|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1,000 elements (in ms)** | ArrayList Add | ArrayList Sort | ArrayList Shuffle | ArrayList get random 1mill | ArrayList get Squential |
| Run 1 | 3.9531 | 1.8174 | 0.848 | 34.0845 | 0.0366 |
| Run 2 | 3.7992 | 2.243 | 0.8328 | 34.1883 | 0.0346 |
| Run 3 | 3.9388 | 2.1234 | 0.9213 | 35.8429 | 0.0278 |
| **Average** | **3.897033333** | **2.061266667** | **0.867366667** | **34.70523333** | **0.033** |
| **1,000 elements (in ms)** | LinkedList Add | LinkedList Sort | LinkedList Shuffle | LinkedList get random 1mill | LinkedList get Squential |
| Run 1 | 0.877 | 2.1923 | 0.5089 | 419.2979 | 0.4217 |
| Run 2 | 0.7023 | 1.8658 | 0.5165 | 596.4537 | 0.4165 |
| Run 3 | 0.5704 | 2.9563 | 0.648 | 469.55 | 0.3968 |
| **Average** | **0.716566667** | **2.338133333** | **0.5578** | **495.1005333** | **0.411666667** |
| **5,000 elements (in ms)** | ArrayList Add | ArrayList Sort | ArrayList Shuffle | ArrayList get random 1mill | ArrayList get Squential |
| Run 1 | 7.3871 | 4.3442 | 1.9712 | 36.7165 | 0.1685 |
| Run 2 | 7.1588 | 4.496 | 1.4708 | 34.5346 | 0.1469 |
| Run 3 | 8.4292 | 5.1246 | 1.4031 | 33.4461 | 0.1861 |
| **Average** | **7.658366667** | **4.654933333** | **1.615033333** | **34.89906667** | **0.167166667** |
| **5,000 elements (in ms)** | LinkedList Add | LinkedList Sort | LinkedList Shuffle | LinkedList get random 1mill | LinkedList get Squential |
| Run 1 | 2.6675 | 4.2439 | 1.2721 | 2540.0556 | 10.5599 |
| Run 2 | 7.3539 | 3.2382 | 1.2132 | 2656.6949 | 10.489 |
| Run 3 | 2.0096 | 4.778 | 1.2036 | 2422.8758 | 10.7534 |
| **Average** | **4.010333333** | **4.0867** | **1.229633333** | **2539.875433** | **10.60076667** |
| **10,000 elements (in ms)** | ArrayList Add | ArrayList Sort | ArrayList Shuffle | ArrayList get random 1mill | ArrayList get Squential |
| Run 1 | 8.7396 | 8.546 | 2.7868 | 35.3602 | 0.3712 |
| Run 2 | 8.2622 | 10.737 | 2.3988 | 36.3443 | 0.3972 |
| Run 3 | 10.1244 | 11.6202 | 2.5198 | 36.2238 | 0.342 |
| **Average** | **9.042066667** | **10.30106667** | **2.568466667** | **35.9761** | **0.370133333** |
| **10,000 elements (in ms)** | LinkedList Add | LinkedList Sort | LinkedList Shuffle | LinkedList get random 1mill | LinkedList get Squential |
| Run 1 | 2.245 | 4.8225 | 4.0737 | 4489.0723 | 47.1017 |
| Run 2 | 3.2322 | 8.1092 | 4.6961 | 4562.8504 | 46.9957 |
| Run 3 | 2.3286 | 6.8621 | 3.2409 | 4619.4939 | 49.3985 |
| **Average** | **2.601933333** | **6.597933333** | **4.003566667** | **4557.138867** | **47.83196667** |

An ArrayList is better at getting elements from the list, whether that is sequentially or randomly. A linkedList is better at adding elements to the list, sorting the list, and shuffling. An ArrayList is faster at getting from the list than a LinkedList because the ArrayList stores its data with an index, meaning the computer can just go to a certain index of the list and get that value. For a linkedList, the computer is forced to go through all the data to find the next element in the list until it gets the desired value.

For adding elements to a list, a linkedList is better since all you need to do is add a reference to the next element of the list to the end tail of the linkedList. For an ArrayList, you need to create an entirely new array just to add another element. This difference is especially prevalent for larger arrays.

For sorting, a linkedList is generally faster at sorting for a larger number of elements. This may be because adding elements and redirecting elements to create a different order in a linkedlist is faster than creating a new array as in an ArrayList.

For shuffling, an Arraylist is generally faster at shuffling a larger number of elements. This is probably due to how much faster it is for an Arraylist to get a random object from the list than it is for a linkedList, allowing it to randomly shuffle much faster than a linkedList.