Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 8 - Command Processor

PDF generated at 22:05 on Monday 20th November, 2023

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace SwinAdventure
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to SwinAdventure!");

            Console.Write("Enter your player's name: ");
            string playerName = Console.ReadLine();
            Console.Write("Enter a description for your player: ");
            string playerDescription = Console.ReadLine();
            Player player = new Player(playerName, playerDescription);

            Item sword = new Item(new string[] { "sword" }, "bronze sword", "This is a mighty fine sword.");
            Item potion = new Item(new string[] { "potion" }, "healing potion", "A magical potion that heals wounds.");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful gemstone.");
            Bag bag = new Bag(new string[] { "bag" }, "small bag", "A small bag.");
            player.Inventory.Put(bag);
            player.Inventory.Put(sword);
            player.Inventory.Put(potion);
            player.Inventory.Put(gem);
            bag.Inventory.Put(gem);

            Location startingLocation = new Location(new string[] { "forest" }, "Dark Forest", "A mysterious and dark forest.");
            player.Location = startingLocation;
            startingLocation.Inventory.Put(sword);
            startingLocation.Inventory.Put(potion);
            startingLocation.Inventory.Put(gem);
            startingLocation.Inventory.Put(bag);

            //Add locations
            Location gaza = new Location(new string[] { "gaza" }, "Gaza", "This is the largest city in the State of Palestine");
            Location westbank = new Location(new string[] { "westbank" }, "West Bank", "This is a biggest territory of Palestine occupied by Israel");
            Location jerusalem = new Location(new string[] { "jerusalem" }, "Jerusalem", "This is Holy land");

            //Setup the paths
            Path pathHtoL = new Path(new string[] { "south", "s", "down" }, "South", "Al-Aqsa Mosque", jerusalem);
            Path pathHtoG = new Path(new string[] { "east", "e", "right" }, "East", "Catholic Church", westbank);

```

```
45              Path pathGtoH = new Path(new string[] { "west", "w", "left" }, "West",
    ↪ "Al Shi-fa", gaza);
46              Path pathGtoL = new Path(new string[] { "sw", "south_west" }, "South
    ↪ West", "Tower of David", jerusalem);
47
48              Path pathLtoH = new Path(new string[] { "n", "north", "up" }, "North",
    ↪ "Al Deira Hotel", gaza);
49              Path pathLtoG = new Path(new string[] { "ne", "north_east" }, "North
    ↪ East", "King David Hotel", westbank);
50
51              //Add the paths to each location
52              gaza.AddPath(pathHtoG);
53              gaza.AddPath(pathHtoL);
54
55              westbank.AddPath(pathGtoL);
56              westbank.AddPath(pathGtoH);
57
58              jerusalem.AddPath(pathLtoG);
59              jerusalem.AddPath(pathLtoH);
60
61              player.Location = gaza;
62
63              string input;
64              do
65              {
66                  Console.Write("Enter a command or type 'exit' to quit: ");
67                  input = Console.ReadLine();
68
69                  if (input.Equals("exit", StringComparison.OrdinalIgnoreCase))
70                      break;
71
72                  string[] commandParts = input.Split(' ');
73                  if (commandParts.Length >= 3 && commandParts[0].Equals("add",
    ↪ StringComparison.OrdinalIgnoreCase) && commandParts[2].Equals("to",
    ↪ StringComparison.OrdinalIgnoreCase))
74                  {
75                      string itemName = commandParts[1];
76                      string containerName = commandParts[3];
77
78                      GameObject item = player.Locate(itemName);
79                      GameObject container = player.Locate(containerName);
80
81                      if (item != null && container != null)
82                      {
83                          if (container is IHaveInventory)
84                          {
85                              IHaveInventory containerWithInventory = container as
    ↪ IHaveInventory;
86                              containerWithInventory.Inventory.Put(item);
87                              Console.WriteLine($"{item.Name} has been added to
    ↪ {container.Name}.");
88                          }
89                          else
```

```
90                          {
91                              Console.WriteLine("Could not perform the action.");
92                          }
93                      }
94                  }
95                  else if (commandParts.Length >= 2 && commandParts[0].Equals("look",
    ↪ StringComparison.OrdinalIgnoreCase))
96                  {
97                      string target = commandParts[1];
98                      if (target.Equals("location",
    ↪ StringComparison.OrdinalIgnoreCase))
99                      {
100                         if (player.Location != null)
101                         {
102                             Console.WriteLine(player.Location.FullDescription);
103                             Console.WriteLine("Items in the location:");
104 
105                             foreach (var item in player.Location.Inventory.itemList)
106                             {
107                                 if (item is Item)
108                                 {
109                                     Item inventoryItem = (Item)item;
110                                     Console.WriteLine("\t" +
    ↪ inventoryItem.ShortDescription);
111                                 }
112                             }
113                             Console.WriteLine(Environment.NewLine);
114                         }
115                         else
116                         {
117                             Console.WriteLine("Player has no location.");
118                         }
119                     }
120                     else
121                     {
122                         string result = player.Locate(target)?.FullDescription ?? "I
    ↪ can't find that.";
123                         Console.WriteLine(result);
124                     }
125                 }
126                 else if (commandParts.Length >= 1 && (commandParts[0].Equals("move",
    ↪ StringComparison.OrdinalIgnoreCase) ||
127                                                       commandParts[0].Equals("go",
    ↪ StringComparison.OrdinalIgnoreCase) ||
128                                                       commandParts[0].Equals("head",
    ↪ StringComparison.OrdinalIgnoreCase) ||
129                                                       commandParts[0].Equals("leave",
    ↪ StringComparison.OrdinalIgnoreCase)))
130                 {
131                     Command moveCommand = new MoveCommand();
132                     string result = moveCommand.Execute(player, commandParts);
133                     Console.WriteLine(result);
134                 }
```

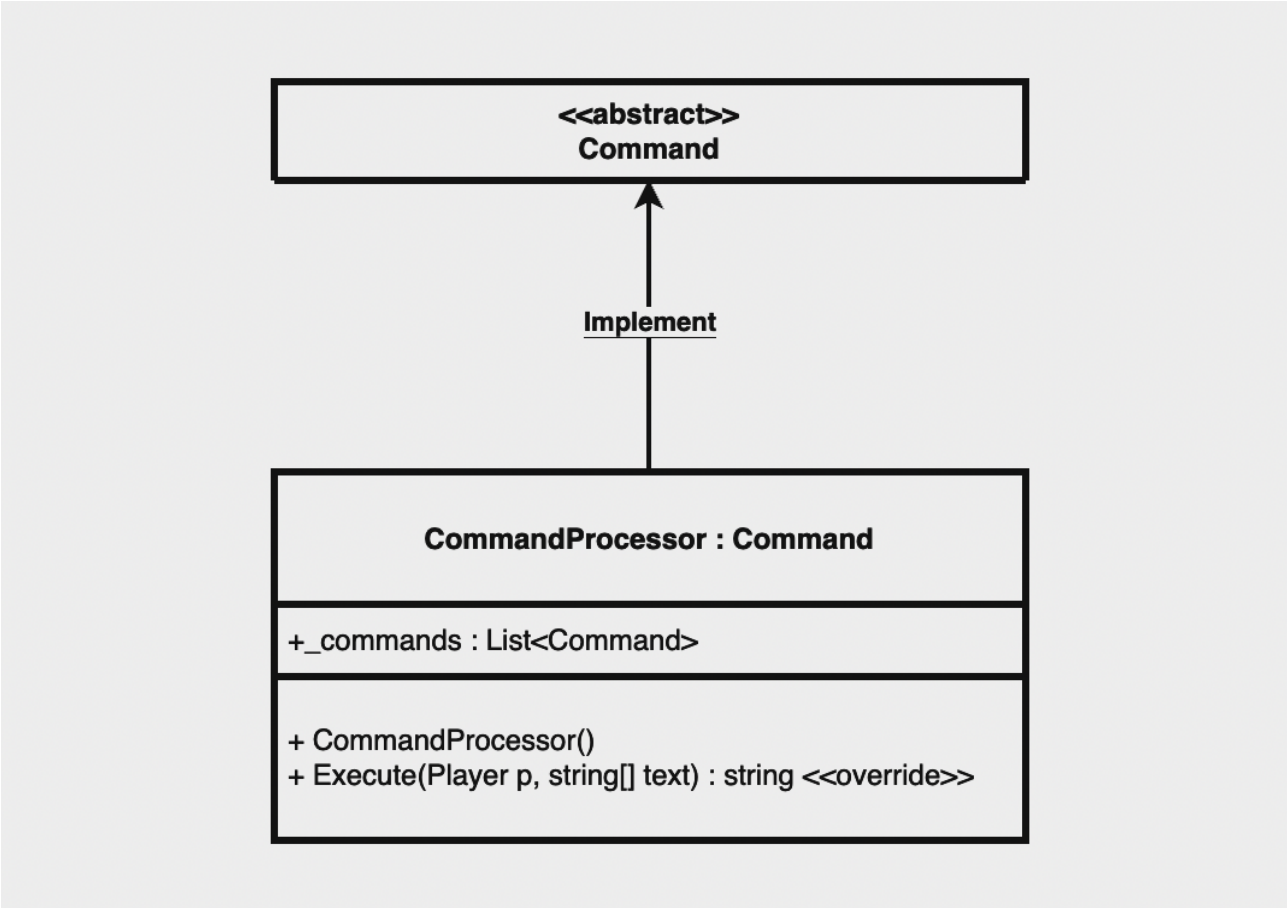
```
135                 else
136                 {
137                     Console.WriteLine("I don't understand that command.");
138                 }
139             } while (true);
140         }
141     }
142 }
```

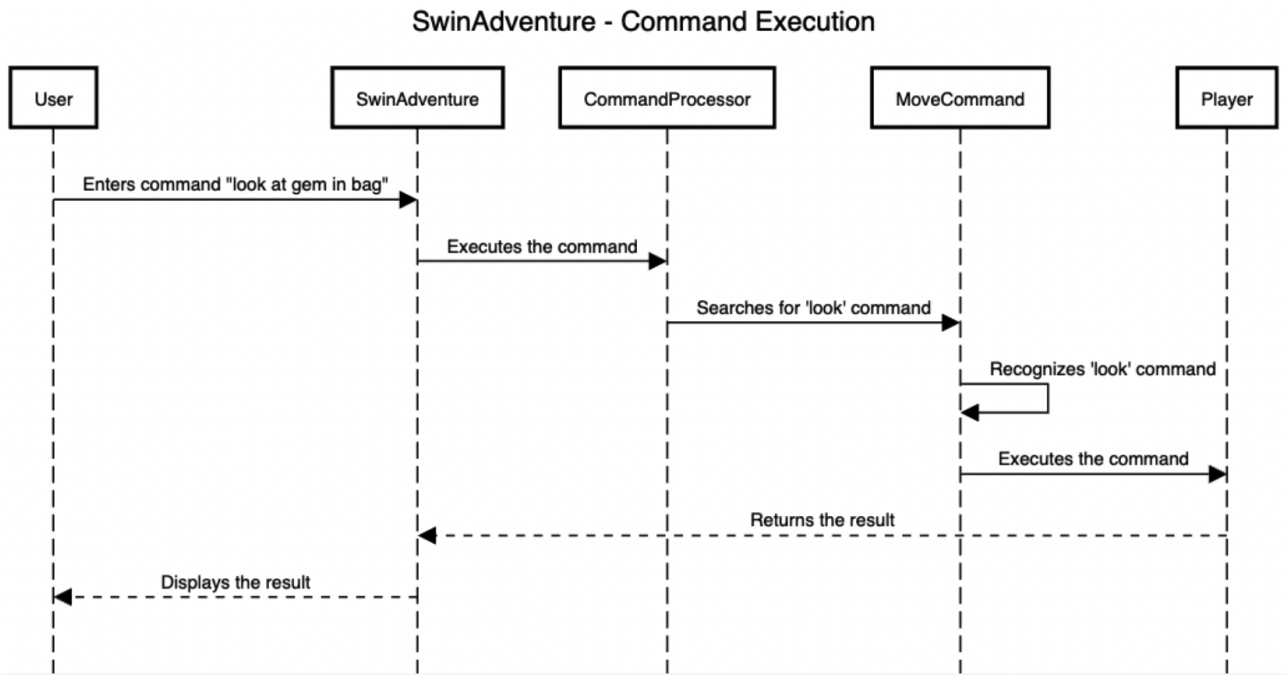
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class CommandProcessor : Command
    {
        private List<Command> _commands = new List<Command>();

        private LookCommand _lookCommand = new LookCommand();
        private MoveCommand _moveCommand = new MoveCommand();

        public CommandProcessor() : base(new string[] { "command" })
        {
            _commands.Add(_lookCommand);
            _commands.Add(_moveCommand);
        }

        public override string Execute(Player p, string[] text)
        {
            foreach (Command c in _commands)
            {
                if (c.AreYou(text[0]))
                {
                    return c.Execute(p, text);
                }
            }

            string search = "";

            foreach (string txt in text)
            {
                search = search + txt + " ";
            }

            return "I don't understand " + search.TrimEnd();
        }
    }
}
```

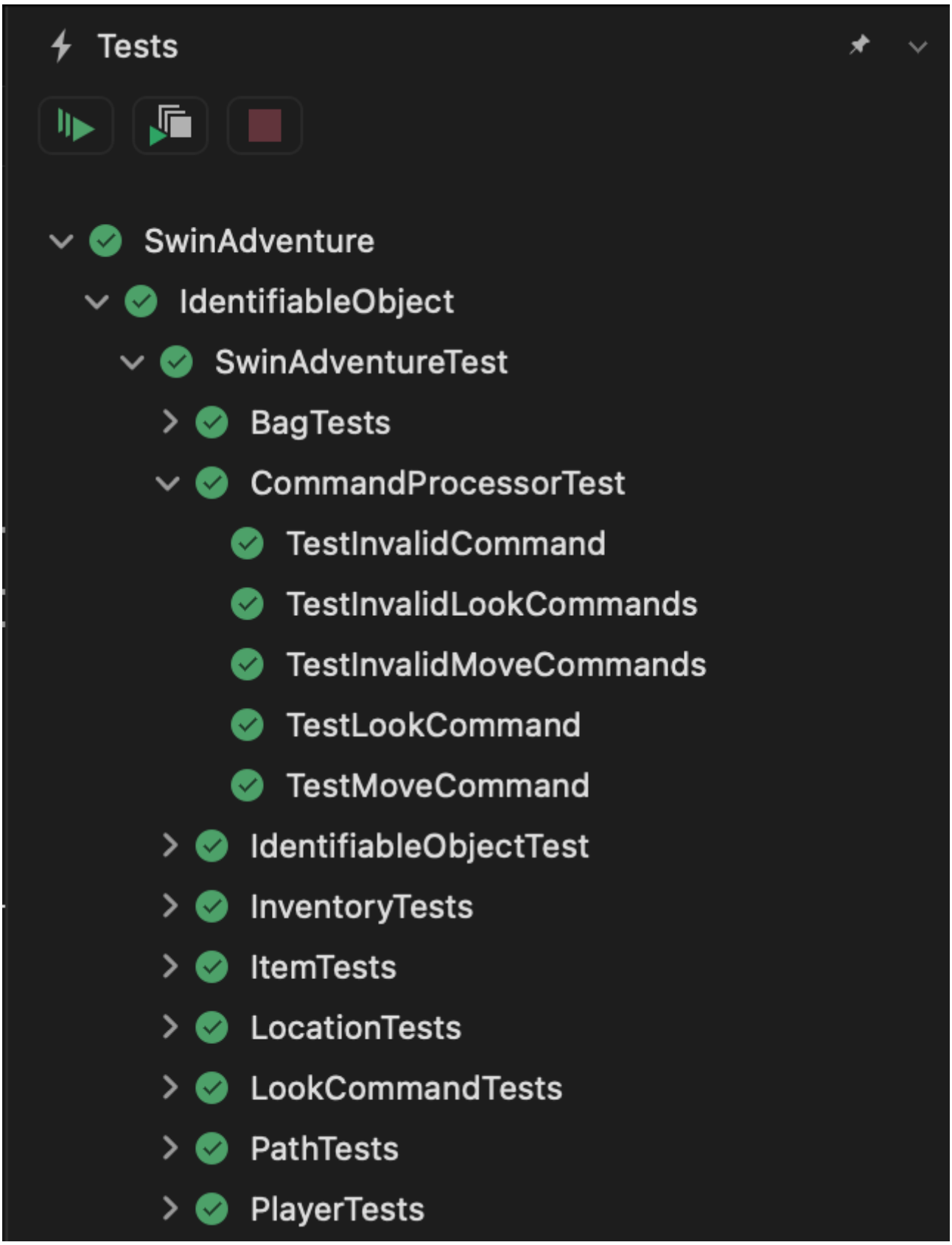
```
using System;
using SwinAdventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class CommandProcessorTest
    {
        Player player;
        Location hall, garden, lab;
        SwinAdventure.Path pathHtoL, pathHtoG, pathGtoH, pathGtoL, pathLtoH,
↪   pathLtoG;
        CommandProcessor command;
        Item shovel, sword, gem, pc;
        Bag b1, b2;

        [SetUp]
        public void Setup()
        {
            player = new Player("Marella", "The amazing player");
            hall = new Location(new string[] { "hallway" }, "Hallway", "This is a
↪   long well lit Hallway");
            garden = new Location(new string[] { "garden" }, "Garden", "This is a big
↪   garden with a lot of secret spots");
            lab = new Location(new string[] { "lab" }, "Laboratory", "This is where
↪   the magic is created");

            pathHtoL = new SwinAdventure.Path(new string[] { "south", "s", "down" },
↪   "South", "slide", lab);
            pathHtoG = new SwinAdventure.Path(new string[] { "east", "e" }, "East",
↪   "small door", garden);

            pathGtoH = new SwinAdventure.Path(new string[] { "west", "w" }, "West",
↪   "small door", hall);
            pathGtoL = new SwinAdventure.Path(new string[] { "sw", "south west" },
↪   "South West", "roller coaster", lab);

            pathLtoH = new SwinAdventure.Path(new string[] { "n", "north", "up" },
↪   "North", "ladder", hall);
            pathLtoG = new SwinAdventure.Path(new string[] { "ne", "north west" },
↪   "North West", "roller coaster", garden);

            hall.AddPath(pathHtoG);
            hall.AddPath(pathHtoL);

            garden.AddPath(pathGtoL);
            garden.AddPath(pathGtoH);

            lab.AddPath(pathLtoG);
            lab.AddPath(pathLtoH);

            player.Location = hall;

```

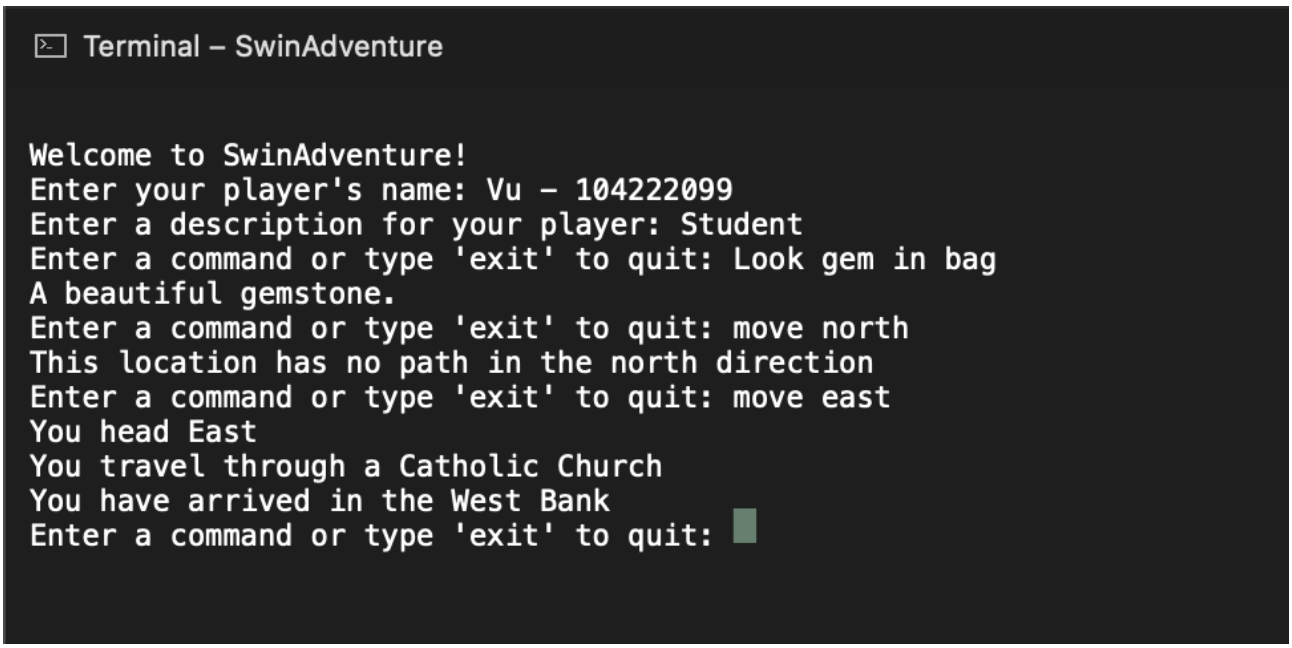
```
44              shovel = new Item(new string[] { "shovel" }, "shovel", "This is a might
    ↪ fine shovel");
45              sword = new Item(new string[] { "sword" }, "bronze sword", "This is a
    ↪ shiny sword");
46              gem = new Item(new string[] { "gem" }, "red gem", "This is a shiny red
    ↪ gem");
47              pc = new Item(new string[] { "pc" }, "small computer", "This is a
    ↪ computer from the future");
48
49              b1 = new Bag(new string[] { "bag" }, "plastic bag", "This is a clear
    ↪ plastic bag");
50              b2 = new Bag(new string[] { "wallet" }, "wallet", "This is a black
    ↪ wallet");
51
52              hall.Inventory.Put(sword);
53              b1.Inventory.Put(pc);
54              hall.Inventory.Put(b1);
55
56              garden.Inventory.Put(shovel);
57              garden.Inventory.Put(gem);
58
59              b2.Inventory.Put(pc);
60              lab.Inventory.Put(b2);
61
62              command = new CommandProcessor();
63          }
64
65          [Test]
66          public void TestLookCommand()
67          {
68              player.Inventory.Put(gem);
69              string expected = "This is a shiny red gem";
70              Assert.AreEqual(expected, command.Execute(player, new string[] { "look",
    ↪ "at", "gem" }), "Test look command failed");
71          }
72
73          [Test]
74          public void TestMoveCommand()
75          {
76              string expected = "You head South\nYou travel through a slide\nYou have
    ↪ arrived in the Laboratory";
77              Assert.AreEqual(expected, command.Execute(player, new string[] { "move",
    ↪ "south" }), "Player cannot be moved south");
78          }
79
80          [Test]
81          public void TestInvalidCommand()
82          {
83              Assert.AreEqual(command.Execute(player, new string[] { "dive", "left" }),
    ↪ "I don't understand dive left");
84          }
85
86          [Test]
```

```
87          public void TestInvalidMoveCommands()
88          {
89              //invalid direction
90              Assert.AreEqual("This location has no path in the somewhere direction",
    ↪  command.Execute(player, new string[] { "move", "somewhere" }));
91              Assert.AreEqual(hall, player.Location, "Player left somewhere...");
92
93              command.Execute(player, new string[] { "go", "east" });
94              command.Execute(player, new string[] { "head", "south west" });
95
96              Assert.AreEqual("This location has no path in the south direction",
    ↪  command.Execute(player, new string[] { "move", "south" })); //cannot move south
    ↪  from the lab
97              Assert.AreEqual("Where do you want to go?", command.Execute(player, new
    ↪  string[] { "move" }));
98          }
99
100         [Test]
101         public void TestInvalidLookCommands()
102         {
103             string expected = "Look at what?";
104             Assert.AreEqual(expected, command.Execute(player, new string[] { "look",
    ↪  "at", "gem", "in" }), "Test text length = 4 not accepted Failed");
105
106             expected = "I don't understand like at gem";
107             Assert.AreEqual(expected, command.Execute(player, new string[] { "like",
    ↪  "at", "gem" }), "Test look word not found Failed");
108
109             expected = "Look at what?";
110             Assert.AreEqual(expected, command.Execute(player, new string[] { "look",
    ↪  "int", "gem" }), "Test look at Failed");
111
112             expected = "Look at what?";
113             Assert.AreEqual(expected, command.Execute(player, new string[] { "look",
    ↪  "at", "gem", "on", "bag" }), "Test look at in Failed");
114         }
115     }
116 }
117
```


<<abstract>>
Command
Implement
CommandProcessor : Command
+_commands : List<Command>
+ CommandProcessor()
+ Execute(Player p, string[] text) : string <<override>>


SwinAdventure - Command Execution
User
SwinAdventure
CommandProcessor
MoveCommand
Player
Enters command "look at gem in bag"
Executes the command
Searches for 'look' command
Recognizes 'look' command
Executes the command
Returns the result
Displays the result


Tests
SwinAdventure
IdentifiableObject
SwinAdventureTest
BagTests
CommandProcessorTest
TestInvalidCommand
TestInvalidLookCommands
TestInvalidMoveCommands
TestLookCommand
TestMoveCommand
IdentifiableObjectTest
InventoryTests
ItemTests
LocationTests
LookCommandTests
PathTests
PlayerTests


Terminal – SwinAdventure
Welcome to SwinAdventure!
Enter your player's name: Vu - 104222099
Enter a description for your player: Student
Enter a command or type 'exit' to quit: Look gem in bag
A beautiful gemstone.
Enter a command or type 'exit' to quit: move north
This location has no path in the north direction
Enter a command or type 'exit' to quit: move east
You head East
You travel through a Catholic Church
You have arrived in the West Bank
Enter a command or type 'exit' to quit: