***Java Collection frame Work***

1). What you mean by Java collection frame work?

2). Difference types of collections in java collection frame work?

***Solutions***

1). What you mean by Java collection frame work?

Java collection frame work is a set of class and interfaces in java which provides a standardized architecture for representing and manipulating collections of objects. It was introduced in java 2.

2). Difference types of collections in java collection frame work?

The framework provides a set of core interfaces that represent different types of collections such as :-

**--> Collection :**  This is the root interface for all collection types. Which defines the basic operations such as adding, removing and querying elements.

**--> List :** It is an ordered collection that allows duplicate elements. Which implement by using *ArrayList, LinkedList* .

**--> Set :** A collection that does not allow duplicate elements . Which implement by using *HashSet, TreeSet.*

**--> Map :** This is a collection of key value pair. Which implemented by using *HashMap, TreeMap.*

**--> Queue :** This is a collection use to hold elements before processing. Which implemented by using *LinkedList, PriorityQueue.*

3). What is the difference between ArrayList and LinkedList?

**🡪 ArrayList :** Which provides dynamically resizable array to store elements. When the array reaches its limit then it resize to accommodate more elements. We can access elements by using index. The scenario where it using is the program needs random access to elements and frequent iteration is required

>. **To add element :** List.add( val );

>. **To remove element :** List.remove( index );

>. **To get element :** List.get( index);

**>. To modify :** List.set( intex, element );

>. **To size:** list.size( );

>. **To know empty :** Boolean res=list.isEmpty( );

>. **To remove all:** list.clear( );

**🡪 LinkedList :** Which initially uses a doubly linked list data structure so each element(node) in contains the reference of previous and next node. The scenario where it using is the program needs the frequent insertion and deletion operation.

>. To add element : List.add( val );

>. To remove element : List.remove( index );

>. To get element : List.get( index);

4). What is the difference between HashSet and TreeSet ?

***🡪 HashSet :*** It implements Set interface and uses a hash table for storage . It does not guarantee the order of its elements and it permit null element. The basic operations on HashSet :-

> **. Add element :** set.add( value )

>. **remove element** : set.remove (value)

>. **Check for exist :** boolean exist = set.contains( value )

>. **Check size :** set.size( )

**>. Iteration :** for (string element :set) { System.out.print( element ) }

**>. Remove all :** set.clear( )

🡪 ***TreeSet :*** It also implements the Set interface but using the Red-Black tree for the data storage. It does not permit null elements Tree set maintains the elements in sorted order.

> **. Add element :** set.add( value )

>. **remove element** : set.remove (value)

>. **Check for exist :** boolean exist = set.contains( value )

>. **Check size :** set.size( )

**>. Iteration :** for (string element :set) { System.out.print( element ) } ,

Iterator<Integer> i=treeset.Iterator( );

While( i.hasNext( )){

Integer element.hasNext();

System.out.println( element); }

**>. Remove all :** set.clear( )

**>. Getting elements beteen two :** tree.subset(from , to )

**>. Getting elements greater than element :** set.headSet( from value )

**>. Getting elements lesser than element :** set.headSet( to value )

5). What is the difference between HashMap and TreeMap?

***🡪. HashMap :*** The HashMap does not guarantee any specific order of the elements. The order of the elements may change over time ,as it based on the hash code of the keys. It allows one null key and multiple null values. It initially uses an array of linked list to store key value pairs.

> **. Add element :** map.put( key , value )

> **. get element :** map.get( key )

>. **remove element** : map.remove (key)

>. **Check for exist :** boolean exist = map.contains( value )

>. **Check size :** map.size( )

**>. Iteration :** for (Map.Entity<String,Integer>entry:map.entrySet( )) {

String key= entry.getkey( );

Integer value=entry.getValue( ) ;

}

**>. Remove all :** map.clear( )

🡪. ***TreeMap :*** The TreeMap is maintains the element in the sorted order. The elements are sorted according to their keys. It does not allow null key. When we make a try it throws “NullPointerException” .It initially uses a Red-Black tree to store the key value pairs. This tree structure maintains the elements in the sorted order.

> **. Add element :** map.put( key , value )

> **. get element :** map.get( key )

>. **remove element** : map.remove (key)

>. **Check for exist :** boolean exist = map.contains( value )

>. **Check size :** map.size( )

**>. Iteration :** for (Map.Entity<String,Integer>entry:map.entrySet( )) {

String key= entry.getkey( );

Integer value=entry.getValue( ) ;

}

**>. Remove all :** map.clear( )