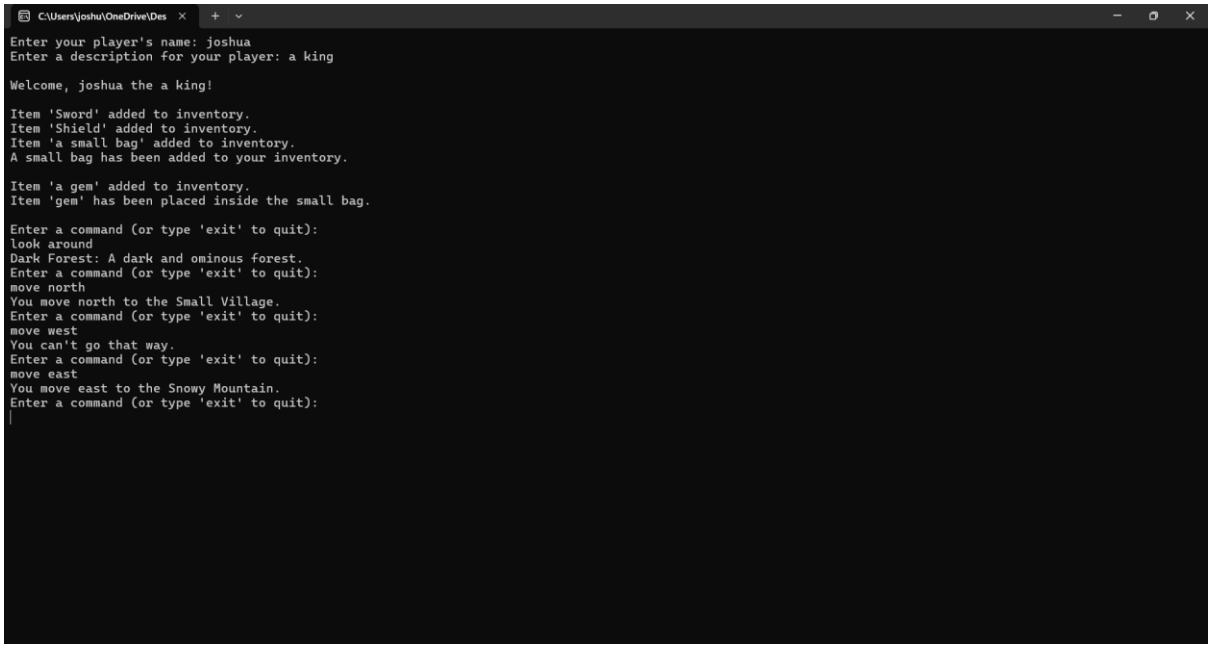
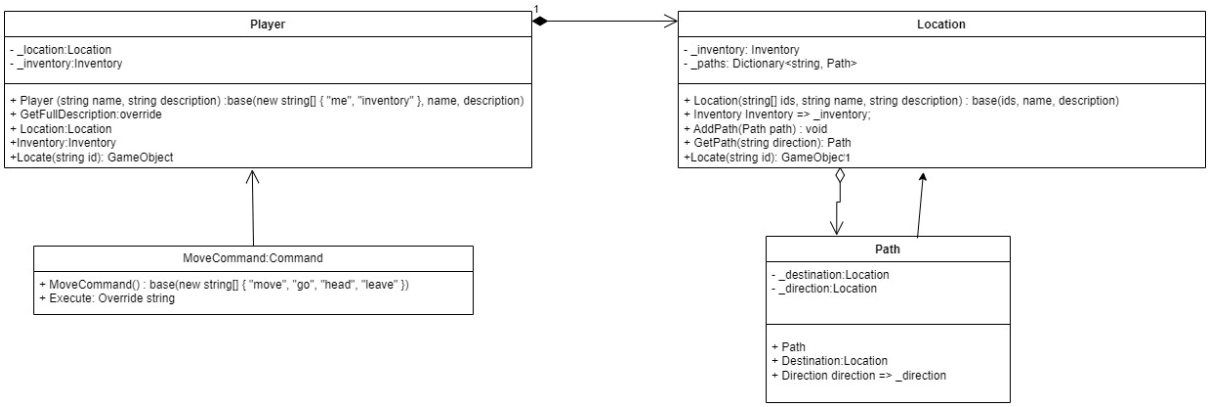
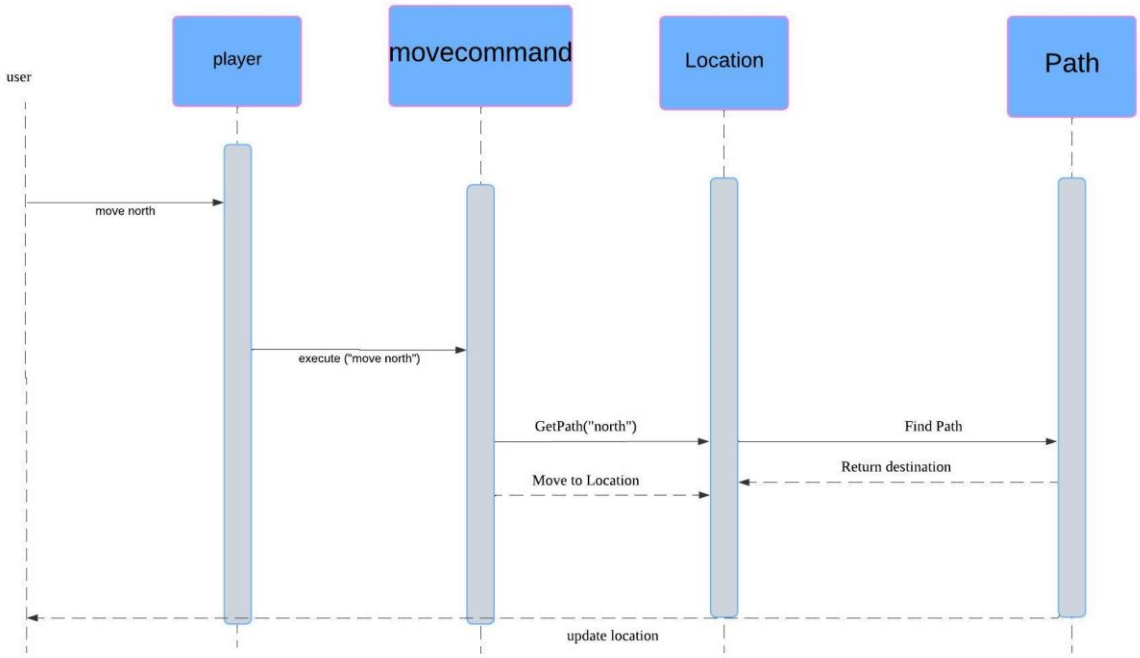

Test
Duration
Traits
Error Message
BagTesting (5) 173 ms
CommandProcessorTesting (4) 789 ms
InventoryUnitTesting (5) 159 ms
ItemUnitTesting (3) 153 ms
LocationPathTesting (2) 165 ms
LocationTesting (3) 155 ms
LookCommandLocationTesting (5) 178 ms
LookCommandTesting (8) 158 ms
MoveCommandTesting (3) 160 ms
PathValidationTesting (1) 159 ms
PlayerLocationTesting (4) 147 ms
PlayerUnitTesting (5) 161 ms
UnitTesting1 (6) 156 ms
Run
Debug
Group Summary
BagTesting
Tests in group: 5
Total Duration: 173 ms
Outcomes
5 Passed


C:\Users\joshu\OneDrive\Des
Enter your player's name: joshua
Enter a description for your player: a king
Welcome, joshua the a king!
Item 'Sword' added to inventory.
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.
Enter a command (or type 'exit' to quit):
look around
Dark Forest: A dark and ominous forest.
Enter a command (or type 'exit' to quit):
move north
You move north to the Small Village.
Enter a command (or type 'exit' to quit):
move west
You can't go that way.
Enter a command (or type 'exit' to quit):
move east
You move east to the Snowy Mountain.
Enter a command (or type 'exit' to quit):





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} the {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");

// Initialize the CommandProcessor with all commands
CommandProcessor commandProcessor = new CommandProcessor();

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

   Console.WriteLine("Enter 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(' ');

   // Execute the command using CommandProcessor and display the result
```

```
            string result = commandProcessor.ExecuteCommand(player, commandWords);
            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);
        }
    }
}
```

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;
            return $"You move {directionString} to the {path.Destination.Name}.";
        }
    }
}
```