Trabajo Práctico Evaluativo

Gimnasio Pokémon

Integrantes: Di Sarro Joaquina, Frangolini Luciano

25-6-2020

Contenido

Primera Parte

Introducción

Se desarrolló un simulador de un Gimnasio Pokémon donde se enfrentan 16 entrenadores con sus respectivos pokemones en un sistema de eliminación. Luego de 4 rondas y un combate final se define el ganador de la competencia.

Cada entrenador tiene a disposición un mazo de cartas, de las cuales sólo podrá hacer uso de cierta cantidad, una colección de pokemones, un nombre y un nivel de clasificación.

El torneo establece un sistema de eliminación sencilla donde al enfrentarse dos entrenadores, aquel que pierde recibirá el castigo de ser eliminado mientras que el ganador pasará a la siguiente ronda. Durante la batalla cada uno de los entrenadores podrá elegir uno de sus pokemones y hacer uso de una de sus cartas; al finalizar el enfrentamiento ganará aquel pokémon que se encuentre en mejor estado, aumentando así su nivel de experiencia como también la del entrenador. En caso de perder su experiencia aumenta en menor proporción pero el entrenador es descalificado del torneo.

Funcionalidades

- **Torneo**: Es el encargado de manipular las distintas rondas y de almacenar los resultados de cada una de las batallas. Existe una única instancia del mismo.
- Reporte: Se encarga de almacenar el detalle de cada una de las batallas donde se enfrentan dos entrenadores con sus respectivos pokemones, y el resultado que se genera.
- **Ronda**: Es el encargado de enfrentar a todos los entrenadores de a pares, obteniendo los ganadores y eliminando a los perdedores del torneo.
- **Batalla**: Se encarga de enfrentar a dos entrenadores permitiéndoles a cada uno seleccionar un pokémon y seleccionar una carta en caso de ser posible, para que luego los pokemones se enfrenten. Además tiene la responsabilidad de actualizar el estado de los entrenadores y sus pokemones.
- **Entrenador**: Cuenta con un repertorio de pokemones y un mazo de cartas de las cuales sólo puede utilizar cierta cantidad. Posee un nombre, una clasificación y créditos.
- **Pokémon**: Cuenta con un nombre, escudo, vitalidad, fuerza, experiencia y una clasificación. Tienen la cualidad de atacar a otro pokémon realizando una secuencia de acciones las cuales son golpe inicial, recarga y golpe final; como también así de ser atacados y hechizados. Éstas cualidades dependerán del elemento al que pertenezcan

(hielo, fuego, hielo recargado, agua, tierra y eléctrico), definiendo así diferentes comportamientos. Existen dos tipos de pokemones posibles, comunes y legendarios, siendo estos últimos únicos dentro de todo el torneo.

- **Carta:** Son objetos utilizables por los entrenadores para hechizar de manera negativa a los pokemones adversarios. Existen 3 tipos, niebla, tormenta y viento, las cuales impactarán en los pokemones dependiendo de su elemento.

UML

Para acceder al UML haga click derecho sobre el hyperlink y seleccione la opción abrir hipervínculo.

Ingresar al UML

Características

Pokémones

Tipo	Golpe Inicial	Recarga	Golpe Final	Daño
Fuego	Se inflige al adversario un daño igual a su fuerza de ataque, luego su fuerza de ataque se reduce a la mitad (cansancio).	Incrementa un 10% la fuerza y la vitalidad.	Provoca al adversario un daño igual a su fuerza más un 20% y luego la fuerza se agota por completo (queda en cero).	El escudo y la vitalidad absorben la mitad del daño cada uno (decrementandose).
Agua			Provoca al adversario un daño igual a su fuerza y luego su fuerza se reduce a la mitad.	El escudo absorbe todo el daño, solo cuando este se agota comienza a decrementarse la vitalidad.
Hielo		Si el pokemon posee la capacidad de “gran recarga”, su fuerza tomara el valor de 400 y no recargara su vitalidad. Si no posee dicha capacidad, tendrá el comportamiento habitual.	Provoca al adversario un daño igual a su fuerza menos un 10% pero conserva la misma fuerza.	El escudo absorbe el 75% del daño y la vitalidad el otro 25%.
Tierra		Recarga toda su fuerza.	Provoca al adversario un daño igual a su fuerza más un 40% pero conserva la misma fuerza.	El escudo absorbe todo el daño, solo cuando este se agota comienza a decrementarse la vitalidad.
Eléctrico		Incrementa un 50% su vitalidad y un 10% su fuerza.	Provoca al adversario un daño normal según su fuerza actual. Decrementa su fuerza en un 10%.	

Cartas

Tipo	Tormenta	Niebla	Viento
Fuego	Disminuye su vitalidad un 40%.	Disminuye sus estadísticas un 10%.	Su ataque se reduce un 10% de su vida.
Agua	Disminuye su escudo un 20%.	Disminuye sus estadísticas un 7%.	Su ataque se reduce un 50% de su escudo.
Hielo	Disminuye su vitalidad y su escudo un 15%.	Sus estadísticas se reducen un 30% de su ataque.	Su ataque se reduce un 10% de su vida y un 10% de su escudo.
Tierra	Se le resta la cantidad de escudo que posea a su vitalidad.	Sus estadísticas se reducen un 20% de su vida.	Su ataque se reduce un 20%.
Eléctrico	Disminuye su vitalidad un 10%.	Disminuye sus estadísticas un 15%.	Su ataque se reduce un 20% de su escudo.

Desarrollo

Para el desarrollo comenzamos evaluando las distintas funcionalidades que debería de tener cada sección y realizando un análisis y diseño de las mismas. Se definieron las 3 secciones principales, siendo éstas el torneo junto a sus enfrentamientos, los entrenadores y los pokemones.

Con respecto al torneo, se estableció que utilizaríamos un sistema de eliminación simple para el enfrentamiento de los entrenadores, generando así diferentes rondas y batallas. Definimos que el torneo debería implementar el patrón Singleton para así tener una única instancia en todo el programa, tener una lista de entrenadores y generar las distintas rondas. Con respecto a las rondas, les dimos la responsabilidad de enfrentar a los entrenadores, eliminando aquellos que perdían las batallas. Para obtener este resultado, se enfrentan dos entrenadores en una batalla, donde cada uno elije un pokémon y en caso de querer, y tener disponible, una carta para hechizar al contrincante. El enfrentamiento consta de tres iteraciones donde los pokemones se atacan, sin embargo, éstas pueden llegar a ser menos si uno de los pokemones muere durante el combate. En caso de que ningún pokémon muera, se define el ganador mediante un cálculo basado en sus estados actuales.
La elección tanto del pokémon como de la carta se implementó de manera al azar para así solucionar el problema de no tener una interfaz gráfica y hacerlo de manera más dinámica.

Respecto a los entrenadores, establecimos que debían de tener un nombre, un mazo de cartas, una cantidad de cartas disponibles y una lista de pokemones. Su responsabilidad es la de elegir un pokémon al batallar y eliminar un pokémon de su lista cuando éste muere.

Luego comenzamos el análisis de los pokemones y decidimos hacer dos tipos, los comunes los cuales se pueden clonar y los legendarios quienes deben ser únicos durante todo el torneo. Cada pokémon contaría con un nombre, vitalidad, escudo y fuerza; sería capaz de recibir ataques y de realizarlos, como también así poder comparar su estado con el de su contrincante. Luego de analizarlo decidimos utilizar el patrón Decorator para decorar al pokémon dependiendo de su elemento, el cual podría ser hielo, hielo recargado, fuego, agua, eléctrico o tierra. Dependiendo de su elemento su comportamiento y su estado inicial lo distinguirían del resto.
Al enfrentarse dos pokemones uno de éstos realizaría una secuencia de ataques mientras que el otro sufriría cierta cantidad de daño. La secuencia de ataques consta de un golpe inicial, una recarga y un golpe final, dependiendo éstos tres del elemento al que pertenezca el pokémon, como también así dependerá el impacto del ataque sobre el pokémon contrincante.

Para crear los diferentes pokemones coincidimos en que la mejor opción sería implementar el patrón Factory dado que éste nos permite una mejor legibilidad y claridad a la hora de instanciar los diferentes objetos. Decidimos desarrollarlo primero estableciendo el tipo del pokémon (común o legendario) y luego decorándolo con su elemento.

Luego de desarrollar las funcionalidades mencionadas anteriormente procedimos a modelar las cartas, implementando una interface que permitiese a los pokemones ser hechizados. Existen tres tipos de cartas, viento, niebla y tormenta, que influirán de distinta forma dependiendo del elemento del pokémon afectado. Estas cartas implementarán una interface en común para realizar sus hechizos.

Como se indicaba que tanto los pokemones como los entrenadores debían ser clasificables, se implementó una interface IClasificable la cual permitiría a ambas clases implementar su propio sistema de calificación. Cada pokémon ahora tendría su clasificación actual la cual aumentaría dependiendo de su experiencia ganada en batallas, por consiguiente también le agregamos como atributo la experiencia. Con respecto a los entrenadores su clasificación dependería de la sumatoria de clasificaciones de su colección de pokemones.
Al finalizar una batalla, ambos pokemones incrementan su experiencia, en el caso de los ganadores en mayor proporción, la cual llegando a cierta cantidad aumentaría sus clasificaciones.

Para mejorar el sistema de premio/castigo, agregamos un atributo a los entrenadores el cual hace referencia a un monto de créditos, el cual aumenta a medida que el entrenador

avanza de ronda. Como mencionamos anteriormente el castigo sería la eliminación del entrenador del torneo.

Para poder llevar un registro de los enfrentamientos y sus resultados, decidimos crear una clase Reporte la cual se encargaría de guardar la información sobre una batalla. Ésta clase tendría como atributos a los dos entrenadores, a sus respectivos pokemones y el resultado. Como consecuencia el torneo tendría una lista de reportes con la información de todas las batallas.

Al desarrollar la implementación del uso de una carta por parte del entrenador, debíamos tener en cuenta el caso de que éste no tuviese más cartas disponibles. Este caso lanzaría una excepción de tipo SinCartasDisponiblesException encargado de informar ésta situación y no permitirle al participante utilizar la carta.

Al momento de implementar la clonabilidad en los pokemones comunes y legendarios decidimos asignarle ésta responsabilidad a la clase abstracta pokémon. Como consecuencia nos dimos cuenta que para que esto sea posible, los decoradores también debían de ser clonables implementando en consecuencia el método clone.

En un principio nos encontramos con el problema que al clonar un pokémon, sus atributos se encontraban sin inicializar, por lo que nos dimos cuenta que estábamos aplicando mal el patrón decorator. Le estábamos asignando los valores al atributo de tipo Pokémon y no a los heredados, por consiguiente, pudimos darnos cuenta del error y solucionarlo.

Como los entrenadores también deben ser clonables, notamos que para clonar sus atributos no primitivos debíamos implementar la clonabilidad en ellos. Los pokemones ya eran clonables pero las cartas no, por consiguiente, debimos de volverlas clonables. Esto último se aplicó creando una clase abstracta Carta que implementa la clonabilidad y la interface ICarta antes mencionada, y de la cual ahora se extienden los tres tipos de cartas.

Una vez solucionado el tema de las batallas, de la clonabilidad y del decorator, nos encontramos con que algunos métodos implementados en los pokemones común y legendario, quedarían vacíos, los cuales hacen referencia a aquellos métodos correspondientes a las cartas y a las batallas. Sin embargo notamos que esto no impacta en el correcto funcionamiento del programa, ya que estos métodos nunca se ejecutarían porque siempre se utilizan los sobrescritos que se encuentran en cada elemento del decorator.

Notas

Para poder trabajar en equipo y de manera simultánea decidimos implementar el uso de GIT utilizando las diferentes herramientas que este nos proporciona. Esto nos permitió administrar mejor el tiempo y coordinar la distribución de tareas.

Se implementó el uso de Maven para eliminar la dependencia del entorno de programación con el que se esté trabajando.

Segunda Parte

Introducción

Al simulador del Gimnasio Pokémon desarrollado en la primera parte se le agregan nuevas funcionalidades y se agregó un nuevo Main para probarlas.

Ahora se cuenta con una interfaz visual con la cual el usuario puede interactuar a la hora de ingresar los participantes del torneo. Además de una ventana en la cual se detallan los eventos que ocurren durante el torneo.

Se añadió un sistema de persistencia el cual se manifiesta cada vez que una ronda o etapa del torneo toma lugar.

El sistema de batallas se vio modificado y ahora cada ronda del torneo cuenta con tres arenas en las cuales desarrollarse. Cada arena cuenta con la capacidad de cambiar su estado, entre los cuales se encuentra el estado de enfrentamiento. De esta forma puede ocurrir que hasta seis participantes se encuentren batallando concurrentemente en una ronda, dos en cada arena.

Funcionalidades

- **Vista:** La parte visual se divide en cuatro ventanas principales, cada una con una funcionalidad en concreto.

  -La primera de ellas permite visualizar en una lista qué participantes se encuentran registrados en el torneo y en otra lista, al seleccionar un entrenador, se despliegan sus pokemones. Además, permite utilizar ciertos botones para agregar más entrenadores y pokemones, así como clonarlos. Cuando se posean suficientes participantes para que inicie el torneo, también habilita un botón para comenzarlo.

  -La segunda permite ingresar los datos de un entrenador para luego agregarlo a la lista de participantes. En esta ventana también se pueden agregar pokemones para hacer más cómodo el proceso.

  -Luego se encuentra la ventana en la cual ingresamos los datos de un Pokémon para añadirlo a la colección de un entrenador.

  -Finalmente una vez iniciado el torneo, se muestra una ventana con los detalles de todo el desarrollo del torneo hasta su finalización.

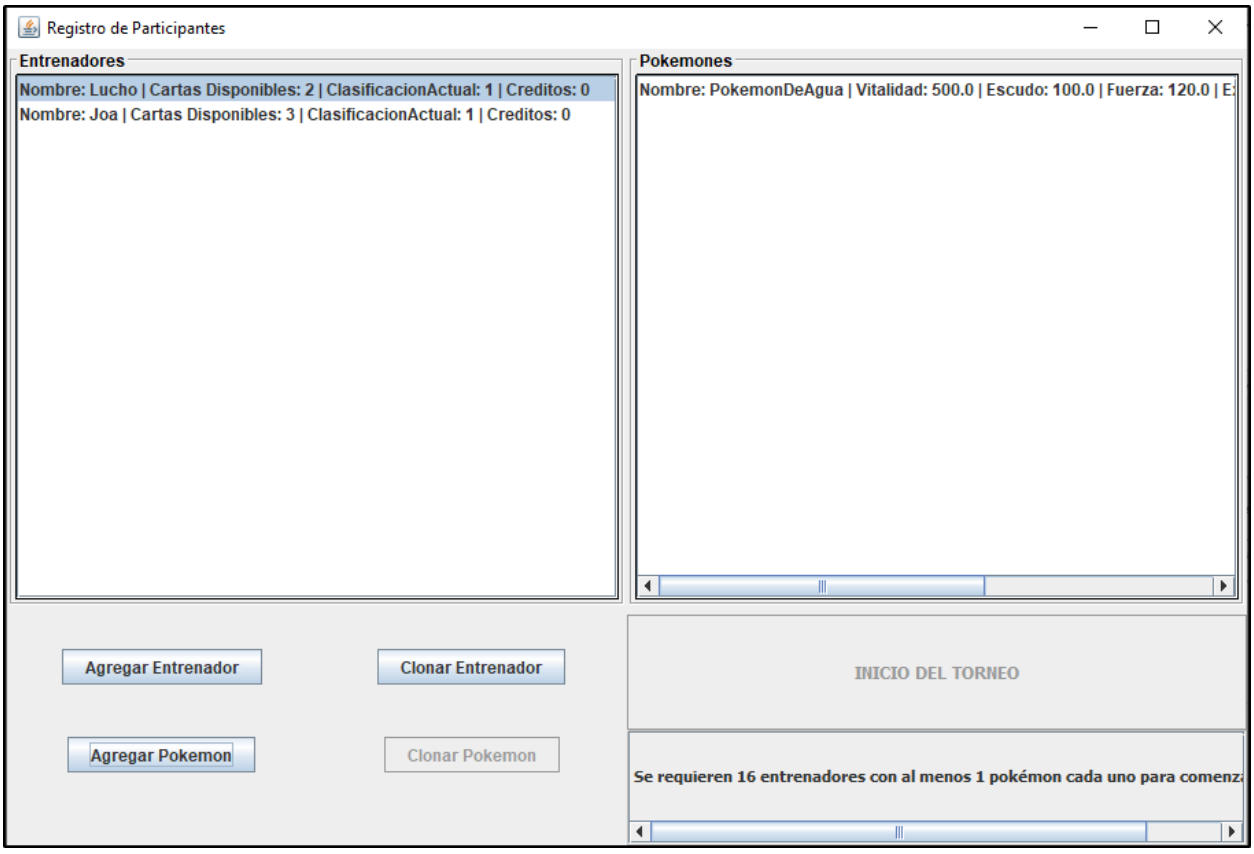
UML

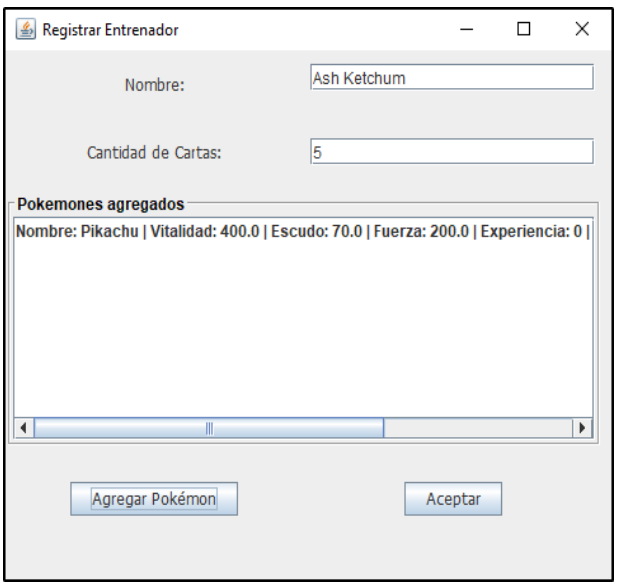
Para acceder al nuevo UML haga click derecho sobre el hyperlink y seleccione la opción abrir hipervínculo.

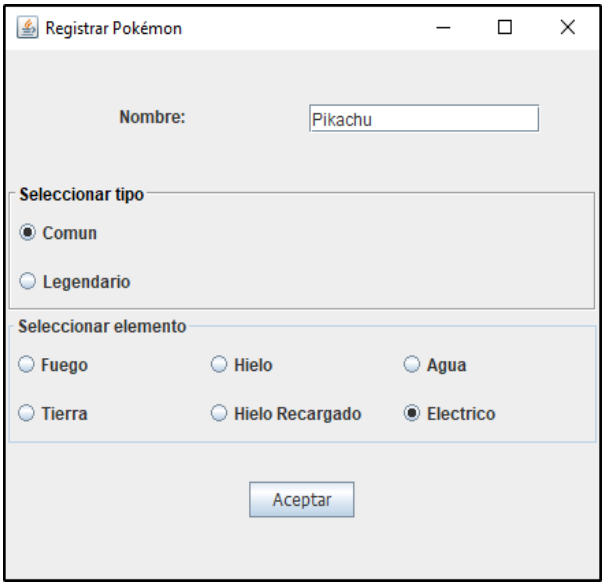
UML

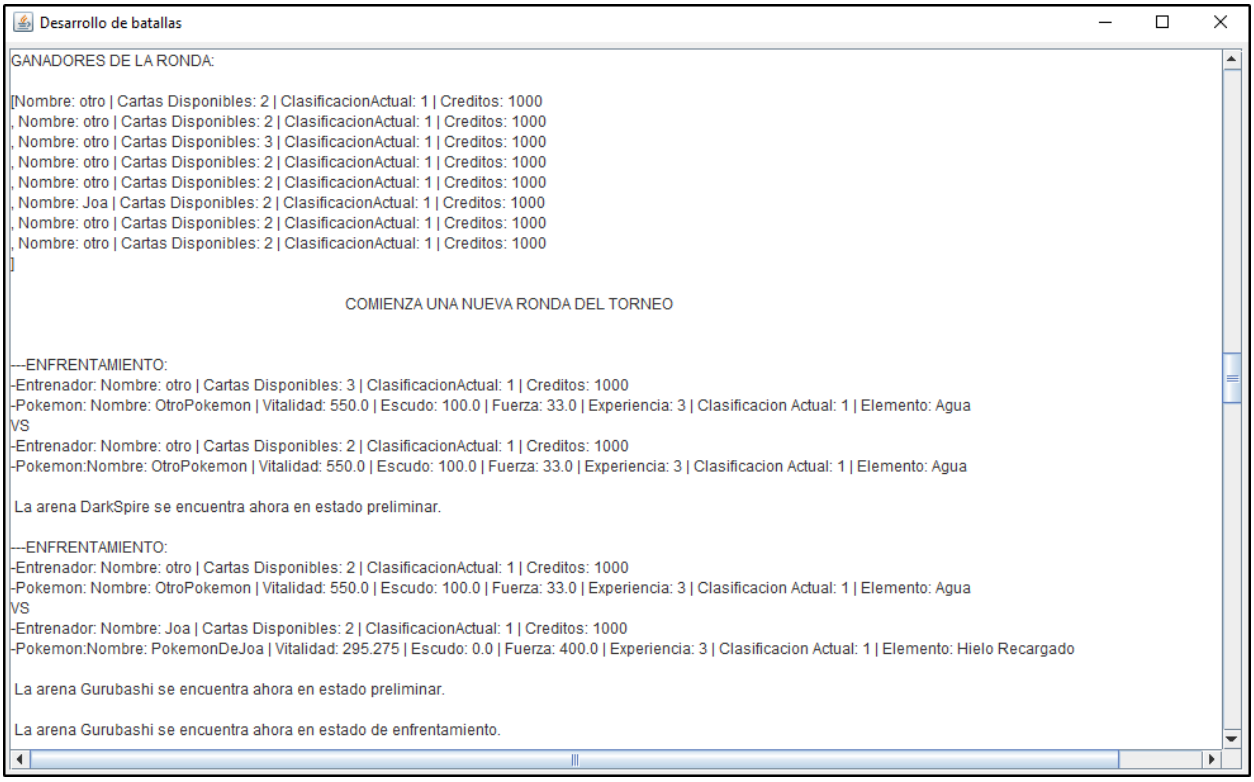
Características

Vistas




Registrar Entrenador
Nombre:
Ash Ketchum
Cantidad de Cartas:
5
Pokemones agregados
Nombre: Pikachu | Vitalidad: 400.0 | Escudo: 70.0 | Fuerza: 200.0 | Experiencia: 0 |
Agregar Pokémon
Aceptar


Registrar Pokémon
Nombre:
Pikachu
Seleccionar tipo
Comun
Legendario
Seleccionar elemento
Fuego
Hielo
Agua
Tierra
Hielo Recargado
Electrico
Aceptar


Desarrollo de batallas
GANADORES DE LA RONDA:
[Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: otro | Cartas Disponibles: 3 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: Joa | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
, Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
]
COMIENZA UNA NUEVA RONDA DEL TORNEO
---ENFRENTAMIENTO:
-Entrenador: Nombre: otro | Cartas Disponibles: 3 | ClasificacionActual: 1 | Creditos: 1000
-Pokemon: Nombre: OtroPokemon | Vitalidad: 550.0 | Escudo: 100.0 | Fuerza: 33.0 | Experiencia: 3 | Clasificacion Actual: 1 | Elemento: Agua
VS
-Entrenador: Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
-Pokemon:Nombre: OtroPokemon | Vitalidad: 550.0 | Escudo: 100.0 | Fuerza: 33.0 | Experiencia: 3 | Clasificacion Actual: 1 | Elemento: Agua
La arena DarkSpire se encuentra ahora en estado preliminar.
---ENFRENTAMIENTO:
-Entrenador: Nombre: otro | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
-Pokemon: Nombre: OtroPokemon | Vitalidad: 550.0 | Escudo: 100.0 | Fuerza: 33.0 | Experiencia: 3 | Clasificacion Actual: 1 | Elemento: Agua
VS
-Entrenador: Nombre: Joa | Cartas Disponibles: 2 | ClasificacionActual: 1 | Creditos: 1000
-Pokemon:Nombre: PokemonDeJoa | Vitalidad: 295.275 | Escudo: 0.0 | Fuerza: 400.0 | Experiencia: 3 | Clasificacion Actual: 1 | Elemento: Hielo Recargado
La arena Gurubashi se encuentra ahora en estado preliminar.
La arena Gurubashi se encuentra ahora en estado de enfrentamiento.

Arena

Estados por los que pasa una arena			
Preliminar	Enfrentamiento	Definición	Limpieza
La arena se encarga de recibir a los participantes junto a sus pokemones para luego hacerlos batallar.	Los participantes que se encuentran en la arena hacen batallar a sus pokemones.	Se decide el resultado del combate ocurrido y su ganador.	La arena se vacía para luego poder recibir nuevos participantes.

Desarrollo

Al desarrollar las nuevas funcionalidades del trabajo, se dividieron dos áreas sobre las cuales trabajar por separado. Siendo una la parte visual y otra la parte interna del modelo.

Implementación visual (Patrón MVC)

La parte visual y el patrón MVC se realizaron en una branch o rama aparte del desarrollo del modelo y sus nuevas funcionalidades. Hacer esto conllevó tener algunos conflictos al querer mergearlo o juntarlo, pero no interrumpía ni afectaba el desarrollo de cualquier otra función necesaria para el funcionamiento interno del modelo.

Lo primero que se hizo fue pensar y diseñar la parte visual exclusivamente. Su primer diseño tenía muchas ventanas, botones y funcionalidades, las cuales se fueron reduciendo a medida que se evaluaba la complicación de controlar cada una de las acciones que el usuario podría realizar o de la complejidad de distribuir componentes visuales en una sola ventana.

Luego de un debate grupal, se llegó a un esquema con el cual quedamos conformes y se realizó el desarrollo visual.

Aunque éste no fue el definitivo, se asemeja mucho al resultado final. Hubo cambios en algunos detalles a medida que se avanzaba en el proyecto, pero en general, ya estaban listas.

Luego se comenzó a desarrollar los controladores, los cuales en un principio recibían todo tipo de parámetros de las ventanas. Era un gran problema, ya que se implementaron en ellos los Listeners de las vistas correspondientes a cada uno, como por ejemplo el MouseListener y el KeyListener, siendo necesarias muchas funciones para obtener las distintas componentes de las ventanas.

Este error fue resuelto luego de una consulta y las ventanas implementaron, cada una,

sus Listeners correspondientes, salvo el ActionListener, el cual si es responsabilidad del controlador.

Lo siguiente fue desarrollar en las vistas los métodos que los controladores deberían utilizar para cambiar la información que se visualiza en ellas. Esto último llevó a buscar un medio con el cual comunicar a los controladores el momento en el cual actualizar, por ejemplo, una lista visual.

La solución a este problema fue el uso del patrón observer, el cual notifica cada cambio en la información de la ventana a sus respectivos observadores, es decir, controladores.

Por ejemplo, un paso a paso de lo que finalmente sucede al presionar un botón para agregar un elemento a una lista es:

-Se presiona el botón con el cual agregar un elemento.

-El controlador correspondiente recibe la señal y utiliza alguna funcionalidad del modelo para agregar un elemento a la lista.

-El modelo notifica al controlador de su cambio a través del patrón observer.

-El controlador recibe la señal y le informa a la ventana que hubo un cambio, y que debe actualizarse

-Finalmente la ventana se actualiza y muestra su nuevo contenido.

Las partes más difíciles de implementar involucran el lograr mostrar en una ventana el desarrollo o historial de los combates en el torneo, y relacionar algunas ventanas entre sí, ya que la información que una recibía podía ser necesaria para otra.

Implementación del Sistema:

Se implementaron nuevas funcionalidades dentro del sistema como las Arenas, y se realizó un cambio en la implementación y lógica de cómo sucedían las batallas, dando como resultado un código más limpio, legible, modularizado y eficiente. Cada batalla de ahora en más está asociada a una Arena en la cual se llevará la misma a cabo. Al existir únicamente tres Arenas, existe la posibilidad de que otra batalla este ocupándola, y por consiguiente esta deba de esperar hasta que la Arena finalmente se libere para poder ingresar. Esto se implementó mediante Hilos (threads), tomando como hilo la batalla, y como monitor la arena.

Una Arena pasa por distintos estados, Preeliminar, Enfrentamiento, Definición y Limpieza. Para esto, se implementó el patrón State logrando así que por cada estado que la arena se encuentre, su funcionamiento sería distinto. A su vez, se implementó el patrón Observer para poder informarte al controlador los distintos cambios y mensajes que toman lugar durante la ejecución de una Batalla en una Arena, dando así como resultado, que luego de

oprimir el botón “Comenzar Torneo” en la vista principal, se pueda observar “en vivo” en la ventana de Arenas, que va sucediendo en cada una de estas, por ejemplo, quiénes la ocupan, con qué pokemones, un detalle de su enfrentamiento y finalmente quién de ambos gana.

Además, se implementó la serialización de datos permitiendo así guardar la información del resultado de cada Ronda. Esta información se utiliza luego para comenzar la siguiente Ronda.