**DIFFERENCES IN JAVA COLLECTIONS**

|  |  |
| --- | --- |
| **VECTOR** | **ARRAYLIST** |
| Synchronized automatically in a multi-threaded environment | Must be synchronized by user |
| Iterator or Enumeration interface used for traversal | Iterator interface used for traversal |
| When an element is added, and the array needs expansion, it is incremented by 50% | Array size is doubled when expansion in required |

|  |  |
| --- | --- |
| **HASHSET** | **TREESET** |
| O (1) i.e. constant time searching, insertion and deletion | O (log n) for these operations |
| No order maintained by default | Elements are ordered in ascending order by default |
| Null objects not allowed | Null objects are allowed |
| equals() method used for comparing objects | compareTo() used for comparison |

|  |  |
| --- | --- |
| **ARRAY** | **LIST** |
| Fixed in size | Can be expanded on run-time |
| May be two-dimensional for visualizing the problem in an easy way | No concept of multi-dimensional lists |

|  |  |
| --- | --- |
| **LIST** | **SET** |
| Duplicates allowed | Duplicates are not allowed |
| Insertion order is preserved | Insertion order not preserved |
| Elements can be accessed using their index | Since no insertion order, no index for any element |

|  |  |
| --- | --- |
| **HASHSET** | **SORTEDSET** |
| Lookup and manipulation efficiency are O (1) due to hashing | O (log n) for lookup and add, delete operations |
| Unordered by default | Elements are ordered |
| Contiguous storage | Non-contiguous storage |
| Uses a hash-table | Uses a red-black binary tree |

|  |  |
| --- | --- |
| **NAVIAGABLESET** | **NAVIGABLEMAP** |
| Extends SortedSet interface | Extends SortedMap interface |
| No key-value pair | Key-value pairs |
| Duplicates not allowed | Same value with different keys allowed |