CS100 – Project Class Diagram Description

1. Interface

The *Interface* class is what our player will see when first running the program. In it, they will be asked to select an option such as starting a new game, loading a saved game, save their current game when given the chance, and the option to quit. The interface will be used when the program is first ran, but also throughout the game whenever the player wants to "pause" their current game and access these options again.

- + Menu() – This is the main function of the interface class as it will host the other options available to the player.
- + Load() – The load function will allow the player to load a previously saved game and will notify the player if they try to do that when there is no saved file to be loaded.
- + Save() – This function will let the player save their current playthrough so that they can pick up here they left off at any time. We plan to do this by having set checkpoints throughout the game, such as at the beginning and end of each floor, and then writing the players current character data into a text file that is to be read whenever they try to load a game.
- + Quit() – Simple function to let the player quit the game for any reason.

2. Level Design

The *Level Design* class is the foundation for our game. Our floors, or dungeons, will be derived from this class and will be uniquely implemented as the functions inside *Level Design* will be virtual, allowing us to alter the events and breaks for each floor as we see fit. We plan to have 4 events and breaks per floor, with events serving as potential fight encounters or anything that pushes the game forward, and breaks being the intermission between encounters where information relevant to the story will be given as well as time for the player to decide on what loot to take after they've felled their foes.

- + Prologue() – This function serves to give the player a little information about the floor they are about to enter while also providing exposition for the story.
- + Event() – These functions will serve as the interactive portions of our game by plunging the player into combat or asking them to make important decisions.
- + Break() – These functions will act as the aftermath of encounters, giving the player a chance to reflect on what has just happened. Time for looting and leveling are some of the things the player will be able to do when in a break.
- + Epilogue() – The epilogue for each floor will be a summary of the events that unfolded in the current floor. The total number of items gained/lost will be displayed as well as the total amount of XP gained. The player can also save their progress once the epilogue is reached.

3. Floors

The *Floor* classes will be the dungeons of our game. They will be derived from the *Level Design* class and will have their own events and breaks to make each floor unique. The planned number of events/breaks per floor is 4 but this number could change depending on how we need each floor to play out in order for out story to progress.

4. Player

The *Player* class will hold the users character data and will be the main source of character customization for our game. We are going to have stats that the user can adjust at the start of every playthrough and these will allow for variety in gameplay and character builds. Because our game will be played entirely through the terminal, we were limited on the traditional forms of character customization such as appearance and character voice, but we allow the user to enter a name and pick a race, each with their own benefits.

- + getStats() – Returns the players current stats and any stat altering effects.
- + getName() – Returns the player character's name.
- + getRace() – Returns the players chosen race.
- + getLevel() – Returns the users current level.
- + getExp() – Return the users current XP and the amount needed to level up.
- + setLevel() – Function to be used when the user has accumulated enough XP to level up.
- + setName() – Sets the user character's name.
- + setRace() – Sets the user character's race.
- + setStats() – Allows the user to adjust stats when creating their character as well as when leveling up.
- + setExp() – Increases users XP count after fights.

5. Enemy

The *Enemy* class will be used whenever we need to create an enemy for the user to fight during encounters. They will have their own HP, XP and item dropped when defeated, and stats like armor, damage, and speed.

- + setHP() – Function to adjust enemies HP when user has attacked.
- + getDrop() – Returns potential item to be dropped when enemy is defeated.
- + getExp() – Return XP to be given to user once defeated.
- + getHP() – Returns the enemies current HP after each user attack.