Swinburne University of Technology

COS20007 Object Oriented Programming

10.1C - Case Study - Iteration 8 - Command Processor

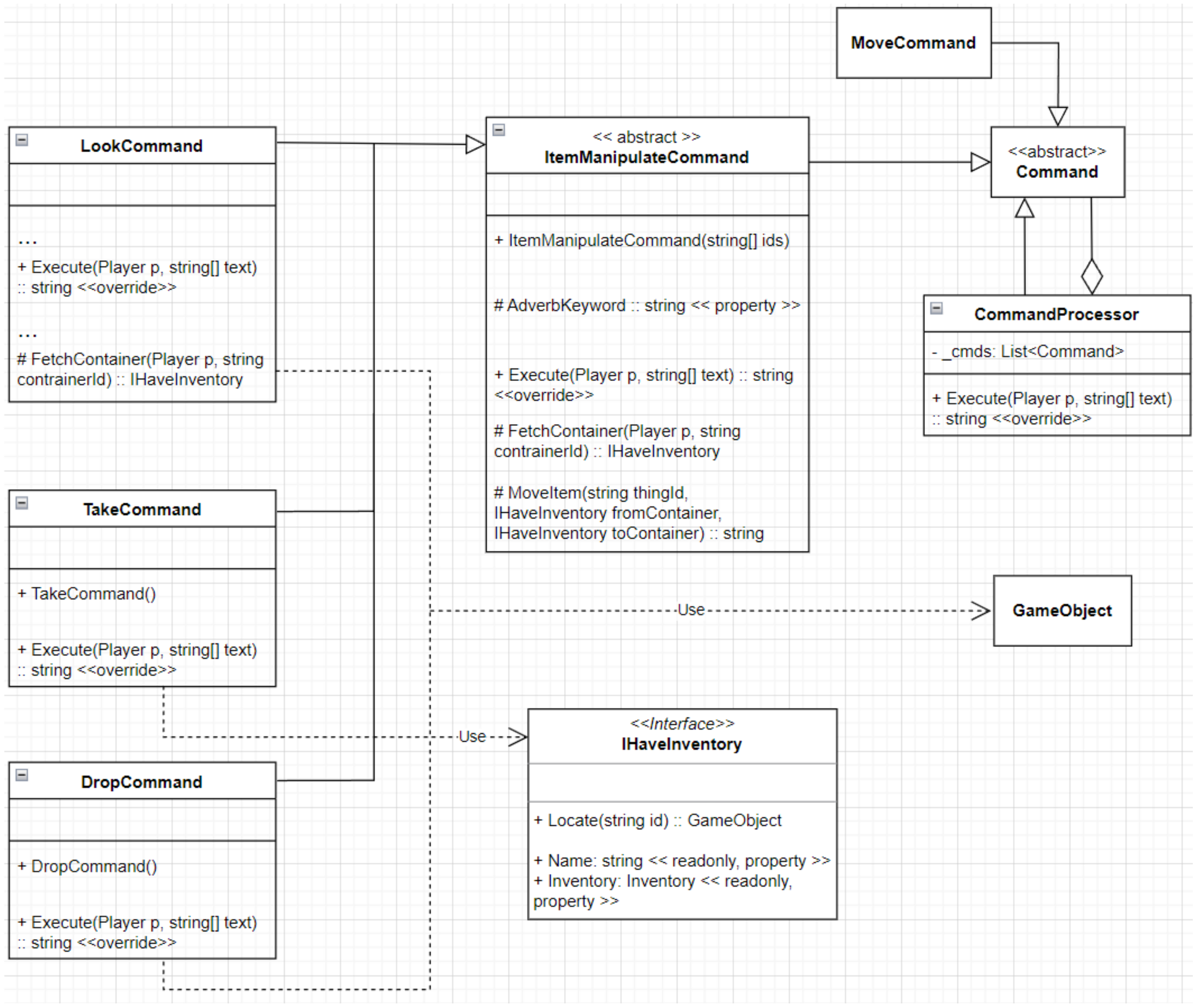
PDF generated at 16:12 on Monday 15th May, 2023

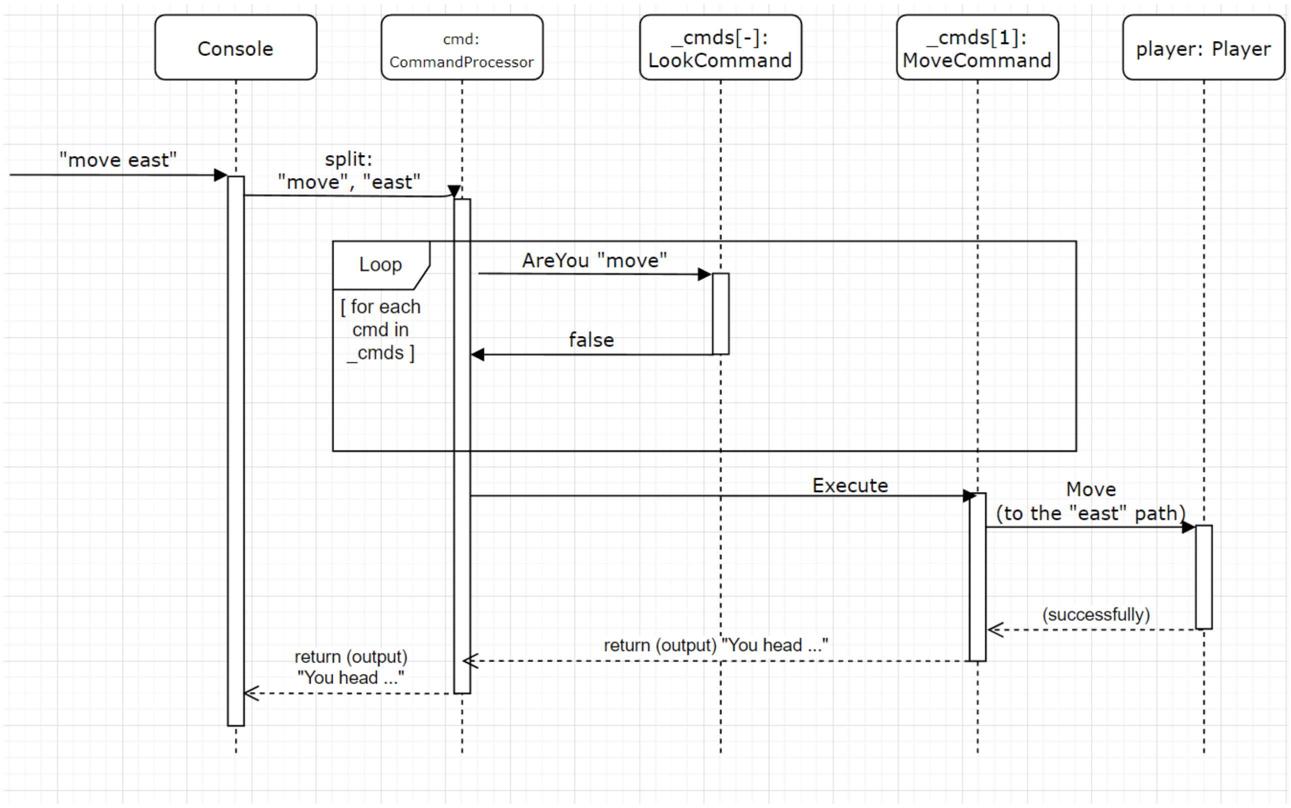
```csharp
namespace SwinAdventure
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Player player = GetPlayerInfomation();
            Bag bag = new Bag(new string[] { "bag" }, "a bag", "This is a bag");

            Item sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword", "This is a bronze sword");
            Item shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel");
            Item computer = new Item(new string[] { "pc", "computer" }, "a small computer", "This is a small computer");
            Item gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");
            Item gun = new Item(new string[] { "gun", "short" }, "a short gun", "This is a short gun");
            Item pen = new Item(new string[] { "pen" }, "a pen", "This is a pen");

            Location studio = new Location(new string[] { "studio" }, "a studio", "A small, beautiful and fully-furnished studio");
            Location closet = new Location(new string[] { "closet" }, "a closet", "A small dark closet, with an odd smell");
            Location garden = new Location(new string[] { "closet" }, "a garden", "A large and beautiful garden");

            Path studioDoor1 = new Path(new string[] { "east", "e" }, "first door", "The first small door", studio, closet);
            Path studioDoor2 = new Path(new string[] { "south", "s" }, "second door", "The second large door", studio, garden);
            Path closetDoor = new Path(new string[] { "west", "w" }, "door", "The small door", closet, studio);
            Path closetWindow = new Path(new string[] { "southwest", "sw" }, "window", "The large window", closet, garden);
            Path gardenDoor = new Path(new string[] { "north", "n" }, "door", "The large door", garden, studio);
            Path gardenWindow = new Path(new string[] { "northeast ", "ne" }, "window", "The large window", garden, closet);

            studioDoor2.Close();
            gardenDoor.Close();

            studio.Inventory.Put(shovel);
            studio.AddPath(studioDoor1);
            studio.AddPath(studioDoor2);
            closet.Inventory.Put(gun);
            closet.Inventory.Put(pen);
            closet.AddPath(closetDoor);
            closet.AddPath(closetWindow);
            garden.Inventory.Put(gem);
            garden.AddPath(gardenDoor);
            garden.AddPath(gardenWindow);

            player.Location = studio;
            player.Inventory.Put(sword);
            player.Inventory.Put(bag);
            bag.Inventory.Put(computer);

            Command cmd = new CommandProcessor();
            string input;
            string output;

            while (true)
            {
                Console.Write("Command: ");
                input = Console.ReadLine().ToLower();

                output = cmd.Execute(player, input.Split());
                Console.WriteLine(output);

                Console.WriteLine("-----------------------------------------------------------------------");
                if (output == "Bye.")
                {
                    break;
                }
            }
        }

        private static Player GetPlayerInfomation()
        {
            Console.WriteLine("====================WELCOME TO SWIN ADVENTURE==========================");

            Player player;
            Console.WriteLine("Please enter your name:");
            string name = Console.ReadLine();
            Console.WriteLine("and your description:");
            string description = Console.ReadLine();

            Console.WriteLine("=======================================================================");

            player = new Player(name, description);
            return player;
        }
    }
}
```

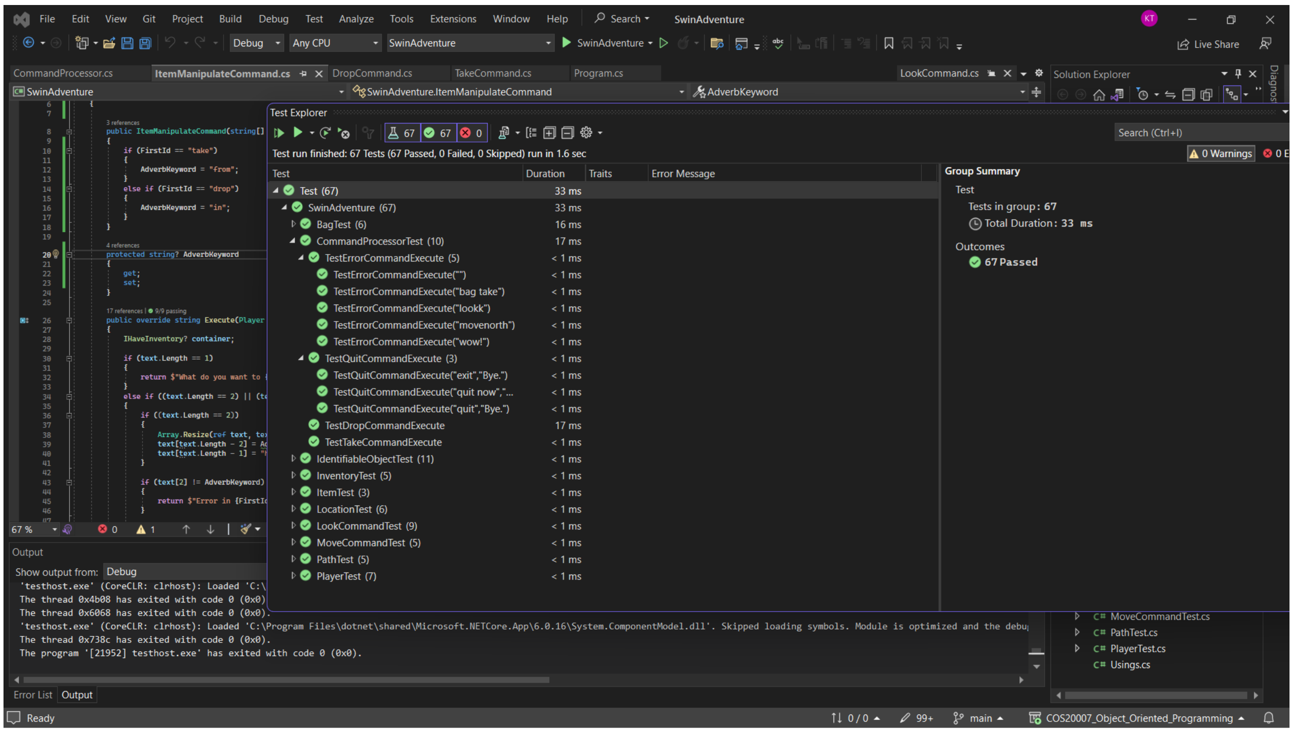
```
namespace SwinAdventure
{
    public class CommandProcessor : Command
    {
        private List<Command> _cmds;

        public CommandProcessor() : base(new string[] { "processor" })
        {
            _cmds = new List<Command>();

            _cmds.Add(new LookCommand());
            _cmds.Add(new MoveCommand());
            _cmds.Add(new TakeCommand());
            _cmds.Add(new DropCommand());
            _cmds.Add(new QuitCommand());
        }

        public override string Execute(Player p, string[] text)
        {
            foreach (Command cmd in _cmds)
            {
                if (cmd.AreYou(text[0]))
                {
                    return cmd.Execute(p, text);
                }
            }
            return "Error in the input.";
        }
    }
}
```

```
using System.IO;
using System.Numerics;

namespace SwinAdventure
{
    public class CommandProcessorTest
    {
        private CommandProcessor _cmdProcessor;
        private Player _player;

        private Bag _bag;

        private Item _sword;
        private Item _shovel;
        private Item _computer;
        private Item _gem;
        private Item _pen;

        private Location _studio;
        private Location _closet;

        private Path _studioDoor;
        private Path _closetDoor;

        [SetUp]
        public void SetUp()
        {
            _cmdProcessor = new CommandProcessor();

            _studio = new Location(new string[] { "studio" }, "a studio", "A small, beautiful and fully-furnished studio");
            _closet = new Location(new string[] { "closet" }, "a closet", "A small dark closet, with an odd smell");

            _player = new Player("Trung Kien Nguyen", "I am the player");

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

            _sword = new Item(new string[] { "sword", "bronze" }, "a bronze sword", "This is a bronze sword");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel");
            _computer = new Item(new string[] { "pc", "computer" }, "a small computer", "This is a small computer");
            _gem = new Item(new string[] { "gem" }, "a gem", "This is a gem");
            _pen = new Item(new string[] { "pen" }, "a pen", "This is a pen");

            _studioDoor = new Path(new string[] { "east", "e" }, "first door", "The first small door", _studio, _closet);
            _closetDoor = new Path(new string[] { "west", "w" }, "door", "The small door", _closet, _studio);

            _studio.Inventory.Put(_shovel);
        _studio.Inventory.Put(_pen);
        _studio.AddPath(_studioDoor);
        _closet.Inventory.Put(_gem);
        _closet.AddPath(_closetDoor);

        _player.Location = _studio;
        _player.Inventory.Put(_sword);
        _player.Inventory.Put(_bag);
        _bag.Inventory.Put(_computer);
    }

    [TestCase("")]
    [TestCase("lookk")]
    [TestCase("movenorth")]
    [TestCase("bag take")]
    [TestCase("wow!")]
    public void TestErrorCommandExecute(string input)
    {
        Assert.AreEqual(_cmdProcessor.Execute(_player, input.ToLower().Split()), "Error in the input.");
    }

    /* LOOK COMMAND AND MOVE COMMAND HAVE BEEN TESTED BEFORE
    [Test]
    public void TestLookCommandExecute()
    {
        // Successful commands
        //      1. Look in the current room
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look".ToLower().Split()), _player.Location.FullDescription);
        //      2. Look at the player and its inventory (me)
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look at me".ToLower().Split()), _player.FullDescription);
        //      3. Look at sword in player inventory
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look at sword".ToLower().Split()), _sword.FullDescription);
        //      4. Look at pen in current room
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look at pen in room".ToLower().Split()), _pen.FullDescription);

        // Unsuccessful commands
        //      1. Look command having two words
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look at".ToLower().Split()), "I don't know how to look like that");
        //      2. Look command having four words
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look at sword in".ToLower().Split()), "I don't know how to look like that");
        //      3. Look command having invalid second word
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look att inventory".ToLower().Split()), "What do you want to look at?");
        //      4. Look command having invalid fourth word
        Assert.AreEqual(_cmdProcessor.Execute(_player, "look at pen innn room".ToLower().Split()), "What do you want to look in?");
```

```
            //      5. Look at gem which is not in player inventory
            Assert.AreEqual(_cmdProcessor.Execute(_player, "look at gem".ToLower().Split()), $"I cannot find the gem in the {_player.Name}");
        }

        [Test]
        public void TestMoveCommandExecute()
        {
            // Successful commands
            //      1. Move east through the studio door to the closet
            _player.Location = _studio;
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move east".ToLower().Split()), $"You head {_studioDoor.FirstId}\nYou go through {_studioDoor.FullDescription}\nYou have arrived {_studioDoor.EndingLocation.Name}.");
            Assert.AreEqual(_player.Location, _closet);

            // Unsuccessful commands
            //      1. Move command having three words
            _player.Location = _studio;
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move to east".ToLower().Split()), "Error in move input.");
            Assert.AreEqual(_player.Location, _studio);
            //      2. Move command having one word only
            _player.Location = _studio;
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move".ToLower().Split()), "Which direction do you want to move to?");
            Assert.AreEqual(_player.Location, _studio);
            //      3. No north path in studio
            _player.Location = _studio;
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move north".ToLower().Split()), $"Could not find the north path.");
            Assert.AreEqual(_player.Location, _studio);
            //      4. No pen path in studio
            _player.Location = _studio;
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move pen".ToLower().Split()), $"Could not find {_pen.Name}.");
            Assert.AreEqual(_player.Location, _studio);
            //      5. Path is assigned wrongly (starting location is not the current location)
            _player.Location = _studio;
            _studio.AddPath(_closetDoor);
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move west".ToLower().Split()), $"Could not move from {_closetDoor.StartingLocation.Name}.");
            Assert.AreEqual(_player.Location, _studio);
            //      6. Ending location is invalid (null)
            _player.Location = _studio;
            Path studioDoor2 = new Path(new string[] { "northeast", "ne" }, "first door", "The first small door", _studio, null);
            _studio.AddPath(studioDoor2);
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move northeast".ToLower().Split()), "Could not move.");
            Assert.AreEqual(_player.Location, _studio);
            //      7. Path is closed
            _player.Location = _studio;
            _studioDoor.Close();
            Assert.AreEqual(_cmdProcessor.Execute(_player, "move east".ToLower().Split()), $"The path {_studioDoor.Name} is closed");
            Assert.AreEqual(_player.Location, _studio);
        }
        */

        [TestCase("quit", "Bye.")]
        [TestCase("exit", "Bye.")]
        [TestCase("quit now", "Error in quit input.")]
        public void TestQuitCommandExecute(string input, string output)
        {
            Assert.AreEqual(_cmdProcessor.Execute(_player, input.ToLower().Split()), output);
        }

        [Test]
        public void TestTakeCommandExecute()
        {
            // Successful commands
            //      1. Take the item shovel from the player current location (studio) to the player Inventory
            Assert.IsFalse(_player.Inventory.HasItem("shovel"));
            Assert.IsTrue(_player.Location.Inventory.HasItem("shovel"));
            Assert.AreEqual(_cmdProcessor.Execute(_player, "take shovel".ToLower().Split()), $"Moved the shovel from {_player.Location.Name} to {_player.Name}.");
            Assert.IsTrue(_player.Inventory.HasItem("shovel"));
            Assert.IsFalse(_player.Location.Inventory.HasItem("shovel"));
            //      2. Take the item pc from the player bag to the player Inventory
            IHaveInventory playerBag = (_player.Locate("bag")) as IHaveInventory ;
            Assert.IsFalse(_player.Inventory.HasItem("pc"));
            Assert.IsTrue(playerBag.Inventory.HasItem("pc"));
            Assert.AreEqual(_cmdProcessor.Execute(_player, "pickup pc from bag".ToLower().Split()), $"Moved the pc from {playerBag.Name} to {_player.Name}.");
            Assert.IsTrue(_player.Inventory.HasItem("pc"));
            Assert.IsFalse(playerBag.Inventory.HasItem("pc"));

            // Unsuccessful commands
            Inventory initialPlayerInvetory = _player.Inventory;
            Inventory initialStudioInvetory = _player.Location.Inventory;
            //      1. Take command having invalid number of words
            Assert.AreEqual(_cmdProcessor.Execute(_player, "take the pen".ToLower().Split()), "Error in take input.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
            Assert.AreEqual(initialStudioInvetory, _player.Location.Inventory);
            //      2. Take command having invalid keyword
            Assert.AreEqual(_cmdProcessor.Execute(_player, "pickup pen frommm here".ToLower().Split()), "Error in take input.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
            Assert.AreEqual(initialStudioInvetory, _player.Location.Inventory);
            //      2. Take item that is not in referred container Inventory
            Assert.AreEqual(_cmdProcessor.Execute(_player, "take pc from here".ToLower().Split()), $"Could not find the pc from {_studio.Name}.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
            Assert.AreEqual(initialStudioInvetory, _player.Location.Inventory);
            //      3. Take item from an unidentifiable container
            Assert.AreEqual(_cmdProcessor.Execute(_player, "pickup pc from bagg".ToLower().Split()), $"Could not find the bagg.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
        }

        [Test]
        public void TestDropCommandExecute()
        {
            IHaveInventory playerBag = (_player.Locate("bag")) as IHaveInventory;
            // Unsuccessful commands
            Inventory initialPlayerInvetory = _player.Inventory;
            Inventory initialBagInvetory = playerBag.Inventory;
            //      1. Drop command having invalid number of words
            Assert.AreEqual(_cmdProcessor.Execute(_player, "drop the sword in bag".ToLower().Split()), "Error in drop input.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
            Assert.AreEqual(initialBagInvetory, _bag.Inventory);
            //      2. Drop command having invalid keyword
            Assert.AreEqual(_cmdProcessor.Execute(_player, "put sword inn bag".ToLower().Split()), "Error in drop input.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
            Assert.AreEqual(initialBagInvetory, _bag.Inventory);
            //      2. Drop item that is not in the player Inventory
            Assert.AreEqual(_cmdProcessor.Execute(_player, "drop gem in bag".ToLower().Split()), $"Could not find the gem from {_player.Name}.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);
            Assert.AreEqual(initialBagInvetory, _bag.Inventory);
            //      3. Drop item from an unidentifiable container
            Assert.AreEqual(_cmdProcessor.Execute(_player, "put sword in bagg".ToLower().Split()), $"Could not find the bagg.");
            Assert.AreEqual(initialPlayerInvetory, _player.Inventory);

            // Successful commands
            //      1. Drop the item sword from the player Inventory in the bag
            Assert.IsTrue(_player.Inventory.HasItem("sword"));
            Assert.IsFalse(playerBag.Inventory.HasItem("sword"));
            Assert.AreEqual(_cmdProcessor.Execute(_player, "drop sword in bag".ToLower().Split()), $"Moved the sword from {_player.Name} to {playerBag.Name}.");
            Assert.IsFalse(_player.Inventory.HasItem("sword"));
            Assert.IsTrue(playerBag.Inventory.HasItem("sword"));
            //      2. Drop the item bag from the player Inventory in the player current location (studio)
            Assert.IsTrue(_player.Inventory.HasItem("bag"));
            Assert.IsFalse(_player.Location.Inventory.HasItem("bag"));
            Assert.AreEqual(_cmdProcessor.Execute(_player, "put
↪   bag".ToLower().Split()), $"Moved the bag from {_player.Name} to
↪   {_player.Location.Name}.");
            Assert.IsFalse(_player.Inventory.HasItem("bag"));
            Assert.IsTrue(_player.Location.Inventory.HasItem("bag"));
        }
    }
}
```


MoveCommand
<< abstract >>
ItemManipulateCommand
+ ItemManipulateCommand(string[] ids)
# AdverbKeyword :: string << property >>
+ Execute(Player p, string[] text) :: string <<override>>
# FetchContainer(Player p, string contrainerId) :: IHaveInventory
# MoveItem(string thingId, IHaveInventory fromContainer, IHaveInventory toContainer) :: string
<<abstract>>
Command
CommandProcessor
- _cmds: List<Command>
+ Execute(Player p, string[] text) :: string <<override>>
LookCommand
...
+ Execute(Player p, string[] text) :: string <<override>>
...
# FetchContainer(Player p, string contrainerId) :: IHaveInventory
TakeCommand
+ TakeCommand()
+ Execute(Player p, string[] text) :: string <<override>>
Use
GameObject
Use
<<Interface>>
IHaveInventory
+ Locate(string id) :: GameObject
+ Name: string << readonly, property >>
+ Inventory: Inventory << readonly, property >>
DropCommand
+ DropCommand()
+ Execute(Player p, string[] text) :: string <<override>>


Console
cmd: CommandProcessor
_cmds[-]: LookCommand
_cmds[1]: MoveCommand
player: Player
"move east"
split: "move", "east"
Loop
[for each cmd in _cmds]
AreYou "move"
false
Execute
Move (to the "east" path)
(successfully)
return (output) "You head ..."
return (output) "You head ..."



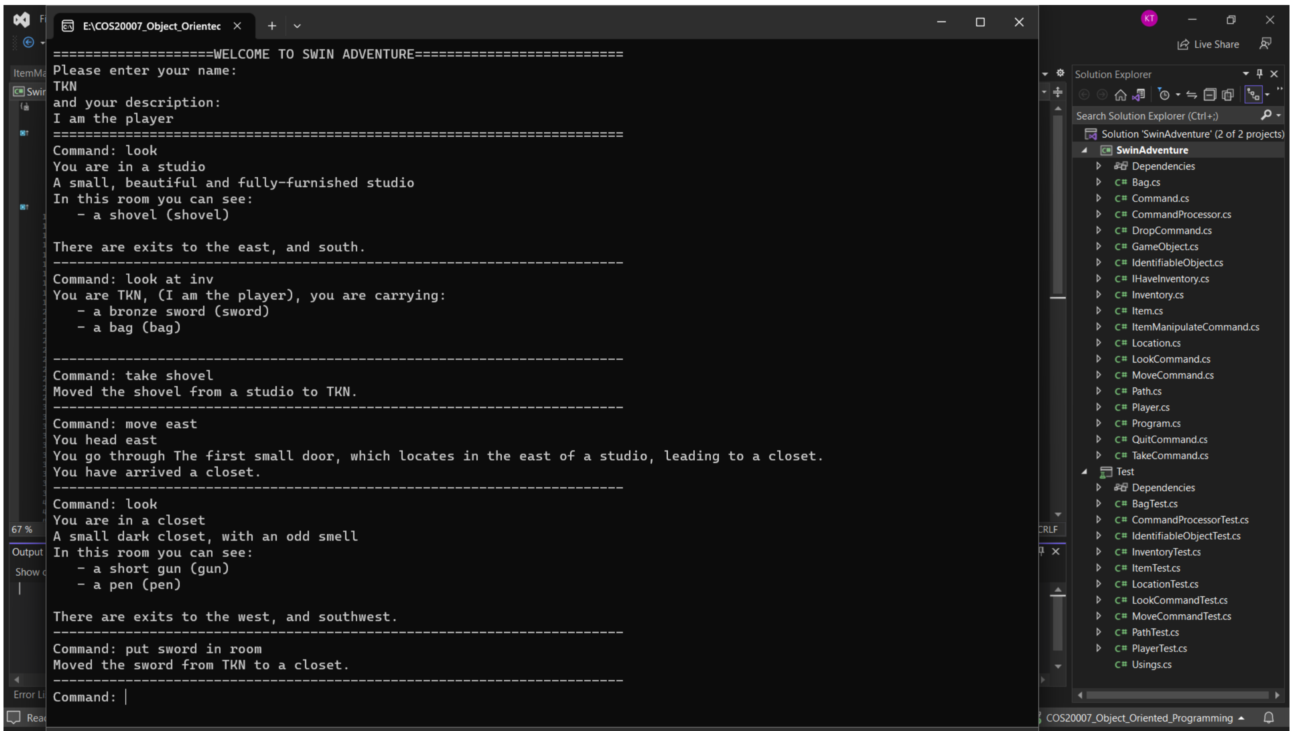
```
    {
        3 references
        public ItemManipulateCommand(string[]
        {
            if (FirstId == "take")
            {
                AdverbKeyword = "from";
            }
            else if (FirstId == "drop")
            {
                AdverbKeyword = "in";
            }
        }

        4 references
        protected string? AdverbKeyword
        {
            get;
            set;
        }

        17 references | 9/9 passing
        public override string Execute(Player
        {
            IHaveInventory? container;

            if (text.Length == 1)
            {
                return $"What do you want to
            }
            else if ((text.Length == 2) || (te
            {
                if ((text.Length == 2))
                {
                    Array.Resize(ref text, te
                    text[text.Length - 2] = Ad
                    text[text.Length - 1] = "
                }

                if (text[2] != AdverbKeyword)
                {
                    return $"Error in {FirstI
                }
```

```
67  67  0
Test run finished: 67 Tests (67 Passed, 0 Failed, 0 Skipped) run in 1.6 sec

Test                                               Duration   Traits   Error Message
◢ ✓ Test (67)                                         33 ms
  ◢ ✓ SwinAdventure (67)                              33 ms
    ▷ ✓ BagTest (6)                                   16 ms
    ◢ ✓ CommandProcessorTest (10)                     17 ms
      ◢ ✓ TestErrorCommandExecute (5)                 < 1 ms
          ✓ TestErrorCommandExecute("")               < 1 ms
          ✓ TestErrorCommandExecute("bag take")       < 1 ms
          ✓ TestErrorCommandExecute("lookk")          < 1 ms
          ✓ TestErrorCommandExecute("movenorth")      < 1 ms
          ✓ TestErrorCommandExecute("wow!")           < 1 ms
      ◢ ✓ TestQuitCommandExecute (3)                  < 1 ms
          ✓ TestQuitCommandExecute("exit","Bye.")     < 1 ms
          ✓ TestQuitCommandExecute("quit now","...    < 1 ms
          ✓ TestQuitCommandExecute("quit","Bye.")     < 1 ms
        ✓ TestDropCommandExecute                      17 ms
        ✓ TestTakeCommandExecute                      < 1 ms
    ▷ ✓ IdentifiableObjectTest (11)                   < 1 ms
    ▷ ✓ InventoryTest (5)                             < 1 ms
    ▷ ✓ ItemTest (3)                                  < 1 ms
    ▷ ✓ LocationTest (6)                              < 1 ms
    ▷ ✓ LookCommandTest (9)                           < 1 ms
    ▷ ✓ MoveCommandTest (5)                           < 1 ms
    ▷ ✓ PathTest (5)                                  < 1 ms
    ▷ ✓ PlayerTest (7)                                < 1 ms
```

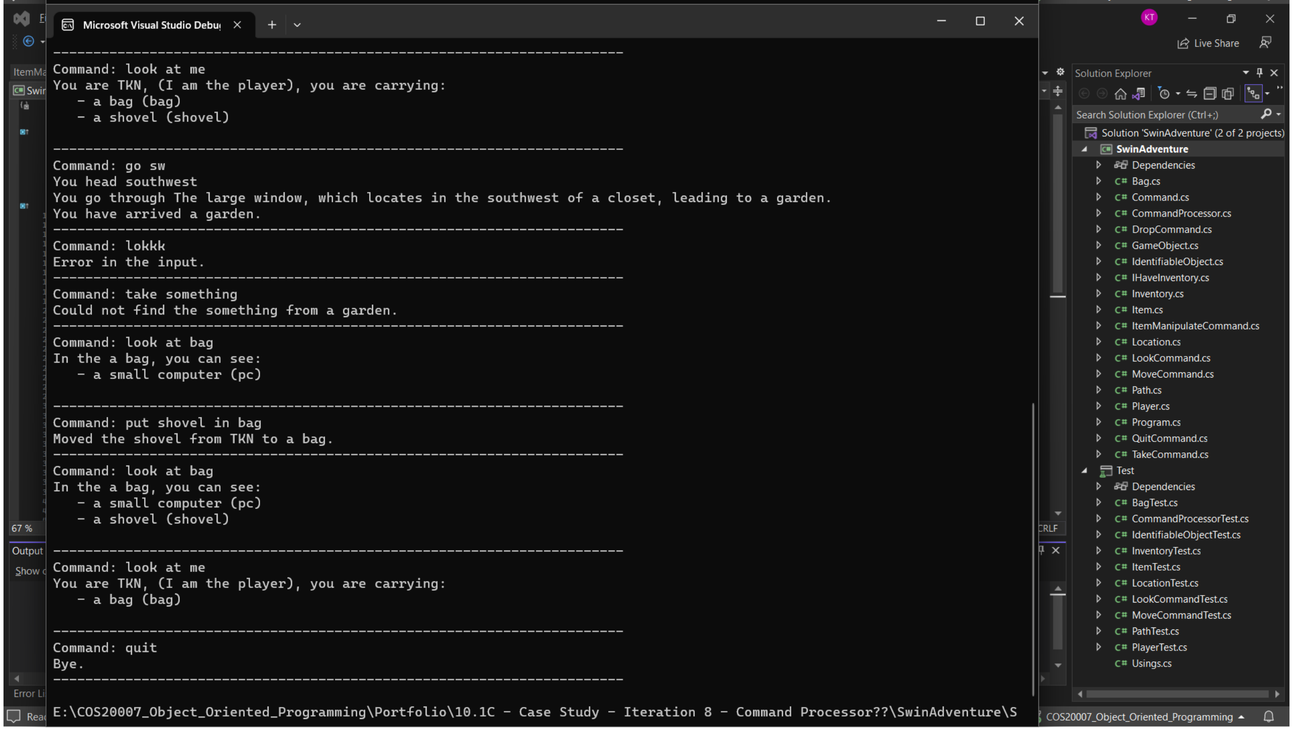
```
Group Summary
Test
  Tests in group: 67
  Total Duration: 33 ms
Outcomes
  67 Passed
```

```
MoveCommandTest.cs
PathTest.cs
PlayerTest.cs
Usings.cs
```

```
Show output from: Debug
'testhost.exe' (CoreCLR: clrhost): Loaded 'C:\
The thread 0x4b08 has exited with code 0 (0x0)
The thread 0x6068 has exited with code 0 (0x0).
'testhost.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.16\System.ComponentModel.dll'. Skipped loading symbols. Module is optimized and the debu
The thread 0x738c has exited with code 0 (0x0).
The program '[21952] testhost.exe' has exited with code 0 (0x0).
```



```
===================WELCOME TO SWIN ADVENTURE=========================
Please enter your name:
TKN
and your description:
I am the player
=====================================================================
Command: look
You are in a studio
A small, beautiful and fully-furnished studio
In this room you can see:
   - a shovel (shovel)

There are exits to the east, and south.
---------------------------------------------------------------------
Command: look at inv
You are TKN, (I am the player), you are carrying:
   - a bronze sword (sword)
   - a bag (bag)

---------------------------------------------------------------------
Command: take shovel
Moved the shovel from a studio to TKN.
---------------------------------------------------------------------
Command: move east
You head east
You go through The first small door, which locates in the east of a studio, leading to a closet.
You have arrived a closet.
---------------------------------------------------------------------
Command: look
You are in a closet
A small dark closet, with an odd smell
In this room you can see:
   - a short gun (gun)
   - a pen (pen)

There are exits to the west, and southwest.
---------------------------------------------------------------------
Command: put sword in room
Moved the sword from TKN to a closet.
---------------------------------------------------------------------
Command: |
```



```
---------------------------------------------------------------------
Command: look at me
You are TKN, (I am the player), you are carrying:
   - a bag (bag)
   - a shovel (shovel)

---------------------------------------------------------------------
Command: go sw
You head southwest
You go through The large window, which locates in the southwest of a closet, leading to a garden.
You have arrived a garden.
---------------------------------------------------------------------
Command: lokkk
Error in the input.
---------------------------------------------------------------------
Command: take something
Could not find the something from a garden.
---------------------------------------------------------------------
Command: look at bag
In the a bag, you can see:
   - a small computer (pc)

---------------------------------------------------------------------
Command: put shovel in bag
Moved the shovel from TKN to a bag.
---------------------------------------------------------------------
Command: look at bag
In the a bag, you can see:
   - a small computer (pc)
   - a shovel (shovel)

---------------------------------------------------------------------
Command: look at me
You are TKN, (I am the player), you are carrying:
   - a bag (bag)

---------------------------------------------------------------------
Command: quit
Bye.
---------------------------------------------------------------------

E:\COS20007_Object_Oriented_Programming\Portfolio\10.1C - Case Study - Iteration 8 - Command Processor??\SwinAdventure\S
```