }, serverRoom);
                Path serverRoomToClassroom = new Path(new string[]
66
                  { "east", "classroom" }, classroom);
67
                // Add Paths to Locations
68
69
                classroom.AddPath(classroomToHallway);
70
                classroom.AddPath(classroomToServerRoom);
71
                hallway.AddPath(hallwayToClassroom);
                serverRoom.AddPath(serverRoomToClassroom);
72
73
74
                // Set Player Location
```

```
... rsity \verb|\Year 2\COS20007\9.2C\SwinAdventure\Program.cs|
```

```
3
```

```
75
                player.Location = classroom;
76
77
                // Distribute Items to Player
                bag.Inventory.Put(ruby);
78
 79
                player.Inventory.Put(shovel);
                player.Inventory.Put(bronzeSword);
 80
 81
                player.Inventory.Put(bag);
 82
 83
                // Distribute items to Classroom
 84
                laptopBag.Inventory.Put(laptop);
                classroom.Inventory.Put(computer);
 85
                classroom.Inventory.Put(laptopBag);
 86
 87
                // Distribute items to Hallway
 88
 89
                hallway.Inventory.Put(bulletinBoard);
 90
 91
                // Distribute items to Server Room
 92
                serverRoom.Inventory.Put(server);
 93
 94
                // Introduction text
 95
                Console.WriteLine("----");
 96
                Console.WriteLine("Welcome to Swin Adventure!");
 97
                Console.WriteLine("You have arrived in {0}",
 98
                                                                              P
                   player.Location.Name);
99
                // Game loop
100
101
                bool gameLoop = true;
                while (gameLoop)
102
103
                     Console.Write("Command -> ");
104
                     string? playerInput = Console.ReadLine();
105
106
                     string[] inputToPass = playerInput.Split(new char[]
                       { }, StringSplitOptions.RemoveEmptyEntries);
107
                     Console.WriteLine("");
                     foreach (string input in inputToPass)
108
109
                         if (lookCommand.AreYou(input))
110
111
112
                             Console.WriteLine(lookCommand.Execute(player,
                       inputToPass));
113
114
                         else if (moveCommand.AreYou(input))
115
                             Console.WriteLine(moveCommand.Execute(player,
116
                       inputToPass));
117
                         }
118
                     }
                }
119
            }
120
        }
121
122 }
123
```

```
1 using SwinAdventure;
2
3 namespace TestPath
4 {
       public class Tests
5
6
7
            private Location hallway;
8
           private SwinAdventure.Path testPathToHallway;
9
           private Player p;
10
11
            [SetUp]
            public void Setup()
12
13
               hallway = new Location(new string[] { "hallway" }, "the
14
                  Hallway", "This is a long well lit hallway, many Swin
                  Adventurers are roaming around.");
               testPathToHallway = new SwinAdventure.Path(new string[]
15
                                                                               P
                  { "east", "hallway" }, hallway);
               p = new Player("Tester", "the mighty test player");
16
           }
17
18
19
            [Test]
20
            public void TestPathIsIdentifiable()
21
               bool testPathIsIdentifiable = testPathToHallway.AreYou
22
                  ("east");
23
               Assert.That(testPathIsIdentifiable, Is.EqualTo(true));
24
            }
25
26
            [Test]
            public void TestPathMovesPlayer()
27
28
29
               testPathToHallway.MovePlayer(p);
30
               Assert.That(p.Location, Is.EqualTo(hallway));
31
           }
32
       }
33 }
```

```
...y\Year 2\COS20007\9.2C\TestMoveCommand\UnitTest1.cs
```

```
1
```

```
1 using SwinAdventure;
2
 3 namespace TestMoveCommand
4 {
5
       public class Tests
6
7
            private MoveCommand move;
            private Player testPlayer;
8
9
            private Location location;
10
            private Location hallway;
11
            private SwinAdventure.Path testPathToHallway;
12
13
14
            [SetUp]
15
            public void Setup()
16
17
                move = new MoveCommand();
18
                testPlayer = new Player("testPlayer", "test player
                  description");
19
                location = new Location(new string[] { "location" }, "the
                  Location", "This is a test location");
                hallway = new Location(new string[] { "hallway" }, "the
20
                  Hallway", "This is a long well lit hallway, many Swin
                  Adventurers are roaming around.");
21
                testPathToHallway = new SwinAdventure.Path(new string[]
                  { "east", "hallway" }, hallway);
22
                testPlayer.Location = location;
23
                location.AddPath(testPathToHallway);
           }
24
25
26
            [Test]
            public void TestMovePlayerToValidDirection()
27
28
29
                string testMoveToDirection = move.Execute(testPlayer, new
                  string[] { "move", "east" });
                Assert.That(testMoveToDirection, Is.EqualTo("You head east
30
                  \nYou have arrived in the Hallway"));
31
                Assert.That(testPlayer.Location, Is.EqualTo(hallway));
            }
32
33
34
            [Test]
35
            public void TestMovePlayerToInvalidDirection()
36
                string testMoveToDirection = move.Execute(testPlayer, new
37
                  string[] { "move", "west" });
                Assert.That(testMoveToDirection, Is.EqualTo("I cannot move
38
                  west"));
39
                Assert.That(testPlayer.Location, Is.EqualTo(location));
40
            }
41
42
            [Test]
43
            public void TestInvalidMove()
44
```

```
... y \verb|\Year 2\COS20007\9.2C\TestMoveCommand\UnitTest1.cs|
                string testTextLengthNot2 = move.Execute(testPlayer, new
                  string[] { "move", "test", "length", "not", "two" });
                string testMoveIsNotFirstWord = move.Execute(testPlayer, new >
46
                   string[] { "test", "move" });
                string testMoveToUnknownDirection = move.Execute(testPlayer, >
47
                   new string[] { "move", "unknown" });
48
49
                Assert.That(testTextLengthNot2, Is.EqualTo("I don't know how →
                   to move like that"));
                Assert.That(testMoveIsNotFirstWord, Is.EqualTo("Error in
50
                  move input"));
                Assert.That(testMoveToUnknownDirection, Is.EqualTo("I don't >
51
                  know that direction"));
                Assert.That(testPlayer.Location, Is.EqualTo
52
                                                                    // Player >
                  (location));
                  stays in current location
53
            }
54
       }
```

```
1 using SwinAdventure;
2
 3 namespace TestLocation
4 {
5
       public class Tests
6
7
            private Location location;
8
            private Item ruby;
9
            private SwinAdventure.Path testPathToHallway;
10
            private Location hallway;
11
12
13
            [SetUp]
14
            public void Setup()
15
                location = new Location(new string[] { "location" }, "the
16
                  Location", "This is a test location");
17
                hallway = new Location(new string[] { "hallway" }, "the
                  Hallway", "This is a long well lit hallway, many Swin
                  Adventurers are roaming around.");
                testPathToHallway = new SwinAdventure.Path(new string[]
18
                  { "east", "hallway" }, hallway);
                ruby = new Item(new string[] { "gem", "ruby" }, "a red gem", >
19
                   "A brilliant ruby, glows with a fiery red hue");
20
                location.Inventory.Put(ruby);
            }
21
22
23
            [Test]
            public void TestLocationIsIdentifiable()
24
25
26
                GameObject? testLocationId = location.Locate("location");
27
                Assert.That(testLocationId, Is.EqualTo(location));
28
            }
29
30
            [Test]
            public void TestLocationLocatesItems()
31
32
33
                GameObject? testLocationLocatesRuby = location.Locate
                  ("ruby");
34
                Assert.That(testLocationLocatesRuby, Is.EqualTo(ruby));
            }
35
36
37
            [Test]
            public void TestLocationLocatesItself()
39
                GameObject? testLocationLocatesItself = location.Locate
40
                  ("location");
41
                Assert.That(testLocationLocatesItself, Is.EqualTo
                  (location));
            }
42
43
44
            [Test]
45
            public void TestLocationlocatesNothing()
```

```
...sity\Year 2\COS20007\9.2C\TestLocation\UnitTest1.cs
                                                                               2
46
47
                GameObject? testLocationLocatesItself = location.Locate
                  ("nothing");
48
                Assert.That(testLocationLocatesItself, Is.EqualTo(null));
            }
49
50
            [Test]
51
52
            public void TestLocationFullDescription()
53
54
                location.AddPath(testPathToHallway);
55
                string testLocationFullDescription =
                  location.FullDescription;
56
                Assert.That(testLocationFullDescription, Is.EqualTo("You are →
                   in the Location\nThis is a test location\nThere are exits >
                  to the east.\nIn this room you can see:\n a red gem
                  (gem)"));
            }
57
58
59
            [Test]
60
            public void TestLocationFetchesPath()
61
62
                location.AddPath(testPathToHallway);
                SwinAdventure.Path? testFetchPath = location.FetchPath
63
                  ("east");
64
                Assert.That(testFetchPath, Is.EqualTo(testPathToHallway));
            }
65
66
67
            [Test]
            public void TestLocationFetchesNoPath()
68
69
                SwinAdventure.Path? testFetchPath = location.FetchPath
70
                  ("east");
71
                Assert.That(testFetchPath, Is.EqualTo(null));
72
            }
       }
73
```

```
...ersity\Year 2\COS20007\9.2C\SwinAdventure\Player.cs
```

```
1
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
7 namespace SwinAdventure
8
9
       public class Player : GameObject, IHaveInventory
10
            private Inventory _inventory;
11
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", "inv" }, name, desc)
43
44
                _inventory = new Inventory();
45
                _location = new Location();
            }
46
47
48
            public GameObject? Locate(string id)
49
                if (AreYou(id))
50
```

```
...ersity\Year 2\COS20007\9.2C\SwinAdventure\Player.cs
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 }
```

```
...iversity\Year 2\COS20007\9.2C\SwinAdventure\Item.cs
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
 5 using System.Threading.Tasks;
7 namespace SwinAdventure
8 {
9
       public class Item : GameObject
       {
10
            public Item(string[] idents, string name, string desc) : base
11
              (idents, name, desc) { }
12
       }
13 }
14
```

```
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
            public LookCommand() : base(new string[] { "look" }) { }
12
13
            public override string Execute(Player p, string[] text)
14
15
                if (!(text.Length == 1 || text.Length == 3 || text.Length == >
16
                   5))
17
                {
                    return "I don't know how to look like that";
18
19
                }
20
                if (text[0] != "look")
21
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
                if (text[1] != "at")
32
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
48
49
                    return itemDescription3;
50
                }
51
```

```
...y\Year 2\COS20007\9.2C\SwinAdventure\LookCommand.cs

// By this point the 1 and 3 element lo
```

```
2
```

```
// 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}", →
                       text[2], text[4]);
63
64
                return itemDescription5;
            }
65
66
67
            private IHaveInventory? FetchContainer(Player p, string
              containerId)
68
69
                IHaveInventory? container = p.Locate(containerId) as
                  IHaveInventory;
70
                return container;
71
            }
72
            private string? LookAtIn(string thingId, IHaveInventory
73
              container)
74
75
                GameObject? item = container.Locate(thingId);
                if (item == null)
76
77
78
                    return null;
79
80
                return item.FullDescription;
            }
81
       }
82
83 }
84
```

```
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
       public abstract class GameObject : IdentifiableObject
10
11
            private string _description;
12
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
            public GameObject(string[] ids, string name, string desc) : base >
39
              (ids)
40
41
                _name = name;
42
                _description = desc;
43
            }
44
        }
45 }
46
```

```
...ear 2\COS20007\9.2C\SwinAdventure\IHaveInventory.cs
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
7 namespace SwinAdventure
8 {
9
       public interface IHaveInventory
10
       {
            public string Name { get; }
11
12
            public GameObject? Locate(string id);
13
14
       }
15 }
16
```

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

```
1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5 using System.Threading.Tasks;
 7 namespace SwinAdventure
 8
 9
        public class Inventory
10
        {
            private List<Item> _items;
11
12
13
            public string ItemList
14
15
                get
16
                     string itemList = "";
17
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
                     if (item.AreYou(id))
46
47
48
                         _items.Remove(item);
49
                         return item;
50
                    }
51
                }
52
                return null;
```

```
...ity\Year 2\COS20007\9.2C\SwinAdventure\Inventory.cs
53 }
54
            public Item? Fetch(string id)
55
56
57
                foreach (Item item in _items)
58
                {
59
                     if (item.AreYou(id))
60
61
                         return item;
62
                     }
63
                }
64
                return null;
            }
65
66
```

```
...rsity\Year 2\COS20007\9.2C\SwinAdventure\Command.cs
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
 5 using System.Threading.Tasks;
7 namespace SwinAdventure
8 {
9
       public abstract class Command : IdentifiableObject
10
       {
            public Command(string[] ids) : base(ids) { }
11
12
            public abstract string Execute(Player p, string[] text);
13
       }
14
15 }
16
```

```
1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5 using System.Threading.Tasks;
 7 namespace SwinAdventure
 8 {
        public class Bag : Item, IHaveInventory
 9
10
        {
11
            private Inventory _inventory;
12
13
            public override string FullDescription
14
15
                get
16
                    return string.Format("In the {0} you can see: {1}",
17
                      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]
            public void TestBagLocatesItems()
8
9
10
                Bag testBag = new Bag(new string[] { "bag", "testingBag"},
                  "test bag", "this is the test bag's description");
                Item shovel = new Item(new string[] { "shovel", "spade" },
11
                  "a shovel", "shovel description");
                testBag.Inventory.Put(shovel);
12
13
                GameObject? testLocateShovel = testBag.Locate("shovel");
14
15
                GameObject? testShovelRemainsInBag = testBag.Locate
                  ("shovel");
                Assert.That(testLocateShovel, Is.EqualTo(shovel));
16
17
                Assert.That(testShovelRemainsInBag, Is.EqualTo(shovel));
            }
18
19
20
            [Test]
            public void TestBagLocatesItself()
21
22
            {
                Bag testBag = new Bag(new string[] { "bag", "testingBag" }, >
23
                  "test bag", "this is the test bag's description");
24
                GameObject? testLocateBagID1 = testBag.Locate("bag");
25
26
                GameObject? testLocateBagID2 = testBag.Locate("testingBag");
               Assert.That(testLocateBagID1, Is.EqualTo(testBag));
27
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
                GameObject? testLocateShovel = testBag.Locate("shovel");
36
               Assert.That(testLocateShovel, Is.EqualTo(null));
37
38
            }
39
40
            [Test]
            public void TestBagFullDescription()
41
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");
                Item bronzeSword = new Item(new string[] { "sword", "bronze >
45
                  sword" }, "a bronze sword", "bronze sword description");
```

```
...niversity\Year 2\COS20007\9.2C\TestBag\UnitTest1.cs
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");
                Bag b2 = new Bag(new string[] { "bag", "testingBag2" },
57
                  "test bag 2", "this is test bag 2's description");
                Item shovel = new Item(new string[] { "shovel", "spade" },
58
                  "a shovel", "shovel description");
                Item bronzeSword = new Item(new string[] { "sword", "bronze
59
                  sword" }, "a bronze sword", "bronze sword description");
60
               b1.Inventory.Put(shovel);
                b2.Inventory.Put(bronzeSword);
61
                b1.Inventory.Put(b2);
62
63
                GameObject? testB1LocatesB2 = b1.Locate("testingBag2");
64
                GameObject? testB1LocatesShovel = b1.Locate("shovel");
65
                GameObject? testB1LocatesBronzeSword = b1.Locate("sword");
66
67
                Assert.That(testB1LocatesB2, Is.EqualTo(b2));
```

Assert.That(testB1LocatesShovel, Is.EqualTo(shovel));

Assert.That(testB1LocatesBronzeSword, Is.EqualTo(null));

68 69

70

71

72 }

}

```
...ity\Year 2\COS20007\9.2C\TestInventory\UnitTest1.cs
```

```
1
```

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

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

...ity\Year 2\COS20007\9.2C\TestInventory\UnitTest1.cs

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

```
...2\COS20007\9.2C\TestIdentifiableObject\UnitTest1.cs
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;
                Assert.That(firstID, Is.EqualTo(""));
60
            }
61
62
            [Test]
63
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
       }
```

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

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

```
...ersity\Year 2\COS20007\9.2C\TestPlayer\UnitTest1.cs
                                                                                2
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
                GameObject? testLocateNothing = p.Locate("nothing");
60
                Assert.That(testLocateNothing, Is.EqualTo(null));
61
62
            }
63
            [Test]
64
            public void TestPlayerFullDescription()
65
66
67
                p.Inventory.Put(shovel);
                p.Inventory.Put(bronzeSword);
68
69
70
                string testFullDescription = p.FullDescription;
                Assert.That(testFullDescription, Is.EqualTo("You are Tester, →
71
                   the mighty test player.\nYou are carrying: \n a shovel
                  (shovel)\n a bronze sword (sword)"));
            }
72
73
74
            [Test]
75
            public void TestPlayerLocatesItemsInLocation()
76
77
                location.Inventory.Put(ruby);
78
                p.Location = location;
79
80
                GameObject? testPlayerLocatesRubyInLocation = p.Locate
                  ("ruby");
81
                Assert.That(testPlayerLocatesRubyInLocation, Is.EqualTo
                  (ruby));
            }
82
83
84
            [Test]
85
            public void TestPlayerLocatesNoItemsInLocation()
86
87
                p.Location = location;
88
89
                GameObject? testPlayerLocatesNothingInLocation = p.Locate
                  ("nothing");
90
                Assert.That(testPlayerLocatesNothingInLocation, Is.EqualTo
                  (null));
91
            }
       }
92
93 }
```

```
using SwinAdventure;
 3
 4 namespace TestLookCommand
 5
 6
       public class Tests
 7
            private LookCommand look;
 8
 9
            private Player testPlayer;
10
            private Item gem;
11
            private Bag bag;
            private Location location;
12
13
            [SetUp]
14
15
            public void Setup()
16
17
                look = new LookCommand();
18
                testPlayer = new Player("testPlayer", "test player
                  description");
                gem = new Item(new string[] { "gem" }, "a gem", "gem's
19
                  description");
                bag = new Bag(new string[] { "bag" }, "a bag", "bag's
20
                  description");
                location = new Location(new string[] { "location" }, "the
21
                  Location", "This is a test location");
                testPlayer.Location = location;
22
            }
23
24
            [Test]
25
            public void TestLookAtMe()
26
27
                string testLookAtInventory = look.Execute(testPlayer, new
28
                  string[] { "look", "at", "inventory" });
29
                Assert.That(testLookAtInventory, Is.EqualTo("You are
                  testPlayer, test player description.\nYou are carrying:
                  "));
            }
30
31
32
            [Test]
33
            public void TestLookAtGem()
34
35
                testPlayer.Inventory.Put(gem);
36
                string testLookAtGem = look.Execute(testPlayer, new string >
                  [] { "look", "at", "gem" });
                Assert.That(testLookAtGem, Is.EqualTo("gem's
37
                  description"));
            }
38
39
40
            [Test]
41
            public void TestLookAtUnknown()
42
43
                string testLookAtUnknown = look.Execute(testPlayer, new
                  string[] { "look", "at", "gem" });
```

```
... y \verb|\Year 2\COS20007\9.2C\TestLookCommand\UnitTest1.cs|
```

```
2
```

```
44
                Assert.That(testLookAtUnknown, Is.EqualTo("I cannot find
                  the gem"));
            }
45
46
47
            [Test]
            public void TestLookAtGemInMe()
48
49
                testPlayer.Inventory.Put(gem);
50
                string testLookAtGem = look.Execute(testPlayer, new string >>
51
                  [] { "look", "at", "gem", "in", "inventory" });
                Assert.That(testLookAtGem, Is.EqualTo("gem's
52
                  description"));
53
            }
54
55
            [Test]
            public void TestLookAtGemInBag()
56
57
58
                bag.Inventory.Put(gem);
                testPlayer.Inventory.Put(bag);
59
60
                string testLookAtGemInBag = look.Execute(testPlayer, new
                  string[] { "look", "at", "gem", "in", "bag" });
                Assert.That(testLookAtGemInBag, Is.EqualTo("gem's
61
                  description"));
            }
62
63
64
            [Test]
            public void TestLookAtGemInNoBag()
65
66
67
                string testLookAtGemInNoBag = look.Execute(testPlayer, new >>
                  string[] { "look", "at", "gem", "in", "bag" });
                Assert.That(testLookAtGemInNoBag, Is.EqualTo("I cannot find >
68
                   the bag"));
69
            }
70
71
            [Test]
            public void TestLookAtNoGemInBag()
72
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]
            public void TestInvalidLook()
80
81
82
                string testIncorrectTextLength = look.Execute(testPlayer,
                  new string[] { "testing", "incorrect", "text",
                  "length" });
                string testLookNotFirstWord = look.Execute(testPlayer, new
83
                  string[] { "testing", "look", "is", "not", "first" });
84
                string testAtNotSecondWord = look.Execute(testPlayer, new
```

```
... y \verb|\Year 2\COS20007\9.2C\TestLookCommand\UnitTest1.cs|
                   string[] { "look", "test", "at", "not", "second" });
85
                 string testInNotFourthWord = look.Execute(testPlayer, new
                                                                               P
                   string[] { "look", "at", "in", "not", "fourth" });
                 Assert.That(testIncorrectTextLength, Is.EqualTo("I don't
86
                   know how to look like that"));
                 Assert.That(testLookNotFirstWord, Is.EqualTo("Error in look →
87
                    input"));
                 Assert.That(testAtNotSecondWord, Is.EqualTo("What do you
 88
                   want to look at?"));
                 Assert.That(testInNotFourthWord, Is.EqualTo("What do you
 89
                   want to look in?"));
             }
 90
 91
             [Test]
 92
 93
             public void TestLook()
 94
 95
                 location.Inventory.Put(gem);
 96
                 string testLook = look.Execute(testPlayer, new string[]
                   { "look" });
                 Assert.That(testLook, Is.EqualTo("You are in the Location
 97
                   \nThis is a test location\nIn this room you can see:\n a >
                   gem (gem)"));
             }
 98
99
100
             [Test]
             public void TestLookAtLocation()
101
102
                 location.Inventory.Put(gem);
103
104
                 string testLookAtLocation = look.Execute(testPlayer, new
                   string[] { "look", "at", "location" });
                 Assert.That(testLookAtLocation, Is.EqualTo("You are in the
105
                   Location\nThis is a test location\nIn this room you can
                   see:\n a gem (gem)"));
             }
106
107
             [Test]
108
109
             public void TestLookAtGemInLocation()
110
                 location.Inventory.Put(gem);
111
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"));
                 Assert.That(testLookAtGemInLocation, Is.EqualTo("gem's
115
                   description"));
116
             }
117
118
             [Test]
119
             public void TestLookAtNoGemInLocation()
120
121
                 string testLookAtGemInLocation = look.Execute(testPlayer,
```

```
\underline{\dots} y \\ Year 2 \\ COS20007 \\ 9.2C \\ TestLook Command \\ UnitTest1.cs
                   new string[] { "look", "at", "gem", "in", "location" });
122
                 Assert.That(testLookAtGemInLocation, Is.EqualTo("I cannot
                    find the gem in the location"));
             }
123
124
125
             [Test]
             public void TestLookAtGemInBagInLocation()
126
127
                 bag.Inventory.Put(gem);
128
                 location.Inventory.Put(bag);
129
130
                 string testLookAtGemInBag = look.Execute(testPlayer, new
                   string[] { "look", "at", "gem", "in", "bag" });
131
                 Assert.That(testLookAtGemInBag, Is.EqualTo("gem's
                   description"));
132
             }
```

134 }