lesson 15 - COLLECTIONS

COLLECTIONS is a class with static utility methods.

COLLECTION is an interface with declarations of methods common to most collection classes including size(), remove(), add(), contain(), iterator().

* Collections framework standardizes the way in which groups of objects are handled by your programs.
* Collections framework defines several interfaces. We begin with collection interface as it defines the fundamental nature of the collection classes.

Interfaces :

1. SET
   1. HashSet
      1. With the process of hashing, hashcode is generated. And generation of hashcode doesn’t guarantee a specific order.
      2. Uses a hashtable for storage.
      3. Elements are stored in a **RANDOM** order
   2. LinkedHashSet
      1. This class **extends** HashSet
      2. Maintains a linked list of the entries in the set, in the order in which they were inserted. This allows insertion order iteration.
   3. TreeSet ( this also extends SortedSet interface)
      1. Sorts the input elements in **ascending order**
2. LIST
   1. ArrayList
      1. Can dynamically increase or decrease in size
      2. Elements can be accessed by specifying index(follows zero-based indexing)
      3. An initial size can be mentioned while creating an arrayList, but the size is not fixed.
   2. Vector
      1. Implements a dynamic array
      2. Similar to ArrayList but with 2 differences
         * 1. Vector is synchronized
           2. Contains many legacy methods that are not part of the collections framework.
      3. Default vector is initialized with size 10. After this initial capacity is reached, and more objects need to be stored in the vector, it automatically allocates space for that object, plus extra room for additional objects. By allocating more than the required memory, it reduces the number of allocations that must take place.
      4. If you don’t specify the increment, vector’s size is doubled in each allocation cycle.
   3. LinkedList (implements both List and Queue interfaces)
      1. Provides a linked list data structure
      2. Elements can be accessed or modified using the index since elements are now store in linked list data structure
      3. Elements are store in the same order as they are inserted.
3. QUEUE
   1. PriorityQueue
4. MAP

-----> Map does not implement the collection interface

---> map interface maps unique keys to values(key-value pairs). Key must be unique, values may be duplicated.

* 1. HashMap
     1. Stores in random order
     2. Elements are stored in the form of key value pairs
  2. LinkedHashMap
  3. TreeMap
     1. Elements stored are sorted in ascending order
     2. However, in treeMap, this order of sorting can be defined by the user by using comparator interface.
  4. Hashtable
     1. Similar to hashMap, but synchronized
     2. Does not directly support iterators. However, you can obtain set-views of the hashtable, which permits the use of iterators. So you can obtain the set-view of the keys(keyset()) and iterate through them.

COMPARATOR INTERFACE

* By default, TreeSet and TreeMap store their elements in an order that java refers to as “natural ordering”, i.e. A before B, 1 before 2 etc..
* By using comparator interface, user can define its own sorting order
* Compaator defines 2 methods –
  + compare()
    - compare(Object obj1, Object obj2) returns 0 if obj1 and obj2 are equal
  + equals()