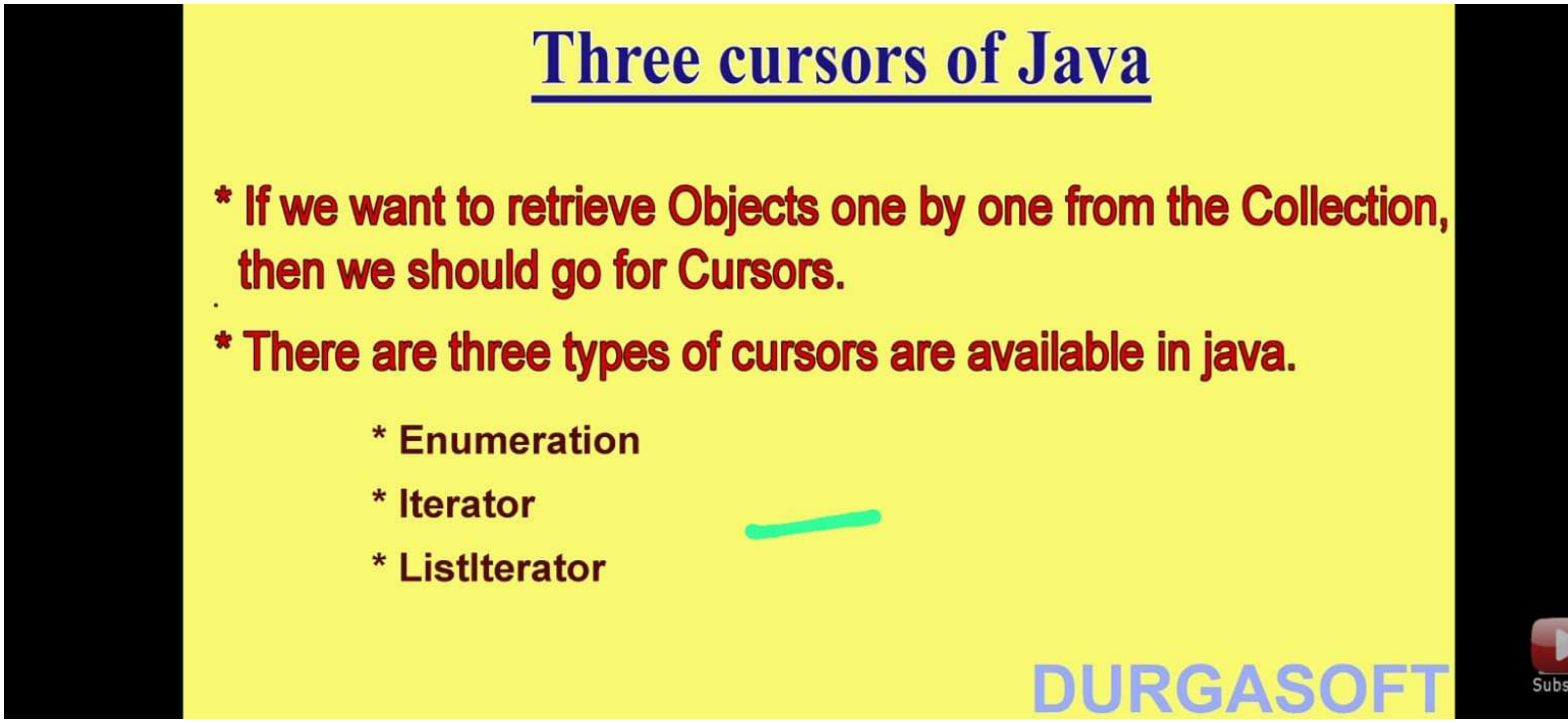
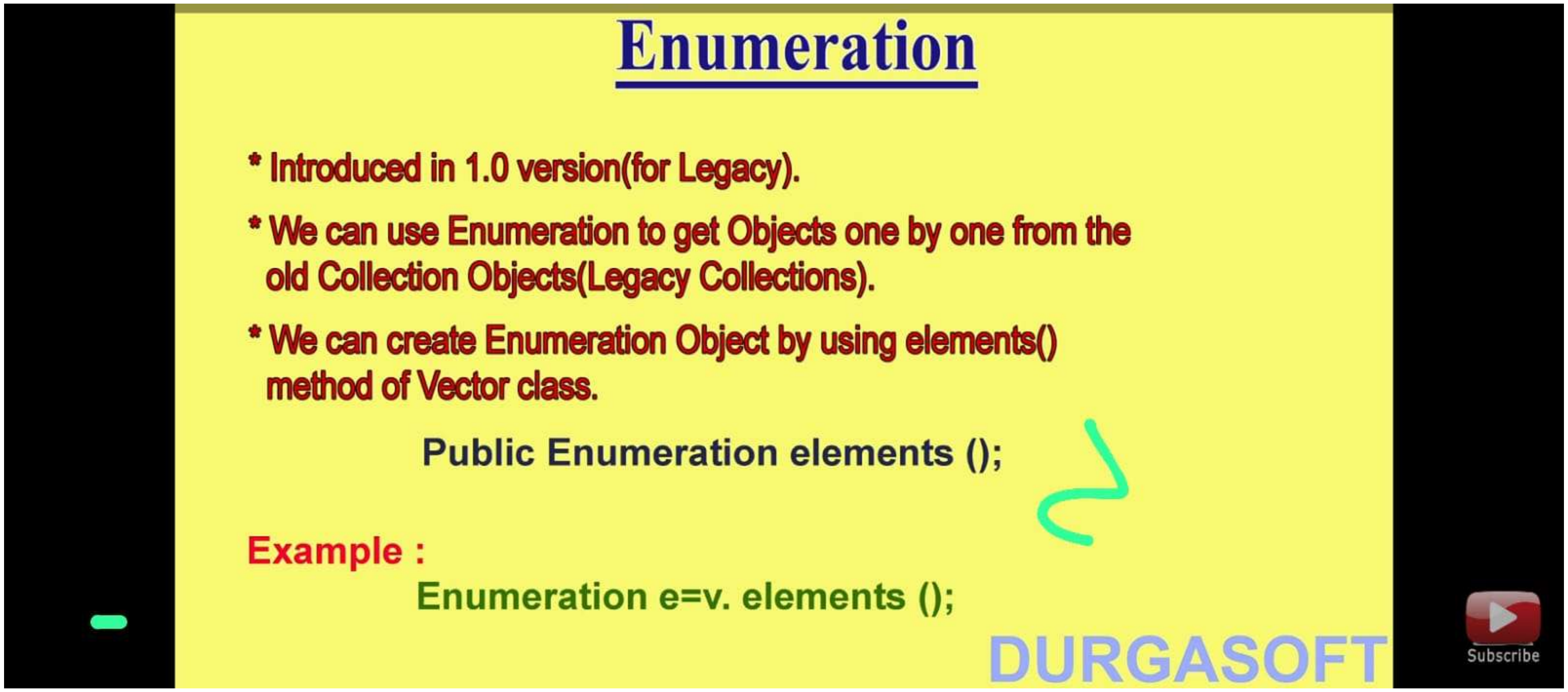
4CursorsInJava



Three cursors of Java

* If we want to retrieve Objects one by one from the Collection, then we should go for Cursors.
* There are three types of cursors are available in java.
    * Enumeration
    * Iterator
    * ListIterator

DURGASOFT



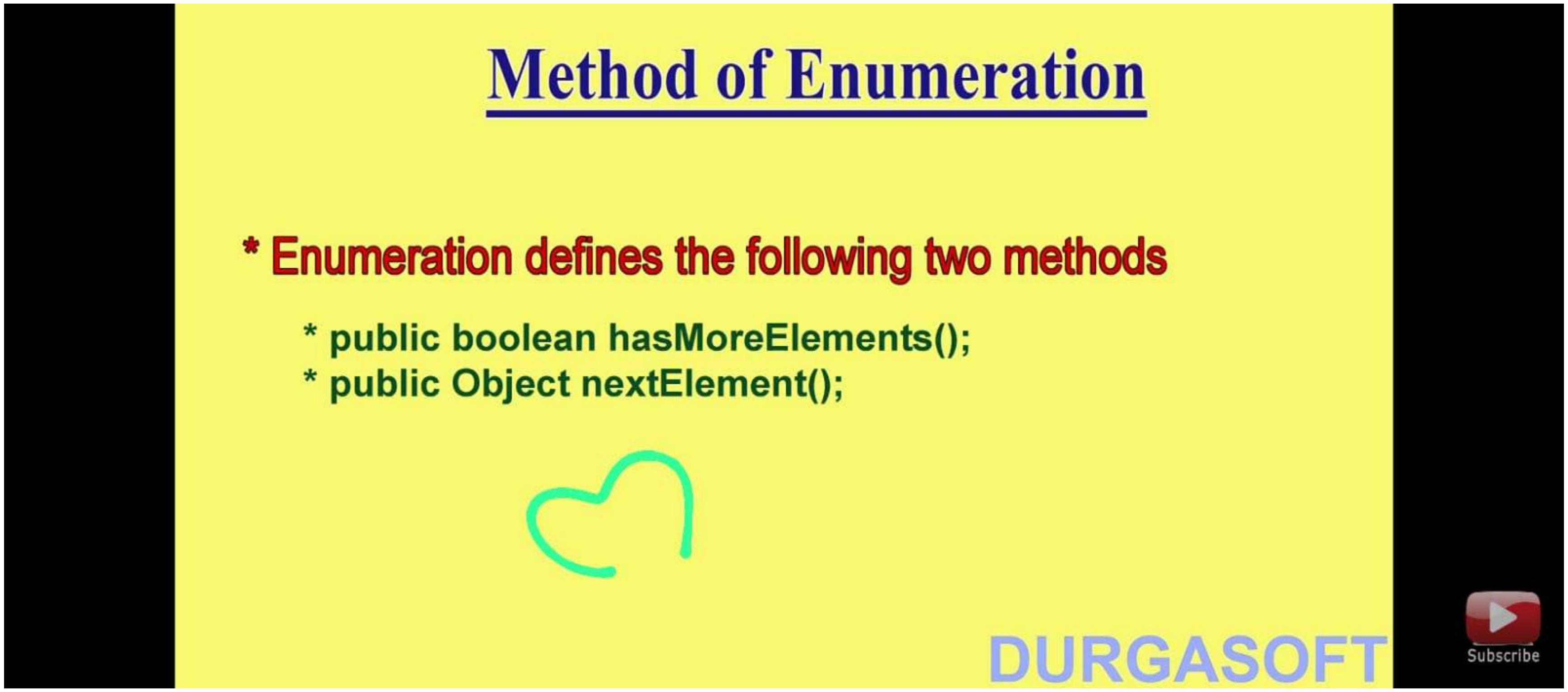
Enumeration

* Introduced in 1.0 version(for Legacy).
* We can use Enumeration to get Objects one by one from the old Collection Objects(Legacy Collections).
* We can create Enumeration Object by using elements() method of Vector class.

```
Public Enumeration elements ();
```

Example :

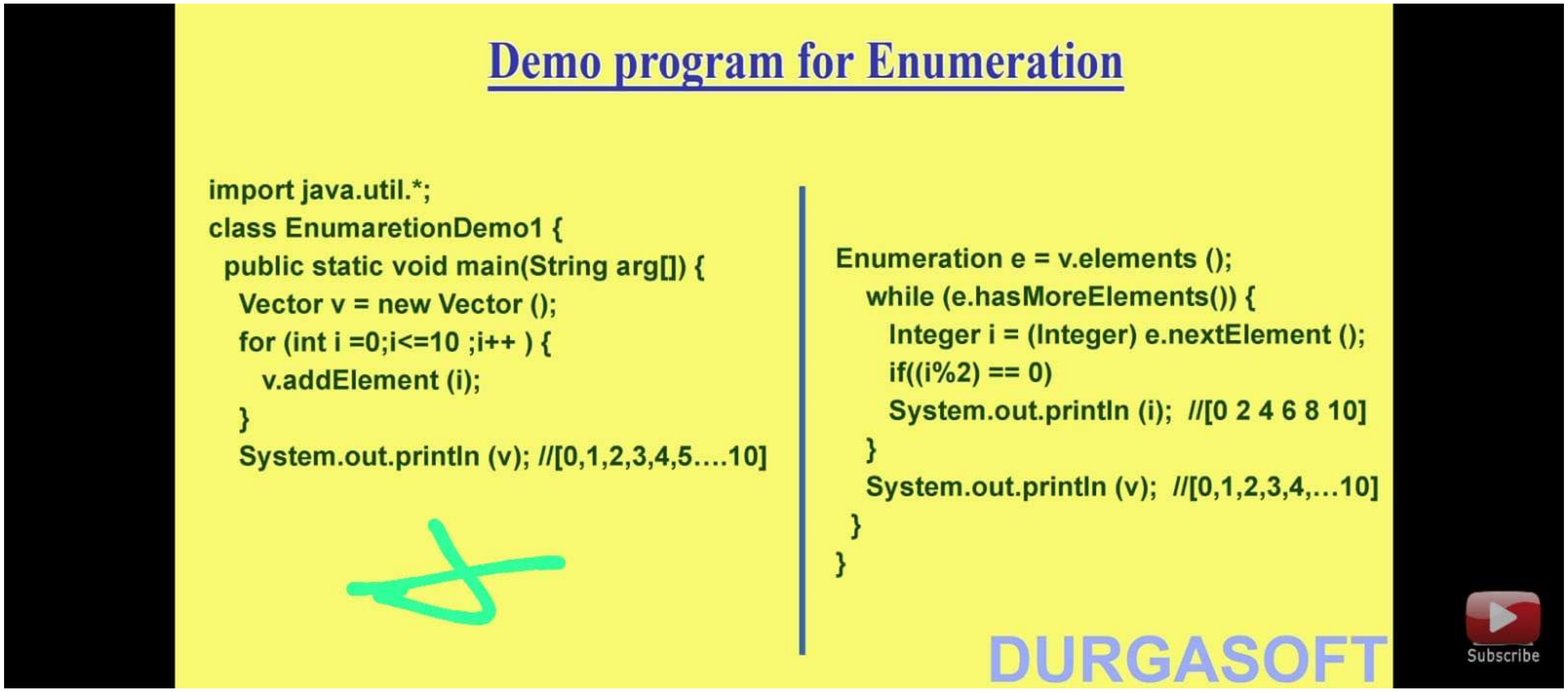
```
Enumeration e=v. elements ();
```



Method of Enumeration

* Enumeration defines the following two methods

    * public boolean hasMoreElements();
    * public Object nextElement();

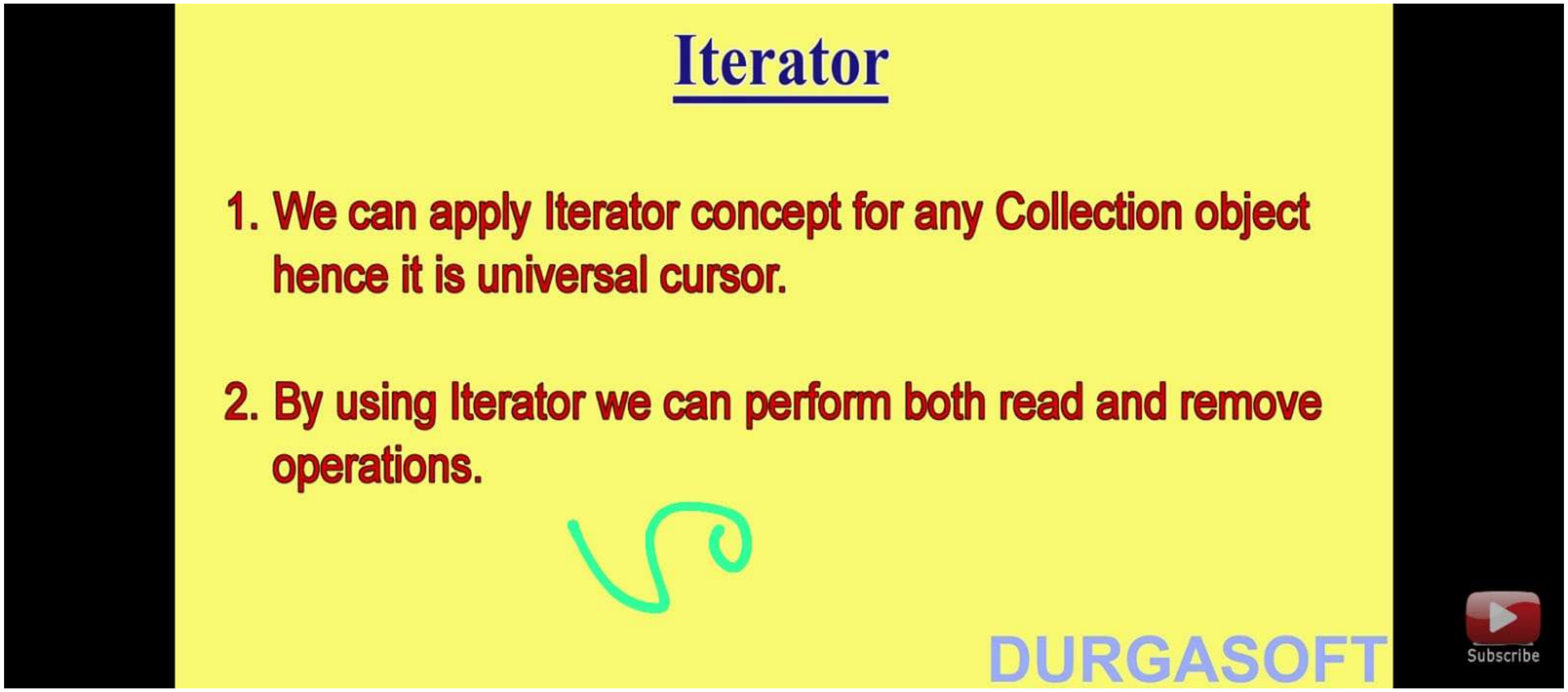


Demo program for Enumeration

```
import java.util.*;
class EnumaretionDemo1 {
 public static void main(String arg[]) {
  Vector v = new Vector ();
  for (int i =0;i<=10 ;i++ ) {
    v.addElement (i);
  }
  System.out.println (v); //[0,1,2,3,4,5….10]

Enumeration e = v.elements ();
   while (e.hasMoreElements()) {
     Integer i = (Integer) e.nextElement ();
     if((i%2) == 0)
     System.out.println (i);  //[0 2 4 6 8 10]
   }
   System.out.println (v);  //[0,1,2,3,4,…10]
  }
}
```

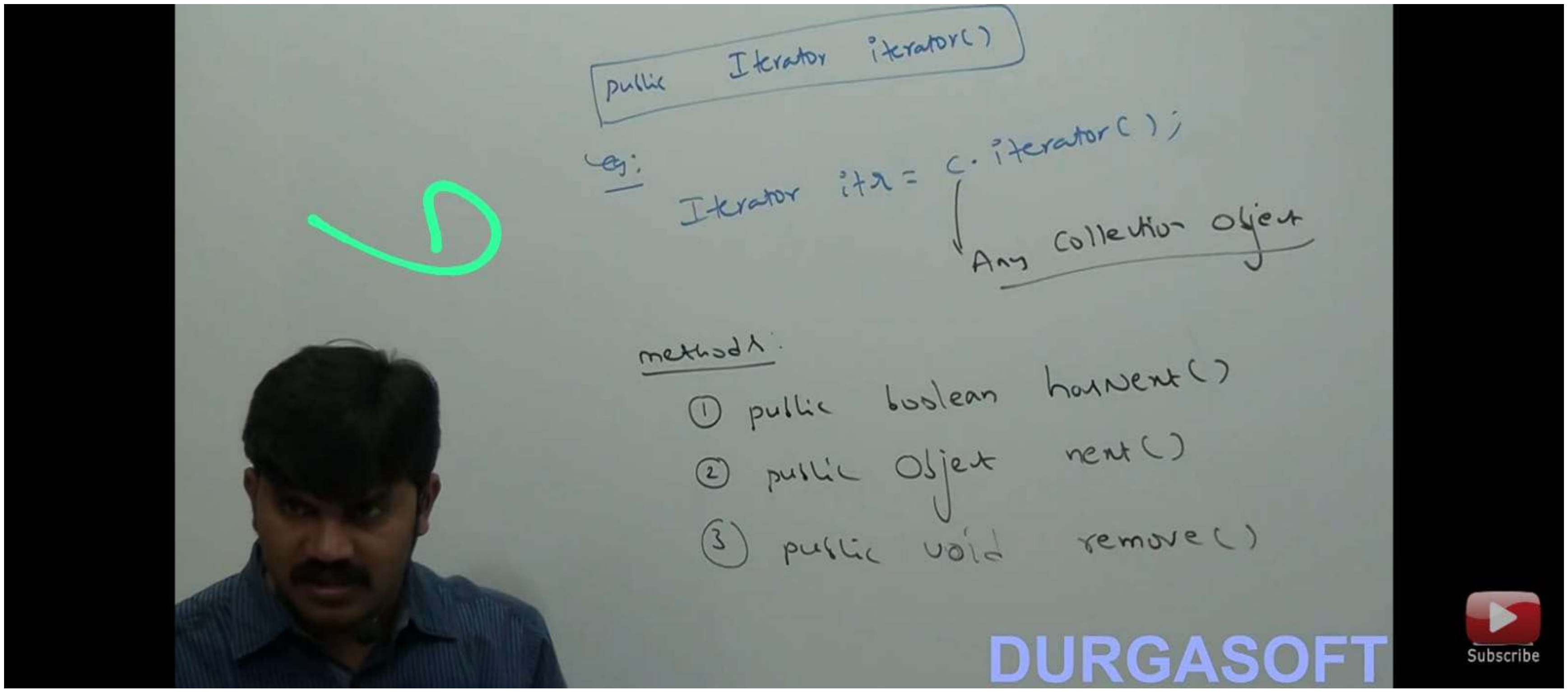
4



Iterator

1. We can apply Iterator concept for any Collection object hence it is universal cursor.

2. By using Iterator we can perform both read and remove operations.



```
public Iterator iterator()
```

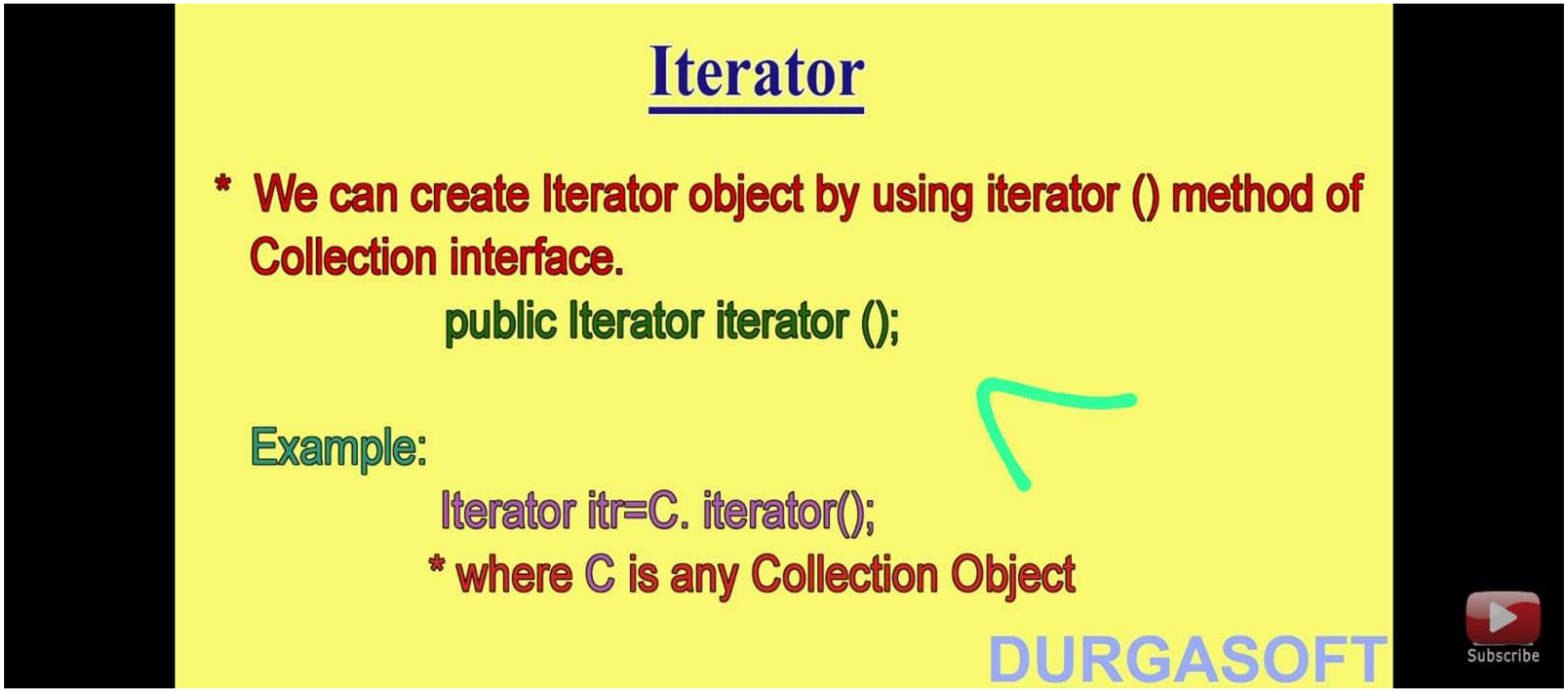
eg:

```
Iterator itr = c.iterator();
```

c → Any Collection Object

methods:

1. public boolean hasNext()
2. public Object next()
3. public void remove()



Iterator

* We can create Iterator object by using iterator () method of Collection interface.

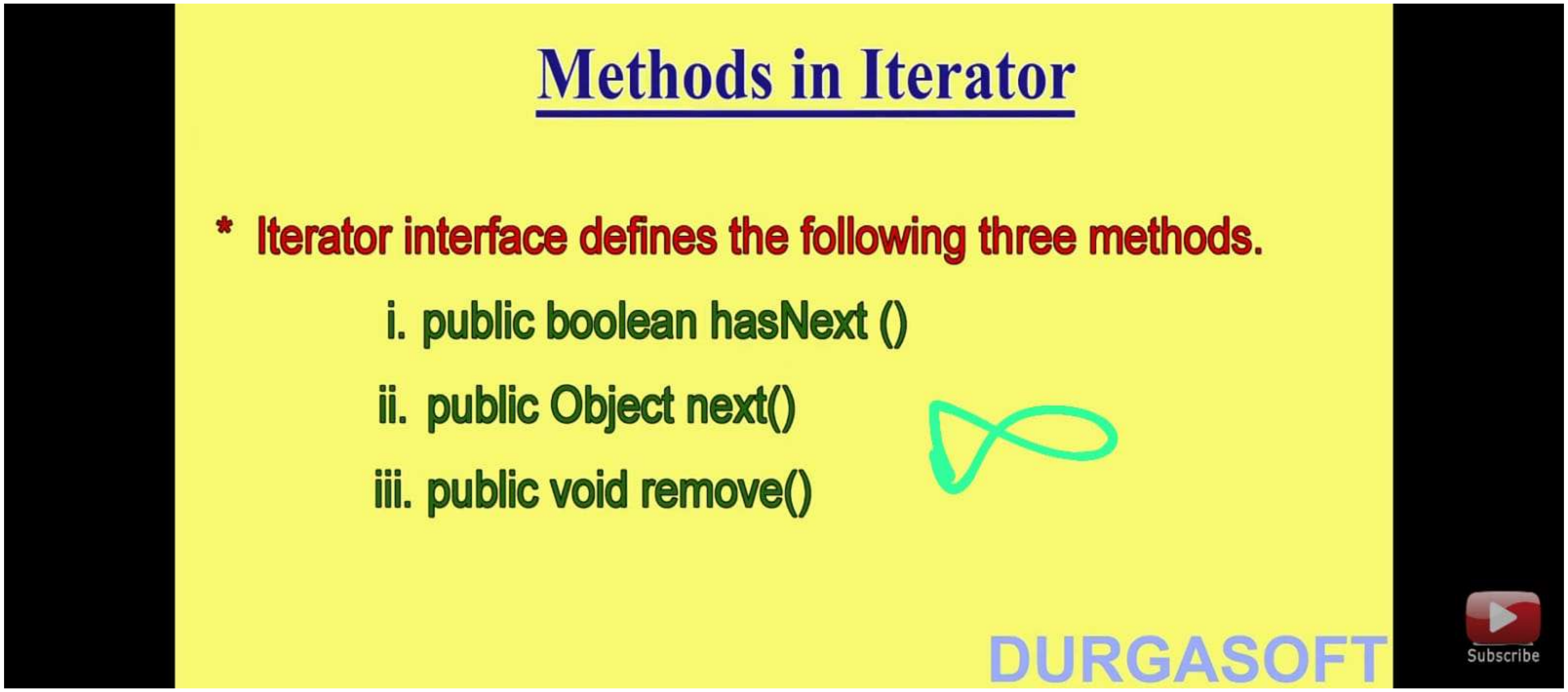
```
public Iterator iterator ();
```

Example:

```
Iterator itr=C. iterator();
```

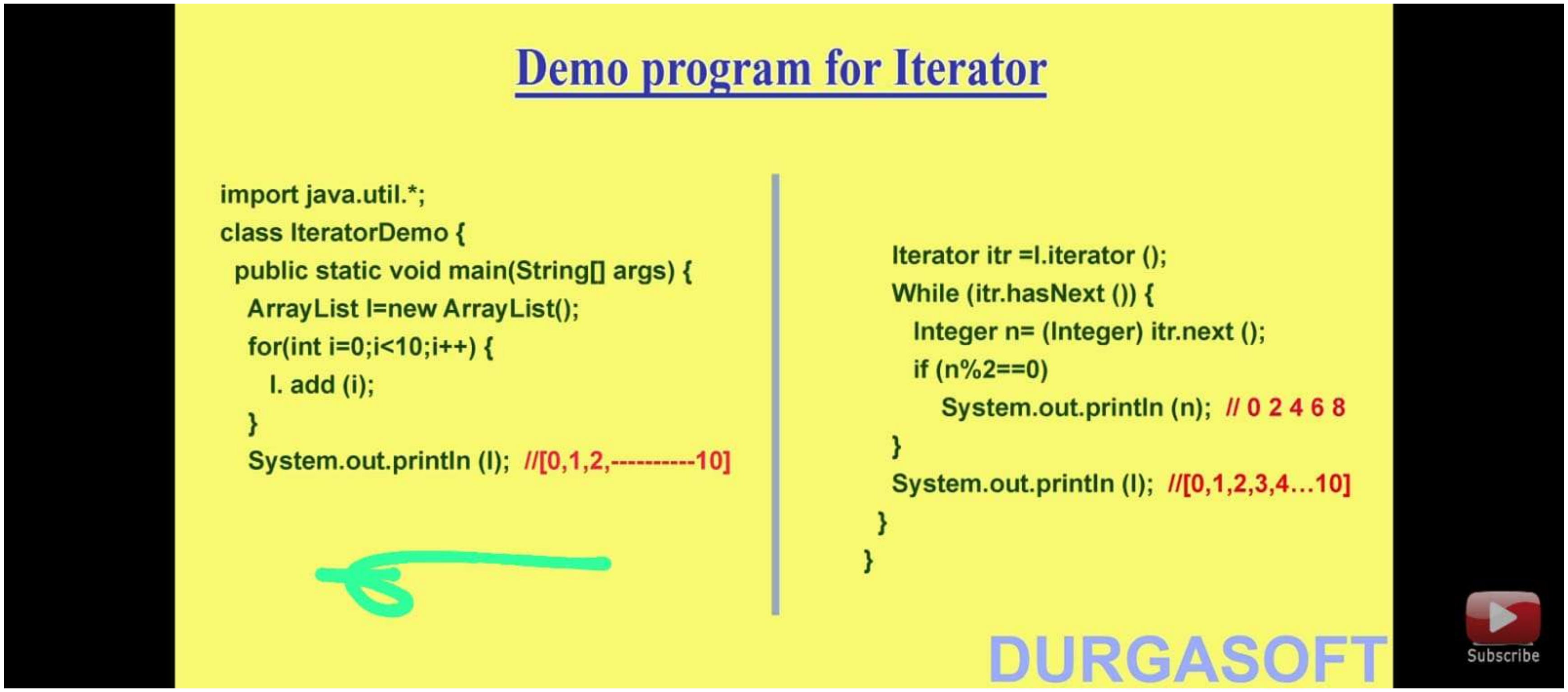
* where C is any Collection Object

DURGASOFT



Methods in Iterator

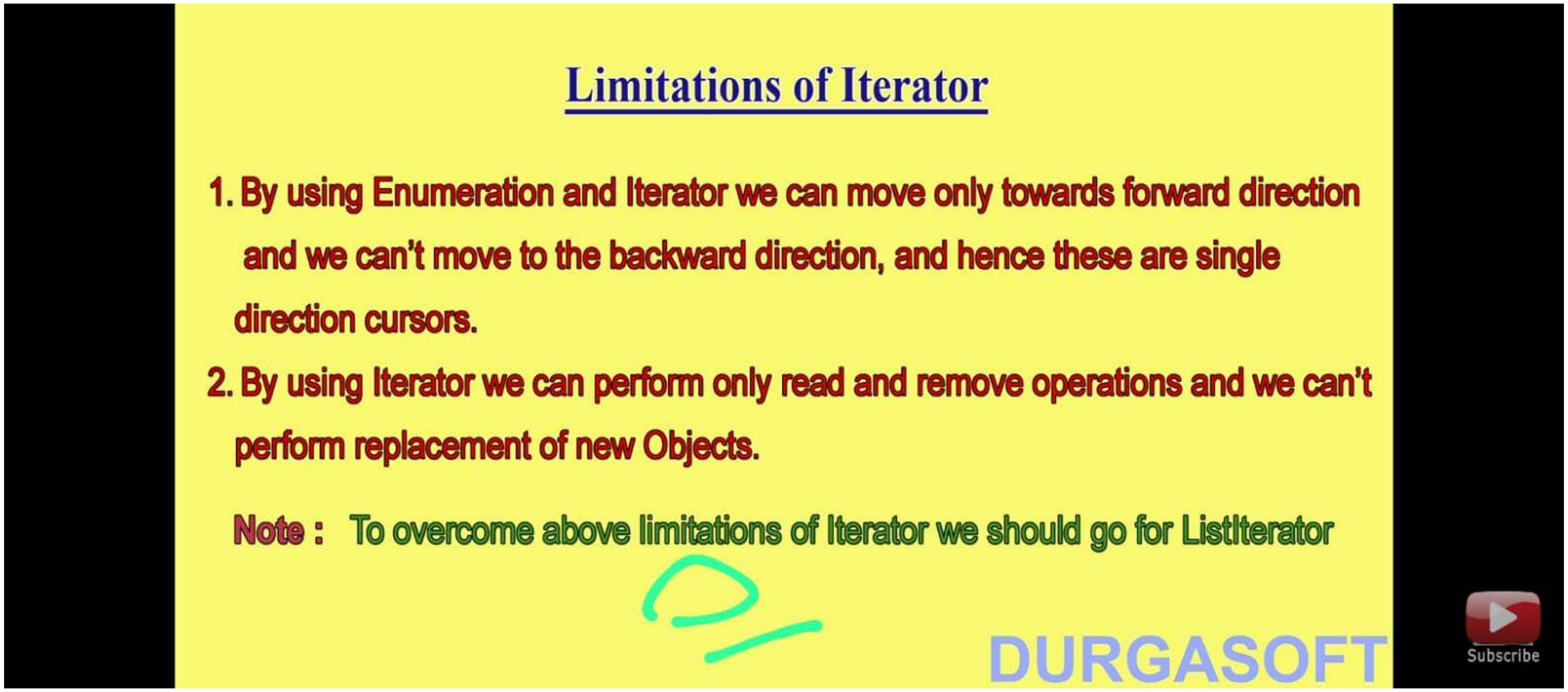
* Iterator interface defines the following three methods.
  - i. public boolean hasNext ()
  - ii. public Object next()
  - iii. public void remove()



Demo program for Iterator

```
import java.util.*;
class IteratorDemo {
 public static void main(String[] args) {
  ArrayList l=new ArrayList();
  for(int i=0;i<10;i++) {
    l. add (i);
  }
  System.out.println (l);  //[0,1,2,----------10]
```

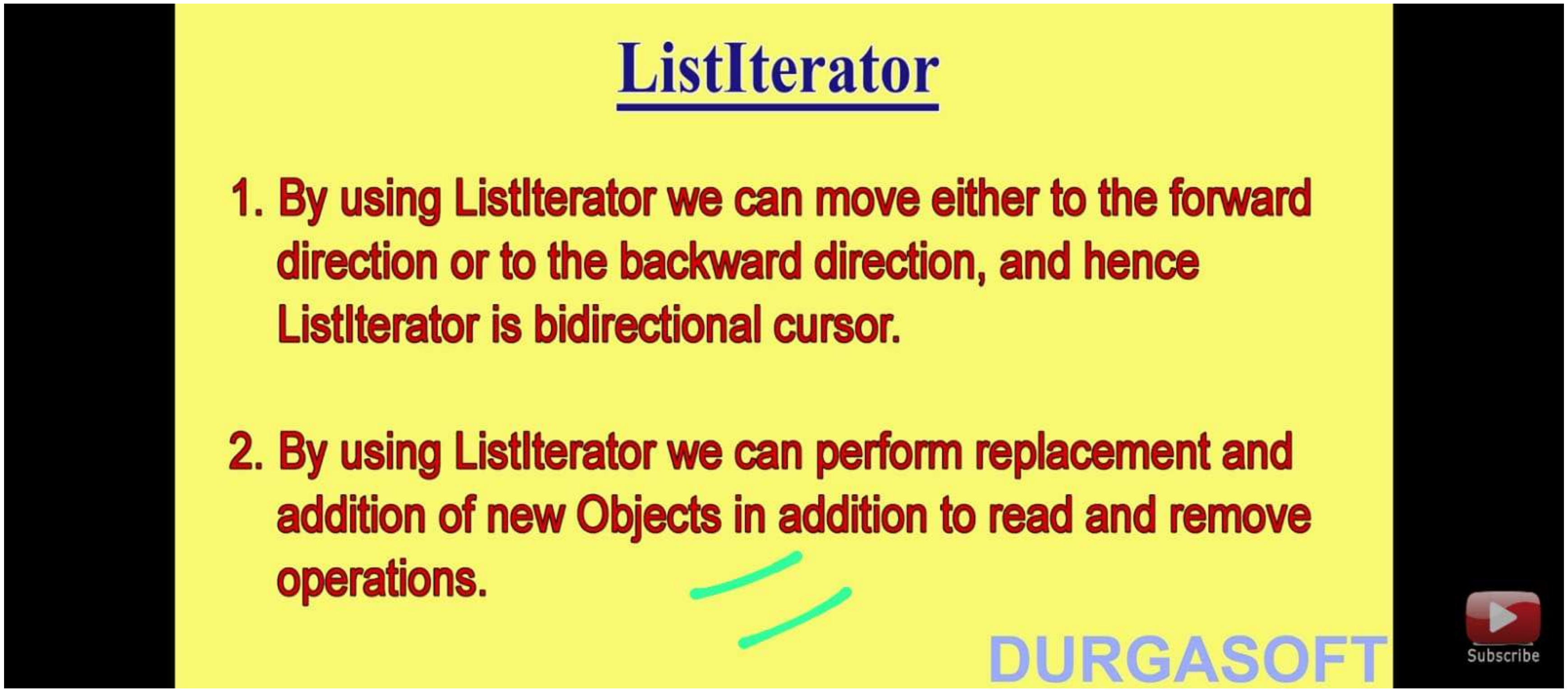
```
   Iterator itr =l.iterator ();
   While (itr.hasNext ()) {
     Integer n= (Integer) itr.next ();
     if (n%2==0)
        System.out.println (n);  // 0 2 4 6 8
   }
   System.out.println (l);  //[0,1,2,3,4…10]
  }
}
```



Limitations of Iterator

1. By using Enumeration and Iterator we can move only towards forward direction and we can't move to the backward direction, and hence these are single direction cursors.
2. By using Iterator we can perform only read and remove operations and we can't perform replacement of new Objects.

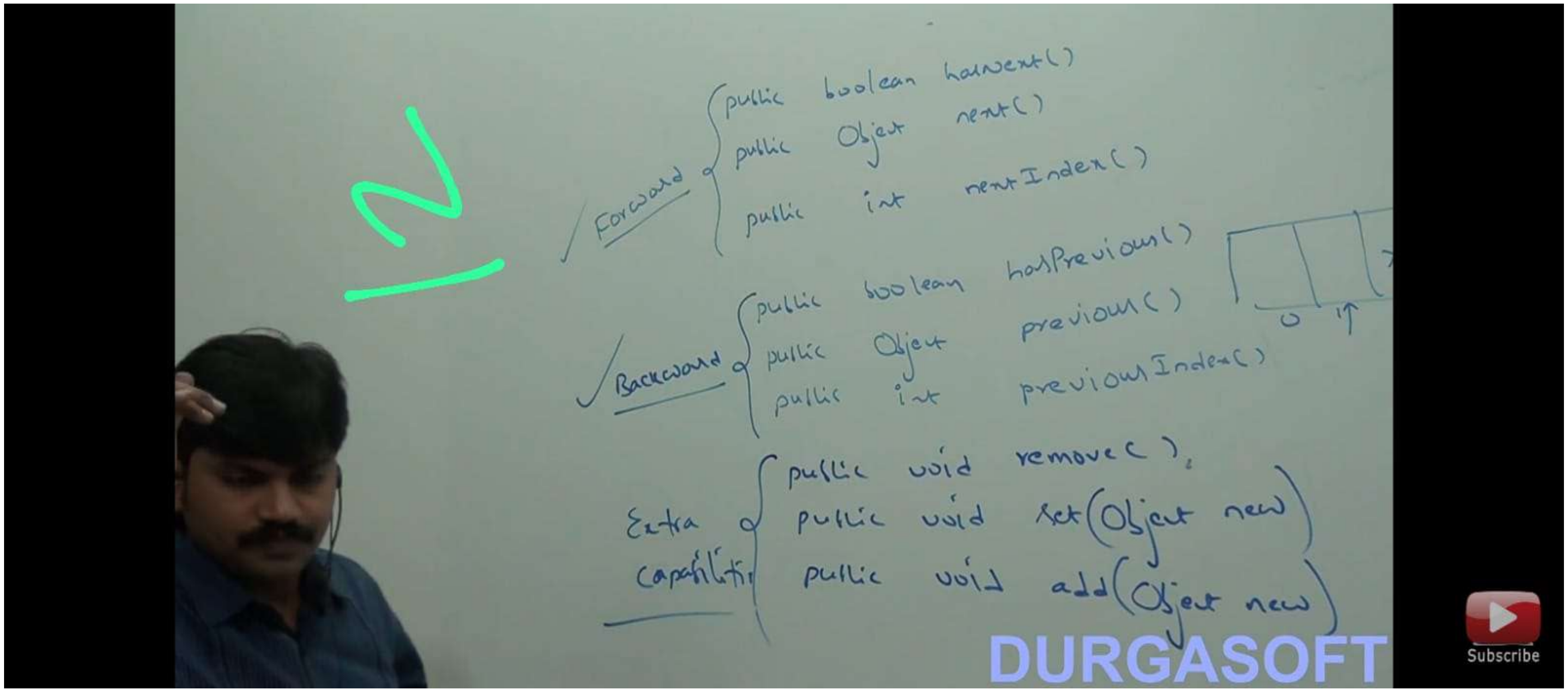
Note : To overcome above limitations of Iterator we should go for ListIterator



ListIterator

1. By using ListIterator we can move either to the forward direction or to the backward direction, and hence ListIterator is bidirectional cursor.

2. By using ListIterator we can perform replacement and addition of new Objects in addition to read and remove operations.



12

✓ Forward

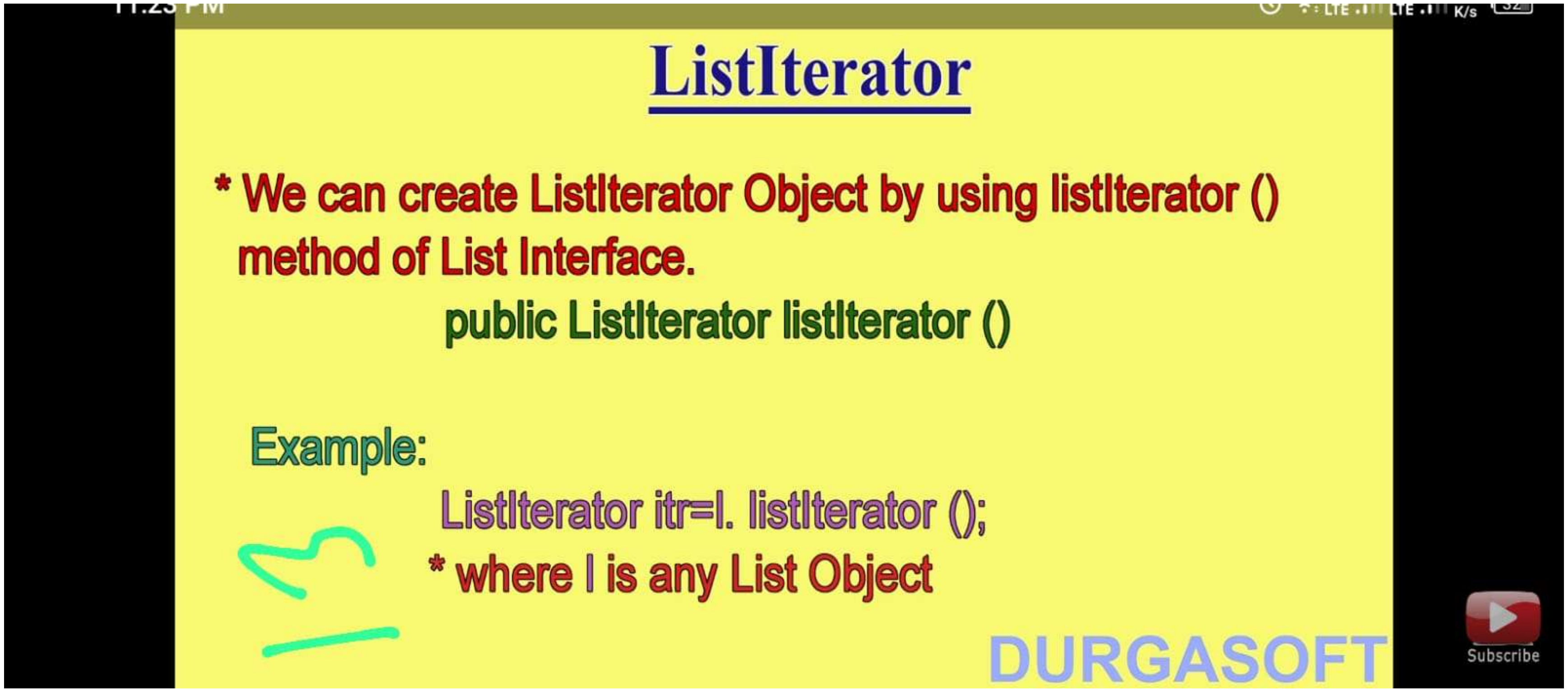
```
public boolean hasNext()
public Object  next()
public int     nextIndex()
```

✓ Backward

```
public boolean hasPrevious()
public Object  previous()
public int     previousIndex()
```

Extra capabilities

```
public void remove();
public void set(Object new)
public void add(Object new)
```



ListIterator

* We can create ListIterator Object by using listIterator () method of List Interface.

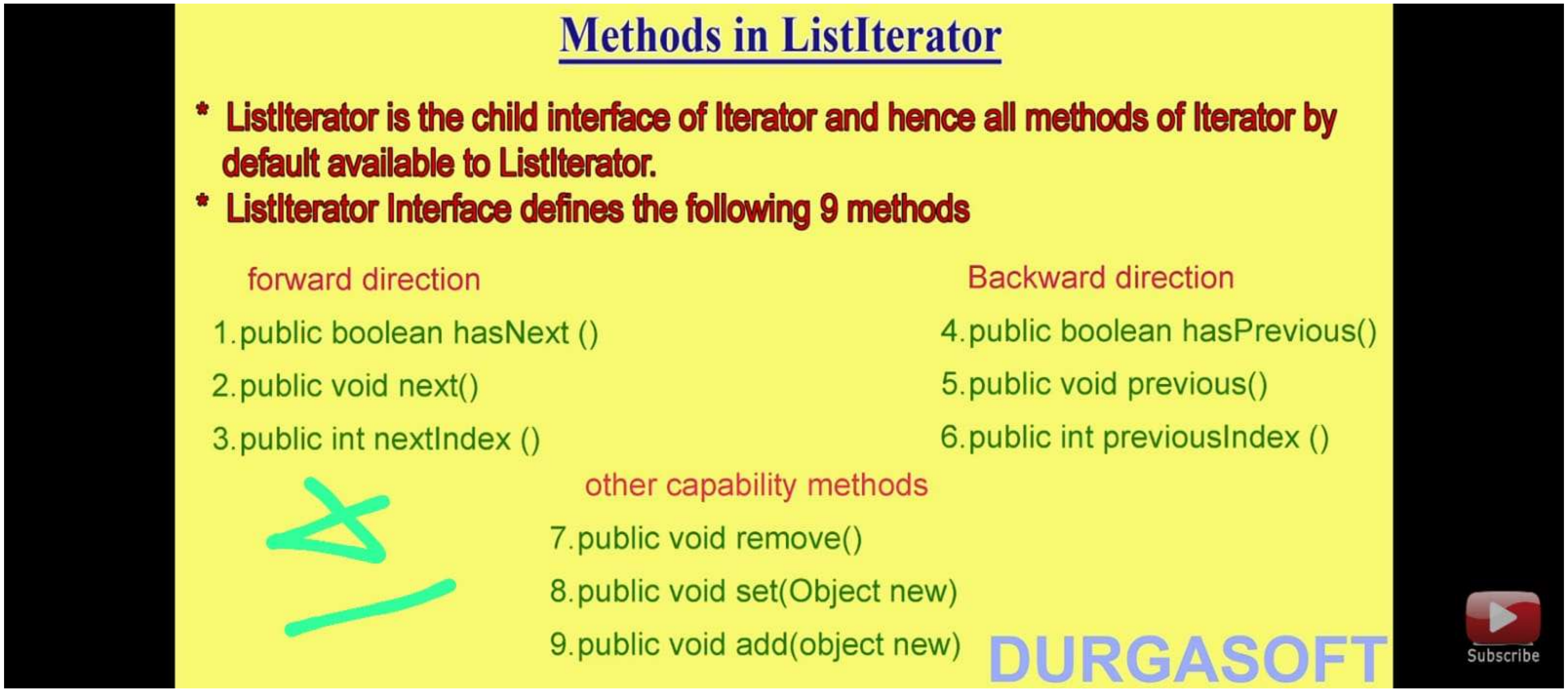
```
public ListIterator listIterator ()
```

Example:

```
ListIterator itr=l. listIterator ();
```

* where l is any List Object

DURGASOFT



Methods in ListIterator

- ListIterator is the child interface of Iterator and hence all methods of Iterator by default available to ListIterator.
- ListIterator Interface defines the following 9 methods

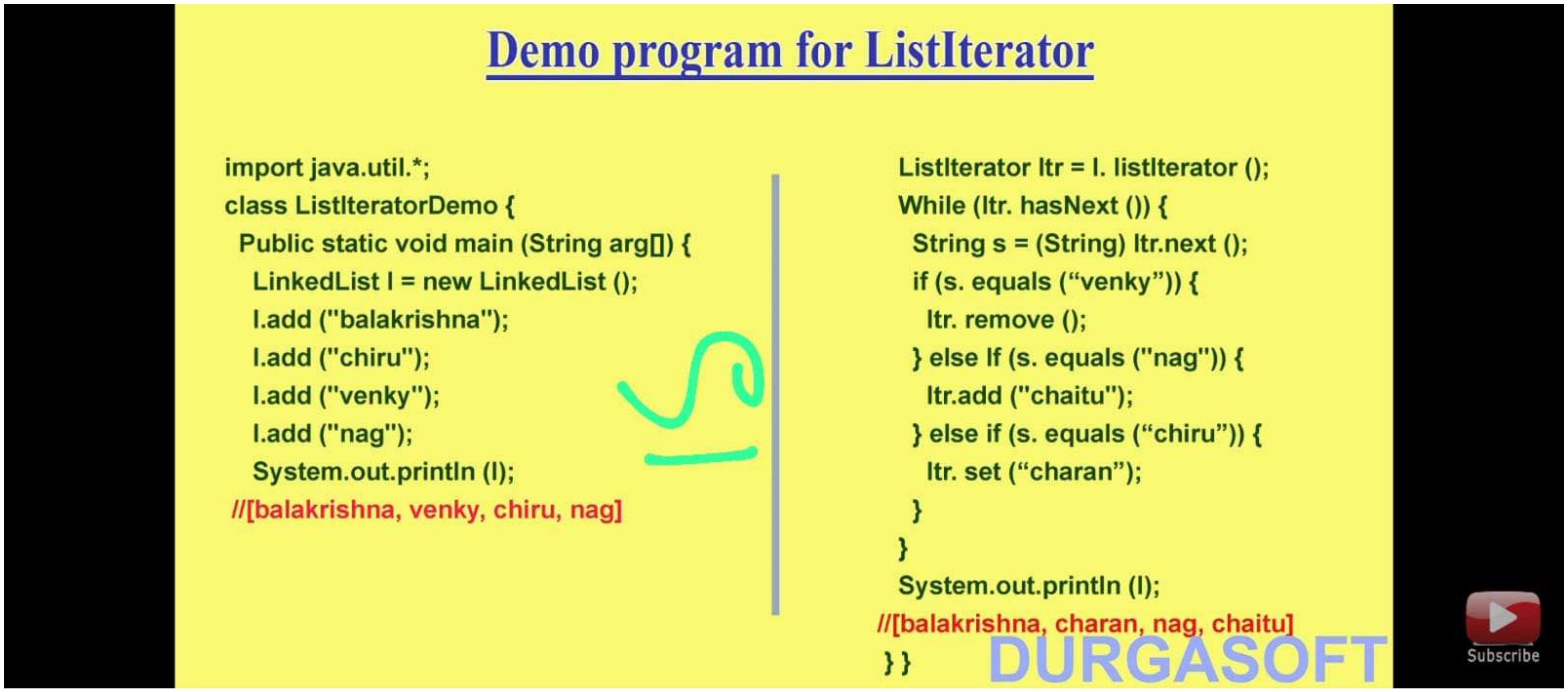
forward direction	Backward direction
1.public boolean hasNext ()	4.public boolean hasPrevious()
2.public void next()	5.public void previous()
3.public int nextIndex ()	6.public int previousIndex ()

other capability methods

7.public void remove()

8.public void set(Object new)

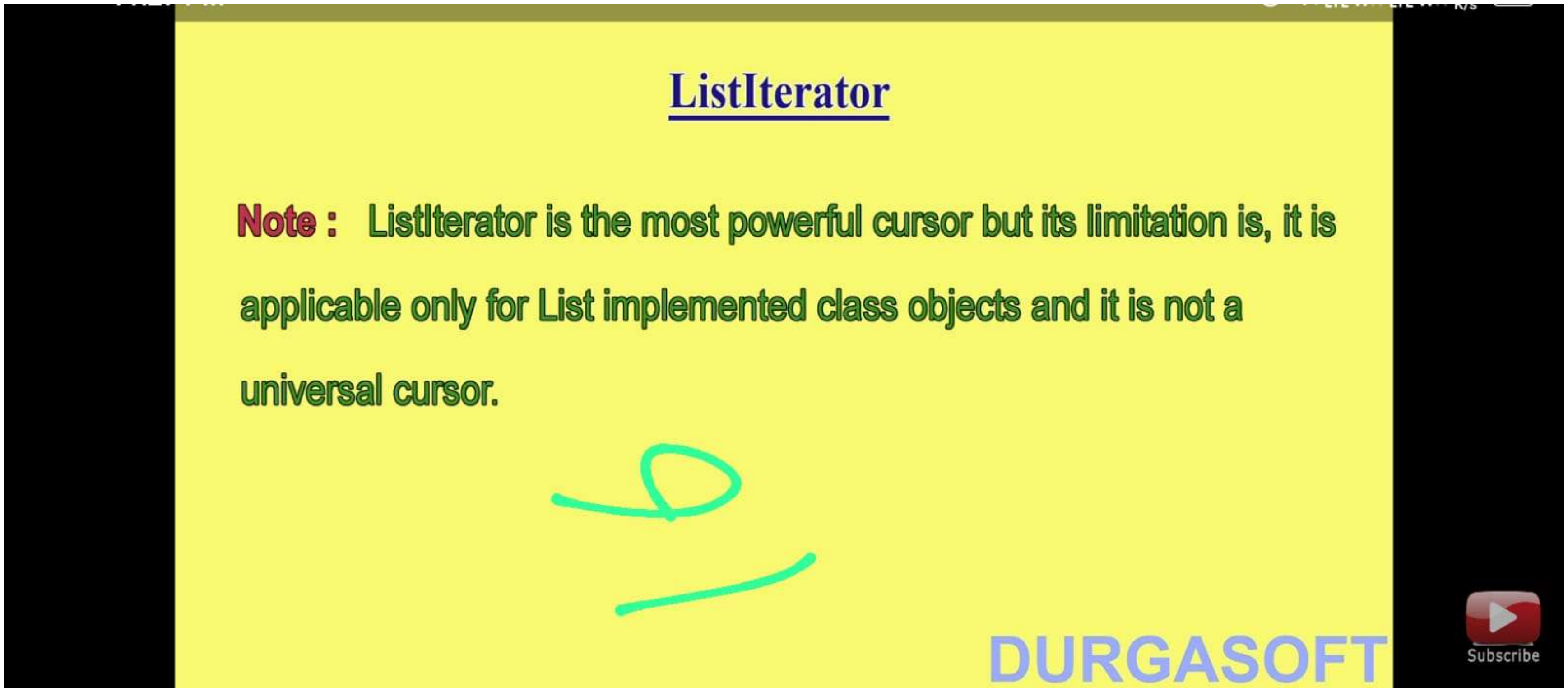
9.public void add(object new)



Demo program for ListIterator

```
import java.util.*;
class ListIteratorDemo {
 Public static void main (String arg[]) {
  LinkedList l = new LinkedList ();
  l.add ("balakrishna");
  l.add ("chiru");
  l.add ("venky");
  l.add ("nag");
  System.out.println (l);
 //[balakrishna, venky, chiru, nag]
```

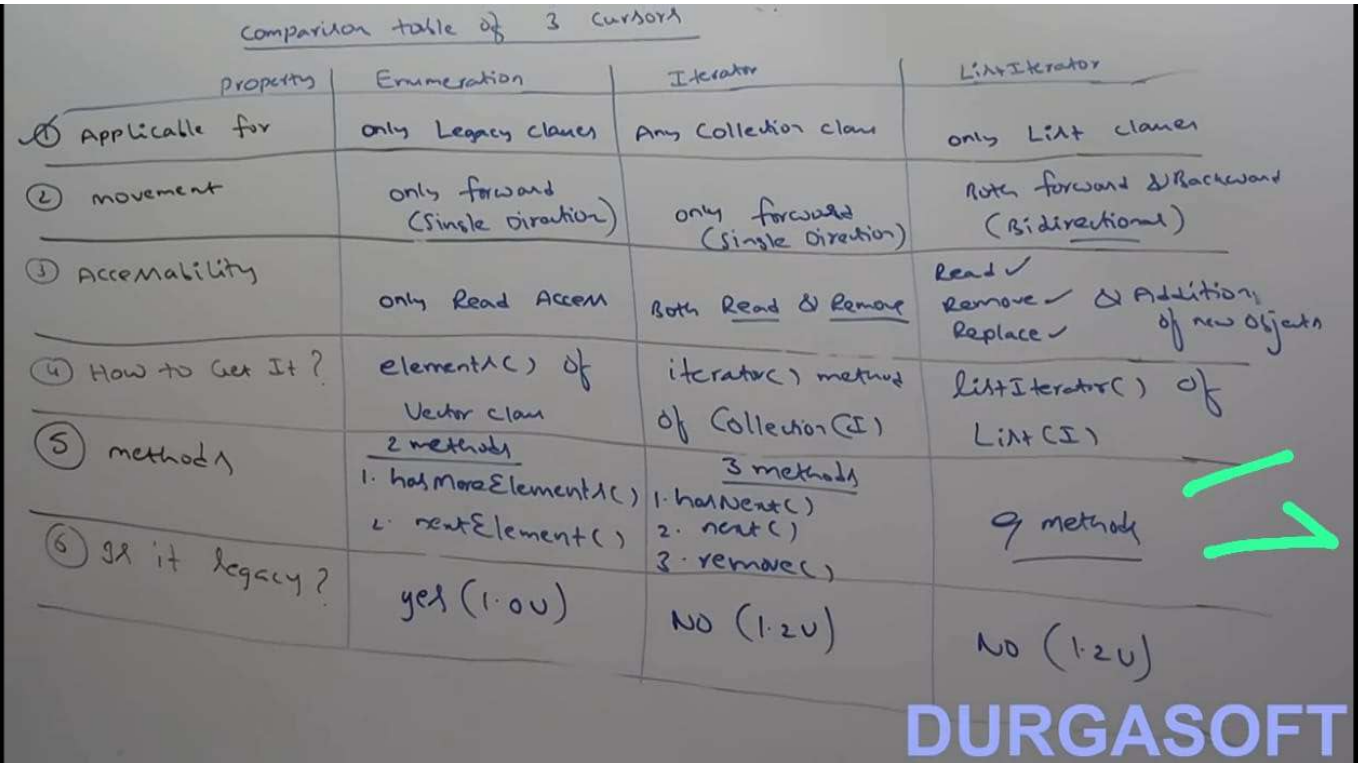
```
ListIterator ltr = l. listIterator ();
While (ltr. hasNext ()) {
 String s = (String) ltr.next ();
 if (s. equals ("venky")) {
  ltr. remove ();
 } else If (s. equals ("nag")) {
  ltr.add ("chaitu");
 } else if (s. equals ("chiru")) {
  ltr. set ("charan");
 }
}
System.out.println (l);
//[balakrishna, charan, nag, chaitu]
} }
```



ListIterator

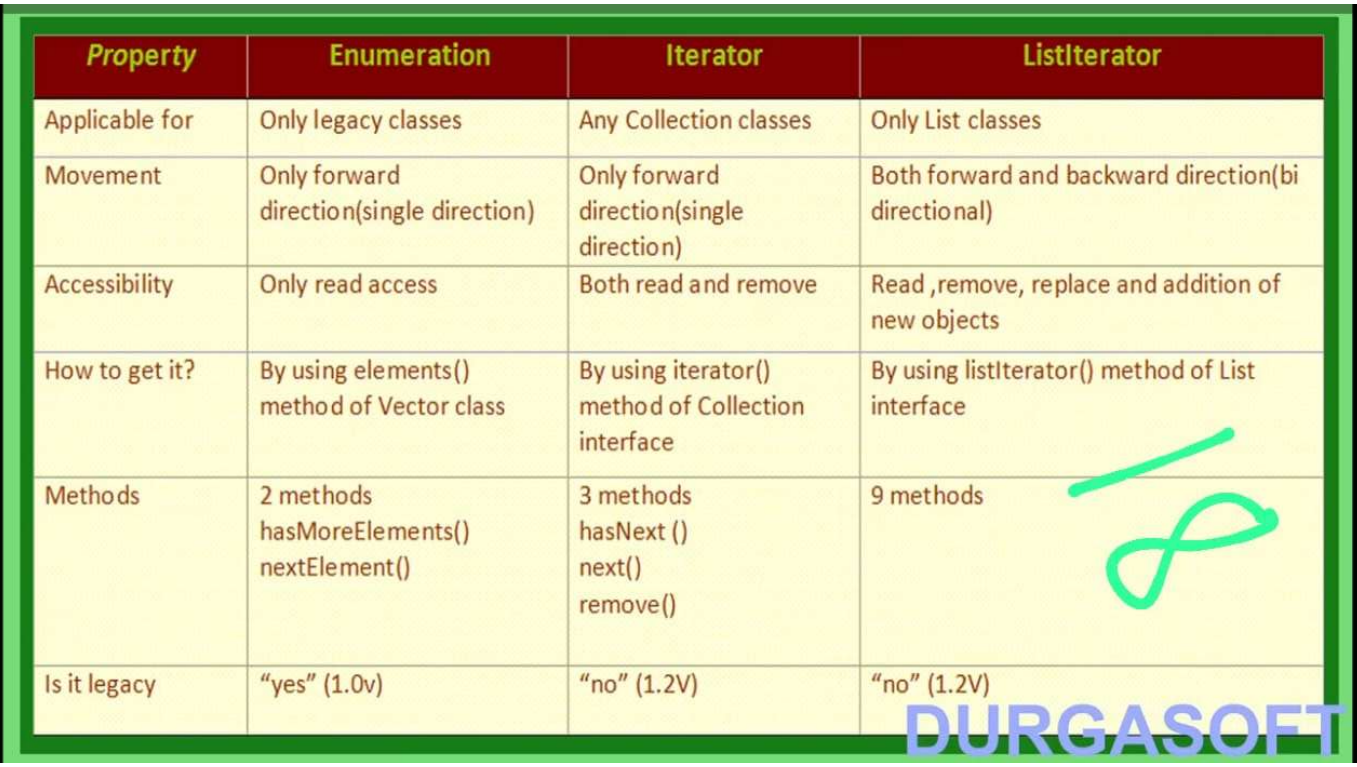
Note : ListIterator is the most powerful cursor but its limitation is, it is applicable only for List implemented class objects and it is not a universal cursor.

16

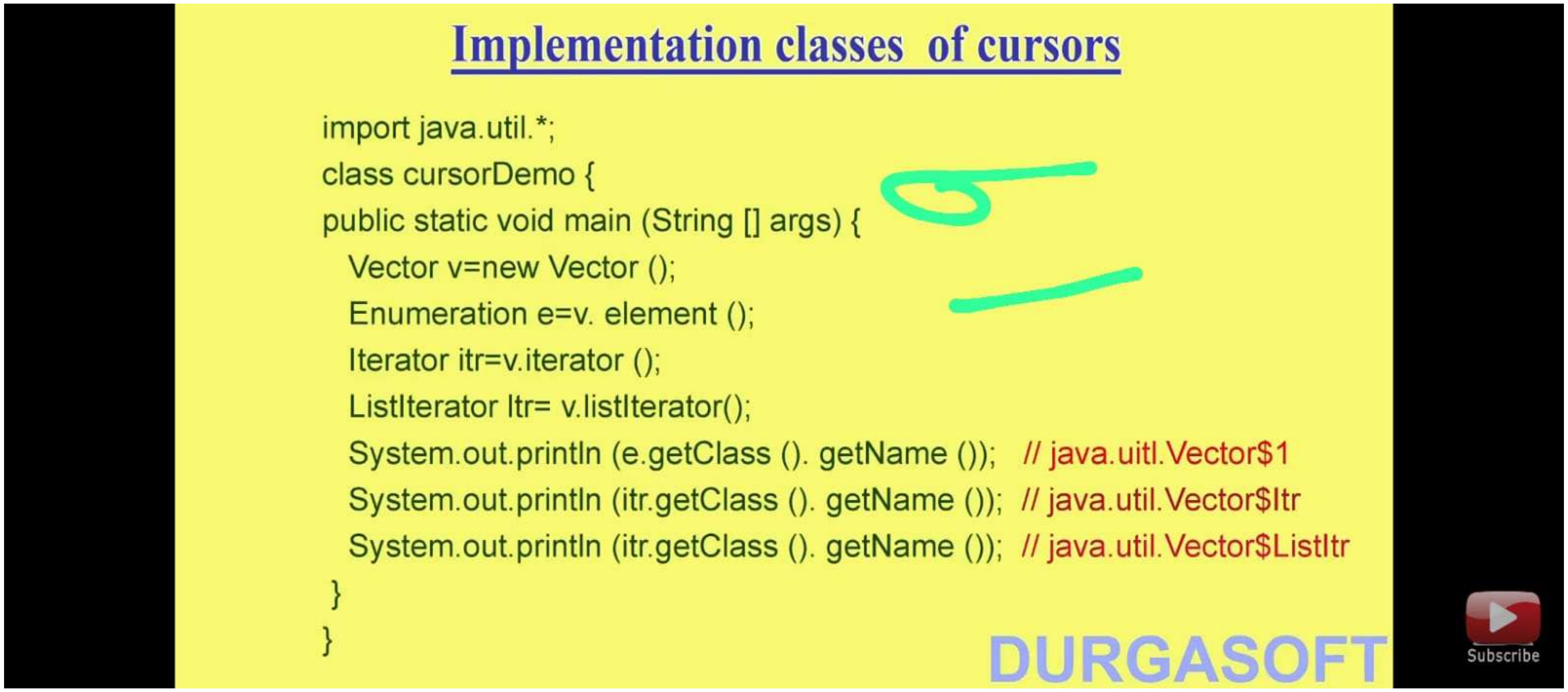


Comparison table of 3 Cursors

Property	Enumeration	Iterator	ListIterator
① Applicable for	only Legacy classes	Any Collection class	only List classes
② movement	only forward (Single Direction)	only forward (Single Direction)	Both forward & Backward (Bidirectional)
③ Accessibility	only Read Access	Both Read & Remove	Read ✓ Remove ✓ Replace ✓ & Addition of new objects
④ How to Get It?	elements() of Vector class	iterator() method of Collection(I)	listIterator() of List(I)
⑤ methods	2 methods 1. hasMoreElements() 2. nextElement()	3 methods 1. hasNext() 2. next() 3. remove()	9 methods
⑥ Is it legacy?	yes (1.0 V)	NO (1.2 V)	NO (1.2 V)



Property	Enumeration	Iterator	ListIterator
Applicable for	Only legacy classes	Any Collection classes	Only List classes
Movement	Only forward direction(single direction)	Only forward direction(single direction)	Both forward and backward direction(bi directional)
Accessibility	Only read access	Both read and remove	Read ,remove, replace and addition of new objects
How to get it?	By using elements() method of Vector class	By using iterator() method of Collection interface	By using listIterator() method of List interface
Methods	2 methods hasMoreElements() nextElement()	3 methods hasNext () next() remove()	9 methods 18
Is it legacy	"yes" (1.0v)	"no" (1.2V)	"no" (1.2V)



Implementation classes of cursors

```
import java.util.*;
class cursorDemo {
public static void main (String [] args) {
  Vector v=new Vector ();
  Enumeration e=v. element ();
  Iterator itr=v.iterator ();
  ListIterator ltr= v.listIterator();
  System.out.println (e.getClass (). getName ());   // java.uitl.Vector$1
  System.out.println (itr.getClass (). getName ());  // java.util.Vector$Itr
  System.out.println (itr.getClass (). getName ());  // java.util.Vector$ListItr
 }
}
```

DURGASOFT