THE WONDELAND SAGA: The Final Showdown

Emil Paul: 3698580

DESIGN DOCUMENT: ASSIGNMENT 4

I created the game in such a way that it navigates through Wonderland, collects all the treasures, defeats the Evil Queen, and finally escapes from Wonderland Successfully. The game runs based on player commands and needs to go through a bunch of locations and finish riddles, trades, and character interactions. The game includes special conditions that the player needs to satisfy to progress through the game, such as having specific items to pick up things and enter places and solving riddles and quizzes to obtain crucial items. As for Gameplay, it is completely driven by player commands that determine actions taken within the game world. Users can explore locations, pick up things, solve riddles, and interact with characters. To finish the game, the player needs to explore the entire wonderland, defeat the evil queen, and finally loot all her treasury and reach the Safe room to win the game.

As for design elements, the game's core mechanics involve exploration, Inventory management, puzzle solving, and character interaction. Firstly, Exploration involves moving along a wonderland, each location has a unique description, characters interaction if present, and available actions. The whole map is designed in such a way with a series of interconnected locations with exits leading to new areas. I also created a pictorial Map and attached it with a Game folder, to get through the game with ease, with available exits and items. Inventory Management is another important aspect, certain items are necessary to overcome obstacles or unlock new locations. Players can view the inventory and strategically decide when to use or trade items. As for puzzle solving, I included riddles at specific locations, where solving puzzles is one way to acquire important items in the game. Characters in my game act as guides, ask riddles, and offer trades. Character interaction is necessary to move forward in the game. The game also lets the user know about what is missing when trying to enter or pick up items that have special conditions. The game concludes when the player collects all the treasures after defeating the Queen and reaches the Safe room, To do that player must defeat the mighty dragon that's guarding the queen's palace. Players may fail to progress if they do not solve the necessary puzzles or collect required items, also players can give up or exit the game at any time, which results in a loss and a general exit message.

Class: **WonderlandGame**

This is my main class that initializes and runs the game. It mostly handles setting up locations, items, and characters from text files and controls the game loop that listens to player commands and updates the game accordingly. This is the brain of my game, coordinates and sets up the game world along with all interactions.

- start() :Starts the game, displays game headers and descriptions, manages game loop, updates current location based on player input
- initializeLocations() =Loads locations,exits,items and actiosn from external text files, also sets up the starting location for the game (Rabbit Hole)
- findLocationByName(ArrayList<Location> locations, String name): Searches for a location by its name within a list of locations and returns the corresponding Location object if found.
- printLocationDescription(Location location): Prints the description of the current location. If the player has visited the location before, it gives a shorter description. It also prints the characters found at that location.
- getItemNames(Location location): Returns a formatted string of item names found at the current location. If no items are present, it returns "None."

Class **Location**

This class represents rooms/locations in my game. It handles the exits that connect locations and add items or characters if present. This class is responsible in storing names and descriptions of a location, maintaining a list of items and characters present there, manage exits to other locations. Also has methods to add and remove items /characters from the location.

- getName(): Returns the location's name.
- getDescription(): Returns the location's description.
- getExit(String direction): Returns the location in the specified direction.
- addItem(Item item): Adds an item to the location.
- removeItem(Item item): Removes an item from the location.

Class **Item**

This class handles any item a player can pick up, interact with, and store in inventory. These items are essential in progressing along the game. This class stores all names and descriptions of the item and has methods to retrieve the item’s name and description for display purposes.

- getName(): Returns the item's name.
- getDescription(): Returns the item's description.

Class **Character**

Controls all NPC characters that a player interacts with each character has a unique name and description/introduction. It stores characters' names and descriptions and provides a method to retrieve character names and descriptions for display during interactions.

- getName(): Returns the character's name.
- getDescription(): Returns the character's description.

Class **Inventory**

The Inventory class manages the player's collection of items. It tracks what the player has collected and provides methods for adding, removing, and checking items. Stores a list of items while playing, allows items to add and removed from inventory, contains a method to check if the specific item is in inventory for game progression, and finally displays the inventory/stash when requested (command= 'inventory')

- addItem(Item item): Adds an item to the inventory.
- removeItem(Item item): Removes an item from the inventory.
- getItems(): Returns the list of items in the inventory.
- hasItem(String itemName): Checks if the inventory contains a specific item.
- showInventory(): Displays all items in the inventory.

Class **Control**

This class is responsible for handling player commands, process commands and updates the game based on it. This class pretty much handles the logic of the game. Apart from handing commands (go,take,inventory,exit,help) it also controls special cases like checking a specific item is in inventory to continue .Also controls the character interactions, enforces game rules, provides feedback based on player actions and facilitates game's help menu with information about controls and exit command that quits the game immediately following a goodbye message.I included my riddles, dialogues, different scenarios here.

- processCommand(String command, Location currentLocation): Processes player commands and updates the current location.
- findItemInLocation(String itemName, Location location): Finds an item in the current location.

Text Files

I used a total of 5 textfiles that is read and handled in the beginning of the game by my main class.These files keep the game more easy to update and clean

Locations.txt =Each line contains names, and descriptions for all locations in the game.

Exits.txt= Each line involves 2 locations followed by the direction of exit.This textfile control all my exits of the locations. I created an extra textfile just for exits to load exits easily and clearly

Items.txt = List of all items in game along with descriptions and location where it belongs.Each line defines a specific unique item

Characters.txt = Details about character's name, description and location they inhabit belongs in this textfile

Actions.txt = This contains possible player command available in the game, this is printed out and displayed in the beginning of the game.

Game Flow

Here I explained the shortest way to win the game discarding extra scenarios and items in the game. Refer to the Picture map provided with the game file for navigating easily

- Player starts in front of Rabbit Hole
- Enters Garden with the only exit possible (south)
- There are items like ‘mandrake root’ here you need to pick up for progression (Trade for Invisibility cloak)
- Then you go to Mad Hatter’s tea party (west) and try to pick up Golden Key (that unlocks the chest for Excalibur sword). Here you need to answer 2 riddles to obtain the Golden Key
- Hogwarts location contains an Invisibility cloak that you need to steal the Amulet from the Cheshire Cat location (pickup without being seen), But you can only pick up if you have Mandrake Root, which is traded for an Invisibility cloak (Mandrake root gets removed from inventory when traded)
- You can Steal the Amulet from the Cheshire Cat location if you have an Invisibility cloak in your inventory (which you need for entering Dark Forest)
- Past Dark Forest, there are Forbidden Ruins that contain a chest with an Excalibur sword (you need a Golden key from Madhatter to obtain this)
- When you enter the Queen’s court with Excalibur you stay the Dragon and defeat the Queen, you can't kill the dragon/enter the palace without having Excalibur in your inventory
- Vault key obtained from the Queen’s court after this lets you enter the Treasury with 4 items
- You need to make sure you pick up ALL 4 ITEMS and enter Safe Room to win the game (WIN condition)

Along the journey, I introduced characters like Cheshire Cat, Mad Hatter for riddles, Oracle at Forbidden Ruins for guidance, and Harry Potter in Hogwarts for trading stuff to make the game more interesting. Apart from this, there are numerous collectible items along the way you can collect, that don’t control the WIN condition of game but add more interactions to the game

To win, the player must collect all four treasures from the Royal Treasury and store them in Safe Room. I created the storyline, items, and characters based on several favorite movies other than Alice in Wonderland as well to make it more personal. Overall, this “THE WONDERLAND SAGA: The Final Showdown” game contains fantasy, riddles, unique character interactions and items, and outwit characters to succeed. The game is more engaging and fun for users who love fantasy movies and mythological items