Advance Programming

Assignment -2

Abdul Sami

(14L-4103)

Array List vs Vector		
SR #	**ArrayList**	**Vector**
1	ArrayList executes List interface and keeps up insertion order, anyway internal implementation is not the same as that of Vector.	Vector actualizes List interface and keeps up request of inclusion, anyway inward usage is not the same as that of Array List.
2	In ArrayList there is no Synchronization. This is key contrast among Arraylist and vector	In vector there is Synchronization. This is key difference between Arraylist and vector
3	Because of no synchronization beyond what one string can get to the ArrayList which makes it less thread safer and more venerable.	Because of synchronization not beyond what one thread can get to the Vector which makes it more thread more secure and hard to misuse.
4	Quick Access to components because of no elite thread lock and holding up of threads in queue	Slow Access to elements due to exclusive thread locking mechanism and waiting of threads in queue
5	Dynamic growth and shrinking of size. ArrayList grows by 50% of current size	Dynamic growth and shrinking of size. Vector grows by doubling the current size.
6	Use only Iterator as traversal tool.	Use Iterator and enumeration as traversal tool
7	Arraylist isn't a legacy class. Presented later and ought to be utilized if secure with single thread access.	Vector is a legacy class.
References: https://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html https://www.geeksforgeeks.org/vector-vs-arraylist-java/		

Differences Between HashSet & Sorted Set		
SR #	**HashSet**	**SortedSet**
1	Unordered collection of unique objects.	A collection of objects that contains no duplicate elements Also arranged in orderly manner
2	HasSet has contiguous storage It can be accessed directly by key.	SortedSet has contiguous storage It can be accessed directly by key
3	Faster access Time Complexity is O(1).	Slow access as compare to Sorted Set Time Complexity is O(log n)
4	HashSet uses a hash-table	SortedSet uses a red-black tree and a balanced binary tree
5	Used when there is no need of sorted elements	Used when there is need of sorted elements
References: https://docs.oracle.com/javase/7/docs/api/java/util/SortedSet.html http://geekswithblogs.net/BlackRabbitCoder/archive/2011/06/16/c.net-fundamentals-choosing-the-right-collection-class.aspx		

Differences Between TreeSet & HashSet		
SR #	**TreeSet**	**SortedSet**
1	It implements the Set interface that uses a tree for storage It inherits AbstractSet class and implements the NavigableSet interface	A collection of objects that contains no duplicate elements Also arranged in orderly manner
2	Does not allow to insert Heterogeneous objects It will throw classCastException at Runtime if trying to add hetrogeneous objects	SortedSet uses a red-black tree and a balanced binary tree
3	For remove and search Time Complexity is O(Log n) For printing it is O(n)	Slow access as compare to Sorted Set Time Complexity is O(log n)
4	All elements inserted must be *mutually comparable* *i.e epm1.compare(emp2)*	All elements inserted must be *mutually comparable* *i.e epm1.compare(emp2)*
References: https://docs.oracle.com/javase/6/docs/api/java/util/TreeSet.html https://docs.oracle.com/javase/6/docs/technotes/guides/collections/index.html		

Differences Between Array & List		
SR #	**Array**	**List**
1	Fixed length data structure Its length cannot be modified once array object is created	Variable length data structure Its length can be modified once list object is created
2	Unlike sets, array typically allow duplicate elements	Unlike sets, lists typically allow duplicate elements
3	Array is not grow able It has fixed size	List is dynamically growable in nature No fixed size
4	Length variable is used to determine the length of the Array	List uses size () method to determine the size of the ArrayList It is rather different from determining the length of the Array
5	Iterating over an array is faster than iterating over a List	Iterating over a List is slower than iterating over an Array
References: http://www.differencebetween.net/technology/software-technology/difference-between-arraylist-and-vector/ https://www.javatpoint.com/java-list		

Differences Between List & Set		
SR #	**List**	**Set**
1	Variable length data structure Its length can be modified once list object is created	A collection Contains no duplicate elements
2	New methods are defined inside interface	No new methods are defined inside interface
3	Order is important	Use sets when all you're interested in is membership
4	Element can be added again	If same element is added again, there won't be any compile-time or runtime error, just that add() method returns false
References: https://docs.oracle.com/javase/8/docs/api/java/util/List.html		

Differences Between NavigableSet & NavigableMap		
SR #	**NavigableSet**	**NavigableMap**
1	It represents a navigable set in Java Collection Framework This interface inherits from the SortedSet interface It behaves like a SortedSet with the exception that we have navigation methods available in addition to the sorting mechanisms of the SortedSet	NavigableMap is an extension of SortedMap It provides convenient navigation method like lowerKey, floorKey, ceilingKey and higherKey
2	It can be accessed and traversed in either ascending or descending key order	It may be accessed and traversed in either ascending or descending key order
3	Navigable Set lies under collections	Unlike Navigable Set, Navigable map doesn't lie under collections
4	NavigableSet has different methods like : NavigableSet lower() method in Java NavigableSet higher() method in Java NavigableSet floor() method in Java NavigableSet subSet() method in Java NavigableSet ceiling() method in Java NavigableSet headSet() method in Java	NavigableMap has different methods like: NavigableMap headMap() in Java NavigableMap put() Method in Java NavigableMap firstEntry() method in Java NavigableMap lowerEntry() method in Java NavigableMap pollFirstEntry() method in Java NavigableMap ceilingEntry() method in Java
References: https://www.geeksforgeeks.org/navigableset-java-examples/