**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**

**PlayPokémon Class** **23**

**PokemonGame Class** **30**

**Player Class** **39**

**AIPlayer Class** **56**

**Abstract Card Class** **59**

**Energy Class61**

**Pokémon Class66**

**Trainer Class73**

**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

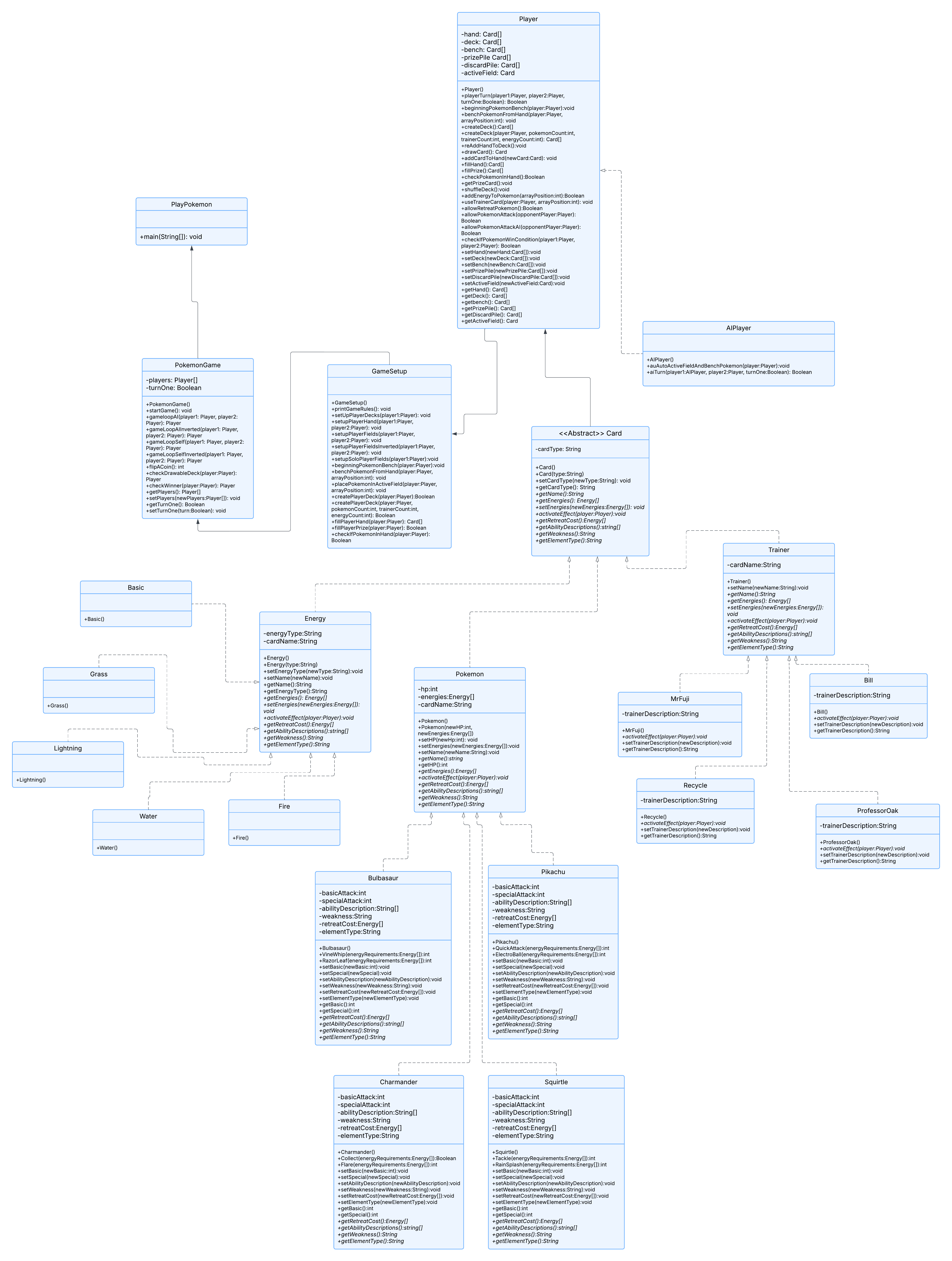
AI-generated content may be incorrect.**

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 an abstract class. It works as a framework for other types of Cards such as a Trainer card, Pokémon card, and Energy card. It still does contain a constructor to ensure that polymorphism would work when identifying the type of card it would be. It contains abstract methods that all other card objects would need to fulfill their role in the game scheme.

The types of Cards are mainly Energy, Trainer, and Pokémon. Thees classes are further subdivided with their child 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.

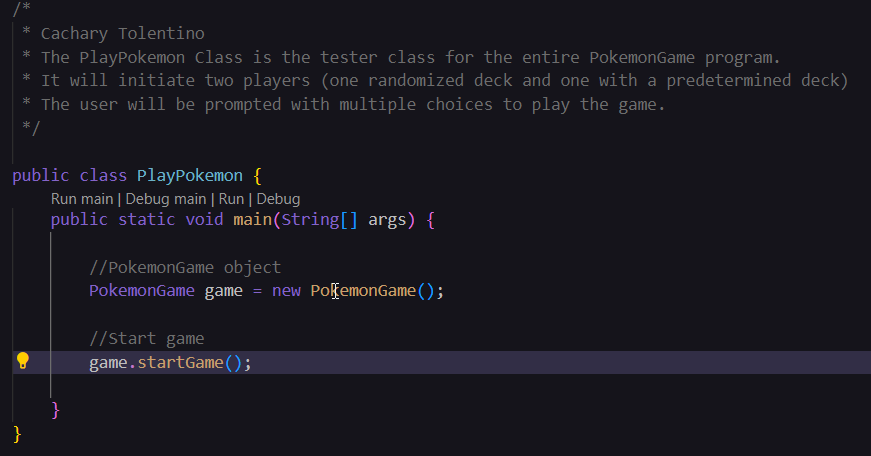
**Game Overview**

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 Pokémon’s. They have the option to bench any extra Pokémon from their hand. To note, the bench has a maximum of six Pokémon’s 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.

**PlayPokemon Class**

The PlayPokemon class contains the main method. It creates a PokemonGame object and calls the startGame() function to initiate the game.

****

**PokemonGame Class**

The PokemonGame class contains two global variables: players and turnOne. The players variable holds the Player objects that would be made during the initialization of the game. These can be either a Player or one Player and one AIPlayer object. The turnOne variable is responsible for keeping track of whether the current turn is the first one or not. This is later changed right after a player/ai turn is over.

The class also contains a constructor that initializes each variable with their own values. Note: even though this initializes the players variable with Player objects, this one of the player objects will be changed to an AIPlayer object if the game mode selected is versus an AI.

**A screenshot of a computer program

AI-generated content may be incorrect.**

In the following code, we can see the initial layout of how the game would greet the player. This gives the player to cancel out of the game at the start and right before selecting the type of game mode. Here we can also see the mechanism for deciding who will be the first player. Once the player chooses a side of the coin, the opposite is automatically assigned to the other player.

A screenshot of a computer program

AI-generated content may be incorrect.

After a game mode is chosen, the following layout is repeated depending on whether the player chose to customize their deck or not. In the code, this shows a customized deck. For a non-customized deck, the player will simply use the same method as the AI (createPlayerDeck). Here a new GameSetup object is made used to initialize all parts of the game for both the player and AI (or player and player). Lastly, the game loop will be called (depending on the results of the coin toss and game mode, this will vary). Once a winner is found, they will be announced, and the program will terminate.

A computer screen shot of a game code

AI-generated content may be incorrect.

In the following code, we can see one of the four variants of the game loop. In total there is gameloopAI, gameLoopAIInverted, gameLoopSelf, and gameLoopSelfInverted. The difference between the inverted and non-inverted versions is simply swapping the chunks of code for either the player or AI (or player) in which they may have the first turn. Here we can also see that the turnOne variable will be changed to false immediately after a turn is performed. Any of the turn functions will return a player if that player is found to be a winner or that the current player has no more cards in their deck, thus losing the game. The checkWinner function simply checks if the current player has an empty prize pile, thus defeating six Pokémon which makes them the winner.

A screenshot of a computer program

AI-generated content may be incorrect.

The following code shows some functions that is used by the class primarily for some of the tasks such as flipping a coin to decide which player will be first, checking if a player still has cards in their deck, and if the player has obtained all their prize cards, thus winning the game.

A screenshot of a computer program

AI-generated content may be incorrect.

In the code below, we can see the setters and getters for the global variable of the class.

A screenshot of a computer program

AI-generated content may be incorrect.

**GameSetup Class**

The GameSetup class contains no global variables. Since it mainly functions as a helper class class for the PokemonGame class it does not need any. Therefore, its constructor does not initialize any variables.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The printGameRules function simply prints all the rules, game setup, a how to play guide, and the win conditions for the game.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The shown function will be used whenever a player has decided to create a custom allocated deck at the start of the game. It simply asks for three inputs and uses those inputs as arguments for creating the deck of the player.

**A screen shot of a computer program

AI-generated content may be incorrect.**

In the following code, the function is used right after each player’s deck has been created. These draw a hand for the players automatically. It also implements a mechanism which prevents any mulligan hand from appearing, it will simply reshuffle the hand into the deck and redraw until atleast one Pokémon is in the current hand.

**A screenshot of a computer program

AI-generated content may be incorrect.**

In the code below, the function will allow the player to choose their starter Pokémon while the AI has its own function to automatically assign their starter and bench any extra Pokémon that they have in hand. The player can also choose to bench any extra Pokémon they want.

**A computer screen shot of text

AI-generated content may be incorrect.**

In the snippet of code, the function defines how the mechanism for benching a Pokémon works. This is primarily used by the player in the beginning portion of the game. However, it is reusable for when the player wants to bench any extra Pokémon they received during the game.

**A screenshot of a computer program

AI-generated content may be incorrect.**

This is a helper function for the function above. This function is the one responsible for moving the Pokémon from the hand and into the bench.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Here we have a function responsible for the action of moving a chosen Pokémon from the hand of a player into the active zone of that same player. It also ensures that the card chosen is of type Pokémon, otherwise it restarts.

**A screen shot of a computer program

AI-generated content may be incorrect.**

Finally, here we have a snippet of most of the setters and getters for the GameSetup class.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Player Class**

For the Player class, we have a lot more global variables. Here we define what a Player can have which are (for the most part an array of Card objects) hand, deck, bench, prizePile, and discardPile. While the activeField is simply a Card object since it will hold the player’s active Pokémon. The class’ constructor also initializes these to their appropriate size (primarily for the beginning phase; other functions will increase or decrease the size depending on usage)

**A screenshot of a computer program

AI-generated content may be incorrect.**

Here we have the main function used in the player class, the playerTurn function. This function defines all the options that a player can perform. It allows the player to continue to perform actions if they are either allowed to or have not ended their turn. The function gives the player the option to attach energy to the active Pokémon, play a trainer card, bench a Pokémon from hand, retreat their active Pokémon, attack with their active Pokémon, or simply end the turn. (Note: the code snippet is too long for a readable image, refer to the java file for the entire code)

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following code snippets are various functions used by the Player class. These fulfill many roles such as creating a player deck (both customized or un-customized), moving cards around (deck to hand), and many more.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Customized variant of creating a player deck.

**A computer screen shot of a program code

AI-generated content may be incorrect.**

Functions are used to move the entire hand (made up of cards) to the deck. This is primarily used by the ProfessorOak card and when a mulligan appears. And a simple draw card method for new turns and any other functions involving needing to draw a card.

**A screenshot of a computer program

AI-generated content may be incorrect.**

More functions are used for setting up the player hands.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following functions are used to create the player prizePile. While the checkPokemonInHand is used for checking if a drawn hand contains a Pokémon, if not then it is deemed as a Mulligan thus having to reshuffle and redraw.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following functions are more utility functions used for manipulating the cards of a player. getPrizeCard is mainly used for whenever a player beats the opponent Pokémon and receives a new prize card.

**A screen shot of a computer code

AI-generated content may be incorrect.**

The shuffleDeck function is also a utility function for reshuffling the current deck. The algorithm used for this function is the Fisher-Yates shuffle algorithm. It was implemented with the help of the following article: <https://www.geeksforgeeks.org/shuffle-a-given-array-using-fisher-yates-shuffle-algorithm/>

**A computer screen with text

AI-generated content may be incorrect.**

The following code is the function that handles adding any chosen energy to the current active Pokémon of the player. It also ensures that the chosen card is an energy card, otherwise it fails to continue the action.

**A screen shot of a computer code

AI-generated content may be incorrect.**

The following function is responsible for activating a chosen trainer card’s effect. It simply finds the desired card and calls on the trainer card’s effect method. This is an example of an interface but more specifically an abstract class. This is due to the Card class being an abstract class and requires its child classes to implement the defined function. Furthermore, since Trainer is the child class while for example Bill is the child of Trainer, this also shows inheritance while performing an override to the parent class’s function.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following code shows the mechanism for how a Pokémon can retreat. It first checks if the current Pokémon contains the required energy. If it does, then it continues to ask which it will want to swap with. The energy card and the Pokémon card itself will be discarded into the discard pile. (Full code in the java file)

**A computer screen shot of text

AI-generated content may be incorrect.**

The following function describes how a Pokémon can perform an attack.

**A screen shot of a computer program

AI-generated content may be incorrect.**

For this code specifically, to call on a specific Pokémon’s ability, the user is given the list of the active Pokémon’s ability. Once an ability is chosen, to precisely call the correct ability, Java’s reflection method is used, primarily the method and invoke function. The method function gets the method with the same name from the correct class that we found through the Card object (since a Pokemon is a Card we are able to do this). Then the invoke function allows the algorithm to call the method with the same ability name.

In here we can also see how the damage effects are handled as well as any weakness multiplier that needs to be applied.

**A screen shot of a computer code

AI-generated content may be incorrect.**

The following function is used before every player turns. The function is responsible for checking the current player has any playable Pokémon. First, first checks the active field. If the Pokémon has fallen, then it automatically replaces it with a Pokémon from the bench. But if there are no Pokémon found, then the current player loses the game.

**A screen shot of a computer program

AI-generated content may be incorrect.**

The following functions are setters and getters for the Player class.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**AIPlayer Class**

The AIPlayer class is a child class of the Player class. Therefore, all of the methods in the Player class are inherited in the AIPlayer class. We will see how some are used. As the AIPlayer class does not have any extra attributes that differ from the Player class, there are no global variables and its constructor is empty.

**A screenshot of a computer

AI-generated content may be incorrect.**

Here we have a small snippet of the aiAutoActiveFieldAndBenchPokemon function. This function is responsible for automatically inserting a Pokémon into the AI’s active field and any extra Pokémon in hand into their bench. This is primarily used in the beginning of the game during game setup.

**A screen shot of a computer program

AI-generated content may be incorrect.**

The following is the beginning portion of the aiTurn function. It contains many similarities as the playerTurn, however, the aiTurn function will decide the plausible actions that an AI can perform through the resources that the AIPlayer has during their turn. It will find every option available and allow the AI to choose randomly from the given list of options. (Full code found in the source code)

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Abstract Card Class**

The abstract Card class acts as a framework for the other types of card that will be discussed later. The Card class was made to be an abstract class specifically due to the need for a constructor and some functions. The card having cardType is necessary to discern between each type of card (Pokemon, Trainer, or Energy). Thus an abstract class was used.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Furthermore, we are able to define abstract methods that mandates its children classes to have. This allows us to perform activateEffect and other functions that may be specific to a specific Trainer card without needing to find the type of Trainer card (i.e. Bill, Recycle, etc…).

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Energy(Basic, Grass, Water, Fire, Lightning) Class**

The Energy class is a child’s class of the Card class. It only contains an energyType and cardName as its global variables. The constructor initializes these as black, except for initializing its cardType to Energy (this helps in finding the type of Card a card object is).

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following are the setters and getters of the Energy class.

**A screenshot of a computer program

AI-generated content may be incorrect.**

These functions specifically are the implementation of the abstract method. These overrides the abstract methods which are denoted by the Override symbol. For Energy, it doesn’t have much use as it will only be necessary for it’s the child classes of the other classes.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following is the implementation of the Fire card, a child class of Energy. There are four others: Water, Grass, Lightning, and Basic. All of these child classes share the same format as one another. All do not contain any global variables while its constructor defines the name of the card.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Pokémon(Bulbasaur, Pikachu, Charmander, Squirtle) Class**

Similarly to the Energy class, the Pokémon class is a child class of the Card class, thus also inheriting its functions. But for the Pokémon class, it defines attributes that each type of Pokémon may have which are their HP, energies (attached energy), and the cardName. Two constructors are made one for an empty variant and another with a given HP and Energy.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following are setters and getters for the Pokémon class.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Here we have an example of a child’s class of Pokémon, the Bulbasaur class. The Pokémon class also has the Pikachu, Charmander, and Squirtle class. All these child classes share a similar format, all of them have a basicAttack, specialAttack, abilityDescription, weakness, retreatCost, and elemetType. The constructors also specify the stats for each Pokémon as they would in their real Card counterpart.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Each type of Pokémon will have their own unique pair of functions that describe the behavior of their abilities. For example, here with Bulbasaur, he has VineWhip and RazorLeaf. These abilities have an energy requirement for them to be fully utilized. If the energy requirement is not met, then an attack fails. Furthermore, some Pokémon have special abilities that come with more effects, for example Charmander’s collect ability does not damage, but instead allows the player to draw an extra card.

**A computer screen shot of text

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

The following are the setters and getters for the Bulbasaur class (other Pokémon have the same implementation).

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Trainer(MrFuji, ProfessorOak, Recycle, Bill) Class**

The Trainer class is the last child class of the Card class. It only contains a cardName variable. The constructor specifices the cardType but not the cardName.

**A screenshot of a computer program

AI-generated content may be incorrect.**

The following are simply the setters and getters for the Trainer class.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

Here is an example of a child class of the Trainer class. Bill only contains a trainerDescription which stores what the ability of the card is. The constructor also specifies it cardType, cardName, and its abilityDescription.

**A computer screen shot of a program code

AI-generated content may be incorrect.**

There are other types of trainer cards which are ProfessorOak, Recycle, and MrFuji. Each of these trainer cards have their own unique effect There are other types of trainer cards which are ProfessorOak, Recycle, and MrFuji. Each of these trainer cards have their own unique. For Bill specifically, he can allow the player that activates the ability to draw an extra two cards. For ProfessorOak, he can withdraw the entire hand and draw seven cards. MrFuji allows the player to choose a benched Pokémon and reshuffle it into the deck. Lastly, the recycle trainer card allows a player to choose a card from the discard pile and place it on the top of the deck.

**A screen shot of a computer program

AI-generated content may be incorrect.**