

Re: can this code be improved

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2006-08/msg01362.html>

- *From:* Patricia Shanahan <pats@xxxxxxx>
 - *Date:* Thu, 17 Aug 2006 05:15:51 GMT
-

Print Guy wrote:

Hi all.

To most of you, what I have been working on is probably trivial and easy, but it's taken me the good part of a week working 2 or 3 hrs a day to finally figure out how to do what I needed to do.

Here in Canada, we have a lottery called 6-49. You pick 6 numbers between 1 and 49. Those numbers are put on a ticket and there is a draw twice a week.

I wanted to come up with a statistically solid way to pick my numbers so I figured that if I were to pick 6 numbers 1,000,000 times and count the number of times each number is selected, the top six would be good numbers to bet on during the lottery.

You are assuming that any bias in the Java Random class also applies to the lottery's random number generation method? I suspect it is using genuinely random numbers, not just pseudo-random, and monitors them for bias.

With that being said, I thought I could write a Java program to do the job. It proved to be a bit more difficult than I thought it would be; the hardest part being ensuring that each of the 6 numbers were unique.

Here is an alternative method for picking the 6 numbers.

There is a well-known linear time algorithm for shuffling an array, the Fisher-Yates shuffle. On iteration *i*, the element with index *i* is swapped with a randomly selected element between *i* and the end of the array.

There are two important loop invariants:

1. At the end of every iteration, the array contains a permutation of its original contents.

Re: can this code be improved

2. After N iterations, the first N elements of the array are randomly selected with equal probability.

In a normal shuffle, the number of iterations is equal to the size of the array minus one, and all elements have been shuffled. (The last iteration is not needed because it makes a random choice from one element, and swaps that element with itself).

Why not run Fisher–Yates for 6 iterations, and then pick up the first 6 elements of the array?

Patricia

```
/**
 * Partially shuffle the elements of an int
 * array
 *
 * On completion, data contains a permutation of
 * its original contents, with the elements in
 * the first count spaces selected randomly.
 * There is no assurance of randomness for
 * elements data[count] and later, and many of
 * those elements may remain in thei
```