```
1  using System;
2  using System.Collections.Generic;
3  using System.Data.Common;
4  using System.Linq;
5  using System.Net.NetworkInformation;
6  using System.Runtime.CompilerServices;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Swin_Adventure
11 {
12     public class CommandProcessor
13     {
14         private List<Command> _commands = new List<Command>();
15
16         public CommandProcessor()
17         {
18             _commands.Add(new Look());
19             _commands.Add(new Move());
20         }
21
22         public string Execute(Player p, string[] text)
23         {
24             // Check every command that is available in _commands list
25             foreach (Command cmd in _commands)
26             {
27                 if (cmd.AreYou(text[0].ToLower()))
28                 {
29                     return cmd.Execute(p, text);
30                 }
31             }
32
33             // Scuffed help command.
34             if (text[0].ToLower() == "help")
35             {
36                 return Help;
37             }
38
39             return "Invalid command. Enter 'help' to see the list of
                 available commands.";
40         }
41
42         private string Help
43         {
44             get
45             {
46                 string helpString = "The list of the available commands:\n
                     \t-help\n";
47
```

```
48                  foreach (Command cmd in _commands)
49                  {
50                      helpString += String.Format("\t-{0}\n", cmd.FirstID);
51                  }
52
53                  return helpString;
54
55              }
56          }
57      }
58  }
59
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Swin_Adventure;
7  using NUnit.Framework;
8  using Path = Swin_Adventure.Path;
9
10 namespace SwinAdventureTests
11 {
12     public class CommandProcessorTests
13     {
14         CommandProcessor _processor;
15         Player _player;
16         Location _location, _destination;
17         Path _path;
18         Move _move;
19
20         [SetUp]
21         public void Setup()
22         {
23             _processor = new CommandProcessor();
24
25
26             _move = new Move();
27             // Setup player and their location
28             _player = new Player("Jacky", "Level 1 Sprout");
29             _location = new Location(new string[] { "town", "starter" },
30                         "Starter Town",
31                         "The town that every new adventurer starts in.");
32             _player.Location = _location;
33
34             // Setup path and its destination
35             _destination = new Location(new string[] { "forest", "slime" },
36                         "Slime Forest",
37                         "Forest full of slimes for beginners.");
38             _path = new Swin_Adventure.Path(new string[] { "north", "up" }, ↵
                   _destination);
39
40             // Put Path in player's location
41             _player.Location.AddPath(_path);
42         }
43
44
45
46         [Test]
47         public void TestLookAndMove()
48         {
```

```
49              string expected = String.Format("Current Location: {0}.\n\t{1}
                  \n\n{2}",
50                              _player.Location.ShortDescription,
51                              _player.Location.FullDescription,
52                              _player.Location.ShowPaths(_player)); ;
53              string actual = _processor.Execute(_player, new string[]
                  { "look" });
54              Assert.That(expected, Is.EqualTo(actual));
55
56
57              expected = "Jacky moved to Slime Forest.";
58              actual = _processor.Execute(_player, new string[] { "move",
                  "north" });
59              Assert.That(actual, Is.EqualTo(expected));
60          }
61
62          [Test]
63          public void TestHelpCommand()
64          {
65              string expected = "The list of the available commands:\n\t-help
                  \n\t-look\n\t-move\n";
66              string actual = _processor.Execute(_player, new string[]
                  { "help" });
67              Assert.That(actual, Is.EqualTo(expected));
68          }
69
70          [Test]
71          public void TestInvalidCommands()
72          {
73              string expected = "Invalid command. Enter 'help' to see the
                  list of available commands.";
74              string actual = _processor.Execute(_player, new string[]
                  { "hello" });
75              Assert.That(actual, Is.EqualTo(expected));
76
77              actual = _processor.Execute(_player, new string[] { "" });
78              Assert.That(actual, Is.EqualTo(expected));
79          }
80      }
81  }
82
```

```
using System;

namespace Swin_Adventure // Note: actual namespace depends on the project name.
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // Create a player
            Player thePlayer = CreatePlayer();

            // Add starting items for player
            InitialiseInventory(thePlayer);

            // Create Locations & Paths for the player
            InitialiseLocations(thePlayer);

            // Introduction message
            Console.WriteLine("\n" + thePlayer.FullDescription);
            Console.WriteLine(String.Format("\nCurrent Location: {0}\n\t{1}", thePlayer.Location.ShortDescription, thePlayer.Location.FullDescription));
            Console.WriteLine("\nEnter 'exit' to quit your adventure.");

            // Create CommandProcessor
            CommandProcessor cmdProcessor = new CommandProcessor();

            // Keep getting commands from the user
            while (true)
            {
                Console.Write("{0}>",thePlayer.Name);
                string[] userInput = Console.ReadLine().Trim().Split();

                if (userInput.Contains("exit"))
                {
                    break;
                }

                Console.WriteLine(cmdProcessor.Execute(thePlayer, userInput));
            }
        }

        private static Player CreatePlayer()
        {
            Console.Write("Enter your name: ");
            string playerName = Console.ReadLine().Trim();
            Console.Write("Enter a description: ");
```

```
46              string playerDescription = Console.ReadLine().Trim();
47
48              // Create player
49              return new Player(playerName, playerDescription);
50          }
51
52          private static void InitialiseInventory(Player p)
53          {
54              // Create items and place them in the player's inventory
55              Item itemSword = new Item(new string[] { "sword" }, "Bronze
                  Sword", "A sword made out of bronze");
56              Item itemWater = new Item(new string[] { "water" }, "Water",
                  "You can drink this.");
57              p.Inventory.Put(itemSword);
58              p.Inventory.Put(itemWater);
59
60              // Create bag and place them in the player's inventory
61              Bag bagStarterBag = new Bag(new string[] { "bag" }, "Starter
                  Bag", "A bag for all new adventurers.");
62              p.Inventory.Put(bagStarterBag);
63              // Create bow and place in the bag
64              Item itemBow = new Item(new string[] { "bow" }, "Wooden Bow",
                  "A bow fit for beginners");
65              bagStarterBag.Inventory.Put(itemBow);
66          }
67
68          private static void InitialiseLocations(Player p)
69          {
70              // Starting Location
71              Location _location1 = new Location(new string[] { "town",
                  "starter" }, "Starter Town", "The town that every new
                  adventurer starts in.");
72              _location1.Inventory.Put(new Item(new string[] { "cabbage" },
                  "Cabbage", "It doesn't look very appetising."));
73
74              // Create second location and add it as a path to starting
                  location.
75              Location _location2 = new Location(new string[] { "forest",
                  "slime" }, "Slime forest", "You are in the Slime forest.
                  Every adventurer has to start somewhere.");
76              _location2.Inventory.Put(new Item(new string[] { "potion" },
                  "Red Potion", "A small red potion. Restores a meager amount
                  of health."));
77              // Create Path and add it to the player
78              Path _path1 = new Path(new string[] { "north", "up" },
                  _location2);
79              _location1.AddPath(_path1);
80              p.Location = _location1;
81
```

```
82              // Setup 2nd path and its destination
83              Location _destination2 = new Location(new string[] { "swamp" },
84                          "Murky Swamp",
85                          "Why is there a dangerous looking swamp next to the
                        town?");
86              Path _path2 = new Swin_Adventure.Path(new string[] { "south",
                  "down" }, _destination2);
87              // Put Path in player's location
88              p.Location.AddPath(_path2);
89          }
90      }
91  }
```