**Advanced Programming Assignment # 2**

**Name:** Muntaha Zaigham

**Roll number:** L164387

**GitHub link:**

# **Array List v/s Vector:**

|  |  |
| --- | --- |
| Array List | Vector |
| * Whenever an element is added in Array List, it increases its internal array size by 50%. | * Whenever an element is added in Vector, it doubles the size of its array. |
| * Array List is not a legacy class. | * Vector is a legacy class. |
| * Array List is not synchronized. | * Vector is synchronized. |
| * Array List uses the Iterator interface to traverse the elements. | * A Vector can use the Iterator interface or Enumeration interface to traverse the elements. |
| * Array List is fast | * Vector is slow because in a multithreading environment, it holds the other threads in runnable or non-runnable state until current thread releases the lock of the object. |

* Legacy Class:

Early version of java did not include the Collections framework. It only defined several classes and interfaces that provide methods for storing objects. When Collections framework were added in J2SE 1.2, the original classes were reengineered to support the collection interface. These classes are also known as Legacy classes.

* Synchronization:

It means that two threads cannot execute the method at the same time and the JVM takes care of enforcing that.

# **Hash Set v/s Sorted Set:**

|  |  |
| --- | --- |
| Hash Set | Sorted Set |
| * Hash Set is fast in performance of operations, as it offers constant time cost. | * Sorted Set is slow as compared to Hash set, as it offers O (lg n). |
| * Hash set doesn’t maintain the order of elements. | * Sorted Set sort the elements in ascending order by default. |
| * Hash Map allows Null. | * Sorted Set throws an exception when null is inserted. |

## Note:

Sorted Set is an interface and Tree set, Hash Set, Linked Hash Set implements it. We cannot make an object of it.

# **Working of Hash Set:**

In Hash Map, each key is unique. So, when an object of Hash Set is created, it will create an object of Hash Map. When an element is passed to Set, it is added as a key in the Hash Map in the add (Element e) method. Now, a value needs to be associated to the key. Java uses a Dummy value (new Object) which is called PRESENT in Hash Set.

In Hash Map, the put (Key k, Value V) method returns:

* Null, if the key is unique. The key will be added to the map.
* Old value of the key, if the key is duplicated

## **Methods in Hash Set:**

* **Boolean add (E e):** Used to add the specified element if it is not present, if it is present then return false.
* **void clear ():** Used to remove all the elements from set.
* **boolean contains (Object o):** Used to return true if an element is present in set.
* **boolean remove (Object o):** Used to remove the element if it is present in set.
* **Iterator iterator ():** Used to return an iterator over the element in the set.
* **boolean isEmpty ():** Used to check whether the set is empty or not. Returns true for empty and false for non-empty condition for set.
* **int size ():** Used to return the size of the set.
* **Object clone ():** Used to create a shallow copy of the set.

# **Working of Tree Set:**

Tree Map each key is unique as it internally uses Hash Map. So what we do in the Tree Set is that we pass the argument in the add (Element E) that is E as a key in the Tree Set . Now we need to associate some value to the key , so what Java developer did is to pass the Dummy  value that is ( new Object () ) which is referred by Object reference PRESENT .  
  
So , actually when we are adding a line in Tree Set like  treeset.add(3)   what java does internally is that it will put that element E here 3 as a key in the TreeMap(created during Tree Set object creation) and some dummy value that is Object's object is passed as a value to the key .  
  
Now if we see the code of the TreeMap put (Key k, Value V) method, you will find something like this  
 public V put (K key, V value) {  
//Some code  
}  
The main point to notice in above code is that put (key, value) will return  
1) Null, if key is unique and added to the map  
2) Old Value of the key, if key is duplicate

**Methods of Tree Set class:** Tree Set implements Sorted Set so it has availability of all methods in Collection, Set and Sorted Set interfaces.

* **void add(Object o):** This method will add specified element according to some sorting order in Tree Set. Duplicate entries will not get added.
* **boolean addAll(Collection c):** This method will add all elements of specified Collection to the set. Elements in Collection should be homogeneous otherwise ClassCastException will be thrown. Duplicate Entries of Collection will not be added to Tree Set.
* **void clear():** This method will remove all the elements.
* **boolean contains(Object o):** This method will return true if given element is present in Tree Set else it will return false.

# **Array v/s List:**

|  |  |
| --- | --- |
| Array | List |
| * Array are of fixed size. | * Lists are dynamic (increases size when a new element is added). |
| * Array is the basic functionality provided by java. | * List is the part of collection frameworks in java (implements collection). |
| * Array is a fixed size data structure. We need to specify size of array at the time of creation of its object. | * We don’t mention size of list initially. Even if we mention the size initially, we can enter more elements. |
| * Array can contain both primitive data types as well as objects of a class depending on the definition of the array. | * List only supports object entries, not the primitive data types. When we do arraylist.add (1); it converts the primitive int data type into an Integer object. |
| * In case of primitive datatypes, Array members are placed contiguously in memory. | * List members are not placed contiguously in memory. |
| * Arrays holds only homogeneous members. | * List holds heterogeneous members. |

## Note:

List is an interface and Array List, Linked list implements it. We cannot make an object of it.

# **List v/s Set:**

|  |  |
| --- | --- |
| List | Set |
| * List is an ordered collection it maintains the insertion order. | * Set is an unordered collection, it doesn’t maintain any order. |
| * List allows duplicates. | * Set doesn’t allow duplicate elements |
| * List is implemented by Array List, Linked List. | * Set is implemented by Hash Set, Linked Hash Set, and Tree Set. |
| * List allows any number of null values. | * Set can have only a single null value at most. |
| * List Iterator can be used to traverse a List in both the directions (forward and backward). | * List Iterator cannot be used to traverse a Set. |
| * List interface has one legacy class called Vector. | * Set interface does not have any legacy class. |

# **Navigable Map v/s Navigable Set:**

|  |  |
| --- | --- |
| Navigable Map | Navigable Set |
| * A Navigable Map extends the Sorted Map interface | * A Navigable Set extends the Sorted Set interface and as well as Navigable Map interface |
| * The Navigable Map represents a Map, is a data structure that associates its elements with certain keys so that these elements could be obtained by that keys. | * Set doesn’t allow duplicate elements. Navigable Set navigates ordered values. |
| * It stores only one object. | * It uses two objects called key and Value |

## Note:

If we want to navigate an ordered set with no duplicates, then we use Navigable set. And if we need an ordered map then we use navigable map.