**Replace the ArrayList in your assignment with a LinkedList from the Java collections.**

There were only a couple changes required because swapping the implementation of lists with the 'java' collection is a rather simple process. I inserted an import statement for the LinkedList class in the java.util package. This is important because, like ArrayList, LinkedList is part of the Java Collections Framework and must be imported before it can be used. In the runSimulation method, when the ArrayList of Contestant objects was first declared, I replaced it with LinkedList. This change was made to the contestants list during its modification. For testing I simply ran the code, waited quite a while, then compared the output numbers.

**Compare the timing for using an ArrayList with the same timing for using a LinkedList with 43 contestants:**

Linked:

A screenshot of a computer program

Description automatically generated

|  |  |
| --- | --- |
| Rounds | Time |
| 10 | 1 |
| 100 | 1 |
| 1000 | 7 |
| 10000 | 23 |

Array:

A screenshot of a computer program

Description automatically generated

|  |  |
| --- | --- |
| Round | Time |
| 10 | 0 |
| 100 | 1 |
| 1000 | 7 |
| 10000 | 29 |

It seems that for the round of 10000, Linked is roughly 21% faster than the Array.

Compare the timing for using an ArrayList with the same timing for using a LinkedList with 45000 contestants:

Linked:

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| Round | Time |
| 10 | 39275 |
| 100 | 392872 |
| 1000 | 3928799 |
| 10000 | 39588020 |

Array:

A screenshot of a computer program

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Rounds | Time |  |
| 10 | 1099 |  |
| 100 | 11060 |  |
| 1000 | 110640 |  |
| 1000 | 1106430 |  |

It seems that for the round of 10000 and 45000 contestants, Linked is roughly 97.21% faster than the Array.

**Consider your proposed algorithm from Unit 1 that rearranges Contestants in the same list. Based on your timing estimates, explain whether you think it would take more or less time using a LinkedList rather than an ArrayList.**

In U1A2 I was I used Random Rearrangement Implementation where contestants are moved to a random position in the list after each round. I think that based on this testing the Linked list would offer much less processing time and CPU pull in the process and is generally more efficient

**In a sentence or two, what did you learn?**

I learned the practical value of each type of array and which is better for each purpose in that random access in an ArrayList is very fast, but inserting or removing elements can be slow because it may require shifting elements and that the larger the array the more this is evident. I learned that a LinkedList would again be more efficient for random insertions and deletions, as it can add or remove elements without needing to shift the rest of the elements in the list.

**In a sentence or two, what did you like about this project?**

I liked being able to directly see the results of two processes and compare in real time.

**In a sentence or two, what did you find confusing or would like to see done differently regarding this project?**

There was nothing particularly confusing in this one.

**In a sentence or two, if you had another hour or two, what would you like to add to the project or how would you do things differently?**

I would have set aside more time and not have been waiting for the completion o the code to run as it took quite a while to wait for.