Collections Notes:

List:

1. ArrayList:

* Resizable array i.e. size increases automatically.
* Implements List interface
* It is unsynchronized i.e. multiple threads can access a list at same time.
* Size , isEmpty, get, set, iterator, listIterator operations runs in constant time.
* Add operation runs in amortized constant time i.e. inserting ‘n’ Elements will take O(n) time.
* All other operations take linearly constant time.

## LinkedList:

* Doubly Linked List implementation of List and Dequeue interface.
* It is not synchronized.

ArrayList vs LinkedList:

* Deletion will be easy in linked list because it doesn’t have to shift

Other elements on right.

* Memory allocation in arraylist is contiguous while in linked list it is

Not contiguous.

* Insertion is fast in the linked list.
* Random access is fast in arraylist.
* Search operation is equal.

Conclusion:

We should choose LinkedList when operations like data addition or deletion occur more frequently than reading the data. Conversely, use ArrayList when data reading scenarios are more common than adding or removing data. This is because ArrayList is better equipped for frequent data retrievals.

The data in an ArrayList is stored more compactly than in LinkedList, making ArrayList more cache-friendly and reducing the likelihood of cache misses. In contrast, LinkedList tends to have poorer cache-locality due to its scattered data storage.

Memory overhead is another critical factor. LinkedList consumes more memory than ArrayList due to the additional links needed to connect nodes. Each node in LinkedList holds the address of both the previous and the next nodes, requiring extra space. This additional link structure is not present in ArrayList, making it more memory-efficient.

1. HashSet:

* It implements Set interface and is based on a hashtable.
* Insertion order is not maintained in HashSet.
* Allows storing one null element.
* Takes constant time for add, remove, size, contains,
* Traversal time depends on the number of elements plus buckets present due to the hashtable.
* It is non-synchronized.

1. LinkedHashSet: implements LinkedList and Set interface. Backed by a hashtable.

* Maintains insertion order.
* Performance would be slightly lower than HashSet due to maintenance of LinkedList.
* It is not synchronized.

1. TreeSet:

* Implements a navigable set and backed by TreeMap.
* Elements are stored in natural order using a natural comparable or a custom comparator provided.
* Takes log(n) time for add, remove and contains operation.
* Not synchronized.

ArrayList vs TreeSet:

1. ArrayList takes less time for insertion.
2. Searching is fast in treeset.

Put vs Patch:

* Put is used to update a resource completely.
* Patch is used to update partially.

## 