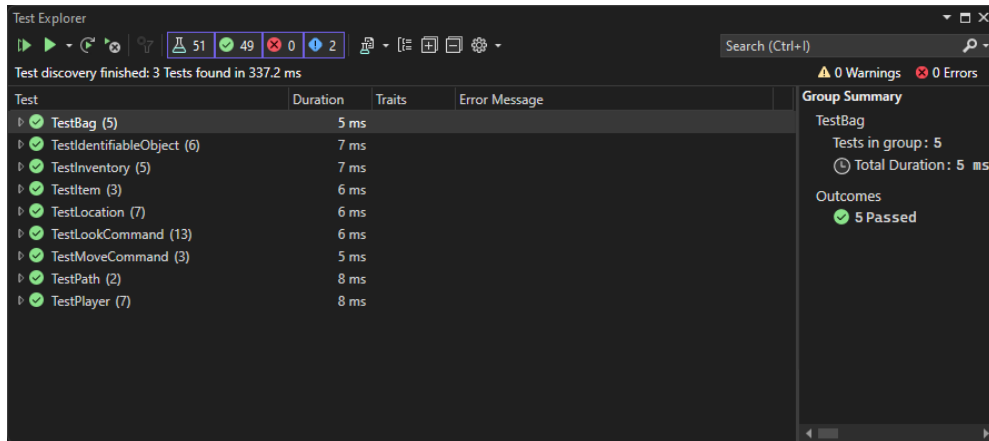


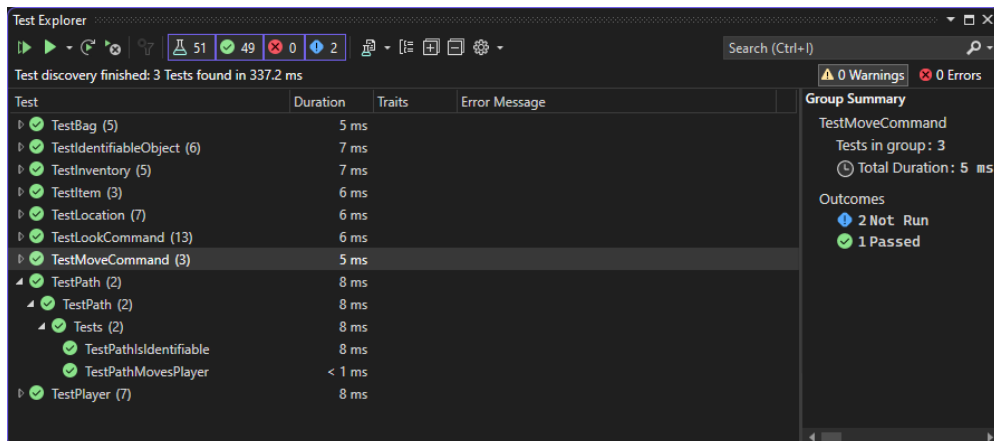
9.2C - Case Study - Iteration 7 – Paths

Jayden Kong, 104547242

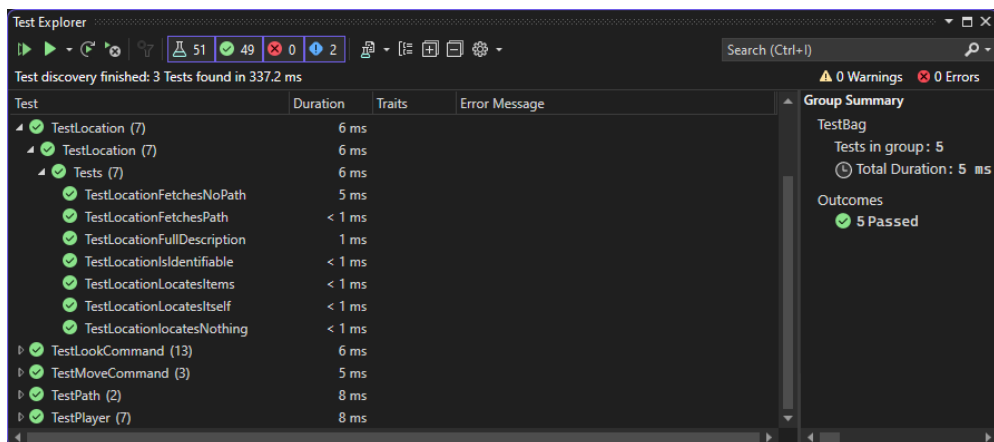
All tests passing:



Path tests:



Location tests:

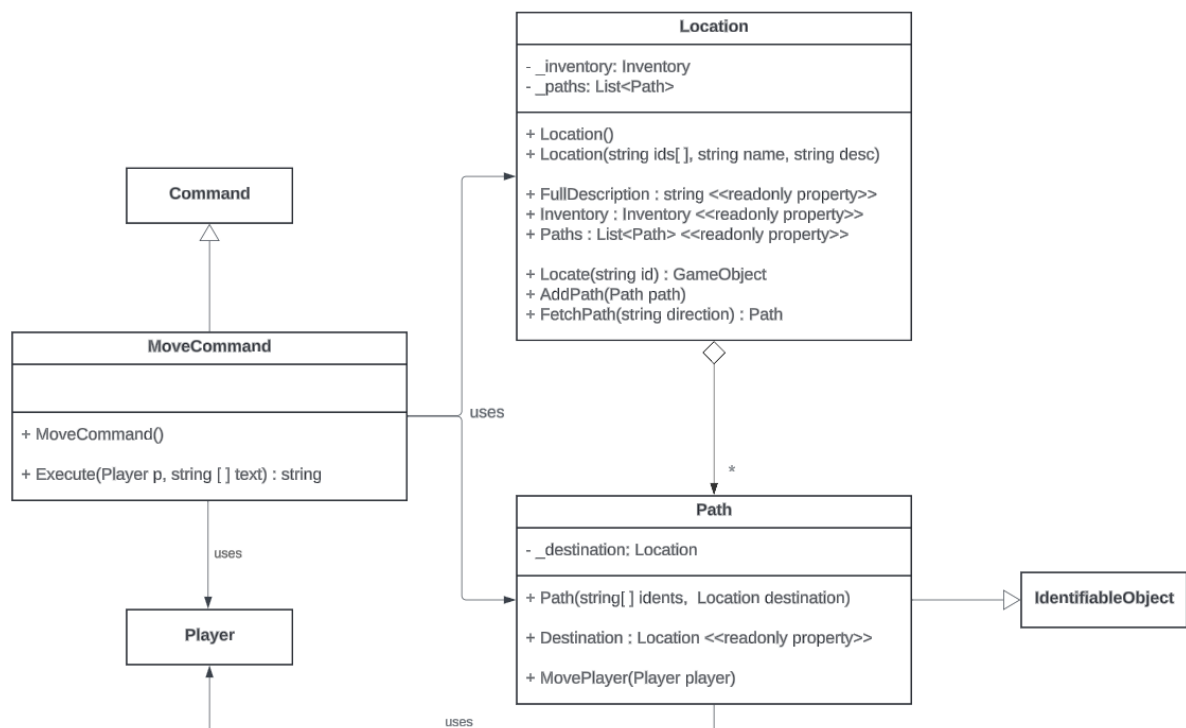


Move command tests:

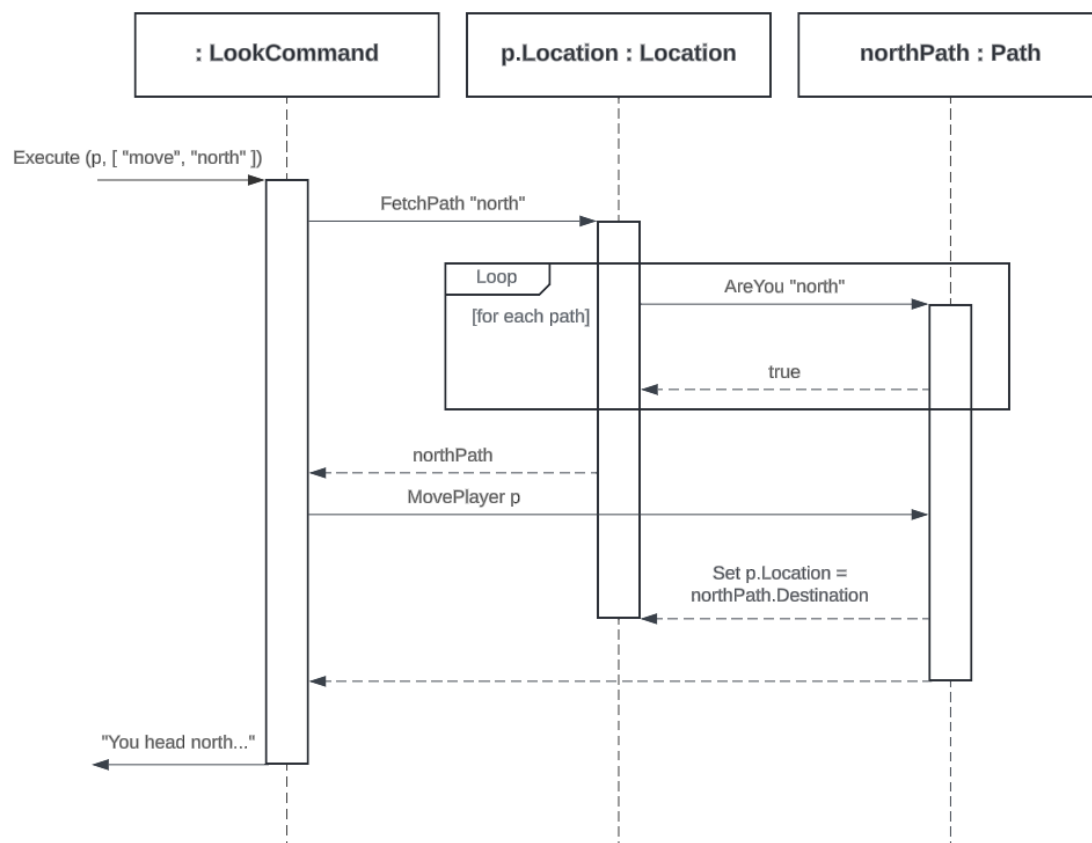
Test Explorer			
Test run finished: 51 Tests (51 Passed, 0 Failed, 0 Skipped) run in 154 ms			
Test	Duration	Traits	Error Message
TestBag (5)	5 ms		
TestIdentifiableObject (6)	7 ms		
TestInventory (5)	7 ms		
TestItem (3)	7 ms		
TestLocation (7)	6 ms		
TestLookCommand (13)	6 ms		
TestMoveCommand (3)	6 ms		
TestMoveCommand (3)	6 ms		
Tests (3)	6 ms		
TestInvalidMove	5 ms		
TestMovePlayerToInvalidDirecti...	1 ms		
TestMovePlayerToValidDirection	< 1 ms		
TestPath (2)	8 ms		
TestPlayer (7)	8 ms		

Group Summary
TestMoveCommand
Tests in group: 3
Total Duration: 6 ms
Outcomes
3 Passed

UML Class Diagram:



UML Sequence Diagram:



Program running:

```
C:\Users\Jayden Kong\OneDr...
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.
There are exits to the east, and west.
In this room you can see:
    a small computer (pc)
    laptop bag (laptop_bag)
Command -> head east

You head east
You have arrived in the Hallway
Command -> look

You are in the Hallway
This is a long well lit hallway, many Swin Adventurers are roaming around.
There are exits to the west.
In this room you can see:
    a bulletin board (bulletin_board)
Command -> head west

You head west
You have arrived in the Classroom
Command -> head west

You head west
You have arrived in the Server Room
Command -> look

You are in the Server Room
This is a dark server room. Rows of humming servers stand within sleek cabinets, bathed in dim light.
There are exits to the east.
In this room you can see:
    a server (server)
Command -> look at server

A sleek server that hums with activity, the heart of Swin Adventure
Command -> |
```

```
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 Path : IdentifiableObject
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
22            (idents)
23        {
24            _destination = destination;
25        }
26
27        public void MovePlayer(Player player)
28        {
29            player.Location = Destination;
30        }
31    }
32
```

```
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 MoveCommand : Command
10    {
11        public MoveCommand() : base(new string[] { "move", "go", "head", ↵
            "leave" }) { }
12
13        public override string Execute(Player p, string[] text)
14        {
15            if (text.Length != 2)
16            {
17                return "I don't know how to move like that";
18            }
19
20            if (!AreYou(text[0]))
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;
35                case string northWest when northWest == "north_west" || ↵
                    northWest == "nw":
36                    direction = "north_west";
37                    break;
38                case string south when south == "south" || south == "s":
39                    direction = "south";
40                    break;
41                case string southEast when southEast == "south_east" || ↵
                    southEast == "se":
42                    direction = "south_east";
43                    break;
44                case string southWest when southWest == "south_west" || ↵
                    southWest == "sw":
45                    direction = "south_west";
46                    break;
47                case string east when east == "east" || east == "e":
48                    direction = "east";
```

```
49         break;
50         case string west when west == "west" || west == "w":
51             direction = "west";
52             break;
53         case "up":
54             direction = text[1];
55             break;
56         case "down":
57             direction = text[1];
58             break;
59         default:
60             return "I don't know that direction";
61     }
62     path = p.Location.FetchPath(direction);
63
64     if (path == null)
65     {
66         return string.Format("I cannot move {0}", text[1]);
67     }
68
69     path.MovePlayer(p);
70     return string.Format("You head {0}\nYou have arrived in {1}", path.FirstID, path.Destination.Name);
71 }
72 }
73 }
74
```

```
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             {
19                 string description = string.Format("You are in {0}\n", Name, base.FullDescription);
20                 if (Paths.Count > 0)
21                 {
22                     description += "\nThere are exits to the ";
23                     int lastIndex = Paths.Count - 1;
24                     int i = 0;
25                     foreach (Path path in Paths)
26                     {
27                         if (Paths.Count > 1)
28                         {
29                             if (i < lastIndex)
30                             {
31                                 description += string.Format("{0}, ", path.FirstID);
32                             }
33                             else
34                             {
35                                 description += string.Format("and {0}.", path.FirstID);
36                             }
37                             i++;
38                         }
39                         else
40                         {
41                             description += string.Format("{0}.", path.FirstID);
42                         }
43                     }
44                 }
45                 description += string.Format("\nIn this room you can see:{0}", Inventory.ItemList);
46                 return description;
47             }
48         }
49     }
50 }
```

```
49     public Inventory Inventory
50     {
51         get
52         {
53             return _inventory;
54         }
55     }
56
57     public List<Path> Paths
58     {
59         get
60         {
61             return _paths;
62         }
63     }
64
65     public Location() : this(new string[] { "location",
66                                     "unknown" }, "an unknown location", "This is a mysterious
67                                     location") { } // Default constructor, to make sure
68                                     the player has a location if not allocated one
69
70     public Location(string[] ids, string name, string desc) : base
71     (ids, name, desc)
72     {
73         AddIdentifier("room");
74         AddIdentifier("here");
75         AddIdentifier("location");
76         _inventory = new Inventory();
77         _paths = new List<Path>();
78     }
79
80     public GameObject? Locate(string id)
81     {
82         if (AreYou(id))
83         {
84             return this;
85         }
86         return Inventory.Fetch(id);
87     }
88
89     public void AddPath(Path path)
90     {
91         _paths.Add(path);
92     }
93
94     public Path? FetchPath(string direction)
95     {
96         foreach (Path path in Paths)
97         {
98             if (path.AreYou(direction))
99             {
100                 return path;
101             }
102         }
103     }
```



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

```
1 using System.Diagnostics;
2 using System.Runtime.InteropServices;
3 using static System.Runtime.InteropServices.JavaScript.JSType;
4
5 namespace SwinAdventure
6 {
7     internal class Program
8     {
9         static void Main(string[] args)
10        {
11            Player? player = null;
12            Command lookCommand = new LookCommand();
13            Command moveCommand = new MoveCommand();
14
15            // Player creation menu
16            while (player == null)
17            {
18                Console.WriteLine("Please enter your name -> ");
19                string? playerName = Console.ReadLine();
20                Console.WriteLine("How would you describe yourself? -> ");
21                string? playerDescription = Console.ReadLine();
22                Console.WriteLine("You are {0}, {1}.\nIs this correct? (yes/no) -> ", playerName, playerDescription);
23                bool confirmationMenuLoop = true;
24                while (confirmationMenuLoop)
25                {
26                    string? decision = Console.ReadLine().ToLower();
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;
36                        default:
37                            Console.WriteLine("Invalid option: please enter yes or no. -> ");
38                            break;
39                    }
40                }
41            }
42
43            // Create Items
44            Item shovel = new Item(new string[] { "shovel", "spade" }, "a shovel", "A sturdy shovel, the perfect tool for digging");
45            Item bronzeSword = new Item(new string[] { "sword" }, "a bronze sword", "A short sword forged from bronze");
46            Item ruby = new Item(new string[] { "gem", "ruby" }, "a red gem", "A brilliant ruby, glows with a fiery red hue");
```

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

```
75         player.Location = classroom;
76
77         // Distribute Items to Player
78         bag.Inventory.Put(ruby);
79         player.Inventory.Put(shovel);
80         player.Inventory.Put(bronzeSword);
81         player.Inventory.Put(bag);
82
83         // Distribute items to Classroom
84         laptopBag.Inventory.Put(laptop);
85         classroom.Inventory.Put(computer);
86         classroom.Inventory.Put(laptopBag);
87
88         // Distribute items to Hallway
89         hallway.Inventory.Put(bulletinBoard);
90
91         // Distribute items to Server Room
92         serverRoom.Inventory.Put(server);
93
94
95         // Introduction text
96         Console.WriteLine("-----");
97         Console.WriteLine("Welcome to Swin Adventure!");
98         Console.WriteLine("You have arrived in {0}",           ↗
99             player.Location.Name);
100
101         // Game loop
102         bool gameLoop = true;
103         while (gameLoop)
104         {
105             Console.Write("Command -> ");
106             string? playerInput = Console.ReadLine();
107             string[] inputToPass = playerInput.Split(new char[]   ↗
108                 { }, StringSplitOptions.RemoveEmptyEntries);
109             Console.WriteLine("");
110             foreach (string input in inputToPass)
111             {
112                 if (lookCommand.AreYou(input))
113                 {
114                     Console.WriteLine(lookCommand.Execute(player,   ↗
115                         inputToPass));
116                 }
117                 else if (moveCommand.AreYou(input))
118                 {
119                     Console.WriteLine(moveCommand.Execute(player,   ↗
120                         inputToPass));
121                 }
122             }
123         }
124     }
125 }
```

```
1 using SwinAdventure;
2
3 namespace TestPath
4 {
5     public class Tests
6     {
7         private Location hallway;
8         private SwinAdventure.Path testPathToHallway;
9         private Player p;
10
11         [SetUp]
12         public void Setup()
13         {
14             hallway = new Location(new string[] { "hallway" }, "the
15                 Hallway", "This is a long well lit hallway, many Swin
16                 Adventurers are roaming around.");
17             testPathToHallway = new SwinAdventure.Path(new string[]
18                 { "east", "hallway" }, hallway);
19             p = new Player("Tester", "the mighty test player");
20         }
21
22         [Test]
23         public void TestPathIsIdentifiable()
24         {
25             bool testPathIsIdentifiable = testPathToHallway.AreYou
26                 ("east");
27             Assert.That(testPathIsIdentifiable, Is.EqualTo(true));
28         }
29
30         [Test]
31         public void TestPathMovesPlayer()
32         {
33             testPathToHallway.MovePlayer(p);
34             Assert.That(p.Location, Is.EqualTo(hallway));
35         }
36     }
37 }
```

```
1 using SwinAdventure;
2
3 namespace TestMoveCommand
4 {
5     public class Tests
6     {
7         private MoveCommand move;
8         private Player testPlayer;
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");
20            hallway = new Location(new string[] { "hallway" }, "the 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]
27        public void TestMovePlayerToValidDirection()
28        {
29            string testMoveToDirection = move.Execute(testPlayer, new string[] { "move", "east" });
30            Assert.That(testMoveToDirection, Is.EqualTo("You head east \nYou have arrived in the Hallway"));
31            Assert.That(testPlayer.Location, Is.EqualTo(hallway));
32        }
33
34        [Test]
35        public void TestMovePlayerToInvalidDirection()
36        {
37            string testMoveToDirection = move.Execute(testPlayer, new string[] { "move", "west" });
38            Assert.That(testMoveToDirection, Is.EqualTo("I cannot move west"));
39            Assert.That(testPlayer.Location, Is.EqualTo(location));
40        }
41
42        [Test]
43        public void TestInvalidMove()
44        {
```

```
45     string testTextLengthNot2 = move.Execute(testPlayer, new
        string[] { "move", "test", "length", "not", "two" });
46     string testMoveIsNotFirstWord = move.Execute(testPlayer, new
        string[] { "test", "move" });
47     string testMoveToUnknownDirection = move.Execute(testPlayer,
        new string[] { "move", "unknown" });
48
49     Assert.That(testTextLengthNot2, Is.EqualTo("I don't know how
        to move like that"));
50     Assert.That(testMoveIsNotFirstWord, Is.EqualTo("Error in
        move input"));
51     Assert.That(testMoveToUnknownDirection, Is.EqualTo("I don't
        know that direction"));
52     Assert.That(testPlayer.Location, Is.EqualTo
        (location)); // Player
        stays in current location
53 }
54 }
55 }
```

```
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        {
16            location = new Location(new string[] { "location" }, "the
17                Location", "This is a test location");
18            hallway = new Location(new string[] { "hallway" }, "the
19                Hallway", "This is a long well lit hallway, many Swin
20                Adventurers are roaming around.");
21            testPathToHallway = new SwinAdventure.Path(new string[]
22                { "east", "hallway" }, hallway);
23            ruby = new Item(new string[] { "gem", "ruby" }, "a red gem",
24                "A brilliant ruby, glows with a fiery red hue");
25            location.Inventory.Put(ruby);
26        }
27
28        [Test]
29        public void TestLocationIsIdentifiable()
30        {
31            GameObject? testLocationId = location.Locate("location");
32            Assert.That(testLocationId, Is.EqualTo(location));
33        }
34
35        [Test]
36        public void TestLocationLocatesItems()
37        {
38            GameObject? testLocationLocatesRuby = location.Locate
39                ("ruby");
40            Assert.That(testLocationLocatesRuby, Is.EqualTo(ruby));
41        }
42
43        [Test]
44        public void TestLocationLocatesItself()
45        {
46            GameObject? testLocationLocatesItself = location.Locate
47                ("location");
48            Assert.That(testLocationLocatesItself, Is.EqualTo
49                (location));
50        }
51
52        [Test]
53        public void TestLocationlocatesNothing()
```



```
46     {
47         GameObject? testLocationLocatesItself = location.Locate
48             ("nothing");
49         Assert.That(testLocationLocatesItself, Is.EqualTo(null));
50     }
51     [Test]
52     public void TestLocationFullDescription()
53     {
54         location.AddPath(testPathToHallway);
55         string testLocationFullDescription =
56             location.FullDescription;
57         Assert.That(testLocationFullDescription, Is.EqualTo("You are
58             in the Location\nThis is a test location\nThere are exits
59             to the east.\nIn this room you can see:\n  a red gem
60             (gem)"));
61     }
62     [Test]
63     public void TestLocationFetchesPath()
64     {
65         location.AddPath(testPathToHallway);
66         SwinAdventure.Path? testFetchPath = location.FetchPath
67             ("east");
68         Assert.That(testFetchPath, Is.EqualTo(testPathToHallway));
69     }
70     [Test]
71     public void TestLocationFetchesNoPath()
72     {
73         SwinAdventure.Path? testFetchPath = location.FetchPath
74             ("east");
75         Assert.That(testFetchPath, Is.EqualTo(null));
76     }
77 }
```

```
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[]
43        { "me", "inventory", "inv" }, name, desc)
44        {
45            _inventory = new Inventory();
46            _location = new Location();
47        }
48
49        public GameObject? Locate(string id)
50        {
51            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.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 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
48                                     [2]);
49                 }
50                 return itemDescription3;
51             }
52         }
53     }
54 }
```

```
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 containerId)
68  {
69      IHaveInventory? container = p.Locate(containerId) as
70      IHaveInventory;
71      return container;
72  }
73  private string? LookAtIn(string thingId, IHaveInventory container)
74  {
75      GameObject? item = container.Locate(thingId);
76      if (item == null)
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 {
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 interface IHaveInventory
10    {
11        public string Name { get; }
12
13        public GameObject? Locate(string id);
14    }
15 }
16
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SwinAdventure
8 {
9     public class 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    }
```

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 }
```

```
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 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.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}",
18                                     Name, Inventory.ItemList);
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,
30                            name, desc)
31            {
32                _inventory = new Inventory();
33            }
34
35            public GameObject? Locate(string id)
36            {
37                if (AreYou(id))
38                {
39                    return this;
40                }
41                return Inventory.Fetch(id);
42            }
43 }
44
```

```
1 using SwinAdventure;
2
3 namespace TestBag
4 {
5     public class Tests
6     {
7         [Test]
8         public void TestBagLocatesItems()
9         {
10             Bag testBag = new Bag(new string[] { "bag", "testingBag"},
11                                     "test bag", "this is the test bag's description");
12             Item shovel = new Item(new string[] { "shovel", "spade" },
13                                     "a shovel", "shovel description");
14             testBag.Inventory.Put(shovel);
15
16             GameObject? testLocateShovel = testBag.Locate("shovel");
17             GameObject? testShovelRemainsInBag = testBag.Locate
18                 ("shovel");
19             Assert.That(testLocateShovel, Is.EqualTo(shovel));
20             Assert.That(testShovelRemainsInBag, Is.EqualTo(shovel));
21         }
22
23         [Test]
24         public void TestBagLocatesItself()
25         {
26             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
27                                     "test bag", "this is the test bag's description");
28
29             GameObject? testLocateBagID1 = testBag.Locate("bag");
30             GameObject? testLocateBagID2 = testBag.Locate("testingBag");
31             Assert.That(testLocateBagID1, Is.EqualTo(testBag));
32             Assert.That(testLocateBagID2, Is.EqualTo(testBag));
33         }
34
35         [Test]
36         public void TestBagLocatesNothing()
37         {
38             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
39                                     "test bag", "this is the test bag's description");
40
41             GameObject? testLocateShovel = testBag.Locate("shovel");
42             Assert.That(testLocateShovel, Is.EqualTo(null));
43         }
44
45         [Test]
46         public void TestBagFullDescription()
47         {
48             Bag testBag = new Bag(new string[] { "bag", "testingBag" },
49                                     "test bag", "this is the test bag's description");
50             Item shovel = new Item(new string[] { "shovel", "spade" },
51                                     "a shovel", "shovel description");
52             Item bronzeSword = new Item(new string[] { "sword", "bronze
53 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
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" },
12                                   "a 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" },
29                                   "a 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" },
44                                   "a 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);
50
51            Item? testShovel = testInventory.Fetch("shovel");
52            Item? testBronzeSword = testInventory.Fetch("sword");
53            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" },
55                                "a 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
64             ("shovel");
65         Assert.That(testFetchShovel, Is.EqualTo(shovel));
66         Assert.That(testShovelInInventory, Is.EqualTo(false));
67     }
68
69     [Test]
70     public void TestItemList()
71     {
72         Item shovel = new Item(new string[] { "shovel", "spade" },
73                                "a shovel", "shovel description");
74         Item bronzeSword = new Item(new string[] { "sword", "bronze
75 sword" }, "a bronze sword", "bronze sword description");
76         Inventory testInventory = new Inventory();
77         testInventory.Put(shovel);
78         testInventory.Put(bronzeSword);
79
80         string testInventoryList = testInventory.ItemList;
81         Assert.That(testInventoryList, Is.EqualTo("\n a shovel
82 (shovel)\n a bronze sword (sword)"));
83     }
84 }
```

```
1 using SwinAdventure;
2
3 namespace TestIdentifiableObject
4 {
5     public class Tests
6     {
7
8         [Test]
9         public void TestAreYou()
10        {
11            IdentifiableObject myIdsents = new IdentifiableObject(new string[] { "fred", "bob" });
12
13            bool fred = myIdsents.AreYou("fred");
14            Assert.That(fred, Is.EqualTo(true));
15            bool bob = myIdsents.AreYou("bob");
16            Assert.That(bob, Is.EqualTo(true));
17        }
18
19        [Test]
20
21        public void TestNotAreYou()
22        {
23            IdentifiableObject myIdsents = new IdentifiableObject(new string[] { "fred", "bob" });
24
25            bool wilma = myIdsents.AreYou("wilma");
26            Assert.That(wilma, Is.EqualTo(false));
27            bool boby = myIdsents.AreYou("boby");
28            Assert.That(boby, Is.EqualTo(false));
29        }
30
31        [Test]
32
33        public void TestCaseSensitive()
34        {
35            IdentifiableObject myIdsents = new IdentifiableObject(new string[] { "fred", "bob" });
36
37            bool fred = myIdsents.AreYou("FRED");
38            Assert.That(fred, Is.EqualTo(true));
39            bool bob = myIdsents.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 myIds = new IdentifiableObject(new string[] {});
58
59         string firstID = myIds.FirstID;
60         Assert.That(firstID, Is.EqualTo(""));
61     }
62
63     [Test]
64
65     public void TestAddIDs()
66     {
67         IdentifiableObject myIds = new IdentifiableObject(new string[] { "fred", "bob" });
68         myIds.AddIdentifier("wilma");
69
70         bool fred = myIds.AreYou("fred");
71         Assert.That(fred, Is.EqualTo(true));
72         bool bob = myIds.AreYou("bob");
73         Assert.That(bob, Is.EqualTo(true));
74         bool wilma = myIds.AreYou("wilma");
75         Assert.That(wilma, Is.EqualTo(true));
76     }
77 }
78 }
```

```
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
16                sword");
17            Assert.That(testBronzeSwordID1, Is.EqualTo(true));
18            Assert.That(testBronzeSwordID2, Is.EqualTo(true));
19        }
20
21        [Test]
22        public void TestItemShortDescription()
23        {
24            Item bronzeSword = new Item(new string[] { "sword", "bronze
25                sword" }, "a bronze sword", "bronze sword description");
26            string testBronzeSword = bronzeSword.ShortDescription;
27            Assert.That(testBronzeSword, Is.EqualTo("a bronze sword
28                (sword)"));
29        }
30
31        [Test]
32        public void TestItemFullDescription()
33        {
34            Item bronzeSword = new Item(new string[] { "sword", "bronze
35                sword" }, "a bronze sword", "bronze sword description");
36            string testBronzeSword = bronzeSword.FullDescription;
37            Assert.That(testBronzeSword, Is.EqualTo("bronze sword
38                description"));
39        }
40    }
41 }
```

```
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]
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",
19                "A 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]
49        public void TestPlayerLocatesItself()
50        {
51            GameObject? testLocatePMe = p.Locate("me");
52            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, 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 }
```

```
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        private Location location;
13
14        [SetUp]
15        public void Setup()
16        {
17            look = new LookCommand();
18            testPlayer = new Player("testPlayer", "test player description");
19            gem = new Item(new string[] { "gem" }, "a gem", "gem's description");
20            bag = new Bag(new string[] { "bag" }, "a bag", "bag's description");
21            location = new Location(new string[] { "location" }, "the Location", "This is a test location");
22            testPlayer.Location = location;
23        }
24
25        [Test]
26        public void TestLookAtMe()
27        {
28            string testLookAtInventory = look.Execute(testPlayer, new 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" });
37            Assert.That(testLookAtGem, Is.EqualTo("gem's description"));
38        }
39
40        [Test]
41        public void TestLookAtUnknown()
42        {
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 string[] { "look", "at", "gem", "in", "location" });
122     Assert.That(testLookAtGemInLocation, Is.EqualTo("I cannot find the gem in the location"));
123 }
124
125 [Test]
126 public void TestLookAtGemInBagInLocation()
127 {
128     bag.Inventory.Put(gem);
129     location.Inventory.Put(bag);
130     string testLookAtGemInBag = look.Execute(testPlayer, new
        string[] { "look", "at", "gem", "in", "bag" });
131     Assert.That(testLookAtGemInBag, Is.EqualTo("gem's description"));
132 }
133 }
134 }
```