	add	sort	shuffle	random get	sequential get
ArrayList (1,000)	5.31	2.42	1.09	57.74	0.076
LinkedList (1,000)	2.62	1.87	0.82	456.57	0.71
ArrayList (5,000)	15.58	16.89	7.69	63.93	1.0358
LinkedList (5,000)	7.87	16.64	3.62	2,919.07	16.96
ArrayList (10,000)	22.57	18.58	10.73	78.71	0.9697
LinkedList (10,000)	7.32	15.54	7.01	11,328.52	83.00

For adding elements to the list, the LinkedList implementation appeared to work the best. For getting elements from the list (sequentially or randomly), the ArrayList implementation appeared to work the best. For sorting and shuffling the list, both implementations worked at relatively the same level, but the LinkedList implementation worked slightly better.

These results are consistent with how the lists are implemented because ArrayList stores its objects in memory, so it is more easily able to find the index a particular object is stored at than LinkedList, which does not use indices, only references. Using references is also why LinkedList is better at adding elements than ArrayList. While ArrayList must create a new index each time an object is added to it, LinkedList simply has to change its pointer, which is much easier.