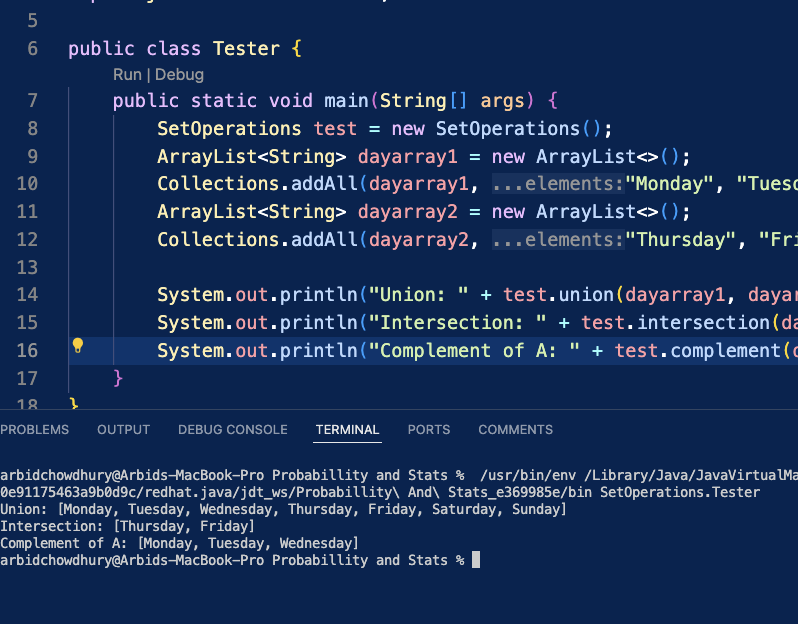
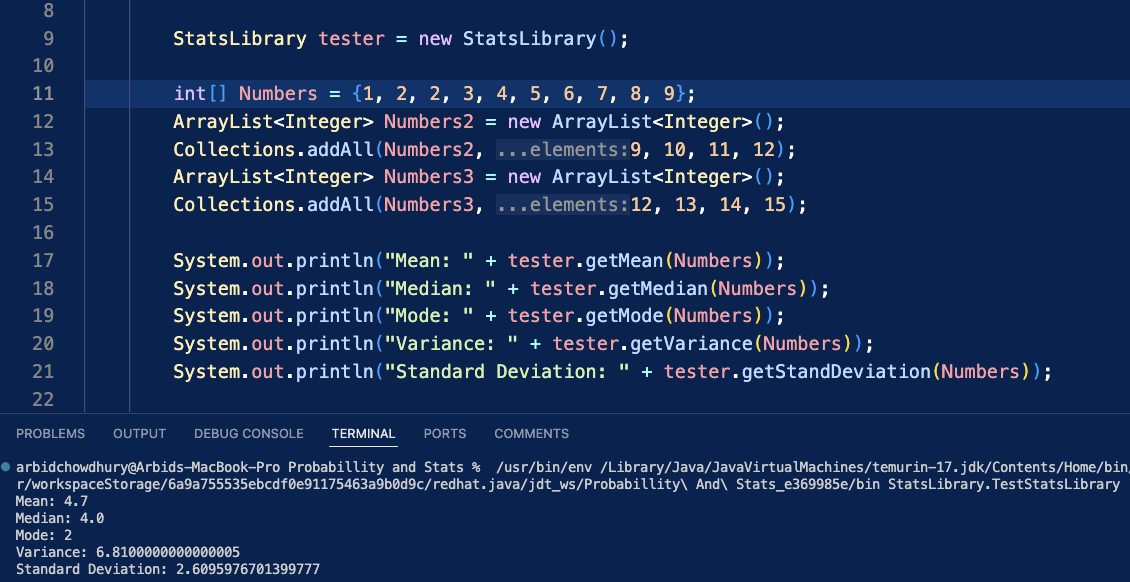
Set Operations

The Set Operations folder consists of SetOperations.java and Tester.java to simulate union, intersection, and complement of sets. There is an empty class called set and three methods to determine union, intersection, and complement. The first of three methods in the SetOperations java file is the method to find union between two ArrayLists of String type. By creating the method with parameters of two ArrayLists, called setA and setB. The union method begins with creating an Arraylist, called union, of all setA elements, proceeding is the for each loop that iterates through each day element in setB. The loop checks if the union array doesn't contain day element, if it doesn't then it adds the day element to the union array. The next method is the intersection method which uses the same format but instead creates an empty Arraylist, called intersection, loops for each day element in setA, then adds the day element if setB contains the day element. This method allows for the Arraylist to contain each intersecting day element that's in both sets. The next method is the complement method which uses an Arraylist, called complement, with all setA elements. Then looping for each day element of subsetA, if the complement Arraylist contains that day, it removes it to generate subsetA’s complement. All the values of subsetA and its intersection with setA are not in the new complement Arraylist. The output for all three methods:



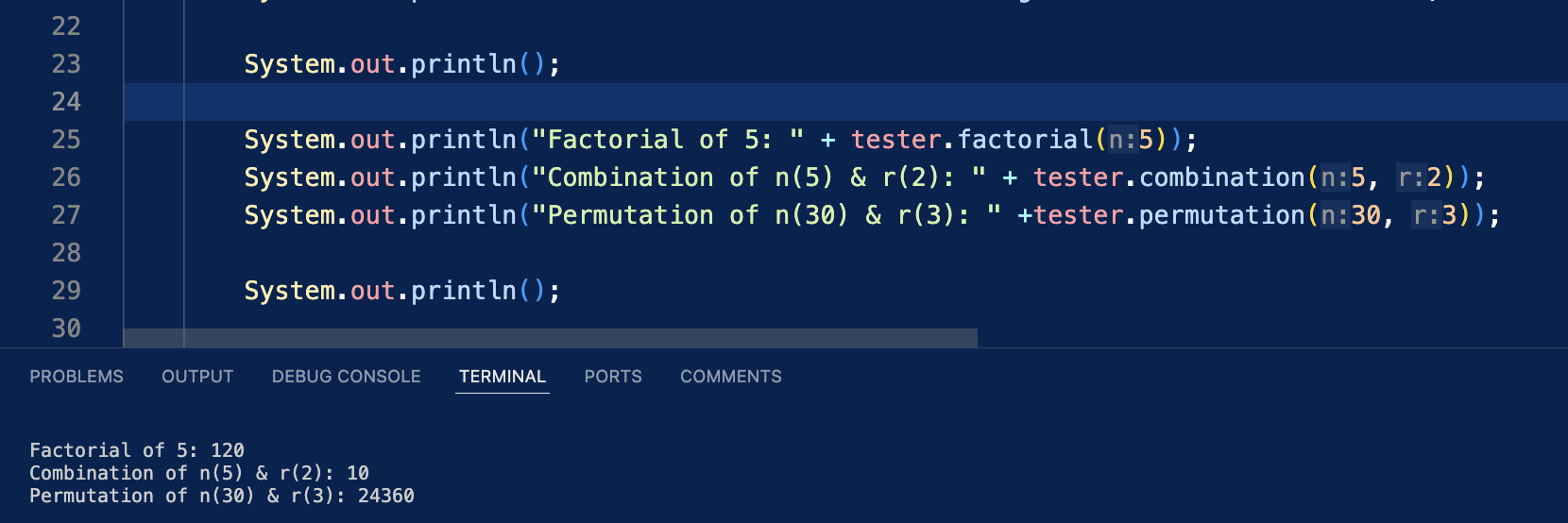
Stats Library – Mean/Median/Mode/Variance/Standard Deviation

The Stats Library folder contains multiple programmed mathematical equations, learned throughout the course of Probability and Statistics, and TestStatsLibrary java file. The first set of equations are mean, median, mode, variance, and standard deviation. The Mean method initializes the total to 0 and then for each number in the userInputNumbers array, the total adds that number to itself and in the end the total is divided by the array length to get the casted double mean result. The Median method begins with using Arrays sort to sort userInputNumbers from least to greatest and initializing the double med (median). If the length of the userInputNumbers array is divisible by 2 (even), then med is the array’s two middle values added together and divided by 2. If the array is odd, then the med is the middle value of the array’s length. The Mode method, which doesn't account for multimodal array inputs, begins with initializing the mode, to first value of the userInputNumbers array, and the duplicates count to 0. Looping through the array and initializing the count of most duplicates, another loop checks through the array to compare both loops’ values to each other each iteration. If there is a match, then the count increases within the second loop and after that there is a check for most duplicates. Outside of the second loop but in the first loop, if the count from the second loop is greater than the duplicates then the duplicates equal count and the mode equals that iteration’s mode which signifies that a higher mode was found. Following that method is the Variance method, which first gets the mean of array, using the Mean method, and mean total to 0. For each number in userInputNumbers array, the mean total equals the addition up each number, subtracted by the mean and squared, to acquire the mean total. The variance is then casted as a double and calculated by having the mean total divided by the array length. The Standard deviation simply square roots the variance from the Variance method to get the standard deviation of the userInputNumbers array.



Stats Library – Factorial/Combination/Permutation

Stats Library also has the Factorial method with int n parameter, using BigInteger for handling bigger factorials. The method starts with initializing BigInteger factorial to BigInteger 1 and then loops from 2 to int n, multiplying the BigInteger value of each iteration i. Starting from 1 doesn't change factorial so 2 is used for less calculation required by the method. This Factorial method is a helper method to Combination method which has parameters of int n and r. Creating new BigInteger combo to equal the combination formula of n factorial divided by r factorial mutiplied by n subtracted by r factorial. Permutation method uses Factorial helper method as well and creates BigInteger permut to equal the permutation formula of n factorial divided by n subtracted by r factorial.



Stats Library – Independence & Dependence

Stats Library also has methods that check if ArrayLists have independent/dependent intersection, exclusive/not exclusive union, and to check if they are independent/dependent of each other. The Independent intersection method loops to check every int in A ArrayList and if B contains any of those ints while looping, then it returns false to show they are dependent and not independent. The Dependent intersection method returns the boolean opposite and for Exclusive Union method, the Not Exclusive method returns the boolean opposite for it. The Exclusive Union method utilizes the independent intersection method because both methods are checking for similar elements in the ArrayLists. There are also methods to determine Independence and Dependence of probabilities A and B. First there are two methods for calculating the probability of A and B, they both take int A/B and divide by the total Sample Space to get their respective probabilities. Following that, there is the checkProbAandB method to calculate the union of A/B’s probability which follows the same logic. The checkDependency method uses checkProbAandB method to see if the union probability is equal to the probability of A multiplied by B’s probability, the checkIndependnece method returning the boolean opposite.



Stats Library – Binomial & Geometric Distribution

Stats Library also has methods for Binomial, Geometric, Hypergeometric, and Negative Binomial distributions. Binomial method uses Big Decimal due to BigInteger not being able to be multiplied by a double that's required in the binomial equation, making it very accurate also. First the method creates a new BigDecimal using Combination method, from earlier part of Stats Library, of n and y multiplying the BigDecimal value of p and y. Next another BigDecimal is made using value of for q to the power of n subtracted by y, then both these BigDecimals are multiplied and rounded to 2 decimal places using BigDecimal setScale. The Geometric method simply creates a double of q, to power of y subtracted by 1, multiplied by p but has error checking implemented. It returns null if y is greater than 0 and does the same for p and q, assuring that both are in between 0 and 1 range. It checks if p and q’s sum is equal to 1 which assures proper probability percentages in the input. The Hypergeometric method is similar in which it has the same error checking but for the Hypergeometric formula but uses BigInteger, BigDecimal not needed because there are no doubles multiplying the BigInteger in the equation. The formula’s numerator and denominator are made into BigIntegers geoA and geoB using the Combination method, then dividing the numerator by the denominator to get the answer. The error cases check if y is less than r or n, r/y/N/n are all less than 0, and n or r are less than N. This ensures that the formula is calculated properly using valid input otherwise the null/error statements will inform user. The last method of Stats Library is Negative Binomial which uses BigDecimal due to the formula having combinations multiplying with doubles. Using the formula, a BigDecimal is made for the combination of y subtracted by 1 and r subtracted by 1. The second BigDecimal is made for the rest of the formula using value of and then both BigDecimals are multiplied and rounded using setScale.

