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

Pokémon TCG simulation:

Winning against the CPU

A screenshot of a computer program

Description automatically generated