Page No.:

Ex:-

```
ArrayList l = new ArrayList();
List ll = Collections.synchronizedList(l);
```

ll ↑ synchronized. l ↳ non-Synchronized

→ Similarly, we can get synchronized version of set, Map objects by using the following methods of collections class.

```
public static Set synchronizedSet(Set s)
public static Map synchronizedMap(Map m).
```

⇒ **LinkedList :→**

→ The underlined data structure is doubly linked list.
→ Insertion order is preserved.
→ Duplicates objects are allowed.
→ Hetrogenious objects allowed.
→ null insertion is possible.
→ Linked list implements Serializable and cloneable interfaces but not Random Access.
→ Linked list is best choice if our frequent opn is insertion or deletion in the middle. and worst choice if our frequent opn is retrieval (get) opn.

⇒ **constructors :→**

(1) `LinkedList l = new LinkedList();`
creates an empty linked list object

(2) `LinkedList l = new LinkedList(Collection c);`
creates an equivalent linked list object for the given collection.

LinkedList class specific methods:-

Usually, we can use LinkedList to develop stacks and queues. To provide support for this requirement LinkedList class defines the following specific methods:-

```
void addFirst(Object o)
void addLast(Object o)
Object getFirst();
Object getLast();
Object removeFirst();
Object removeLast();
```

Ex:-

```
import java.util.*;
public class Test {
    p s v m (String[] args) {
LinkedList l=new LinkedList();
l.add("durga");
l.add(30);
l.add(null);
l.add("durga");
S.O.p(l); → [durga, 30, null, durga]
l.set(0,"software"); → [software, 30, null, durga]
l.add(0,"venky"); → [venky, software, 30, null, durga]
l.removeLast(); → [venky, software, 30, null]
l.addFirst("ccc");
S.O.p(l); → [ccc, venky, software, 30, null]
```

→ Diff b/w ArrayList & LinkedList

ArrayList	LinkedList
i) Retrieval opn is faster.	i) opn is insertion/deletion
ii) Arraylist is the best choice if our frequent opn is Retrieval opn.	ii) LinkedList is the best choice if our frequent opn is insertion/deletion ~~opn~~ in the middle.