Name: Lucas Monico

Athabasca ID: 3572445

Course: COMP 206

Date: 25 February 2025

**ASSIGNMENT 4 – DESIGN DOCUMENTATION**

This document is to highlight and outline the general flow of this program. I decided to try and emulate some other projects I researched and use headers and source files for this program.

Header files act as declaration for functions and methods for the classes and the source files (.cpp) are the definitions of those declarations to flesh out the actual functionality of the programs and classes.

**Source files:**

* main.cpp : Self explanatory, this is the main file that creates a Game class object and calls the startGame() method then begins the while loop for user input commands that feeds to the game method userInput()
* Game.cpp : By far the biggest chunk of this project that holds the Game class and creates vector objects of the Character, Item, Location, and Inventory classes and works in tandem with the Actions.hpp to filter action commands and directional keywords
* Characters.cpp : Holds and defines the Character class definitions for the getters and setters
* Locations.cpp : Holds and defines the Location class definitions for the getters and setters
* Items.cpp : Holds and defines the Item class definitions for the getters and setters
* Inventory.cpp : Holds and defines the Inventory class definitions for a few specific methods to add, decrease, get, and print the items vector objects within the inventory class – works directly in tandem with the Item class by holding and manipulation an Item vector object

**Header Files:**

I won’t go through each individual one like with the source files but essentially the declare their respective classes. They organize the access specifiers, variables, getters, setters, helper methods, and other miscellaneous methods. The Actions.hpp is a special case as it does not define normal classes but simply defines the two enum classes for Action commands and Direction commands.

**Text Files:**

There are two types of text files in this program. Class specific defining and event/exposition text files.

The class specific define the individual objects that will be the assets used within the game; such as individual item, character, and location objects. They go into detail line by line for relevant game behaviors for their class vector object. They are loaded in through the Game.cpp source file and inserted into their respective vector class object.

The second type of text file is the exposition text file that are used for longer dialogues and events. They are triggered in several ways; through travel between locations, through talking dialogues, through using of items within the inventory, or through the consuming of items within the inventory.

**Game Flow:**

The player controls the character Alice from the novel Alice In Wonderland written by Lewis Carroll and their goal is to explore the lands – starting by following a curious character called the White Rabbit. The idea is to follow the novel closely, in a linear fashion. This game is more of a visual novel style and therefore has no combat and lacks ‘emergent gameplay’ – mostly because this is supposed to be a small project and more importantly, because I’m way too inexperienced to code that sort of reactivity in a game of this scale.

The game is a basic text adventure where the player navigates using directional commands through the terminal. There are also several other command key words to contextual help and interactivity. To list them off the player has access to these commands:

* HELP: Prints to terminal a help guide to explain all the action commands.
* INSPECT: Can be used for contextual descriptions; like directions, characters, objects, etc. Used in tandem with an object name or direction.
* TALK: Command to talk with a character. Used in tandem with a character name.
* TAKE: Command to pickup and add an object to the player’s inventory.
* USE: Command to use an object within the player’s inventory
* MOVE: Command to navigate in a specific direction to change location. Used in tandem with a direction word.
* INVENTORY: Prints off the player’s currently held inventory items to the terminal
* CONSUME: Command to consume an item object within the player’s inventory
* QUIT: Command to quit the program effectively ending the game.