Arek Ohanian

Project 1 Big document

Github Link: https://github.com/ArekOhanian/Prob---Stats

Stats Library:

The stats library folder has four java files the stats library file with all of the code and a tester file that executes the code for stats library. There is also the set operations file and tester. The stats library tester will output whatever you fill in as the numbers within the code not through the terminal. It also will do all of the methods all at once if you want just a specific one put two back slashes before the operations that you don’t want to see, and the line outputs that print them to the terminal. I followed the formulas given in class as a guideline for each of the methods.

A screen shot of a computer

Description automatically generated

The String tester takes in strings and does union, intersection, and compliments with days of the week in array lists.

A screen shot of a computer program

Description automatically generated

Monte Carlo Simulations:

1. Birthday Problem

The birthday problem has three files the tester, FindBirhtday, and Person

The tester asks how many people are within the class aka you are choosing the sample size then it asks how many times you want the problem to be run this is how we brute force it.

A screenshot of a computer program

Description automatically generated

The person file is a class that holds a person. A person has a birthday.

Then the Find Birthday file creates the amount of people you have selected and assigns them a random birthday. Then checks if at least one person shares a birthday. This happens the number of times you input until it averages out and gives you the final percentage.

1. Monty Hall Problem

The Monty Hall tester executes the program set by the other files and prints out the result.

The door class contains a door which has a placement and an alignment. The placement is the position of the door in respect to the problem so 0-2. And the alignment is wither it is the winning door or the loosing one. The game file then creates a list of doors and makes it so that only one of them is a good/winning door. Then the program randomly selects one of the doors. One script always switches, and another never does. We run this 10,000 times. Then add up the times each have succeeded then divide by 10,000 or the times run to get the percentage change of succeeding if you switch or stay. A screenshot of a computer program

Description automatically generated

1. TCG monte carlo simulations

Inside of the TCG simulation there are two monte carlo simulations in the player class. The first is the card mulligan chance which is the chance of you getting a forced mulligan at the start of the game with an increasing ammount of pokemon cards in your deck it outputs the number of poekmon in hand and then the change of mulligan with that ammount of pokemon in your deck. It uses the basis used for the pokemon card tcg simulation same goes for the rare candy bricking chance.

A screen shot of a computer

Description automatically generated

The rare candy bricking chance populates the deck with an increasing ammount of rare candies up to 4. Then you draw seven cards then populates the prize pool with seven cards from the deck. Then goes through the prize pool and if it finds a rare candy it counts it. At the end of searching the six cards if the counter is equivalent to the ammount of rare candies in the deck you have bricked. It does this for every amount of rare canides in the deck ten thousand times. Then it takes the times you have bricked and divdes it by 10,000. Then it is multiplied by 100 to get the percentage chance of bricking with the ammount of rare candies in your deck.

A screen shot of a computer

Description automatically generated

Pokemon stadium:

The pokemon stadium folder holds a pokemon class, a pikachu class, a charmander class, a stadium class, and a tester class. What it does it take the pokemon pikachu and charmander give them their base stats from the videogames. And then depending on their speed attack each other in a turn based battle. No moves are selected and it takes their attack minus the opponents defence then subtracts that result to their health this loop runs untill one of the pokemons hp is 0 or lower then it stats if player 1 or player 2 has won. Mostly a practice for the next folder.

A screenshot of a computer

Description automatically generated

Pokémon TCG simulation:

So, this is a relatively big project. It has 8 java files. It has a tester, a player, an overall card class, which is inherited by the Pokémon card class, the trainer card class, and the energy card class. Then there is the game class which is where the game flow is handled. And a move class which is what defines the Pokémon moves. So how the game works is first it initializes the players decks, their benches, and it draws them seven cards. Then if they do not have a Pokémon in hand, it forces a mulligan which shows the hand in the terminal (everything is done through the terminal) then put’s their hand back into the deck. Shuffles it and draws a new set of seven cards. A card is drawn from the top of the deck or index zero of the deck which is then put into the hand. The deck then shifts every card up one. Continuing with the setup phase player 1 or the human player is asked to put one of their Pokémon in hand into the active slot. The cup will put the first Pokémon card it finds in its hand from left to right to put into its active slot. Then the player will be asked if they want to put any more Pokémon cards in hand onto the bench if yes then the player selects the card in hand and the position on the bench to put it on if not it moves on. The CPU will search it’s hand for anymore cards and automatically put them onto the bench.

A screenshot of a computer program

Description automatically generated

Moving onto the main phase. If the players active Pokémon feinted the prior turn, they are asked to put a Pokémon from their bench to the active slot. If it is the CPU, it will find the first Pokémon it can on their bench or more accurately the 0 position on the bench because it always will have one there unless their bench Is empty but if it is they have already lost. The game Will ask the player if they want to skip their main phase, play a card, or retreat their active Pokémon. If they skip their main phase, it moves on. When you choose to play a card, you are asked to select which card in hand you want to play. From there if it is an energy card it asks if you want to play it on your active Pokémon on one on the bench if you select the active it attaches it to the Pokémon via an array list. You can only play an energy card once per turn. If you try to it will tell you “You have already played an energy card per turn” and send you back to asking if you want to play a card. If you select the bench, it will display it and ask for the index of the Pokémon on the bench you want to attach it to it is added to the array list if it is a valid index if not it will ask you again. If your bench is empty, you cannot place an energy on the bench it will tell you that and send you back to playing a card. If a trainer card is selected it will do what the trainer card does depending on its name, it may or may not require additional input from the player. But if it does a similar line of questions are asked depending on the card. The CPU will try to play as many cards as possible it legally can then go to the attack phase it will never retreat. All energies are attached to the active, same goes for any healing. (The CPU was not fun to program especially with all the hand stuff you can do especially with the penny card).

A black screen with white text

Description automatically generated

If it is turn one, it will skip over the attack phase and go to the opponents turn. A turn counter is kept and will display the turn whenever a new one starts. Turn two and onwards the player and CPU can start attacking. For the player it will ask them if they want to attack if yes move onto the attack phase, if not then it ends their turn. The CPU will always try to attack with the first move on their active card. The player will be shown their active Pokémon’s moves and energy costs and asked to select which move to use. Once one is selected and the energy costs are met the damage formula similar to the one in the Pokémon stadium is used. With an added weakness feature where if the attacked Pokémon is weak to the energy type used for the move it takes double damage. (The CPU’s Pokémon are weak to the players). If the attacked players Pokémon is knocked out, you retrieve a prize card. If all prize cards are retrieved that player wins the game and the program stops.

A computer screen with text on it

Description automatically generated

The loop of playing cards and attacking is done until one player wins. The win conditions are the opponent has no more active Pokémon, the opponent’s deck is empty, and they try to draw a card at the start of their turn, or the winner gets all six of their prize cards. Once one of those conditions are met the game states that the player’s name has won and ends.

Winning against the CPU

A screenshot of a computer program

Description automatically generated