Gruppo 6
Assignment 2

10.04.2016

Indice

1 Componenti del gruppo

- Fabio Rosada 851772
- Daniele Crosariol 847056
- Federico Fontolan 854230
- Sebastiano Filippetto 854510
- Franco Scarpa 842833

2 Modifiche rilevanti al codice originale

2.1 CombatPhase

Questa classe non è stata modificata, ma bensì era lasciata a noi l'implementazione. Sono stati fatti dei metodi per gestire la dichiarazione delle creature attaccanti, la dichiarazione delle creature bloccanti e l'assegnazione danni. Per gestire i vari scontri tra le creature è stata fatta una classe interna con i campi dei giocatori, la creatura attaccante e le varie creature bloccanti, dentro questa classe c'è un metodo per settare la priorità degli attaccanti, e un metodo per eseguire lo scontro. Quindi per comodità se non ci sono creature nel campo del giocatore attaccante non si esegue nulla, altrimenti si eseguono i vari metodi: dichiarazione attaccanti, dichiarazione bloccanti, assegnazione danni. In dichiarazione attaccanti si utilizza un array dove si inserisce ogni creatura attaccante, in dichiarazione bloccanti si utilizza un array dove si inseriscono tutte le creature bloccanti, in assegnazione danni si crea un array dove si inseriscono i vari scontri che vengono fatti.

2.2 Targettable

Aggiunta l'interfaccia Targettable che estende l'interfaccia Effect, aggiunge solo il metodo boolean setTarget(), in questo modo, per esempio nella Deflection (3.9), possiamo distinguere le carte che hanno un target da quelle che non ce l'hanno, e successivamente accedere al metodo setTarget per cambiare il target.

2.3 Utils

Classe importata dal nostro codice dell'assignment 1. Contiene dei metodi che semplificano la lettura dei numeri e delle stringhe, ed è stato implementato il metodo statico creatureTargetChoice(ArrayList¡Creature¿ creature) che dato un array di creature, stampa il contenuto e dopo aver chiesto all'utente di scegliere, ritorna la creatura target.

2.4 Player

Qualche piccola modifica per consentire la prevenzione dei danni da parte di Darkness (3.7) e BevenolentAncestor (3.3).

2.5 CardStack

Aggiunto un metodo per vedere se una carta è presente o no.
In questo modo se vengono lanciate due Cancel con lo stesso target, non verrà stampato due volte il messaggio di eliminazione della carta. Il metodo non era strettamente necessario ma rende più gradevole il gioco.

3 Carte

3.1 Afflict

Afflict implementa l'interfaccia Targettable (che a sua volta estende Effect) e TriggerAction. Targettable ci fa implementare il metodo setTarget, mentre TriggerAction richiede il metodo execute(). Quando viene giocata la carta per prima cosa verrà chiamato il metodo play, dove verrà richiamato setTarget che mostrerà tutte le carte presenti in campo, una volta effettuata la scelta se il target è presente allora verrà richiamato super.play(), altrimenti se per esempio il campo è vuoto, ritornerà false. Nel momento in cui lo stack va a risolversi, se è presente un target, verranno modificati i valori di attacco e di difesa con -1/-1 e verrà settato il trigger. Il trigger non fa altro che aspettare che arrivi la EndPhase e a quel punto modifica l'attacco mettendo +1. DamageLeft non serve resettarlo in quanto si resetta in automatico alla fine del turno.

3.2 AggressiveUrge

AggressiveUrge ha esattamente lo stesso comportamento di Afflict (3.1), solo che su resolve imposta +1/+1 ad attacco e difesa, e sulla execute di nuovo resetta solo l'attacco passando il valore -1.

3.3 BenevolentAncestor

Semplice creatura con l'aggiunta dell'effetto che utilizza è un targettable (2.2) che estende effect, si utilizza questa interfaccia per gestire la scelta di un target, nel resolve si va ad aumentare il damage left di una creatura o si aumenta il numero di danni prevenibili di un giocatore.

3.4 BoilingEarth

Questa carta ha una struttura molto semplice: su resolve creo una nuova ArrayList (solo per semplicità di codice) di creature contenente le creature di entrambi in giocatori, e con un'unico ciclo for each faccio un danno a tutte le creature.

3.5 BronzeSable e Norwood Ranger

Norwood Ranger e Bronze Sable sono 2 semplici creature senza effetto con valori di attacco e difesa rispettivamente 1/2 e 2/1. Queste carte vengono messe nella lista contenente il board del gicoatore che le ha evocate nel momento in cui la pila viene risolta.

3.6 Cancel

Cancel implementa l'interfaccia Targettable (2.2) (che a sua volta estende Effect). Cancel ci fa implementare il metodo setTarget. Quando viene giocata la carta per prima cosa verrà chiamato il metodo play, dove verà richiamato setTarget che mostrerà tutte le magie della pila. Selezionata la magia bersaglio questa viene salvata in una variabile e si attende che la pila risolvi la magia. Nel momento in cui lo stack va a risolversi, deflection controllerà se la magia da neutralizzare è ancora presente nello stack: se è presente allora cancel la neutralizza e la rimuove dalla pila.

3.7 Darkness

Questa carta implementa l'interfaccia TriggerAction. Nel fare l'override del metodo 'resolve()', viene impostata a 'true' la proprietà 'combatPrevent' sia dell'owner che dell'avversario, facendo così in modo che essi siano tutelati da eventuali danni da combattimento durante questo turno. Successivamente viene impostato il trigger per richiamare l'execute nella EndPhase. Sulla 'execute()', tali proprietà vengono re-impostate a false, così che alla fine del turno i danni da combattimenti vengano inflitti normalmente ai giocatori.

3.8 DayOfJudgment

Day Of Judgment quando viene risolta accede al board di entrambi i giocatori invocando il metodo Clear, svuotando così la lista contenente le creature. In questo modo le creature di entrambi i giocatori vengono distrutte.

3.9 Deflection

Deflection implementa l'interfaccia Targettable (2.2) (che a sua volta estende Effect). Targettable ci fa implementare il metodo setTarget. Quando viene giocata la carta per prima cosa verrà chiamato il metodo play, dove verrà richiamato setTarget che mostrerà tutte le magie in pila che implementano Targettable e quindi che possono essere "deviate". Selezionata la magia bersaglio questa viene salvata in una variabile e si attende che la pila risolvi la magia. Nel momento in cui lo stack va a risolversi, deflection controllerà se la magia da deviare è ancora presente nello stack: se è presente allora deflection rilancia il setTarget della magia selezionata permettendo all'utente di selezionare un nuovo bersaglio valido per la carta "deviata".

3.10 FalsePeace

Anche questa carta implementa Targettable, quindi all'interno del metodo play avremo solamente una chiamata a setTarget, che ci farà scegliere uno dei due giocatori su cui attivare l'effetto. Su resolve avremo l'effetto vero e proprio, quindi verrà creata una nuova SkipPhase, che avrà come parametro Combat (la fase da sostituire) e la durata, in questo caso di un turno. Dopo aver creato la SkipPhase la sostituiremo alla Combat del player target. La SkipPhase alla fine della durata provvederà in automatico a settare la DefaultCombatPhase.

3.11 Fatigue

Struttura identica a FalsePeace (3.10) con l'unica differenza che andreamo a sostituire la DrawPhase al posto della CombatPhase, anche in questo caso la durata sarà di un turno e la SkipPhase provvederà a sistemare la DefaultDrawPhase alla fine della sua durata.

3.12 SavorTheMoment

questa carta implementa l'interfaccia TriggerAction. Nel fare l'override del metodo 'resolve()', viene creata una nuova skipPhase (nello specifico, di Untap), di cui si effettua una push nell'array 'phases' (proprietà della classe Player). Sulla 'execute()', invece, viene soltanto lanciato il metodo 'nextPlayer()' per passare il turno al prossimo giocatore, successivamente seguendo il corso naturale del gioco si passerà il prossimo giocatore quindi è come richiamare due volte nextPlayer(), in questo modo il proprietario della carta avrà due turni consecutivi.

3.13 VolcanicHammer

Questa carta implementa l'interfaccia Targettable. Al suo interno è presente un oggetto targetCreature di tipo Creature, e uno targetPlayer di tipo Player, entrambi inizializzati a NULL. Quando la carta viene giocata, verrà lanciato il metodo 'play', al cui interno si richiama 'setTarget()' con cui si può scegliere se voler colpire un giocatore o meno. Rispondendo 1 (sì), viene lanciato il 'choosePlayer()', appunto perché si vuole colpire un giocatore; rispondendo 2 (no), viene lanciato il metodo 'chooseCreature()', proprio perché si vuole colpire non un giocatore, bensì una creatura. In entrami i metodi si deve fornire un indice (del giocatore/creatura da colpire). Se si è scelto come target un giocatore, l'oggetto targetPlayer verrà impostato uguale ad 'owner' o 'opponent', in base a chi si vuole colpire, mentre se il target scelto è una creatura, allora targetCreatura verrà impostata uguale alla creatura designata. Infine, nel 'resolve()', si controlla innanzitutto qual'è stata la scelta, andando a vedere se targetPlayer != NULL: se lo è, viene inflitto un danno di 3 al giocatore scelto, se non lo è, il danno viene inflitto alla creatura scelta.

3.14 WorldAtWar

Questa carta implementa l'interfaccia TriggerAction. Nel fare l'override del metodo 'resolve()', va ad effettuare l'untap di tutte le creature che hanno attaccato durante questo turno. Sulla 'execute()', vengono semplicemente create due nuove fasi, una Combat e una Main, che vengono eseguite proprio in quest'ordine, lanciandone il rispettivo metodo 'execute()'.