Player.cs


```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Reflection.Metadata.Ecma335;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Iteration1
9  {
10     public class Player : GameObject, IHaveInventory
11     {
12         private Inventory _inventory;
13
14         public Player(string name, string desc) : base(new string[] {"me",
             "inventory"}, name, desc)
15         {
16             _inventory = new Inventory();
17         }
18
19         public GameObject Locate(string id)
20         {
21             if (AreYou(id))
22             {
23                 return this;
24             }
25             else if (_inventory.HasItem(id))
26             {
27                 return _inventory.Fetch(id);
28             }
29             return null;
30         }
31
32         public string Name
33         {
34             get => base.Name;
35         }
36
37         public override string FullDescription
38         {
39             get
40             {
41                 string fulldesc = "";
42                 fulldesc += $"You are {Name}, {base.FullDescription}\n";
43                 fulldesc += "You are carrying\n";
44                 fulldesc += $"{_inventory.ItemList}";
45                 return fulldesc;
46             }
47         }
48
```

```
49          public Inventory Inventory
50          {
51              get => _inventory;
52          }
53      }
54  }
55
```

Bag.cs


```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Iteration1
8  {
9      public class Bag : Item, IHaveInventory
10     {
11         private Inventory _inventory;
12
13         public Bag(string[] ids, string name, string desc) : base(ids,
             name, desc)
14         {
15             _inventory = new Inventory();
16         }
17
18         public GameObject Locate(string id)
19         {
20             if (AreYou(id))
21             {
22                 return this;
23             }
24             else if (_inventory.HasItem(id))
25             {
26                 return _inventory.Fetch(id);
27             }
28             return null;
29         }
30
31         public string Name
32         {
33             get => base.Name;
34         }
35
36         public override string FullDescription
37         {
38             get
39             {
40                 string description = null;
41                 description += $"In the {this.Name} you can see:\n";
42                 description += _inventory.ItemList;
43                 return description;
44             }
45         }
46
47         public Inventory Inventory
48         {
```

```
49              get => _inventory;
50          }
51      }
52  }
```

IHaveInventory.cs


```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5 using System.Threading.Tasks;
 6
 7 namespace Iteration1
 8 {
 9     interface IHaveInventory
10     {
11         GameObject Locate(string id);
12
13         string Name
14         {
15             get => Name;
16         }
17     }
18 }
19
```

Command.cs


```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Iteration1
8  {
9      public abstract class Command : IdentifiableObject
10     {
11         public Command(string[] ids) : base(ids)
12         {
13
14         }
15
16         public abstract string Execute(Player p, string[] text);
17     }
18 }
19
```

LookCommand.cs


```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Iteration1
 8  {
 9      public class LookCommand : Command
10      {
11          public LookCommand() : base(new string[] { "look" })
12          {
13
14          }
15
16          public override string Execute(Player p, string[] text)
17          {
18              string containerId;
19              string thingId;
20              string thing;
21              IHaveInventory container;
22              Item item;
23              var array = text[0].Split(" ");
24
25              if (array.Length != 3 && array.Length != 5)
26              {
27                  return "I don't know how to look like that!";
28              }
29              else if (array[0] != "look")
30              {
31                  return "Error in look input";
32              }
33              else if (array[1] != "at")
34              {
35                  return "What do you want to look at?";
36              }
37
38              if (array.Length == 5)
39              {
40                  if (array[3] != "in")
41                  {
42                      return "What do you want to look in?";
43                  }
44                  else
45                  {
46                      thingId = array[2];
47                      containerId = array[4];
48                      container = FetchContainer(p, containerId);
49                      if (container != null)
```

```
50                      {
51                          thing = LookAtIn(thingId, container);
52                          if (thing != null)
53                          {
54                              return thing;
55                          }
56                          else
57                          {
58                              return $"I cannot find the {thingId} in the
                        {containerId}";
59                          }
60                      }
61                      else
62                      {
63                          return $"I cannot find the {containerId}";
64                      }
65                  }
66              }
67              else if (array.Length == 3)
68              {
69                  thingId = array[2];
70                  thing = LookAtIn(thingId, p);
71                  if (thing != null)
72                  {
73                      return thing;
74                  }
75                  else
76                  {
77                      return $"I cannot find the {thingId}";
78                  }
79              }
80              else
81              {
82                  return null;
83              }
84
85          }
86
87          private IHaveInventory FetchContainer(Player p, string
              containerId)
88          {
89              if (p.Locate(containerId) == null)
90              {
91                  return null;
92              }
93              return p.Locate(containerId) as IHaveInventory;
94          }
95
96          private string LookAtIn(string thingId, IHaveInventory container)
```

```
 97          {
 98              if (container.Locate(thingId) == null)
 99              {
100                  return null;
101              }
102              return container.Locate(thingId).FullDescription;
103          }
104      }
105
106  }
107
```

LookCommandUnitTests.cs


```
1  using Iteration1;
2  using System.Numerics;
3
4  namespace LookCommandUnitTests
5  {
6      public class Tests
7      {
8          private Player _player;
9          private Item sword;
10         private Item ak47;
11         private Item gems;
12         private Bag _bag;
13         private LookCommand look;
14
15         [SetUp]
16         public void Setup()
17         {
18             _bag = new Bag(new string[] { "bag", "1" }, "Bag 1", "This is
                 the 1st bag of the player!");
19             _player = new Player("Chien", "A boy with high curiosity");
20             sword = new Item(new string[] { "sword", "melee" }, "bronze
                 sword", "Melee weapon. High damage.");
21             ak47 = new Item(new string[] { "ak47", "gun" }, "ak47", "Gun.
                 High Damage.");
22             gems = new Item(new string[] { "gem" }, " collectible gems",
                 "Using for buying weapons");
23             _player.Inventory.Put(sword);
24             _player.Inventory.Put(ak47);
25             look = new LookCommand();
26         }
27
28         [Test]
29         public void TestLookAtMe()
30         {
31             Assert.That((look.Execute(_player, new string[] { "look at
                 inventory" })), Is.EqualTo(_player.FullDescription));
32             Assert.Pass();
33         }
34
35         [Test]
36         public void TestLookAtGem()
37         {
38             _player.Inventory.Put(gems);
39             Assert.That((look.Execute(_player, new string[] { "look at
                 gem" })), Is.EqualTo(gems.FullDescription));
40             Assert.Pass();
41         }
42
43         [Test]
```

```
44          public void TestLookAtUnk()
45          {
46              Assert.That((look.Execute(_player, new string[] { "look at
                  gem" })), Is.EqualTo("I cannot find the gem"));
47              Assert.Pass();
48          }
49
50          [Test]
51          public void TestLookAtGemInMe()
52          {
53              _player.Inventory.Put(gems);
54              Assert.That((look.Execute(_player, new string[] { "look at gem
                  in inventory" })), Is.EqualTo(gems.FullDescription));
55              Assert.Pass();
56          }
57
58          [Test]
59          public void TestLookAtGemInBag()
60          {
61              _bag.Inventory.Put(gems);
62              _player.Inventory.Put(_bag);
63              Assert.That((look.Execute(_player, new string[] { "look at gem
                  in bag" })), Is.EqualTo(gems.FullDescription));
64              Assert.Pass();
65          }
66
67
68          [Test]
69          public void TestLookAtGemInNoBag()
70          {
71              _bag.Inventory.Put(gems);
72              Assert.That((look.Execute(_player, new string[] { "look at gem
                  in bag" })), Is.EqualTo("I cannot find the bag"));
73              Assert.Pass();
74          }
75
76          [Test]
77          public void TestLookAtNoGemInBag()
78          {
79              _player.Inventory.Put(_bag);
80              Assert.That((look.Execute(_player, new string[] { "look at gem
                  in bag" })), Is.EqualTo("I cannot find the gem in the bag"));
81              Assert.Pass();
82          }
83
84          [Test]
85          public void TestInvalidLook()
86          {
87              Assert.That((look.Execute(_player, new string[] { "look around
```

```
                me" })), Is.EqualTo("What do you want to look at?"));
88              Assert.That((look.Execute(_player, new string[] { "hello me
                friend" })), Is.EqualTo("Error in look input"));
89              Assert.That((look.Execute(_player, new string[] { "look at gem
                a b" })), Is.EqualTo("What do you want to look in?"));
90              Assert.That((look.Execute(_player, new string[] {"look"})),
                Is.EqualTo("I don't know how to look like that!"));
91              Assert.Pass();
92          }
93      }
94  }
```

Screenshot of Test Passing