**ArrayList vs Vector**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| Maintains order of insertion | Maintains order of insertion |
| Allows duplicates | Allows duplicates |
| Not Synchronized | Synchronized |
| Can only use Iterator for traversing through list | Can use Iterator as well as Enumerator to traverse |
| Size increased by 50% when list is full | Size increased by 100% when list is full (doubles the capacity) |

**HashSet vs SortedSet**

|  |  |
| --- | --- |
| **HashSet** | **SortedSet** |
| Solid class extending the AbstractSet class which implements the Set interface | Interface that extends the Set interface |
|  | Can be implemented using TreeSet |
| Has unique values | Unique values |
| Order of insertion is not preserved | Order of insertion is not preserved |
| Allows null values (only 1 because 2nd null will be a duplicate) | Does not allow null values |
|  | Values are sorted |

**TreeSet vs HashSet**

|  |  |
| --- | --- |
| **TreeSet** | **HashSet** |
| TreeSet implements the Navigable interface which extends the SortedSet interface | Solid class extending the AbstractSet class which implements the Set interface |
| Order of insertion is not preserved (saved in sorted order) | Order of insertion is not preserved |
| Has unique values | Has unique values |
| Does not allow null values | Allows null values (only 1 because 2nd null will be a duplicate) |
| Has extra utility methods inherited from NavigableSet like Floor, Ceiling etc. | No extra utility methods |

**Array vs List**

|  |  |
| --- | --- |
| **Array** | **List** |
| Basic functionality provided by Java | List is a part of collection framework in Java |
| Members are accessed using [] | Has set of methods used to access and modify data (remove(object), get(object) etc.) |
| No dynamic growth by itself | Dynamic growth by itself |
| Fixed size | Increases size automatically as soon as it fills up |
| Can contain both primitive and non-primitive data type objects | Only supports object entries and can only contain non-primitive data members (String, Integer, Double etc.) |

**List vs Set**

|  |  |
| --- | --- |
| **List** | **Set** |
| Interface | Interface |
| Can have duplicates | Distinct set of elements/ cannot have duplicates/ values are unique |
| Ordered sequence of elements (order is preserved) | Order is not preserved and different implementation classes have different rules regarding order |
| The user can access elements by their integer index | Cannot access elements by their index |
| Implemented using ArrayList or LinkedList | Different implementation classes available e.g HashSet, TreeSet etc. |

**NavigableSet vs NavigableMap**

|  |  |
| --- | --- |
| **NavigableSet** | **NavigableMap** |
| Is an interface that extends the SortedSet interface | Is an interface that extends the SortedMap interface |
|  | Key-value pairs are stored |
| No duplicates allowed | No duplicate “keys” allowed  If duplicate entered, the previous value on that “key” will be replaced by the new value entered. |
| Order according to solid class you use to implement the NavigableSet | Order according to solid class you use to implement the NavigableMap.  e.g. a TreeMap is always sorted based on its keys |
| Utility methods are given in this class to help modify data e.g. Floor(object), Ceiling(object) etc. | Utility methods are given in this class to help modify data e.g. Floor(object), Ceiling(object) etc. |