



SAPIENZA  
UNIVERSITÀ DI ROMA

Università "Sapienza" di Roma  
Facoltà di Informatica  
**Corso:** Metodologie Di Programmazione

# Documentazione JPokeBattle

**Author:** Alessia Cassetta  
**Matricola:** 2113909

Giugno 2024

# Indice

# 1 Feature Sviluppate

- **Minimo**

- Implementati 55 pokemon di prima generazione, con le loro mosse base e statistiche.
- Assegnati a tutti i Pokémon le due mosse di tipo neutro apprese al livello 1.
- Implementare le schermate “start”, battaglia, cambio pokémon, “you win”, e “game over”.
- Adottata Java Swing
- Far affrontare al giocatore una serie di avversari NPC, fino alla sua prima sconfitta.

- **Tipico**

- Preservare lo stato dei pokémon del giocatore nella serie di lotte.
- Implementate tutte le mosse dei Pokémon scelti, rispettando le loro meccaniche di funzionamento dipendenti dai loro tipi, ma ignorando i cambiamenti di stato come avvelenamento, stordimento, etc.
- Implementare una schermata leaderboard che mantenga i 10 record migliori.

- **Extra**

- Set crescita:
  - \* Implementare punti individuali e punti allenamento che migliorino le capacità dei pokémon sulla base delle vittorie, aggregandoli appropriatamente.
  - \* Implementare i meccanismi di passaggio di livello ed evoluzione dei Pokémon, incluso l'apprendimento di nuove mosse a determinati livelli.
- Set battaglia:
  - \* Implementare strategie per un comportamento “intelligente” degli avversari NPC, per supportare un'esperienza di gioco appagante.

Queste Feature sono state implementate in modo incrementale, partendo dal Minimo e aggiungendo le funzionalità Tipiche e Extra. Ogni Feature è stata testata singolarmente, per garantire il corretto funzionamento del codice.

## 2 Decisioni e design pattern adottati

Il progetto è basato su un'architettura a classi, in cui ogni classe rappresenta un'entità del gioco. Le classi principali sono:

- **Pokemon**: rappresenta un Pokémon, con le sue statistiche, mosse e punti esperienza.
- **Ability**: rappresenta una mossa, con il suo nome, tipo, potenza, precisione e punti mossa.
- **Coach**: rappresenta un allenatore, con il suo nome, i suoi Pokémon e il suo stato.
- **Battle Frame**: rappresenta una schermata, con i suoi elementi grafici e le sue azioni.
- **Battle**: rappresenta una battaglia, con i suoi stati e le sue azioni.
- **Change Pokemon**: rappresenta un cambio Pokémon, con i suoi stati e le sue azioni.

Ho deciso di realizzare tre classi principali: **CreateObjectPokemon**, **Ability** e **Coach**.

- **CreateObjectPokemon**: è una classe che si occupa di creare i Pokémon, assegnando loro le mosse e le statistiche. Essa contiene un metodo per creare i Pokémon che dato in input un intero che rappresenta l'indice del Pokémon nel pokédex e il livello, restituisce un oggetto di tipo **Pokemon**. Se noi non forniamo il livello del pokémon, il metodo lo assegnerà in modo randomico seguendo dei parametri specifici. In totale sono stati implementati 55 Pokémon di prima generazione. Ma all'inizio del gioco si potranno scegliere solo 6 pokémon base.
- **Ability**: è una classe che si occupa di creare le mosse, assegnando loro il nome, il tipo e la potenza. Esso si basa su una concatenazione di if che in base al livello del pokémon verifica se la mossa può essere appresa o meno.
- **Npc**: è una classe che si occupa di creare l'allenatore nemico, assegnandogli il nome, il Team dei pokemon e lo stato.

Ogni classe è stata progettata per essere il più possibile coesa e con un'alta coesione. Inoltre, è stata adottata l'ereditarietà per le classi **Pokemon** e **Ability**, in modo da poter creare facilmente nuovi Pokémon e nuove Mosse.

## 2.1 BattleFrame e Battle

La schermata più importante è la schermata della **battaglia**, in cui il giocatore può scegliere la mossa da usare o cambiare Pokémon. Questa schermata viene implementata da **BattleFrame** che estende la classe `JFrame`, e contiene tutti gli elementi grafici e le azioni che si possono fare durante la battaglia. Essa implementa l'interfaccia **BattleEventListener** che contiene i metodi per gestire le azioni del giocatore durante la battaglia. **BattleEventListener** comunica con la classe **Battle**, che contiene la logica della battaglia e i suoi stati.

## 2.2 La classe RecapBattle e MVC

Per quanto riguarda la classe **RecapBattle**, essa contiene il recap di fine battaglia, ossia un pannello che mostra il risultato della battaglia e le statistiche dei Pokémon. In questo caso ogni pokémon del team avrà la possibilità di poter o evolversi o cambiare una mossa già appresa in precedenza e in caso sia in possesso di 4 mosse potrà decidere di dimenticarne una per poterne apprendere una nuova.

La schermata di **RecapBattle** e di **EvolutionFrame** comunicano tramite il modulo **PokemonModule** che contiene i metodi per gestire l'evoluzione ed estende la classe **Observable**. In questo modo la classe **EvolutionFrame** può notificare la classe **PokemonModule** che si occuperà di gestire l'evoluzione del pokémon e che aggiornerà il frame di **RecapBattle**.

Questo tipo di approccio viene comunemente chiamato **Observer Pattern** e permette di notificare e aggiornare automaticamente tutti gli oggetti che sono interessati a un determinato evento.

Oppure più comunemente chiamato MVC ossia **Model-View-Controller**, in cui il Model rappresenta i dati, la View rappresenta l'interfaccia grafica e il Controller rappresenta la logica del programma.

## 2.3 La classe Battle

In **Battle** c'è tutta la logica della battaglia ed il metodo principale è **SetTurn** che si occupa di gestire la battaglia tra il giocatore e l'avversario. Quando non è il turno del player i bottoni delle mosse sono disabilitati e viceversa. Inoltre, se il giocatore decide di cambiare il pokémon, la classe **ChangePokemon** si occuperà di gestire il cambio del pokémon e il turno viene passato all'avversario. Un altro metodo molto importante è **npcLogic** che si occupa di svolgere tutta la logica del nemico, ossia di scegliere la mossa da usare e di attaccare il giocatore e se vede che il suo pokémon è in difficoltà decide di cambiarlo. Il check del pokémon viene eseguito in caso ogni 5 turni. IL nemico sceglie quale mossa far scagliare dal suo pokémon in modo randomico

### 3 Risorse Utilizzate

Per quanto riguarda le risorse utilizzate, ho utilizzato il sito **pokeapi.co** per ottenere i dati dei Pokémon e delle mosse inserendo i dati nel metodo della classe **CreateObjectPokemon**. Inoltre, ho utilizzato il sito **pokemondb.net** per ottenere i dati delle statistiche dei Pokémon. Ogni pokemon che ho inserito nel progetto ha anche una sprite (immagine) fronte e retro che reperisco tramite chiamata URI dal GitHub **<https://github.com/PokeAPI/sprites>**.

Background e altro sono immagini reperite da internet e si trovano nella cartella **Image** del progetto. L'immagine Background del Pokédex l'ho realizzata io stessa modificando graficamente un'immagine che ho fatto generare da un AI chiamato CraftBench.

Nel progetto è presente una cartella **Javadoc** che contiene la documentazione del codice in formato HTML e una cartella chiamata **Shared** che contiene classi comuni a tutti. Infatti ho creato una classe chiamata **RelativePath** che mi permette di restituire il path **assoluto** di ogni file di cui necessito, questo perché il progetto è stato sviluppato su due sistemi operativi diversi e quindi per evitare problemi di path ho deciso di creare questa classe.

In Shared sono presenti anche dei file .ttf che ho utilizzato in **Style.java** o in altre parti per cambiare il font del testo.

**Style** è una classe che permette di cambiare lo stile del testo, del bottone e del titolo. Questa classe può essere utilizzata in qualsiasi parte del progetto.

### 4 Paradigmi di Programmazione

Il progetto è stato sviluppato seguendo il paradigma di programmazione ad oggetti, in cui ogni classe rappresenta un'entità del gioco. Inoltre, ho utilizzato il paradigma di programmazione procedurale per la gestione delle battaglie e delle schermate.

## 5 UML del Progetto

## 6 Descrizione del progetto

Il progetto è organizzato in package, in cui ogni package rappresenta una funzionalità del gioco. I package principali sono:

- **Battle**: vi sono le classi che rappresentano la battaglia tra il giocatore e l'avversario.
- **Generic**: vi sono le classi che rappresentano le funzionalità generiche del gioco.
- **Game**: vi sono classi per la gestione del gioco. Come le classi Team, Coach, Npc e etc.
- **Pokemon**: vi sono le classi che rappresentano i Pokémon e le loro mosse.
- **Shared**: vi sono le classi comuni a tutti i package.

## 7 Descrizione delle schermate di gioco

### 7.0.1 Schermata Start

La schermata Start è la schermata iniziale del gioco, in cui il giocatore può iniziare una nuova partita o vedere la leaderboard.

## 8 Conclusioni



Figura 1: Schermata Start