Izzah Manzoor

16L-4048

Advance Programming

Assignment 2

Submitted to: Mr. Ahmad Shahwaiz

1. **ArrayList vs Vector**

|  |  |
| --- | --- |
| ArrayList | Vector |
| ArrayList is non-synchronized which means multiple threads can work on ArrayList at the same time. | Vector is synchronized which means if one thread is working on Vector, no other thread can get a hold of it. |
| ArrayList is **not a legacy** class. It is introduced in JDK 1.2. | Vector is a **legacy** class. |
| ArrayList is fast as it is non-synchronized. | Vector is slow as it is synchronized. |
| Iterator interface is used by it to traverse the elements. | Iterator interface or Enumeration interface is used by it to traverse the elements. |
| ArrayList increments 50% of the current array size if the number of elements exceeds its capacity. | Vector increments 100% of the current array size means doubles the array size if the number of elements exceeds its capacity. |

1. **HashSet vs SortedSet**

|  |  |
| --- | --- |
| HashSet | SortedSet |
| HashSet uses a **hash-table to store data**. | SortedSet uses a **red-back tree**which is a balanced binary tree to store data. |
| HashSet is O(1) meaning it will do basic operations independent of the size of input data in a constant period of time. | The complexity of SortedSet is log(N) meaning it depend on the size of input, it will do the basic operations logarithmic. |
| HashSet is an unordered collection containing unique elements. | SortedSet is an ordered set collection where duplicates are not allowed. |

1. **TreeSet vs HashSet**

|  |  |
| --- | --- |
| HashSet | TreeSet |
| Java HashSet class is used to create a collection that uses a hash table for storage. | Java TreeSet class implements the Set interface that uses a tree for storage. |
| It inherits the AbstractSet class and implements Set interface. | It inherits AbstractSet class and implements the NavigableSet interface. |
| HashSet allows null value. | Java TreeSet class doesn't allow null element. |
| HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode | TreeSet orders the elements according to supplied Comparator. If no comparator is supplied, elements will be placed in their natural ascending order. |
| HashSet is backed by HashMap in java. | TreeSet is backed by TreeMap in java. |
| HashSet uses equals() method to compare two object in Set and for detecting duplicates. | TreeSet uses compareTo() method for same purpose. |
| HashSet is fast as compared to TreeSet. Its performance time is constant for most operations (add,remove ,contains and size) . | TreeSet is fast as compared to HashSet. Its performance time is log for most operations (add,remove ,contains and size) . |

1. **Array vs List**

|  |  |
| --- | --- |
| Array | List |
| In Java, an array is a data structure generally consisting of sequential memory storing a collection of objects. | List is an [interface](http://en.wikipedia.org/wiki/Interface_%28Java%29) in Java, which means that it may have multiple implementations.  One of these implementations is ArrayList, which is a class that implements the behavior of the List interface using arrays as the data structure. |
| Array is a fixed length data structure. You can not change length of Array once created in Java. | ArrayList is a variable length [Collection class](http://java67.blogspot.sg/2012/09/java-collection-interview-questions.html). ArrayList re-size itself when gets full depending upon capacity and load factor. |
| Array can contain both primitives and Objects in Java. | You can not store primitives in ArrayList, it can only contain Objects. |
| All kinds of Array provides length variable which denotes length of Array . | ArrayList provides size() method to calculate size of ArrayList in Java. |
| You can simply use assignment operator to store element into Array | Java provides add() method to insert element into ArrayList. |
| Its mandatory to provide size of Array while creating either directly or indirectly by initializing Array while creating it. | You can create instance of ArrayList without specifying size, Java will create Array List with default size. |
| Array can be multi-dimensional | ArrayList is always single dimensional. |

1. **Set vs List**

|  |  |
| --- | --- |
| Set | List |
| Set is an unordered collection, it doesn’t maintain any order. | List is an ordered collection it maintains the insertion order |
| All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value. | List allows duplicate values. |
| Set implementations: [HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/), [LinkedHashSet](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/), [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/) etc. | List implementations: [ArrayList](https://beginnersbook.com/2013/12/java-arraylist/), [LinkedList](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/) etc. |
| Set can have only a single null value at most. | List allows any number of null values. |
| Set interface does not have any legacy class. | List interface has one legacy class called Vector. |
| No new methods are defined inside Set interface, so we have to use Collection interface methods only with Set subclasses. | New methods are defined inside List interfce. |
| Set can be traversed only in forward direction with the help of iterator. | List can be inserted in in both forward direction and backward direction using ListIterator. |

1. **NavigableSet vs NavigableMap**

|  |  |
| --- | --- |
| NavigableSet | NavigableMap |
| The NavigableSet interface extends the SortedSet interface in which the elements are ordered either by natural ordering or using a Comparator. | The NavigableMap interface extends the SortedMap interface in which the elements are ordered either by natural ordering or by using a Comparator. |
| It contains methods to find out the closest matches for given search targets. | It contains methods to find out the closest matches for given search targets. |
| The elements of the NavigableSet interface must be mutually comparable. | All keys inserted into a sorted map must implement the Comparable interface and must be mutually comparable. |
| A NavigableSet may be accessed and traversed in either ascending or descending order. The descendingSet method returns a view of the set with the senses of all relational and directional methods inverted. | A NavigableMap may be accessed and traversed in either ascending or descending key order. The descendingMap method returns a view of the map with the senses of all relational and directional methods inverted. |