Pokemon Battle Simulator

Project 7 Summary

Thomas Starnes and Brian Glassman

Summary

After having difficulties and troubleshooting TKinter, we decided to forego the option to switch out Pokemon from battle. Now the player must keep their Pokemon in the battle until it faints, upon which, it will automatically be switched out to the next Pokemon until none are left. For the same reason the stats and types for Pokemon and Moves are implemented in the backend but not displayed in the GUI. We implemented an opponent AI that is “just above stupid” in that it checks to make sure that it’s only using useful moves and has a sequence of preferred move types, but does not have any sort of forethought or planning. These were the only aspects of our initial design that were not implemented. The player is able to create their team of Pokemon each with moves chosen from a Pokemon-specific learnset, or use a pre-built demonstration team. Battles function as-designed, except the note above, with each trainer selecting a move to use and the effects occurring according to the original Pokemon games. When either trainer runs out of Pokemon the player is presented with a win/loss message, as appropriate.

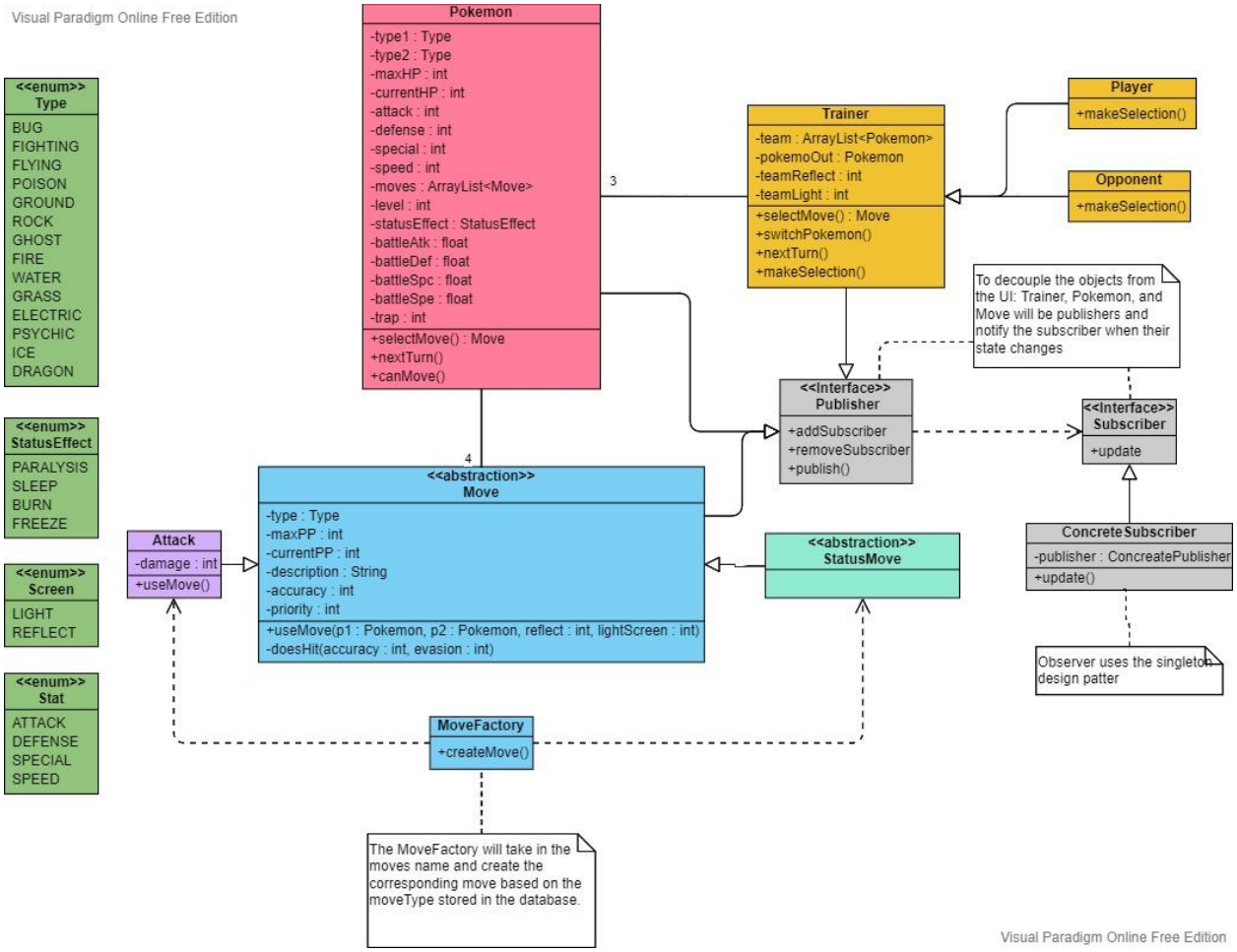
The only significant change from Projects 5 and 6 is the design and implementation of the GUI. For the Project 6 demonstration the GUI worked by creating a new window every time a move selection was needed, and returning the selected move on close. In the final version we have implemented an abbreviated MVC system, where the TKinter GUI functions as both view and controller. The GUI display is kept up-to-date with a combination of Publisher-Subscriber for simple tasks like displaying a selecting Pokemon’s name, and Observer-Observable for more nuanced information like displaying the Pokemon’s health and status effects during combat.

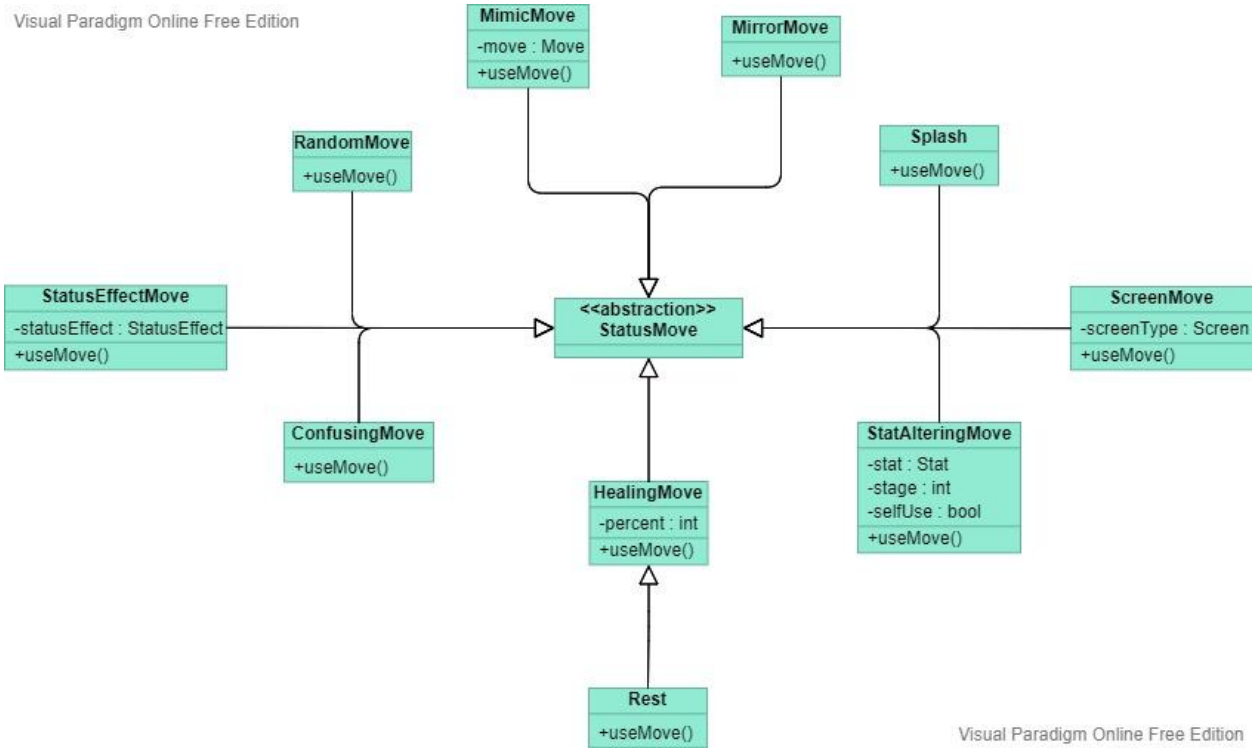
Class Diagrams

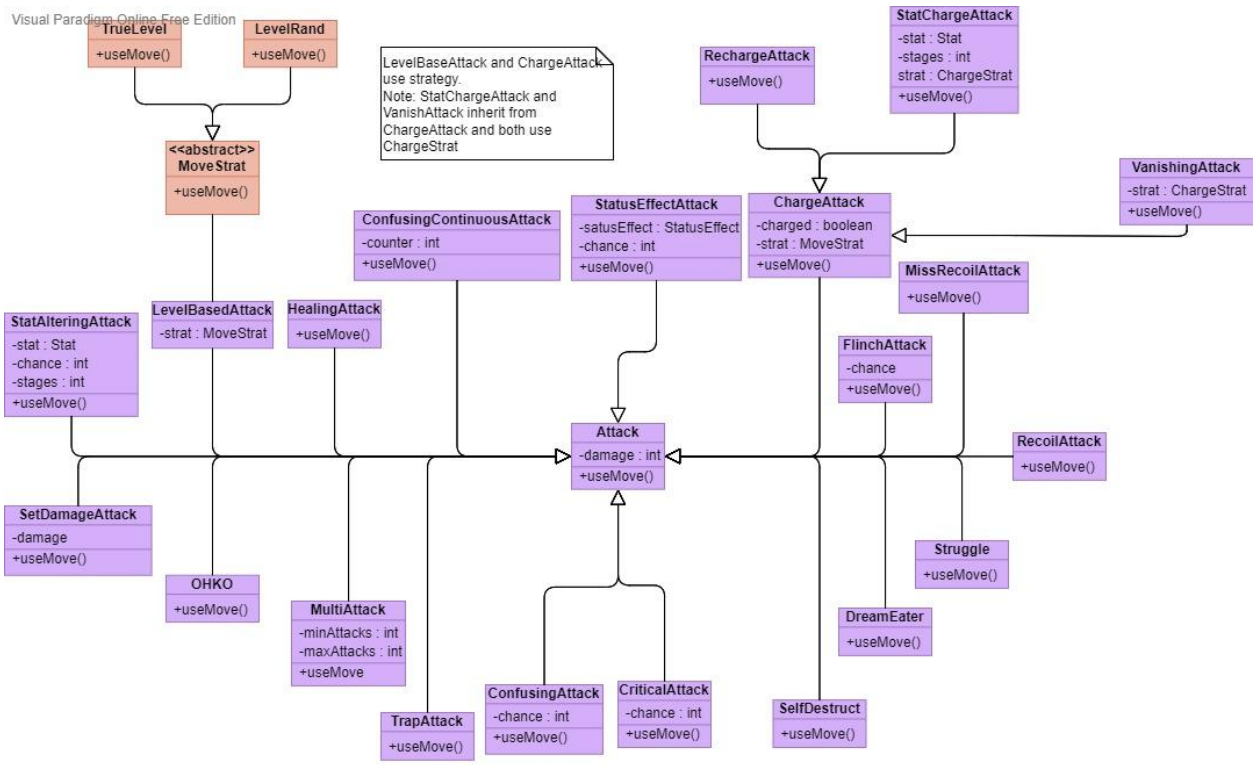
For the StatusMoves class diagram, we removed SwitchingMove and Transform. Since we removed the option for the trainer to switch out the Pokemon in battle, it didn’t make sense to keep SwitchingMoves like Roar, Whirlwind, or Teleport in the game. We also removed Transform from the game. We tried implementing it but it created too much overhead and the deadline was fast approaching. Removing Transform meant we had to remove Ditto from the game since that is the only move it knows.

For the Attack class diagram, we made RechargeAttack inherit from ChargingAttack since it didn’t make sense to have StatChargeAttack and VanishingAttack inherit from a class that uses strategy to distinguish between Charge and Recharge.

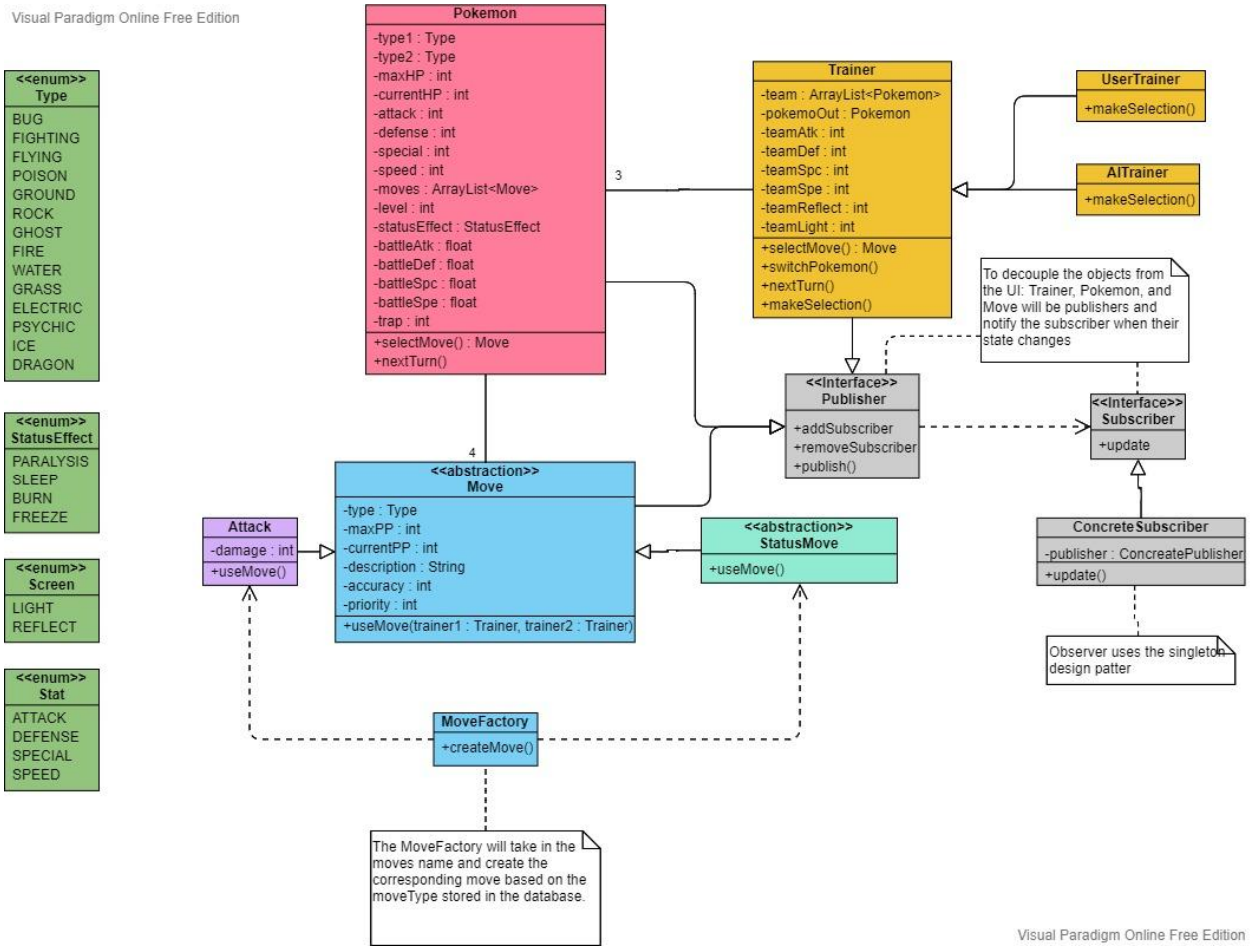
Project 7 Diagrams

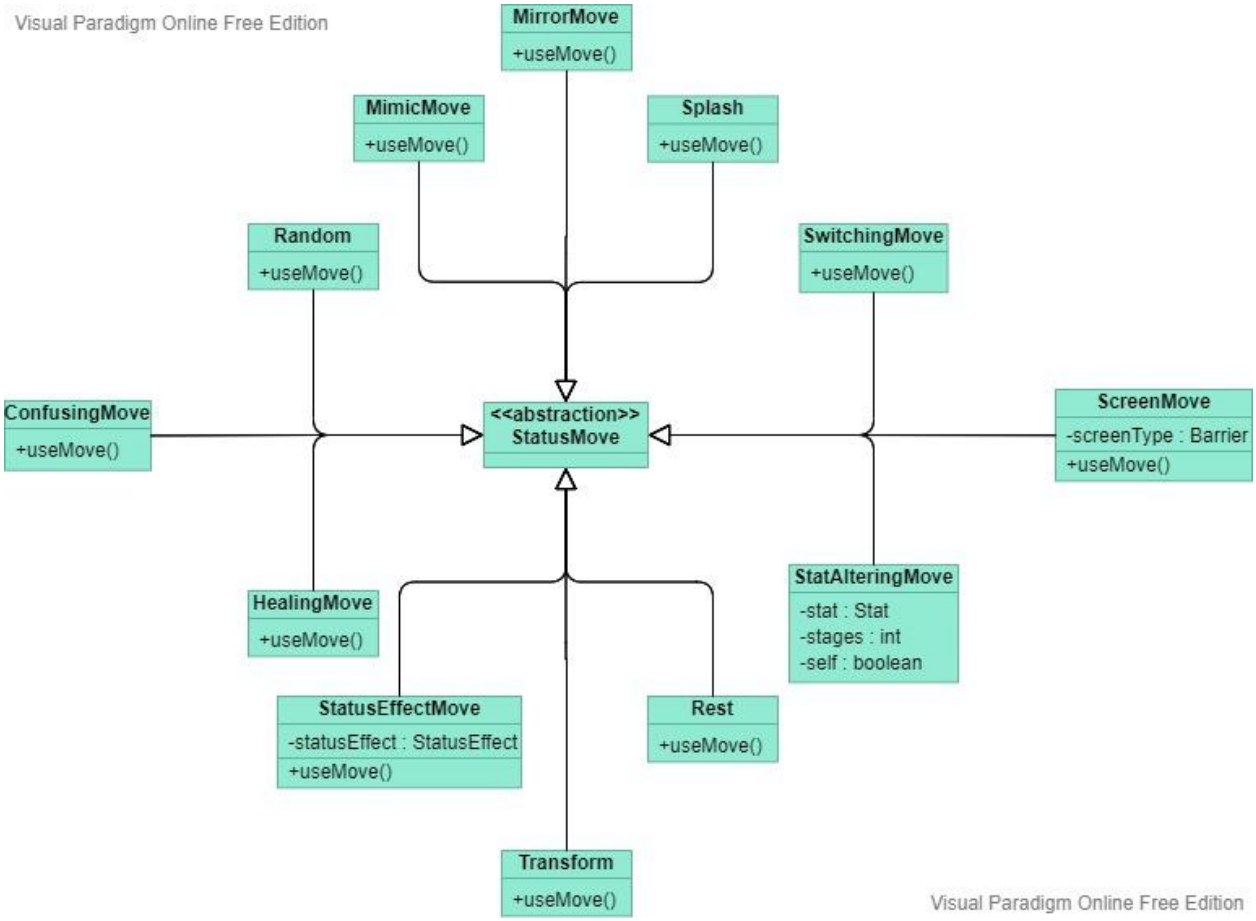


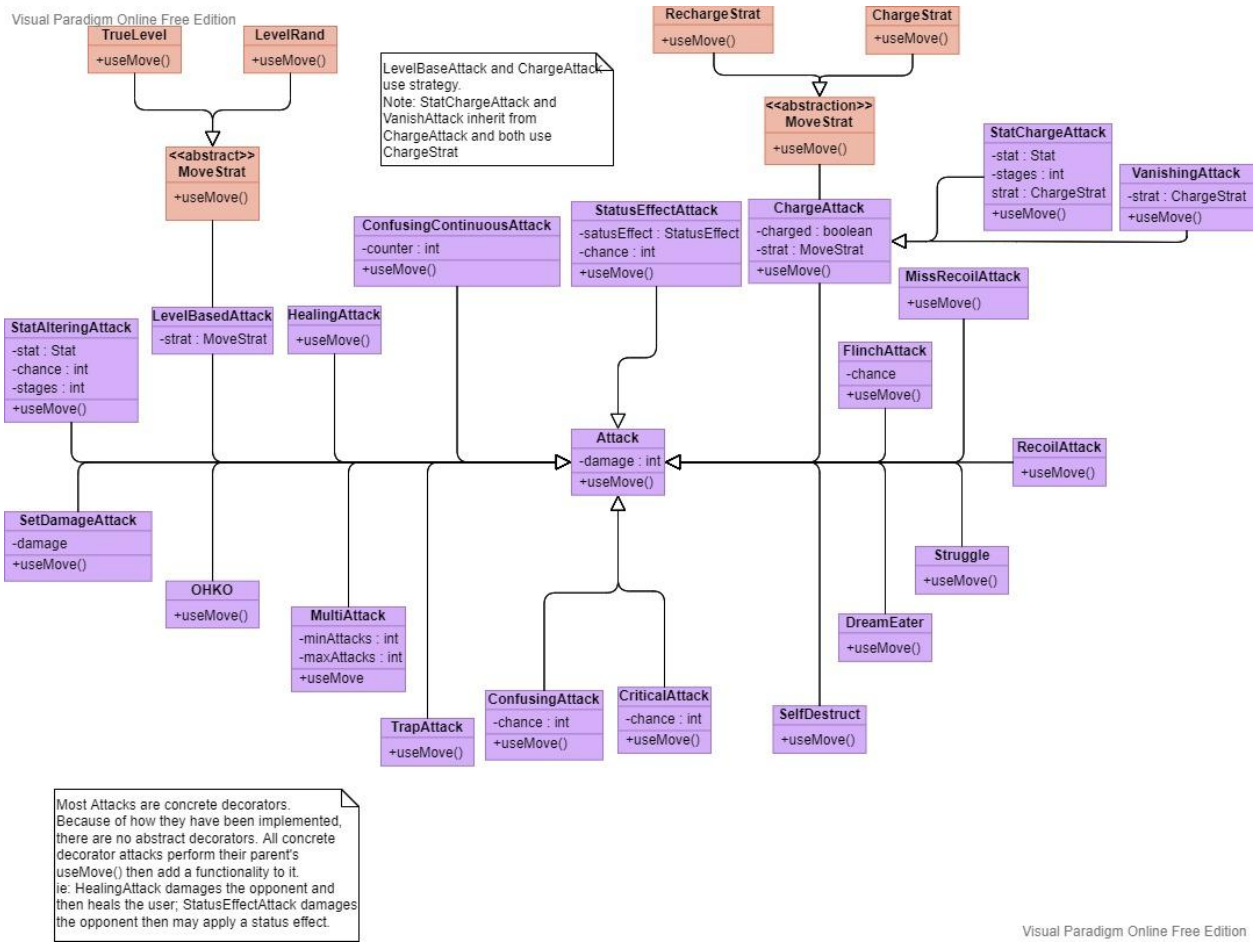




Project 5 Diagrams







Third Party Information

Since Pokemon is a popular game that has been around since the 90s, there are multiple websites dedicated to the mechanics and information on Pokemon. While coding, we used Bulbapedia (bulbapedia.bulbagarden.net), as a reference guide for the mechanics and calculations. For the data on the Pokemon, we downloaded a csv (https://gist.github.com/armgilles/194bcff35001e7eb53a2a8b441e8b2c6) and removed all Pokemon outside of the original 151. For the moves, we copied and pasted the data from PokemonDB to a csv (https://pokemondb.net/move/generation/1). For the learnsets of every Pokemon, we wrote a simple web scraper script to extract the data from Serebii.net into a csv (https://www.serebii.net/pokedex/001.shtml -- https://www.serebii.net/pokedex/151.shtml).

Numerous sources were used to help with coding, listed below and commented where used:

- https://www.pythontutorial.net/tkinter/tkinter-object-oriented-window/ For the methodology of inheriting from TKinter objects
- https://stackoverflow.com/a/3295463/14501840 For how to cause a TKinter window to perform custom actions on closing

- https://www.pythontutorial.net/tkinter/tkraise/ For bringing a TKinter window to the front of the stack so that it is shown and clickable
- https://www.pythontutorial.net/tkinter/tkinter-listbox/ For working with a TKinter listbox
- https://stackoverflow.com/questions/14910858/how-to-specify-where-a-tkinter-window-opens For always opening the window centered on the screen
- https://anzeljg.github.io/rin2/book2/2405/docs/tkinter Was used extensively as it is an easier source for TKinter information than the official documentation.

OOAD

1. One of the issues we ran into early, which we expected but did not fully understand the scope of, was the difficulty integrating the GUI library we were using with the other classes and functions we had created. By nature TKinter is object-oriented internally, but works best when the GUI itself functions as the sole means of interacting with the system. This is because TKinter's main loop blocks any other execution, so all operations have to be initiated by the GUI. Our original design concept was to have an MVC or Publisher/Subscriber system where a single controller would interface with both the opponent AI and the GUI. We were forced to modify this so that when a player clicks a button it performs the chosen action as well as causing the opponent AI to respond.
2. Another issue is one that seems to be common to game development, which is that many game mechanics require either tightly coupled classes or clever implementations to avoid this situation. In our case, the wide range of ways in which a Move can change a Pokemon's behavior made it difficult to decouple the Move and Pokemon classes. For example, a Move that causes the target to go to sleep requires the target Pokemon to change their behavior such that they no longer make a move on their turn but instead either sleep through their turn or wake up.
3. At the beginning of the project we analyzed the commonalities and variabilities within the problem space so that we could set up our classes in the way that would make future implementation work easiest. For example, Moves were divided into StatusMoves or Attacks, then further subdivided where there were multiple moves that had similar behavior, such as Guillotine and Horn Drill which are both one-hit knockout moves.