Swinburne University of Technology

COS20007 Object Oriented Programming

10.1C - Case Study - Iteration 8 - Command Processor

PDF generated at 00:43 on Friday 12th May, 2023

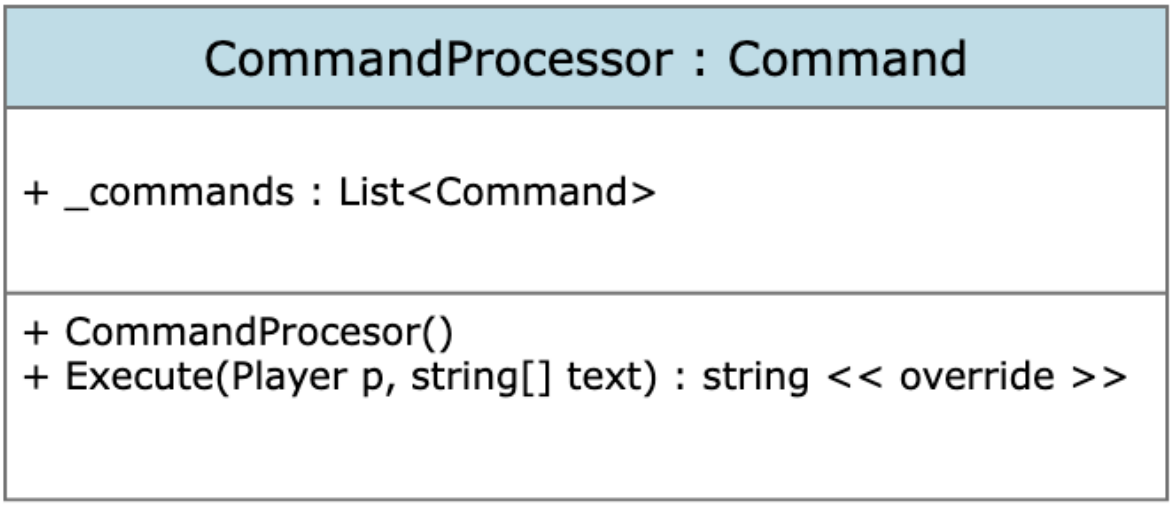
```
namespace SwinAdventure
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            string name;
            string desc;
            Player player;

            Console.WriteLine("Welcome to SwinAdventure!\n");

            // PLayer Descriptions
            Console.WriteLine("Enter Player Name:");
            name = Console.ReadLine();

            Console.WriteLine("Enter Player Description:");
            desc = Console.ReadLine();

            player = new Player(name, desc);

            // Setting items and inventory
            Item sword = new Item(new string[] { "Sword" }, "a golden sword", "This is a golden sword");
            Item knife = new Item(new string[] { "Knife" }, "a sharp knife", "This is a sharp knife");
            Item gem = new Item(new string[] { "Diamond" }, "a valuable gem", "This is an expensive item");

            Bag bag = new Bag(new string[] { "Bag" }, "big bag", "This is a big bag");

            player.Inventory.Put(sword);
            player.Inventory.Put(knife);
            player.Inventory.Put(bag);
            bag.Inventory.Put(gem);

            // Setting up location
            Item skull = new Item(new string[] { "skull" }, "a skull", "This is a creepy skull");
            Item torch = new Item(new string[] { "torch" }, "a torch", "This is a bright torch");
            Location jungle = new Location("a jungle", "This is a scary jungle");
            Location tower = new Location("a tower", "This is a tilted tower");

            Path jungleSouth = new Path(new string[] { "south", "s" }, "South", "South Path", tower);
            Path towerNorth = new Path(new string[] { "north", "n" }, "North", "North Path", jungle);
            jungle.AddPath(jungleSouth);
            tower.AddPath(towerNorth);


```

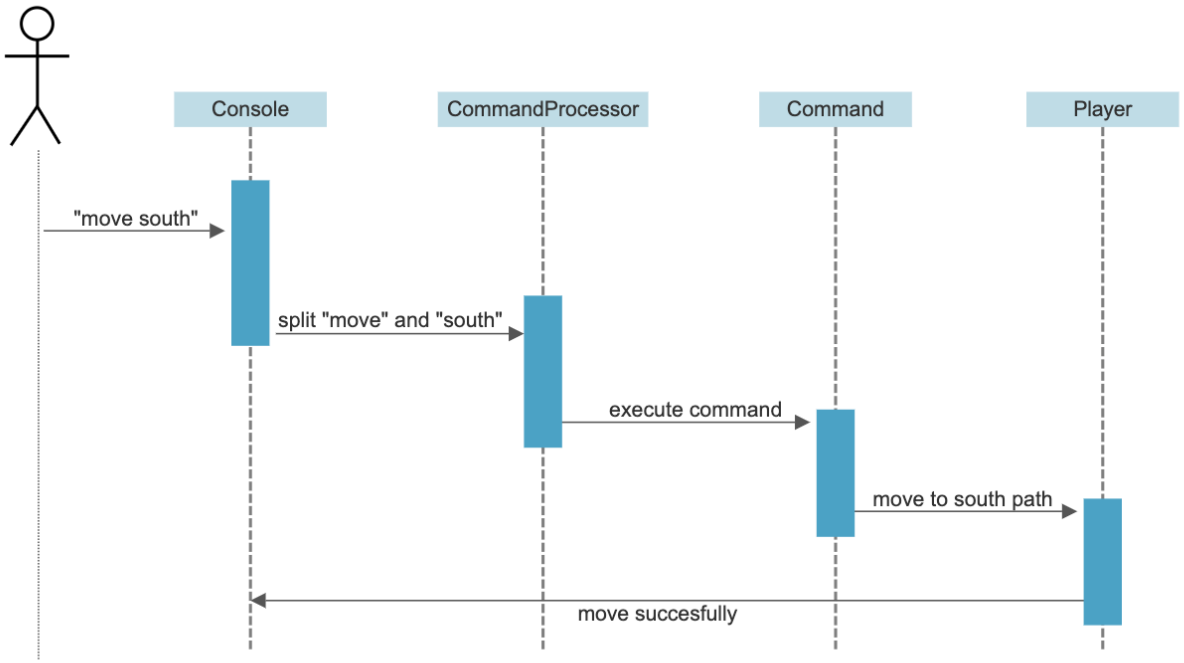
```
46              jungle.Inventory.Put(skull);
47              tower.Inventory.Put(torch);
48              player.Location = jungle;
49
50              // Command
51              while (true)
52              {
53                  Console.WriteLine("\nCommand: ");
54                  string input = Console.ReadLine();
55                  if (!string.IsNullOrEmpty(input))
56                  {
57                      string[] execommand = input.ToLower().Split(' ');
58                      if (execommand[0] == "quit")
59                      {
60                          break;
61                      }
62                      else
63                      {
64                          Console.WriteLine(new CommandProcessor().Execute(player,
    ↪  execommand));
65                      }
66                  }
67
68              }
69          }
70      }
71  }
```

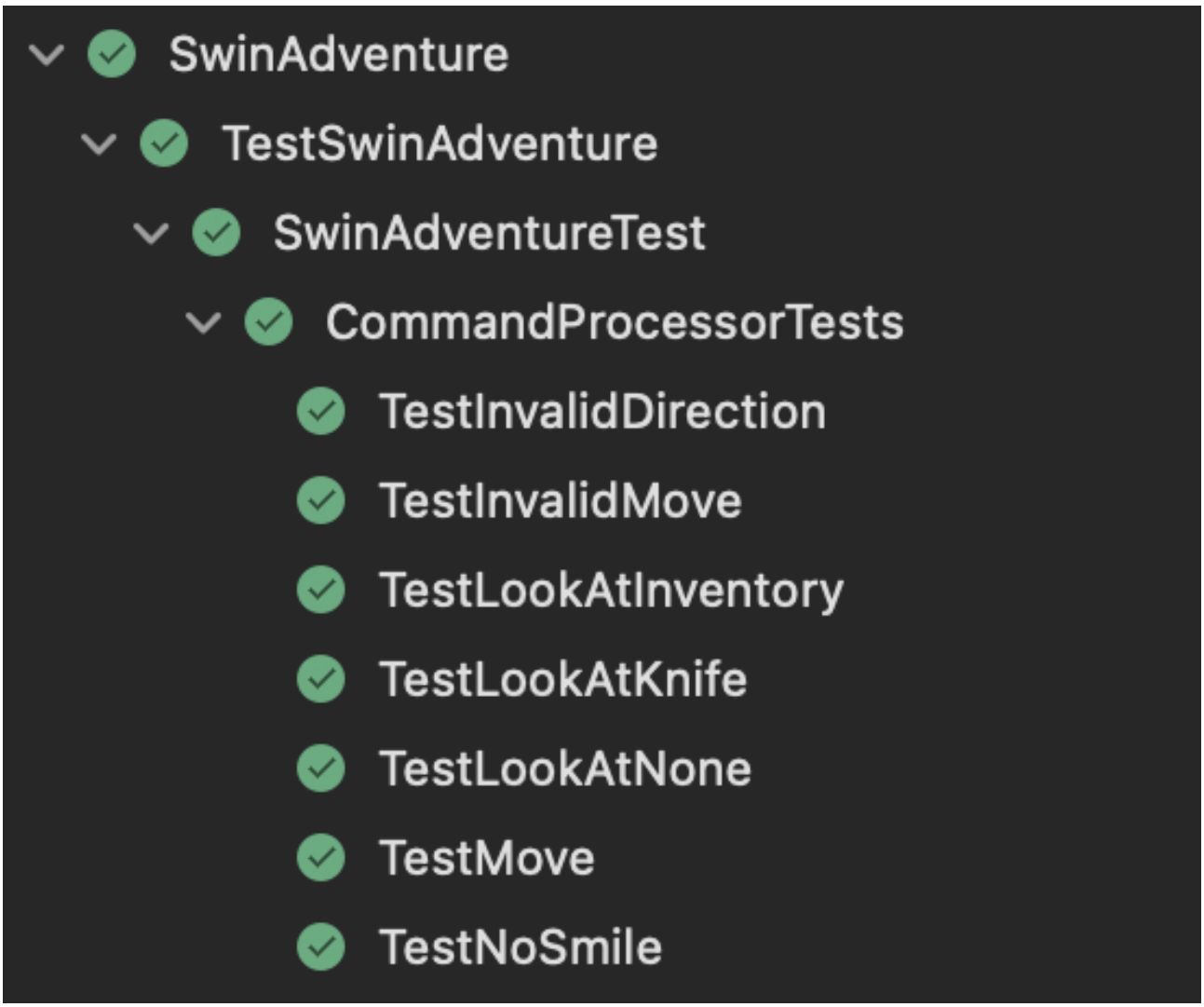
```
using System;
using System.Numerics;

namespace SwinAdventure
{
    public class CommandProcessor : Command
    {
        List<Command> _commands;

        public CommandProcessor() : base(new string[] {"command"})
        {
            _commands = new List<Command>();
            _commands.Add(new LookCommand());
            _commands.Add(new MoveCommand());
        }

        public override string Execute(Player p, string[] text)
        {
            string input = text[0].ToLower();
            Command commandToExecute = null;
            // loop to find the most suitable command
            foreach (Command command in _commands)
            {
                if (command.AreYou(input))
                {
                    commandToExecute = command;
                    break;
                }
            }
            // if can't find the suitable command
            if (commandToExecute == null)
            {
                return "I don't know how to " + input + ".";
            }
            return commandToExecute.Execute(p, text);
        }
    }
}

```

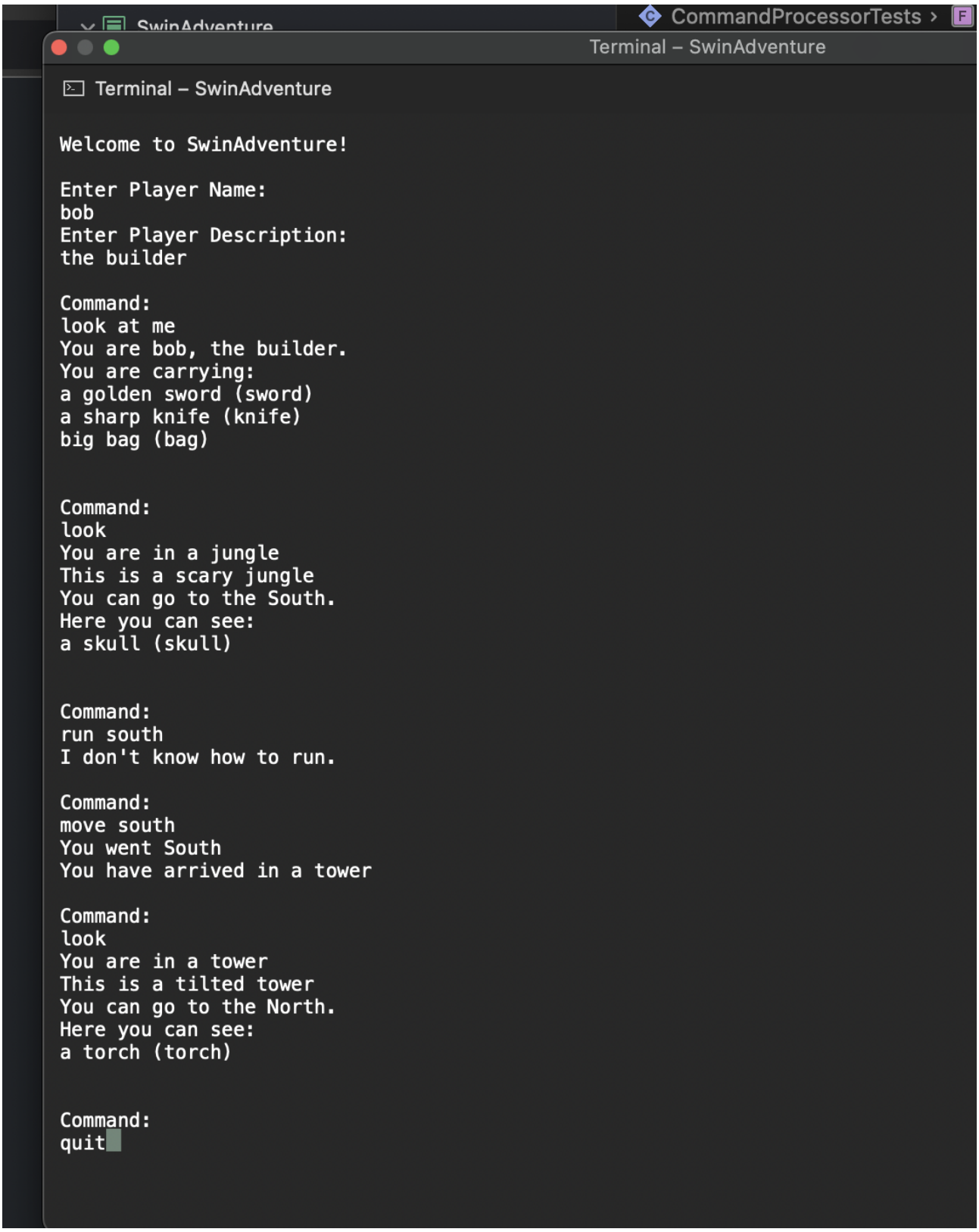
```
using SwinAdventure;
using System.Numerics;
using Path = SwinAdventure.Path;

namespace SwinAdventureTest
{
    [TestFixture]
    public class CommandProcessorTests
    {
        CommandProcessor command;
        Location location;
        Location destination;
        Path path;
        Player player;
        Item knife;


        [SetUp]
        public void Setup()
        {
            command = new();
            location = new Location("a jungle", "This is a creepy jungle");
            destination = new Location("a tower", "This is a tilted tower");
            path = new Path(new string[] { "south" }, "south", "this is south",
↪   destination);
            player = new Player("bob", "the builder");
            knife = new Item(new string[] { "Knife" }, "a sharp knife", "This is a
↪   sharp knife");

            player.Location = location;
            location.AddPath(path);
        }

        [Test]
        public void TestLookAtNone()
        {
            string actual = command.Execute(player, new string[] { "look", "at",
↪   "none" });
            string expected = "I can't find the none";
            Assert.That(actual, Is.EqualTo(expected));
        }
        [Test]
        public void TestLookAtInventory()
        {
            string actual = command.Execute(player, new string[] { "look", "at",
↪   "inventory" });
            string expected = "You are bob, the builder.\nYou are carrying:\n";
↪
            Assert.That(actual, Is.EqualTo(expected));
        }
        [Test]
        public void TestLookAtKnife()
        {
```

```
49              player.Inventory.Put(knife);
50              string actual = command.Execute(player, new string[] { "look", "at",
    ↪  "knife" });
51              Assert.That(actual, Is.EqualTo(knife.FullDescription));
52          }
53          [Test]
54          public void TestNoSmile()
55          {
56              string actual = command.Execute(player, new string[] { "smile" });
57              string expected = "I don't know how to smile.";
58              Assert.That(actual, Is.EqualTo(expected));
59          }
60          [Test]
61          public void TestMove()
62          {
63              Assert.That(player.Location, Is.SameAs(location));
64              string actual = command.Execute(player, new string[] { "move", "south"
    ↪  });
65              Assert.That(player.Location, Is.SameAs(destination));
66          }
67          [Test]
68          public void TestInvalidMove()
69          {
70              Assert.That(player.Location, Is.SameAs(location));
71              string actual = command.Execute(player, new string[] { "move", "north"
    ↪  });
72              Assert.That(player.Location, Is.SameAs(location));
73          }
74          [Test]
75          public void TestInvalidDirection()
76          {
77              string actual = command.Execute(player, new string[] { "move", "west" });
78              string expected = "Error in direction!";
79              Assert.That(actual, Is.EqualTo(expected));
80          }
81      }
82  }
```


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


Console
CommandProcessor
Command
Player
"move south"
split "move" and "south"
execute command
move to south path
move succesfully


SwinAdventure
TestSwinAdventure
SwinAdventureTest
CommandProcessorTests
TestInvalidDirection
TestInvalidMove
TestLookAtInventory
TestLookAtKnife
TestLookAtNone
TestMove
TestNoSmile


SwinAdventure
CommandProcessorTests
Terminal – SwinAdventure
Terminal – SwinAdventure
Welcome to SwinAdventure!
Enter Player Name:
bob
Enter Player Description:
the builder
Command:
look at me
You are bob, the builder.
You are carrying:
a golden sword (sword)
a sharp knife (knife)
big bag (bag)
Command:
look
You are in a jungle
This is a scary jungle
You can go to the South.
Here you can see:
a skull (skull)
Command:
run south
I don't know how to run.
Command:
move south
You went South
You have arrived in a tower
Command:
look
You are in a tower
This is a tilted tower
You can go to the North.
Here you can see:
a torch (torch)
Command:
quit