# **Pokemon TCG Deck Simulator Documentation**

## **1. Introduction**

### **Overview**

The Pokemon TCG Deck Simulator is designed to simulate Pokemon Trading Card Game matches between AI-controlled players. It accurately replicates core gameplay mechanics, including deck building, energy management, status conditions, and AI-driven battle strategies.

### **Purpose and Goals**

The primary goal of this simulator is to provide a high-speed, object-oriented simulation of Pokemon TCG matches. This allows users to analyze deck performance, refine strategies, and explore AI-driven decision-making in a controlled environment.

### **Faithfulness to Pokemon TCG**

The simulator follows closely to official Pokemon TCG rules and mechanics:

* **Accurate Damage Calculations:** Includes type effectiveness, resistance, and energy requirements.
* **Turn-Based Gameplay:** Implements draw phases, attack phases, and status condition management.
* **Deck Management:** Simulates real-world deck-building strategies with Pokemon, Trainer, and energy cards.
* **AI-Driven Strategy:** Different play styles (Aggressive, Defensive, Setup) mimic real player behaviors.

### **Key Features**

* **Object-Oriented Design:** The simulator follows strong OOP principles with encapsulation, inheritance, and polymorphism.
* **High-Performance execution:** Optimized for fast execution, capable of simulating tens of thousands of games efficiently.
* **Comprehensive Statistics:** Tracks game performance, deck efficiency, and AI decision outcomes.

## **2. Technical Overview**

### **Object-Oriented Programming (OOP) Principles**

The simulator follows a robust object-oriented design, ensuring modularity, maintainability, and scalability:

* **Encapsulation:** each class has well-defined responsibilities, ensuring data is protected and accessed only through appropriate methods.
* **Inheritance:**
  + Pokemon serves as a base class, with specific Pokemon (Pikachu, Charizard, etc.) extending it to implement unique stats and abilities.
  + Trainer extends Card, defining Trainer-specific effects like Potion and RareCandy.
* **Polymorphism:**
  + Different AI strategies (AggressivePlayStyle, DefensivePlayStyle, SetupPlayStyle) override shared methods to implement unique behaviors.
  + performAttack() is overridden by specific Pokemon to modify base attack mechanics.

### **Performance Considerations**

To ensure the simulator runs efficiently, several optimizations are implemented:

* **Efficient Data Structures:**
  + HashMap for storing energy attachments to Pokemon, enabling quick retrieval and updates.
  + ArrayList for managing deck cards, ensuring fast shuffling and draw operations.
* **Optimized AI Decision-Making:**
  + AI evaluates multiple possible actions each turn, but uses pruning to eliminate redundant calculations.
  + Strategies prioritize key factors like energy attachment, Pokemon selection, and attack effectiveness.
* **Multithreading Potential:**
  + The simulator is designed with future scalability in mind, allowing parallel execution of multiple game simulations.

## **3. Core Components**

### **Game Logic**

* **GameState**: Manages the overall state of the game, including turn tracking, phase transitions, and win conditions.
* **Player**: Represents a player in the game, handling deck management, energy attachment, and Pokemon selection.
* **DeckBuilder**: Constructs predefined decks for AI players, incorporating different play styles.

### **Card System**

* **Pokemon**: Base class for all Pokemon cards, defining attributes such as HP, attacks, energy costs, and evolution mechanics.
* **Trainer**: Represents Trainer cards, implementing effects such as healing (Potion), evolution (RareCandy), and card draw (Professor's Research).
* **Energy**: Represents energy cards that are required for Pokemon attacks.
* **Attack**: Defines attack attributes, including name, damage, status effects, and energy requirements.

### **AI Strategies**

* **AggressivePlayStyle**: Focuses on high-damage attacks, prioritizing powerful Pokemon and immediate offense.
* **DefensivePlayStyle**: emphasizes healing, status conditions, and maintaining board control.
* **SetupPlayStyle**: Prioritizes evolution chains and long-term game planning, preparing for late-game dominance.

### **Battle Mechanics**

* **PerformAttack**: executes attack logic, applying damage calculations and checking for weaknesses, resistances, and status conditions.
* **EnergyCost**: Manages energy requirements for attacks, ensuring proper attachment before execution.
* **StatusEffects**: Implements conditions like Burned, Paralyzed, and Asleep, affecting gameplay accordingly.

### **Simulation System**

* **SingleAIGameTester**: Runs individual AI matches to analyze performance.
* **AITournament**: Conducts multiple AI matches in a tournament-style format, gathering statistical insights.

## **4. Faithfulness to Pokemon TCG**

### **Evolution Mechanics**

* Pokemon evolve following proper evolutionary chains (e.g., Charmander → Charmeleon → Charizard).
* RareCandy allows direct evolution from Basic to Stage 2 Pokemon, mirroring official game mechanics.
* Evolution increases HP, unlocks new attacks, and modifies retreat costs.
  + Damage counter and status effects carry

### **Attack Damage Calculations**

* Attack damage considers:
  + **Base damage** assigned to the move.
  + **Energy requirements** needed for execution.
  + **Weakness, Same Type Attack Bonus, and Resistance modifiers** affecting total damage output.
* Status conditions (e.g., Sleep Powder causing Asleep) impact Pokemon’s ability to attack or retreat.

### **Trainer Cards and effects**

* Cards like Potion, Professor’s Research, and UltraBall function identically to their real-world cards:
  + **Potion** heals 30 HP from a Pokemon.
  + **Professor’s Research** discards the player’s hand and draws 7 new cards.
  + **UltraBall** allows the player to discard two cards and search for any Pokemon.

### **Deck-Building Rules**

* Decks are built following standard Pokemon TCG deck construction rules:
  + 60-card limit per deck.
  + Balanced distribution of Pokemon, Trainer, and energy cards.
  + AI considers synergy between cards when constructing decks.

### **Turn-Based Gameplay**

* The game follows the same turn order as the Pokemon TCG:
  + Draw Phase → Setup Phase → Main Phase (playing Pokemon, attaching energy, using Trainers) → Attack Phase.
  + Players alternate turns, with effects carrying over appropriately.

### **Error Handling**

* The simulator includes exception handling to prevent unexpected crashes.
* Errors such as invalid card actions, incorrect energy attachments, and game rule violations are caught and handled somewhat gracefully.
* Comprehensive debugging tools include log messages and error reports to trace issues efficiently.
  + Toggled on and off at the top of each file with a boolean ‘debug’
    - Intentional design choice to help isolate bugs

### **Event Logging and Statistic Tracking**

* The simulator records key game events such as attacks, knockouts, and energy attachments.
* AI decision-making steps are logged to allow performance analysis and debugging.
* After each game, detailed statistics such as:
  + Win rates by AI strategy
  + Average turns per game
  + Effectiveness of different deck compositions
  + Usage frequency of specific trainer cards
* Logs can be exported for further analysis and tuning AI behavior.

### **Game Balancing**

* + Each player’s strategy has been meticulously optimized and edited to ensure that there is a somewhat fair probability of either playstyle or deck securing a win.
  + Conducted a round robin style tournament to test and prove balancing efforts.

## **5. Performance & Optimization**

### **High-Speed execution**

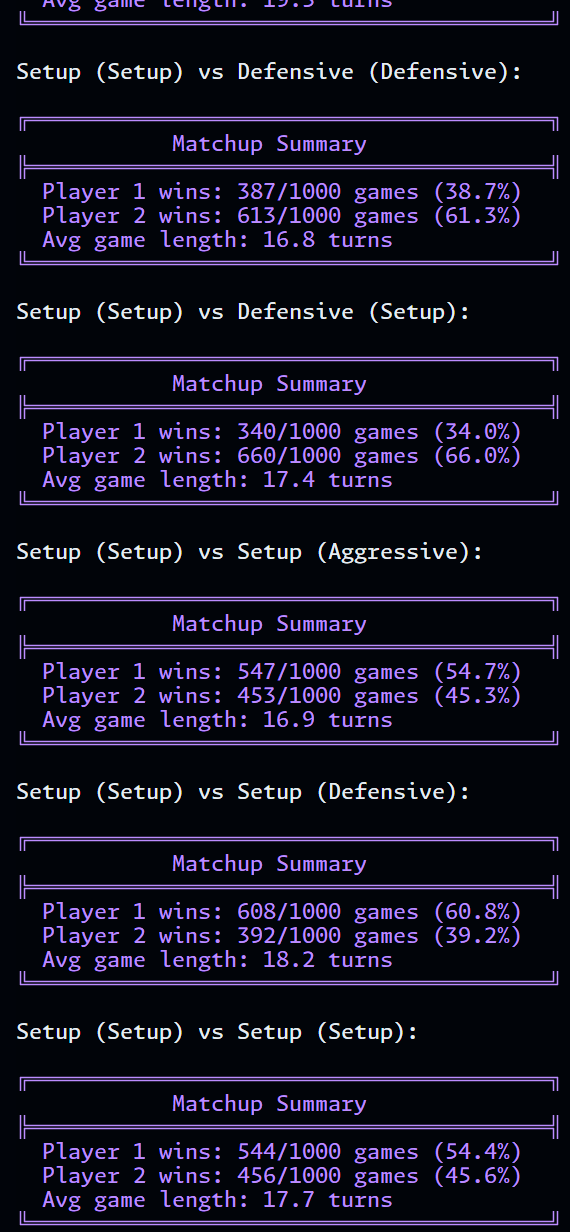
* Poorly but somewhat optimized algorithms ensure each simulation runs in minimal time.
* AI decisions are streamlined using caching mechanisms and decision-tree pruning.

### **AI Decision-Making efficiency**

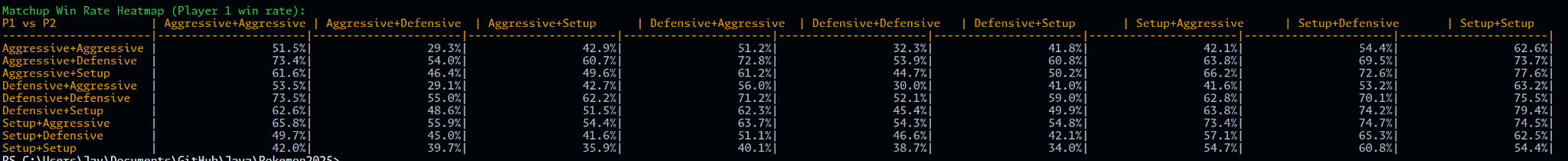
* AI selects optimal moves based on heuristics, considering:
  + expected damage output.
  + energy availability.
  + Probability of drawing key cards.
* Decision-making is refined through statistical evaluation of past game outcomes.

### **Scalable Multi-Game Simulations**

* The simulator can execute thousands of matches in batch mode, gathering statistical insights.
* I wanted to try to multithread it to test more decks and playstyles faster

**OUTPUT:**

**AI tournament**



SINGLE GAMES:  
