Code Documentation for PokemonMonteCarloSimulations Project

By Rachel Hussmann

Overview

The purpose of the PokemonMonteCarloSimulation project was to answer two questions about certain situations that can happen in a Pokémon Trading Card game. The first question was "What is the probability of a mulligan occurring when there is 1, 2, 3, . . ., 59, 60 Pokémon cards in a deck?" The second question was "Given that a player is using only basic cards and stage 2 cards, what is the probability that they would get stuck (bricked) if all of their prize cards were in their prize pile?" The project has three classes: BrickMonteCarlo, MulliganMonteCarlo, and PokemonSimulationsTester, all which work together to answer the questions.

Background

What is a Mulligan?

In Pokémon TCG, a mulligan is when a player picks seven cards to start the game, and they have no Pokémon cards in that hand. They then must show the other player their hand, return the cards to the deck, shuffle and pick a new hand. As compensation, the other player gets to add one additional card to their hand.

What is a brick?

A brick is when the player is stuck without any options to play the game. The game is not technically over, there are just no possible actions for the player to take.

How It Works

MulliganMonteCarlo

The MulliganMonteCarlo class is responsible for answering the first question. It contains two methods:

- runMulliganSimulation
    - Parameter: int numOfTrials – The number of trials the user would like to run

    - Functionality – Runs the number of trials for each amount of Pokemon in the deck
    - Returns: ArrayList<Double> - The list of percentages for each number of Pokémon in the deck
- getMulliganChance
    - Parameters: int numOfPokemon – The number of Pokémon to be added to the deck, int numOfEnergies – The number of energy cards to be added to the deck, int numOfTrials – The number of trials to be run
    - Functionality – Runs the mulligan trials
    - Returns: double – The percentage of mulligans that happened during that set of trials

BrickMonteCarlo

The BrickMonteCarlo class is responsible for answering the second question. It contains two methods:

- runBrickSimulation
    - Parameter: int numOfTrials
    - Functionality – Runs the trials of the brick simulation
    - Returns: ArrayList<Double> - The list of percentages from 1 to 4 rare candies in the deck
- getBrickChange
    - Parameters: int numOfTrainer – The number of trainer (rare candy) cards to be added to the deck, int numOfPokemon – The number of Pokémon cards to be put in the deck, int numOfEnergies – The number of energies to be put in the deck, int numOfTrials – The number of trials to be run
    - Functionality – Creates the deck to be used for the trial and runs the trial
    - Returns: double – The percentage of times that the player was bricked

PokemonSimulationsTester

The PokemonSimulationsTester class contains the main method and runs the program. It contains a tester object of the MulliganMonteCarlo class and the BrickMonteCarlo class. It runs the trials and outputs the percentages to the user.

Output

The program outputs the probabilities calculated for each problem. The output at the top is for the MulliganMonteCarlo simulation and the output at the bottom is for the BrickMonteCarlo simulation.

Screenshots

```
[88.58, 77.34, 68.45, 59.45, 52.400000000000006, 46.52, 39.96, 34.97, 29.909999999999997, 25.61, 21.95, 18.26, 15.920000000000002, 14.610000000000001,
 12.379999999999999, 9.66, 7.88, 6.99, 5.53, 4.99, 3.81, 2.93, 2.34, 2.21, 1.79, 1.39, 1.0999999999999999, 0.9900000000000001, 0.65, 0.45999999999999996, 0.49,
 0.3, 0.25, 0.22999999999999998, 0.16999999999999998, 0.21, 0.06999999999999999, 0.05, 0.04, 0.01, 0.01, 0.01, 0.0, 0.01, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[9.879999999999999, 0.8999999999999999, 0.04, 0.0]

Process finished with exit code 0
```