**Advanced Programming**

**Assignment 2**

**Fatima Naseer**

**16L-4257 (CS)**

**Difference between ArrayList between Vector**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| 1. ArrayList increases its size by 50% of the array size. | 1. Vector increases its size by doubling the array size. |
| 1. ArrayList is a non-synchronized data structure that uses a dynamic array for storing elements and is fast. | 1. Vector is synchronized data structure that uses a dynamic array for storing elements and is slow. |
| 1. ArrayList is a non-legacy class. | 1. Vector is a legacy class. |
| 1. ArrayList is not thread safe as multiple threads are allowed to operate on ArrayList object at a time. | 1. Vector is thread safe as only one thread is allowed to operate on Vector Object at a time. |
| 1. ArrayList uses Iterator interface to traverse elements. | 1. Vector uses Enumeration interface to traverse the elements. But it can use Iterator too. |

**Difference between HashSet and SortedSet**

|  |  |
| --- | --- |
| **HashSet** | **SortedSet** |
| 1. Represent group of individual objects as a single entity where duplicates are NOT allowed and NO insertion order is preserved. | 1. Represent group of individual objects as a single entity where duplicates are NOT allowed and insertion order is preserved. |
| 1. It creates a collection that **uses** a hash table for storage. A hash table stores information by using a mechanism called hashing. | 1. All elements of a SortedSet must implement the Comparable interface and all such elements must be mutually comparable. |
| 1. It allows null value and is not thread-safe. | 1. All methods of ‘Set’ are inherited in ‘SortedSet’. So, we can use set-operations like add, addAll, clear etc. |
| 1. It has faster mechanism. | 1. It provides operations like Range view, Endpoints of the sorted set, Comparator access etc. |

**Difference between TreeSet and HashSet**

|  |  |
| --- | --- |
| **TreeSet** | **HashSet** |
| 1. TreeSet provides ordering/sorting guarantee. | 1. HashSet doesn’t provide any ordering guarantee. |
| 1. Treeset is backed by Red-Black Tree. | 1. HashSet is backed by hash table. |
| 1. TreeSet uses compareTo() method for comparison. | 1. HashSet uses equals() method for comparison. |
| 1. It is slower for most of the general operations e.g. add, remove and search and requires more memory. | 1. It is faster and requires less memory. |
| 1. TreeSet doesn’t allow null objects. | 1. HashSet allows one null element. |
| 1. It is internally implemented using TreeMap to store its elements. | 1. It is internally implemented using HashMap to store its elements. |

**Difference between Array and List**

|  |  |
| --- | --- |
| **Array** | **List** |
| 1. Array is a data structure generally consisting of sequential memory storing a collection of objects. | 1. List is a generic Collection interface in Java, which means that it may have multiple implementations. |
| 1. It is fixed sized i.e. after we created an array, we cannot increase or decrease the size. | 1. It is grow-able in nature as we can increase or decrease its size. |
| 1. Array can hold primitives as well as objects. | 1. Lists can hold only objects but not primitives. |
| 1. Elements in an array are iterated through for or for-each loops and are faster. | 1. It is slower than array. |
| 1. It can hold only homogeneous elements. | 1. It can hold both homogeneous as well as heterogeneous elements. |

**Difference between List and Set**

|  |  |
| --- | --- |
| **List** | **Set** |
| 1. Duplicates are allowed in list. | 1. No duplicates allowed in Sets. |
| 1. Multiple null values are allowed. | 1. Null value is allowed just once. |
| 1. List preserves insertion order and can retrieve elements using index. | 1. No insertion order preserved in Sets. |
| 1. Data structure used is array. | 1. It has underlying Map implementation. |

**Difference between NavigableSet and NavigableMap**

|  |  |
| --- | --- |
| **NavigableSet** | **NavigableMap** |
| 1. **NavigableSet**extends the SortedSet interface and as well as NavigableMap interface. | 1. NavigableMap is an extension of SortedMap which provides convenient navigation 2. methods. |
| 1. **It** represents a Set that is sorted in terms of a client. | 1. **It** represents a Map that is additionally sorted in terms of a client. |
| 1. It is basically unordered collection of distinct elements i.e. it doesn't store duplicates. | 1. It is a data structure that associates its elements with certain keys so that these elements could be obtained by that keys. |
| 1. It is used when you need an ordered Set and operations it provides. | 1. It has a few extensions to the SortedSet which makes it possible to navigate the map. |
| 1. The classes that implement this interface are TreeSet and ConcurrentSkipListSet. | 1. it also provide ways to create a Sub Map from existing Map in Java. |