SAPIENZA
UNIVERSITÀ DI ROMA

Progetto Battaglia Pokémon - Java

RIGGI FILIPPO

Corso di laurea in Informatica, Università "La Sapienza" di Roma

Indice

1 Introduzione

Il progetto è stato sviluppato utilizzando le funzionalità di creazione e gestione dei repository offerte da GitHub. Il codice sorgente è pubblico, permettendo così la visualizzazione e la valutazione da parte del docente.

1.1 Strumenti Utilizzati

- IDE IntelliJ: per lo sviluppo iniziale del progetto.
- Eclipse: conversione del progetto in formato Eclipse su richiesta del docente, per agevolare l'importazione e la correzione.

2 Funzionalità Implementate

2.1 Creazione e Gestione dei Pokémon

- La classe `Pokemon` funge da classe base, mentre le sottoclassi rappresentano tipi specifici di Pokémon.
- La classe base contiene informazioni e caratteristiche comuni a tutti i Pokémon.
- Le sottoclassi includono le mosse caratteristiche e un `TreeMap` che associa ciascun livello del Pokémon alle mosse che può apprendere.

2.2 Gestione dei Giocatori

- I giocatori possono essere salvati e caricati, mantenendo informazioni sulle partite giocate e sui risultati ottenuti.
- Ogni giocatore possiede una squadra di Pokémon che cresce in livello e migliora le statistiche durante gli scontri.

2.3 Gestione delle Mosse

- Il metodo `getMoveByLevel(int pokeLevel)` restituisce la mossa che un Pokémon può apprendere a un determinato livello.
- Quando un Pokémon attaccante rende esausto un avversario, accumula esperienza. Se il livello raggiunge il valore necessario, si apre la finestra `FrameImparaMossa` per decidere se apprendere la nuova mossa e quale dimenticare.
- L'utilizzo di `TreeMap` consente un accesso ordinato e rapido alle mosse per livello.

2.4 Gestione delle Immagini

- Le immagini dei Pokémon e dei giocatori vengono serializzate per il salvataggio dei dati.
- L'attributo `image` è dichiarato `transient`, mentre `imageBase64` memorizza l'immagine come stringa codificata in Base64.

2.5 Stato di Salute dei Pokémon

- Il metodo `isAlive()` verifica se un Pokémon è vivo durante e dopo gli scontri.
- Se un Pokémon viene reso esausto, il giocatore sceglie un sostituto dalla squadra disponibile.

3 Menu del Gioco

3.1 GameSelect

- **Nuova Partita:** crea due nuovi giocatori e permette di selezionare la squadra Pokémon.
- **Continua Partita:** carica giocatori e Pokémon salvati, aggiornando caratteristiche e statistiche.
- La selezione dei giocatori avviene tramite pulsanti `player1` e `player2`.

3.2 ChoosePlayer

Consente di creare un nuovo giocatore, selezionando genere e nome.

3.3 ChooseTeam

- I giocatori selezionano Pokémon dal proprio Pokédex personale.
- È possibile visualizzare le statistiche dei Pokémon, sostituire quelli scelti e completare la squadra prima della battaglia.

4 Battaglia Pokémon

4.1 BattleView

Gestisce l'interfaccia grafica della battaglia, comprendendo:

- **PannelloAzioni:** mostra le mosse del Pokémon attivo e la squadra del giocatore.
- **PannelloInfoPokemon:** visualizza salute, esperienza e livello dei Pokémon.
- **Immagini Pokémon:** aggiornate in caso di cambio in campo.
- **Area di testo:** informazioni sul giocatore in attacco.
- **Score della battaglia:** punteggio in tempo reale ("best of 3").
- **VictoryPanel:** mostra il vincitore, statistiche aggiornate e stato dei Pokémon.

4.2 BattleController

Coordina la comunicazione tra la parte grafica (`BattleView`) e la logica della battaglia (`BattleModel`).

4.3 BattleModel

Gestisce la logica della battaglia:

- Riduzione HP durante gli attacchi.
- Incremento esperienza del Pokémon attaccante.
- Salita di livello dei Pokémon e aggiornamento delle statistiche.
- Gestione dei Pokémon esausti e sostituzioni obbligatorie.
- Controllo fine battaglia e aggiornamento punteggio.
- Ripristino salute Pokémon e salvataggio dati a fine partita.
- Calcolo del danno basato su mossa, attacco e difesa.

5 Design Pattern Adottati

Model-View-Controller (MVC): separazione della logica di gioco (Model), interfaccia utente (View) e gestione input (Controller). Migliora manutenibilità e gestione del codice.

6 Risorse Esterne Utilizzate

- **Java Swing:** gestione dell'interfaccia grafica, finestre, pulsanti e immagini.
- **ImageIO:** lettura e scrittura immagini, conversione verso formati serializzabili.
- **Base64:** codifica e decodifica immagini per la serializzazione.

7 Info ulteriori

- Il mio profilo GitHub.
- Repository GitHub del Progetto.
- Repository GitHub del collega Leonardo Gismondi, con cui ho svolto il progetto.