Joy Hartsig

**Project II**

**Stats Library – Distributions**

The Stats Library Program has been updated with a new option to access the Distribution Calculator. This calculator outputs the expected mean, variance, and probabilities for each distribution. The distributions calculated are Binomial, Geometric, Hypergeometric, and Poisson. The calculator also calculates the cumulative probabilities as well. Each Distribution states the purpose of the variables being inputted and allows for user input of those variables. Almost every distribution utilizes the getCombo() method to reduce the repetition of code when a factorial or combination needs to be calculated in the formula. Each distribution contains the methods: EqualTo(), LessThan(), LessThanEqual(), GreaterThan(), and GreaterThanEqual(). For some distributions it was easier to use the formula 1-P(X) to determine the probability then to loop through and keep a running sum of every probability. The program heavily relies on the Math utility to do calculations such as: to the power of a variable, the use of *e* in the Poisson distribution, etc. The program was fairly simple as the main goal is to perform calculations using the given formulas with the user inputted variables.

**When to Use What Distribution**

**Binomial**

1. The experiment has n trials
2. There are two possible outcomes: success or failure
3. The trials are independent but identical
4. The probability of successes is the same between trials where 1-p is the probability of a failure

**Geometric**

1. The experiment has independent trials
2. There are two possible outcomes: success or failure
3. The probability of success is defined as p
4. The random variable is the trial where the first success occurs
5. There could be 1 to ∞ number of trials

**Hypergeometric**

1. Contains a sample size of n, the sample is randomly selected without replacement from the population of N.
2. The random variable is the number of successes
3. Not independent trials
4. The probability of success changes with each trial

**Poisson**

1. The experiment is based on the occurrence of an event during a specific interval
2. The probability that the event occurs is the same
3. The events of the experiment are independent

**Distributions Screenshots**

**Binomial Distribution**

**Graphical user interface, application

Description automatically generated**

**Geometric Distribution**

**Graphical user interface, text, application

Description automatically generated**

**Hypergeometric Distribution**

**Graphical user interface, text, application

Description automatically generated**

**Poisson Distribution**

**Graphical user interface, text, application

Description automatically generated**

**Function Plotter, Salter, and Smoother**

The function plotter uses three main classes, the Plotter, the Salter, and the Smoother. The equation I chose to implement is the equation of a line in the form y = mx + b. The class LinearEquation is set with the initial variables of m = .5 and b = 2 for the values of the slope and the x-intercept. The method accepts the value for x and solves the equation for y. There are also two methods to get the slope and x-intercept variables, for later use to display the value of the variables. The Plotter class is used to create a csv file that outputs the x and y values associated with the give equation. This class creates a csv file called “plotter.csv” that loops through the x values from 0 to 20 and records the resulting y value. The values are written to the file with comma separation. The Salter class creates a csv file called “salter.csv” that adds a random integer to all the y values. The random value is determined by the Random() method. The x and resulting y values are written to the file. Lastly, the Smoother class creates a csv called “smoother.csv” that changes the y values based on the average of the sum of the surrounding y values. Two variables are used in the loop as placeholders for the previous and next y value. The values are then average together and added to the specified y value. The values are then written to the csv file.

**Plotting Program Screenshots and Resulting Graphs**

**Plotter Graph**

**Chart, line chart, scatter chart

Description automatically generated**

**Salter Graph**

**Chart, line chart

Description automatically generated**

**Smoother Graph**

**Chart, line chart, scatter chart

Description automatically generated**

**Poker Hand Tester**

The Poker Hand Tester will run 10,000 hands from a shuffled deck, and calculate the probability of getting a pair, getting three of a kind, getting two pairs, getting a straight, getting a full house, getting flush, and getting four of a kind.

The Card class initializes the card data type with suit and number attributes. There are two setters and two getters for the suit and number related to the card. A custom comparator class was created to be able to sort the cards from lowest number to highest number. The myComparator class implements a comparator of the card data type. After the implementation of this class the hand can be sorted through the collections sort method.

The Deck class initializes an ArrayList to hold the 52 possible cards in a deck, with the Ace equal to 1, Jack equal to 11, Queen equal to 12, and the King equal to 13. A loop is needed to create every card with every type of suit for a total of 52 card combinations. The deck is then shuffled. The drawCard class takes the first item out of the ArrayList, which is the same as drawing one card off the top of the deck.

The HandEvaluator class tests each of the seven possible hands for poker. The method getHand() creates a new deck and uses the drawCard method from the Deck class to draw five cards and set them to a new ArrayList called hand. The collections sort method is then used to sort the hand from lowest to highest number. Each of the seven poker testers have a return type of boolean and returns whether the test is true or false. The testPair() method loops through the five-card hand to determine if there are two and only two matching cards. The testThreeKind() method loops through the five-card hand to determine if there are three matching numbers (excluding a three of a kind and pair which counts as a full house). The testTwoPair() loops through the five-card hand to determine if there are two sets of pairs in the hand. The testStraight() method loops through the entire hand and tests to see if starting from the first card (the lowest number) the numbers increment by one for each card after. The testFullHouse() method loops through the five-card hand to determine if there is a set of one pair and a set of three of a kind in the same hand. The testFlush() method loops through the five-card hand to determine if all the cards in the hand are the same suit. Lastly, the testFourKind() method loops through the five-card hand to determine if four cards are the same number.

The main class initializes count variables for each test to keep a running sum of the instances of each test. A loop is used to test a hand through each of the seven tests 10,000 times. If the test returns true, then the corresponding count is incremented for that test. Each test is run for the same hand for 10,000 different hands. The program then displays the probability of each test by passing the count into a getProbability method to determine the percentage by dividing the count by 10,000 and then multiplying it by 100. The program also displays the expected probability percentage for reference purposes. The outputs were all close to the expected values.

**Poker Hand Test for Correctness**

**A picture containing logo

Description automatically generated**

**A picture containing text

Description automatically generated**

**A picture containing text

Description automatically generated**

**A picture containing text

Description automatically generated**

**A picture containing text

Description automatically generated**

**A picture containing graphical user interface

Description automatically generated**

**A picture containing chart

Description automatically generated**

**Poker Tester Output**

**Text

Description automatically generated**

Probabilities from <https://www.vcalc.com/wiki/MichaelBartmess/Poker+Calculator+%28five+card%29>