**PokemonGame User Manual**

A Java-based emulation of the popular trading card game, Pokemon.

Cachary Tolentino

CSCI-3327

**Table of Contents**

**Software Description3**

Detailed Description3.1

System Requirements3.2

**Installation Guide4**

**Example Gameplay6**

**Class Overview19**

**Game Overview22**

**Notable Code** **23**

# **Software Description**

**A Java-based emulation of the popular trading card game, Pokémon**.

**Detailed Description**

The PokemonGame program is an emulation of the popular trading card game, Pokémon. It uses multiple classes that represent the type of cards you might find in the actual trading card game, such as Charmander, a Fire Card, and even Bill the trainer card! Users will be able to play the game via an AI or by versing themselves. Users are also given the ability to choose the allocation of their deck, to ensure that everyone has almost full flexibility!

**System Requirements**

* A working device, primarily a desktop or laptop
* An IDE (ex: VSCode, Eclipse, etc…)
* Java JDK (Ver. 17 & up) & JRE (SE 17 & up)

**Disclaimer: code images are not fully captured; refer to raw java file for full code**

# **Installation Guide**

To start playing the game, you need to download a couple of files and place them into a folder. The files are as follows:

**A screenshot of a computer

AI-generated content may be incorrect.**

Once download, ensure that it is placed in a folder that you can access via your choice of Code Editor (such as VSCode). From there, simply open your code editor, open the folder, find the folder name, and open it.

A screenshot of a computer

AI-generated content may be incorrect.

Lastly, you can start playing by simply running the “PlayPokemon.java” file. The terminal will output the following:

A screen shot of a computer

AI-generated content may be incorrect.

Happy playing! 😊

# **Example Gameplay**

Here is an example of how a game might go. Here we can see that the game greets the player. The player is given a choice of continuing or not. Then it as what kind of game mode to play. Afterwards the player gets to choose what coin side they want to bet on.

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Afterwards, it will announce whether the player guessed right or wrong. It will then decide who will be the first player. It then proceeds to ask if the player wants to customize the deck. Here, the player is given the choice of the allocation for each type of card. It then proceeds to create the deck and even automatically fixes any mulligan hands. It allows the opposite player to draw an extra card. Finally, it then asks which Pokémon they want to start with.

**A screenshot of a computer program

AI-generated content may be incorrect.**

After picking a Pokémon, the game automatically draws for the player. Then the player is given a couple of options about what to do. Here we can see how the player chooses to add energy to their Squirtle. The changes are shown in the following images.

**A screenshot of a computer

AI-generated content may be incorrect.**

Here we can see the added energy.

**A screenshot of a computer

AI-generated content may be incorrect.**

And here we can see that adding energy is restricted to once per turn.

**A screenshot of a computer

AI-generated content may be incorrect.**

In addition, during turn one, we cannot attack.

**A screenshot of a computer

AI-generated content may be incorrect.**

In the following result we can see an example of using a trainer card, in this case a ProfessorOak.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Then here we can also see how we can bench Pokémon’s from our hand.

**A screenshot of a computer screen

AI-generated content may be incorrect.**

And then in the following result we can observe the AI trying to perform an attack, although it failed due to not having enough energy on their Pikachu.

**A screenshot of a computer

AI-generated content may be incorrect.**

Here we can see how a successful attack would look. The player is given the option of which abilities to perform.

**A screenshot of a computer program

AI-generated content may be incorrect.**

And then here it shows the recycle trainer card being used and also showing our current benched Pokémons.

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

We can also show that retreating a Pokémon works if they have enough energy to retreat.

**A screenshot of a computer

AI-generated content may be incorrect.**

Here we can also see how some Pokémons, like Pikachu have special effects with their abilities, since we rolled a head at random, our attack did extra damage.

**A screenshot of a computer program

AI-generated content may be incorrect.**

In the result below we can see that upon beating the enemy Pokémon, we are granted a prize pile card.

**A computer screen shot of a black background

AI-generated content may be incorrect.**

In this instance, it shows the AI performing an attack with their Pokémon and showing that the damage multiplier applies given that the enemy Pokémon is weak to their active Pokémon.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Here we can also see the AI performing a retreat, as their Pokémon changed to Pikachu.

**A screenshot of a computer program

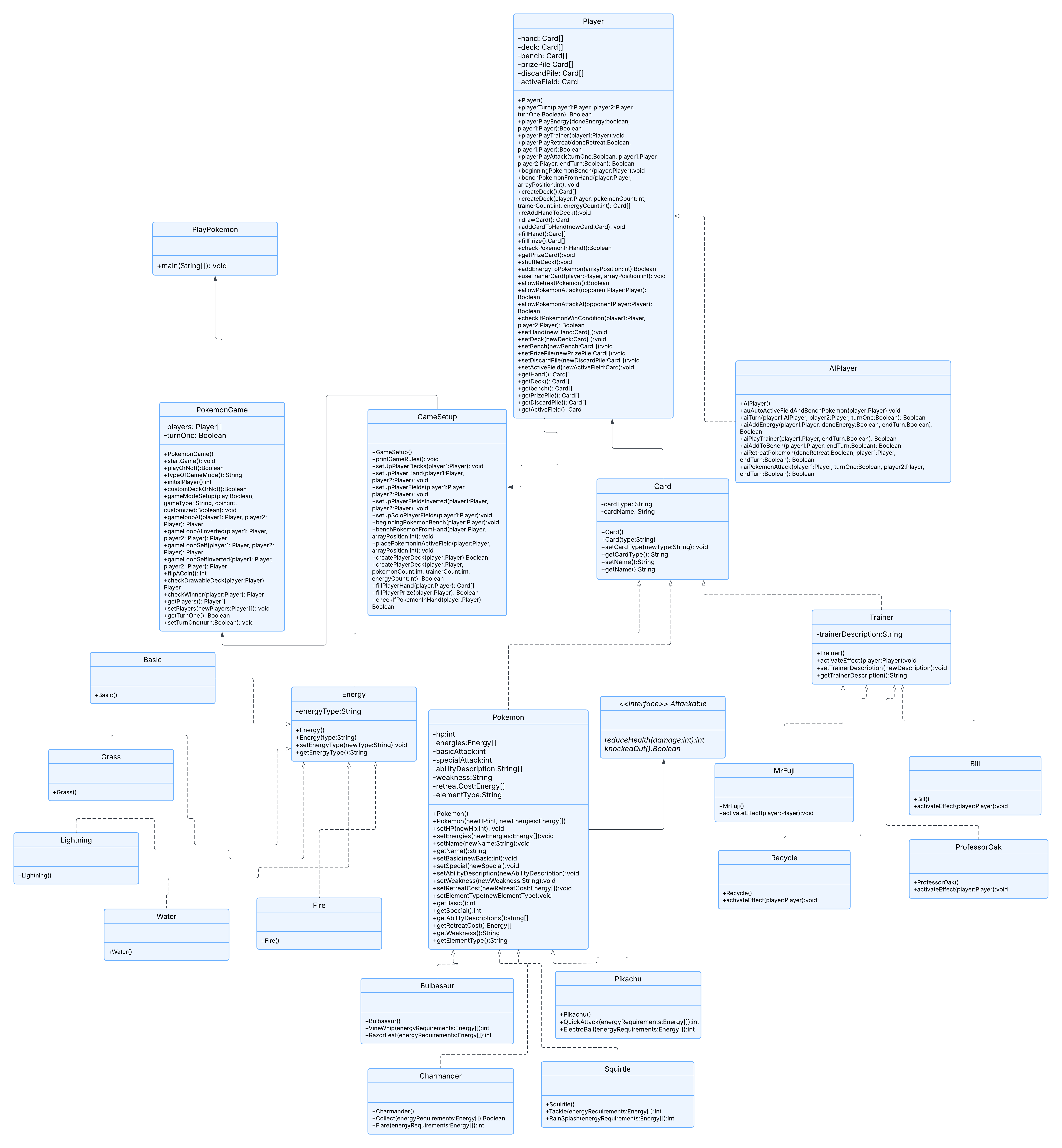
AI-generated content may be incorrect.**

After a couple more rounds and swapping Pokémons, we beat six Pokémons and won the game.

**A screenshot of a computer program

AI-generated content may be incorrect.**

# **Class Overview**



The UML Diagram describes the relationship between all the classes within the program. PlayPokemon is simply the runner/tester class for running the entire program. This is associated with the PokemonGame class. The PokemonGame class is the main class that creates the entire game. Thus, the PlayPokemon class creates a PokemonGame object and calls the startGame() function.

The PokemonGame class is responsible for deciding the type of game loop that the user chooses (AI or Self) as well as checking each turn and which player has won the game. This class is also associated with the GameSetup class. The GameSetup class is responsible for initiating the game’s initial objects such as the player objects, their fields such as their deck, active zone, hand, prize pile and discard pile.

The GameSetup class uses the Player class to create player objects. These player objects would represent each player and the information they would have. The Player class is responsible for performing any action that a player can perform. Some would be drawing a card, activating a trainer card’s effect, and attacking with their active Pokémon. Furthermore, Player has a specified child class, AIPlayer. The AIPlayer is specifically made for an AI game mode. This class is responsible for any actions that an AI can perform, which is primarily describing their turn action and automatically filling their fields.

Each player would have Cards. The Card class is a parent class. It works as a framework for other types of Cards such as a Trainer card, Pokémon card, and Energy card. It primarily defines that each Card object will have a cardType and cardName, this is later useful for finding the type of card object the program is handling.

The types of Cards are mainly Energy, Trainer, and Pokémon. These classes are further subdivided with their children’s classes which define specific types of their respective categories.

The Energy class has five children, Basic, Water, Fire, Grass, and Lightning. These classes simply serve the purpose of being an identifiable energy card that the Pokémon objects may use.

The Trainer class has four children: Bill, MrFuji, ProfessorOak, and Recycle. These classes have their own specific effects that a player can activate during their turn.

Lastly, the Pokémon class has four child classes: Squirtle, Bulbasaur, Pikachu, and Charmander. Each Pokémon class contains specific information that relates to that type of Pokémon. It primarily consists of information such as their HP, Basic Attack, Retreat Cost, and Weakness. Furthermore, each Pokémon has their own specific abilities that cost a certain amount of energy to perform. To retreat the Pokémon, there is also a required energy count.

A single interface was implemented, Attackable. This works as a contract for the Pokémon class. It defines reduceHealth() and knockedOut() functions. It serves as a guide for the Pokémon class to implement which will be used later when a player decides to attack with their Pokémon.

# **Game Overview**

To play the game, two players are required. The user has the option of playing against an AI or against themselves. To do this, PokemonGame initializes two Player objects or one Player and one AIPlayer, depending on the game mode. Afterwards, the game will ask the user which side of the coin they would like to bet on. Depending on the result and the user’s guess, the first turn will belong to the one who guessed the correct side. Thus, inverting the player positions within the class. In the beginning, the player will have the option of customizing the allocation of their deck (i.e. the amount of trainer, Pokémon, and energy cards. It is important to note that the type of cards being made will be randomized despite custom allocation. In the AI game mode, the AI’s fields will be automatically filled depending on their drawn hand. For the player, they are given the choice of which Pokémon they want to start with and the chance to bench any extra Pokémon that they have on hand.

During the first round, the player with the first turn will have a restriction on which they cannot perform any attacks. But once the first turn is over, the restriction is lifted. In the game, the user has multiple options. They can attach an energy card onto their active Pokémon. This is restricted to once per turn. A player can also play an unlimited amount of trainer cards that they have in their hand. Furthermore, a player has a few options regarding their Pokémons. They have the option to bench any extra Pokémon from their hand. To note, the bench has a maximum of six Pokémons allowed. The player can also retreat their active Pokémon, although the Pokémon must have the required amount of energy to retreat attached to them. Lastly, they can make their Pokémon attack the opponent Pokémon. These attacks will only be successful given that Pokémon also has the required energy attached to them. If a player does not have any other option, then they can simply pass their turn.

# **Notable Code**

**Inheritance**

A simple example of inheritance in the program would be the relationship between Player and AIPlayer. In the code below, we can see in the aiTurn() function it is using another function called checkifPokemonWinCondition(). This function is not implemented in the AIPlayer class. Instead, it is found in the Player class. This works because of inheritance, as AIPlayer is a type of Player, it can access this function.

**A computer screen with text

AI-generated content may be incorrect.**

Smore instance of these usage can be found in the other parent child relationships such as Card and its children (Energy, Pokémon, and Trainer). This also includes the children of Energy, Pokémon, and Trainer.

**Interface**

An example of interface usage in the program is through the interface, Attackable. Attackable defines abstract methods: reduceHealth(int damage) and knockedOut(). These functions are meant for any Pokémon to have the ability to take damage and check if they have fallen. In the code below, we can see these implemented and utilized in the Pokémon attack handler.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**AI Behavior**

In the following code, the program has a function for the AI to behave like that of a player. This is utilized in the beginning stages of the game where each player would usually choose their active Pokémon and bench any extra Pokémon they have. This works both as a balancing feature and AI to human like behavior (although Pokémon selection will always be the first Pokémon found).

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Damage Mechanic**

In the code provided below, we can see how the Pokémons are able to damage one another. For this to work, it was necessary to use Java’s reflection package. In particular, it’s getMethod() and invoke() functions. With the getMethod() function we are able to find the method of a particular class given the string name of the method (abilityChosen). From there we simply “invoke” the method found while passing the necessary parameter values as Object type.

(Article: <https://www.geeksforgeeks.org/reflection-in-java/> )

**A screen shot of a computer code

AI-generated content may be incorrect.**

**Pokémon Effect**

In the following code below, we can see some extra implementation for various Pokémon effects. One such effect is Charmander’s collect. Although it does not damage (which is handled by the program correctly as shown above), instead it adds another card into the player’s deck. For Pikachu, the ability QuickAttack gives the player a chance to deal extra damage based on a coin flip.

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Trainer Effect**

In the following code, we can see some implementation of MrFuji and Recycle trainer cards. These are a more complex variance of trainer cards in comparison to Bill or ProfessorOak as they simply add cards to the player deck. For MrFuji, it allows the player to choose a Pokémon in the bench and reshuffle it into the deck to allow the player to free up a spot (if necessary) in their bench. For Recycle, it involves allowing the player to choose a discarded card from the discard pile and returning it to the top of the deck. This is especially helpful if it’s a card crucial to their next turn.

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**