# Code Documentation for PokemonProject1

By Rachel Hussmann

## Overview

The purpose of PokemonProject1 was to create a fully working Pokémon TCG simulator. This project has twenty-four classes, each that handle a different function within the game.

## How It Works

### Attack

The Attack class is responsible for holding information about a Pokemon's attack. The class has three variables:

- ArrayList<Energy> costOfAttack
- String attackName
- int damage

Each variable has its own set of getters and setters. The class also has an additional constructor that accepts values for each variable and assigns it to them. Attack also has an overridden toString method to make it easier to output information about the Attack object.

### Card

The Card class is responsible for holding information about a specific card. This class is a superclass to multiple classes including:

- Deck
- Energy
- Pokémon
- Trainer

The Card class has the default constructor and three additional constructors, one for each of the different types of cards. The class has five variables:

- String typeOfCard
- String nameOfCard
- Pokemon pokemon
- Trainer trainer
- Energy energy

The class has getters and setters for each variable. The class also has an additional method called checkForMulligan. The checkForMulligan method accepts an ArrayList<Card> of the player's hand and it checks to see if the player has any Pokemon in their hand. If they do not, the method returns true, because a mulligan has been found. It returns false if a Pokemon has been found in the player's hand. The class also has an overridden toString method to make it easier to output information about the Card object.

**Charmander**

The Charmander class is a subclass of the Pokemon class. It does not hold any information about Charmander but instead sends the information to the Pokemon class. The class has one constructor: the default constructor, which sends all information about the Charmander, including the name of the card, the HP, the attacks, the weakness, the type, the type of card and the retreat cost to the superclass.

**Cynthia**

The Cynthia class is a subclass of the Trainer class. It does not hold any information about Cynthia but instead sends the information to the Trainer class. The class has one constructor: the default constructor, which sends all information about the Cynthia card, including the name of the card, the type of card, the type of trainer card and the description of the card. The class also has an overridden method called useAbility. The useAbility method accepts one parameter: The player who used the card. The useAbility method uses the card's special ability, which for Cynthia is "Shuffle your hand into your deck. Then, draw 6 cards." The method returns nothing.

**Deck**

The Deck class is a subclass of the Card class. The only variable of the Deck class is an ArrayList of cards called deckOfCards. This variable has a getter and setter. The Deck class has one constructor: the default constructor that initializes the deckOfCards variable. The Deck class has eight methods:

- generateDeck
    - Parameter: None
    - Functionality – Generates a standard deck of cards with 20 pokemon, 15 trainer cards, and 20 energy cards.
    - Returns: Nothing
- generateDeckMulligan

    - Parameters: int numOfPokemon – The number of pokemon to be added to the deck, int numOfEnergies – The number of energies to be added to the deck
    - Functionality – Generates deck of 60 cards with the specified number of pokemon and energies for the MulliganMonteCarlo class and simulation
    - Returns: Nothing
- generateDeckRareCandy
    - Parameters: int numOfPokemon – The number of pokemon to be added to the deck, int numOfEnergies – The number of energies to be added to the deck, int numOfTrainers – The number of trainer cards to be added to the deck
    - Functionality – Generates a new deck of 60 cards for use in the BrickMonteCarlo class and simulation
    - Returns: Nothing
- shuffle
    - Parameter: None
    - Functionality – Shuffles the current deckOfCards
    - Returns: Nothing
- pickTopCard
    - Parameter: None
    - Functionality – Picks the first card from the deckOfCards variable
    - Returns: Card – The first card from the deckOfCards variable
- returnHandToDeck
    - Parameter: ArrayList<Card> hand – The hand of cards that the player needs returned to the deck
    - Functionality – Returns a hand of cards back to the deck.
    - Returns: Nothing
- findFirstPokemon
    - Parameter: None
    - Functionality – Finds the first Pokemon card in the deck
    - Returns: Pokemon – The first Pokemon card in the deck
- findFirstEnergy
    - Parameter: None
    - Functionality – Finds the first energy card in the deck
    - Returns: Energy – The first energy card in the deck

**Eevee**

The Eevee class is a subclass of the Pokemon class. It does not hold any information about Eevee but instead sends the information to the Pokemon class. The class has one constructor: the default constructor, which sends all information about the Eevee, including the name of the card, the HP, the attacks, the weakness, the type, the type of card and the retreat cost to the superclass.

**Electric**

The Electric class is a subclass of the Energy class. It does not hold any information about the Electric energy but instead sends the information to the Energy class. The class has one constructor: the default constructor, which sends all information about the Electric energy, including the type of energy, the name of the card and the type of card to the superclass.

**Energy**

The Energy class is a subclass of the Card class. It holds information about the type of energy card but sends all other information to the Card class. This information is held in a string variable called type, which has a getter and setter method. The Energy class has one constructor: the default constructor, which holds the information about the type of energy the card is, but it sends all other information, such as the type of card (trainer, energy, or pokemon) and the name of the card to the superclass. The Energy class has one additional method called useEnergy which accepts one parameter: The player who used the card. The useEnergy method attaches the energy being used to a pokemon of the player's choice. The method returns nothing.

**Fire**

The Fire class is a subclass of the Energy class. It does not hold any information about the Fire energy but instead sends the information to the Energy class. The class has one constructor: the default constructor, which sends all information about the Fire energy, including the type of energy, the name of the card and the type of card to the superclass.

**Lapras**

The Lapras class is a subclass of the Pokemon class. It does not hold any information about Lapras but instead sends the information to the Pokemon class. The class has one constructor: the default constructor, which sends all information about the Lapras, including the name of the card, the HP, the attacks, the weakness, the type, the type of card and the retreat cost to the superclass.

**Pikachu**

The Pikachu class is a subclass of the Pokemon class. It does not hold any information about Pikachu but instead sends the information to the Pokemon class. The class has one constructor: the default constructor, which sends all information about the Pikachu, including the name of the card, the HP, the attacks, the weakness, the type, the type of card and the retreat cost to the superclass.

**Player**

The Player class is responsible for handling the state of the player and contains methods to run the player's turn. The class has eleven variables:

- Deck deck
- ArrayList<Card> hand
- ArrayList<Card> prizeDeck
- ArrayList<Card> bench
- Pokemon activePokemon
- ArrayList<Card> discardPile
- String playerName
- Scanner userInput
- int numberOfSupporterCardsUsed
- int numberOfEnergiesUsed
- boolean xSpeedUsed

The class has one constructor: the default constructor, which creates a new deck, shuffles the deck, creates a new hand, prize deck, bench, discard pile, sets the numberOfSupporterCardsUsed and numberOfEnergiesUsed to 0 and sets the xSpeedUsed variable to false. All variables have getters and setters. The class has eighteen other methods:

- createHand
    - Parameter: None
    - Functionality – Picks 7 cards to add to the player's hand
    - Returns: Nothing
- pickPrizeDeck
    - Parameter: None
    - Functionality – Picks 6 cards to add to the player's prize deck
    - Returns: Nothing
- removeCardFromHand
    - Parameter: Card cardToBeRemoved – The card to be removed from the player's hand

  - Functionality – Removes the given card from the hand of the player
  - Returns: Nothing
- removeCardFromBench
  - Parameter: Card cardToBeRemoved – The card to be removed from the player's bench
  - Functionality – Removes the given card from the player's bench
  - Returns: Nothing
- addCardToDiscard
  - Parameter: Card cardToBeAdded – The card to be added to the discard pile
  - Functionality – Adds the given card to the player's discard pile
  - Returns: Nothing
- addCardToHand
  - Parameters: None
  - Functionality – Takes the top card from the deck and adds it to the player's hand
  - Returns: Nothing
- addPrizeCardToHand
  - Parameters: None
  - Functionality – Takes the top card from the player's prize deck and adds it to their hand
  - Returns: Nothing
- addCardToBench
  - Parameter: Card cardToAdd – The card to be added to the bench
  - Functionality – Adds a card to the player's bench
  - Returns: Nothing
- isKnockedOut
  - Parameters: None
  - Functionality – Checks to see if the player's active pokemon
  - Returns: boolean – True if the player's active pokemon is knocked out, False if the player's active pokemon is still alive
- displayPlayerStats
  - Parameter: None
  - Functionality – Displays the player's current statistics including number of prize cards left in their prize deck, the current active pokemon, the HP of their active pokemon, the cards they have in their hand, the pokemon in their bench and the current number of cards they have left in their deck.
  - Returns: Nothing, but prints out statements
- runPlayerTurn
  - Parameter: boolean firstTurnCheck – Checks to see if it is the first turn of the game

  - Functionality – Runs through a player’s turn, allowing them to place pokemon on their bench, place an energy and use trainer cards
  - Returns: Nothing, but prints out statements
- displayPlayerStatsNoActive
  - Parameters: None
  - Functionality – Displays the player’s current statistics when they do not have an active pokemon
  - Returns: Nothing, but prints out statements
- noActivePokemon
  - Parameters: None
  - Functionality – Handles when a player does not have an active pokemon, specifically when the game begins, since the player is allowed to pick a pokemon from their hand
  - Returns: Nothing, but prints out statements
- noActivePokemonFromBench
  - Parameters: None
  - Functionality – Allows the player to pick a new active pokemon from their bench, typically done after their active pokemon has been knocked out
  - Returns: Nothing, but prints out statements
- battlePhase
  - Parameter: Player defendingPlayer – The player who is defending
  - Functionality – Allows the attacking player to pick an attack to use, checks if it can be used, and attacks the defending player’s active pokemon
  - Returns: Nothing, but prints out statements
- RetreatActivePokemon
  - Parameters: None
  - Functionality – Checks to see if the active pokemon can be retreated, retreats the pokemon to the bench, then allows the player to choose a new active pokemon from their bench
  - Returns: Nothing, but prints out statements
- pickPokemonFromBench
  - Parameters: None
  - Functionality – Allows the player to pick a pokemon from their bench to replace their active pokemon, either due to a knock out or retreating
  - Returns: Nothing, but prints out statements
- useCard
  - Parameter: Card cardToUse – The card to be used
  - Functionality – Allows the player to use a card from their hand
  - Returns: Nothing, but prints out statements

**PokeBall**

The PokeBall class is a subclass of the Trainer class. It does not hold any information about PokeBall but instead sends the information to the Trainer class. The class has one constructor: the default constructor, which sends all information about the PokeBall card, including the name of the card, the type of card, the type of trainer card and the description of the card. The class also has an overridden method called useAbility. The useAbility method accepts one parameter: The player who used the card. The useAbility method uses the card's special ability, which for PokeBall is "Flip a coin. If heads, search your deck for a Pokemon, reveal it, and put it into your hand. Then, shuffle your deck." The method returns nothing.

**Pokemon**

The Pokemon class is a subclass of the Card class. It holds information about the Pokemon object. The Pokemon class has six variables:

- int hp
- ArrayList<Attack> attacks
- String weakness
- String type
- int retreatCost
- ArrayList<Energy> energiesAttached

All of the variables have their own getter and setter methods. The class has one constructor: the default constructor, which initializes the energiesAttached variable. The Pokemon class has four additional methods:

- checkIfAttackValid
  - Parameter: Attack attack – The attack that the player would like the pokemon to use
  - Functionality – Checks to see if the player can use the chosen attack by checking the number of energies required for the attack
  - Returns: boolean – True if the pokemon can use the attack, False if the pokemon cannot use the attack
- battle
  - Parameters: Attack pokemonAttack – The attack the attacking pokemon will use, Pokemon defendingPokemon – The defending pokemon that will be losing HP
  - Functionality – Manages a battle between two pokemon
  - Returns: Nothing
- attachEnergy

    - Parameter: Energy cardToAttach - The energy to attach to this pokemon object
    - Functionality – Attached the energy from the parameter to this pokemon object
    - Returns: Nothing
- checkForRetreat
    - Parameter: None
    - Functionality – Checks to see if the pokemon can retreat from active status
    - Returns: boolean – True if the pokemon has enough energies attached to retreat from active status, False if the pokemon does not have enough energies to retreat

**PokemonCardGame**

The PokemonCardGame class is responsible for running the game and holding variables associated with the game state. The class has three variables that help to contain information about the game state:

- Player player1
- Player player2
- Boolean firstTurnCheck

The class has seven methods:

- startGame
    - Parameter: None
    - Functionality –
    - Returns: Nothing, but prints out statements
- coinflip
    - Parameter: None
    - Functionality – Randomly generates a value and assigns a String value of heads or tails to it
    - Returns: String – The result of the coin flip
- mulligan
    - Parameter: Player playerWithMulligan
    - Functionality – Takes the hand of the player with a mulligan, returns it to their deck, and picks them a new hand
    - Returns: Nothing, but prints out statements
- player1Turn
    - Parameter: None
    - Functionality – Handles the core functionalities of player 1's turn
    - Returns: Nothing, but prints out statements

- player2Turn
    - Parameter: None
    - Functionality – Handles the core functionalities of player 2's turn
    - Returns: Nothing, but prints out statements
- runGame
    - Parameter: None
    - Functionality – Runs the Pokemon TCG game, by alternating the player turns and checking for win conditions
    - Returns: Nothing, but prints out statements
- addCardToOtherPlayer
    - Parameter: Player playerToGetExtraCard
    - Functionality – Handles when the other player should get another card due to a mulligan
    - Returns: Nothing, but prints out statements

**PokemonCardGameTester**

The PokemonCardGameTester class contains the main method and runs the program. It contains a tester object of the PokemonCardGame class. It tells the PokemonCardGame class to begin the game.

**ProfsLetter**

The ProfsLetter class is a subclass of the Trainer class. It does not hold any information about ProfsLetter but instead sends the information to the Trainer class. The class has one constructor: the default constructor, which sends all information about the ProfsLetter card, including the name of the card, the type of card, the type of trainer card and the description of the card. The class also has an overridden method called useAbility. The useAbility method accepts one parameter: The player who used the card. The useAbility method uses the card's special ability, which for ProfsLetter is "Search your deck for up to 2 basic energy cards, reveal them, and put them into your hand. Shuffle your deck afterwards." The method returns nothing.

**ProfsResearch**

The ProfsResearch class is a subclass of the Trainer class. It does not hold any information about ProfsResearch but instead sends the information to the Trainer class. The class has one constructor: the default constructor, which sends all information about the ProfsResearch card, including the name of the card, the type of card, the type of trainer card and the description of the card. The class also has an overridden method called useAbility. The useAbility method accepts one parameter: The player who used the card. The useAbility method uses the card's special ability, which for ProfsResearch is "Discard your hand and draw 7 cards." The method returns nothing.

**RareCandy**

The RareCandy class is a subclass of the Trainer class. It does not hold any information about RareCandy but instead sends the information to the Trainer class. The class has one constructor: the default constructor, which sends all information about the RareCandy card, including the name of the card, the type of card, the type of trainer card and the description of the card.

**Skarmory**

The Skarmory class is a subclass of the Pokemon class. It does not hold any information about Skarmory but instead sends the information to the Pokemon class. The class has one constructor: the default constructor, which sends all information about the Skarmory, including the name of the card, the HP, the attacks, the weakness, the type, the type of card and the retreat cost to the superclass.

**Steel**

The Steel class is a subclass of the Energy class. It does not hold any information about the Steel energy but instead sends the information to the Energy class. The class has one constructor: the default constructor, which sends all information about the Steel energy, including the type of energy, the name of the card and the type of card to the superclass.

**Trainer**

The Trainer class is a subclass of the Card class. It has two variables:

- String typeOfTrainerCard
- String descriptionOfCard

Each of these variables have a getter and a setter. The class also has a method called useAbility. It accepts one parameter: the player who used the card. The card prints out a statement "This trainer card does not have a special ability." To implement this method in all subclasses, this method should be overridden so that it does the trainer card's appropriate ability. The method returns nothing.

### Water

The Water class is a subclass of the Energy class. It does not hold any information about the Water energy but instead sends the information to the Energy class. The class has one constructor: the default constructor, which sends all information about the Water energy, including the type of energy, the name of the card and the type of card to the superclass.

### XSpeed

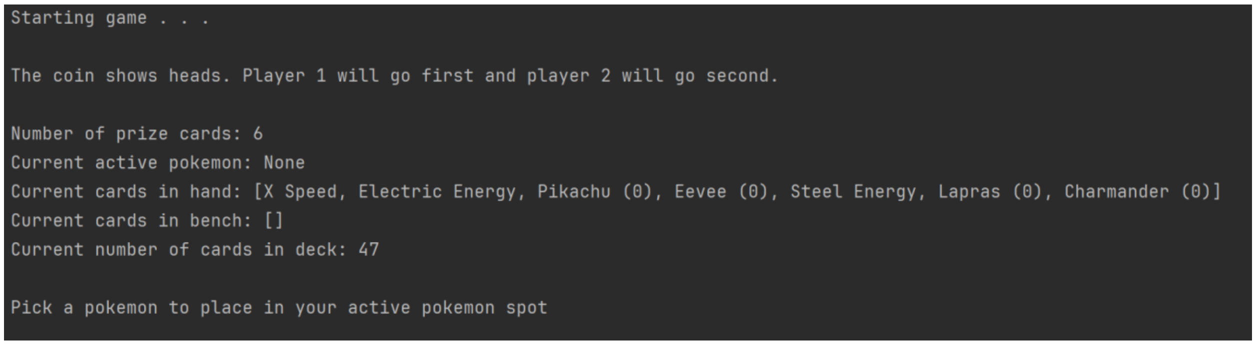
The XSpeed class is a subclass of the Trainer class. It does not hold any information about XSpeed but instead sends the information to the Trainer class. The class has one constructor: the default constructor, which sends all information about the XSpeed card, including the name of the card, the type of card, the type of trainer card and the description of the card. The class also has an overridden method called useAbility. The useAbility method accepts one parameter: The player who used the card. The useAbility method uses the card's special ability, which for XSpeed is "During this turn, the retreat cost of your active pokemon is one less." The method returns nothing.

## Output

The program outputs the state of the game and allows the user to give input.

### Screenshots

Beginning of game



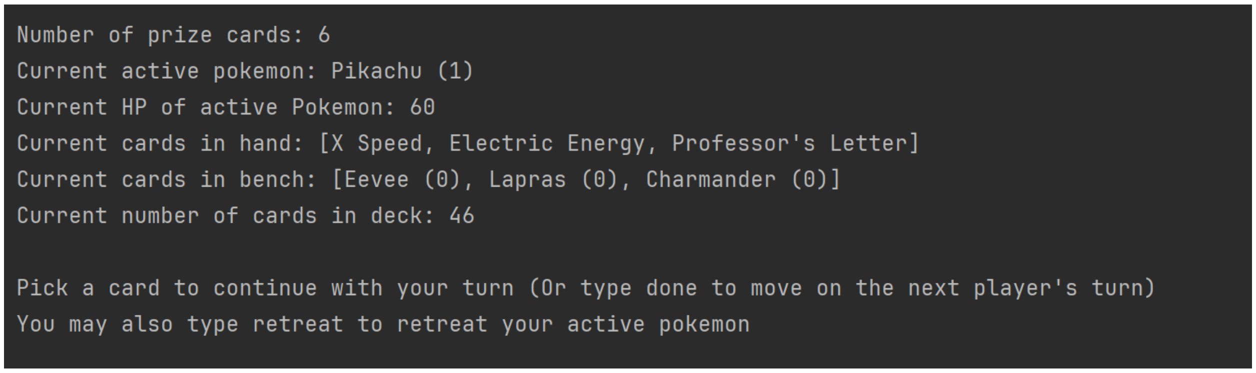
```
Starting game . . .

The coin shows heads. Player 1 will go first and player 2 will go second.

Number of prize cards: 6
Current active pokemon: None
Current cards in hand: [X Speed, Electric Energy, Pikachu (0), Eevee (0), Steel Energy, Lapras (0), Charmander (0)]
Current cards in bench: []
Current number of cards in deck: 47

Pick a pokemon to place in your active pokemon spot
```

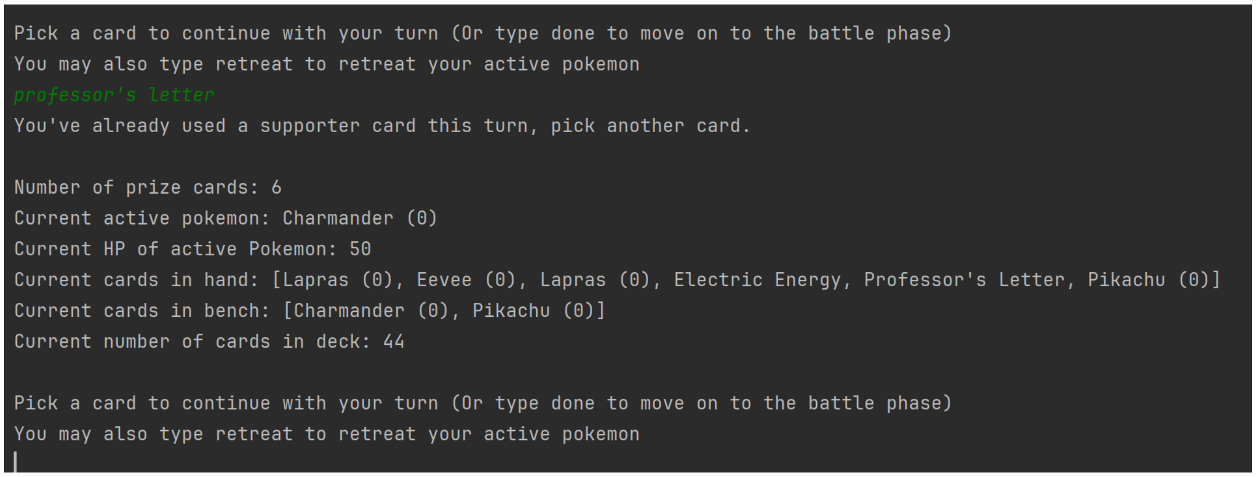
Example of midgame output



```
Number of prize cards: 6
Current active pokemon: Pikachu (1)
Current HP of active Pokemon: 60
Current cards in hand: [X Speed, Electric Energy, Professor's Letter]
Current cards in bench: [Eevee (0), Lapras (0), Charmander (0)]
Current number of cards in deck: 46

Pick a card to continue with your turn (Or type done to move on the next player's turn)
You may also type retreat to retreat your active pokemon
```

Example of supporter card check

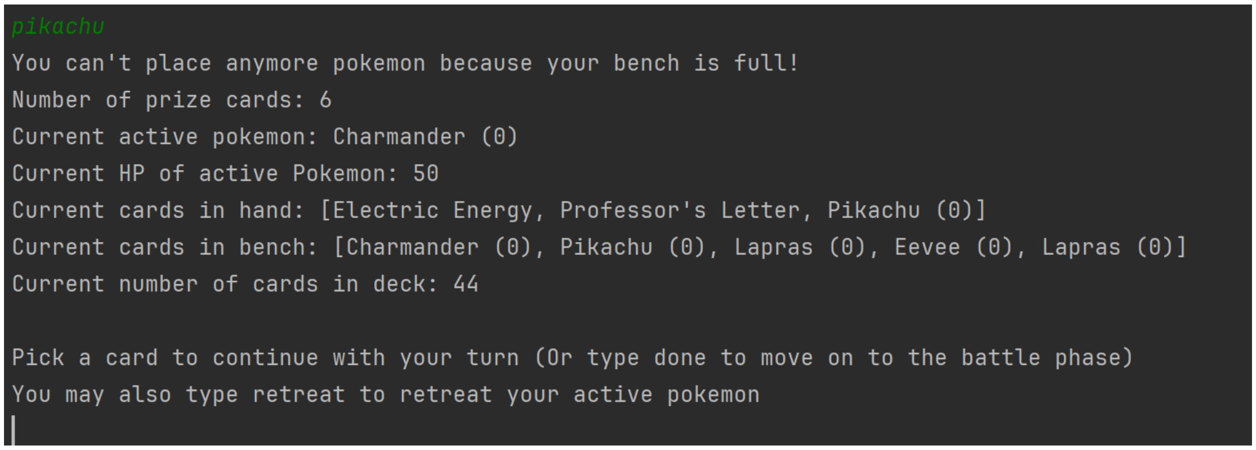


```
Pick a card to continue with your turn (Or type done to move on to the battle phase)
You may also type retreat to retreat your active pokemon
professor's letter
You've already used a supporter card this turn, pick another card.

Number of prize cards: 6
Current active pokemon: Charmander (0)
Current HP of active Pokemon: 50
Current cards in hand: [Lapras (0), Eevee (0), Lapras (0), Electric Energy, Professor's Letter, Pikachu (0)]
Current cards in bench: [Charmander (0), Pikachu (0)]
Current number of cards in deck: 44

Pick a card to continue with your turn (Or type done to move on to the battle phase)
You may also type retreat to retreat your active pokemon
```

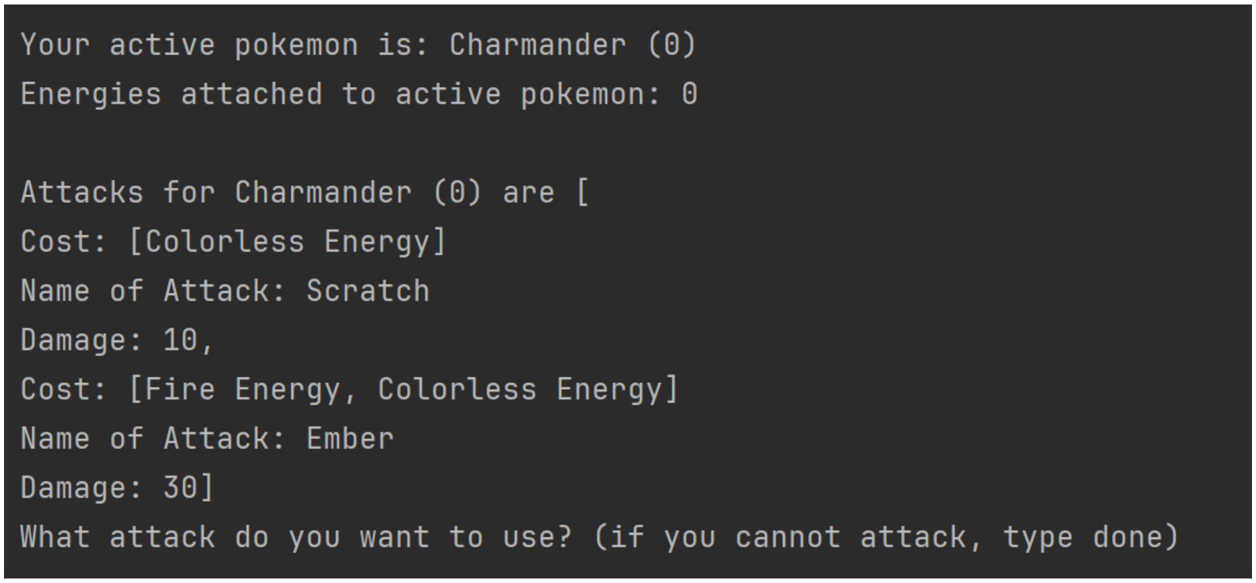
Example of bench size check



```
pikachu
You can't place anymore pokemon because your bench is full!
Number of prize cards: 6
Current active pokemon: Charmander (0)
Current HP of active Pokemon: 50
Current cards in hand: [Electric Energy, Professor's Letter, Pikachu (0)]
Current cards in bench: [Charmander (0), Pikachu (0), Lapras (0), Eevee (0), Lapras (0)]
Current number of cards in deck: 44

Pick a card to continue with your turn (Or type done to move on to the battle phase)
You may also type retreat to retreat your active pokemon
```

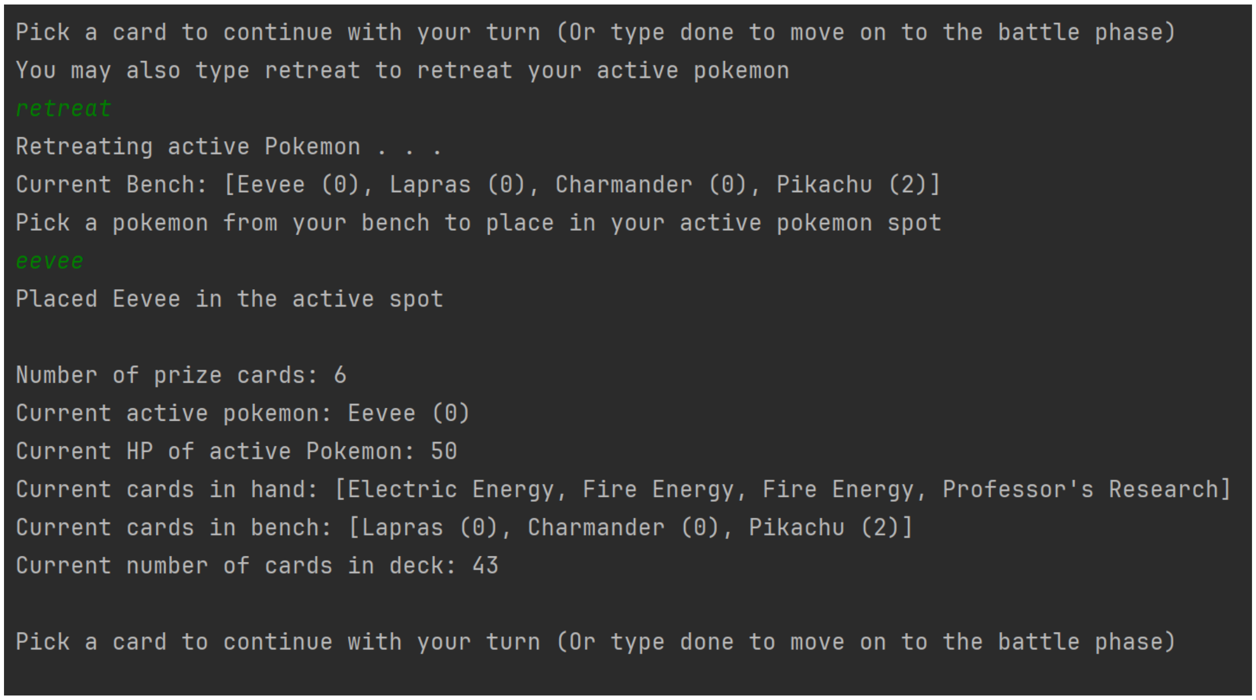
Battle phase



```
Your active pokemon is: Charmander (0)
Energies attached to active pokemon: 0

Attacks for Charmander (0) are [
Cost: [Colorless Energy]
Name of Attack: Scratch
Damage: 10,
Cost: [Fire Energy, Colorless Energy]
Name of Attack: Ember
Damage: 30]
What attack do you want to use? (if you cannot attack, type done)
```

Example of retreating active Pokemon

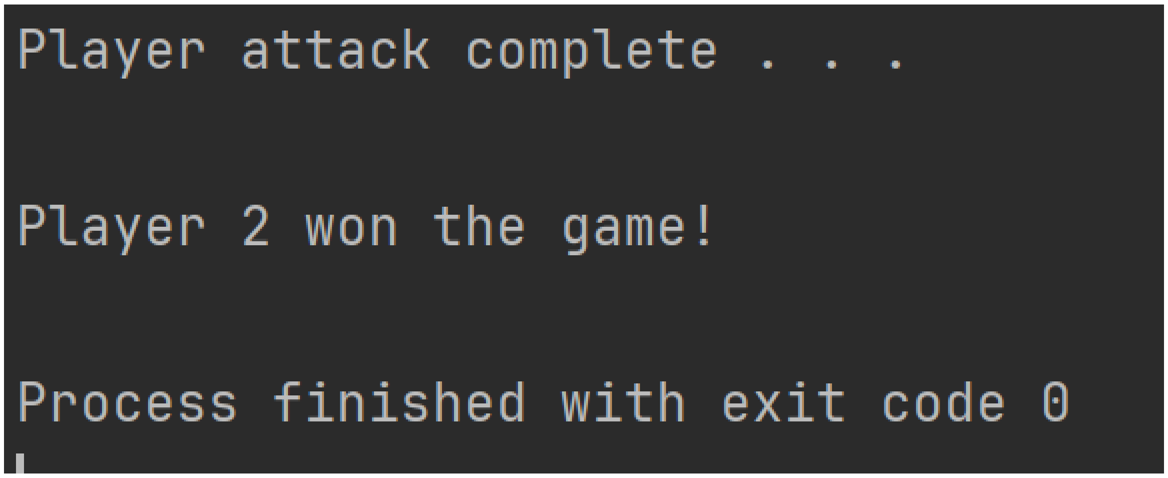


```
Pick a card to continue with your turn (Or type done to move on to the battle phase)
You may also type retreat to retreat your active pokemon
retreat
Retreating active Pokemon . . .
Current Bench: [Eevee (0), Lapras (0), Charmander (0), Pikachu (2)]
Pick a pokemon from your bench to place in your active pokemon spot
eevee
Placed Eevee in the active spot

Number of prize cards: 6
Current active pokemon: Eevee (0)
Current HP of active Pokemon: 50
Current cards in hand: [Electric Energy, Fire Energy, Fire Energy, Professor's Research]
Current cards in bench: [Lapras (0), Charmander (0), Pikachu (2)]
Current number of cards in deck: 43

Pick a card to continue with your turn (Or type done to move on to the battle phase)
```

Example of winning



```
Player attack complete . . .

Player 2 won the game!

Process finished with exit code 0
```