Conceptualización

-Consideraciones y Supuestos:

Nuestro “Bitmon Stadium” trata principalmente a los bitmons como cartas. Donde cada una de estas, tiene ciertas características propias que sólo son de ese Bitmon en específico, las cuáles sirven para determinar el daño de ataque en el momento de la batalla, la stamina que recupera, la vida , etc.

Un jugador puede tener muchas cartas, pero al momento del juego, sólo puede entrar 3 de ellas a la batalla. Es por esto que creamos como atributos en la clase JUGADOR dos listas de BITMON’S. Una que tiene cierta capacidad para guardar muchos Bitmons que tenga en su poder el jugador y otra lista, que es su equipo activo, algo así como su mazo de batalla para cuando le toque combatir.

También, en esta clase JUGADOR, hicimos un método que es “delete_bitmon(string): void” y un “add_button(Bitmon): void”, tomando como supuesto que los Bitmons ya existen y que podrían ser capturados , como también eliminados para darle espacio a otros.

Nuestra segunda clase importante en nuestro juego es BITMON. Bitmon cómo decíamos más arriba, son estas especies de cartas. Al tener cada una características distintas y particulares entre objetos de estas clases, el método “atacar()” será el más largo y importante , ya que este ocupa casi todos los atributos de BITMON para hacer los cálculos y calcular la vida que quita.

También BITMON está asociado a una clase PODER. PODER tiene los atributos distintivos de cada ataque, estos eran necesarios ya que según el enunciado general, habían algunos poderes que podrían causar eventos secundarios y que tenían que tener su habilidad especial en alguna parte. Y si no tiene, esta de igual forma puede ser de tipo “NULL” y no devolver nada.

Luego viene nuestra clase central, LUCHA. En LUCHA, nosotros asumimos que al principio de la partida, el jugador podrá elegir al primer Bitmon que lanzará a la batalla desde su “equipo_activo”. Para esto tuvimos que poner “eleguir_bitact()” donde mostrará a los Bitmons del equipo, y uno podrá elegir al que mejor estime conveniente como primero en su lista de Pelea. Como podemos elegir al bitmon activo, también necesitamos un método que nos retorne el Bitmon activo, esto lo supusimos ya que al ir el sistema de TURNOS entre jugadores en la interfaz de consola, sería funcional.

También asumimos que los Bitmons deben quedar cómo entraron a la batalla, es por esto que hicimos un método “reset_stats()”, que deja a los bitmons como nuevos una vez acabada la batalla.

Cómo último supuesto, sabemos que algunos métodos de nuestro modelo no serán 100% funcionales al momento de pasarlo a interfaz gráfica, pero al menos nos servirán para esta primera parte.

-Listado de Clases y sus funciones: ~JUGADOR:

La principal función del jugador es almacenar los Bitmons y administrarlos para la batalla, de hecho, dos de sus atributos son listas de Bitmons. Sus métodos van enfocados a este punto, ya que contiene dos métodos que agregan y borran Bitmons como también uno para elegir el equipo que sale a la batalla al momento de luchar. Como distintivo le pusimos un atributo “nombre”, que sólo sirve para ser más agradable la interfaz al momento de jugarlo.

~BITMON:

Esta clase es la más usada dentro de nuestro modelo, es por eso que tuvimos que tener cuidado con la privacidad de sus datos. También la principal funcionalidad de sus atributos es saber el estado actual del Bitmon en el campo de batalla. Sus métodos de igual forma van muy enfocado a eso y las acciones que toman estos bitmons en batalla.

~LUCHA:

LUCHA es la batalla misma, acá ocurre todo y ocupa todas las clases de nuestro modelo. Su funcionalidad es proporcionar los datos de la batalla y los Bitmons activos, para ver a quién le llega el ataque y quien lo hace,entre otras funciones.También tiene los métodos para determinar al jugador ganador.

~PODER:

Su funcionalidad va ligada directamente con la clase BITMON. BITMON almacena una lista de estos poderes donde cada uno es distinto al otro, tiene daño, costo y habilidad especial distinta. Es por esto que esta clase no tiene métodos, sólo es ocupada para almacenar los atributos de los poderes que ocupará el bitmon al atacar a su oponente.

~MODELO:

La clase MODELO se utiliza principalmente para facilitar la eleccion de los Bitmons disponible para el usurario, su funcionalidad es mostrar el nombre y de qué tipo de Bitmon es (es decir si es un Bitmon tipo agua o eléctrico, entre otros más disponibles) para que el usuario pueda escoger conociendo estos aspectos de los Bitmons.