Assignment 1 Part 2

|  |  |
| --- | --- |
| List type | Time (milliseconds) |
| Array | 261938 |
| ArrayList | 261785 |
| LinkedList | 782296 |

For part 2 of the assignment, we used 1,000,000 elements. For number one of part 2, this would have a run time about .016 seconds of run time. We chose to use this number of elements because it allows us to run other tests with the different list types. When we tried to use a higher amount such as 10,000,000 or 2,000,000 then the LinkList test would run for many hours and eventually give an error. This error was an outOfMemory Exception. A solution that we tried would be to increase the heap size for the virtual machine. This does not work; the program would run for more than 6 hours and give an outOfMemory error in the class LinkedList. The number of elements in this case that would work is 1,000,000 elements.

The run times of the different lists can show you a lot about the design of each. In the table above, it could be seen that Array, and ArrayList are similar in run times. This is to be expected because they are very similar in their code. ArrayList is basically the same as an Array, but the methods in ArrayList allows the user or the person using the code to be more flexible with an array. It could also be seen that in a LinkedList, it takes almost triple the amount of time to run compared to the Array, and ArrayList. This is due to the fact that in order for a LinkedList to go to the next element, it must traverse everything before it. In an array, it can access the elements wherever it wants. This would make it very fast since we are trying to add the element at each index of the array. So, in this case, arrays are created to be accessed faster than LinkedList.

One way that designers can make the LinkedList faster for access is to put a divider. For example, if you have 10 elements in a LinkedList, you can save the object at the 5th index. And if the person using the code wanted the 7th element, then it can start at the 5th element, instead of having to go through an extra 5 elements.