|  |  |
| --- | --- |
| **NavigableSet** | **NavigableMap** |
| A NavigableSet extends the SortedSet interface and as well as NavigableMap interface provides methods for navigating elements over your set.  For example, it has pollFirst() method that retrieves and removes the lowest element in the set  The NavigableSet interface represents a Set that is sorted in terms of a client. A set is an unordered collection of distinct elements (i.e it doesn't store duplicates). | A NavigableMap extends the SortedMap interface which represents a sorted map. NavigableMap offers methods for obtaining the elements of the map relative to other ones, in the context of a sorted collection.  For instance, it has ceilingEntry(K obj) method that returns an entry with the lowest key that is greater or equal to the object passed as an argument. Or the lastEntry() method, that returns an entry with the greatest key, and so on.  The NavigableMap represents a Map, that is additionally sorted in terms of a client. The Map is a data structure that associates its elements with certain keys so that these elements could be obtained by that keys. |

Name: Husnain Rafiq

Roll No: 14L-5865

|  |  |
| --- | --- |
| **List** | **Set** |
| Lists generally allow duplicate objects. Lists must be ordered, and are therefore accessible by index.  Implementation classes include: Array List, LinkedList, Vector | Sets do not allow duplicate objects. Most implementations are unordered, but it is implementation specific.  Implementation classes include: HashSet (unordered), Linked HashSet (ordered),Tree Set (ordered by natural order or by provided comparator) |

|  |  |
| --- | --- |
| **Array** | **List** |
| Array is a fixed length data structure.    Array instance knows about what kind of type it can hold  While Array can contain both primitives and Objects in Java | List is a variable length Collection class.  List allows you to use Generics to ensure type-safety  you can not store primitives in Array List, it can only contain Objects. |

|  |  |
| --- | --- |
| **TreeSet** | **HashSet** |
| TreeSet is implemented using a tree structure(red-black tree in algorithm book). The elements in a set are sorted, but the add, remove, and contains methods has time complexity O(log (n))  Treeset preserved sorted order.  TreeSet is backed by TreeMap in Java. | HashSet is Implemented using a hash table. Elements are not ordered. The add, remove, and contains methods have constant time complexity O(1).  Hash set not preserved sorted order.  HashSet is backed by HashMap. |

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| ArrayList is synchronized by default  ArrayLists grow by 50% of the previous size when space is not sufficient for new element, | Vector is synchronized by default.  Vector will grow by 100% of the previous size when there is no space for new incoming element. |