Probability and Applied Statistics

Project 1

Screenshots and Documentation

Pokemon Card Game

Card

Card is the superclass of all the other subclasses. Every card has the variable name with two getter methods, getName() and getInfo().

Pokemon

Pokemon Class extends card. There are 4 Pokemon - Charmander, Bulbasaur, Squirtle and Pikachu. The Pokemon class has a multitude of instance variables used by its child classes. Methods consist of a constructor, getter and setter for HP. Getter methods for name, type, attackOneName, and attackTwoName. Pokemon also has an ArrayList of type Energy called storage. This tracks if the Pokemon is able to use energy to unleash a strong attack. To manage this there is the receiveEnergy() and useEnergy() methods.

Charmander

Charmander Class extends Pokemon and implements Attackable. Charmander shares characteristics with its parent class Pokemon and implements Attackable in order to be able to attack other Pokemon. Charmander has two attacks, Scratch and Ember. Scratch costs 0 energy to be played and does 10 damage. Ember costs 2 energy to be played and does 30 damage. The attack will fail if there is insufficient energy. Charmander has multiple getter methods to return name, attackOneName, and attackTwoName. There is also the getInfo() method which will print all the cards characteristics if you were unfamiliar with certain attacks and how much they cost.

Charmander cards visible in Active and Bench

A black background with white text

Description automatically generated

Charmander uses Scratch

A screen shot of a computer

Description automatically generated

Charmander uses Ember on the enemy

A black screen with white text

Description automatically generated

Ember attack fails.

A screen shot of a computer

Description automatically generated

Info output

A screenshot of a computer program

Description automatically generated

Squirtle

Squirtle Class extends Pokemon and implements Attackable. Squirtle shares characteristics with its parent class Pokemon and implements Attackable in order to be able to attack other Pokemon. Squirtle has two attacks, Tackle and Rain Splash. Tackle costs 0 energy to be played and does 10 damage. Rain Splash costs 2 energy to be played and does 30 damage. The attack will fail if there is insufficient energy. Squirtle has multiple getter methods to return name, attackOneName, and attackTwoName. There is also the getInfo() method which will print all the cards characteristics if you were unfamiliar with certain attacks and how much they cost.

Bulbasaur

Bulbasaur Class extends Pokemon and implements Attackable. Bulbsaur shares characteristics with its parent class Pokemon and implements Attackable in order to be able to attack other Pokemon. Bulbasaur has two attacks, Tackle and Leech Seed. Tackle costs 0 energy to be played and does 10 damage. Leech Seed costs 2 energy to be played and does 20 damage and also heals for 20 damage. The attack will fail if there is insufficient energy. Bulbasaur has multiple getter methods to return name, attackOneName, and attackTwoName. There is also the getInfo() method which will print all the cards characteristics if you were unfamiliar with certain attacks and how much they cost.

Pikachu

Pikachu Class extends Pokemon and implements Attackable. Pikachu shares characteristics with its parent class Pokemon and implements Attackable in order to be able to attack other Pokemon. Pikachu has two attacks, Quick Attack and Electro Ball. Quick Attack costs 0 energy to be played and does 10 damage. Electro Ball costs 3 energy to be played and does 40 damage. The attack will fail if there is insufficient energy. Pikachu has multiple getter methods to return name, attackOneName, and attackTwoName. There is also the getInfo() method which will print all the cards characteristics if you were unfamiliar with certain attacks and how much they cost.

Trainer

Trainer class extends card. There are 3 trainer cards - Professor's Research, Nest Ball, and Ace Trainer. The Trainer Class has the instance variable name which has two constructors and a getter method. The main functionality of Trainer is it implements TrainerAction which has the playable method. The playable requires two objects of type Player, one for the card user and another for the opponent. All trainer cards have the playable method and output a different effect respectively.

Nest Ball

Nest Ball is a trainer card that searches the deck for a basic pokemon and adds it to the players bench. It then shuffles the player's deck. The method creates ArrayList of tempPokemon to find all the basic pokemon throughout the deck. Then, using the Random import, picks a random pokemon from tempPokemon to add to bench. Then, the deck is shuffled. Included is a getName() method and getInfo() method.

Nest Card Played

A screen shot of a computer

Description automatically generated

Info Output

A screen shot of a computer

Description automatically generated

Professor’s Research

Professor's Research is a trainer card that discards your current hand and draws 7 random cards from the players deck. Utilizes the discardHand() method and drawHand() method. Included is a getName() method and getInfo() method.

Professor’s Research is played.

A black screen with white text

Description automatically generated

Info Output

A black background with white text

Description automatically generated

Ace Trainer

Ace Trainer is a trainer card that can be played if you have more prize pile cards then your opponent. Both players' hands are shuffled into their decks. Then the player draws 6 cards and your opponent draws 3. Calls the returnHand() method to reshuffle hand into deck. Then the draw() method is played to draw cards into hand. Outputs an error if card doesn't meet play requirements. Included is a getName() method and getInfo() method.

Ace Trainer is played.

A black screen with white text

Description automatically generated

Ace Trainer fails to play.

A black screen with white text

Description automatically generated

Info Output

A black screen with white text

Description automatically generated

Energy

Energy class extends Card and consists of Fire energy. Used to power up pokemon and unlock special attacks. Includes getType(), getName(), and getInfo() methods.

Charmander with 1 stored Energy



Info Output

A screen shot of a computer

Description automatically generated

Player

The Player Class is the main class of the PokemonCardGame. It manages the creation of the deck, hand, prizePile, etc. The player constructor fills the deck with the specificed cards of whichever Pokemon and Trainer card are chosen. The main methods of this class are play and attack which do exactly what the name suggests. attack() takes object of type Player and makes the two players' active pokemon battle one another and also uses the activePokemonStatus() method. play() takes object of type Player and gives the user options and allows them to play their cards in hand. Throughout the class are various helper methods such as drawCard(), drawHand(), reshuffleHand(), etc. These all help certain trainer cards and methods do their specificed action. At the end are print methods which help give a visual of all the variables like deck and hand as well as multiple getter and setter methods. There is also a constructor of Player which takes parameters however this is only used with the brickCalculator() method found in the PokemonCardGame Class.

A players’ pokemon is defeated automatically moving its bench Pokemon to Active and decreasing Prize Pile.

A black background with white text

Description automatically generated

No Pokemon in bench or hand. User discards hand for a new one.

A screenshot of a computer

Description automatically generated

User inputs 9 with a hand of length 8.

A screen shot of a computer

Description automatically generated

PokemonCardGame

PokemonCardGame Class is the main game engine. Creates two Player type objects and a Scanner for userInput. PokemonCardGame class has playerTurn methods for each player which prints the cards for a visual as well as calls play and attack. The preTurn() method is used so players can play cards before attacking so activePile is not empty. The setUpGame() draws the cards from the deck for each player and sets up their hand, prizePile, etc. The runGame() is the main game engine and continuously runs playerTurn until a players' prizePile is depleted. It then calls the checkIfWinner() to output a winner and stop the program.

Additional Methods - printInstructions() outputs instructions for the game on how to play. runNumber() will calculate the probability of having a pokemon if your starting hand. brickCalculator() will calculate the probability that your deck is "bricked" and the hypothetical trainer card of Rare Candy is in your prize pile.

printInstructions() method

A screen shot of a computer screen

Description automatically generated

Visual Player Turn Layout

A screenshot of a computer

Description automatically generated

A winner is declared.

A black background with white text

Description automatically generated

TestPokemonCardGame

Tester class. Creates object of PokemonCardGame and calls the runGame method.

Stats Library

Central Tendencies

Mean, Median, Mode and Standard Deviation of Sample Numbers Set

Accepts userInput of type ArrayList<Integer> and returns double values. Median and Mode call Collections.sort to organize userInput. If two or more modes are found, null will be returned.

A screen shot of a computer

Description automatically generated

Null Output

A black screen with white text

Description automatically generated

Set Operations

Union, Intersection and Complement of 2 Sample Numbers Sets

These methods accept two inputs of ArrayList<Integer> and returns one ArrayList<Integer>. While working on union and intersection I was having trouble removing duplicates from the result array so I proceeded to google if there was a convenient way to remove duplicates I found the .contains method which confirms an element is in an arraylist. Using this I made another arraylist in Union and Intersection for .contains which would get rid of duplicate numbers. This method was also helpful in creating the complement method. By using the opposite thinking, if the arraylist did not contain the element, then it would be added to the new arraylist.

A number on a black background

Description automatically generated

Factorial

Factorial has three methods for different data types. One for BigInteger, one for long, and one for double. BigInteger being the main one used in methods such as permutation and combination. The double and long are limited when a

number grows too large the result will become negative.

A black background with white text

Description automatically generated

A black background with white numbers

Description automatically generated

Permutation & Combination

Both permutation and combination uses BigInteger, its methods, and the main factorial of BigInteger. Then converts and returns a number of type double.

A black background with white text

Description automatically generated

Conditional Probability

Conditional Probability has two methods. One for two independent events. One for two dependent events which assumes the probability of intersection is given and calculates based on any missing value.

Intersection

Intersection has two methods. One for independent and one for dependent. The independent method simply multiplies the probability of the two events. The dependent method assumes the conditional probability of event 2 given event 1 and multiplies by the probability of event 1.

Union

Union has two methods. One for exclusive and one for nonexclusive. The exclusive method simply adds the probabilities. The nonexclusive method assumes probability of intersection is given and subtracts that from the sum of probabilities.

Dependency

Dependency returns whether two events are independent or not. The method assumes the intersection of events is given to calculate conditional probability.

A black screen with white text

Description automatically generated

Binomial Distribution

Binomial Distribution uses combination method and Math.pow to return a result of double. Included methods to calculate Expectance, Variance, and Standard Deviation.

A black screen with white text

Description automatically generated

Geometric Distribution

Geometric Distribution uses Math.pow to return a result of double. Included methods to calculate Expectance, Variance, and Standard Deviation.

A black screen with white text

Description automatically generated

Negative Binomial Distribution

Negative Binomial Distribution uses combination method and Math.pow to return a result of double. Included methods to calculate Expectance, Variance, and Standard Deviation.

A screen shot of a computer

Description automatically generated

Hypergeometric Distribution

Hypergeometric Distribution splits the numerator and denominator to separate variables and uses combination method to return a result of double. Included methods to calculate Expectance, Variance, and Standard Deviation.

A black background with white numbers and symbols

Description automatically generated