Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 8 - Command Processor

PDF generated at 12:23 on Thursday 23rd November, 2023

```
// See https://aka.ms/new-console-template for more information
using MazeGame;
Player _player = new Player("Hoang An", "the comtemplator of infinity");
Item knife = new Item(new string[] { "knife" }, "an obsidian knife", "A hunting knife cast from obsidian");
Item axe = new Item(new string[] { "axe" }, "a stone axe", "An axe made of cobblestone");

_player.Inventory.Put(knife);
_player.Inventory.Put(axe);

Bag _bag = new Bag(new string[] { "b1" }, "leather bag", "a bag made and stiched with leather.");
_player.Inventory.Put(_bag);

Item shovel = new Item(new string[] { "shovel" }, "a shovel", "A durable shovel borrowed from the village");
_bag.Inventory.Put(shovel);

////////Locations/////
//Location 1
Location garden = new Location(new string[] { "garden" }, "green garden", "A garden blooming with natural plants, trees, and flowers");
Item water = new Item(new string[] { "water" }, "a bottled water", "A 1 Litres bottle of spring water to keep you hydrated");
Item pearl = new Item(new string[] { "pearl" }, "a pearl", "A pearl picked from pearl tree. A fruit great for snack");
garden.Inventory.Put(water);
garden.Inventory.Put(pearl);

//Location 2
Location area51 = new Location(new string[] { "area51" }, "area 51", "Special labratory for aliens");

//Location 3
Location library = new Location(new string[] { "library" }, "archive library", "area that contains old history book");
Item book = new Item(new string[] { "book" }, "a history book", "A book that captures the history of this city");
library.Inventory.Put(book);

//Location 4
Location bakery = new Location(new string[] { "bakery" }, "bakery shop", "A shop that sells freshly made breads and deserts");
Item bread = new Item(new string[] { "bread" }, "a loaf of bread", "A freshly baked loaf of white bread");
Item cake = new Item(new string[] { "cake" }, "a piece of cake", "A sweet cake. The shop's signature desert");
bakery.Inventory.Put(bread);
bakery.Inventory.Put(cake);

//////////Command Processor - Identify command type before executing/////////////
CommandProcessor processor = new CommandProcessor();
```

```
41
42  /////Each location has a number of paths that is linked to a different Location////
43  ///Paths in Garden
44  Paths gardenPath1 = new Paths(new string[] { "n" }, "north",
45      "You got in your car and travelled through the road up North", area51);
46  Paths gardenPath2 = new Paths(new string[] { "e" }, "east",
47      "You walked for a kilometer to a library in East", library);
48  garden.PathList.Add(gardenPath1);
49  garden.PathList.Add(gardenPath2);
50
51  ///Paths in Area51
52  Paths area51Path1 = new Paths(new string[] { "s" }, "south",
53      "You got in your car and travelled through the road down South", garden);
54  area51.PathList.Add(area51Path1);
55
56  ///Paths in Library
57  Paths libraryPath1 = new Paths(new string[] { "w" }, "west",
58      "You walked for a kilometer to a garden in East.", garden);
59  Paths libraryPath2 = new Paths(new string[] { "nw" }, "northwest",
60      "You crossed the road and turned left to a bakery in NorthWest", bakery);
61  library.PathList.Add(libraryPath1);
62  library.PathList.Add(libraryPath2);
63
64  ///Paths in Bakery
65  Paths bakeryPath1 = new Paths(new string[] { "ne" }, "northeast",
66      "You turned right and crossed the road to a library in NorthEast", library);
67  bakery.PathList.Add(bakeryPath1);
68
69
70  Console.WriteLine("Swin-Adventure Maze Game");
71  Console.WriteLine($"Welcome");
72  Console.WriteLine($"{_player.FullDescription}");
73  Console.WriteLine($"{_player.ChangeLocation(garden)}");
74  Console.WriteLine();
75
76  while (true)
77  {
78      Console.WriteLine("For look command, type in command 'look'");
79      Console.WriteLine("Note: The input must be either 3 or 5 words only");
80      Console.WriteLine("Example: 'look at ...' or 'look at ... in ...'\n");
81      Console.WriteLine("For move command, type in command with directions (n, e, s,
    ↪  w):\n" +
82          " 'go'\n" +
83          " 'move'\n" +
84          " 'head'\n" +
85          " 'leave'");
86      Console.WriteLine("Note: The input must be 2 words only\n");
87
88      Console.Write("Command: ");
89      string command = Console.ReadLine();
90
91      if (command == "exit" || command == "Exit")
92      {
```

```
93          Console.WriteLine();
94          Console.WriteLine("Bye Bye");
95          break;
96      }
97
98      string[] cmdArray = command.Split(' ');
99      Console.WriteLine();
100     Console.Clear();
101
102     Console.WriteLine("Output: ");
103     Console.WriteLine(processor.Execute(_player, cmdArray));
104 }
105
```

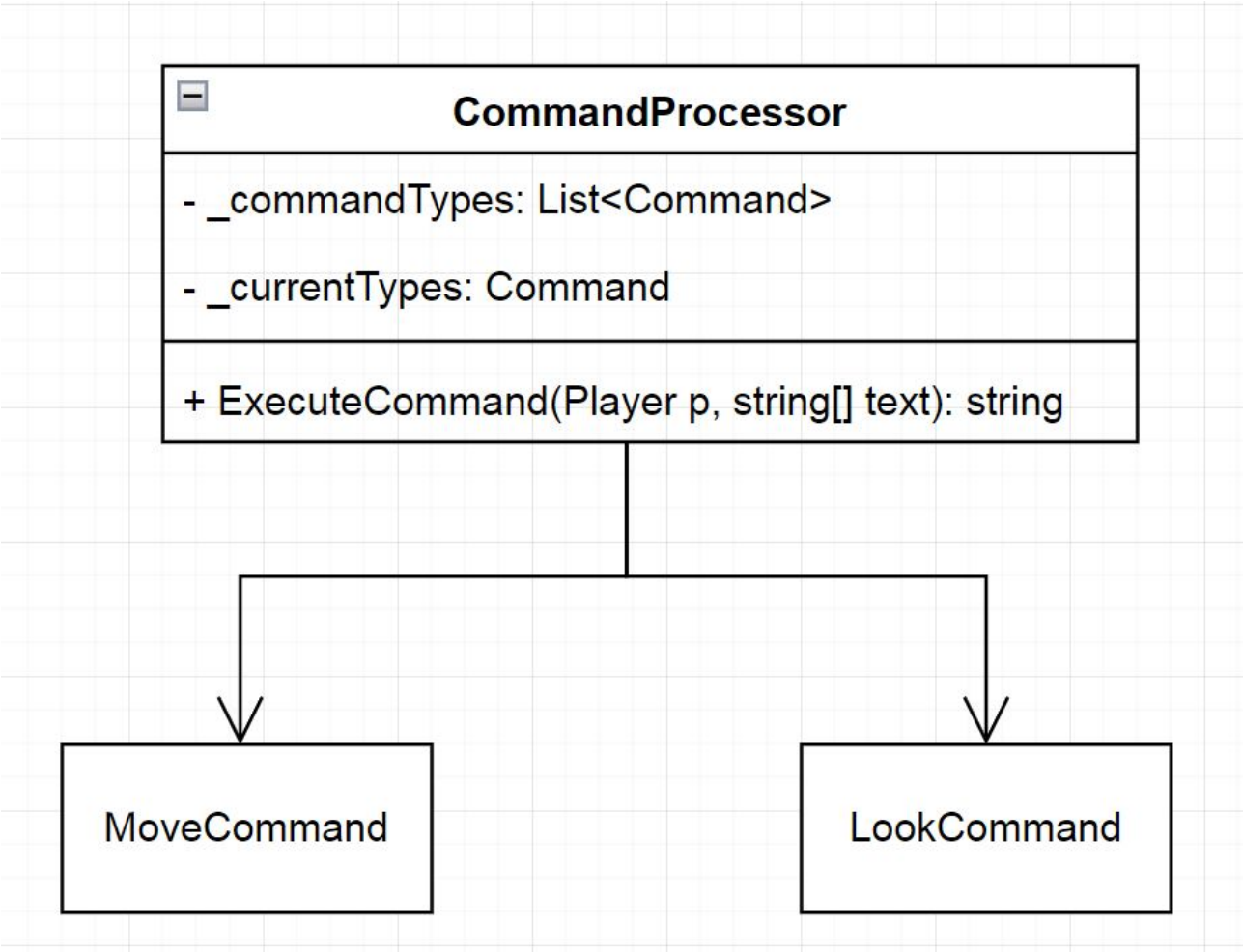
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MazeGame
{
    public class CommandProcessor
    {
        private List<Command> commands;
        private Command? _currentTypes;

        public CommandProcessor()
        {
            _currentTypes = null;

            Command look = new LookCommand(new string[] { "look" });
            Command look2 = new LookCommand(new string[] { "Look" });
            Command move = new MoveCommand(new string[] { "move" });
            Command move2 = new MoveCommand(new string[] {"go" });
            Command move3 = new MoveCommand(new string[] { "head" });
            Command move4 = new MoveCommand(new string[] { "leave" });

            commands = new List<Command>
            {
                look,
                look2,
                move,
                move2,
                move3,
                move4
            };
        }

        public string Execute(Player p, string[] text)
        {
            foreach (var command in commands)
            {
                if(string.Equals(text[0], command.FirstId,
↪   StringComparison.OrdinalIgnoreCase))
                {
                    _currentTypes = command;
                }
            }

            if (_currentTypes == null)
            {
               return "Please input a valid command type (move or look)\n";
            }
            else
            {
                return _currentTypes.Execute(p, text);
```

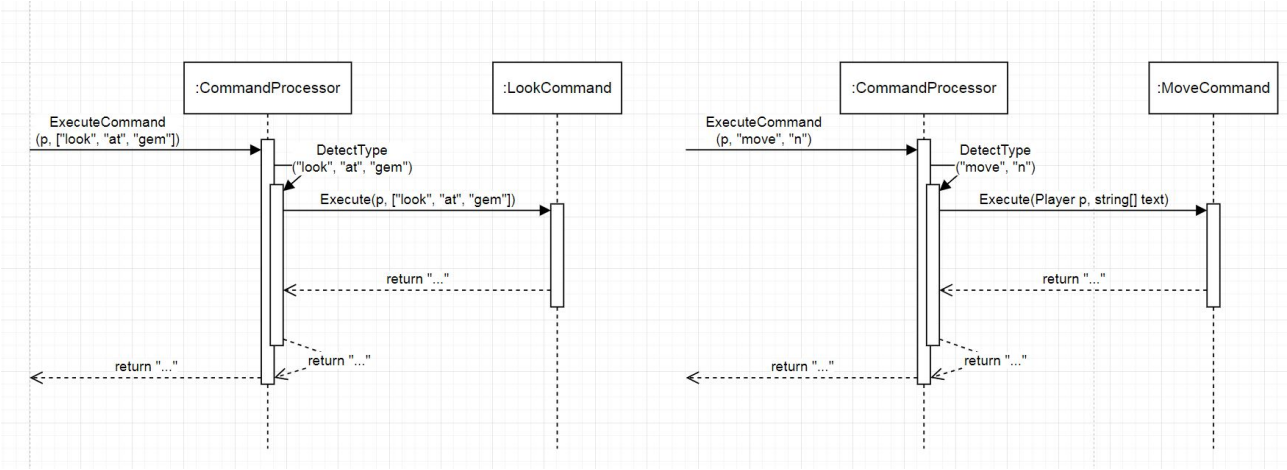
```
53                  }
54              }
55          }
56  }
```

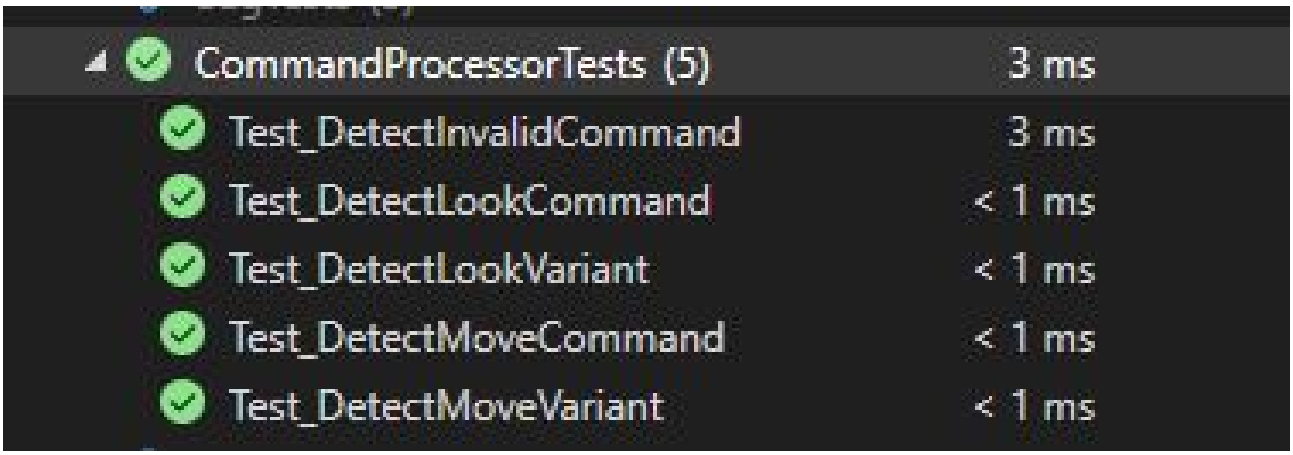
```
namespace MazeGame.nUnitTests
{
    public class CommandProcessorTests
    {
        private Command look { get; set; } = null!;
        private Command look2 { get; set; } = null!;
        private Command move { get; set; } = null!;
        private Command move2 { get; set; } = null!;
        private Command move3 { get; set; } = null!;
        private Command move4 { get; set; } = null!;
        private List<Command> commands;
        private CommandProcessor processor {  get; set; } = null!;
        //private Player _player { get; set; } = null!;

        [SetUp]
        public void SetUp()
        {
            look = new LookCommand(new string[] { "look" });
            look2 = new LookCommand(new string[] { "Look" });
            move = new MoveCommand(new string[] { "move" });
            move2 = new MoveCommand(new string[] { "go" });
            move3 = new MoveCommand(new string[] { "head" });
            move4 = new MoveCommand(new string[] { "leave" });
            commands = new List<Command>
            {
                look,
                look2,
                move,
                move2,
                move3,
                move4
            };
            processor = new CommandProcessor();
        }

        [Test]
        public void Test_DetectMoveCommand()
        {
            string input = "move";

            foreach(Command command in commands)
            {
                if(input == command.FirstId)
                {
                    var sut = true;
                    Assert.IsTrue(sut);
                    Console.WriteLine($"Search Command: {input}\n" +
                        $"Command Returned: {command.FirstId}");
                    break;
                }
            }
        }

```

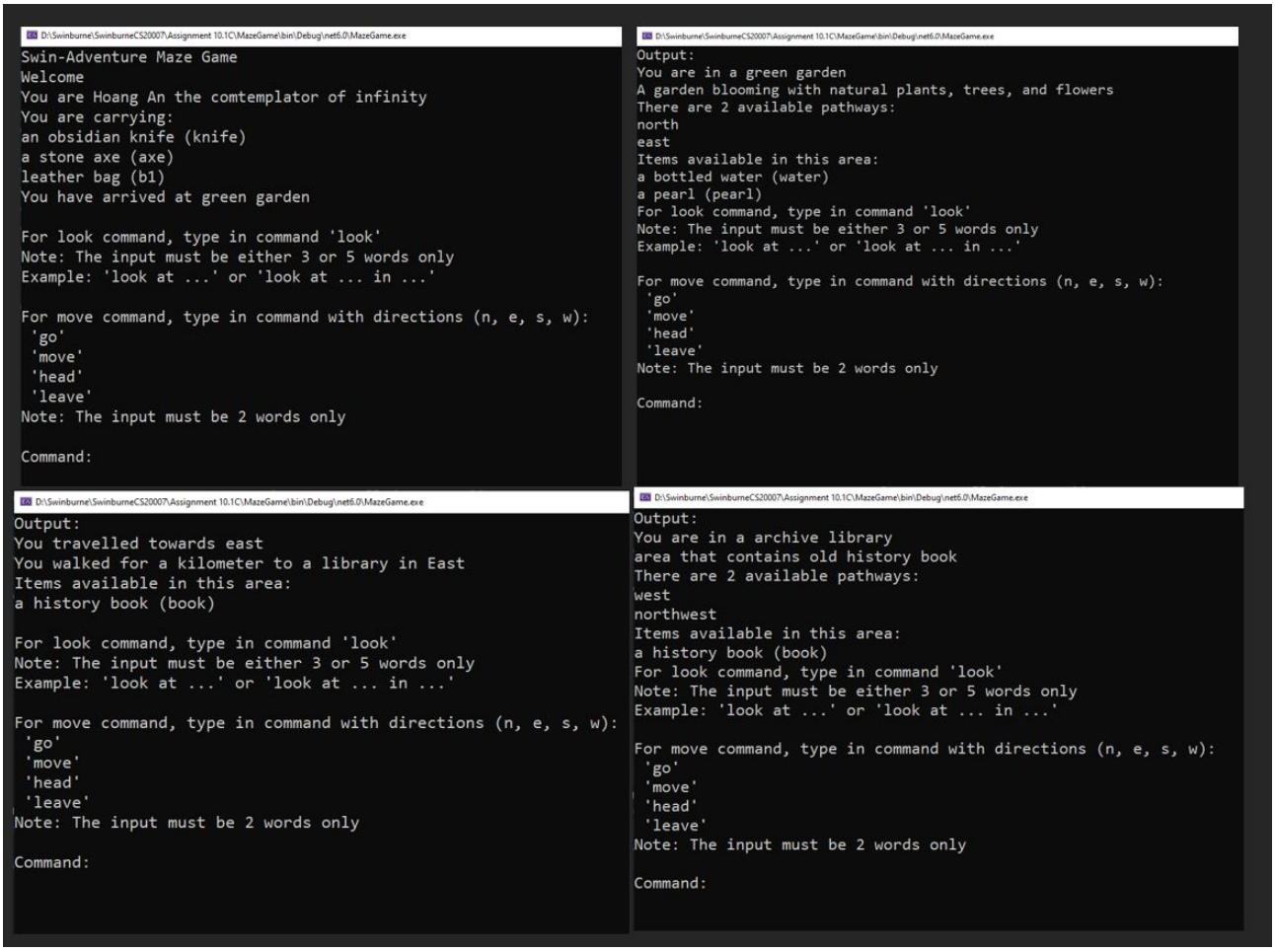
```
54          [Test]
55          public void Test_DetectMoveVariant()
56          {
57              string input = "head";
58
59              foreach (Command command in commands)
60              {
61                  if (input == command.FirstId)
62                  {
63                      var sut = true;
64                      Assert.IsTrue(sut);
65                      Console.WriteLine($"Search Command: {input}\n" +
66                          $"Command Returned: {command.FirstId}");
67                      break;
68                  }
69              }
70          }
71
72
73          [Test]
74          public void Test_DetectLookCommand()
75          {
76              string input = "look";
77
78              foreach (Command command in commands)
79              {
80                  if (input == command.FirstId)
81                  {
82                      var sut = true;
83                      Assert.IsTrue(sut);
84                      Console.WriteLine($"Search Command: {input}\n" +
85                          $"Command Returned: {command.FirstId}");
86                      break;
87                  }
88              }
89          }
90
91          [Test]
92          public void Test_DetectLookVariant()
93          {
94              string input = "Look";
95
96              foreach (Command command in commands)
97              {
98                  if (input == command.FirstId)
99                  {
100                     var sut = true;
101                     Assert.IsTrue(sut);
102                     Console.WriteLine($"Search Command: {input}\n" +
103                         $"Command Returned: {command.FirstId}");
104                     break;
105                 }
106             }
```

```
107         }
108
109         [Test]
110         public void Test_DetectInvalidCommand()
111         {
112             string input = "run";
113
114             foreach (Command command in commands)
115             {
116                 if (input == command.FirstId)
117                 {
118                     Console.WriteLine("Command found");
119                     break;
120                 }
121                 else
122                 {
123                     var sut = false;
124                     Assert.IsFalse(sut);
125                     Console.WriteLine($"Search Command: {input}\n" +
126                         $"Command Returned: None");
127                 }
128             }
129         }
130     }
131 }
```


CommandProcessor
- _commandTypes: List<Command>
- _currentTypes: Command
+ ExecuteCommand(Player p, string[] text): string
MoveCommand
LookCommand


:CommandProcessor
:LookCommand
ExecuteCommand
(p, ["look", "at", "gem"])
DetectType
("look", "at", "gem")
Execute(p, ["look", "at", "gem"])
return "..."
return "..."
return "..."
:CommandProcessor
:MoveCommand
ExecuteCommand
(p, "move", "n")
DetectType
("move", "n")
Execute(Player p, string[] text)
return "..."
return "..."
return "..."


CommandProcessorTests (5) 3 ms
Test_DetectInvalidCommand 3 ms
Test_DetectLookCommand < 1 ms
Test_DetectLookVariant < 1 ms
Test_DetectMoveCommand < 1 ms
Test_DetectMoveVariant < 1 ms


D:\Swinburne\SwinburneCS20007\Assignment 10.1C\MazeGame\bin\Debug\net6.0\MazeGame.exe
Swin-Adventure Maze Game
Welcome
You are Hoang An the comtemplator of infinity
You are carrying:
an obsidian knife (knife)
a stone axe (axe)
leather bag (b1)
You have arrived at green garden
For look command, type in command 'look'
Note: The input must be either 3 or 5 words only
Example: 'look at ...' or 'look at ... in ...'
For move command, type in command with directions (n, e, s, w):
'go'
'move'
'head'
'leave'
Note: The input must be 2 words only
Command:
D:\Swinburne\SwinburneCS20007\Assignment 10.1C\MazeGame\bin\Debug\net6.0\MazeGame.exe
Output:
You are in a green garden
A garden blooming with natural plants, trees, and flowers
There are 2 available pathways:
north
east
Items available in this area:
a bottled water (water)
a pearl (pearl)
For look command, type in command 'look'
Note: The input must be either 3 or 5 words only
Example: 'look at ...' or 'look at ... in ...'
For move command, type in command with directions (n, e, s, w):
'go'
'move'
'head'
'leave'
Note: The input must be 2 words only
Command:
D:\Swinburne\SwinburneCS20007\Assignment 10.1C\MazeGame\bin\Debug\net6.0\MazeGame.exe
Output:
You travelled towards east
You walked for a kilometer to a library in East
Items available in this area:
a history book (book)
For look command, type in command 'look'
Note: The input must be either 3 or 5 words only
Example: 'look at ...' or 'look at ... in ...'
For move command, type in command with directions (n, e, s, w):
'go'
'move'
'head'
'leave'
Note: The input must be 2 words only
Command:
D:\Swinburne\SwinburneCS20007\Assignment 10.1C\MazeGame\bin\Debug\net6.0\MazeGame.exe
Output:
You are in a archive library
area that contains old history book
There are 2 available pathways:
west
northwest
Items available in this area:
a history book (book)
For look command, type in command 'look'
Note: The input must be either 3 or 5 words only
Example: 'look at ...' or 'look at ... in ...'
For move command, type in command with directions (n, e, s, w):
'go'
'move'
'head'
'leave'
Note: The input must be 2 words only
Command: