Friday, November 19, 2021 10:36 AM

Collection interface ----- java.util package
Methods representing common operations on data structures

Subinterface of Collection ---- List interface (index based access, duplicates allowed)

ArrayList<E> ==== array list is a resizable array (elements are on consecutive location)

APIs of ArrayList
add(ele) , add(index,ele) , remove(index) , remove(ele) ,get(index) , size() , contains(ele) , addAll(anotherlist)

Vector ---------- Similar to ArrayList

Vector is thread safe
ArrayList is not thread safe

____

Vector, Stack, LinkedList, ArrayList } all of them implement List interface , No change at API level , same at API level
Difference lies in the IMPLEMENTATION of the API !!!

User of the hierarchy benefits!!!

HW ---- Modify yesterdays assignment for LinkedList, Stack, Vector

ArrayList<Integer> al = new LinkedList<>() //ArrayList<>() } this will not work
LHS must be List<Integer> l = ……

____

TO **TRAVERSE** A LIST without index ------------ following interfaces !!!

API ----- java.util.Enumeration (old version)
hasMoreElements --- boolean is returned
if next element is present true
If end of list then false

nextElement() ---- returns **Object** that is the next element in the list

API -------Interface ----- java .util. Iterator <T> (new version)

hasNext () --- boolean is returned
if next element is present true

If end of list then false

next() ----------- returns the **Generic Type** that is next element

remove () --- removing the object from the list

Enumeration ----- read only API
Iterator ------ read + modify API } This raises ConcurrentModificationException if the list whose iterator is created is modified else where

---

HW -----------Try out the Iterator and Enumeration example done in class (play with code)

---

java.util. Collections ------------------------ Utiltiy class !!!

API s of Collections utility class ----- sort, shuffle , reverse , max ,min ,………

Comparable interface = NATURAL ORDERING !!!
int compareTo(T parameter) ----abstract

It should return +ve if this > parameter
-ve if this < parameter
0 if this == parameter

OTHER basis of ORDERING ---- use java.util. Comparator interface

Int compare(T parameter1 , T parameter 2)
It should return +ve if parameter 1 > parameter 2
-ve if parameter1 < parameter2
0 if parameter1 == parameter2

HW ---- Write a class Invoice
MyDate3 dateOfInvoice
double amount
String invoiceGivenBy
String invoiceGivenTo

Create a List of Invoices in User
Show a menu to user
Switch case

1. Show invoice sorted by dateOfInvoice (default/natural ordering) --- show using Iterator
2. Show sorted by amount in descending order ---show for normal for loop
3. Sorted on invoiceGivenBy --- show using for(Invoice v : invoices)
4. Sorted on invoiceGivenTo --- show by Enumeration
5. Show the Invoice details of invoice with max amount (Collections.max)
6. Show the Invoice details of invoice with latest date (Collections.max)
7. quit

Today------------------------(FRIDAY)

    2.30 to 5 pm   }} LAB

    5.30 pm to 7.30 pm }} Core Java lecture

Tomorrow (SATURDAY)

  8am to 10 am }} Core Java lecture

  10.30am to 1.30pm }} Core Java lecture

    Afternoon LAB

____