# Summary of Classes

During the previous Lab (Lab 2), I was able to already identify the classes that I needed in the game. I was also able to implement a modularized prototype. For this lab, I will be documenting the purpose of each class in the game. I will summarize its attributes and its operations.

## Character Class

This is an abstract class that represents the characters in the game. All characters has a name, a location to represent their position in the map, and a health to indicate whether they’re still alive or not.

The character class has an abstract method called “act”. This operation is to be implemented by classes that extends the character. In the game, there are 2 characters categorized as “Player” and “Pokemon”. This is where I will be applying inheritance and polymorphism. Players and Pokemon objects has the same attribute but they act differently. More of how they act will be discussed below.

## Player Class

A player class represents the character controlled by users. The game as 2 player objects. It was discussed awhile back that it has to implement its own “act” operation. For the player class, the “act” operation will allow the user to input commands from the keyboard then the program will execute those command. The player can perform “act” commands like moving the character, scanning for items, picking up items, using items, attacking Pokemon, and so on.

What makes the player different from the Pokemon character is that a player can additionally carry a list of items (to be discussed below).

## Pokemon Class

A Pokemon class represents also a character but controlled by the computer. The game has only 1 Pokemon object in which players need to capture to win the game. As it was discussed in the Character class, the Pokemon will implement its own “act” operation. The Pokemon acts by randomly moving from one location to another or attacking a player object if meets them.

## Item Class

The item class is an abstract class representing any item that can be used in the game. These items can be picked up by the user and put it in their inventory. The only attribute the item has is the name and has an abstract method called “act” to be implemented by inheriting classes.

Technically, the only item in the game is the Pokeball. I decided to make an abstract class item for it just in case in the future I will be implementing other types of item like Candies, Healing Potions, Pokedex, and so on. The advantage of abstracting it is I can easily extend the code by adding new types of item without having to re-code other related classes.

## Pokeball Class

The Pokeball class is an item class in which it acts on character. When the Pokeball is acted on a Pokemon, it will attempt to catch it with a success rate depending on the health of the Pokemon. The lower the health of the Pokemon, the higher the chance I can be captured. A player can carry multiple Pokeballs but each Pokeballs can only be used once so the Player has to pick as many Pokeballs as they can.

## Location Class

The location class represents a spot in a world (this will be discussed below). The location basically has a name and a description which is shown to a player when they arrived/landed to it. In a location, it has a list of exit locations in which characters can move on to if it’s their turn to move. Furthermore, a location can have a list of characters that are currently on the location and a list of items that can be found and picked up in the location.

## World Class

The world class is a class representing a list of connected location objects, list of characters, and list of items. What the world class basically do is contains and manage all game objects. Basically what world object do is initialize all game objects for playing.

## Game Engine Class

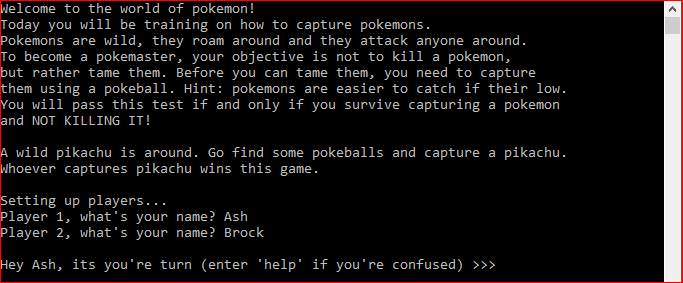
The game engine class is the main controller of the game. It controls the flow of the game such as telling which character has to move next by calling their appropriate operations and indicating whether it is time to end the game.

# Prototype Demonstration

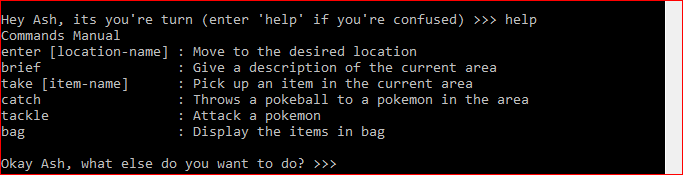
The output of this prototype is the same as the previous lab except that the code is now a full modularized prototype utilizing classes and functions.

The game works but incomplete, the only feature that works is where players can move around the world, the Pokémon randomly moves around the world, distribution of Pokeballs randomly around the world.

The game starts by introducing the rules and objectives of the game. The name of both players are prompted.



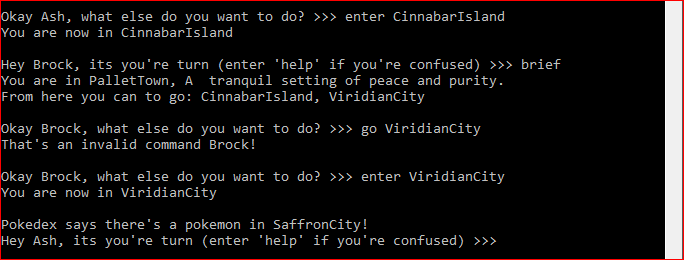
Entering the “help” command will display the commands available for both players to execute:



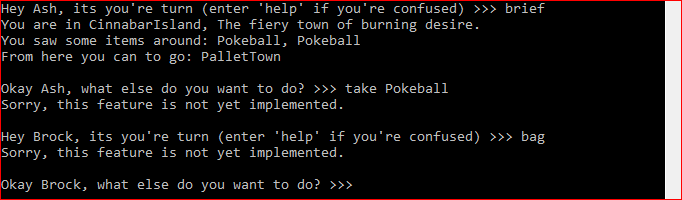
Entering the “brief” command basically displays the description of the current location of the player. It also displays the adjacent locations where the player can move next:



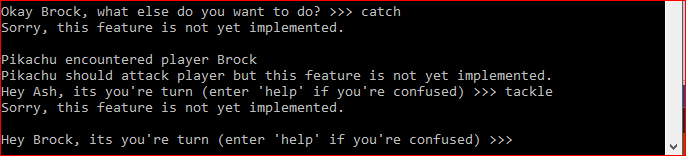
Entering a location requires the user to execute the “enter [location-name]” command where [location-name] is the adjacent location to the current location. The user can choose to enter any of the available location. After the player is done moving, then player 2 will have its turn. After all players made their turn, the Pokémon somewhere in the world will also randomly move and indicate a clue to both players where the Pokémon is currently at:



Somewhere along the game, players would be able to see Pokeballs in some places in which they can take using the “take [item-name]” command. However the command does not work yet on this prototype. The take command will supposedly pick up an item and put it into the player’s bag. The “bag” command will supposedly display the contents of the bag of a player:



Somewhere along the game, players would finally encounter the Pokémon. This is where the player is supposed to capture the Pokémon provided that they have Pokeballs in their bag. The “catch” command is supposed to do this but this feature is not yet implemented. Optionally, the player can also execute the “tackle” command to lower the health of the Pokémon. The “tackle” command does not also work on this prototype yet. Pokémon can fight back but the feature on this prototype does not work too:



That ends the prototype which demonstrates the game. So whoever player stays alive and is able to capture the Pokémon without killing it wins the game.

There are a lot of missing features in the game prototype but the source code has met all the concepts and topics in C++ that is required (Classes, Two data structures, and Inheritance/Polymorphism). The only missing parts of this project is to complete the game mechanics.