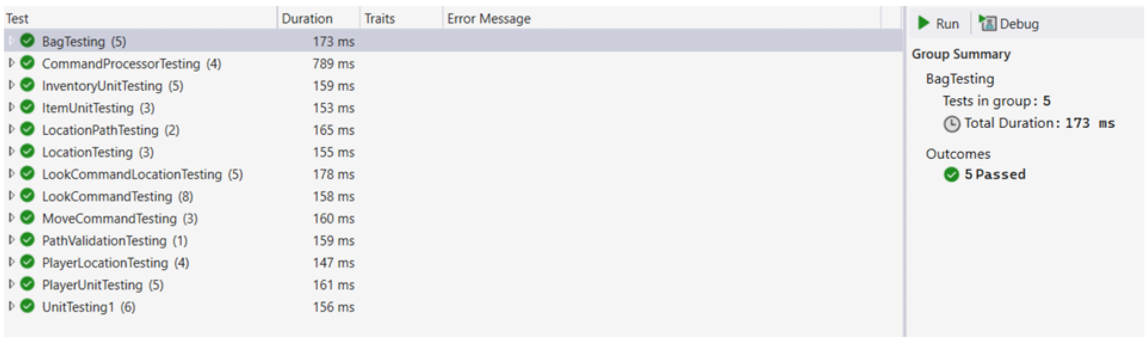
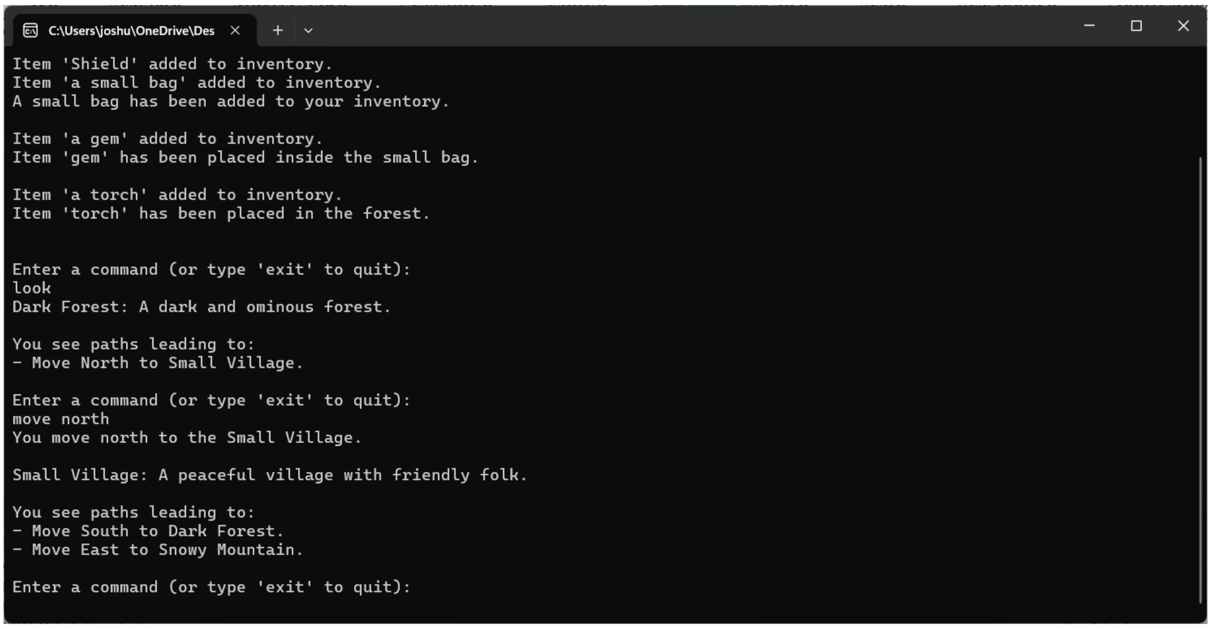
9.2 corrections

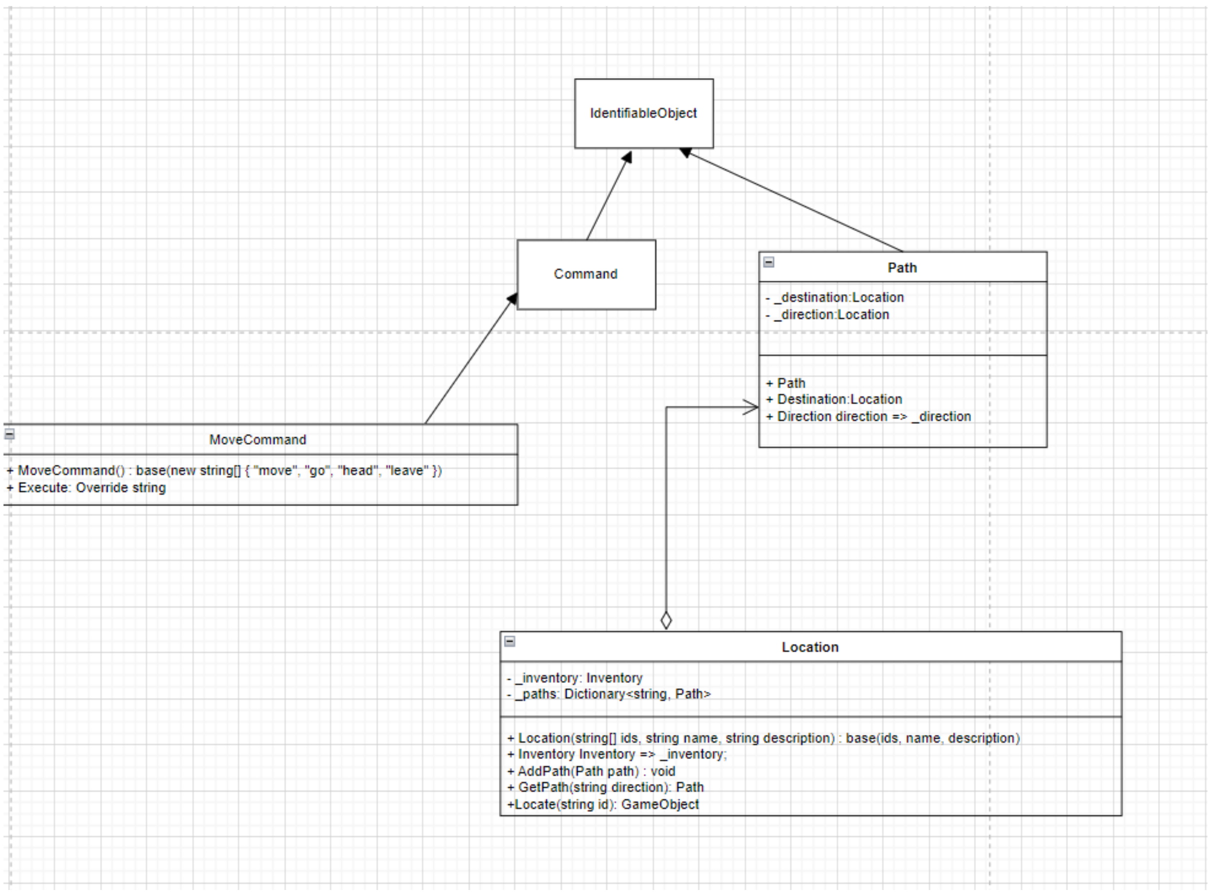


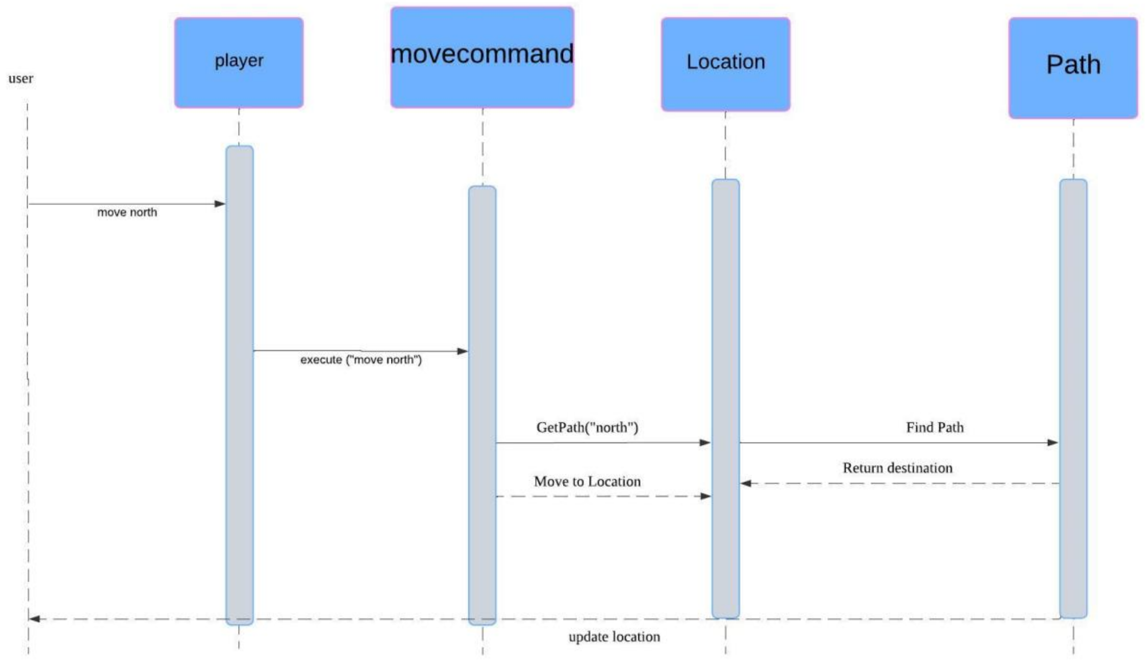
Test	Duration	Traits	Error Message
✓ BagTesting (5)	173 ms		
✓ CommandProcessorTesting (4)	789 ms		
✓ InventoryUnitTesting (5)	159 ms		
✓ ItemUnitTesting (3)	153 ms		
✓ LocationPathTesting (2)	165 ms		
✓ LocationTesting (3)	155 ms		
✓ LookCommandLocationTesting (5)	178 ms		
✓ LookCommandTesting (8)	158 ms		
✓ MoveCommandTesting (3)	160 ms		
✓ PathValidationTesting (1)	159 ms		
✓ PlayerLocationTesting (4)	147 ms		
✓ PlayerUnitTesting (5)	161 ms		
✓ UnitTesting1 (6)	156 ms		

Group Summary
BagTesting
Tests in group: 5
Total Duration: 173 ms
Outcomes
✓ 5 Passed



```
Item 'Shield' added to inventory.
Item 'a small bag' added to inventory.
A small bag has been added to your inventory.

Item 'a gem' added to inventory.
Item 'gem' has been placed inside the small bag.

Item 'a torch' added to inventory.
Item 'torch' has been placed in the forest.


Enter a command (or type 'exit' to quit):
look
Dark Forest: A dark and ominous forest.

You see paths leading to:
- Move North to Small Village.

Enter a command (or type 'exit' to quit):
move north
You move north to the Small Village.

Small Village: A peaceful village with friendly folk.

You see paths leading to:
- Move South to Dark Forest.
- Move East to Snowy Mountain.

Enter a command (or type 'exit' to quit):
```


IdentifiableObject
Command
Path
- _destination:Location
- _direction:Location
+ Path
+ Destination:Location
+ Direction direction => _direction
MoveCommand
+ MoveCommand() : base(new string[] { "move", "go", "head", "leave" })
+ Execute: Override string
Location
- _inventory: Inventory
- _paths: Dictionary<string, Path>
+ Location(string[] ids, string name, string description) : base(ids, name, description)
+ Inventory Inventory => _inventory;
+ AddPath(Path path) : void
+ GetPath(string direction): Path
+Locate(string id): GameObject


user
player
movecommand
Location
Path
move north
execute ("move north")
GetPath("north")
Find Path
Return destination
Move to Location
update location

Program.cs:

```
using System;
using SwinAdventure;

class Program
{
    static void Main(string[] args)
    {
        // Step 1: Set up the player with their name and description.
        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);
        Console.WriteLine($"\nWelcome, {playerName}, {playerDescription}!\n");

        // Step 2: Set up locations and paths
        Location forest = new Location(new string[] { "forest" }, "Dark Forest", "A dark and ominous
forest.");
        Location village = new Location(new string[] { "village" }, "Small Village", "A peaceful village with
friendly folk.");
        Location mountain = new Location(new string[] { "mountain" }, "Snowy Mountain", "A tall,
snowy mountain peak.");
        Location lake = new Location(new string[] { "lake" }, "Crystal Lake", "A clear, sparkling lake.");

        // Connect locations with paths
        forest.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.North, village));
        village.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.South, forest));
        village.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.East, mountain));
        mountain.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.West, village));
        mountain.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.North, lake));
        lake.AddPath(new SwinAdventure.Path(SwinAdventure.Path.Direction.South, mountain));

        // Step 3: Place the player in the initial location
        player.Location = forest;

        // Step 4: Add some items to the player’s inventory
        Item sword = new Item(new string[] { "sword" }, "Sword", "A sharp, shiny sword.");
        Item shield = new Item(new string[] { "shield" }, "Shield", "A sturdy wooden shield.");
        player.Inventory.Put(sword);
        player.Inventory.Put(shield);

        Bag smallBag = new Bag(new string[] { "bag", "small bag" }, "a small bag", "A small leather
bag.");
        player.Inventory.Put(smallBag);
        Console.WriteLine("A small bag has been added to your inventory.\n");

        Item gem = new Item(new string[] { "gem" }, "a gem", "A shiny, valuable gem.");
        smallBag.Inventory.Put(gem);
        Console.WriteLine("Item 'gem' has been placed inside the small bag.\n");

        Item torch = new Item(new string[] { "torch" }, "a torch", "A torch that provides light.");
        forest.Inventory.Put(torch);
        Console.WriteLine("Item 'torch' has been placed in the forest.\n");

        // Initialize commands
        LookCommand lookCommand = new LookCommand();
        MoveCommand moveCommand = new MoveCommand();

        // Step 5: Main game loop
        while (true)
```

```
{
    Console.WriteLine("\nEnter a command (or type 'exit' to quit):");
    string command = Console.ReadLine();

    if (command.ToLower() == "exit")
    {
        break; // End the game
    }

    // Split the command into an array of words
    string[] commandWords = command.Split(' ');

    // Determine which command to execute
    string result;
    if (lookCommand.AreYou(commandWords[0]))
    {
        result = lookCommand.Execute(player, commandWords);
    }
    else if (moveCommand.AreYou(commandWords[0]))
    {
        result = moveCommand.Execute(player, commandWords);
    }
    else
    {
        result = "I don't understand that command.";
    }

    Console.WriteLine(result);
}

Console.WriteLine("Goodbye!");
```

```
    }
}

Location.cs:
namespace SwinAdventure
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _inventory;
        private Dictionary<string, Path> _paths;
        public Location(string[] ids, string name, string description) : base(ids, name, description)
        {
            _inventory = new Inventory();
            _paths = new Dictionary<string, Path>();
        }

        public Inventory Inventory => _inventory;
        public void AddPath(Path path)
        {
            _paths[path.direction.ToString().ToLower()] = path;
        }

        // Method to retrieve a Path based on direction
        public Path GetPath(string direction)
        {
            _paths.TryGetValue(direction.ToLower(), out Path path);
            return path;
        }

        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            return _inventory.Fetch(id);
        }
        public string AvailablePaths()
        {
            if (_paths.Count == 0)
            {
                return "There are no paths from here.";
            }

            string pathList = "You see paths leading to:\n";
            foreach (var path in _paths.Values)
            {
                pathList += $"- Move {path.direction} to {path.Destination.Name}.\n";
            }
            return pathList.TrimEnd();
        }

    }
}
```

Path.cs:

```
using static MiNET.Entities.Entity;

namespace SwinAdventure
{
    public class Path : GameObject
    {
        private Location _destination;

        private Direction _direction;

        public enum Direction

        {

            North,

            South,

            East,

            West,


        }

        public Path(Direction direction, Location destination)

            : base(new string[] { direction.ToString().ToLower() }, direction.ToString(), $"A path to the
{direction}")

        {

            _direction = direction;

            _destination = destination;

        }


        public Location Destination => _destination;


        public Direction direction => _direction;

    }

}
```

Movecommand.cs:

```
namespace SwinAdventure

{

    public class MoveCommand : Command

    {

        public MoveCommand() : base(new string[] { "move", "go", "head", "leave" }) { }
        public override string Execute(Player player, string[] text)
        {
            if (text.Length != 2)
            {
                return "Where do you want to go?";
            }

            string directionString = text[1];
            Path path = player.Location?.GetPath(directionString);

            if (path == null)
            {
                return "You can't go that way.";
            }

            player.Location = path.Destination;



            // Show the current location description and paths
            return $"You move {directionString} to the {path.Destination.Name}.\n\n" +
                   $"{path.Destination.GetFullDescription()}\n\n" +
                   $"{path.Destination.AvailablePaths()}";
        }
    }
}
```