**Collections:**

**Q. what is the difference between Collection and collections?**

Ans: - Collection is an interface while collections is a class.

* Collection interface provides normal functionality of data structures to List, Set, Queue, while Collection class is to sort and synchronize collection elements.

**Q: What is the difference between ArrayList and Vector?**

Ans:

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList is not synchronized. | Vector is synchronized. |
| 2) | ArrayList is not a legacy class. | Vector is a legacy class. |
| 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by specified size if not specified then double. |

**Q: Difference between ArrayList and LinkedList** ?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because a lot of shifting is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |

**Q.what is the difference between Iterator and Enumeration**

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | Iterator is fail-fast. | Enumeration is not fail-fast. |
| 3) | Iterator is slower than Enumeration. | Enumeration is faster than Iterator. |

When a problem occurs, a **fail**-**fast** system **fails** immediately.

A fail-fast system is nothing but immediately report any failure that is likely to lead to failure. When a problem occurs, a fail-fast system fails immediately. In Java, we can find this behavior with iterators. Incase, you have called iterator on a collection object, and another thread tries to modify the collection object, then concurrent modification exception will be thrown. This is called fail-fast.

"Fail fast" means: it may fail ... and the failure condition is checked aggressively so that the failure condition is detected before damage can be done.

**Q.What is the Difference between List and Set ?**

List may contain duplicate items where as set will contain only unique items.

**Q. what is the difference between HashSet and TreeSet?**

Ans: HashSet maintains no order while TreeSet maintains ascending order.

Q: **what is the difference between Iterator and ListIterator** ?

1) Iterator is used for traversing List and Set both.

We can use ListIterator to traverse List only, we cannot traverse Set using ListIterator.

2) We can traverse in only forward direction using Iterator.

Using ListIterator, we can traverse a List in both the directions (forward and Backward).

3) We cannot obtain indexes while using Iterator

We can obtain indexes at any point of time while traversing a list using ListIterator. The methods nextIndex() and previousIndex() are used for this purpose.

4) We cannot add element to collection while traversing it using Iterator, it throws ConcurrentModificationException when you try to do it.

We can add element at any point of time while traversing a list using ListIterator.

5) We cannot replace the existing element value when using Iterator.

By using set(E e) method of ListIterator we can replace the last element returned by next() or previous() methods.

6) Methods of Iterator:

* hasNext()
* next()
* remove()

Methods of ListIterator:

* add(E e)
* hasNext()
* hasPrevious()
* next()
* nextIndex()
* previous()
* previousIndex()
* remove()
* set(E e)

Q: **What is difference between HashMap and HashTable?**

|  |  |
| --- | --- |
| **HashTable** | **HashMap** |
| HashTable is ThreadSafe and Synchronized | HashMap is not Thread safe and synchronized |
| HashTable does not allow null value of null keys in HashTable objects. | HashMap allows one null key and multiple null values. |
| HashTable uses enumerator | HashMap uses Iterator |
| The iterator in hashMap is fail-fast | Enumerator is not fail fast. |
| hashTable is slower and uses more memory | HashMap is faster and uses less memory. |
| Its legacy class that is obsolete | Its collection class currently in use |

**Thread Safe:** A piece of code is **thread**-**safe** if it only manipulates shared data structures in a manner that guarantees**safe** execution by multiple **threads** at the same time

#### Q: List of classes that implement Set interface?

* **SortedSet** - It is an interface which extends Set. A the name suggest , the interface allows the data to be iterated in the ascending order or sorted on the basis of Comparator or Comparable interface. All elements inserted into the interface must implement Comparable or Comparator interface.
* **TreeSet** - It is the implementation of SortedSet interface.This implementation provides guaranteed log(n) time cost for the basic operations (add, remove and contains). The class is not synchronized.
* **HashSet**: This class implements the Set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the null element. This class offers constant time performance for the basic operations (add, remove, contains and size), assuming the hash function disperses the elements properly among the buckets

#### Q:Specify conditions When to use ArrayList or LinkedList ?

#### Adding new elements is pretty fast for either type of list. For the ArrayList, doing  random lookup using "get" is fast, but for LinkedList, it's slow. It's slow because there's no efficient way to index into the middle of a linked list. When removing elements, using ArrayList is slow. This is because all remaining elements in the underlying array of Object instances must be shifted down for each remove operation. But here LinkedList is fast, because deletion can be done simply by changing a couple of links. So an ArrayList works best for cases where you're doing random access on the list, and a LinkedList works better if you're doing a lot of editing in the middle of the list.

#### Q: possible ways to read a list, which one is better and why, which one takes less time ?

#### For each loop

#### Iterator

#### It can be done in two ways, using for loop or using iterator of ArrayList. The first option is faster than using iterator. Because value stored in arraylist is indexed access. So while accessing the value is accessed directly as per the index.For loop does not allow the updation in the array(add or remove operation) inside the loop whereas Iterator does. Also Iterator can be used where there is no clue what type of collections will be used because all collections have iterator.

#### Q: What is difference between iterator access and index access?

* Index based access allow access of the element directly on the basis of index. The cursor of the data structure can directly goto the 'n' location and get the element. It does not traverse through n-1 elements.
* In Iterator based access, the cursor has to traverse through each element to get the desired element.So to reach the 'n'th element it need to traverse through n-1 elements.
* Insertion,updation or deletion will be faster for iterator based access if the operations are performed on elements present in between the datastructure.
* Insertion,updation or deletion will be faster for index based access if the operations are performed on elements present at last of the datastructure.
* Traversal or search in index based datastructure is faster.
* ArrayList is index access and LinkedList is iterator access.

#### Q: which interface is required , if you want to sort list in reverse order?

#### Comparator and comparable

#### Q: can you make a List (ArrayList,Vector,LinkedList) read only, how?

#### Collections.unModifiableList()

#### Q: Can you obtain array from ArrayList ? how?

#### Using toArray() method of ArrayList class

List<String> list = new ArrayList<String>();

list.add("India");

list.add("Switzerland");

list.add("Italy");

list.add("France");

String [] countries = list.toArray(new String[list.size()]);

#### Q: Why insertion and deletion in ArrayList is slow compared to LinkedList ?

* **ArrayList**internally uses and array to store the elements, when that array gets filled by inserting elements a new array of roughly 1.5 times the size of the original array is created and all the data of old array is copied to new array.
* During deletion, all elements present in the array after the deleted elements have to be moved one step back to fill the space created by deletion. In linked list data is stored in nodes that have reference to the previous node and the next node so adding element is simple as creating the node an updating the next pointer on the last node and the previous pointer on the new node. Deletion in linked list is fast because it involves only updating the next pointer in the node before the deleted node and updating the previous pointer in the node after the deleted node.

Q**: Why are Iterators returned by ArrayList called Fail Fast ?**

Because, if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.