Execution Time

Note: Execution time is in milliseconds

| Tests | Array | ArrayList | Vectors |
| --- | --- | --- | --- |
| Adding to the list of size 5000000\*5 | 1: 1908  2: 2308  3: 2067  4: 1985  Avg: ~2067 | 1: 996  2: 986  3: 1001  4: 992  Avg: ~994 | 1: 986  2: 1103  3: 1050  4: 1108  Avg: ~1062 |
| Adding to the list of size 30000000\*5 | 1: 9695  2: 9068  3: 10659  4: 10007  Avg: ~9857  1: 9781  2: 11194  3: 12504  4: 11162  Avg: ~11160 | Without Capacity  1: 2885  2: 3100  3: 3006  4: 3108  Avg: ~3025  With Capacity  1: 2474  2: 2433  3: 2403  4: 2423  Avg: ~2433 | Without Capacity  1: 4472  2: 4548  3: 4615  4: 4449  Avg: ~4521  With Capacity  1: 3065  2: 3299  3: 3190  4: 3134  Avg: ~3172 |
| Adding to the list of size 50000000\*5 | 1: 26871  2: 22688  3: 27042  4: 26322  Avg: ~25731  1: 20104  2: 23883  3: 24621  4:25856  Avg: ~ 23616 | Without Capacity  1: 12365  2: 12345  3: 12284  4: 12393  Avg: ~12347  With Capacity  1: 4032  2: 3954  3: 3833  4: 3932  Avg: ~3938 | Without Capacity  1: 5702  2: 5827  3: 5769  4: 5741  Avg: ~7760  With Capacity  1: 5060  2: 5282  3: 5422  4: 5298  Avg: ~5265 |
| Adding integers in the list of size 30000000\*5 with ArrayList and Vector as a for-loop | 1: 52  2: 120  3: 134  4: 223  Avg: ~132 | 1: 84  2: 194  3: 59  4:57  Avg: ~99 | 1: 737  2: 747  3: 724  4: 741  Avg: ~737 |
| Adding integers in the list of size 30000000\*5 with ArrayList and Vector as an iterator | 1: 207  2: 53  3: 97  4: 59  Avg: ~104 | 1: 157  2: 219  3: 179  4: 80  Avg: ~159 | 1: 825  2: 781  3: 779  4: 807  Avg: ~798 |

To conclude, overall using arraylist is more efficient but when adding more functions into the list, vector is more stable. Thus, depending on what you are doing, if your goal is to have infinite size list then a vector is best. But if you are using the list, then an arraylist if it does not have a finite size.