Pokemon Game

By: Jacob Attia

1. Program Overview

This project simulates a simplified version of the Pokémon Trading Card Game (TCG). It allows:

* Two players draw from 60‑card decks (each containing Pokémon, Trainer, and Energy cards).
* Player 1 to interactively choose actions each turn (attack, attach Energy, play Trainer).
* Player 2 to operate automatically by drawing, attaching correct Energy to its active Pokémon, and attacking if possible.

The game continues until one player’s Pokémon can’t continue, or a player has no Pokémon left to replace a knocked‑out active Pokémon.

1. Class Structure

The code is split into multiple classes, each in its own file. This keeps the project organized and makes the program easier to maintain. Below is a description of each class:

2.1 Card (Card.java)

Purpose:

* A marker interface that represents any type of card in the game.
* Classes like EnergyCard, TrainerCard, and Pokemon implement this interface, grouping them all under a common type.

2.2 Attack (Attack.java)

Purpose:

* Represents a single attack that a Pokémon can perform.
* Stores the attack’s name, a list of required energies, and the amount of damage it does.

Key Fields:

* name: The attack’s name (Ex. “Scratch”)
* energyCost: A list of energy types needed to perform this attack (Ex. [“Fire”, “Basic”]
* damage: The amount of damage the attack does to the opponent

2.3 EnergyCard (EnergyCard.java)

Purpose:

* Represents an energy card that can be attached to a Pokémon
* Each energy card has a specific type ( "Fire", "Water", "Grass", "Electric", or "Basic").

Key Methods:

* getType(): Returns the energy type

2.4 TrainerCard (TrainerCard.java)

Purpose:

* Represents a trainer card that applies some effect when played.
* Typical effects include drawing extra cards, healing damage, or retrieving Energy cards.

Key Method:

* applyEffect(Player player): Applies the card’s effect to the specified player. For example:
* **Professor's Research**: Discards the player’s hand and draws 7 cards.
* **Bill**: Draws 2 cards.
* **Lana**: Heals 50 damage if the player’s active Pokémon is Water type.
* **Gym Trainer**: Draws 2 cards, plus an extra 2 if the player had a Pokémon knocked out last turn.

2.5 Deck (Deck.java)

Purpose:

* Stores a list of 60 cards and manages shuffling, drawing, and adding cards back to the deck.

Key Methods:

* shuffle(): Randomly reorders the deck.
* drawCard(): Removes and returns the top card, or null if empty.
* drawCards(int n): Removes and returns up to “n” cards from the top.
* addCard(Card card): Adds a card to the bottom of the deck.

2.6 Pokemon (Pokemon.java)

Purpose:

* Represents a Pokemon card.
* Stores the Pokémon’s name, type (Fire, Water, Grass, Electric, or Basic), current HP, a list of attacks, and attached energies.

Key Methods:

* Type: Pokemon elemental type
* hp/maxHP: Current and maximum HP
* Attacks: A list of Attack objects the Pokemon can perform
* attachedEnergies: Tracks the type of energy cards currently attached
* AttachEnergy(EnergyCard energy): Attaches an energy card to the Pokémon (only if proficient).
* IsEnergyProficient(EnergyCard energy): True if the energy card is "Basic" or matches the Pokémon’s type.
* CanUseAttack(int attackIndex): Checks if the Pokémon’s attached energies satisfy the cost for the attack at attackIndex.
* Attack(Pokemon opponent, int attackIndex): Applies the attack’s damage to the opponent Pokémon’s HP.

### 2.7 Player (Player.java)

Purpose:

* Models a player in the game, storing their deck, hand, active Pokémon, bench, and relevant game state like whether a Pokémon was knocked out last turn.

Key Fields/Methods:

* hand: The player’s current cards.
* activePokemon: The Pokémon currently in battle for this player.
* bench: Up to 5 additional Pokémon waiting to battle.
* drawInitialHand(): Draws 7 cards to start the game.
* mulligan(): Returns the player’s hand to the deck, reshuffles, draws 7 new cards (if no Pokémon is drawn initially).
* setupActiveAndBench(): Chooses the first Pokémon found in the hand as the active Pokémon, then up to 5 more go onto the bench.
* replaceActivePokemon(): When the active Pokémon is knocked out, tries to promote the first Pokémon from the bench.
* playTrainerCard(String trainerName): Searches the hand for a Trainer card with the given name and plays it.
* hasValidMove(): Checks if the player can do anything at all on their turn (attack, attach proficient energy, or play a Trainer card).

### 2.8 PokemonGame (PokemonGame.java)

Purpose:

* Orchestrates the overall game flow, from initial shuffling and drawing to the main battle loop where each player takes turns.

Key Steps:

1. Construction: Builds two 60‑card decks, shuffles them, and creates two Player objects.

2. startGame():

* Draws initial 7‑card hands for both players.
* Checks for a playable hand (mulligan if needed).
* Demonstrates playing a Trainer card (Bill) for Player 1.
* Each player chooses an active Pokémon and sets up a bench.
* A coin flip decides who attacks first.
* Battle Loop:
* Player 1 is interactive: you choose actions (attack, attach Energy, play Trainer, or end turn).
* Player 2 is automated: it attaches all proficient energies if it cannot attack, then tries to use its first attack.
* If a Pokémon is knocked out, the Player tries to replace it from their bench; if impossible, the other Player wins.

Running the Game:

1. Compile
2. Run: java PokemonGame
3. Interaction:

* As Player 1, you see a menu each turn with options.
* Player 2 is fully automated.
* The game ends when one player loses all their Pokémon.

## 3. How to Use the Program

1. Setup: Place all Java files (Card.java, Attack.java, EnergyCard.java, TrainerCard.java, Deck.java, Pokemon.java, Player.java, and PokemonGame.java) in the same directory (or appropriate packages).
2. Compile:
3. Run:  
   - java PokemonGame

4. Gameplay:

* You will be prompted to choose actions on your turn (Attack, Attach Energy, Play Trainer, End Turn).
* The automated opponent tries to attach all proficient energy in its hand if it can’t already attack, then tries to attack.

## 4. Program Flow Summary

1. Deck Creation: Each player’s deck is built with 60 cards:

* 20 Pokémon (Charmander, Bulbasaur, Squirtle, Psyduck, Bellsprout, Flareon)
* 20 Trainer cards (Professor’s Research, Bill, Lana, Gym Trainer)
* 20 Energy cards (Water, Grass, Fire, Electric, Basic)

1. Initial Setup:

* Each player draws 7 cards. If no Pokémon, mulligan triggers.
* A demonstration Bill card is played by Player 1 if available.
* Each player’s first Pokémon becomes active; up to 5 more are placed on the bench.

1. Coin Flip & Battle:

* The coin flip decides who goes first.
* Player 1 is interactive, while Player 2 automatically tries to attach proficient energy and then attack.

1. Knockouts:

* If the opponent’s active Pokémon is knocked out, they try to replace it with one from the bench.
* The game ends when a player has no Pokémon left to continue

## 

## 

## 

## **5. Conclusion**

This refactored Pokémon TCG simulation provides a structured approach to battling with simplified TCG rules. With each class responsible for one key aspect of gameplay (cards, decks, players, or the game flow), it is easier to maintain and extend. Player 1 experiences an interactive command line menu each turn, while Player 2 is automated for demonstration purposes.