Dt : 14/11/2022

(b)LinkedList<E>:

=>LinkedList<E> organizes elements in NonSequence and which is also NonSynchronized class.

=>The elements in LinkedList<E> are available in the form of "nodes".

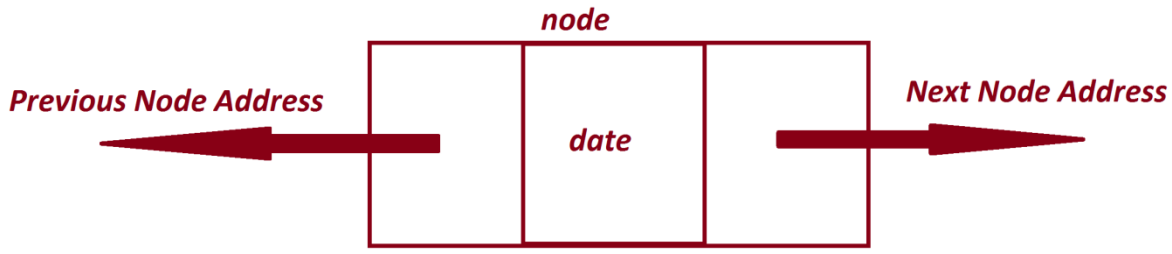
=>The LinkedList<E> node internally divided into the following partitions:

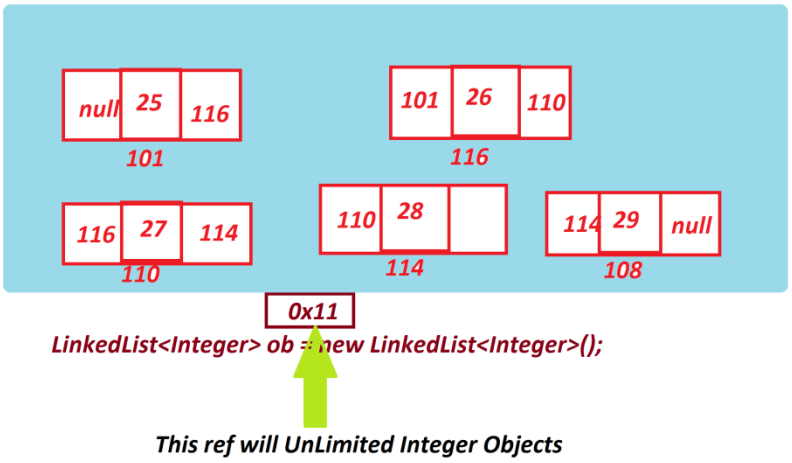
(i)Previous Node Address

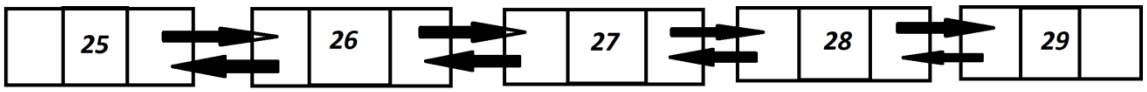
(ii)Data

(iii)Next Node Address

Diagram:







Ex : DemoList2.java

```
package maccess;
import java.util.*;
public class DemoList2 {
	@SuppressWarnings("removal")
	public static void main(String[] args) {
		LinkedList<Integer> ob = new LinkedList<Integer>();
		for(int i=25;i<=29;i++)
		{
		  ob.add(new Integer(i));
		}//end of loop
		System.out.println(ob.toString());
	}
}
```

==========

***imp**

(c)Vector<E>:

=>Vector<E> organizes elements in Sequence and which is Synchronized class.

(Thread-Safe class)

=>Vector<E> is known as Legacy class,which means replacement not available for

Vector<E> class.

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

public synchronized E elementAt(int);

public synchronized E firstElement();

public synchronized E lastElement();

public synchronized void setElementAt(E, int);

public synchronized void removeElementAt(int);

public synchronized void insertElementAt(E, int);

public synchronized void addElement(E);

public synchronized boolean removeElement(java.lang.Object);

public synchronized void removeAllElements();

Ex : DemoList3.java

```
package maccess;
import java.util.*;
public class DemoList3 {
    @SuppressWarnings("removal")
    public static void main(String[] args) {
        Vector<Integer> v = new Vector<Integer>();
        for(int i=1;i<=5;i++)
        {
          v.addElement(new Integer(i));
        }//end of loop
        System.out.println(v.toString());
        System.out.println("ele at index 2:"+v.elementAt(2));
        System.out.println("First Element : "+v.firstElement());
        System.out.println("Last Element : "+v.lastElement());
```

```
            v.setElementAt(new Integer(400), 2);
            System.out.println(v.toString());
            v.removeElementAt(2);
            System.out.println(v.toString());
            v.insertElementAt(new Integer(13), 2);
            System.out.println(v.toString());
            v.removeElement(new Integer(13));
            System.out.println(v.toString());
            System.out.println("size:"+v.size());
            v.removeAllElements();
            System.out.println("size:"+v.size());

        }
}
```

o/p:

[1, 2, 3, 4, 5]

ele at index 2:3

First Element : 1

Last Element : 5

[1, 2, 400, 4, 5]

[1, 2, 4, 5]

[1, 2, 13, 4, 5]

[1, 2, 4, 5]

size:4

size:0

==========

Note:

=>In realtime Vector<E> will hold multiple pre-initialized Database connections

in Connection Pooling Concept.

==========

faq:

define Stack<E>?

=>Stack<E> is a ChildClass of Vector<E> and which organizes elements based on the algorithm First-In-Last-Out or Last-In-First-Out

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

public E push(E);

public synchronized E pop();

public synchronized E peek();

public boolean empty();

public synchronized int search(java.lang.Object);

Ex : DemoStack.java

```
package maccess;
import java.util.*;
public class DemoStack {
	@SuppressWarnings("removal")
	public static void main(String[] args) {
	   Stack<Integer> ob = new Stack<Integer>();
	   Scanner s = new Scanner(System.in);
	   try(s;){
	     try {
	          while(true) {
	          System.out.println("*****Choice*****");

System.out.println("1.push(E)\n2.pop()\n3.peek()\n4.search()\n5.
exit");
	          System.out.println("Enter the Choice:");
	          switch(s.nextInt()) {
	          case 1:
	               System.out.println("Enter the ele:");
	               ob.push(new Integer(s.nextInt()));
	               System.out.println(ob.toString());
	               break;
	          case 2:
```

```
                        if(ob.empty()) {
                              System.out.println("Stack is
empty...");
                        }else {
                              ob.pop();
                              System.out.println("Ele deleted
from top of stack");
                              System.out.println(ob.toString());
                        }
                        break;
                  case 3:
                        if(ob.empty()) {
                              System.out.println("Stack is
empty...");
                        }else {
                              System.out.println("peek
ele:"+ob.peek());
                              System.out.println(ob.toString());
                        }
                        break;
                  case 4:
                        if(ob.empty()) {
                              System.out.println("Stack is
empty...");
                        }else {
                              System.out.println("Enter the ele
to searched:");
                              Integer ele = new
Integer(s.nextInt());
                              int pos = ob.search(ele);
                              if(pos>0) {
                              System.out.println("Ele found at
position :"+pos);
                              }else {
                                    System.out.println("Ele not
found...");
                              }
                        }
                        break;
                  case 5:
                        System.out.println("Stack operation
Stopped...");
                        System.exit(0);
                  default:
                        System.out.println("Invalid
Choice...");
                  }//end of switch
```

```
                }//end of loop
            }catch(Exception e) {e.printStackTrace();}
        }//end of try
    }
}
```

o/P:

******Choice******

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

4

Stack is empty...

******Choice******

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

1

Enter the ele:

11

[11]

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

1

Enter the ele:

12

[11, 12]

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

1

Enter the ele:

13

[11, 12, 13]

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

1

Enter the ele:

14

[11, 12, 13, 14]

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

1

Enter the ele:

15

[11, 12, 13, 14, 15]

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

4

Enter the ele to searched:

11

Ele found at position :5

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

4

Enter the ele to searched:

15

Ele found at position :1

*****Choice*****

1.push(E)

2.pop()

3.peek()

4.search()

5.exit

Enter the Choice:

5

Stack operation Stopped...

==========

Note:

=>search() method on Stack<E> searches the element from top-of-stack to

Bottom-of-Stack and display the position of an element.

==========