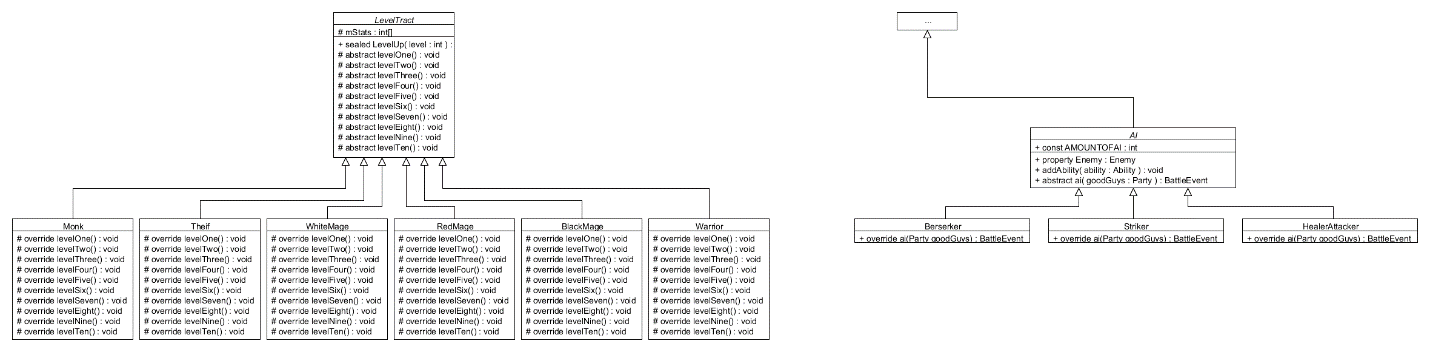
**Pattern Pattern Document**

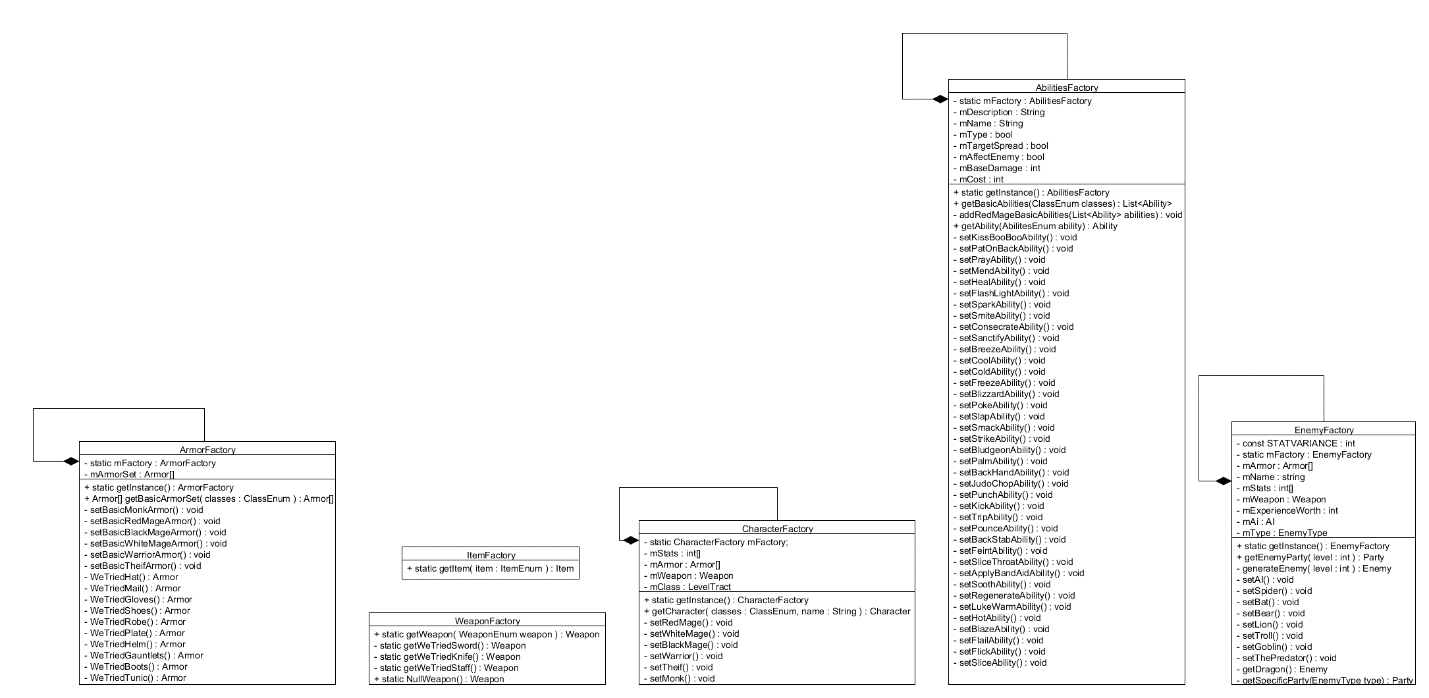
# Character System

## Strategy Pattern:



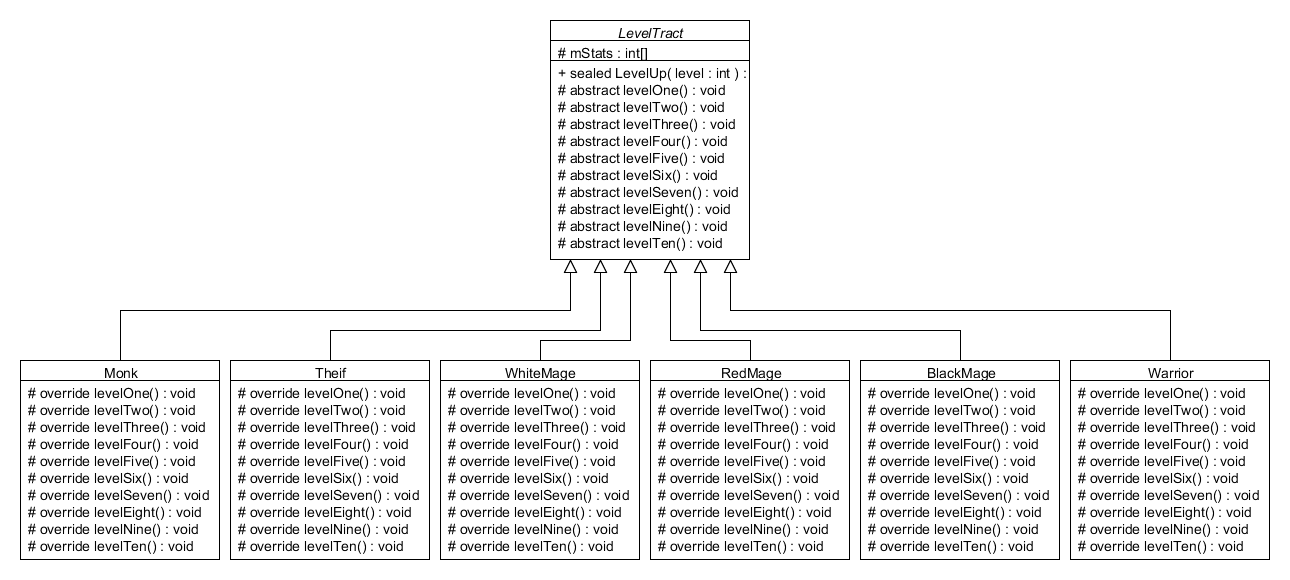
The strategy pattern is mainly used in the character system. All the player characters level, they just level in different ways, which strategy is perfect for. The enemies all have AI as well, they just have different ways of doing AI.

## Factory & Singleton Pattern:



Factories were used extensively for everything that needed to be created en masse. A database would have been helpful here, but these factories get the job done. They all also implement the Singleton Pattern, as you don’t need multiple factories.

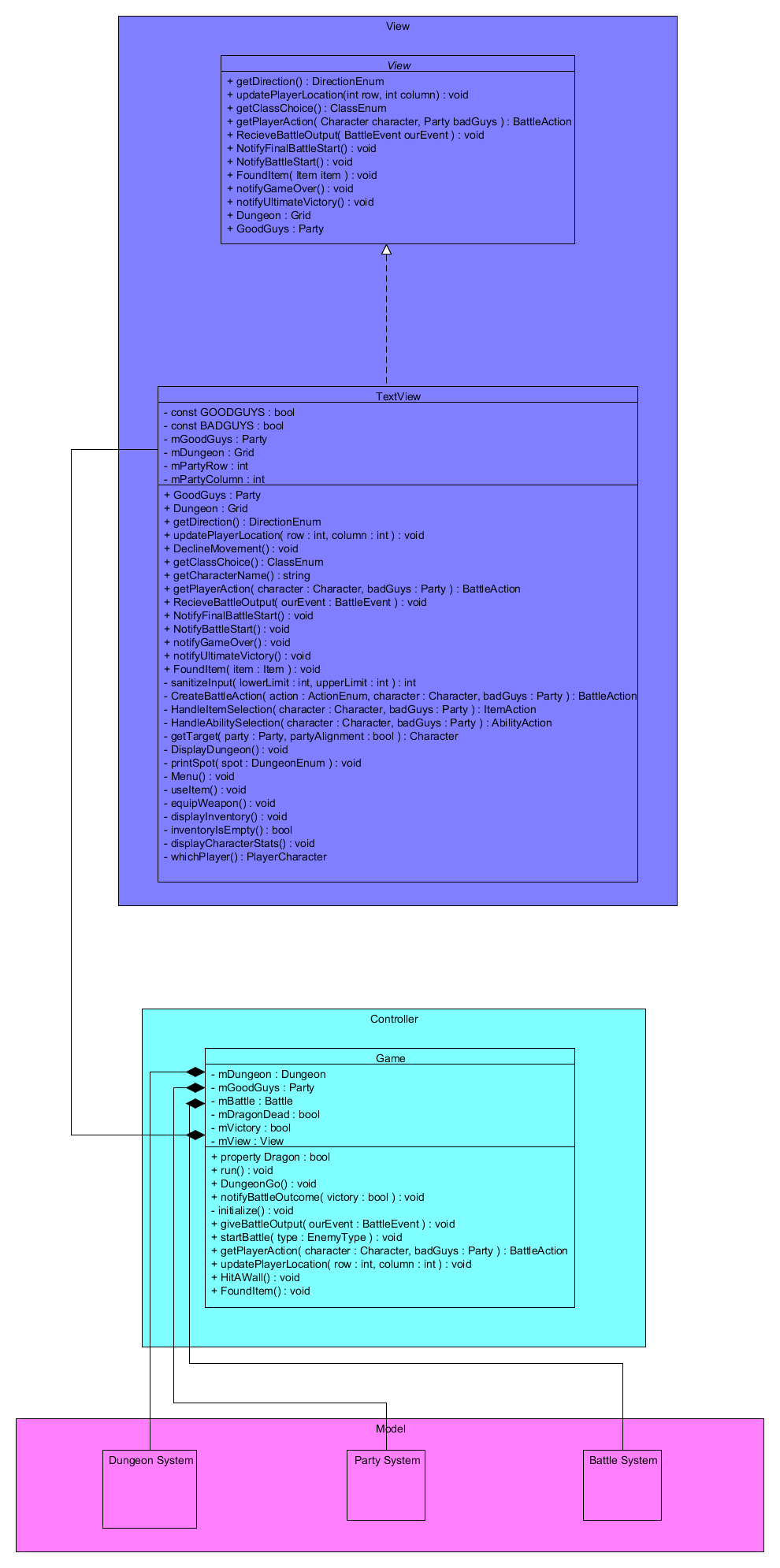
## Template Method:



The template method was used for leveling. All the template methods are called in the final LevelUp method, however they are defined in each individual class.

# Whole game

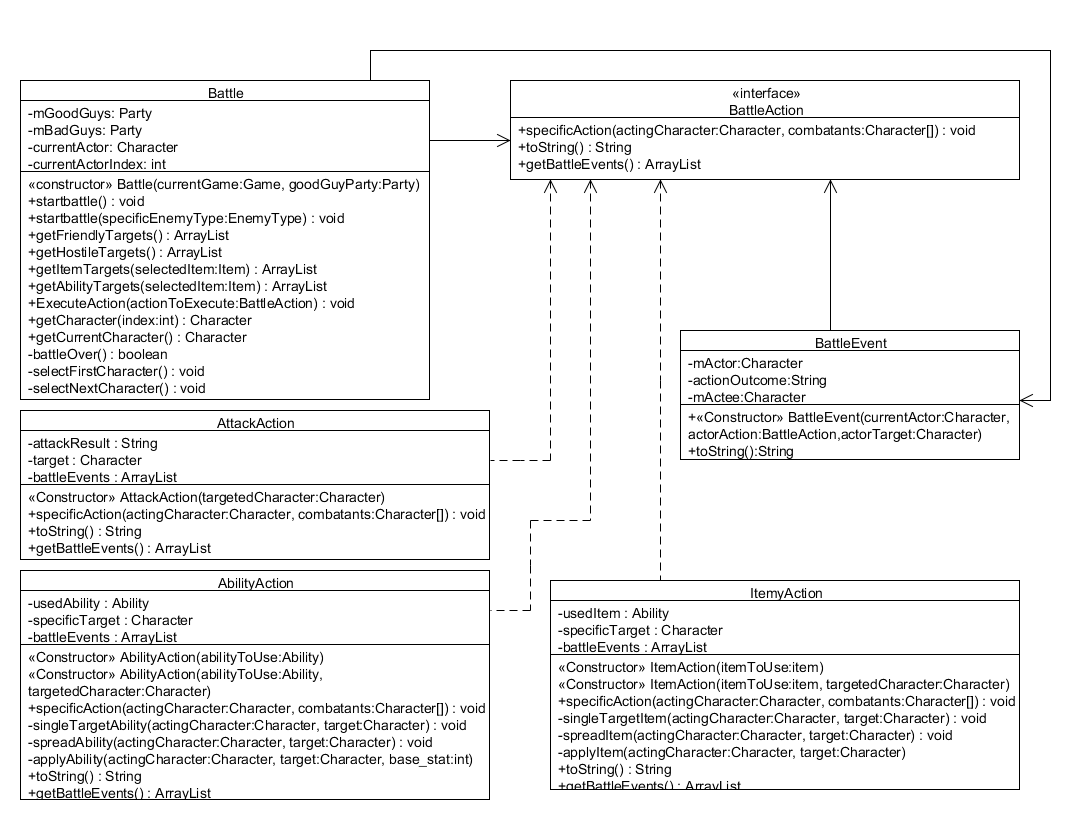
## Model, View, Controller:



The whole game implements model, view, controller. The model was the most complex and time consuming thing to create, however, once it was done, the controller and view came together rather easily thanks to this pattern.

# Battle System

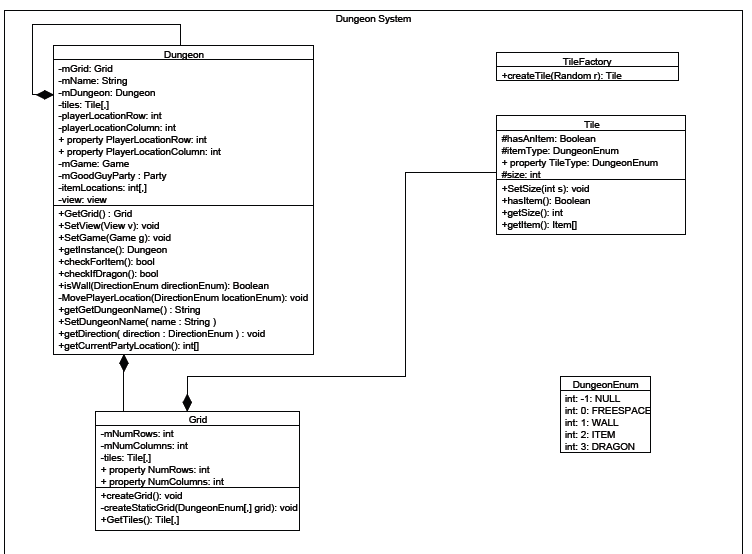
## Strategy Pattern:



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.

# Dungeon System

## Singleton & Factory Patterns



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.