Justin Oh

[Jjoh1993@gmail.com](mailto:Jjoh1993@gmail.com)

02/17/2019

Analysis of Java Math.Random Class

In many of my personal projects, I utilize the Math.Random class in Java in order to “reliably” produce random numbers. As a curiosity, I wanted to see how well distributed the numbers produced by the Math.Random class would be with a run of 1,000, 100,000, 10,000,000, and 1,000,000,000 numbers produced by the Math.Random method. For this analysis, I multiplied the number output by the Math.Random method by a multiplier of 100 to produce numbers from the range 0 to 99, inclusive.

In my source code, I initialize an array of varying length, which was the number of random numbers we wanted to test and assign a random number to each index in the array. The program then outputs a file, “RandomTest.txt”, that has csv values.

From the graphs, it was determined that the random number distribution of 1,000 random numbers would be insufficient for much of my programs where I required reliable “randomness.” Only when between 100,000 and 10,000,000 random numbers were generated did we yield a uniform distribution curve approaching 99% uniformity.

I determined that although the Math.Random class would produce sufficient randomness for games such as blackjack, in which true randomness is not expected due to the reshuffling of the deck, I would need to produce a sufficient amount of random numbers in order to provide a uniform distribution of random numbers.

From this experiment, I also determined that to produce an even distribution of random numbers in a memory-limited program, an effective approach would be to initialize an array of length 1000 with random numbers and output a random number by randomizing the index in the array at which the random number is chosen. Although this method takes O(n) time, and O(n) space, it is the most reliable method to provide uniformity where randomness is required.