COS20007 - Object Oriented Programming

10.1C - Case Study - Advanced Iteration 8 - Command Processor

Student name: Nguyen Duc Manh
ID: 105547489

Newer codes

```
1  namespace SwinAdventure;
2  public class Program
3  {
4      public static void Main(string[] args)
5      {
6          Console.WriteLine("Enter your name: ");
7          string name = Console.ReadLine();
8          Console.WriteLine("Enter your current major: ");
9          string major = Console.ReadLine();
10
11         //Create player and command processor
12         Player player = new(name, major);
13         CommandProcessor cp = new();
14
15         //Create other thing
16         Item itm1 = new(["hdmi"], "HDMI cord", "can connect to large
             screen");
17         Item itm2 = new(["usb"], "an USB", "can store up to 1TB of data");
18         Item itm3 = new(["mouse"], "a mouse", "gaming mouse with 0
             latency");
19         Bags bag = new(["bag"], "a bag", "this bag is made by leather");
20         Location duytan = new(["duytan"], "duytan", "Innovation Center of
             Swinburne");
21         Location duongkhue = new(["duongkhue"], "duongkhue", "Global
             Citizen Education");
22         Path north = new(["north"], "north move", "Duy Tan street",
             duytan);
23         Path south = new(["south"], "south move", "Cau Giay street",
             duongkhue);
24
25         //Add them
26         player.Inventory.Put(itm1);
27         player.Inventory.Put(itm2);
28         player.Inventory.Put(bag);
29         bag.Inventory.Put(itm3);
30         duytan.Inventory.Put(itm1);
31         player.Location = duytan;
32         duytan.AddPath(south);
33         duongkhue.AddPath(north);
34
35         //Start the program
36         while (true)
37         {
38             Console.Write("What do you want to do? (type 'exit' to quit)\
                 n> ");
39             string input = Console.ReadLine()?.Trim();
40
41             if (string.IsNullOrWhiteSpace(input))
42                 continue;
```

```
43
44              if (input.ToLower() == "exit")
45                  break;
46
47              string[] commandWords = input.Split(' ',
                  StringSplitOptions.RemoveEmptyEntries);
48              string result = cp.Execute(player, commandWords);
49              Console.WriteLine(result);
50          }
51      }
52  }
```

```
1  namespace SwinAdventure
2  {
3      public class CommandProcessor : Command
4      {
5          readonly List<Command> commands;
6
7          public CommandProcessor() : base(["Command"])
8          {
9              commands =
10             [
11                 new MoveCommand(),
12                 new LookCommand(),
13             ];
14         }
15
16         public override string Execute(Player p, string[] text)
17         {
18             if (text.Length == 0 || string.IsNullOrWhiteSpace(text[0]))
19                 return "Please enter a command.";
20
21             string userInput = text[0].ToLower();
22
23             foreach (Command command in commands)
24             {
25                 if (command.AreYou(userInput))
26                 {
27                     return command.Execute(p, text);
28                 }
29             }
30
31             return "I can not find that command!";
32         }
33
34     }
35 }
36
```

```
1  using SwinAdventure;
2  using Path = SwinAdventure.Path;
3
4  namespace ObjTest;
5
6  public class CommandProcessorTest
7  {
8      Player player;
9      CommandProcessor cp;
10     Item itm;
11     Bags bag;
12     Location duytan;
13     Location duongkhue;
14     Path north;
15     Path south;
16
17     [SetUp]
18     public void Setup()
19     {
20         player = new("Duy", "Software Engineering");
21         itm = new(["hdmi"], "HDMI cord", "can connect to large screen");
22         bag = new(["bag"], "a bag", "this bag is made by leather");
23         duytan = new(["duytan"], "duytan", "Innovation Center of
             Swinburne");
24         duongkhue = new(["duongkhue"], "duongkhue", "Global Citizen
             Education");
25         north = new(["north"], "north move", "Duy Tan street", duytan);
26         south = new(["south"], "south move", "Cau Giay street", duongkhue);
27
28         cp = new CommandProcessor();
29     }
30
31     [Test]
32     public void LookCommand()
33     {
34         bag.Inventory.Put(itm);
35         player.Inventory.Put(bag);
36         LookCommand lookCommand = new();
37
38         string input1 = cp.Execute(player, ["look", "at", "me"]);
39         string input2 = cp.Execute(player, ["look", "at", "hdmi", "in",
             "bag"]);
40         string expect1 = lookCommand.Execute(player, ["look", "at", "me"]);
41         string expect2 = lookCommand.Execute(player, ["look", "at", "
             hdmi", "in", "bag"]);
42
43         Assert.That(input1, Is.EqualTo(expect1));
44         Assert.That(input2, Is.EqualTo(expect2));
45     }
```

```
46
47      [Test]
48      public void MoveCommand()
49      {
50          duytan.AddPath(south);
51          duongkhue.AddPath(north);
52          player.Location = duytan;
53          MoveCommand moveCommand = new();
54          string input1 = cp.Execute(player, ["move", "south"]);
55          Assert.That(player.Location.AreYou("duongkhue"), Is.True);
56          string input2 = cp.Execute(player, ["move", "north"]);
57          Assert.That(player.Location.AreYou("duytan"), Is.True);
58      }
59
60      [Test]
61      public void CommandNotFind()
62      {
63          string input = cp.Execute(player, ["not", "a", "command"]);
64          Assert.That(input, Is.EqualTo("I can not find that command!"));
65      }
66
67      [Test]
68      public void CommandWithNoInput()
69      {
70          string input = cp.Execute(player, []);
71          Assert.That(input, Is.EqualTo("Please enter a command."));
72      }
73  }
74
```

Earlier codes

```
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 Bags : Item, IHaveInventory
10     {
11         Inventory _inventory;
12
13         public Bags(string[] idents, string name, string desc) : base
             (idents, 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             } return null;
28         }
29
30         public override string FullDescription
31         {
32             get { return $"In the {Name} you can see:\n{_inventory.ItemList
                 ()}" ; }
33         }
34
35         public Inventory Inventory
36         {
37             get { return _inventory; }
38         }
39     }
40 }
41
```

```
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 : IdenObj
10     {
11         public Command(string[] ids) : base(ids)
12         {
13             //
14         }
15
16         public abstract string Execute(Player p, string[] text);
17     }
18 }
19
```

```
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 GameObject : IdenObj
10     {
11         string _description;
12         string _name;
13
14         public GameObject(string[] idents, string name, string desc) : base
             (idents)
15         {
16             _name = name;
17             _description = desc;
18         }
19
20         public string Name { get { return _name; } }
21
22         public string ShortDescription { get { return $"{_name}
             ({FirstId})"; } }
23
24         public virtual string FullDescription { get { return
             _description; } }
25     }
26 }
27
```

```
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 IdenObj
10     {
11         //fields
12         private List<string> _identifiers;
13         string _myStudentID = "7489";
14
15         //constructor
16         public IdenObj(string[] idents)
17         {
18             _identifiers = new List<string>();
19             if (idents != null)
20             {
21                 for (int i = 0; i < idents.Length; i++)
22                 {
23                     _identifiers.Add(idents[i].ToLower());
24                 }
25             }
26         }
27
28         //methods
29         public bool AreYou(string id)
30         {
31             return _identifiers.Contains(id.ToLower());
32         }
33
34         public string FirstId
35         {
36             get
37             {
38                 if( _identifiers.Count == 0)
39                 {
40                     return "";
41                 } else { return _identifiers.First(); }
42             }
43         }
44
45         public void AddIdentifier(string id)
46         {
47             _identifiers.Add(id.ToLower());
48         }
49
```

```
50          public void PrivilegeEscalation(string pin)
51          {
52              if(pin.Length == 4)
53              {
54                  if(pin == _myStudentID) //105547489
55                  {
56                      _identifiers[0] = _myStudentID;
57                  }
58              }
59              else
60              {
61                  return;
62              }
63
64          }
65      }
66  }
67
```

```
 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 GameObject Locate(string id);
12
13          public string Name { get; }
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 Item : GameObject
10      {
11          public Item(string[] idents, string name, string desc) : base
              (idents, name, desc)
12          {
13              //not yet
14          }
15      }
16  }
17
```

```
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         List<Item> _items;
12
13         public Inventory()
14         {
15             _items = new List<Item>();
16         }
17
18         public bool HasItem(string id)
19         {
20             foreach (Item item in _items)
21             {
22                 if (item.AreYou(id))
23                 {
24                     return true;
25                 }
26             }
27             return false;
28         }
29
30         public void Put(Item itm)
31         {
32             _items.Add(itm);
33         }
34
35         //public void RemoveItm(Item itm)
36         //{
37         //    if (_items.Contains(itm))
38         //    {
39         //        _items.Remove(itm);
40         //    }
41         //}
42
43         public Item Take(string id)
44         {
45             foreach (Item item in _items)
46             {
47                 if (item.AreYou(id))
48                 {
49                     _items.Remove(item);
```

```
50                      return item;
51                  }
52              }
53              return null;
54          }
55
56          public Item Fetch(string id)
57          {
58              foreach (Item item in _items)
59              {
60                  if (item.AreYou(id))
61                  {
62                      return item;
63                  }
64              }
65              return null;
66          }
67
68          public string ItemList()
69          {
70              string listitm = "";
71              foreach (Item item in _items)
72              {
73                  listitm = listitm + item.ShortDescription + "\n";
74              }
75              return listitm;
76          }
77      }
78  }
79
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace SwinAdventure
 8  {
 9      public class Location : GameObject, IHaveInventory
10      {
11          Inventory _inventory;
12
13          public Location(string[] idents, string name, string desc) : base
              (idents, 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              return _inventory.Fetch(id);
25          }
26
27          public override string FullDescription
28          {
29              get
30              {
31                  return $"You are in: {Name}, {base.FullDescription}. Here
                      you can see: {_inventory.ItemList()}";
32              }
33          }
34
35          public Inventory Inventory
36          {
37              get { return _inventory; }
38          }
39      }
40  }
41
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Xml.Linq;

namespace SwinAdventure
{
    public class LookCommand : Command
    {
        IHaveInventory container;
        GameObject item;
        Player p;
        Location location;

        public LookCommand() : base(["look"]) { }

        public override string Execute(Player p, string[] text)
        {
            if (text.Length == 1 && text[0].ToLower() == "look")
            {
                return p.Location.FullDescription;
            }
            if (text.Length == 3 || text.Length == 5)
            {
                if (text[0] != "look")
                    return "Error in look input";
                if (text[1] != "at")
                    return "What do you want to look at?";
                if (text.Length == 5 && text[3] != "in")
                    return "What do you want to look in?";
                if (text.Length == 3)
                {
                    container = p;
                }
                else
                {
                    container = FetchContainer(p, text[4]);
                    if (container == null)
                        return $"I cannot find the {text[4]}";
                }

                return LookAtIn(text[2], container);
            }
            else
                return "I don't know how to look like that";
```

```
50          }
51
52          private IHaveInventory? FetchContainer(Player p, string
              containerId)
53          {
54              return p.Locate(containerId) as IHaveInventory;
55          }
56
57          private string LookAtIn(string thingId, IHaveInventory container)
58          {
59              if (container.Locate(thingId) != null)
60              {
61                  return container.Locate(thingId).FullDescription;
62              }
63              else
64                  return $"I cannot find the {thingId}";
65          }
66      }
67  }
68
```

```
1  namespace SwinAdventure
2  {
3      public class MoveCommand : Command
4      {
5          Player p;
6
7          public MoveCommand() : base(["move", "go", "head", "leave"])
8          { }
9
10         public override string Execute(Player p, string[] text)
11         {
12             if (text.Length < 2)
13             {
14                 return "I don't know how to move like that";
15             }
16
17             string id = text[1].ToLower();
18             GameObject obj = p.Location.Locate(id);
19
20             if (obj == null)
21             {
22                 return "There is no path in that direction.";
23             }
24
25             Path path = obj as Path;
26             if (path != null)
27             {
28                 p.Location = path.Destination;
29                 return $"You move {id} to {path.Destination.Name}";
30             }
31             else
32                 return "That doesn't seem like a valid path.";
33         }
34     }
35 }
36
```

```
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         Inventory _inventory;
12
13         public Player (string name, string desc) : base(new string[] {"
             me", "inventory"}, 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             var itm = _inventory.Fetch(id);
25             if (itm != null)
26             {
27                 return itm;
28             }
29             if (Location != null)
30             {
31                 return Location.Locate(id);
32             }
33             return null;
34         }
35
36         public override string FullDescription
37         {
38             get
39             {
40                 return $"{Name}, {base.ShortDescription}.You are carrying:
                     {_inventory.ItemList()}";
41             }
42         }
43
44         public Inventory Inventory { get { return _inventory; } }
45
46         public Location Location { get; set; }
47     }
```

```
48  }
49
```

```
1  namespace SwinAdventure;
2
3  public class BagsTest
4  {
5      Item _item1;
6      Item _item2;
7      Bags _bag1;
8      Bags _bag2;
9
10     [SetUp]
11     public void Setup()
12     {
13         _item1 = new Item(["Ram"], "a Ram", "an NVIDIA Ram");
14         _item2 = new Item(["CPU"], "a CPU", "an Intel CPU");
15         _bag1 = new Bags(["Bag1"], "bag test 1", "This bag is huge");
16         _bag2 = new Bags(["Bag2"], "bag test 2", "This bag is small");
17         _bag1.Inventory.Put(_item1);
18         _bag1.Inventory.Put(_item2);
19     }
20
21     [Test]
22     public void BagLocatesItemTest()
23     {
24         Assert.That(_bag1.Inventory.HasItem("Ram"));
25         Assert.That(_bag1.Inventory.HasItem("CPU"));
26         Assert.That(_bag1.Locate("Ram"), Is.EqualTo(_item1));
27         Assert.That(_bag1.Locate("CPU"), Is.EqualTo(_item2));
28     }
29
30     [Test]
31     public void BagLocatesItselfTest()
32     {
33         Assert.That(_bag1.Locate("Bag1"), Is.EqualTo(_bag1));
34         Assert.That(_bag2.Locate("Bag2"), Is.EqualTo(_bag2));
35     }
36
37     [Test]
38     public void BagLocatesNothingTest()
39     {
40         Assert.That(_bag1.Locate("abc"), Is.Null);
41         Assert.That(_bag2.Locate("xyz"), Is.Null);
42     }
43
44     [Test]
45     public void BagFullDescriptionTest()
46     {
47         Assert.That(_bag1.FullDescription, Is.EqualTo("In the bag test 1
           you can see:\na Ram (ram)\na CPU (cpu)\n"));
48     }
```

```
49
50      [Test]
51      public void BagInBagTest()
52      {
53          Item _item3 = new Item(["Mouse"], "a Mouse", "a wireless mouse");
54          _bag1.Inventory.Put(_bag2);
55          _bag2.Inventory.Put(_item3);
56
57          Assert.That(_bag1.Locate("Bag2"), Is.EqualTo(_bag2)); //Can locate
              bag2 in bag1's inventoyr
58          Assert.That(_bag1.Locate("Ram"), Is.EqualTo(_item1)); //bag1 still
              can locates other items
59          Assert.That(_bag1.Locate("Mouse"), Is.Null); //bag1 can't search
              for bag2's item
60      }
61  }
62
```

```
1  namespace SwinAdventure
2  {
3      public class InventoryTest
4      {
5          Item _item;
6          Inventory _inventory;
7
8          [SetUp]
9          public void Setup()
10         {
11             _item = new(["HDMI"], "HDMI cord", "can connect to large
                 screen");
12             _inventory = new Inventory();
13         }
14
15         [Test]
16         public void FindItemTest()
17         {
18             _inventory.Put(_item);
19             Assert.That(_inventory.HasItem(_item.FirstId), Is.True);
20         }
21
22         [Test]
23         public void NoItemFindTest()
24         {
25             Assert.That(_inventory.HasItem("Mouse"), Is.False);
26         }
27
28         [Test]
29         public void FetchItemTest()
30         {
31             _inventory.Put(_item);
32             Assert.That(_inventory.Fetch(_item.FirstId), Is.EqualTo
                 (_item));
33         }
34
35         [Test]
36         public void TakeItemTest()
37         {
38             _inventory.Put(_item);
39             _inventory.Take(_item.FirstId);
40             Assert.That(_inventory.HasItem(_item.FirstId), Is.False);
41         }
42
43         [Test]
44         public void TestItemList()
45         {
46             _inventory.Put(_item);
47             Assert.That(_inventory.ItemList, Is.EqualTo("HDMI cord (hdmi)
```

```
                \n"));
48          }
49      }
50  }
51
```

```
1  using Microsoft.VisualStudio.TestPlatform.ObjectModel;
2
3  namespace SwinAdventure
4  {
5      public class ItemTest
6      {
7          Item laptop;
8
9          [SetUp]
10         public void Setup()
11         {
12             laptop = new Item(new string[] { "laptop" }, "a laptop", "This
               is a Swinburne laptop");
13         }
14
15         [Test]
16         public void TestItemIdentifiable()
17         {
18             var areyou2 = laptop.AreYou("laptop");
19             Assert.IsTrue(areyou2);
20         }
21
22         [Test]
23         public void TestShortDescription()
24         {
25             Assert.That(laptop.ShortDescription, Is.EqualTo("a laptop
               (laptop)"));
26         }
27
28         [Test]
29         public void TestFullDescription()
30         {
31             Assert.That(laptop.FullDescription, Is.EqualTo("This is a
               Swinburne laptop"));
32         }
33
34         [Test]
35         public void PrivilegeEscalationTest()
36         {
37             var firstID = new string[] { "sword", "blade" };
38             var item = new Item(firstID, "Sword", "A sharp blade");
39             item.PrivilegeEscalation("7489");
40
41             Assert.That(item.FirstId, Is.EqualTo("7489"));
42         }
43
44     }
45 }
```

```
1  namespace SwinAdventure;
2
3  public class LocationTest
4  {
5      Location _location;
6      Player _player;
7
8      [SetUp]
9      public void Setup()
10     {
11         _location = new(["duytan", "location"], "80 duy tan", "Innovation
             Space");
12         _player = new("Manh", "A student at Swinburne");
13     }
14
15     [Test]
16     public void LocationLocateItself()
17     {
18         Assert.IsTrue(_location.AreYou("duytan"));
19     }
20
21     [Test]
22     public void LocationLocateItem()
23     {
24         Item item = new(["usb"], "an usb", "this is an usb");
25         _location.Inventory.Put(item);
26         Assert.IsTrue(_location.Locate("usb").AreYou("usb"));
27     }
28
29     [Test]
30     public void PlayerInLocation()
31     {
32         _player.Location = _location;
33         string expect = _location.FullDescription;
34         Assert.That(_player.Location.FullDescription, Is.EqualTo(expect));
35     }
36
37     [Test]
38     public void PlayerLocateLocation()
39     {
40         _player.Location = _location;
41         Assert.IsTrue(_player.Locate("duytan").AreYou("duytan"));
42     }
43
44     [Test]
45     public void PlayerLocateItem()
46     {
47         Item item = new(["usb"], "an usb", "magical usb can stores 1TB");
48         _location.Inventory.Put(item);
```

```
49          _player.Location = _location;
50          Assert.IsTrue(_player.Locate("usb").AreYou("usb"));
51      }
52  }
53
```

```
using System.Numerics;

namespace SwinAdventure;

public class LookCommandTest
{
    Player me;
    Item gem;
    Bags bag;
    Command look;

    [SetUp]
    public void Setup()
    {
        me = new Player("me", "the main character of the game");
        gem = new Item(["gem"], "a bright red stone", "it sparkles in the
          light");
        bag = new Bags(["bag"], "a small bag", "it has a zipper");
        look = new LookCommand();

        me.Inventory.Put(gem);
        me.Inventory.Put(bag);
        bag.Inventory.Put(gem);
    }

    [Test]
    public void TestLookAtMe()
    {
        string prompt = look.Execute(me, ["look", "at", "inventory"]);
        string expected = me.FullDescription;
        Assert.That(prompt, Is.EqualTo(expected));
    }

    [Test]
    public void TestLookAtGem()
    {
        string prompt = look.Execute(me, ["look", "at", "gem"]);
        string expected = gem.FullDescription;
        Assert.That(prompt, Is.EqualTo(expected));
    }

    [Test]
    public void TestLookAtUnk()
    {
        string prompt = look.Execute(me, ["look", "at", "unk"]);
        string expected = "I cannot find the unk";
        Assert.That(prompt, Is.EqualTo(expected));
    }

```

```
49      [Test]
50      public void TestLookAtGemInMe()
51      {
52          string prompt = look.Execute(me, ["look", "at", "gem", "in",
              "inventory"]);
53          string expected = gem.FullDescription;
54          Assert.That(prompt, Is.EqualTo(expected));
55      }
56
57      [Test]
58      public void TestLookAtGemInBag()
59      {
60          Assert.That(me.Locate("bag"), Is.EqualTo(bag)); //test that bag
              is in player's inventory
61          string prompt = look.Execute(me, ["look", "at", "gem", "in",
              "bag"]);
62          string expected = gem.FullDescription;
63          Assert.That(prompt, Is.EqualTo(expected));
64      }
65
66      [Test]
67      public void TestLookAtGemInNoBag()
68      {
69          me.Inventory.Take("bag"); //remove bag from player's inventory
70          string prompt = look.Execute(me, ["look", "at", "gem", "in",
              "bag"]);
71          string expected = "I cannot find the bag";
72          Assert.That(prompt, Is.EqualTo(expected));
73      }
74
75      [Test]
76      public void TestLookAtNoGemInBag()
77      {
78          bag.Inventory.Take("gem"); //remove gem from player's inventory
79          string prompt = look.Execute(me, ["look", "at", "gem", "in",
              "bag"]);
80          string expected = "I cannot find the gem";
81          Assert.That(prompt, Is.EqualTo(expected));
82      }
83
84      [Test]
85      public void TestInvalidLook()
86      {
87          string prompt = look.Execute(me, ["look", "at", "gem", "not in",
              "bag"]);
88          string expected = "What do you want to look in?";
89          Assert.That(prompt, Is.EqualTo(expected));
90      }
91
```

```
 92      [Test]
 93      public void TestInvalidLook2()
 94      {
 95          string prompt = look.Execute(me, ["kool", "at", "gem", "in",
             "bag"]);
 96          string expected = "Error in look input";
 97          Assert.That(prompt, Is.EqualTo(expected));
 98      }
 99  }
100
```

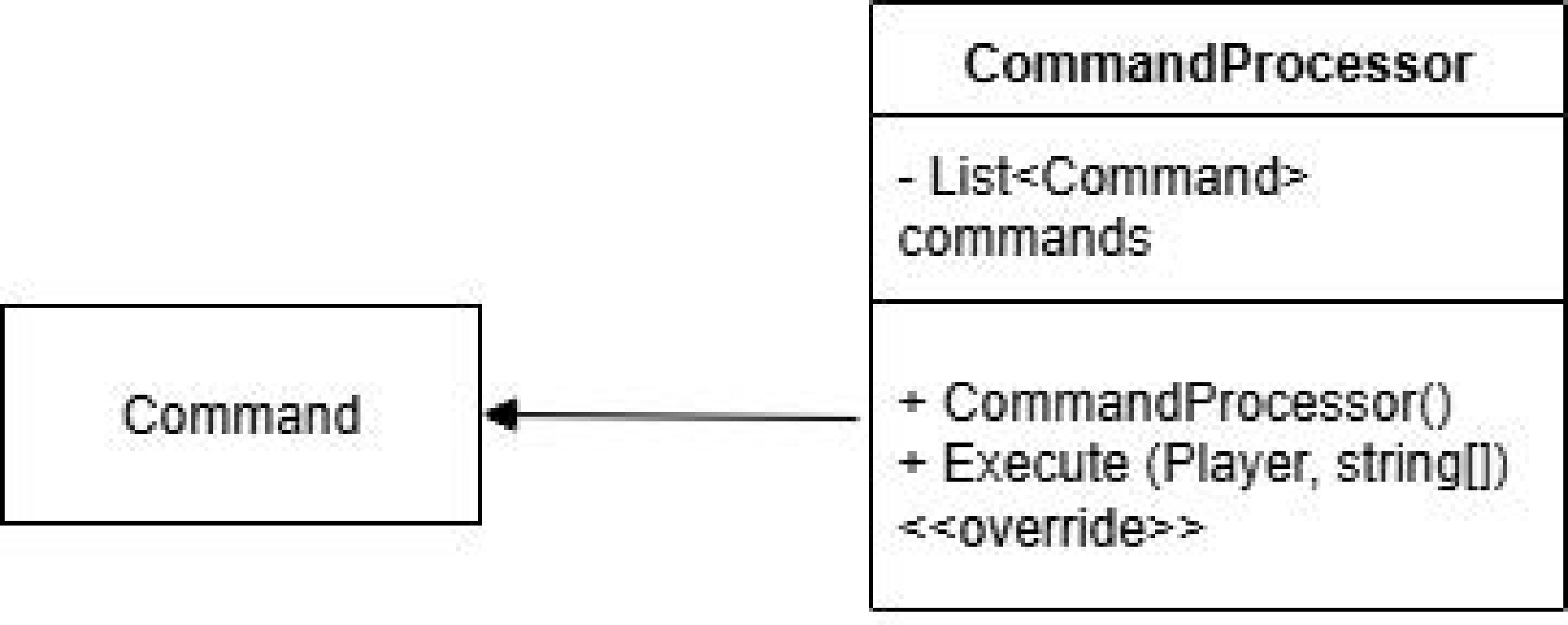
```
1  namespace SwinAdventure;
2
3  public class MoveCommandTest
4  {
5      Location start;
6      Location end;
7      Path north;
8      Player player;
9      MoveCommand moveCommand;
10
11     [SetUp]
12     public void Setup()
13     {
14         start = new Location(["duytan"], "80 Duy Tan", "Innovation Space");
15         end = new Location(["vovinam"], "3.1 VOV", "Martial Art");
16         north = new Path(["north"], "north move", "go through the north
             forrest", end);
17         player = new Player("TestPlayer", "A test player");
18         player.Location = start;
19         moveCommand = new MoveCommand();
20     }
21
22     [Test]
23     public void TestValidMove()
24     {
25         start.AddPath(north);
26         string result = moveCommand.Execute(player, ["move", "north"]);
27         string expect = $"You move north to {end.Name}";
28         Assert.That(result, Is.EqualTo(expect));
29     }
30
31     [Test]
32     public void TestInvalidMove()
33     {
34         string result = moveCommand.Execute(player, ["move", "south"]);
35         string expect = "There is no path in that direction.";
36         Assert.That(result, Is.EqualTo(expect));
37     }
38
39     [Test]
40     public void TestNonPath()
41     {
42         Item item = new Item(["north"], "a test item", "this is a test
             item");
43         start.Inventory.Put(item);
44         string result = moveCommand.Execute(player, ["move", "north"]);
45         string expect = "That doesn't seem like a valid path.";
46         Assert.That(result, Is.EqualTo(expect));
47     }
```

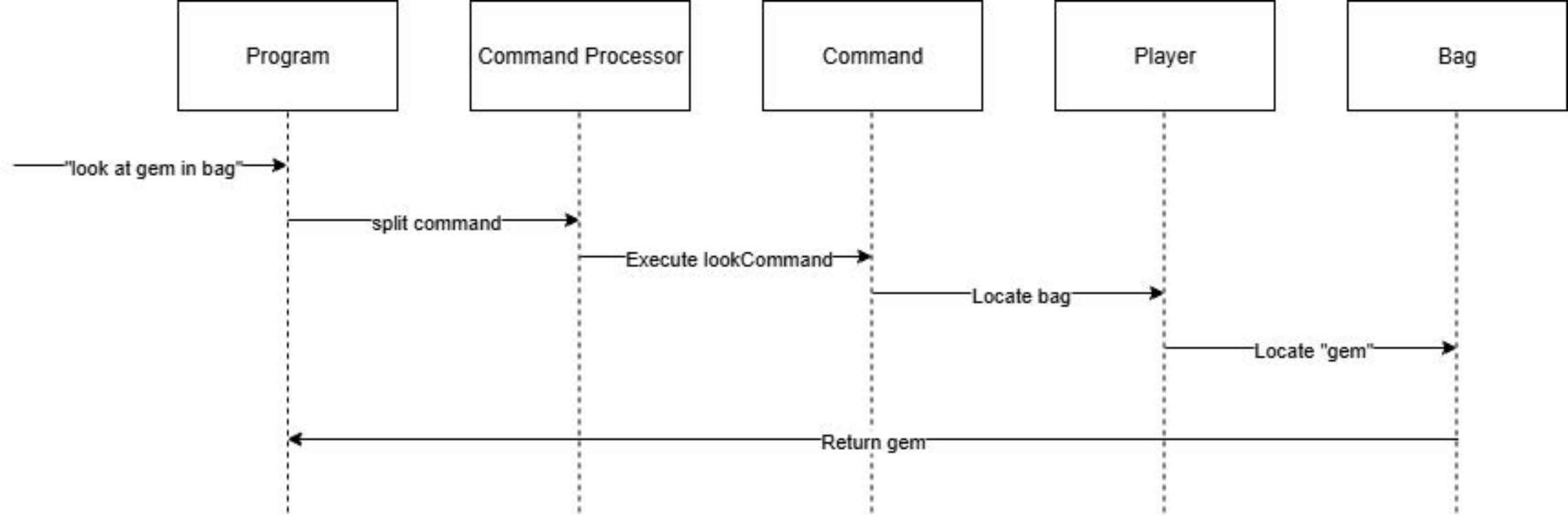
```
48
49      [Test]
50      public void TestInvalidCommand()
51      {
52          string result = moveCommand.Execute(player, ["move"]);
53          string expect = "I don't know how to move like that";
54          Assert.That(result, Is.EqualTo(expect));
55      }
56  }
57
```

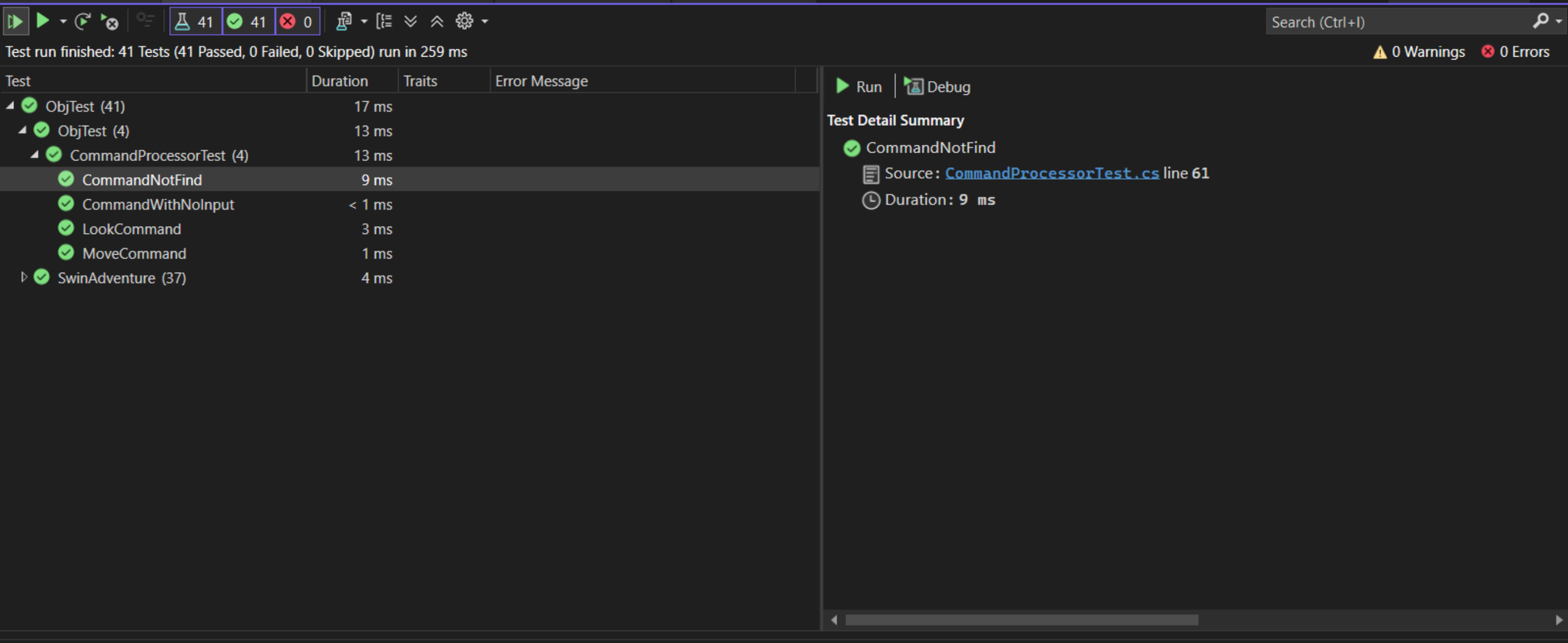
```
1  namespace SwinAdventure
2  {
3      public class PlayerTest
4      {
5          Item _item;
6          Inventory _inventory;
7          Player _swinburneStudent;
8
9          [SetUp]
10         public void Setup()
11         {
12             _item = new(new String[] { "sword" }, "diamond sword", "can
                   destroy enemies");
13             _inventory = new Inventory();
14             _swinburneStudent = new Player("Duc Manh", "OOP Student");
15         }
16
17         [Test]
18         public void PlayerIsIdentifiableTest()
19         {
20             Assert.Multiple(() =>
21             {
22                 Assert.That(_swinburneStudent.AreYou("me"), Is.True);
23                 Assert.That(_item.AreYou("sword"), Is.True);
24             });
25         }
26
27         [Test]
28         public void PlayerLocatesItemsTest()
29         {
30             _swinburneStudent.Inventory.Put(_item);
31             Assert.That(_swinburneStudent.Locate(_item.FirstId), Is.EqualTo
                   (_item));
32         }
33
34         [Test]
35         public void PlayerLocatesItselfTesr()
36         {
37             Assert.That(_swinburneStudent, Is.EqualTo
                   (_swinburneStudent.Locate("me")));
38             Assert.That(_swinburneStudent, Is.EqualTo
                   (_swinburneStudent.Locate("inventory")));
39         }
40
41         [Test]
42         public void PlayerLocatesNothingTest()
43         {
44             Assert.That(_swinburneStudent.Locate("shield"), Is.EqualTo
                   (null));
```

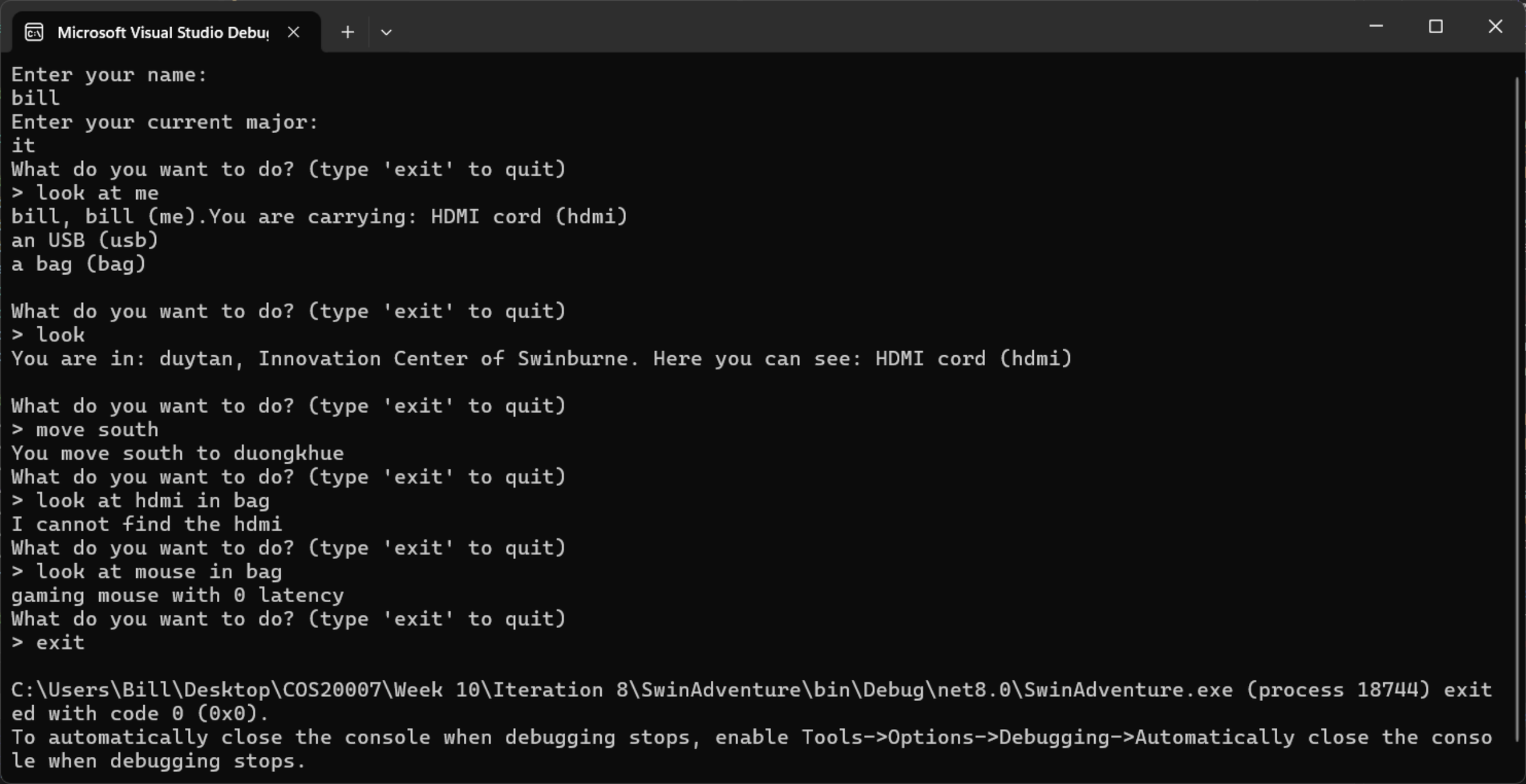
```
45          }
46
47          [Test]
48          public void PlayerFullDescriptionTest()
49          {
50              string expectedOutput = "Duc Manh, Duc Manh (me).You are
                  carrying: diamond sword (sword)\n";
51              _swinburneStudent.Inventory.Put(_item);
52              Assert.That(expectedOutput, Is.EqualTo
                  (_swinburneStudent.FullDescription));
53          }
54      }
55  }
56
```

UML, Sequence, Test Explorer and Program screenshot


CommandProcessor
- List<Command>
commands
Command
+ CommandProcessor()
+ Execute (Player, string[])
<<override>>


Program
Command Processor
Command
Player
Bag
"look at gem in bag"
split command
Execute lookCommand
Locate bag
Locate "gem"
Return gem


41
41
0
Search (Ctrl+I)
Test run finished: 41 Tests (41 Passed, 0 Failed, 0 Skipped) run in 259 ms
0 Warnings
0 Errors
Test
Duration
Traits
Error Message
ObjTest (41) 17 ms
ObjTest (4) 13 ms
CommandProcessorTest (4) 13 ms
CommandNotFind 9 ms
CommandWithNoInput < 1 ms
LookCommand 3 ms
MoveCommand 1 ms
SwinAdventure (37) 4 ms
Run
Debug
Test Detail Summary
CommandNotFind
Source: CommandProcessorTest.cs line 61
Duration: 9 ms


Microsoft Visual Studio Debu
Enter your name:
bill
Enter your current major:
it
What do you want to do? (type 'exit' to quit)
> look at me
bill, bill (me).You are carrying: HDMI cord (hdmi)
an USB (usb)
a bag (bag)
What do you want to do? (type 'exit' to quit)
> look
You are in: duytan, Innovation Center of Swinburne. Here you can see: HDMI cord (hdmi)
What do you want to do? (type 'exit' to quit)
> move south
You move south to duongkhue
What do you want to do? (type 'exit' to quit)
> look at hdmi in bag
I cannot find the hdmi
What do you want to do? (type 'exit' to quit)
> look at mouse in bag
gaming mouse with 0 latency
What do you want to do? (type 'exit' to quit)
> exit
C:\Users\Bill\Desktop\COS20007\Week 10\Iteration 8\SwinAdventure\bin\Debug\net8.0\SwinAdventure.exe (process 18744) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.