Mia Watts

**Pokémon Card Game Program Documentation / User Manual**

The first part of this document will be descriptions of the methods and classes implemented in the Pokémon Card Game section of the entire project. The second part of this document will be a user manual describing how to use the Pokémon 2-player game. Enjoy!

**Documentation**

**Attackable Interface**

This interface is used for each Pokémon, their classes requiring that they use the methods described within the interface itself. The methods here include the following:

* attackOne() – Takes a parameter that represents a Pokémon target. The parameter is of type PokemonInheritance. The attack method, when implemented within a Pokémon class, can define what the attack does.
* attackTwo() – This method also takes a parameter that represents a Pokémon target and is identical to the previous method minus the fact that it represents the second attack a Pokémon may have.

**Bulbasaur**

Represents Bulbasaur, a grass-type Pokémon implemented in this card game. The main constructor initializes Bulbasaur with an HP of 70 and implements its first attack, Leech Seed. Leech Seed deals 20 damage to a target Pokémon and heals Bulbasaur with the amount of damage dealt if Bulbasaur’s HP is less than 50 (the total cannot exceed 70). The second attack method is not used because the Bulbasaur we implemented for the case of this project only has one attack.



**Card**

The Card class is used to save a String variable that represents a card’s name. This class is the extension class for Trainer cards, Energy cards, Pokémon cards, and Rare Candy cards.

**Chimchar**

Represents Chimchar, a fire-type Pokémon that was added to my repository by Garret Chmielewski. The main constructor initializes Chimchar’s HP to 50 and implements its first attack which deals 10 damage to a target Pokémon. The first attack is called Scratch. The second attack method is not implemented because Chimchar does not have a second attack in this case.

Cartoon of a monkey

Description automatically generated

**Energy**

Implements the Energy cards which have general properties for the case of this card game. The Energy class extends class Card and is used to build decks and hands in the main game. The cards can be of type Fire, Grass, Water, etc., and can change certain game elements. However, for this game and for simplicity, the card simply represents Energy and nothing else.

**Fennekin**

Represents Fennekin, my favorite Pokémon (including all subsequent evolutions)! The main constructor initializes Fennekin with an HP of 70. Two attacks are implemented in the Fennekin class, the first being Live Coal and the second being Rear Kick. Fennekin is a fire-type Pokémon. The first attack deals 10 damage to the target Pokémon and the second attack deals 20 damage.

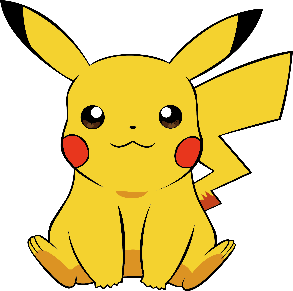


**Nest Ball**

This class implements the Nest Ball Trainer card which searches the entire deck for all Basic Pokémon, selects a random one for the player to add to their bench, and reshuffles the rest of the Pokémon back into the deck. This class and all other implemented Trainer cards follow the Trainer Actions interface which has each Trainer add the playable method to its class.

**Pikachu**

This class represents Pikachu, an electric-type Pokémon. The main constructor sets Pikachu’s HP to 70, adding two attacks, the first being Quick Attack and the second being Electro Ball. Quick Attack deals 10 damage to the target Pokémon, and Electro Ball deals 60 damage to the target Pokémon.



**Player**

The Player class represents a player object which will be used to play the 2-player game. Each player is given a hand, a prize pile, a deck, a discard pile, and a bench, along with an active Pokémon slot that can be used to play whichever card is in the slot. The main constructor initializes these items and gives the player 60 cards in the deck, 7 cards in a hand, and 6 cards in a prize pile. The player wins when their prize pile runs out or when the other person has a bricked deck.

**Pokémon**

This class represents all cards of type Pokémon, extending the Card class.

**Pokémon Card Game**

**Pokémon Inheritance**

The Pokémon Inheritance class implements the HP for Pokémon and all getter and setter methods. The HP is initialized and has getter and setter methods, the attack names are initialized and have getter and setter methods, and the damage values dealt by Pokémon are initialized and have getter and setter methods.

**Professor’s Research**

The Professor’s Research Trainer card states that players must put back and redraw their hand when this card is played and is implemented as such. The current game state is passed as a parameter, so the redrawn hand is drawn properly and placed back into the deck with no issues.

**Rare Candy**

The Rare Candy class is an extension of Card class, being used in the second Monte Carlo assignment along with having the ability to be implemented in the 2-player game. For the case of this assignment, these items will not be playable in the game due to a lack of current Pokémon cards. In the actual card game, Rare Candies can be used to evolve Pokémon. For example, if there is a Pokémon in a player’s hand that can be evolved, they can play a Rare Candy if they have one which can either level up their Pokémon or evolve them immediately.

**Test Pikachu**

The Test Pikachu class was one designed to test certain attacks and methods created within the Pokémon Inheritance and Pokémon card classes. In this test example, Pikachu and Bulbasaur attack each other, with Bulbasaur using Leech Seed and Pikachu using Quick Attack. The game ends when the opponent’s Pokémon has an HP that is either 0 or less.

**Test Pokémon Card Game**

This is the class that tests the Pokémon card game methods implemented within the PokemonCardGame class. Here, the Player objects are created, and they will be used to play the game. The playGame() method is also called in this class which plays the 2-player game. More information on how everything works can be found in the section called “Pokémon Card Game.”

**Trainer**

The Trainer class is used to implement anything associated with the Trainer cards. This is the main class that contains a playable method that isn’t used since the Trainer cards are separately implemented, including more methods that contain getters and setters for certain developments that need the Rare Candy Trainer card. These include returning the damage for Trainer Leon, who can allow a Pokémon to attack another with 30 more damage.

**Trainer Actions**

This is an interface that implements the playable method. Each trainer, including Nest Ball, Professor’s Research, Rare Candy, Leon, and Lillie, must implement this method even if it isn’t used (like for the case of the Rare Candy which is simply used for the second Monte Carlo and nothing else). The playable method takes the current state of the game as a parameter even if it is not used for the case of maintaining the proper game progress. The rest of the Trainer cards each have their own individual implementation of the method, and each card must implement the Trainer Actions interface.

**Trainer Leon**

This Trainer card adds 30 damage to the damage dealt by a Pokémon. For example, for Pikachu’s Electro Ball attack, it originally does 60 damage, but with this Trainer card played along with it, Pikachu can do 90 damage. The method within the Leon class makes sure the current turn is still going, and if it is, it adds 30 damage to a Pokémon’s attack.



**Trainer Lillie**

This Trainer card lets the player draw cards until their hand reaches a certain number of cards. If it’s the first turn, the player can play this card and draw cards until they have 8 in their hand. If it’s not the first turn, the player can play this card and draw cards until they have 6 in their hand. The program checks to make sure their hand size doesn’t exceed these values before drawing cards because the player is not allowed to draw cards if they already have a card amount that is higher than the drawn card amount shown on the Trainer card.



**User Instruction Manual**

**Pokémon Card Game 2-Player Game**

*Written and programmed by: Mia Watts*

