**Maham Masood (16L-4072)**

**Advanced Programming (CS) Assignment 2 – Java Collections**

**1.**

|  |  |  |
| --- | --- | --- |
|  | **ArrayList** | **Vector** |
| i. | It is not synchronized i.e. multiple threads can access it at one time. | It is synchronized i.e. only one thread at a time can access its code and has an acquired lock on it. |
| ii. | If the number of elements exceeds its capacity, it increments 50% of the current size. | If number of elements exceeds its capacity, it doubles or increments 100% of its current size. |
| iii. | It has a faster performance, due to being non-synchronized. | It has a slower performance, due to being synchronized. |
| iv. | It uses Iterator for traversal. | It can be traversed using both **Enumerator and Iterator.** |

**2.**

|  |  |  |
| --- | --- | --- |
|  | **HashSet** | **SortedSet** |
| i. | Insertion order of elements is not maintained, but it allows quick determination of whether an object is already in the set. | Insertion order of elements is maintained and it is an ordered set collection. |
| ii. | It has a faster performance. Add and remove operations are carried out in O(1) time, using hash-based implementation. | It has a comparatively slower performance. Add and remove operations are carried out in O(log(n)) time. |
| iii. | It uses a hash-table data structure to store data. | It uses a red-back tree data structure, a balanced binary tree, to store data. |
| iv. | It allows null value to be inserted. | Null insertion is not allowed and exception is thrown. |

**3.**

|  |  |  |
| --- | --- | --- |
|  | **TreeSet** | **HashSet** |
| i. | Elements are stored in sorted order. | Order of elements is not maintained. |
| ii. | It used compareTo() method to compare two objects in a set and detect duplicates. | It used equals() method to compare two objects in a set and detect duplicates. |
| iii. | It does not allow null object and throws null pointer exception. | Null object insertion is allowed. |
| iv. | It is comparatively slower, and takes O(log n) time for search, insert and delete operations. | It is faster, and takes constant time for search, insert and delete operations. |

**4.**

|  |  |  |
| --- | --- | --- |
|  | **Array** | **List** |
| i. | It has a fixed size, which needs to be specified at time of declaration. | Its size is not fixed. Even if size is specified at time of declaration, it can be changed later. |
| ii. | It can contain primitive data as well as objects of a class. | It only supports object entries, not primitive data types. |
| iii. | For primitive data, the entries are at contiguous locations while for object entries, this is not necessary. | Objects are not stored at contiguous locations. |

**5.**

|  |  |  |
| --- | --- | --- |
|  | **List** | **Set** |
| i. | Duplicate values are allowed. | Duplicate values are not allowed, it contains distinct elements. |
| ii. | It allows any number of null values. | It can have one null value at most. |
| iii. | Insertion order is preserved. | Insertion order is not preserved. |

**6.**

|  |  |  |
| --- | --- | --- |
|  | **NavigableSet** | **NavigableMap** |
| i. | It is a navigable set in Java Collection Framework, and inherits from the SortedSet interface. | It is an extension of SortedMap which is used to navigate maps and provides navigation methods such as lowerKey, floorKey, ceilingKey and higherKey. |
| ii. | Methods [subSet(E, E)](https://docs.oracle.com/javase/8/docs/api/java/util/NavigableSet.html#subSet-E-E-), [headSet(E)](https://docs.oracle.com/javase/8/docs/api/java/util/NavigableSet.html#headSet-E-), and [tailSet(E)](https://docs.oracle.com/javase/8/docs/api/java/util/NavigableSet.html#tailSet-E-) are specified to return SortedSet to allow existing implementations of SortedSet to be compatibly retrofitted to implement NavigableSet. | Methods [subMap(K, K)](https://docs.oracle.com/javase/7/docs/api/java/util/NavigableMap.html#subMap(K,%20K)), [headMap(K)](https://docs.oracle.com/javase/7/docs/api/java/util/NavigableMap.html#headMap(K)), and [tailMap(K)](https://docs.oracle.com/javase/7/docs/api/java/util/NavigableMap.html#tailMap(K)) are specified to return SortedMap to allow existing implementations of SortedMap to be compatibly retrofitted to implement NavigableMap. |
| iii. | It behaves like a SortedSet and has its own navigation methods available, in addition to the sorting mechanisms of its parent set. | In addition to navigation methods, it also has the functionality of creating a Sub Map from an existing map in Java. |