Status Summary

- Team Member

  Scott McCall, Sitong Lu, Adithiyya Kishoore

- Project Name

  Warped Souls - A 2D Dungeon Game

- Work Done

  We finished the general UI layout, including the board that currently contains 12 x 12 cells in it, and interactable buttons (each with a connected method to operate once been clicked) corresponding to movement, inventory, and ending the player's turn.

  Scott was responsible for developing the player's inventory and allowing interactions with the board to update the inventory.
  Sitong was responsible for making each cell on the board clickable and developing methods to allow each cell on board to change their corresponding icons (either background or background with character standing on it) at the right time.

  Adithiyya was responsible for the basic setup, creating the fundamental UI components that went into making the layout of the game, developing the character class, enemy class, movement and attack strategies patterns (enemy AI).

- Changes or Issues Encountered

  One significant change made between Project 5 and now was that we are no longer directly implementing an MVC pattern as the way to display and update the game board. The reason for this was due to simplicity, in that the non-MVC implementation we ended up using was more intuitive, and we were struggling to develop ways to create controller algorithms that didn't end up being redundant or not add much functionality to the program. This was an instance where using a pattern initially seemed like a good idea, but we didn't want to use a pattern just for the sake of using a pattern when it wouldn't end up benefiting our overall design.

- Patterns

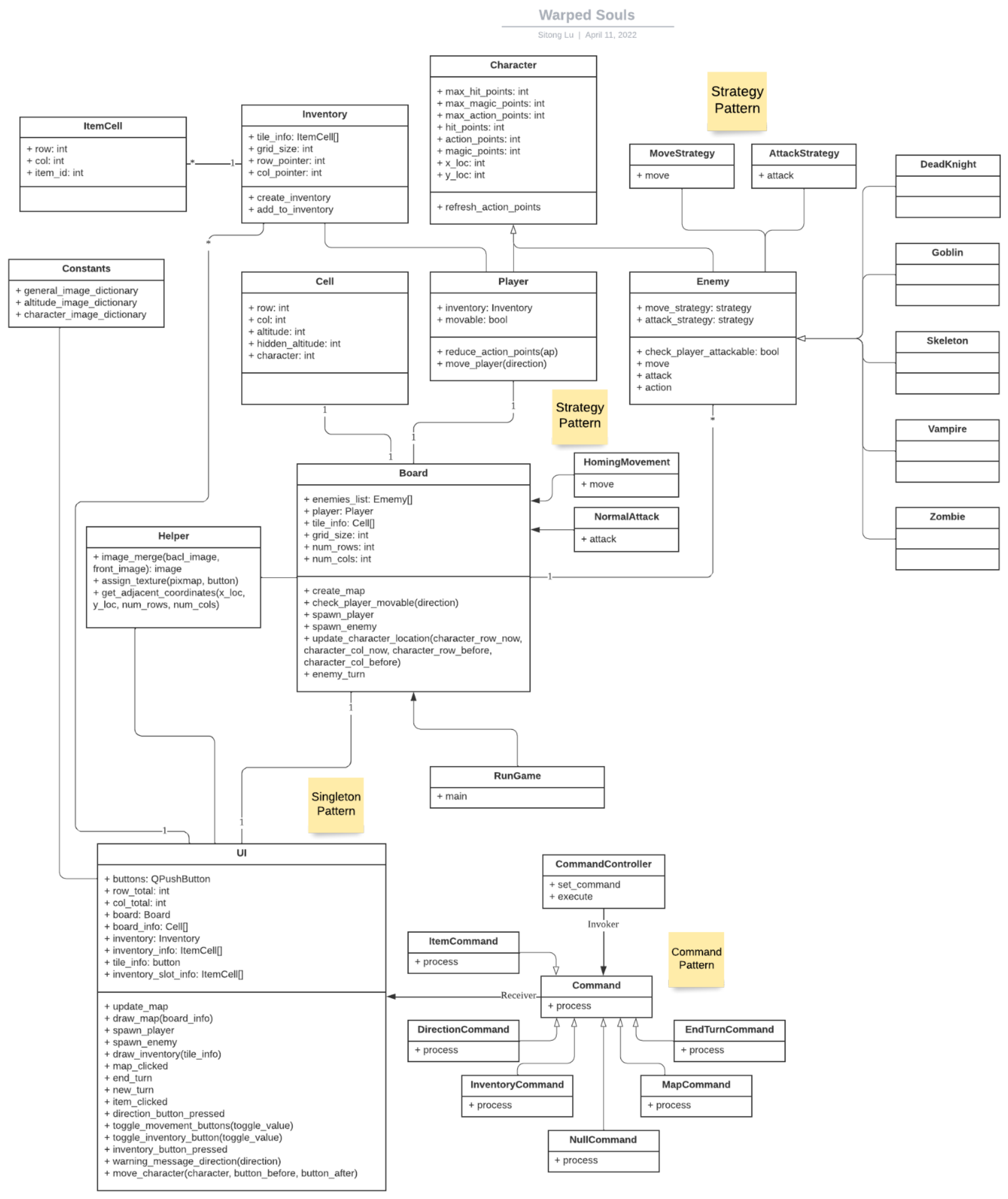
  Singleton Pattern: allows the game to get one and only one board during each playthrough. By applying this pattern, the game will save some storage space and even processing time from mistakenly generating extra boards in the back end.

  Command Pattern: allows the game to control all the commands in one method rather than having separated control all over the classes. By doing so, we can easily modify and add commands into our game.

  Strategy Pattern: allows the enemy to choose different movements and attacks based on their own characteristics. By doing so, we can easily modify and add all kinds of variability into our game.

Class Diagram

Link to the hi-res version: https://drive.google.com/file/d/1v_22hs5CoPWLKLdlAOHfcOqAOZE-vicq/view?usp=sharing



Plan

In terms of how much progress we have made towards implementing our initial vision for Project 5, we are roughly a little bit more than halfway there. Most of the framework needed for our game has already been successfully implemented, and much of the remaining work that needs to be done will simply be expanding off of this basic framework.

For the Project 7 delivery, Scott plans on implementing a shop to purchase items, as well as an Observer pattern to keep track of game statistics, such as enemies killed, items collected, and more. Sitong is planning on applying **Command Pattern** into the game, rather than simply creating a bunch of methods and connecting each of them with a corresponding button. Adithiyya is planning on applying **Factory Pattern** into the game to generate enemy characters on board, and the corresponding algorithm for each enemy's movement per turn.