Universidad de los Andes > INGENIERÍA

Proyecto: Hearthstone
Programación orientada a objetos

Profesor: Ignacio Parada
Ayudante: Diego Beckdorf
Francisco Moreira
Gianluca Fenzo
Gianluca Troncossi

Entrega 2:
Modelo relacional - Hearthstone
CLASES:

Tablero:
El tablero es donde se crea el juego y sobre el cual los jugadores llevan la partida. Tiene una relación de asociación con la clase Jugador.
Atributos:

- Tiempo: El cual lleva el tiempo total que se juega en la partida. (Este atributo y su método se dejaron expresados ya que no se pide en la entrega.)
- Reloj: Reloj que lleva los segundos, minutos y horas que ha durado una partida. (Al igual que tiempo se dejó expresado, ya que no se pide para esta entrega).
- Turno: Un contador que lleva el mando de quien es el turno actual.
- J1: Objeto Jugador.
- J2: Objeto Jugador. (obs: El tablero recibe dos jugadores, para así ejecutar todo el juego, cada jugador se diferencia mediante un bool y tiene su zona asignada).
- Manager: Objeto manager, el cual lleva todos los avisos y interacciones con el jugador.
- Random Rdm: Generador de números aleatorios para decir quien parte.
- Historial: Lista la cual lleva los esbirros jugados en la partida (A diferencia del juego original este historial no explicitan los ataques y a quien están dirigidos, es en el fondo, una versión más simple del historial.)

Métodos:

- Partir(): Método con el que se inicia el juego, determina al jugador que parte, entrega las respectivas cartas de la mano a cada jugador y les ofrece cambiar cartas utilizando el método CambiarCarta de la clase Jugador y luego llama a InicioTurno.
- InicioTurno(): Recibe como parámetro al jugador que tiene que jugar, le muestra su mano y le pregunta qué desea hacer, con opciones que llaman a distintos métodos tales como: BajarCarta(), UsarCarta(), Conceder(), FinTurno() y UsarHabilidad().
- TiempoTurno():
- BajarCarta(): Método que muestra la mano del jugador actual y le permite elegir una para bajarla al tablero..
- UsarCarta(): Método que permite que el jugador que está jugando, ataque al enemigo utilizando los minions que se encuentran en juego.
- UsarHabilidad(): Método que permite que el jugador utilice la habilidad especial de su héroe, llama al método de mismo nombre que se encuentra en Jugador.
- FinTurno(): Método que finaliza el turno de un jugador y inicia el turno de el otro mediante el método InicioTurno();
- Conceder(): Método que modifica la vida del jugador que lo utiliza por 0 y luego finaliza el turno, para que el jugador pierda.
- Menu(): Despliega en consola el menú de opciones, para que el jugador determine que hace en su turno.
- Ataque(): Ejecuta el ataque en el juego, ya sea entre minion/minion, minion/heroe, heroe/minion y heroe/heroe.
- InfoCartas(): Entrega información con respecto a las cartas que posee el jugador.

Cartas:
Es una clase abstracta con tres clases hijas: Armas, Hechizos y Minions. Compone a la clase Mazo.
Atributos:

- Nombre: contiene un String con el nombre de la carta.
- Costo: Tiene un número entero con el costo de maná de la carta.

OBSERVACIÓN: Para esta entrega, encontramos un problema al querer acceder a los atributos de la clase minions cuando estos se encontraban dentro de la lista tablero, y la consola arrojaba un error, donde informaba que se estaba asumiendo que el elemento al que queríamos acceder era un minion, cuando la lista es de cartas, por lo que para poder arreglar el problema se pasaron los atributos, de la clase minion a cartas, en un futuro se pretende arreglar la clase minions, (cuando se pidan además cartas y armas)

- Ataque: Contiene un número entero el cual representa el ataque del minion.
- Vida: Contiene un número entero el cual representa la vida del minion.
- Memoria: Contiene un número entero el cual representa el número de turnos que un esbirro permanece vivo en el tablero.
- Estado: Booleano que da el estado de activo o inactivo a una carta que esté sobre el tablero (leer método a continuación para más detalle).

Métodos:

- AplicarEfecto(): Aplica la habilidad especial que tiene la carta. (Para la entrega 2 no existe ningún minion u otra carta con un efecto, por lo que solo se deja expresado, para en una futura entrega evaluar si se agrega el metodo/s.)

Armas:
Clase para las cartas de tipo “Arma”, es una clase hija de la clase “Cartas”.
Atributos:

- Ataque: Contiene un número entero el cual representa el ataque del arma
- Durabilidad: Contiene un número entero el cual representa la durabilidad del arma

Esta clase no contiene métodos.
Hechizos:
Clase para las cartas de tipo “Hechizo”, es una clase hija de la clase “Cartas”.
Observación: Las cartas de hechizo solo poseen un valor de maná (guardado en la clase abstracta padre) y su efecto.
Esta clase no contiene métodos.
Minions:
Clase para las cartas de tipo “Minion”, es una clase hija de la clase “Cartas”.
Atributos:

- Ataque: Contiene un número entero el cual representa el ataque del minion.
- Vida: Contiene un número entero el cual representa la vida del minion.
- Memoria: Contiene un número entero el cual representa el número de turnos que un esbirro permanece vivo en el tablero.
- Estado: Booleano que da el estado de activo o inactivo a una carta que esté sobre el tablero (leer método a continuación para más detalle).

Esta clase no contiene métodos.

Mazo:

Clase la cual guarda las cartas del “mazo” de cada jugador. Tiene una relación de asociación con la clase Jugador y utiliza la interfaz IJugar, además tiene una relación de composición con Cartas.

Atributos:

- miMazo[]: Lista miMazo[] la cual guarda las cartas del mazo.
- NombreMazo: String el cual contiene el nombre del mazo.
- FatigaCount: int el cual lleva el valor en que se disminuye la vida del jugador cuando se activa fatiga, va aumentando de acuerdo al número de veces que se activa el método fatiga.

Métodos:

- Fatiga(): Método el cual, cuando no quedan cartas en el mazo, reduce la vida del jugador cuando este roba una carta o bien mediante algún efecto de otra carta se produce que el jugador robe una carta.

Jugador:

Clase que contiene la información del jugador, tiene una relación de asociación con las clases Mano, Mazo y Tablero. Además utiliza la interfaz Jugar la cual se relaciona con tablero y mano, además tiene una clase hija Héroe.

Atributos:

- Nombre: Contiene un String con el nombre del jugador.
- ID: Boolean con un id para cada jugador.
- Héroe: Cada jugador contiene uno de la clase héroe.
- Mano: Lista con las cartas en la mano.
- Maná: Contiene un número entero que representa el maná que tiene el jugador.
- Tablero[]: Zona del tablero (una lista) donde los jugadores van jugando sus cartas, notar que se asignan dos listas una por cada jugador.
- Armor: Método que contiene la cantidad de armadura que tiene el jugador.
- Vida: Contiene un número entero que representa la vida que tiene el jugador.
- Varauxdmg: Variable auxiliar que guarda un entero para poder aplicar efectos en heroe.
- Habmej: Habilidad Mejorada.
- UsoHab: bool para identificar que el heroe no pueda usar la habilidad del mismo más de una vez por turno
- Mazo: Lista que contiene las cartas de un jugador para una partida.
- rdm: Variable de tipo random para elegir una posición aleatoria en el mazo para devolver una carta.
- Manager: Objeto manager, el cual lleva todos los avisos y interacciones con el jugador.

Métodos.

- BajarCarta(): Método que remueve una carta de la Mano y la mueve a la lista Memoria (Jugador 1 o jugador 2) del tablero.
- Revmano(): Método el cual revisa que un jugador no pueda tener más de 10 cartas en su mano en el caso de robar con 10 cartas en mano la carta robada se “quema”, desaparece del mazo y la mano y pasa al historial.

- CambiarCarta(): Método que toma una carta de la Mano y la devuelve al Mazo, luego saca una carta de Mazo y la agrega a Mano, sólo funciona si el turno es 0.
- Conceder():
- RobarCarta(): Método que saca una carta de Mazo y la agrega a Mano.
- UsarCarta(): Método que utiliza el ataque de un minion o arma. Además resta el costo de la carta (en mana) a la cantidad de maná disponible del jugador.
- UsarHabilidad(): Método que aplica la habilidad especial del jugador(Héroe).
- Fatiga(): Método el cual baja la vida del jugador cuando este debe robar una carta y no posee más cartas en su mazo, el valor de la vida dañada va aumentando con el número de veces que se activa el método.

Héroe:
Clase hija de la clase Jugador, que define el héroe que se está utilizando.
Atributos:
- NombreHeroe: String el cual contiene el nombre del Héroe.
- Clase: Contiene un String que representa el tipo de héroe. (Importante al momento de ver la habilidad del Héroe, notar que no se puede usar el nombre del héroe para este propósito ya que existen héroes adicionales que presentan las mismas habilidades, ejemplo de esto puede ser Tyrande y Anduin, ambos curan en dos a unos de sus esbirros o a sí mismos, así también Morgl y Thrall generan totems aleatorios.)
- Manager: Recibe avisos y entrega las respuestas del usuario.

Métodos.
- Mensaje(): Muestra mensajes (pre-hechos) que un jugador puede mostrarle al adversario.

IJugar:
Interfaz que contiene todos los métodos necesarios para jugar.
Métodos:
- RobarCarta(): Método que saca una carta de Mazo y la agrega a Mano.
- BajarCarta(): Método que remueve una carta de la Mano y la mueve a la lista Memoria (Jugador 1 o jugador 2) del tablero.
- CambiarCarta(): Método que toma una carta de la Mano y la devuelve al Mazo, luego saca una carta de Mazo y la agrega a Mano, sólo funciona si el turno es 0.
- UsarCarta(): Método que utiliza el ataque de un minion o arma. Además resta el costo de la carta (en mana) a la cantidad de maná disponible del jugador.
- UsarHabilidad(): Método que aplica la habilidad especial del jugador(Héroe).
- Fatiga(): Método el cual baja la vida del jugador cuando este debe robar una carta y no posee más cartas en su mazo, el valor de la vida dañada va aumentando con el número de veces que se activa el método.
- RevMano(): Método el cual revisa que un jugador no pueda tener más de 10 cartas en su mano en el caso de robar con 10 cartas en mano la carta robada se “quema”, desaparece del mazo y la mano y pasa al historial.
- Conceder(): Método que cambia la vida del jugador a 0 y pierde el juego.

Manager:
Clase conectada con Jugador y Tablero la cual lleva todos los avisos y respuestas del juego. Creada para abstraer la implementación del modelo de la utilización de consola, esto para no tener problemas al momento de desarrollar la interfaz gráfica.
Métodos:
Aviso(): Método por el cual se da aviso de todos los métodos y avisos que se están ejecutando.
RecibirRespuesta(): Da las respuestas de los usuarios para rellenar los datos del juego.

Modelo Relacional:

Del modelo relacional de la entrega 1, no se realizaron grandes cambios, principalmente se trabajó en las clases tablero y jugador. Se decidió mover las zonas de tablero a la clase jugador ya que este baja las cartas al mismo, a ambas clases se les agregaron atributos y métodos para poder completar la entrega. El cambio con respecto a la clase cartas a la cual se le asignan los atributos de la clase minions, debe ser tomado como un cambio temporal, el cual se va a retomar y optimizar para la entrega 3, por ahora la clase minions no toma parte real de la funcionalidad del programa y queda propuesta (como un camino avanzado) para la entrega 3.

Supuestos:

A continuación describiremos los supuestos que hacemos para nuestro juego Hearthstone.

En esta entrega se debe entregar el código c# el cual contenga el juego (la base de este), dentro del código agregamos funcionalidades extras a las que se piden en el enunciado con el fin de tener un juego más completo, algunas funcionalidades se dejaron propuestas(a modo de tener trabajo hecho para futuras entregas, en el caso que se pidan ya se tendra parte del codigo listo.). Se actualizó el modelo relacional, el cual se anexará en un archivo visio, se recomienda ver el modelo relacional antes de evaluar el código.

1. Se supone que en consola se ingresarán solo opciones válidas (Por tiempo se irán agregando mensajes “Opcion invalida” en las siguientes entregas).
2. El programa actualmente está como público en todas sus clases, los archivos se empezaran a corregir y poner privados (los necesarios) después de recibir la corrección del programa base.
3. Suponemos que siempre juegan sólo dos jugadores.
4. Los jugadores utilizan las misma de cartas, sin embargo, cada uno posee su lista(mazo) con ellas ordenadas de manera aleatoria .
5. No existen clases de minions, se supondrá que no van a existir clases tipo mech, beasts, totem, dragon, murloc, pirate, demon, etc… (En el caso que se pida este tipo de clase solo basta agregarla como atributo a la clase minion.)
6. Se asume que por cada turno se le agrega una unidad de maná a cada jugador, hasta llegar al máximo de 10, habilidades especiales de cartas que otorguen cristales de mana o refresquen cristales de mana al jugador solo podrá hacerlo hasta

llegar a un máximo de 10 (el ya establecido), en el caso de algunas cartas de héroe druida que aumentan en un cristal de maná vacío, si se llega al límite de 10 cristales y se juega una carta de este estilo se sigue como lo hace el jugo y el efecto de esa carta cambia a robar una carta. La carta “Moneda”, se creará en el momento que se pida por enunciado, por el momento el segundo jugador no cuenta con esta medida, y por lo tanto parte con una desventaja.

7. Al ser dos jugadores con el mismo mazo, y al ser pedido por enunciado que solo deben jugar dos jugadores suponemos que no se ocupará la función de silenciar en juego de hearthstone (click derecho sobre el héroe oponente).
8. Los Hechizos del subtipo Secretos, se aplican como efectos pero en si son hechizos por lo que no ameritan una clase separada de ésta y no se animaran con un signo ?, sino que se supone que el oponente sabe que se jugó un secreto.
9. UsarCarta(), también contempla el uso de armas.
10. La clase héroe es hija de la clase jugador, esto quiere decir que jugador seria el usuario de battlenet(blizzard) osea la cuenta del jugador que se instancia mediante la elección de un héroe (Gul’dan, Thrall, Jaina etc…). La creacion de clase héroe no esta demas, ya que un jugador puede existir en el juego sin un héroe asignado, y al momento de entrar a una partida este debe elegir un héroe.
11. Fatiga() : La vida del jugador al robar la primera carta (con el mazo vacío) al jugador se le reducirá su vida en 1 al jugador y en 1 adicional cada vez que se active Fatiga(), es decir si se activa 3 veces la primera dañara en 1 al héroe luego en 2 y finalmente en 3.
12. Cada turno dura un máximo de 75 segundos. La cuerda que se activa mediante el método TiempoTurno(), el cual es un método propuesto y se activa cuando al jugador le quedan 20 segundos por jugar.
13. Los efectos individuales de cada carta se guardan y aplican con AplicarEfecto() en la clase Cartas. (Este método queda propuesto ya que para esta entrega no se pide.)