# **Pokemon TCG Deck Simulator Manual**

## **1. Creating a Deck**

### **Steps to Build a Deck**

1. **Initialize a New Deck:**
   * Use **DeckBuilder.buildAggressiveDeck(int playerNum)** for an aggressive deck.
   * Use **DeckBuilder.buildDefensiveDeck(int playerNum)** for a defensive deck.
   * Use **DeckBuilder.buildSetupDeck(int playerNum)** for a setup-oriented deck.
2. **Add Cards to the Deck:**
   * Use **deck.addCard(Card card)** to manually add a card.
   * Ensure a balanced composition of Pokemon, Trainer, and Energy cards.
     1. MUST BE 60 OR LESS

### **Useful Methods:**

* **deck.removeCard(Card card):** Removes a specific card.
* **deck.getCardCount():** Returns the number of cards in the deck.
* **deck.getTopCard():** Draws the top card from the deck.

## **2. Creating a Pokémon**

### **Steps to Build a Pokémon**

1. **Extend the Pokemon Class:**

public class Examplemon extends Pokemon {

public Examplemon(int playerNum) {

super("Examplemon", "Example Pokémon", 100, 100, PokemonType.FIRE,

PokemonType.WATER, PokemonType.GRASS, 1, playerNum, 1, MajorStatusCondition.NONE);

Attack ember = new Attack("Ember", 30, PokemonType.FIRE, 1,

MajorStatusCondition.NONE, false, false, false);

ember.setEnergyCost(Energy.EnergyType.FIRE, 1);

this.addAttack(ember);

}

}

1. **Define Attacks and Energy Costs:**
   * Use **addAttack(Attack attack)** to add attacks.
   * Use **setEnergyCost(Energy.EnergyType type, int amount)** to assign energy requirements.
2. Set evolutions using stage and overriding **Evolve();**

### **Useful Methods:**

* **pokemon.setStatus(MajorStatusCondition condition)**: Applies a status effect.
* **pokemon.performAttack(String attackName, Pokemon target)**: Executes an attack.

## **3. Creating a Trainer Card**

### **Steps to Build a Trainer**

1. **Extend the Trainer Class:**

public class ExampleTrainer extends Trainer {

public ExampleTrainer() {

super("Example Trainer", "Draw two cards.", false, Pokemon.UNOWNED);

}

@Override

public boolean playCard(GameState gameState, Player player) {

player.drawCard();

player.drawCard();

return true;

}

}

1. **Define Effects in playCard Method:**
   * Manipulate the game state (e.g., drawing cards, modifying Pokémon stats, searching the deck).

### **Useful Methods:**

* **player.drawCard()**: Draws a card from the player’s deck.
* **gameState.applyEffect(Trainer trainer, Player player)**: Applies trainer effects.

## **4. Adding a New Playstyle**

### **Steps to Define a Playstyle**

1. **Extend the PlayStyle Class:**

public class ExamplePlayStyle extends PlayStyle {

@Override

public void makeSetupDecisions(GameState gameState, Player player) {

System.out.println("Setting up Example PlayStyle");

}

@Override

public void makeTurnDecisions(GameState gameState, Player player) {

System.out.println("Executing Example PlayStyle decisions");

}

}

1. **Implement Decision-Making:**
   * makeSetupDecisions(GameState gameState, Player player): Defines how the AI sets up its board.
   * makeTurnDecisions(GameState gameState, Player player): Defines AI behavior during its turn.

### **Useful Methods:**

* **player.playCard(Card card)**: Plays a Trainer or Pokémon card.
* **player.attachEnergy(Energy energy, Pokemon pokemon)**: Attaches energy to a Pokémon.
* **gameState.runAITurn()**: Executes the AI’s turn logic.