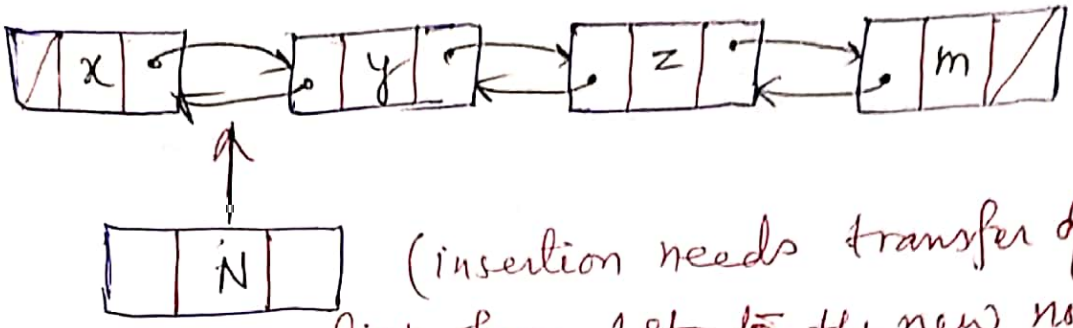
30-5-2020

Today we will try to learn about LinkedList & Vector class.



(insertion needs transfer of link from 1st to the new node and new node link transfer to the second — we know from data structure concept.

Some important conclusions of LinkedList are —

① The underlying data structure is Doubly Linked List.

② Insertion Order is preserved.

③ Duplicates are allowed.

④ Heterogeneous objects are allowed.

⑤ Null insertion is possible.

⑥ Linked List implements — Serializable, Cloneable but not RandomAccess interface.

LinkedList is the best choice whenever —
- our frequent operation is Insertion or deletion in the middle.

LinkedList is the worst choice whenever —
- our frequent operation is retrieval operation.

Methods of LinkedList class :- only 6 methods :-

void addFirst(Object ob); void addLast(Object ob);

Object getFirst(); Object getLast();

Object removeFirst(); Object removeLast();

Usually we can use LinkedList to implement Stacks and Queues. In order to support this requirement LinkedList class defines the above mentioned 6 methods:- addFirst(); addLast(); getFirst(); getLast(); removeFirst(); and removeLast();

Now, Constructors:-

① LinkedList ll = new LinkedList();
Creates an empty LinkedList object.

② LinkedList ll = new LinkedList(Collection c);
Creates an equivalent LinkedList object for the given Collection.

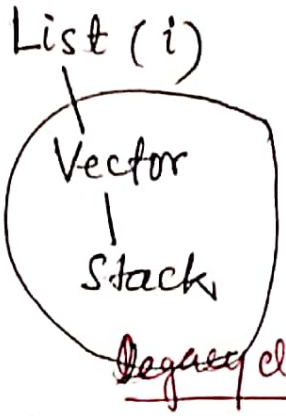
Example:- LLdemo.java

```
import java.util.*;
class LLdemo
{
    public static void main(String args[])
    {
        LinkedList ll = new LinkedList();
        ll.add("Shershah");
        ll.add("1975");
        ll.add(303);  ll.add(null);
        ll.add("Shershah");
        ll.set(0, "BCA"); //replace Shershah with BCA
        ll.add(0, "Department");
        ll.removeLast();
        ll.addFirst("B+");
        System.out.println(ll);
    }
}
```

o/p will be:-

[B+, Department, BCA, 1975, 303, null]

Vector — As we know earlier, Vector and Stack are legacy classes of List interface.



- ★ The underlying Data Structure for the Vector is resizable array or growable array.
- ★ We can store duplicate objects
- ★ We have insertion order preserved.
- ★ We can insert null value.
- ★ Heterogeneous objects are allowed to insert.
- ★ Vector class implements Serializable, Cloneable and RandomAccess interfaces.
- ★ Most of the methods present in Vector class are synchronized. Hence Vector object is Thread-Safe.
- ★ Vector is the best choice when frequent operation is retrieval.

All above points except synchronized (thread-safe) property of Vector class is same as LinkedList.

Vector specific methods for adding objects:—

add (Object o) ——— From Collection → List (i)
add (int index, Object ob) — From List (i)
addElement (Object ob) — From Vector (class)

Vector Specific methods for Removing Objects:—

remove (Object ob) ——— from Collection (i)
removeElement (Object ob) — from Vector (c)
remove (int index) ——— from List (i)
removeElementAt (int index) — from Vector (c)
clear () ——— from Collection (i)
removeAllElements () ——— from Vector (c)

Vector Specific methods for retrieve elements:—

Object get (int index) ——— from Collection
Object elementAt (int index) — from Vector
Object firstElement () ——— from Vector
Object lastElement () ——— from Vector

Other methods :—

int size ();
int capacity ();
Enumeration elements ();

Vector class Constructors:-

① Vector vobject = new Vector();
empty Vector object with capacity (10) created
new_capacity = 2 * currentCapacity

② Vector vob2 = new Vector(int initialcapacity);

③ Vector vob3 = new Vector(int initialcapacity, int incrementalcapacity);

④ Vector vob4 = new Vector(Collection c);

→ Creates an empty vector object - "vobject" with default initial capacity 10, Once vector reaches its maximum capacity i.e:,10, a new Vector object will be created with new capacity = 2 * current capacity.

→ Creates an empty vector object with specified initial capacity.

→ Creates a Vector object with specified initial capacity and also specifies incremental capacity.

→ Creates an equivalent Vector object for the given Collection.

Let us see with small example:-

Example:- Vectordemo.java

```
import java.util.*;
class Vectordemo {
    public static void main(String args[])
    {
        Vector v = new Vector();
        System.out.println("Capacity :" + v.capacity());
for(int i= 5 ; i <=50 ; i*=5)
  {
    v.addElement(i);
  }
System.out.println("Capacity Now:" + v.capacity());
  v.addElement("A");
System.out.println("Capacity Now:" + v.capacity());
System.out.println(v);
  }
}
```

output will be:-

Capacity : 10
Capacity Now : 10
Capacity Now : 20
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, A]

you can also change:-

```
Vector v = new Vector(10, 2);
```

```
Vector v = new Vector(15);
```

and realize the outputs.

Capacity : 10
" Now : 10
Capacity Now : 12
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, A]

Capacity : 15
Capacity Now : 15
Capacity Now : 15
[5, 10, 15, 20, ... 50, A]