**Pattern Description Document**

Dungeon System

The Dungeon class used the Singleton pattern to prevent more than one dungeon from being created. The Dungeon has a Grid that contains Tiles, and each of the tiles can be randomly generated or have a static grid. For the dynamically created grid, I use a TileFactory to abstract away the logic for creating the random tiles from the Grid to the TileFactory. The corresponding UML for the Dungeon System is located in the DungonSystemUML.pdf.

Battle System

The Battle system makes use of the strategy pattern. Whenever a character takes their turn in the battle, the Battle system gets an object with the BattleAction interface from the game. This BattleAction object may have the character attack, use an ability, or an item depending on which BattleAction object is sent to the battle. This prevents a lot of messy, hard to update and read conditional logic and allows for easy modification, addition, and removal of actions that may be taken in battle.

# Character System

## Strategy Pattern:

## Factory Pattern:

## Template Method:

# Whole game

## Model, View, Controller: