# 3 Doors

Split into answering questions A and B

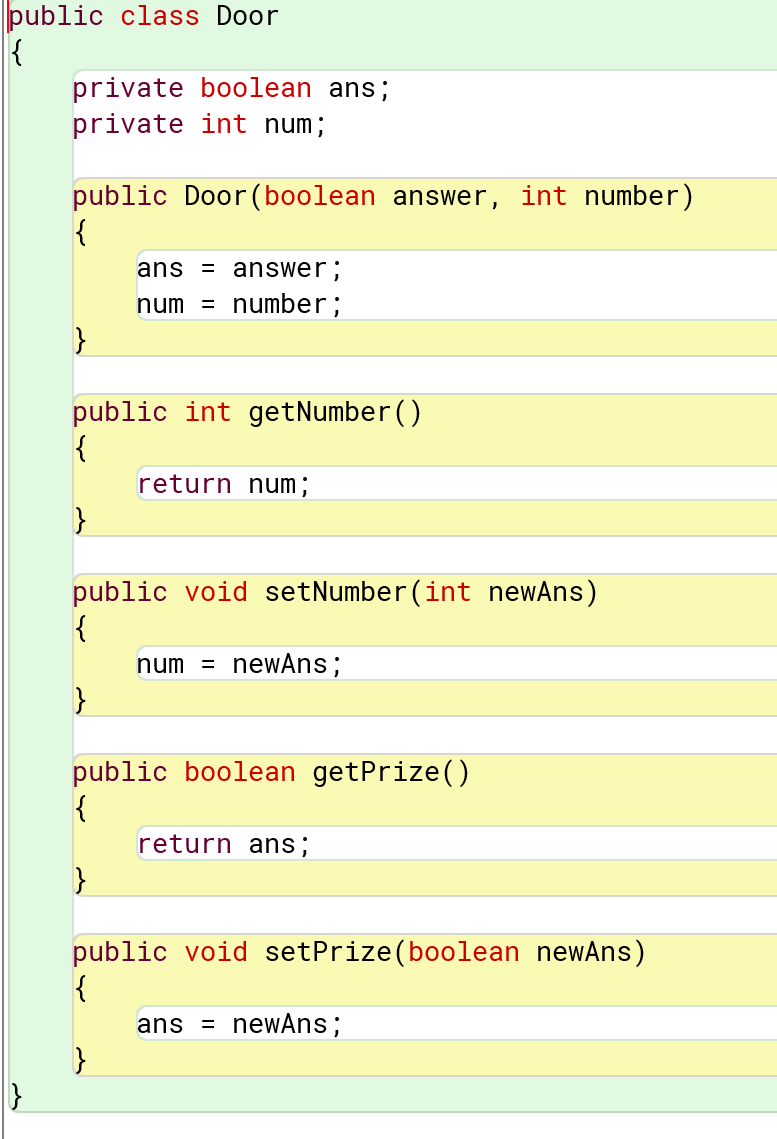
Below shows the contestant picking the same door every time.



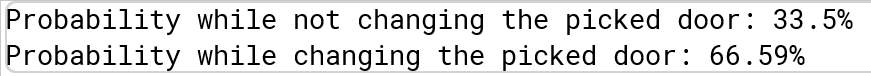
Below shows the contestant picking a different door every time and choosing a second door if it gets the first one wrong.

****

Below is the Door class that holds the constructor that assigns it an answer (whether it holds the prize or not) and a number with setter and getter methods.

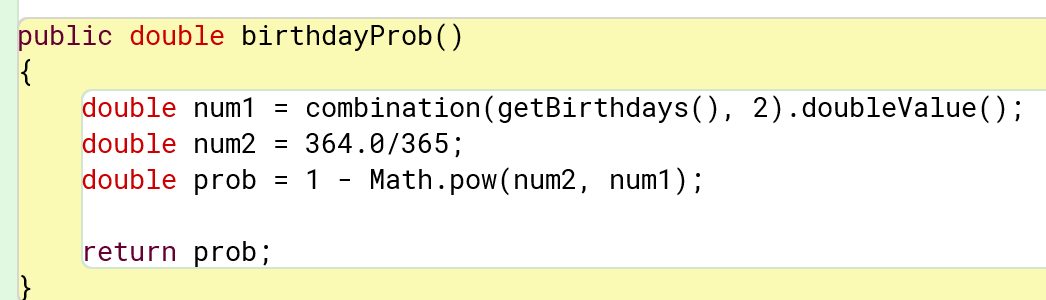


Result:



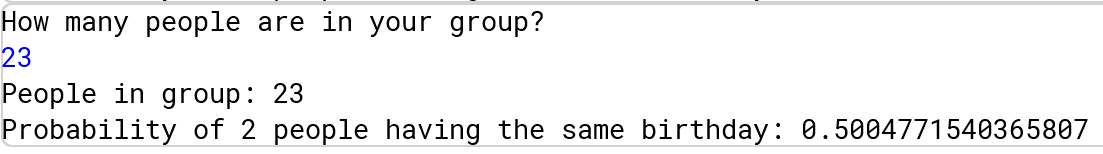
# Birthday

Factorial and combination methods - same one from Stats Library.



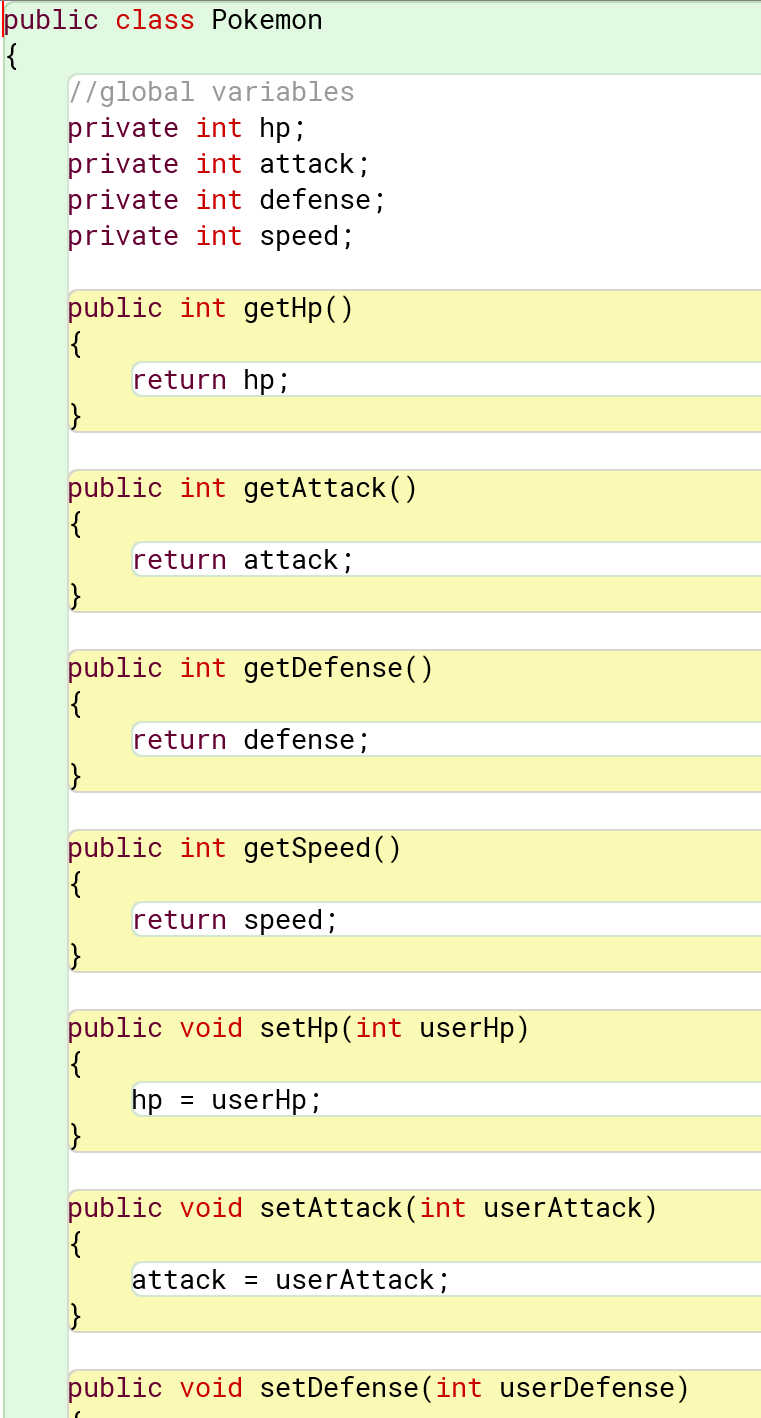
It uses combinations for the formula to find the answer to the Birthday Paradox given a set amount of people in a group.

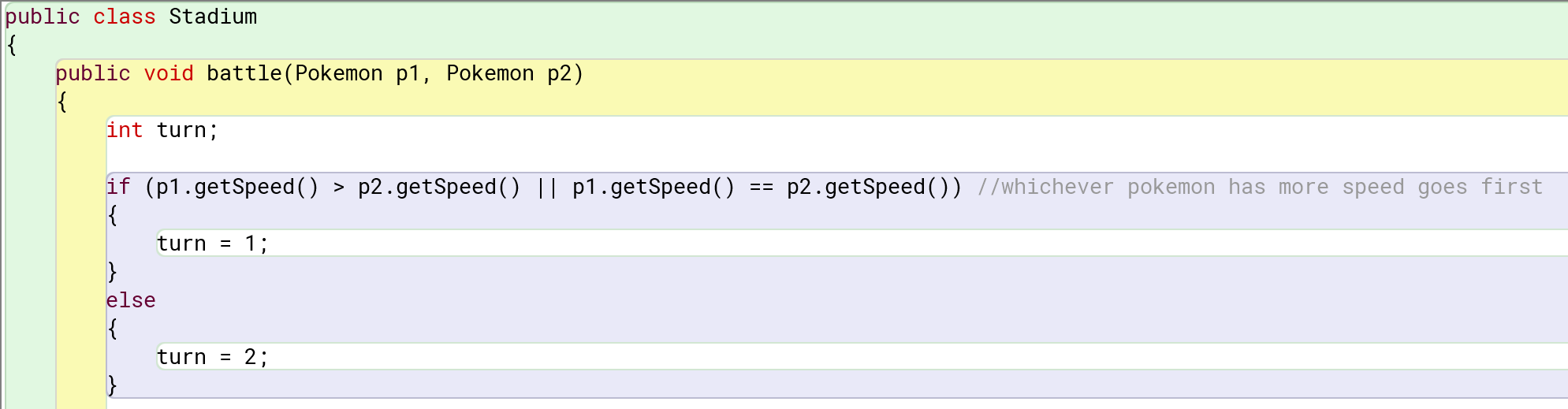
Result:



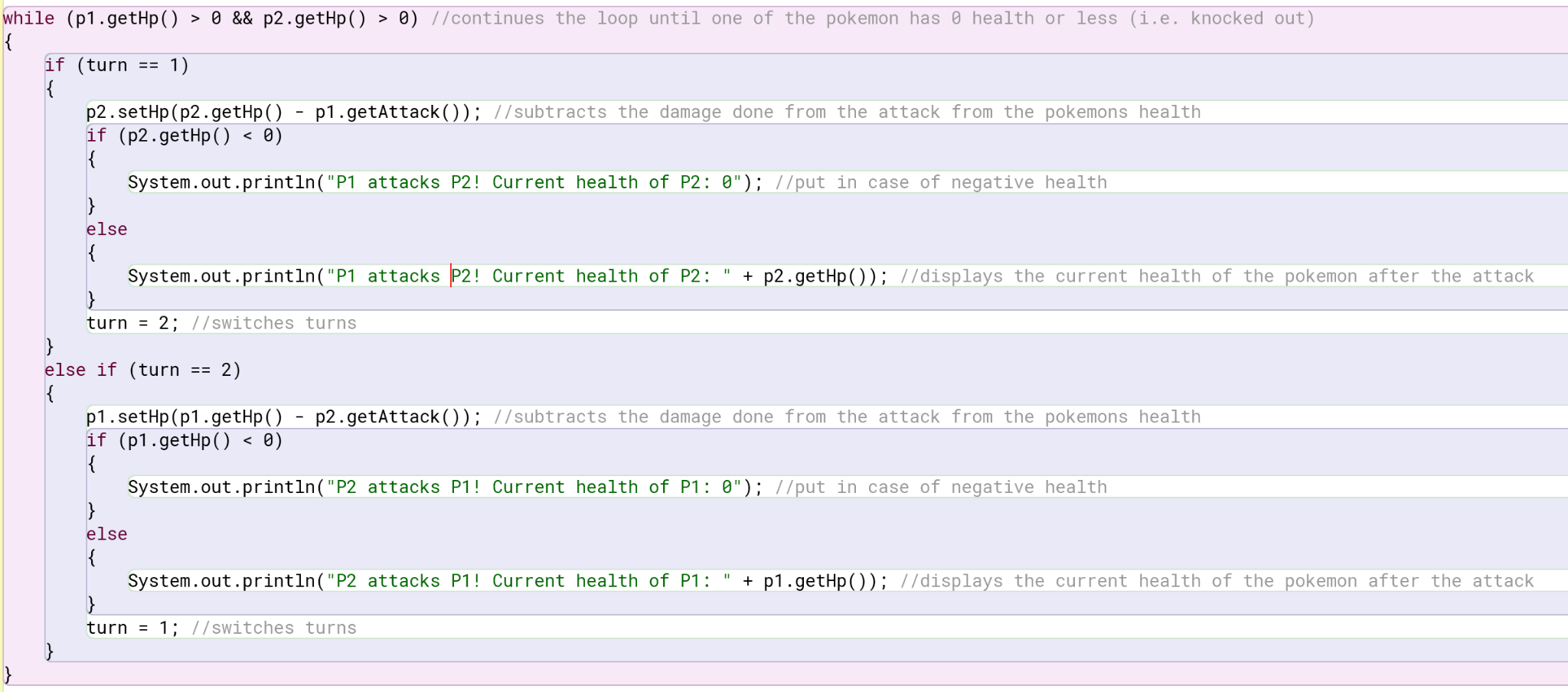
# Pokemon Battler

Below is a list of variables, getter, and setter methods for all Pokemon to inherit and use.

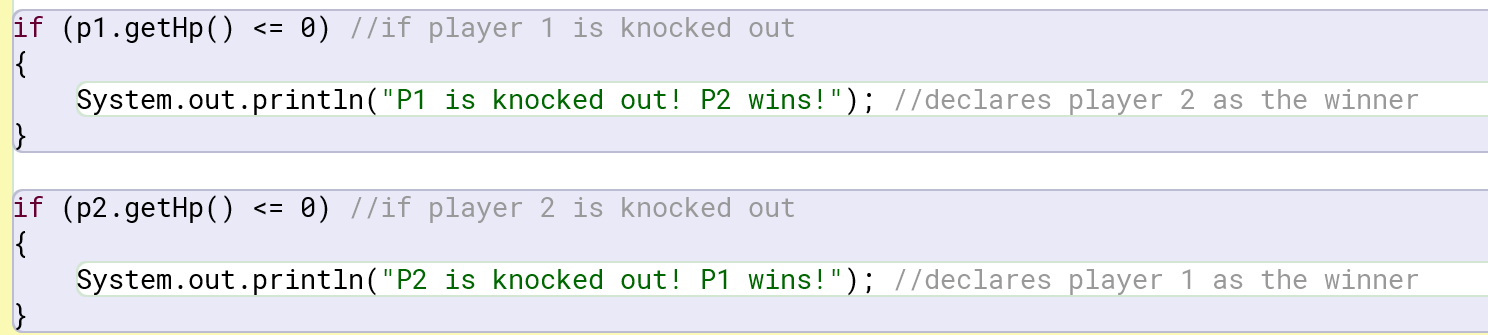


Below is the game determining who goes first by using methods to find which Pokemon has a higher speed stat.

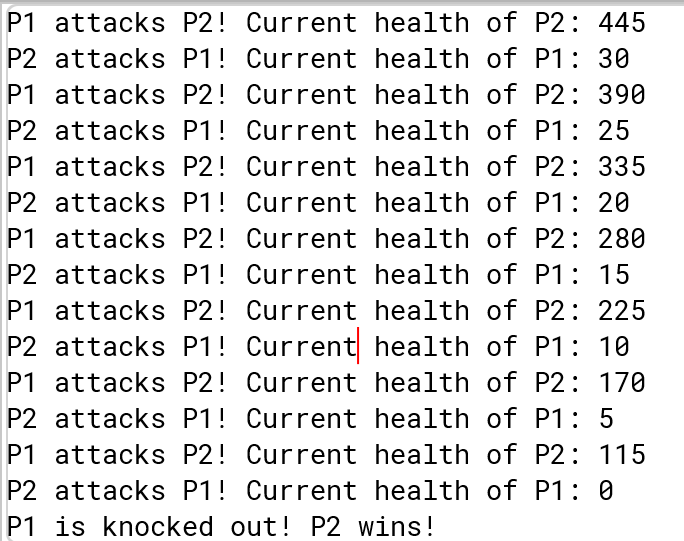
Below is the core gameplay loop. It follows the same pattern of attacking, displaying the health, changing turns, and then checking the health.



Once out of the main loop, it checks which Pokemon was knocked out and declares the winner.



Result:



# 

# Prizes (rare candy)

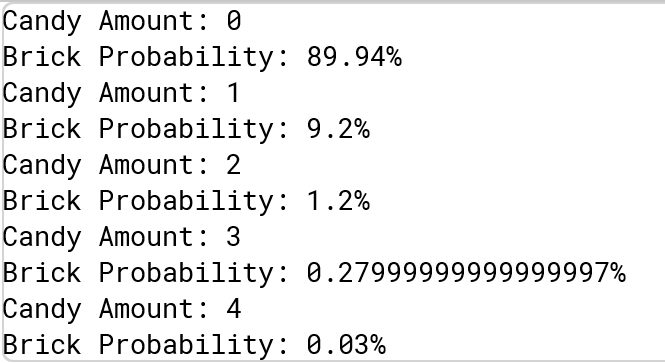
Below shows the deck, current hand, prizes, and candy amount being declared. The deck is filled with a specified number of rare candies and the rest is filled with Pokemon.



Below is the main Monte Carlo experiment. It fills the player's hand with random cards from the deck, checks to make sure there’s at least 1 Pokemon in their hand, fills the prize pile, checks for rare candy in the prize pile, and then compares the number of candies found to the total number of candies in the deck to determine if the player has bricked.



Result:

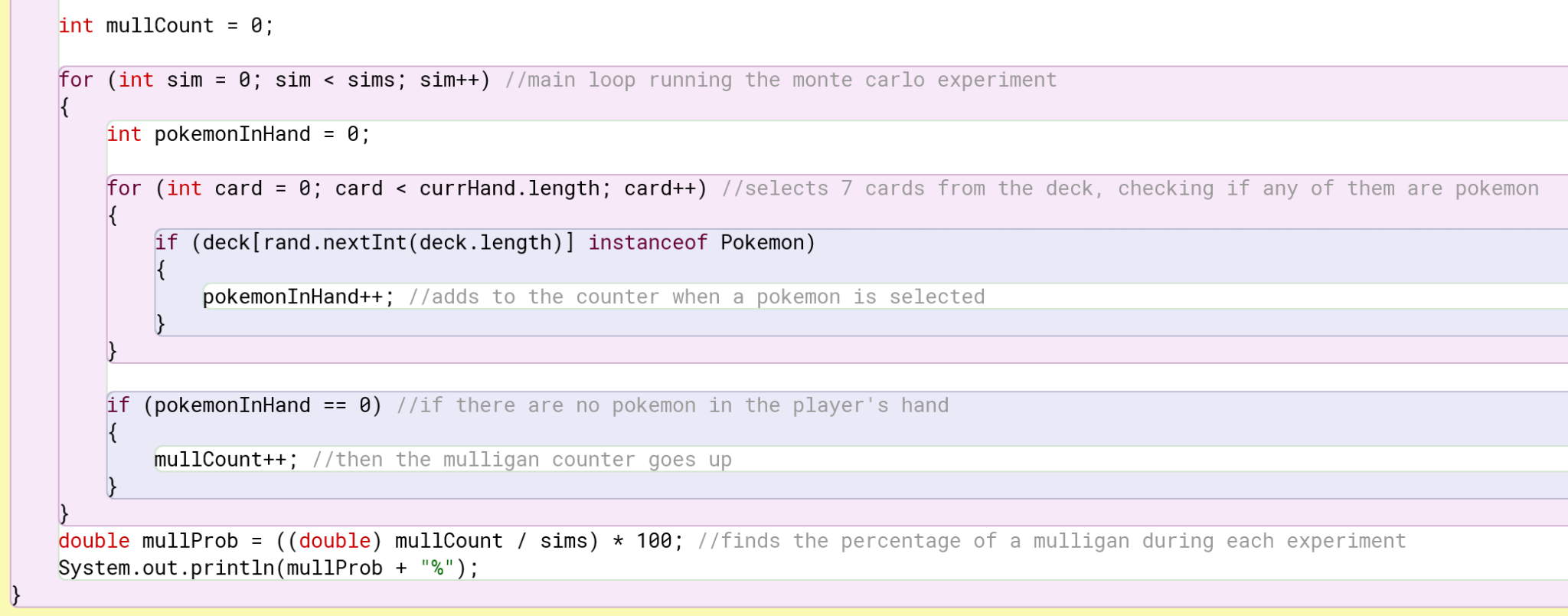


# Pokemon in Opening Hand

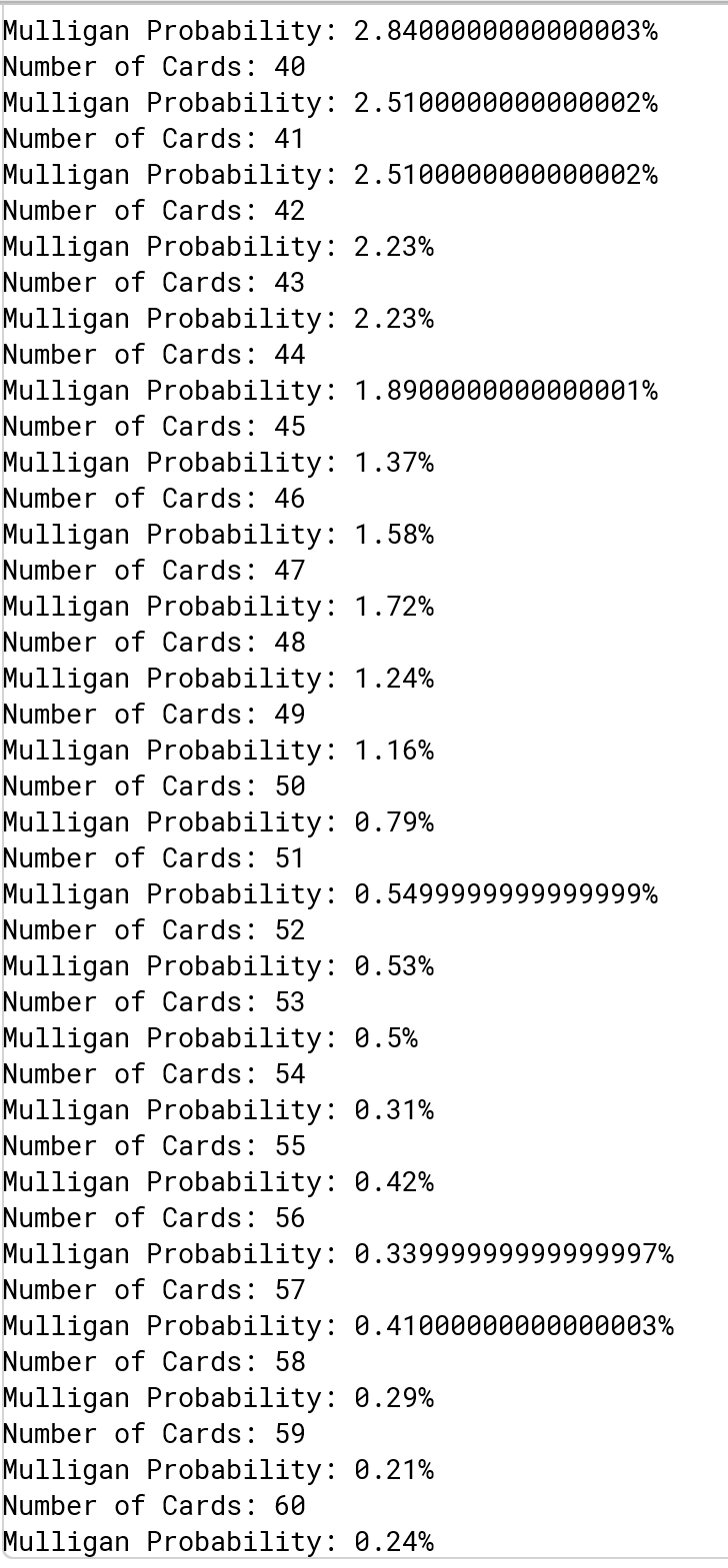
Below shows the deck, current hand, and simulations being declared. Afterward, a deck is filled with a certain number of Pokemon (according to the count) and then fills the rest of the deck with energy to have a full deck.



Below is the main loop running the Monte Carlo experiment. It selects 7 random cards from the deck and checks if any of them are Pokemon. If none of them are, then the mulligan counter goes up. After the thousands of simulations, the mulligan count is compared to the number of simulations to find the percent chance of running into a mulligan with a certain number of Pokemon in the deck.

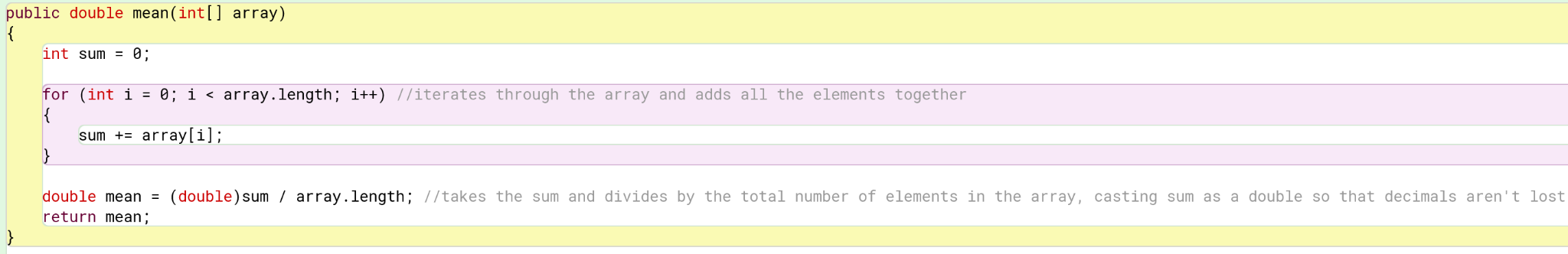


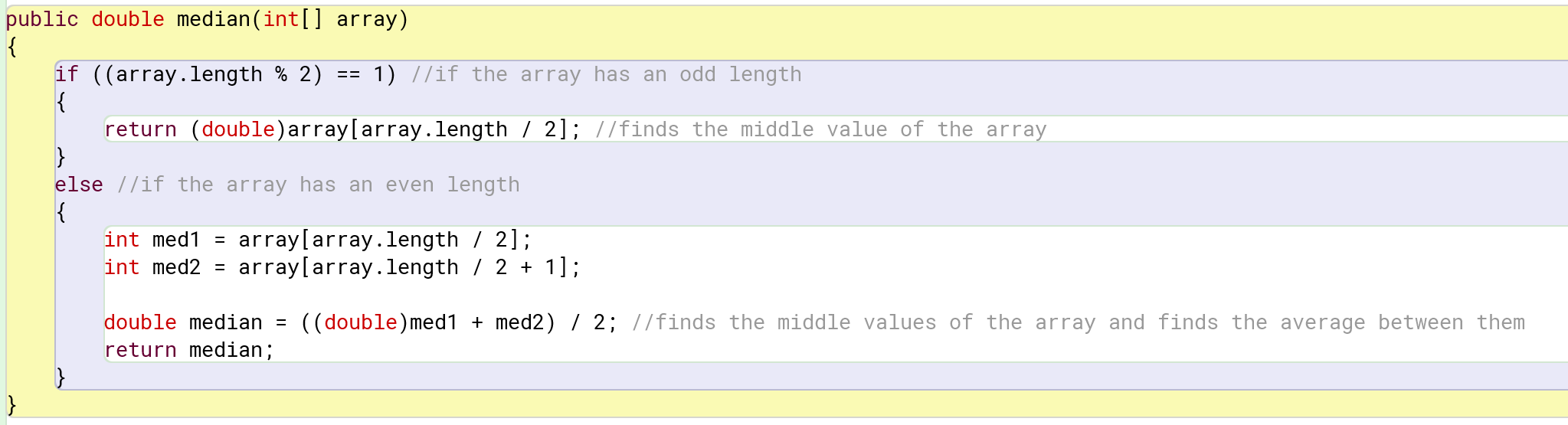
Result:

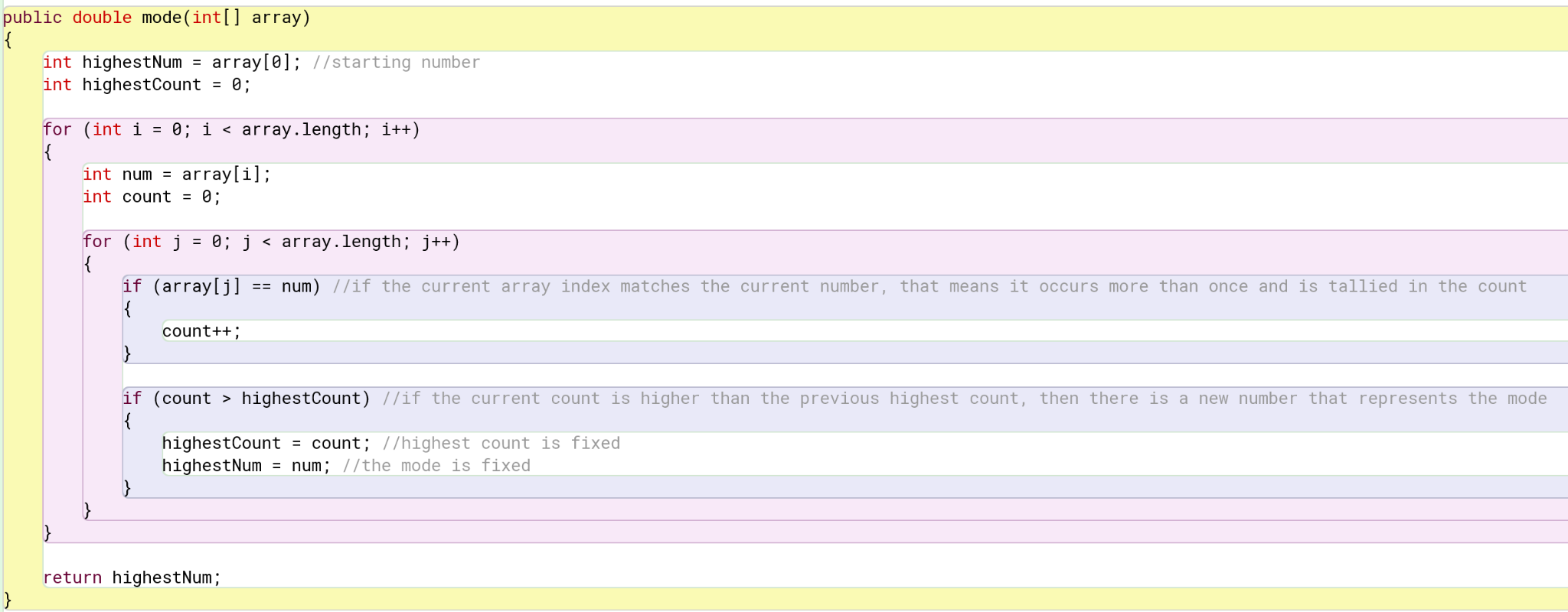


# Stats Library

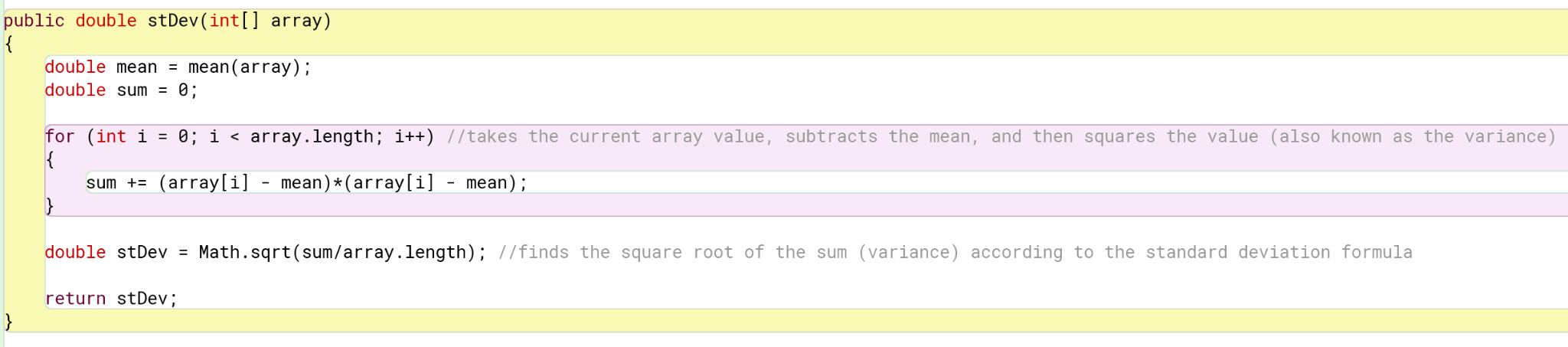
Below is the mean method that adds all the numbers in the array and divides by the array length

Below is the median method that finds the number in the middle of an odd-length array, or finds the middle two numbers in an even-length array and takes the average of the two.

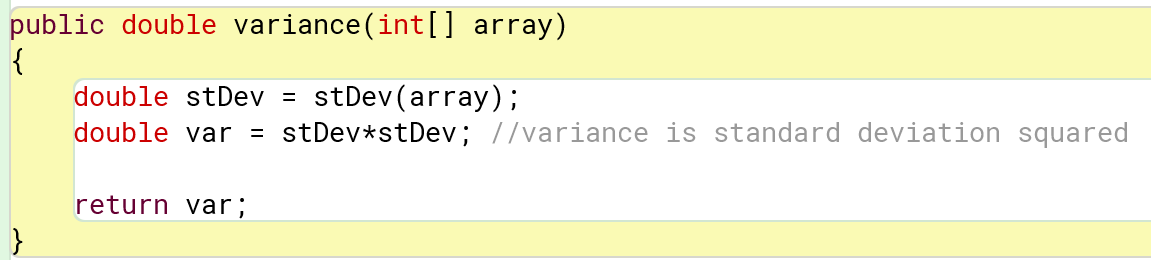
  
Below is the mode method that keeps track of the highest-appearing number.



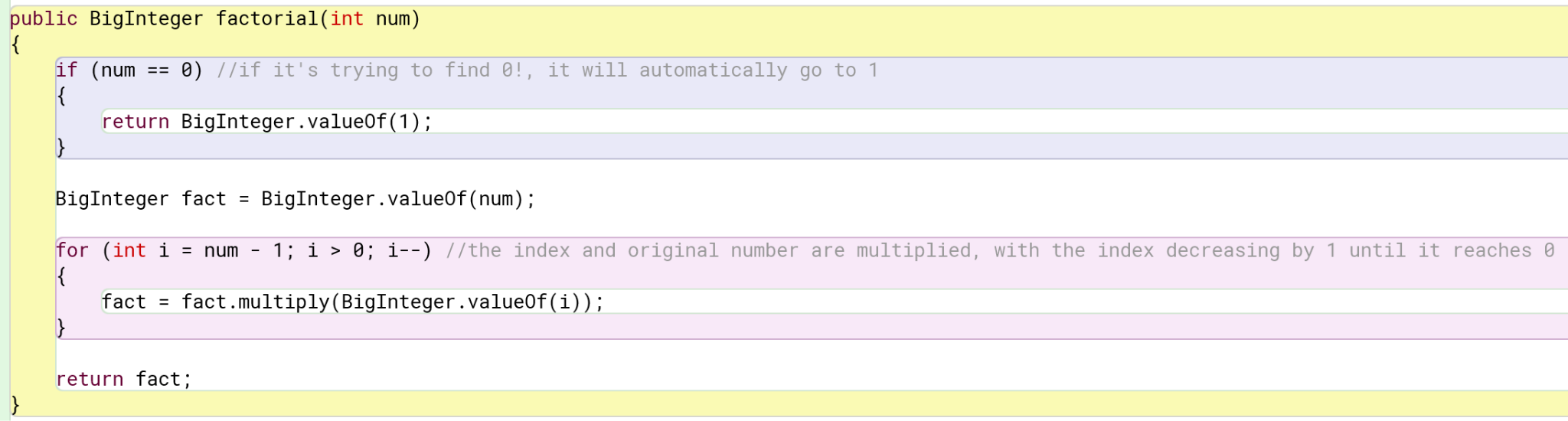
Below is the standard deviation method that uses a for loop to find the variance and then takes the square root of the result.



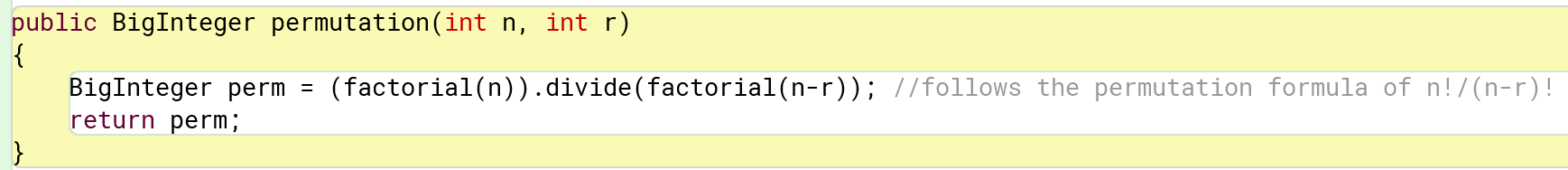
Below is the variance method that uses the standard deviation method to find the standard deviation and square it.



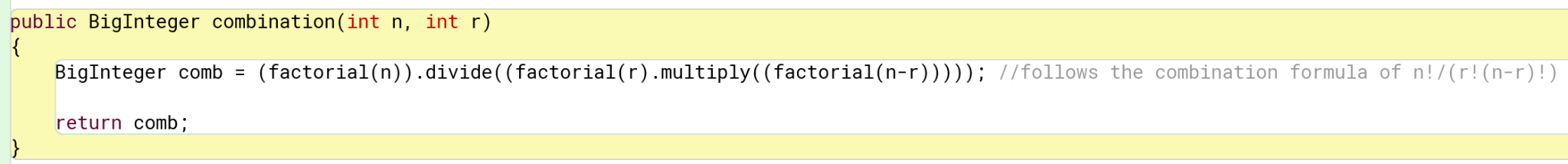
Below is the factorial method that repeatedly multiplies and decreases the index until it reaches 0.



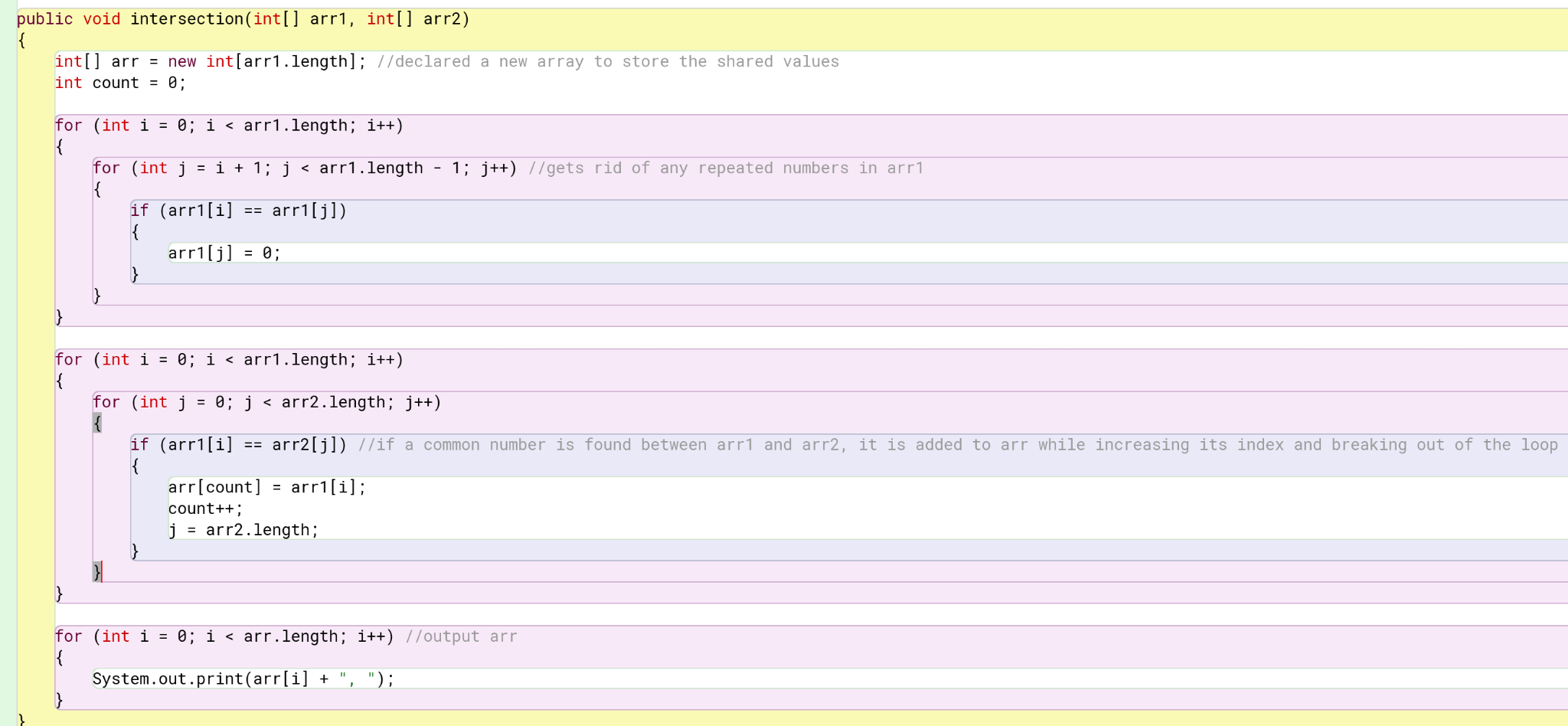
Below is the permutation method that uses the factorial method to follow the permutation formula.



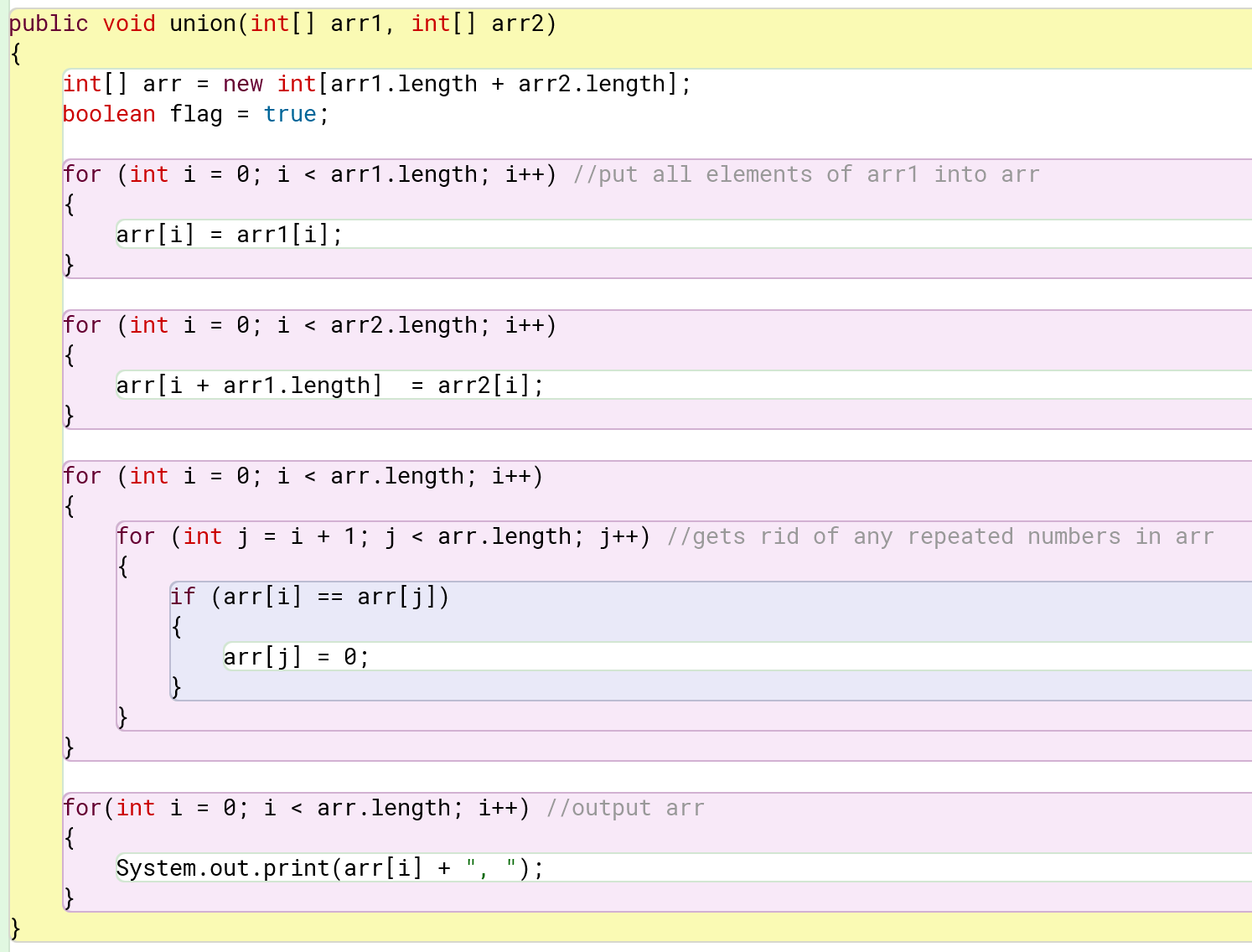
Below is the combination method that uses the factorial method to follow the combination formula.



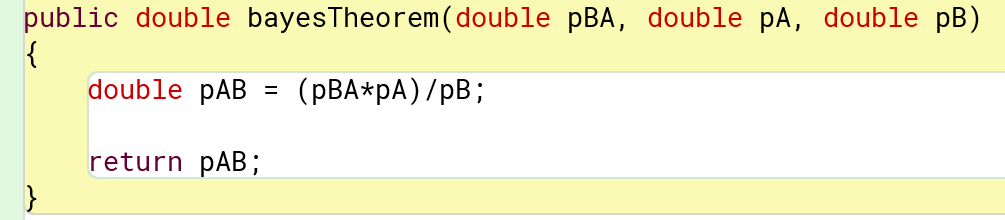
Below is the intersection method that combines two arrays and finds their common numbers. It gets rid of any repeated numbers in the first array and then checks if any of the remaining numbers are in the second array where they are then put into a third array.



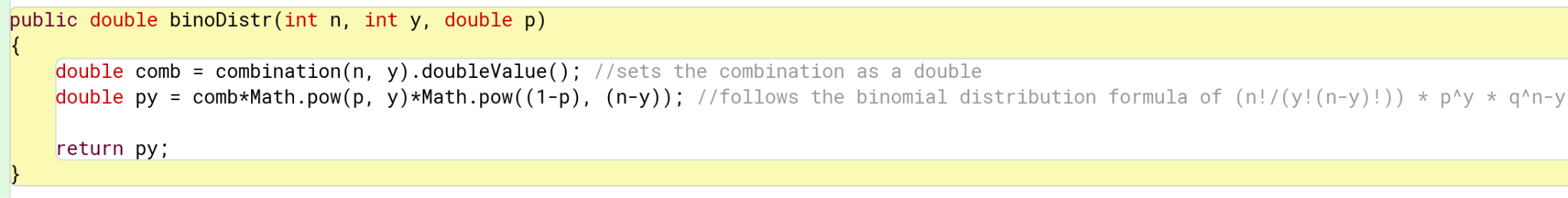
Below is the union method that combines two arrays without repeating numbers. It puts all elements of arrays 1 and 2 into a third array that is then checked for repeating numbers and gets rid of them.



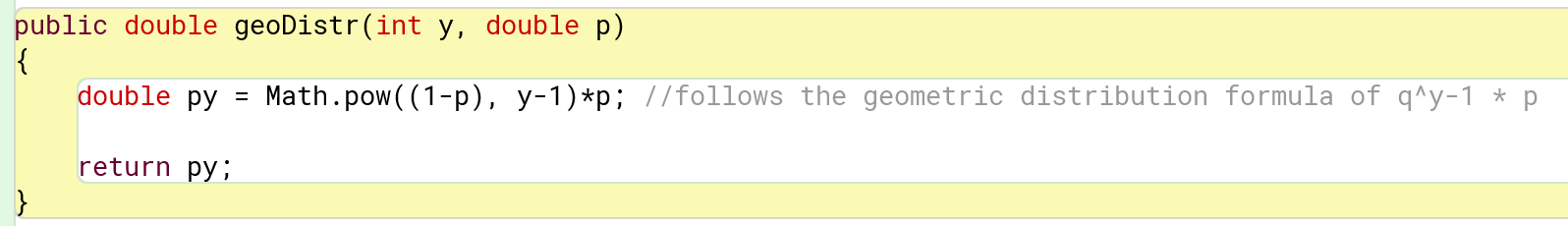
Below is the Bayes’ theorem method that takes P(B|A), P(A), and P(B) to calculate the probability of P(A|B).



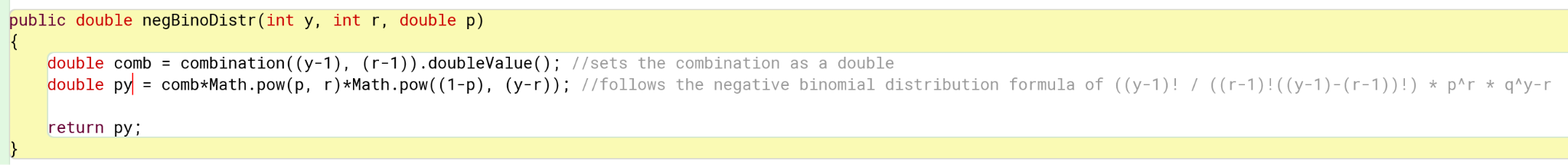
Below is the binomial distribution method that uses the combination and Math.pow methods to follow the binomial distribution formula.



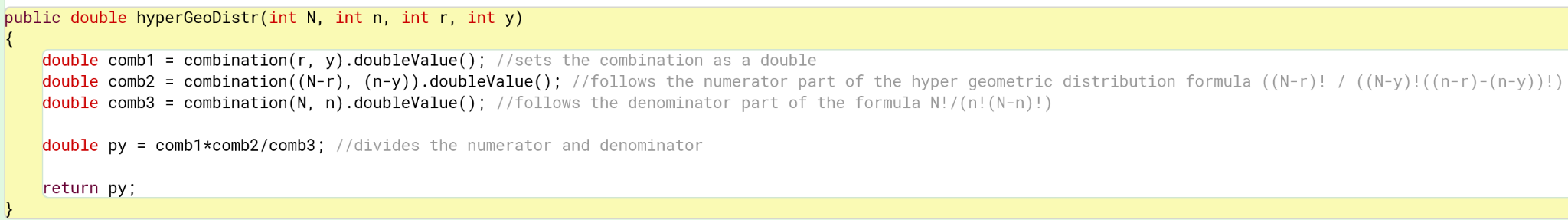
Below is the geometric distribution method that uses the Math.pow method to follow the geometric distribution formula.



Below is the negative binomial distribution method that uses the combination and Math.pow methods to follow the negative binomial distribution formula.



Below is the hypergeometric distribution method that uses the combination method multiple times to follow the hypergeometric distribution formula.



# Pokemon Game

I tried to make the Pokemon Game easy to understand, building my own toString method in order to show each card (and their indices) in your hand and bench. Your options (attack, retreat, etc) are also labeled with numbers.

Result:  
