

**National University of Computer and**

**Emerging Sciences**



**Advanced Programming**

**Assignment - 2**

**Instructor: Mr. Ahmad Shahwaiz**

**Presented By:**

15L-4204 Tarviha Fatima

**Differences between the following:**

1. **Array List vs Vector**

|  |  |
| --- | --- |
| **Array List** | **Vector** |
| ArrayLists are not synchronized. | Vectors are synchronized. |
| ArrayList increases its array size by 50 percent. | Vector defaults to doubling the size of its array. |
| ArrayList is not a legacy class, it is introduced in JDK 1.2. | Vector is a legacy class. |
| ArrayList uses Iterator interface to traverse the elements. | Vector uses Enumeration interface to traverse the elements. But it can use Iterator also. |
| ArrayList is fast because it is non-synchronized. | Vector is slow because it is synchronized i.e. in multithreading environment, it will hold the other threads in runnable or non-runnable state until current thread releases the lock of object. |

1. **HashSet vs SortedSet**

|  |  |
| --- | --- |
| **HashSet** | **SortedSet** |
| HashSet in Java is Set implementation in collection framework and like many others it is used to store objects. | SortedSet implementation which allows it to keep elements in the sorted order defined by either Comparable or Comparator interface. |
| HashSet is fastest. | SortedSet is bit slower because of sorting operation it needs to perform on each insertion. |
| HashSet offers constant time performance e.g. O(1) for add, contains and remove given hash function uniformly distribute elements in bucket. | SortedSet provides guaranteed O(log(n)) time for common operations like add, remove and contains. |
| HashSet does not maintain any order. | SortedSet maintains sorting order or elements. |
| HashSet is backed by an HashMap instance. | SortedSet is backed up by NavigableMap in Java and by default it uses SortedMap. |

1. **TreeSet vs HashSet**

|  |  |
| --- | --- |
| **TreeSet** | **HashSet** |
| TreeSet implementation which allows it to keep elements in the sorted order defined by either Comparable or Comparator interface. | HashSet in Java is Set implementation in collection framework and like many others it is used to store objects. |
| TreeSet is bit slower because of sorting operation it needs to perform on each insertion. | HashSet is fastest. |
| TreeSet provides guaranteed O(log(n)) time for common operations like add, remove and contains. | HashSet offers constant time performance e.g. O(1) for add, contains and remove given hash function uniformly distribute elements in bucket. |
| TreeSet maintains sorting order or elements. | HashSet does not maintain any order. |
| TreeSet is backed up by NavigableMap in Java and by default it uses SortedMap. | HashSet is backed by an HashMap instance. |

1. **Array vs List**

|  |  |
| --- | --- |
| **Array** | **List** |
| Arrays are specially optimized for arithmetic computations. | List are not. |
| Arrays are used as containers for elements of the same data type. | Lists are containers for elements having differing data types. |
| You can divide an array by a constant number, and each number in the array will be divided by that number and the result will be printed if you request it. | If you try to divide a list by 3, it can't be done, and an error will be thrown. |

1. **List vs Set**

|  |  |
| --- | --- |
| **Set** | **List** |
| A collection that contains no duplicate elements. More formally, sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element. As implied by its name, this interface models the mathematical set abstraction. | An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list. |
| Sets cannot have any duplicates. | Lists can have duplicates. |
| Set can both be ordered and unordered. | List is always ordered. |
| Sets cannot have positional access. | Lists can have positional access. |
| 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 interface. |

1. **NavigableSet vs NavigableMap**

|  |  |
| --- | --- |
| **NavigableSet** | **NavigableMap** |
| The NavigableSet interface inherits from the SortedSet interface. It behaves like a SortedSet with the exception that we have navigation methods available in addition to the sorting mechanisms of the SortedSet. | NavigableMap is an extension of SortedMap which provides convenient navigation method like lowerKey, floorKey, ceilingKey and higherKey, and along with these popular navigation method it also provide ways to create a Sub Map from existing Map in Java. |