***Dt : 11/11/2022***

***define Comparable<T>?***

***=>Comparable<T> is an interface from java.lang package and specify***

***Sorting process.***

***structure of Comparable<T>:***

***public interface java.lang.Comparable<T>***

***{***

***public abstract int compareTo(T);***

***}***

***Note:***

***=>To perform sorting process the classes must be implemented from***

***"java.lang.Comparable<T>" interface.***

***=>WrapperClasses from Lib are implemented from "java.lang.Comparable"***

***interface.***

***=>String classesfrom Lib are implemented from "java.lang.Comparable"***

***interface.***

***=>To perform sorting process on User defined classes,the classes must be***

***implemented from java.lang.Comparable<T> and we must specify the sorting***

***support-logic in compareTo(T) method.***

***=================================================================***

***\*imp***

***2.List<E>:***

***=>List<E> organizes elements based on index values and can hold duplicate***

***elements.***

***=>The following are some important methods of List<E>:***

***public abstract int size();***

***public abstract boolean isEmpty();***

***public abstract boolean contains(java.lang.Object);***

***public abstract boolean add(E);***

***public abstract boolean remove(java.lang.Object);***

***public abstract boolean containsAll(java.util.Collection<?>);***

***public abstract boolean addAll(java.util.Collection<? extends E>);***

***public abstract boolean addAll(int, java.util.Collection<? extends E>);***

***public abstract boolean removeAll(java.util.Collection<?>);***

***public abstract boolean retainAll(java.util.Collection<?>);***

***public default void replaceAll(java.util.function.UnaryOperator<E>);***

***public abstract void clear();***

***public abstract E get(int);***

***public abstract E set(int, E);***

***public abstract void add(int, E);***

***public abstract E remove(int);***

***public abstract int indexOf(java.lang.Object);***

***public abstract int lastIndexOf(java.lang.Object);***

***public abstract java.util.List<E> subList(int, int);***

***public default void sort(java.util.Comparator<? super E>);***

***public abstract java.util.Iterator<E> iterator();***

***public abstract java.util.ListIterator<E> listIterator();***

***public default java.util.Spliterator<E> spliterator();***

***public abstract java.lang.Object[] toArray();***

***public abstract <T> T[] toArray(T[]);***

***-----------------------------------------------------------------***

***=>The following are the implementation classes of List<E>:***

***(a)ArrayList<E>***

***(b)LinkedList<E>***

***(c)Vector<E>***

***(a)ArrayList<E>:***

***=>ArrayList<E> organizes elements in sequence and which is NonSynchronized***

***class.***

***(b)LinkedList<E>:***

***=>LinkedList<E> organizes elements in NonSequence and which is also***

***NonSynchronized class.***

***(c)Vector<E>:***

***=>Vector<E> organizes elements in Sequence and which is synchronized***

***class.***

***===================================================================***

***Ex-program : DemoList1.java***

***package maccess;***

***import java.util.\*;***

***public class DemoList1 {***

***@SuppressWarnings("removal")***

***public static void main(String[] args) {***

***Scanner s = new Scanner(System.in);***

***String c = null;***

***List<Integer> ob = null;***

***try(s;){***

***try {***

***while(true) {***

***System.out.println("====Choice====");***

***System.out.println***

***("1.ArrayList\n2.LinkedList\n3.Vector\n4.exit");***

***System.out.println("Enter the choice:");***

***switch(s.nextInt())***

***{***

***case 1:***

***ob = new ArrayList<Integer>();***

***c = "ArrayList<E>";***

***break;***

***case 2:***

***ob = new LinkedList<Integer>();***

***c = "LinkedList<E>";***

***break;***

***case 3:***

***ob = new Vector<Integer>();***

***c = "Vector<E>";***

***break;***

***case 4:***

***System.out.println("List operation Stopped...");***

***System.exit(0);***

***default : System.out.println("Invalid choice");***

***continue; //skip the below lines from the Iteration***

***}//end of switch***

***System.out.println("\*\*\*\*perform operations ob "+c+"\*\*\*\*");***

***xyz:***

***while(true) {***

***System.out.println("====Choice====");***

***System.out.println("1.add\n2.remove\n3.add(index,E)\n4.remove(index)\n5.set(index,E)\n6.get(index)\n7.exit");***

***System.out.println("Enter the Choice:");***

***switch(s.nextInt())***

***{***

***case 1:***

***System.out.println("Enter the ele:");***

***ob.add(new ~~Integer~~(s.nextInt()));***

***System.out.println(ob.toString());***

***break;***

***case 2:***

***if(ob.isEmpty()) {***

***System.out.println("List<E> is empty...");***

***}else {***

***System.out.println("Enter the ele to be removed:");***

***Integer ele = new ~~Integer~~(s.nextInt());***

***if(ob.remove(ele)) {***

***System.out.println("Ele removed Successfully...");***

***System.out.println(ob.toString());***

***}else {***

***System.out.println("Ele not available...");***

***}***

***}***

***break;***

***case 3:***

***if(ob.isEmpty()) {***

***System.out.println("List<E> is empty...");***

***}else {***

***System.out.println("Enter the index:");***

***int index1 = s.nextInt();***

***if(index1>=0 && index1<ob.size()) {***

***System.out.println("Enter the ele to be added:");***

***ob.add(index1, new ~~Integer~~(s.nextInt()));***

***System.out.println(ob.toString());***

***}else {***

***System.out.println("Invalid index value..");***

***}***

***}***

***break;***

***case 4:***

***if(ob.isEmpty()) {***

***System.out.println("List<E> is empty...");***

***}else {***

***System.out.println("Enter the index:");***

***int index2 = s.nextInt();***

***if(index2>=0 && index2<ob.size()) {***

***ob.remove(index2);***

***System.out.println(ob.toString());***

***}else {***

***System.out.println("Invalid index value..");***

***}***

***}***

***break;***

***case 5:***

***if(ob.isEmpty()) {***

***System.out.println("List<E> is empty...");***

***}else {***

***System.out.println("Enter the index:");***

***int index3 = s.nextInt();***

***if(index3>=0 && index3<ob.size()) {***

***System.out.println("Enter the ele to be setted:");***

***ob.set(index3,new ~~Integer~~(s.nextInt()));***

***System.out.println(ob.toString());***

***}else {***

***System.out.println("Invalid index value..");***

***}***

***}***

***break;***

***case 6:***

***if(ob.isEmpty()) {***

***System.out.println("List<E> is empty...");***

***}else {***

***System.out.println("Enter the index:");***

***int index4 = s.nextInt();***

***if(index4>=0 && index4<ob.size()) {***

***Integer ob2 = (Integer)ob.get(index4);***

***System.out.println("Ele at "+index4+" is "+ob2);***

***System.out.println(ob.toString());***

***}else {***

***System.out.println("Invalid index value..");***

***}***

***}***

***break;***

***case 7:***

***System.out.println("Operations stopped on "+c);***

***break xyz;***

***default:***

***System.out.println("Invalid choice...");***

***}//end of switch***

***}//end of loop***

***}//end of loop***

***}catch(Exception e) {e.printStackTrace();}***

***}//end of try***

***}***

***}***

***=====================================================================***

***Assignment:***

***wap to perform operations on BookDetails using List<E>?***

***===================================================================***

***Note:***

***=>ArrayList<E> organizes elements in sequence.***

***Limitation of ArrayList<E>:***

***=>when we perform add() operation on ArrayList<E> the elements are moved***

***backward and,when we remove() operation elements moved forward.***

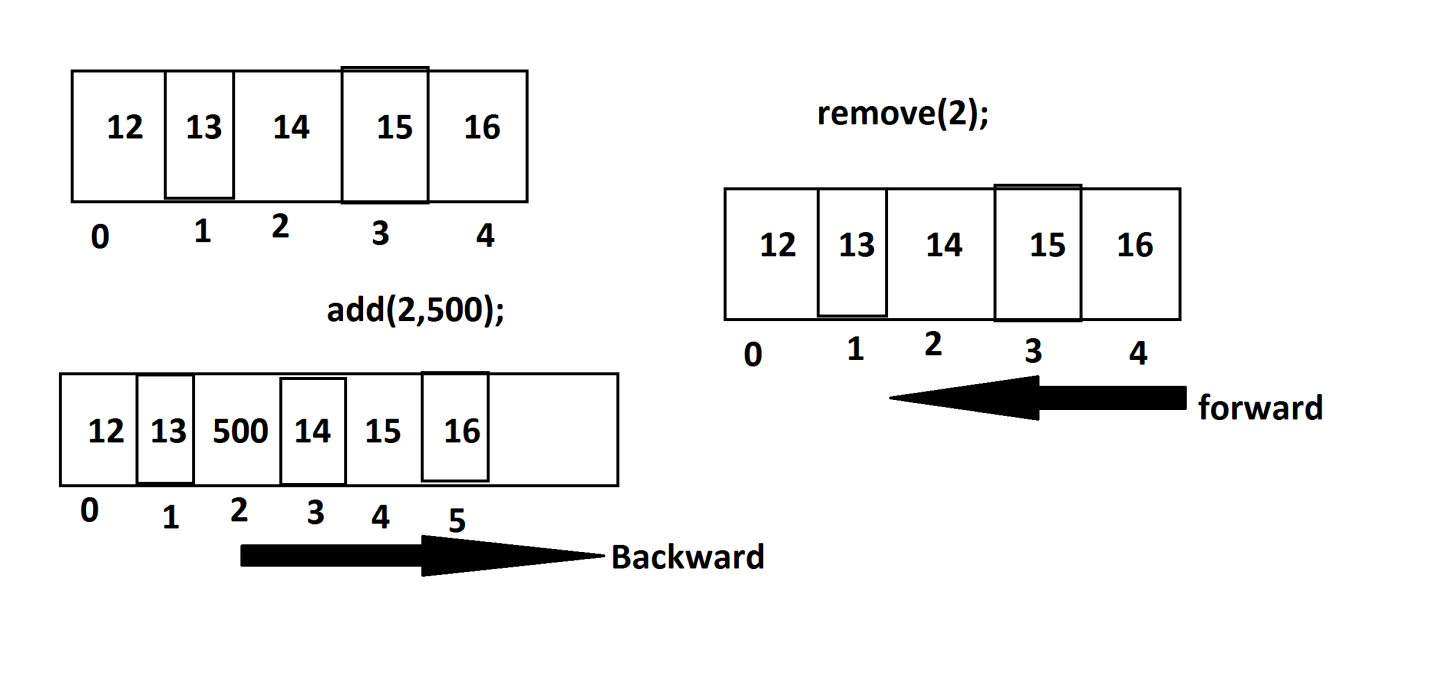
***=>In this process the execution time is wasted in moving the elements***

***forward and backward,which degrades the performance of an application***

***Note:***

***=>In realtime ArrayList<E> is used in the applications where we have***

***less number of add() and remove() operations.***

******

***=============================================================***