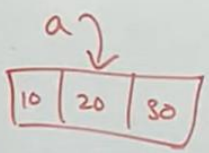
Arrays

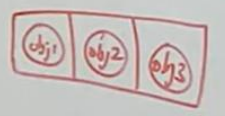
1. Arrays can store primitive & non-primitive (objects) type of data

→ `int[] a={10,20,30};`



→ class Test
{
}
3 ≡

`Test[] t={obj1,obj2,obj3};`



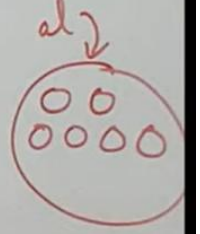
2. Arrays can store only homogeneous (same) type of data
3. Array size is fixed, we cannot increase or [illegible] an array at runtime.

Arrays are inbuilt feature of java & thus we have to develop algorithms

Collection Framework

1. Collection Framework can contain only non-primitive type of data

```
ArrayList al=new ArrayList();
al.add(obj1);
al.add(10);  al.add('z');
```



2. We can store hetrogeneous (different) type of data
3. We can increase or decrease the size of collections at runtime.
4. Collection framework is an API which provides predefined classes, interfaces & methods.

Collection framework:-

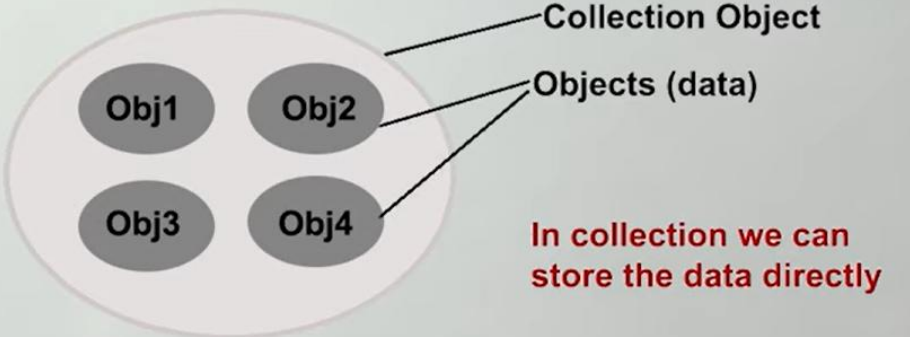
→ Collection:- It is the single entity or object which can store multiple data

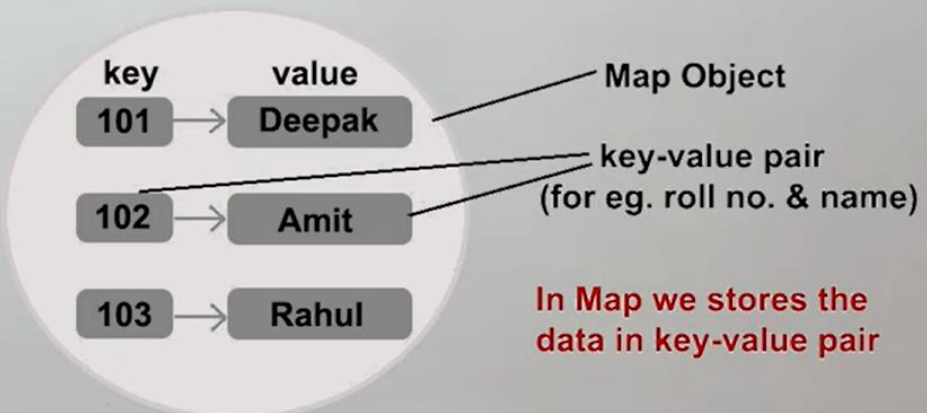
→ Framework:- represents the library

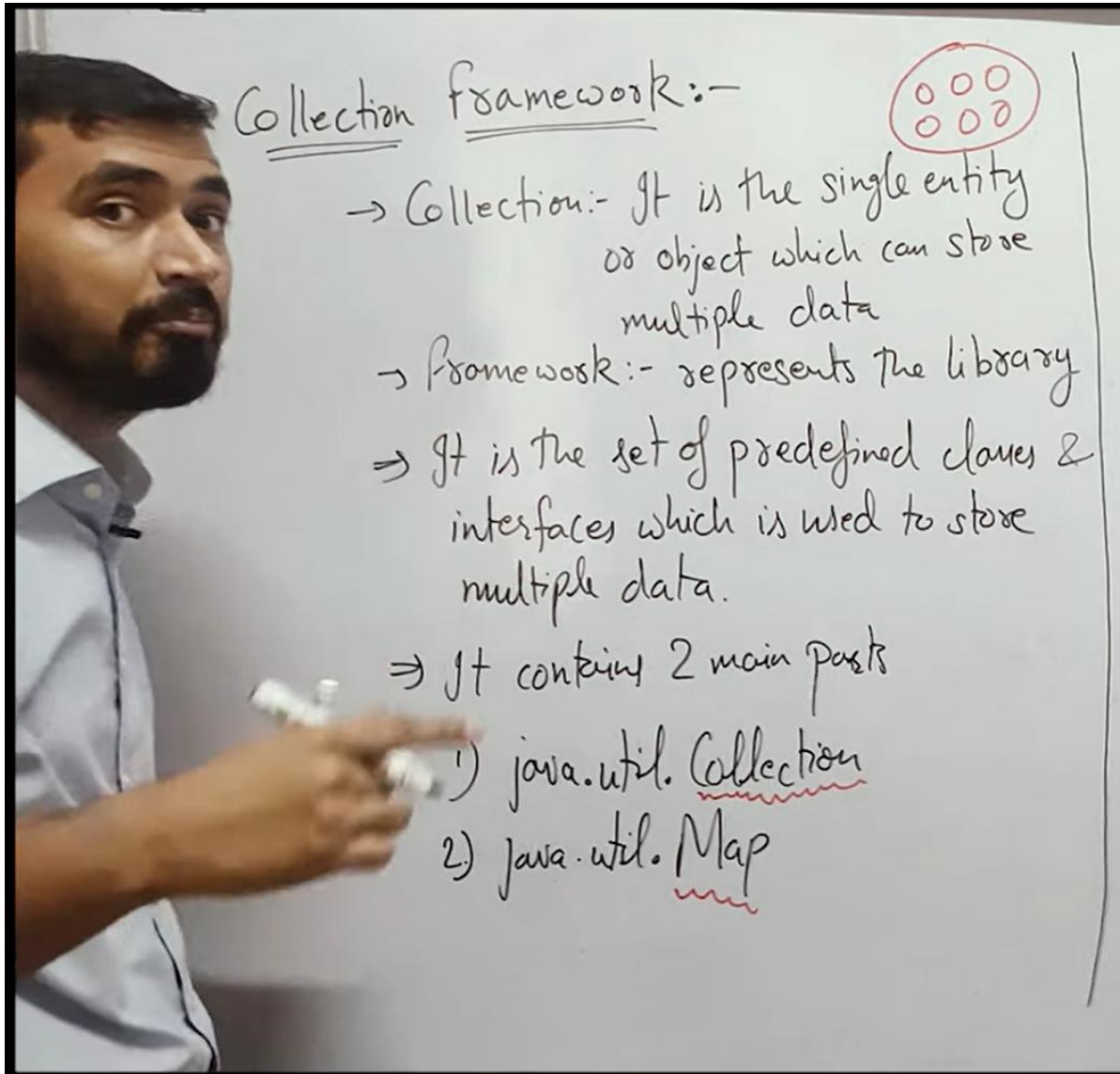
⇒ It is the set of predefined classes & interfaces which is used to store multiple data.

⇒ It contains 2 main parts

1) java.util.Collection
2) java.util.Map







What is Collection Framework, Collection & Collections ?

COLLECTION FRAMEWORK (API) :-

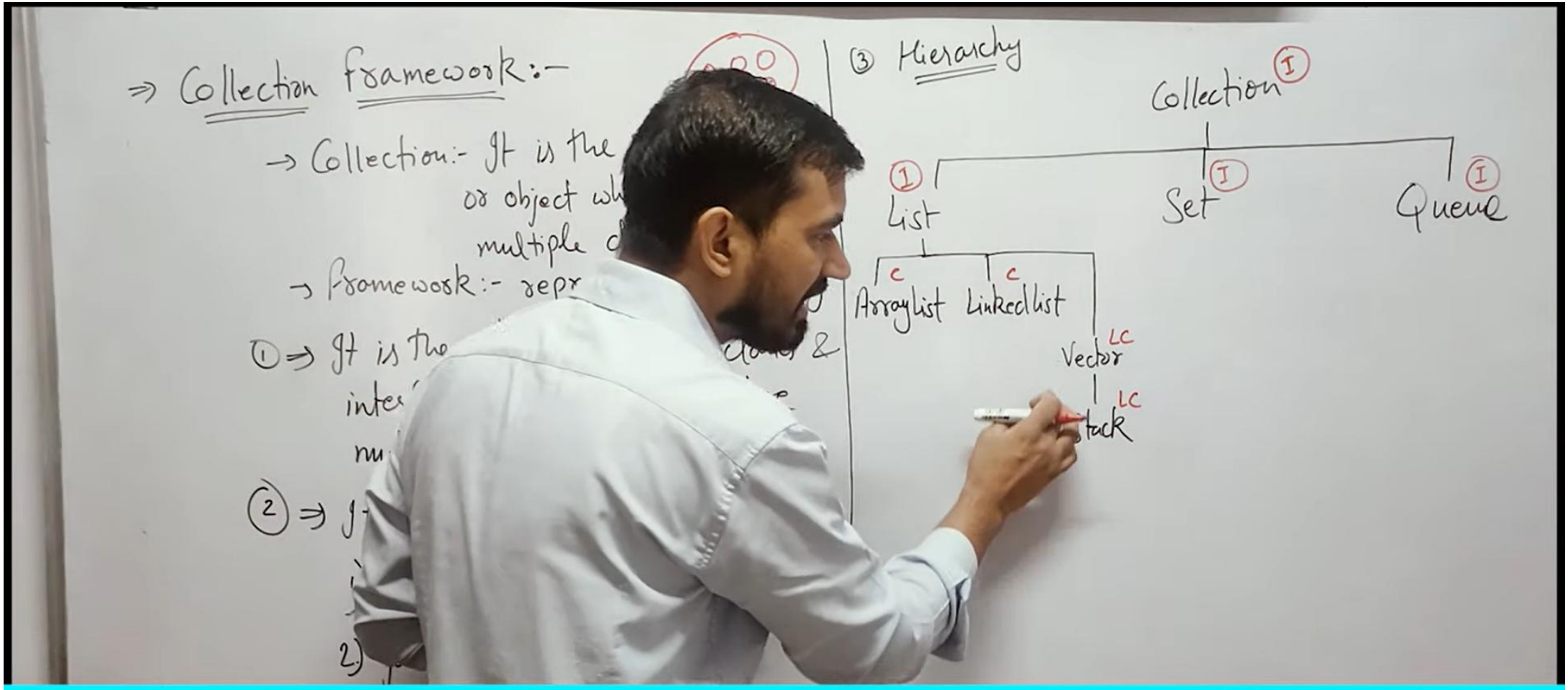
It is an API which contains predefined classes & interfaces

COLLECTION (Interface):-

It is the root interface (present in java.util package) of all the collection objects

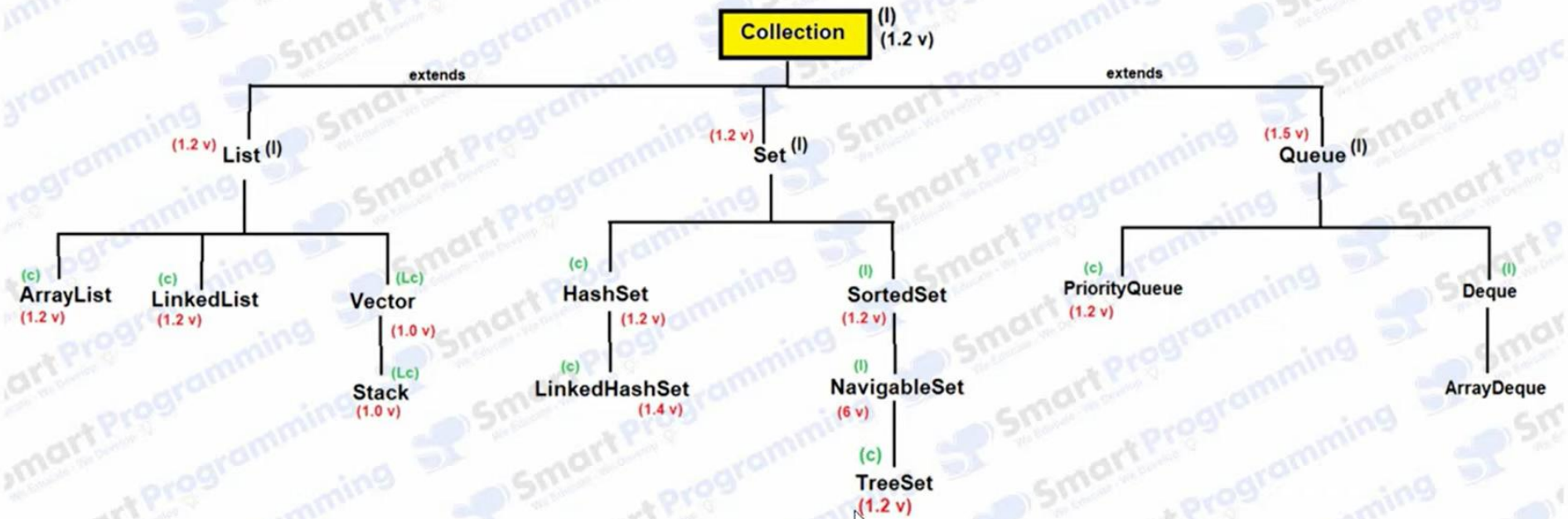
COLLECTIONS (Utility Class) :-

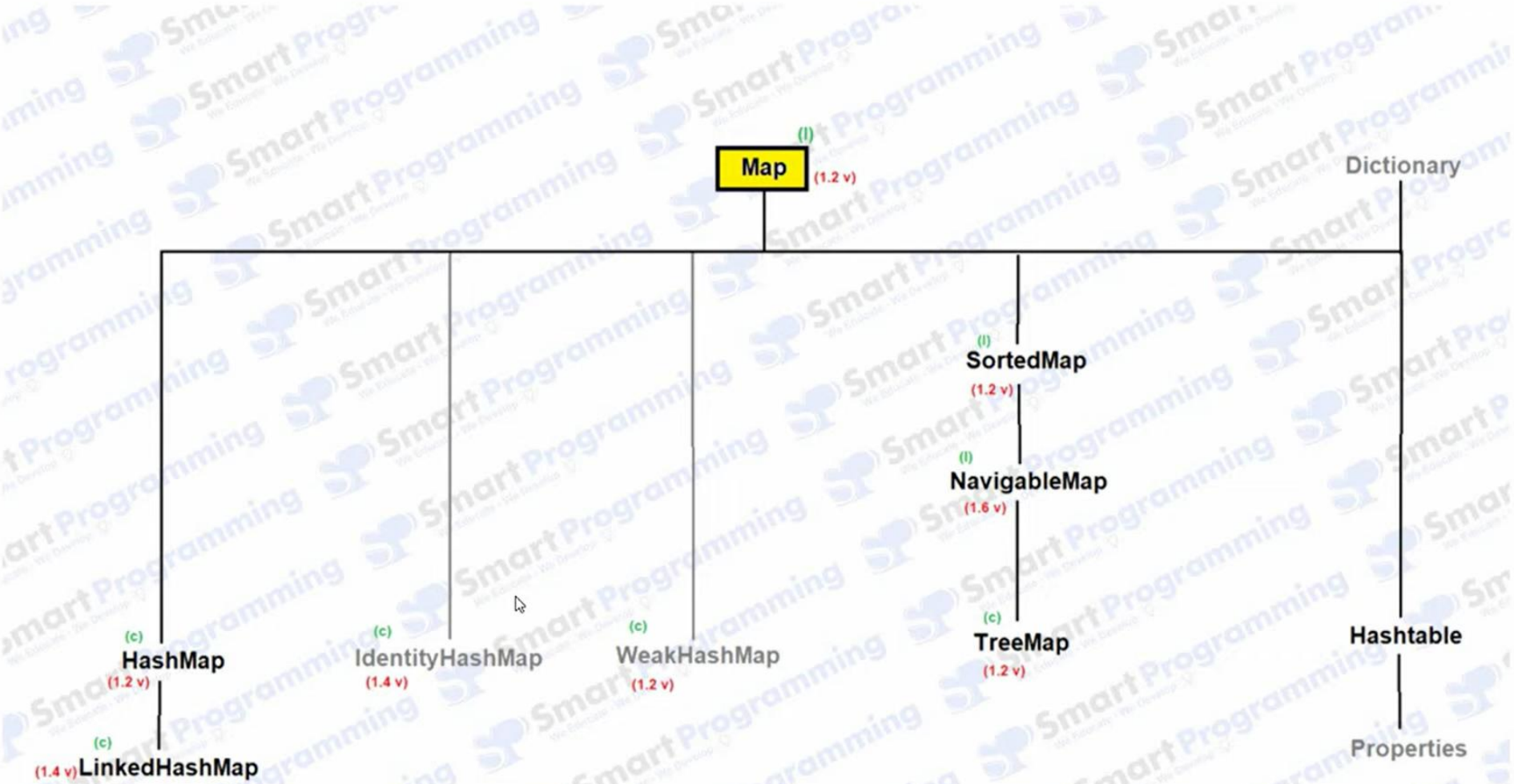
It is the utility class which contains only static methods

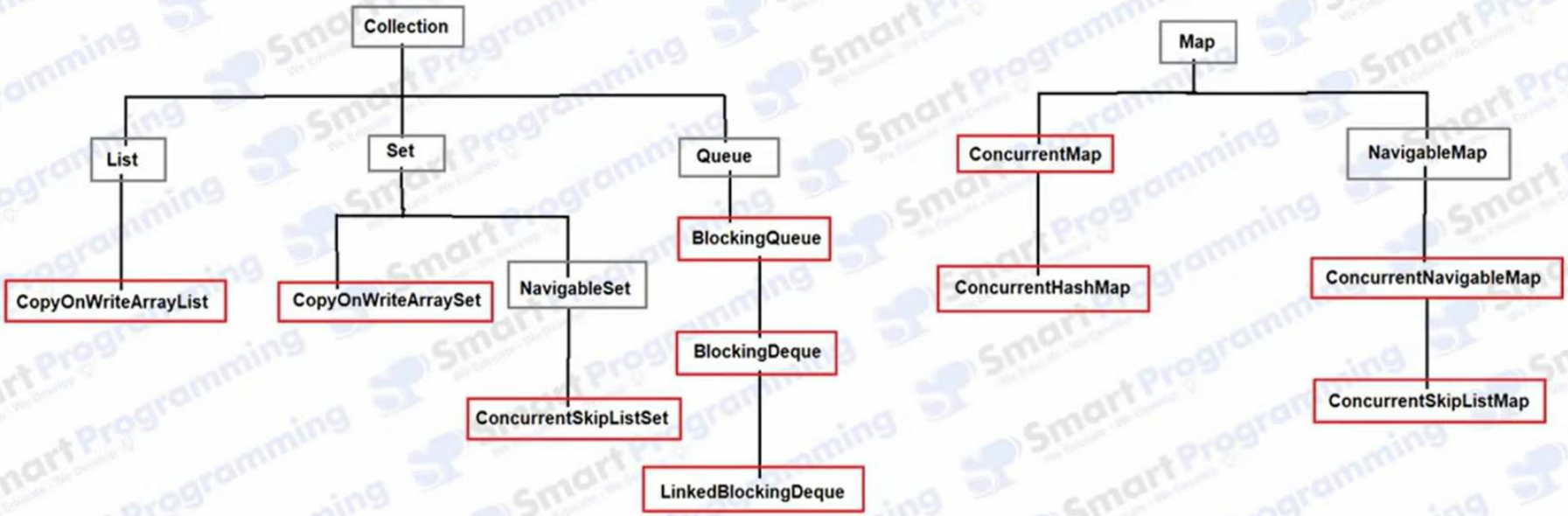


What is Legacy Class :- Collection Framework was not present in early versions of java. Instead it defines only several classes and one interface to store the objects.

But when Collection Framework came, these old classes were re-engineered or modified to support the Collection Framework. These old classes are known are legacy classes.


Collection
(I)
(1.2 v)
extends
extends
(1.2 v)
List (I)
(1.2 v)
Set (I)
(1.5 v)
Queue (I)
(c)
ArrayList
(1.2 v)
(c)
LinkedList
(1.2 v)
(Lc)
Vector
(1.0 v)
(Lc)
Stack
(1.0 v)
(c)
HashSet
(1.2 v)
(c)
LinkedHashSet
(1.4 v)
(I)
SortedSet
(1.2 v)
(I)
NavigableSet
(6 v)
(c)
TreeSet
(1.2 v)
(c)
PriorityQueue
(1.2 v)
(I)
Deque
ArrayDeque


(I)
Map
(1.2 v)
Dictionary
(I)
SortedMap
(1.2 v)
(I)
NavigableMap
(1.6 v)
(c)
TreeMap
(1.2 v)
(c)
HashMap
(1.2 v)
(c)
LinkedHashMap
(1.4 v)
(c)
IdentityHashMap
(1.4 v)
(c)
WeakHashMap
(1.2 v)
Hashtable
Properties


Collection
List
Set
Queue
CopyOnWriteArrayList
CopyOnWriteArraySet
NavigableSet
ConcurrentSkipListSet
BlockingQueue
BlockingDeque
LinkedBlockingDeque
Map
ConcurrentMap
NavigableMap
ConcurrentHashMap
ConcurrentNavigableMap
ConcurrentSkipListMap

Collection:-

1) Collection is an interface which is present in java.util package (1.2 version)

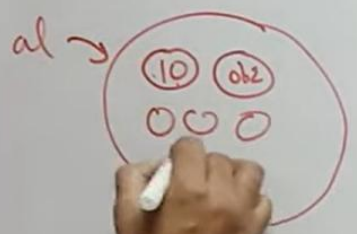
2) Syntax:- public interface Collection<E> extends Iterable<E>

```
{
    // methods
}
```

3) Hierachy of Collection interface

4) Methods of Collection interface

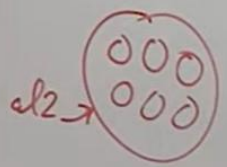
→ public boolean add(Object obj)
public boolean addAll(Collection c)



Arr... al=new AL();

...dd(10);

...add(ob2);

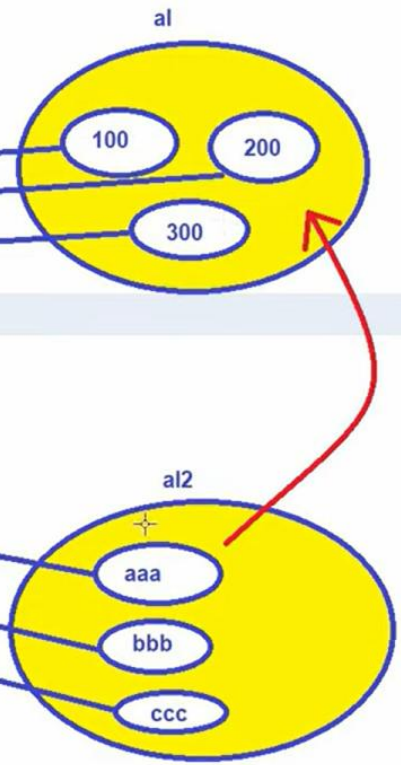


Untitled - Paint

File Home View

Paste Cut Copy Clipboard | Select Crop Resize Rotate Image | Brushes Tools | Outline Fill Shapes | Size | Color 1 Color 2 Edit colors Colors | Edit with Paint 3D

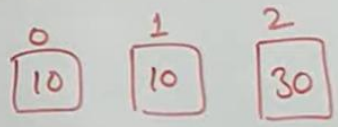
```
public class CollectionDemo
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        System.out.println(al);

        ArrayList al2=new ArrayList();
        al2.add("aaa");
        al2.add("bbb");
        al2.add("ccc");

        System.out.println(al2);

        al.addAll(al2);
        System.out.println(al);
    }
}
```



1070, 502px 1908 × 816px 100%

List

1.) List is an index based data structure



2). List can store duplicate elements

3). List can store any numbers of null values

4). List follows the insertion order.

5). We can iterate (get) the List elements by

It & ListIterator

Set

1.). Set is not an index based data Structure. It stores the data according to the hashcode values.

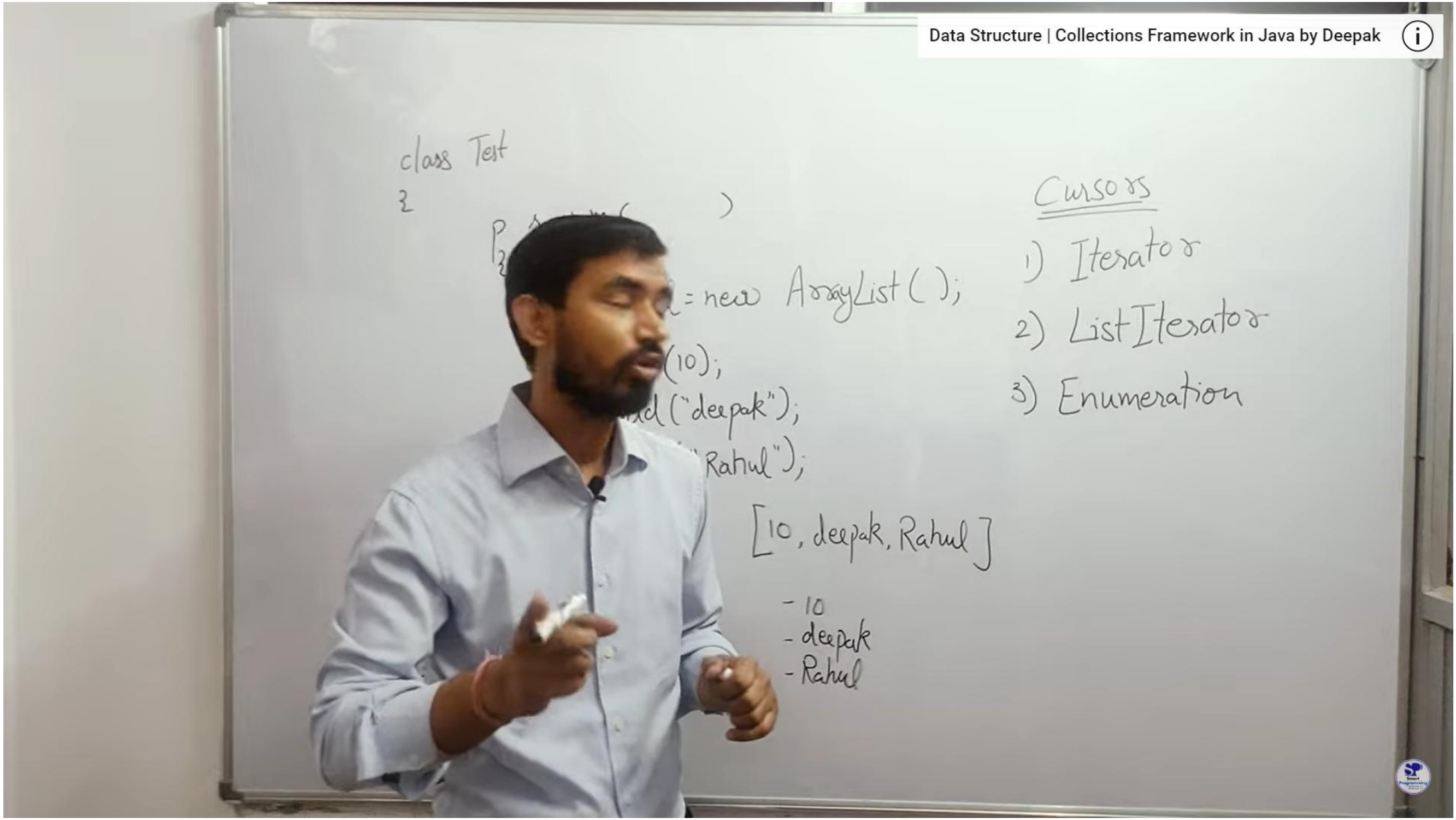
2). Set does not allow to store duplicate elements.

3) Set can store only one null value

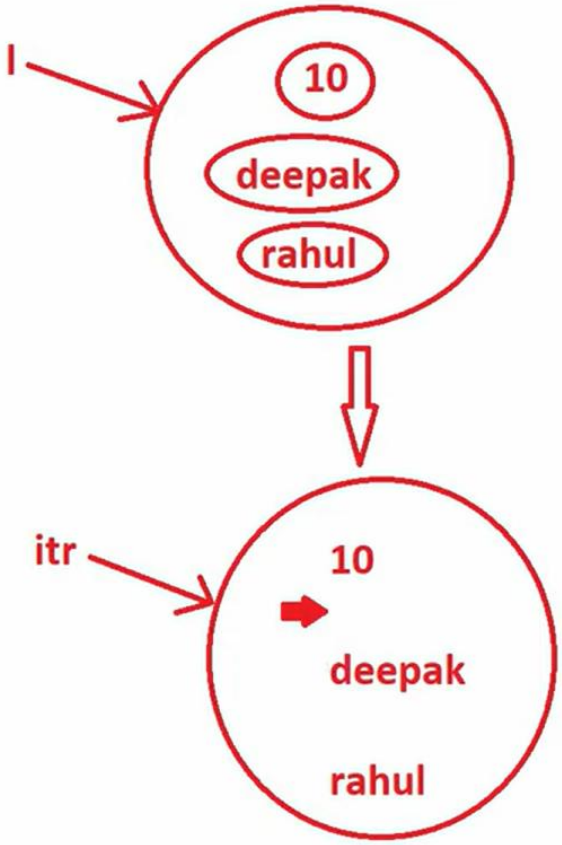
4). Set does not follows the insertion order.

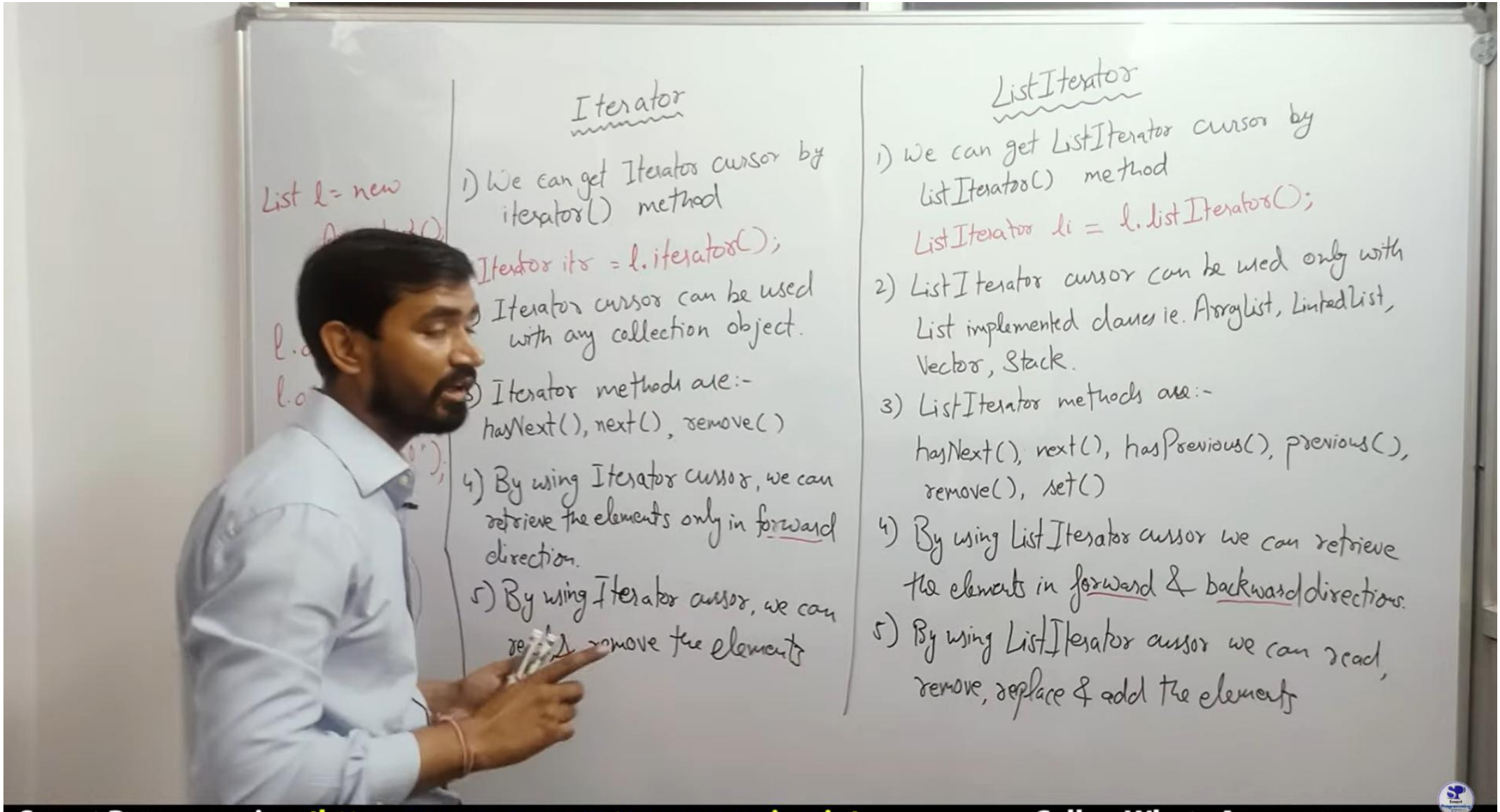
5) We can iterate the Set elements by

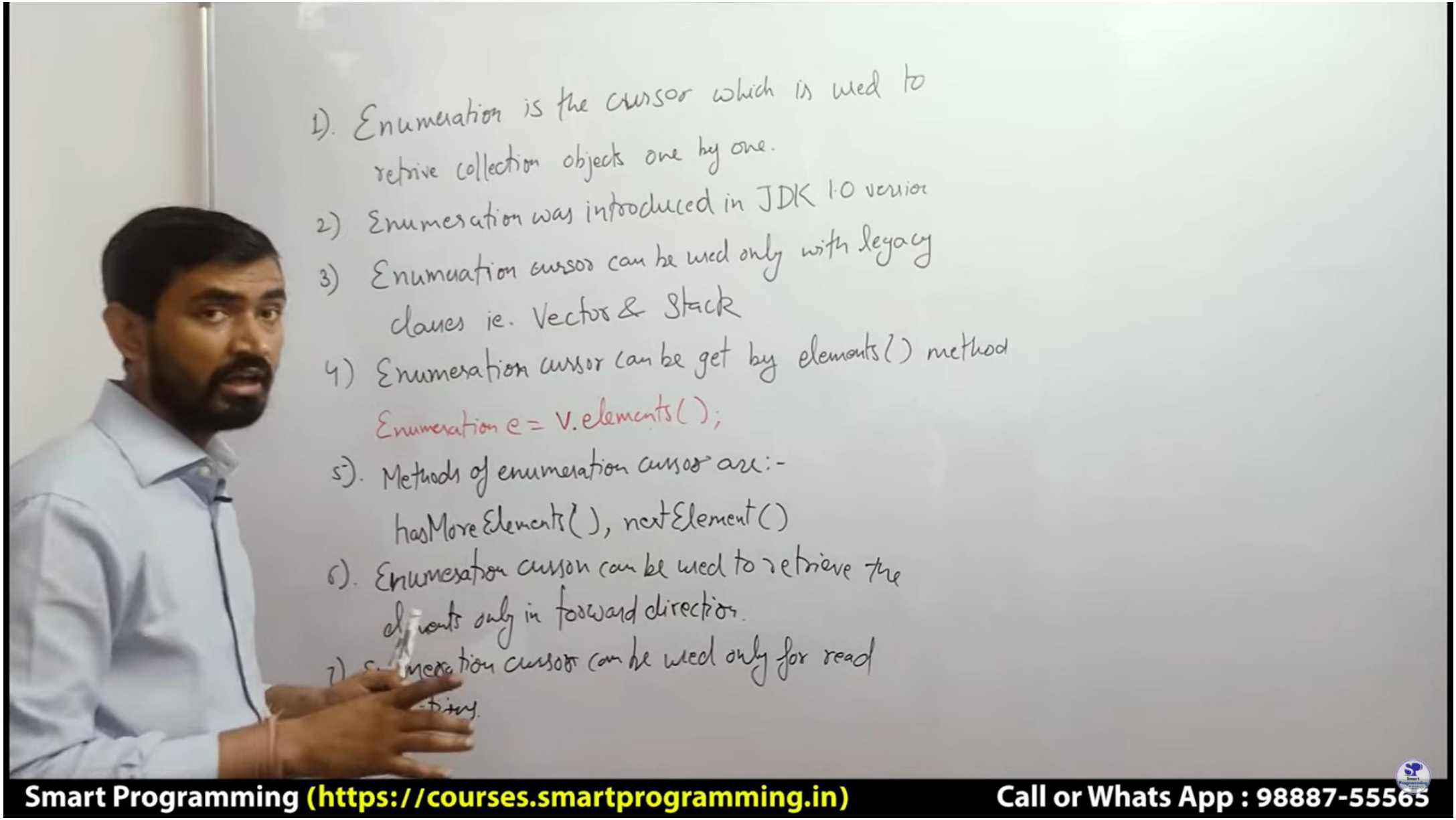
Iterator


Data Structure | Collections Framework in Java by Deepak
class Test
= new ArrayList();
(10);
("deepak");
Rahul");
[10, deepak, Rahul]
- 10
- deepak
- Rahul
Cursors
1) Iterator
2) ListIterator
3) Enumeration

```
List l=new ArrayList();

l.add(10);
l.add("deepak");
l.add("rahul");

//System.out.println(l);    [10, deepak, rahul]

Iterator itr=l.iterator();

while(itr.hasNext())
{
   System.out.println(itr.next());
}
```







1 ArrayList is an implemented class of List interface which is present in java.util package.

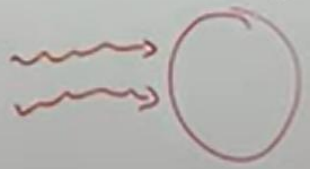
⇒ Syntax:-

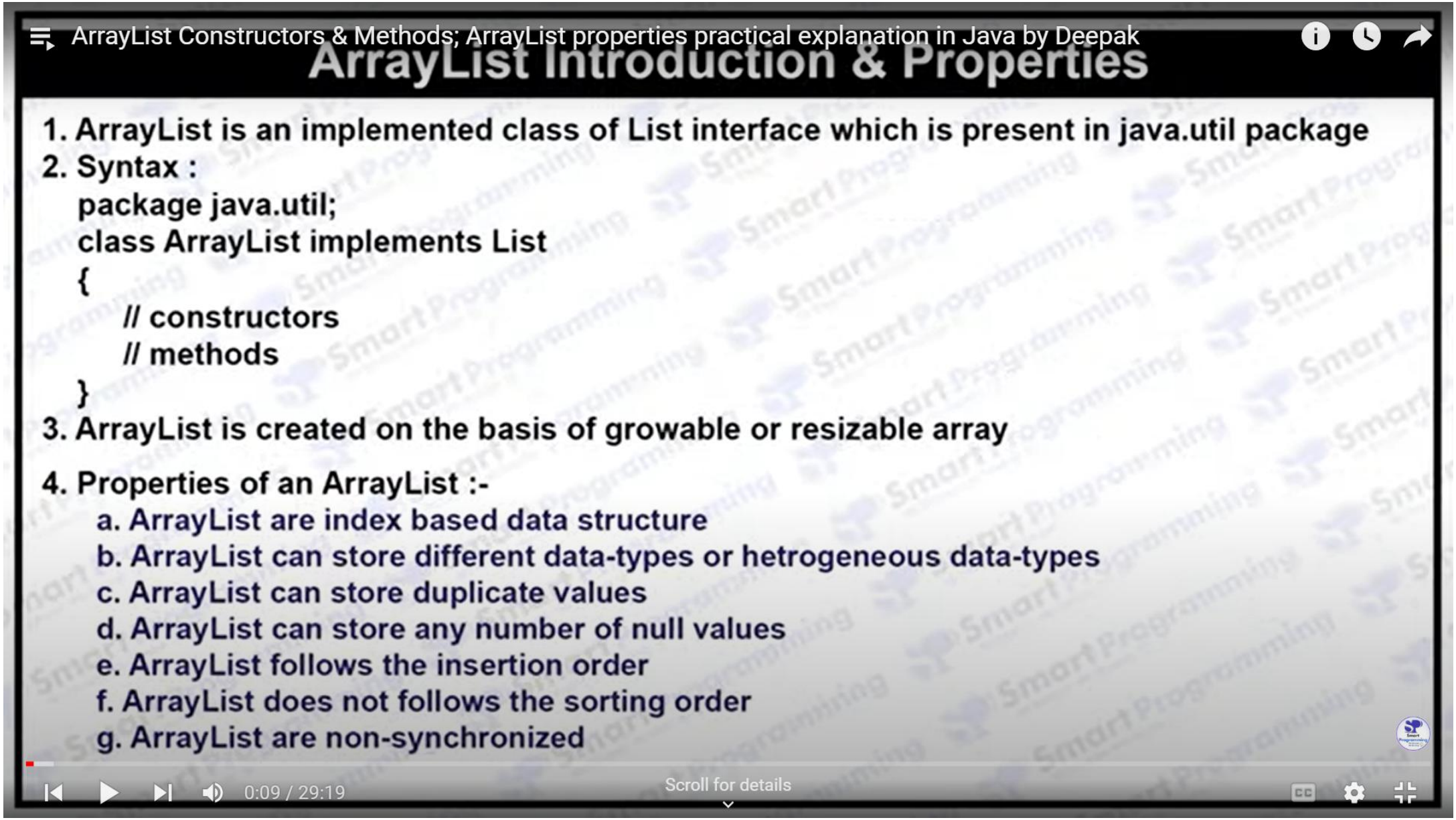
```
package java.util;
class ArrayList implements List
{
    // constructors
    // methods.
}
```

2. ArrayList is created on the basis of growable or resizable array.

Properties of ArrayList

1. ArrayList are index based DS
2. ArrayList can store different data-types or hetrogenous data-types.
3. ArrayList can store duplicate values.
4. ArrayList can store any number of null values.
5. ArrayList follows the insertion order.
6. ArrayList does not follows the sorting order.
7. ArrayList are non-Synchronized




ArrayList Constructors & Methods; ArrayList properties practical explanation in Java by Deepak
ArrayList Introduction & Properties
1. ArrayList is an implemented class of List interface which is present in java.util package
2. Syntax :
package java.util;
class ArrayList implements List
{
// constructors
// methods
}
3. ArrayList is created on the basis of growable or resizable array
4. Properties of an ArrayList :-
a. ArrayList are index based data structure
b. ArrayList can store different data-types or hetrogeneous data-types
c. ArrayList can store duplicate values
d. ArrayList can store any number of null values
e. ArrayList follows the insertion order
f. ArrayList does not follows the sorting order
g. ArrayList are non-synchronized
0:09 / 29:19
Scroll for details

ArrayList Constructors & Methods

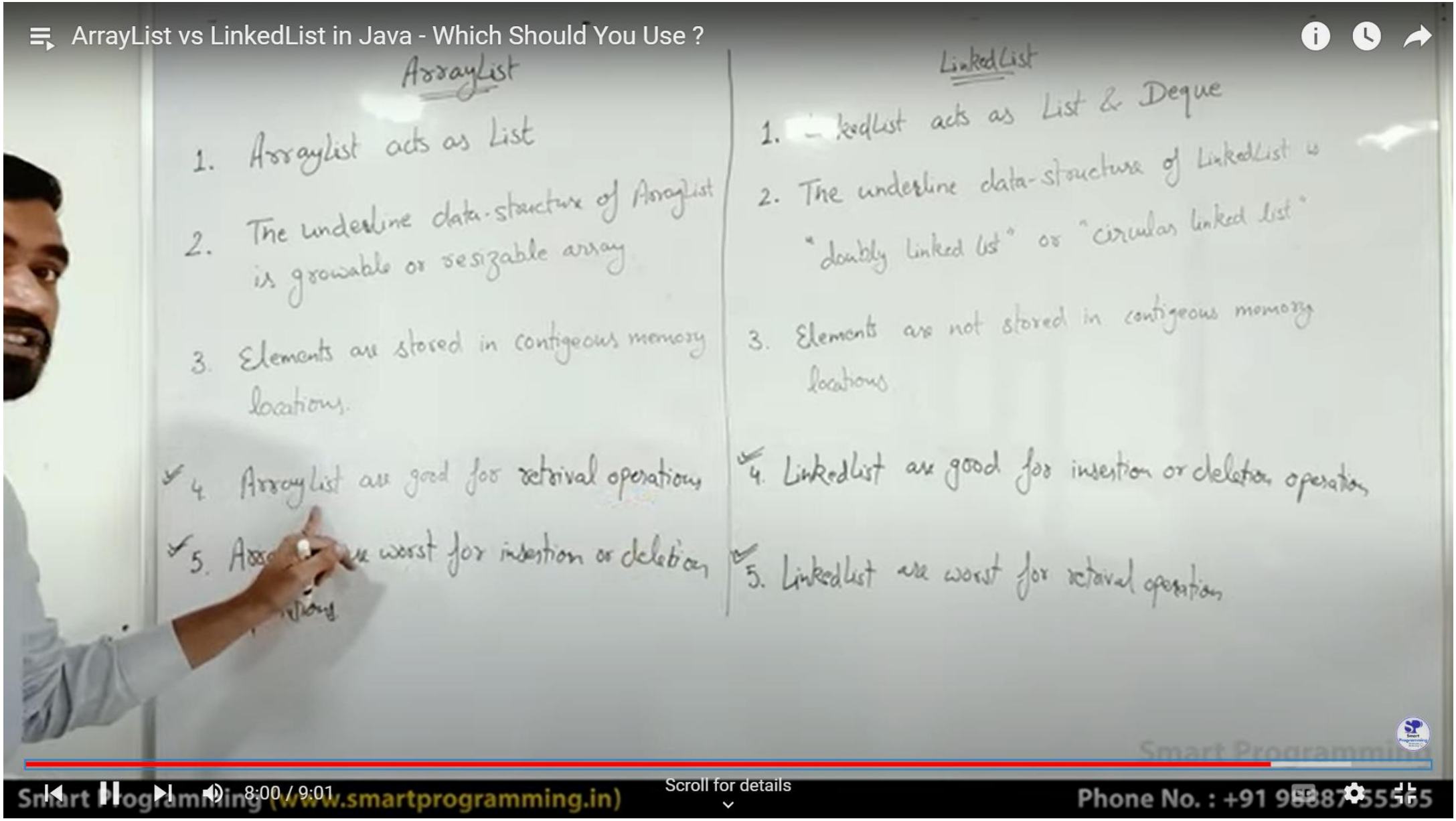
```
class ArrayList implements List
{
   // constructors
      1. ArrayList() :- It is used to build an empty array list with initial capacity as 10
      2. ArrayList(int capacity) : It is used to build an array list that has the specified initial capacity
      3. ArrayList(Collection c) :- It is used to build an array list that is initialized with the elements of the collection c

   // methods
      1. add() :- is used to add the elements or objects in an ArrayList or say colllection object
      2. addAll() :- addAll() method is used to add or append all the elements in the specified collection to the end of
                     ArrayList or collection object
      3. remove() :- It is used to remove the element at the specified position in the ArrayList
      4. removeAll() :- It is used to remove the elements from the current ArrayList which are contained in the specified
                        ArrayList or say collection object
      5. clear() :- It is used to remove all of the elements from the ArrayList or say collection object
      6. contains() :- It is used to check if the specified element is present in the given ArrayList or not, if its present it will
                       return true else false
      7. size() :- It is used to return the number of elements in the ArrayList i.e. the size
      8. get() :- It is used to get the element of a specified index within the ArrayList
      9. set() :- It is used to replace the element at the specified index position in the ArrayList with the specified element
      10. indexOf() :- It is used to return the index of first occurrence of the element in the ArrayList. If this list does not
                       contain the element then it will return -1
      11. iterator() :- It is used to get an iterator over the elements in this list in proper sequence
      // and many more methods
}
```

LinkedList Introduction & Properties

1. LinkedList is an implemented class of List interface which is present in java.util package
2. Syntax :

```
package java.util;
class LinkedList implements List, Deque
{
    // constructors
    // methods
}
```

3. LinkedList underline data-structure is “Doubly Linked List” or “Circular Linked List”
4. Properties of LinkedList :-
   a. LinkedList are index based data structure
   b. LinkedList can store different data-types or hetrogeneous data-types
   c. LinkedList can store duplicate values
   d. LinkedList can store any number of null values
   e. LinkedList follows the insertion order
   f. LinkedList does not follows the sorting order
   g. LinkedList are non-synchronized


ArrayList vs LinkedList in Java - Which Should You Use ?
ArrayList
1. Arraylist acts as List
2. The underline data-structure of Arraylist is growable or resizable array
3. Elements are stored in contigeous memory locations.
4. Arraylist are good for retrival operations
5. Arraylist are worst for insertion or deletion operations
LinkedList
1. LinkedList acts as List & Deque
2. The underline data-structure of LinkedList is "doubly linked list" or "circular linked list"
3. Elements are not stored in contigeous memory locations
4. LinkedList are good for insertion or deletion operations
5. LinkedList are worst for retrival operations
8:00 / 9:01
Scroll for details
Smart Programming (www.smartprogramming.in)
Phone No. : +91 98887 55565

ArrayList

1. ArrayList was introduced in JDK 1.2 version.
2. ArrayList is not legacy class.
3. ArrayList are non-synchronized collection.
4. ArrayList is not thread-safe
5. In case of ArrayList, application speed is fast.

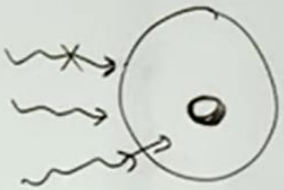
6. ArrayList does not guarantee for data-consistency
7. In case of ArrayList

```
newCapacity = (oldCapacity * 3)/2 + 1;
```

8. ArrayList does not provide any method to find its capacity.

Vector

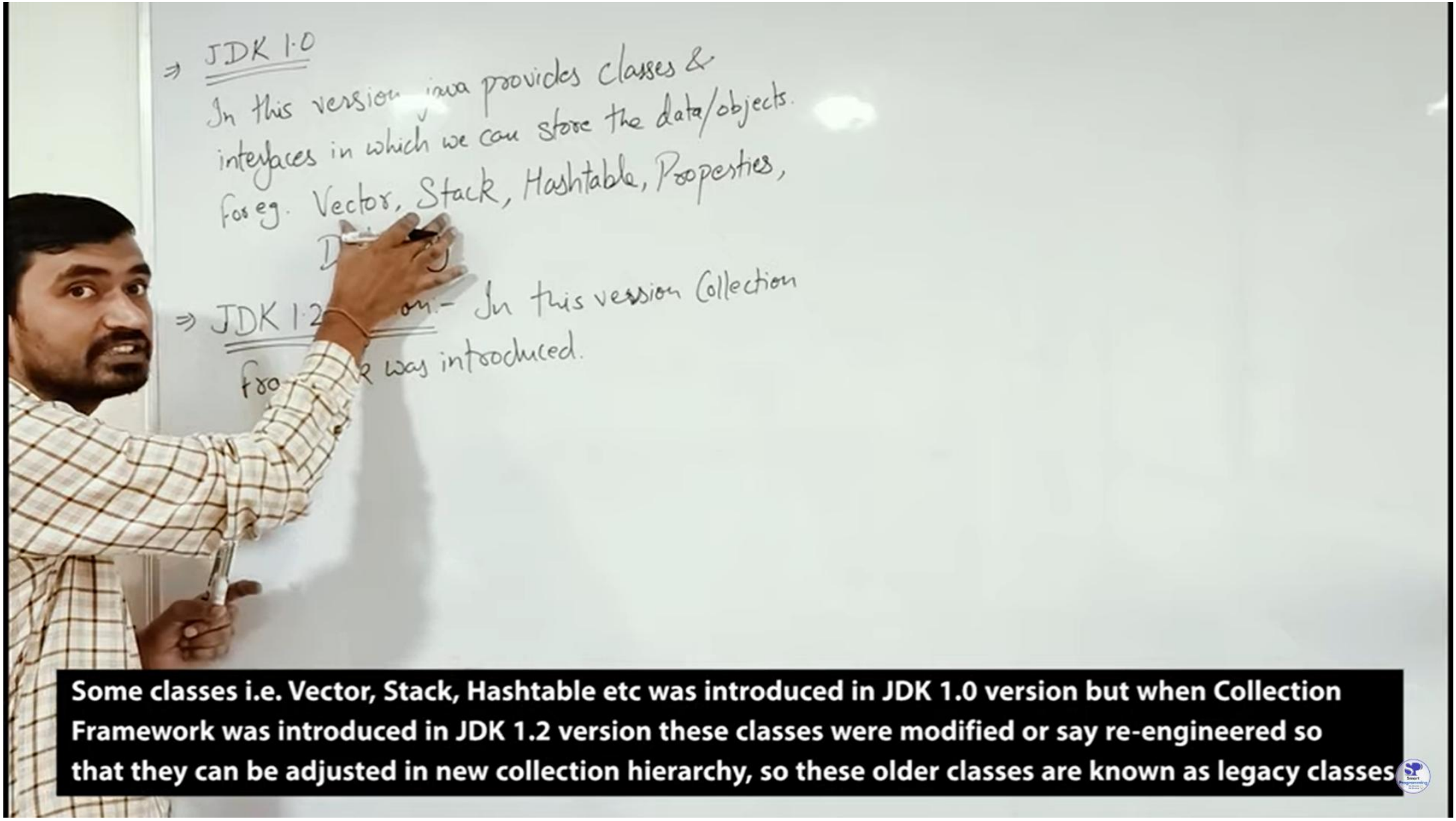
1. Vector was introduced in JDK 1.0 version.
2. Vector is legacy class
3. Vector are synchronized collection.
4. Vector is thread-safe.
5. In case of Vector, application speed is slow



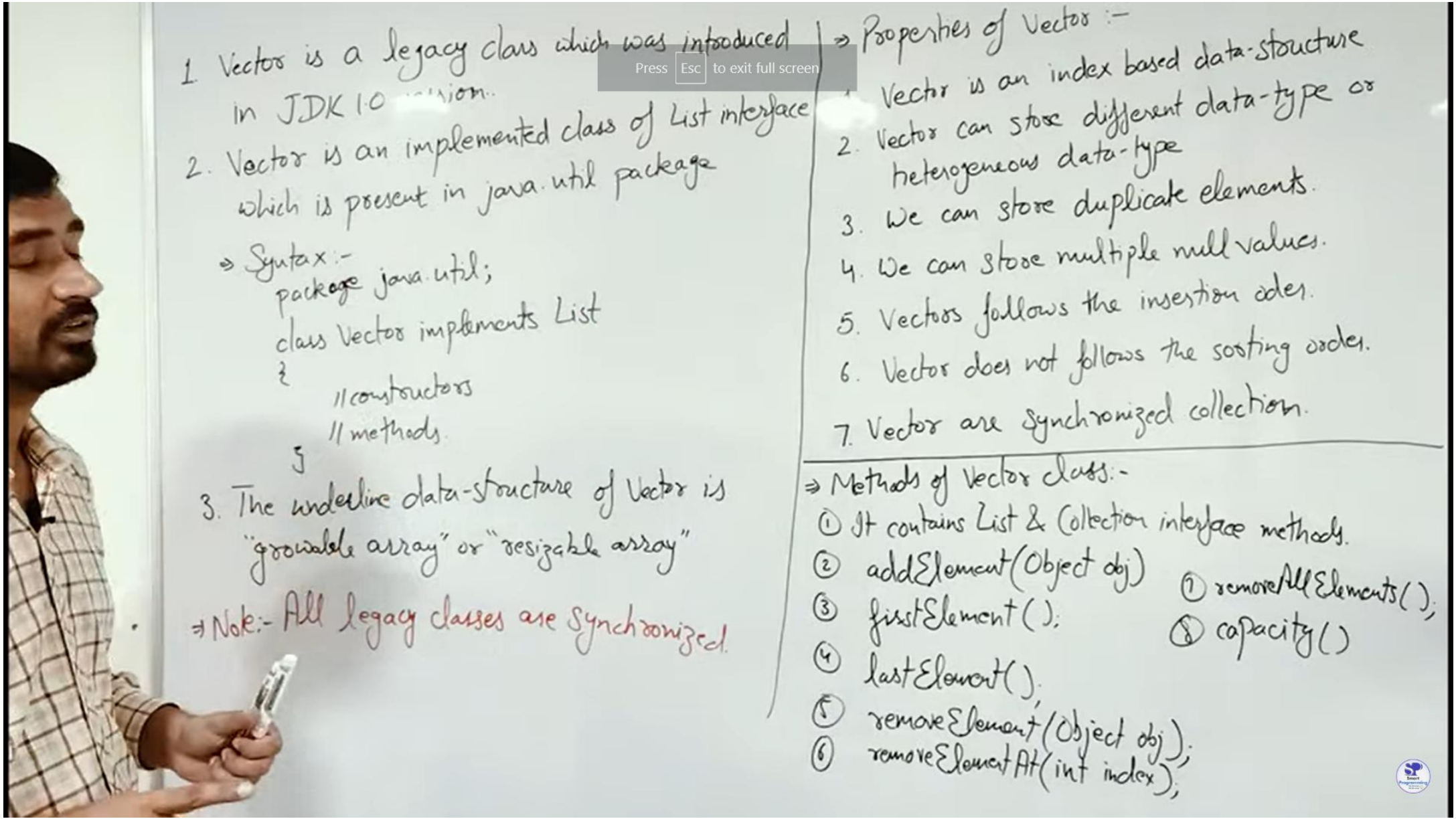
6. Vector provides the guarantee for data-consistency
7. In case of Vector

```
new_capacity = (old_capacity * 2);
```

8. Vector class provides a method ie "int capacity()" to find the capacity of Vector



Some classes i.e. Vector, Stack, Hashtable etc was introduced in JDK 1.0 version but when Collection Framework was introduced in JDK 1.2 version these classes were modified or say re-engineered so that they can be adjusted in new collection hierarchy, so these older classes are known as legacy classes


1. Vector is a legacy class which was introduced in JDK 1.0 version.
2. Vector is an implemented class of List interface which is present in java.util package
⇒ Syntax:-
package java.util;
class Vector implements List
{
// constructors
// methods
}
3. The underline data-structure of Vector is "growable array" or "resizable array"
⇒ Note:- All legacy classes are synchronized.
Press Esc to exit full screen
⇒ Properties of Vector :-
1. Vector is an index based data-structure
2. Vector can store different data-type or heterogeneous data-type
3. We can store duplicate elements.
4. We can store multiple null values.
5. Vectors follows the insertion order.
6. Vector does not follows the sorting order.
7. Vector are synchronized collection.
⇒ Methods of Vector class:-
① It contains List & Collection interface methods.
② addElement(Object obj)
③ firstElement();
④ lastElement();
⑤ removeElement(Object obj);
⑥ removeElementAt(int index);
⑦ removeAllElements();
⑧ capacity()

1. Stack is the legacy class which was introduced in JDK 1.0 version.

2. Stack is the child class of Vector class which is present in java.util package.

⇒ Syntax:-

```
package java.util;
class Stack extends Vector
{
    // constructors
    // methods
}
```

Constructor :-

① Stack()

⇒ Methods :-

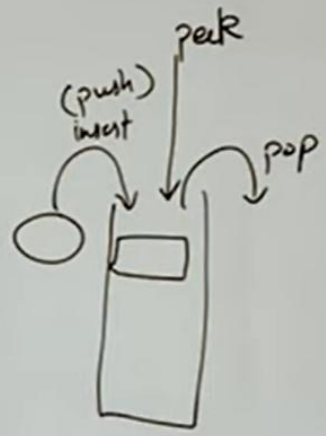
① push(-)

② pop()

③ peek()

④ search(-)

⑤ empty()



HashSet :-

① It is an implement class of Set interface which is present in java.util package

② Syntax :-

```
package java.util;
class HashSet implements Set
{
    // constructors
    // methods
}
```

③ HashSet underline data-structure is "Hashtable".

* HashSet is backed up by "Map".

④ HashSet was introduced in JDK 1.2 version

⇒ Properties of HashSet :-

① HashSet are not an index based data-structure They store the elements according to their "hashcode" value

② HashSet does not store duplicate elements.

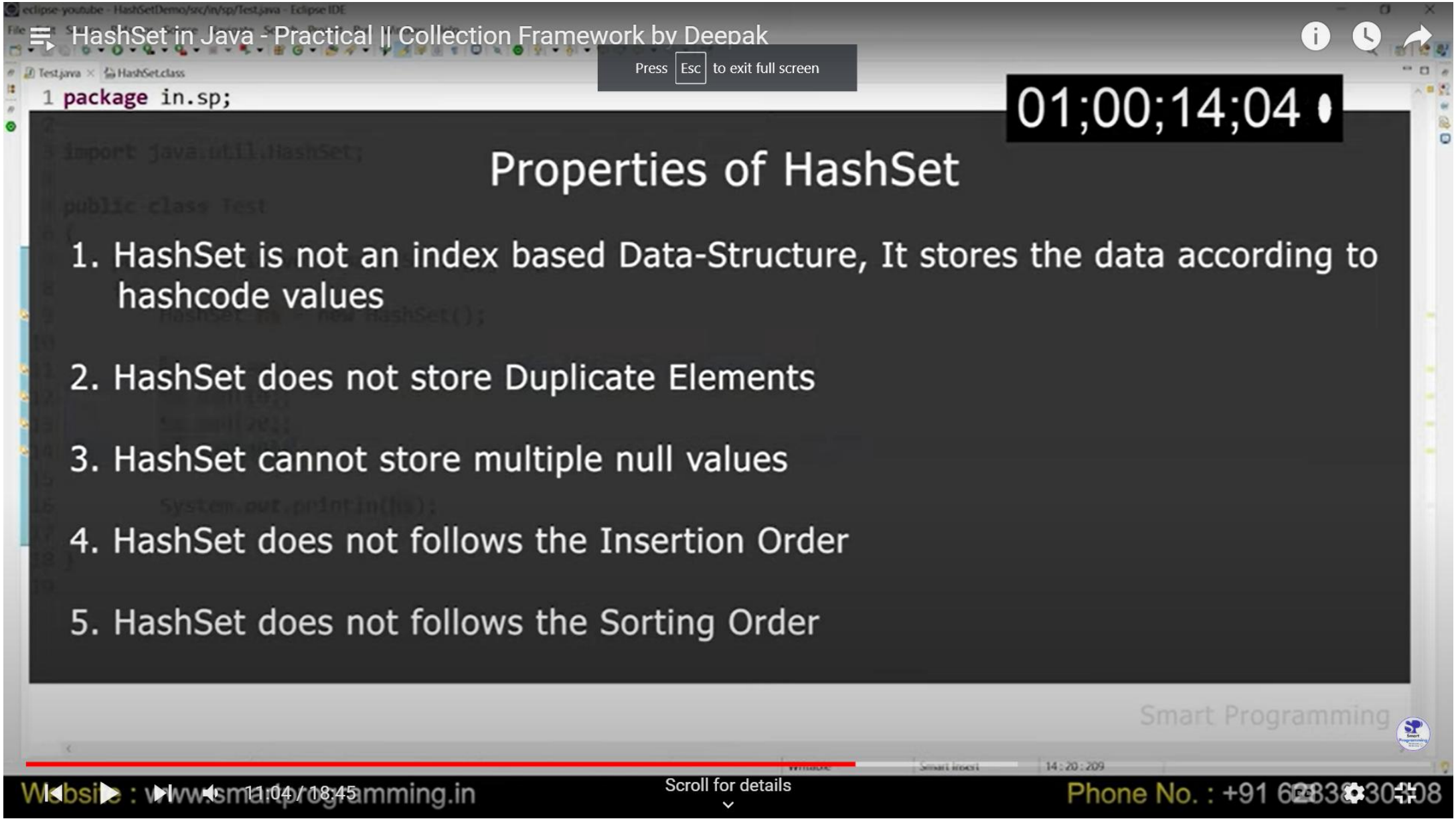
③ HashSet cannot store multiple null values

④ HashSet can store different data-types ie. heterogeneous elements.

⑤ HashSet does not follows the "insertion order".

⑥ HashSet does not follows the "sorting order"

⑦ HashSet are non-synchronized data-structure


HashSet in Java - Practical || Collection Framework by Deepak
Press Esc to exit full screen
01;00;14;04
Properties of HashSet
1. HashSet is not an index based Data-Structure, It stores the data according to hashcode values
2. HashSet does not store Duplicate Elements
3. HashSet cannot store multiple null values
4. HashSet does not follows the Insertion Order
5. HashSet does not follows the Sorting Order
Smart Programming
Website : www.smartprogramming.in
11:04 / 18:45
Scroll for details
Phone No. : +91

TreeSet :- ① TreeSet is the direct implement class of NavigableSet but indirectly implements t & Set interface

java.util;

reeSet implements NavigableSet

// constructors
methods

introduced in JDK 1.2 version.

⇒ Properties of TreeSet :-

① It is not an index based data-structure
② It does not follows the insertion order
③ It follows the sorting order.

① → 40
② → 60
③ → 20
④ → 30

compareTo(Object obj) is Comparable interface method which is used to compare the current object with the specified object. It returns as follows :-
=> +ve integer : if the current object is greater than the specified object
=> -ve integer : if the current object is less than the specified object
=> 0 : if the current object is equal to the specified object

TreeSet :- ① TreeSet is the direct implement class of NavigableSet but indirectly implements SortedSet & Set interface

② Syntax -

```
package java.util;
class TreeSet implements NavigableSet
{
    // constructors
    // methods
}
```

③ TreeSet was introduced in JDK 1.2 version.

④ TreeSet underline data-structure is "Balanced Tree".

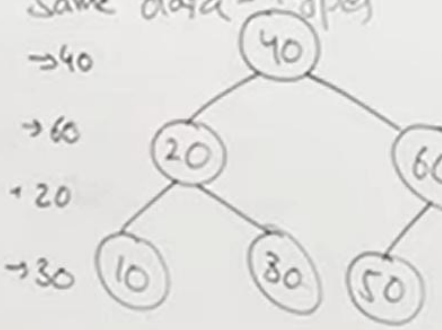
⇒ Properties of TreeSet :-

① It is not an index based data-structure

② It does not follows the insertion order

③ It follows the sorting order

④ It stores homogeneous elements ie. same data-types

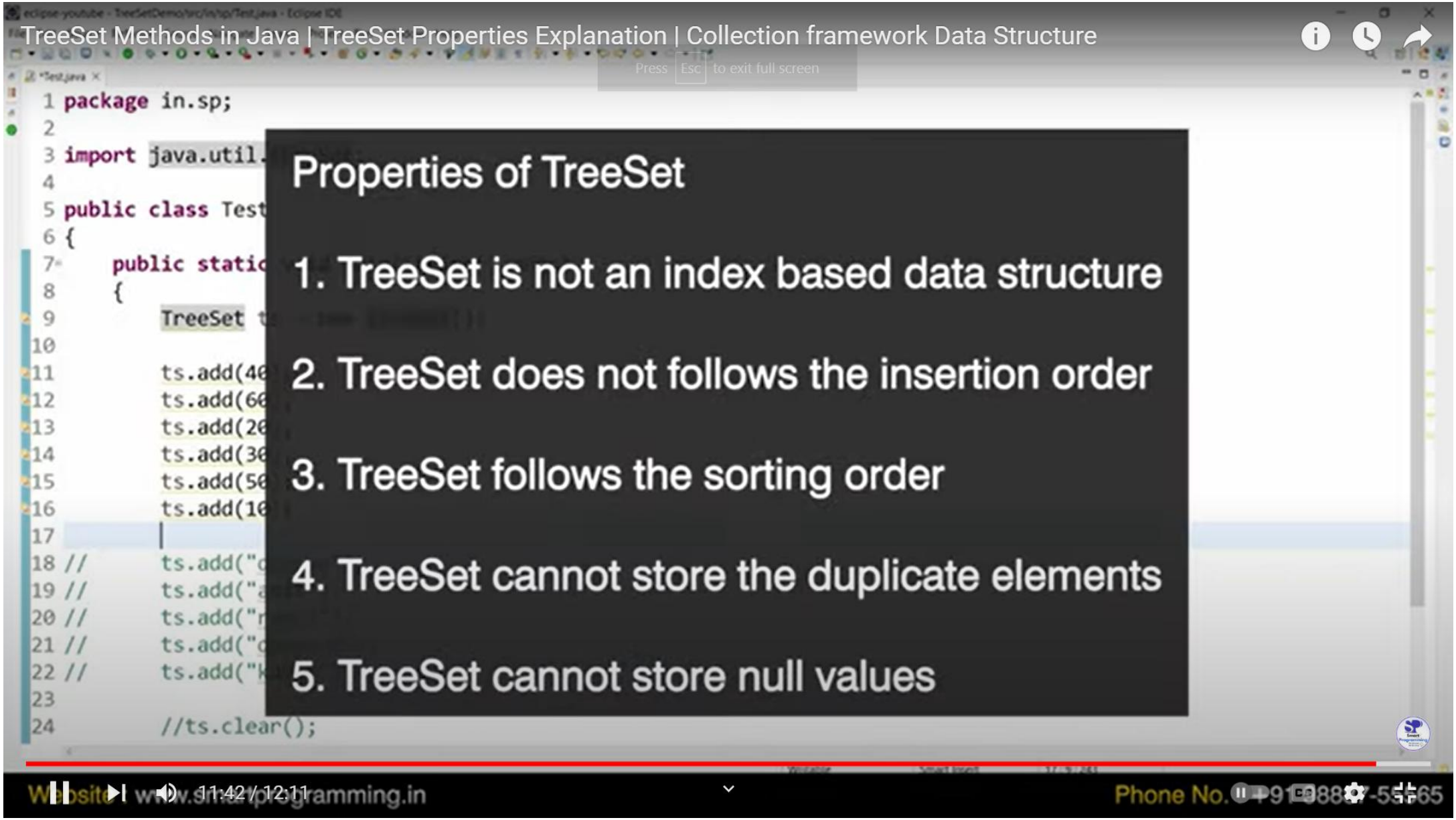


⑦ null value cannot be inserted in TreeSet

10 - 20 - 30 - 40 - 50 - 60

⑤ It cannot stores the duplicate elements

⑥ It is non-synchronized data-structure


TreeSet Methods in Java | TreeSet Properties Explanation | Collection framework Data Structure
Press Esc to exit full screen
package in.sp;
import java.util.
public class Test
{
public static
{
TreeSet
ts.add(40
ts.add(60
ts.add(20
ts.add(30
ts.add(50
ts.add(10
//ts.clear();
Properties of TreeSet
1. TreeSet is not an index based data structure
2. TreeSet does not follows the insertion order
3. TreeSet follows the sorting order
4. TreeSet cannot store the duplicate elements
5. TreeSet cannot store null values
11:42 / 12:11
Website : www.smartprogramming.in
Phone No. +91 8888 7 55565

⇒ Map :- ① Map is an interface which is present in "java.util" package

☆ Map does not inherit Collection interface

② Syntax :-

```
package java.util;
public interface Map
{
    // methods.
}
```

③ Map was introduced in JDK 1.2 version.

④ Hierarchy of Map interface.

⇒ Properties of Map :-

① Map stores the data in key-value pair

Map

key	value
101	deepak
102	amit
103	rahul
104	deepak

→ Entry

② In Map, keys should be unique but value can be duplicate

③ In Map, we can store maximum one null value in key but in values we can store any number of null values.

④ Map does not follows the sorting & insertion order.

⇒ HashMap :-

① HashMap is an implemented class of Map interface.

② Syntax :-

```
package java.util;
class HashMap implements Map
{
    // constructors
    // methods.
}
```

③ HashMap was introduced in JDK 1.2 version

④ HashMap underline data-structure is "Hashtable"

⇒ Properties of HashMap :-

① HashMap stores the value in key-value pair

※ Each key-value pair is known as Entry

key	value
101	Deepak → Entry
102	aaa
103	Rahul
104	Deepak

102 → aaa

② In HashMap keys should always be unique but values can be duplicate

③ HashMap contains max one null value in key but it can store multiple null values in value

④ HashMap can store heterogeneous elements

⑤ HashMap does not follows the sorting & insertion order

⑥ HashMap is non-synchronized data-structure

eclipse-youtube - HashMapDemo/src/Test.java - Eclipse IDE

File Edit Source Refactor Source Navigate Search Project Run Window Help

*Test.java

```
1 import java.util.HashMap;
2
3 public class Test
4 {
5     public static void main(String[] args)
6     {
7         HashMap hm = new HashMap();
8     }
9 }
10
```

HashMap() - HashMap
HashMap(int initialCapacity) - HashMap
HashMap(Map m) - HashMap
HashMap(int initialCapacity, float loadFactor) - HashMap
HashMap() Anonymous Inner Type
HashMap(int initialCapacity) Anonymous Inner Type
HashMap(int initialCapacity, float loadFactor) Anonymous Inner Type
HashMap(Map m) Anonymous Inner Type
asl - Apache Copyright License embedded in Java Comments
formatter-off - Disable formatter with formatter:off/on tags
new - create new object
nls - non-externalized string marker
Press 'Ctrl+Space' to show Template Proposals

Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).

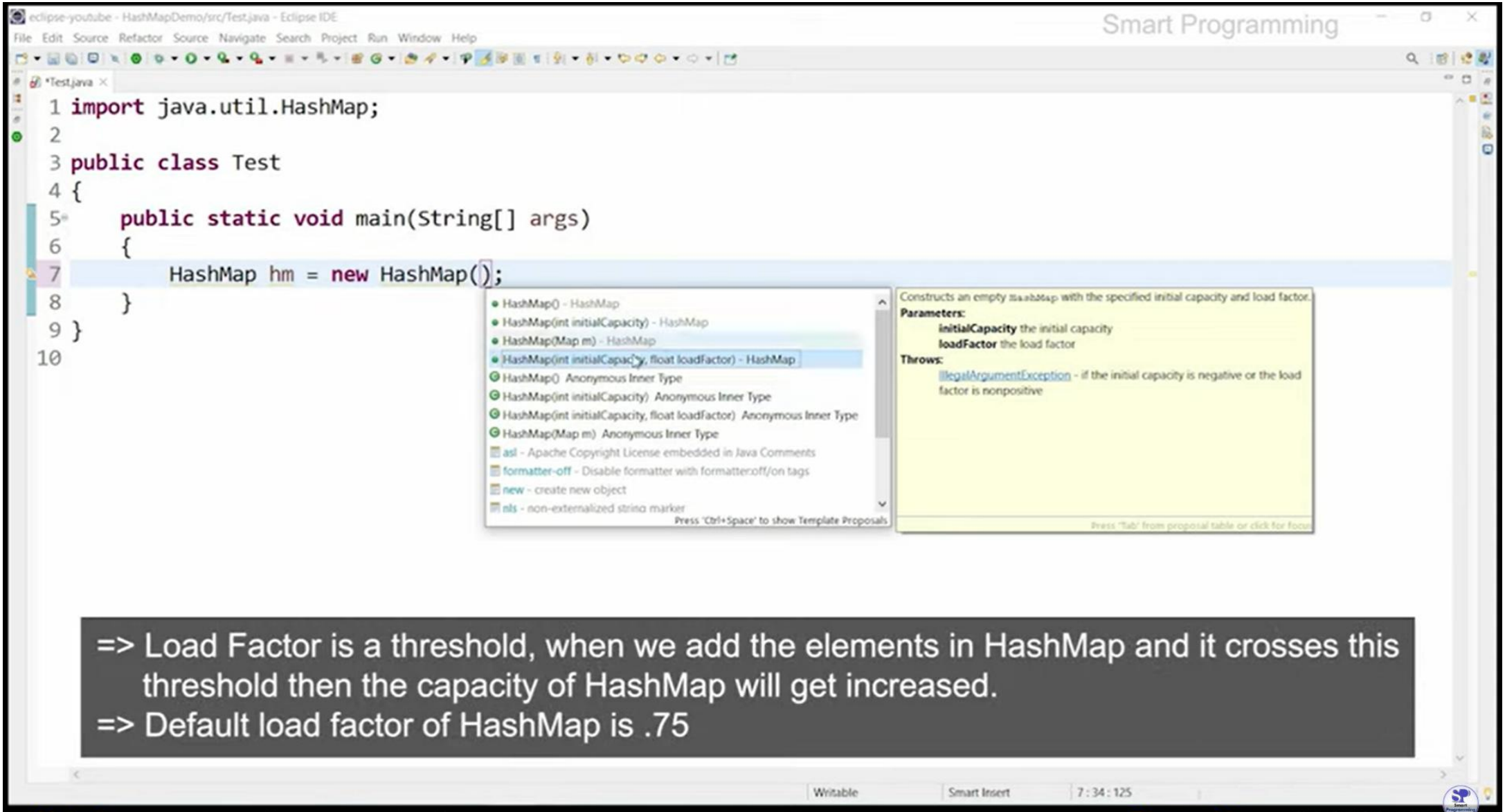
Parameters:
initialCapacity the initial capacity.

Throws:
IllegalArgumentException - if the initial capacity is negative.

Press 'Tab' from proposal table or click for focus

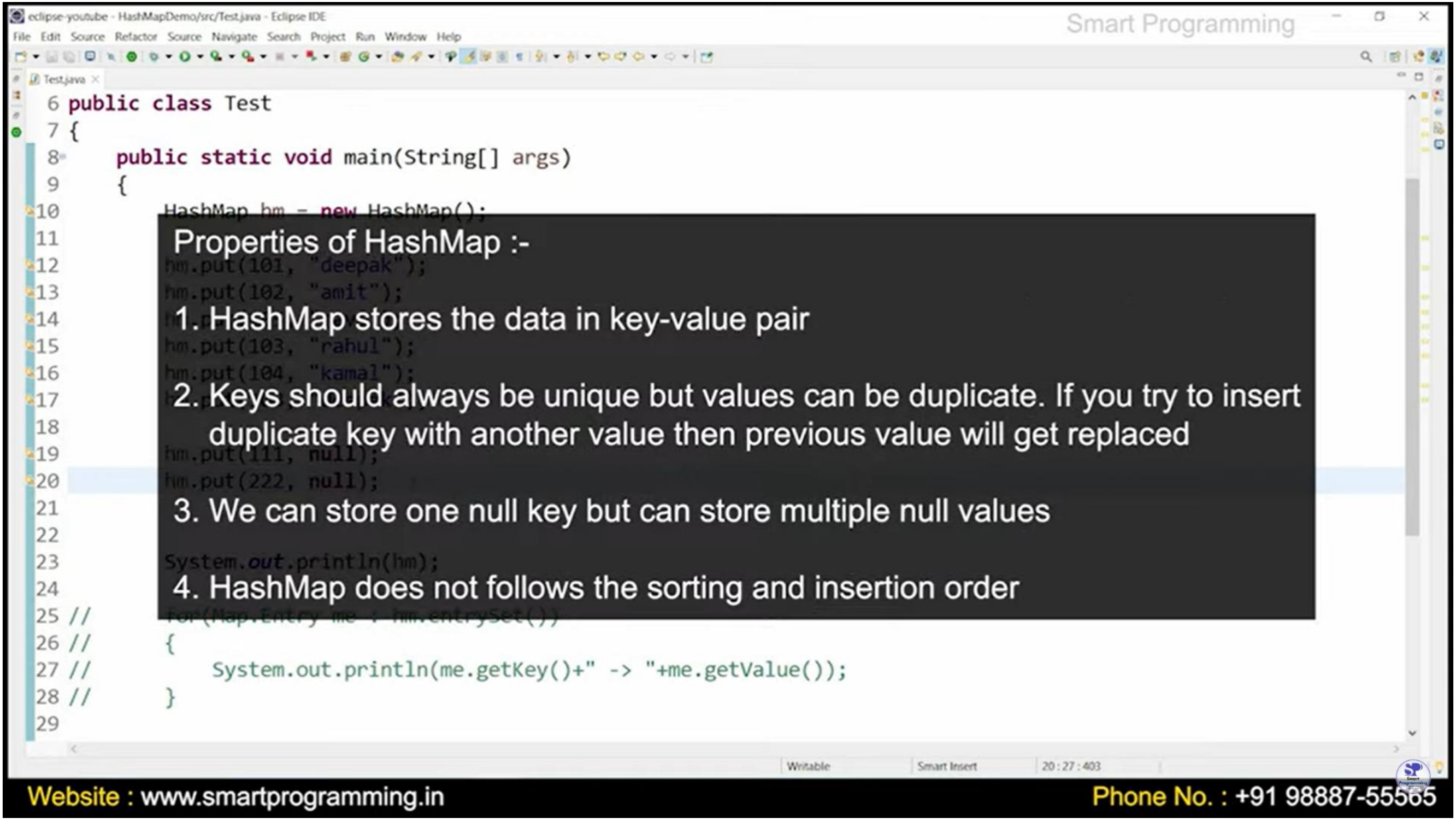
=> HashMap capacity means the number of buckets in the hash table (Note that, HashMap underline data-structure is hash table).
=> When we create HashMap, initialCapacity will be 16 which we can changed

Writable Smart Insert 7 : 34 : 125



=> Load Factor is a threshold, when we add the elements in HashMap and it crosses this threshold then the capacity of HashMap will get increased.

=> Default load factor of HashMap is .75


eclipse-youtube - HashMapDemo/src/Test.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Smart Programming
Test.java
public class Test
{
public static void main(String[] args)
{
HashMap hm = new HashMap();
hm.put(101, "deepak");
hm.put(102, "amit");
hm.put(103, "rahul");
hm.put(104, "kamal");
hm.put(111, null);
hm.put(222, null);
System.out.println(hm);
// for(Map.Entry me : hm.entrySet())
// {
// System.out.println(me.getKey()+" -> "+me.getValue());
// }
Properties of HashMap :-
1. HashMap stores the data in key-value pair
2. Keys should always be unique but values can be duplicate. If you try to insert duplicate key with another value then previous value will get replaced
3. We can store one null key but can store multiple null values
4. HashMap does not follows the sorting and insertion order
Writable
Smart Insert
20 : 27 : 403
Website : www.smartprogramming.in
Phone No. : +91 98887-55565

Properties of Red-Black Tree

1. Every node is either red or black color.
2. The root node is always black.
3. Every leaf node (NULL node) is black.
4. If a node is red, then both its children are black.
5. For each node, all paths from the node to its descendant leaf nodes contain the same number of black nodes.

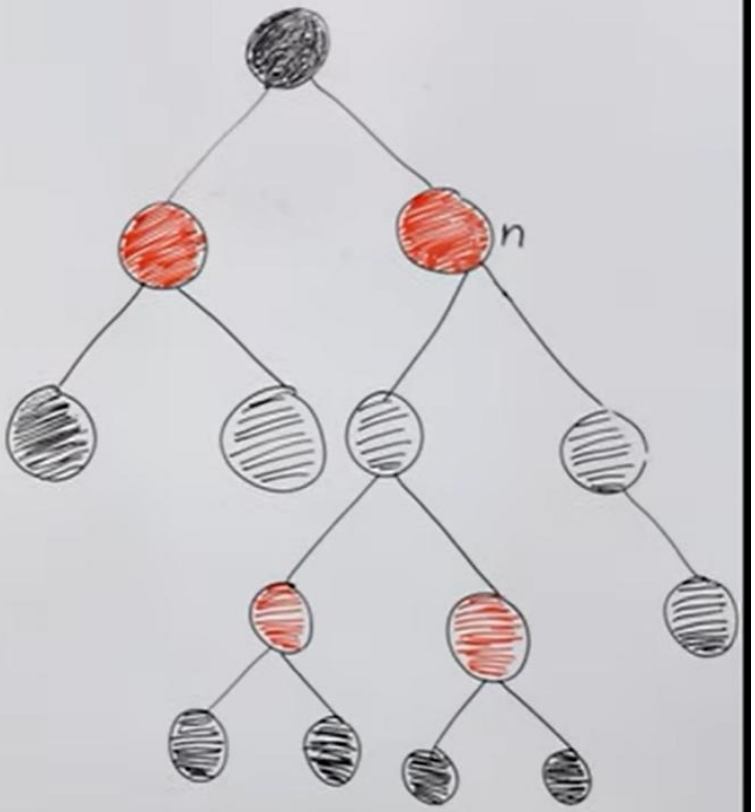
⇒ TreeMap :- ① It is an implemented class of NavigableMap but it also inherits the properties of SortedMap & Map interface

② Syntax:-

```
package java.util;
class TreeMap implements NavigableMap
{
    // constructors
    // methods
}
```

③ It was introduced in JDK 1.2 version

④ TreeMap underline data-structure is "Red-Black tree".



⇒ TreeMap :- ① It is an implemented class of NavigableMap but it also inherits the ... of SortedM... terface

② Syntax:-
package java.ut...
class Tree... ...s NavigableMap
{

③ ...sion

④ ...s

⇒ Properties of TreeMap :-

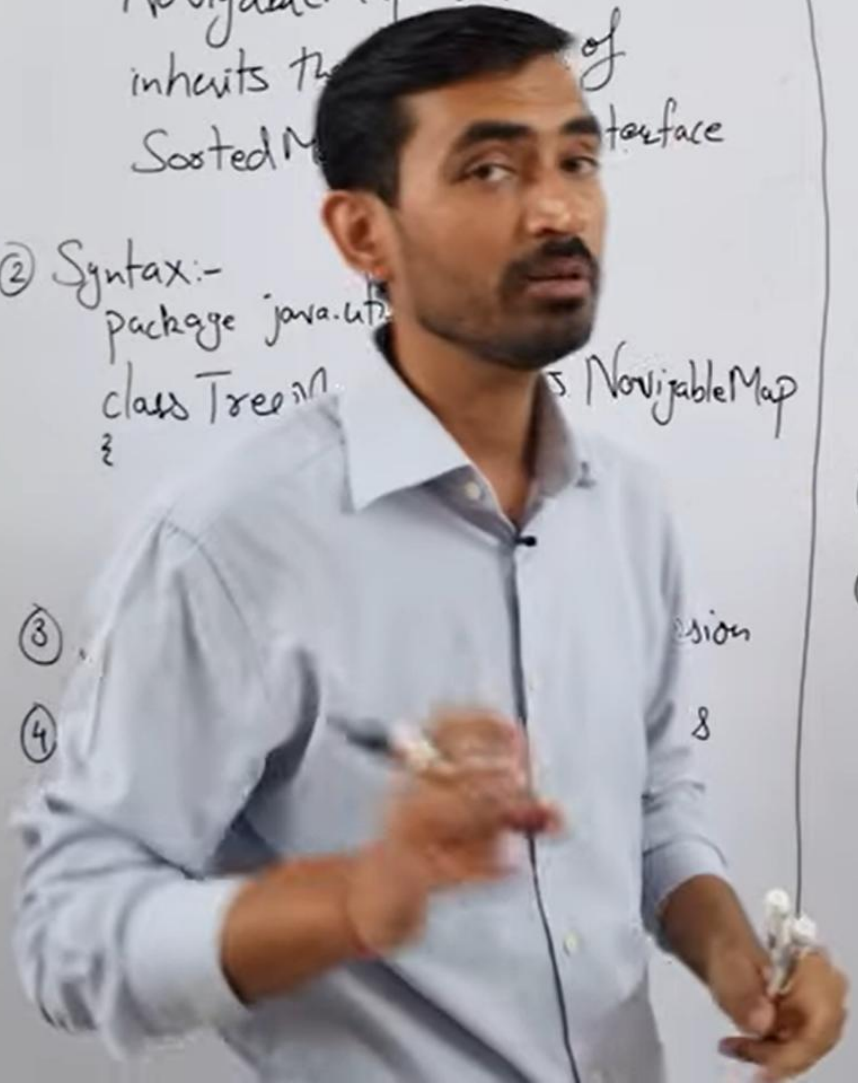
① We store the data in key-value pair in which keys should be unique but values can be duplicate

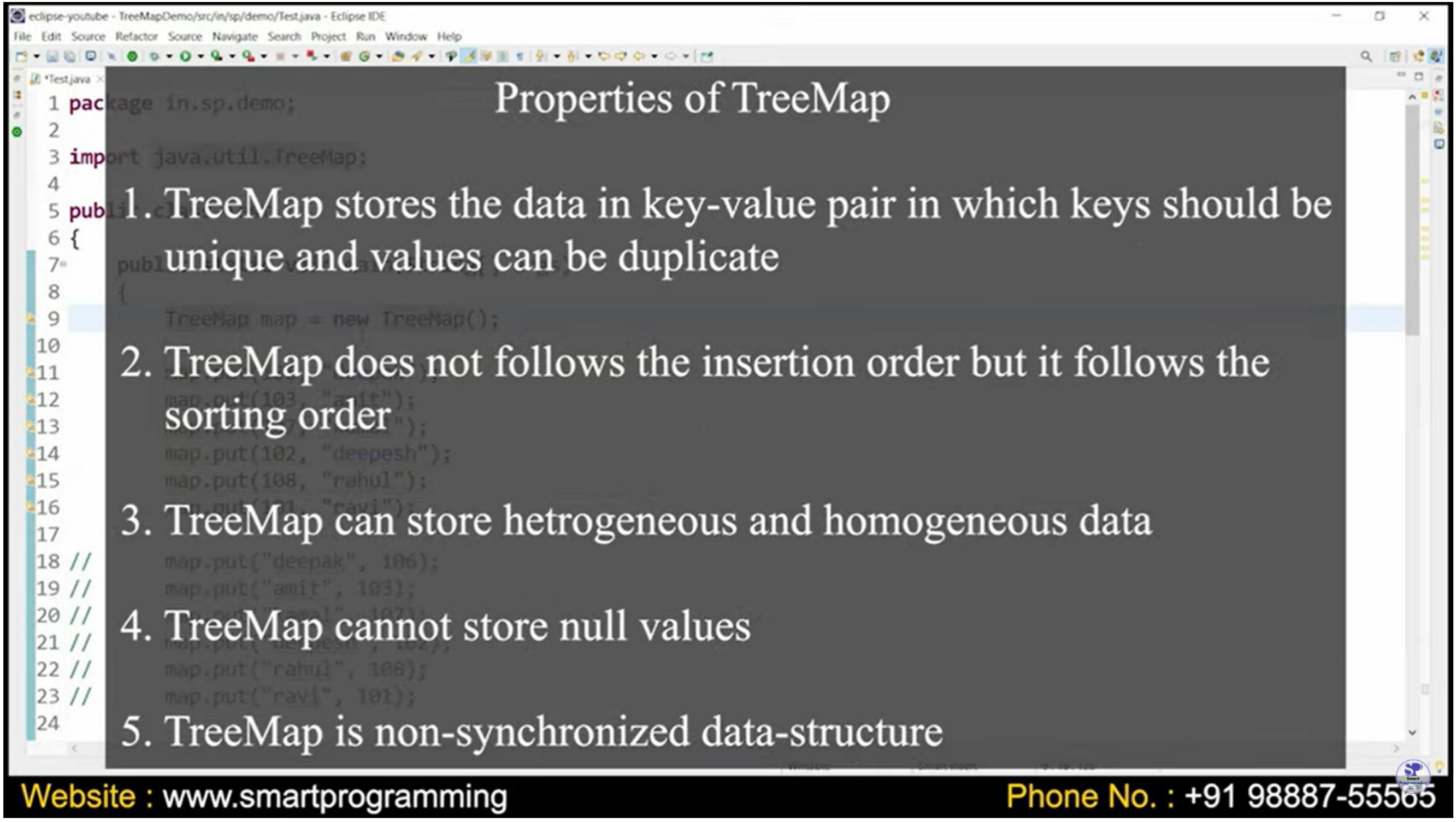
② TreeMap does not follows the insertion order but follows the sorting order wrt. keys

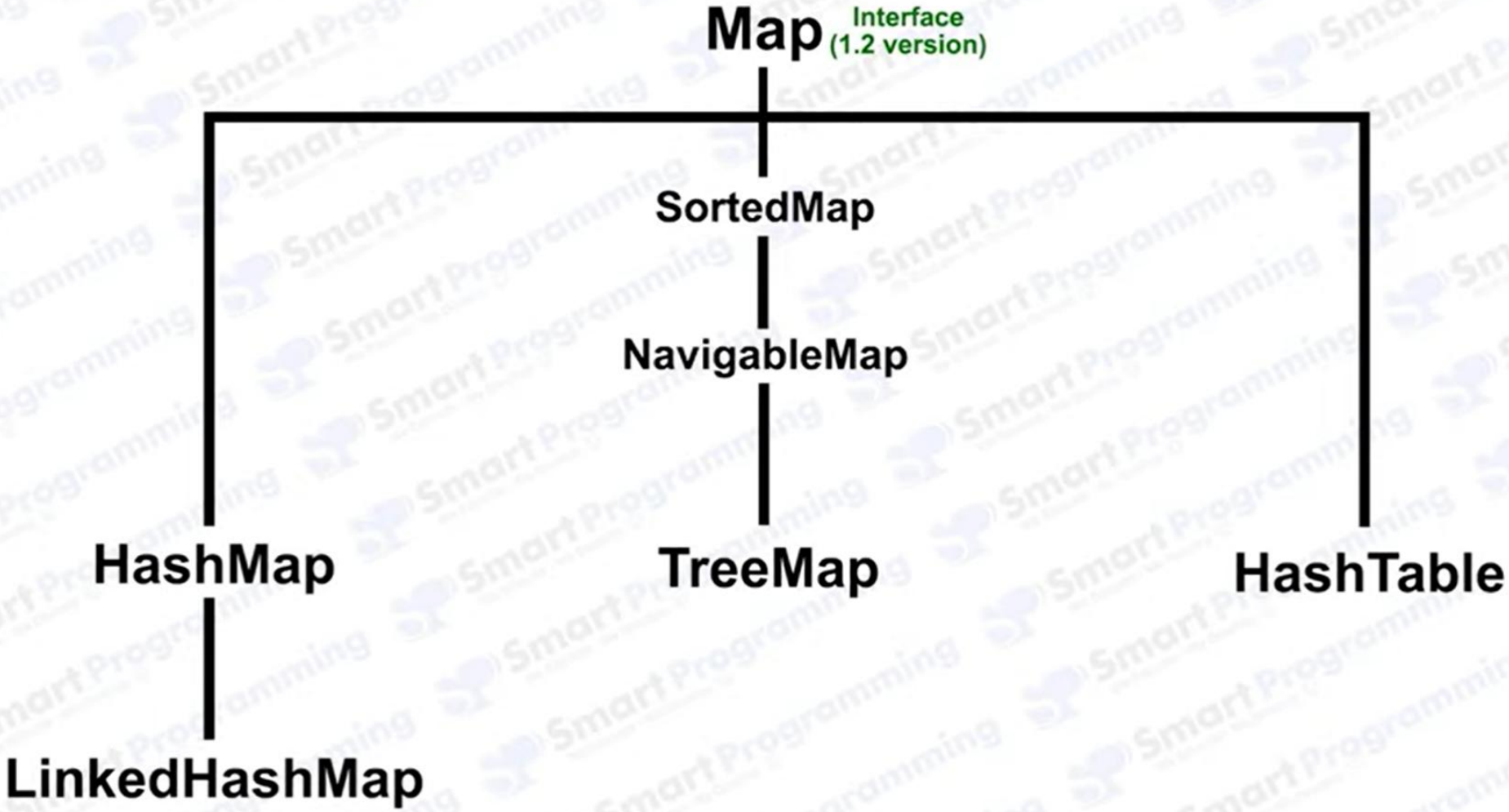
③ It can store homogenous & heterogeneous elements (default sorting nature)

④ TreeMap cannot store null values.

⑤ TreeMap is non-synchronized DS






Map
Interface
(1.2 version)
SortedMap
NavigableMap
HashMap
TreeMap
HashTable
LinkedHashMap

Properties of Hashtable

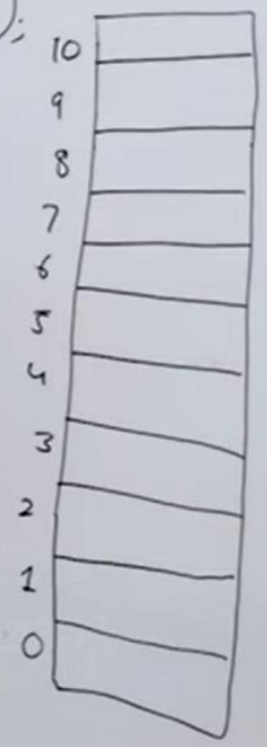
1. Hashtable contains the data in key-value pair & each key-value pair is known as entry
2. In Hashtable, keys should always be unique but values can be duplicate
3. Hashtable can store hetrogeneous elements or different type of elements at key position
4. We cannot store null value in Hashtable
5. Hashtable does not follows the insertion and sorting order
6. Hashtable are synchronized data-structure

⇒ Hashtable:- ① Hashtable is the direct implemented class of Map interface

tax:-
age java.util;
Hashtable implements Map
// constructors
// methods
class & was introduced
ion
S is hashtable

⇒ Working of Hashtable

- Hashtable ht=new Hashtable();
- Ht.put(10 6, "deepak");



In java, hash code is a unique integer value that is generated for every object

⇒ Hashtable:- ① Hashtable is the direct implemented class of Map interface

② Syntax:-

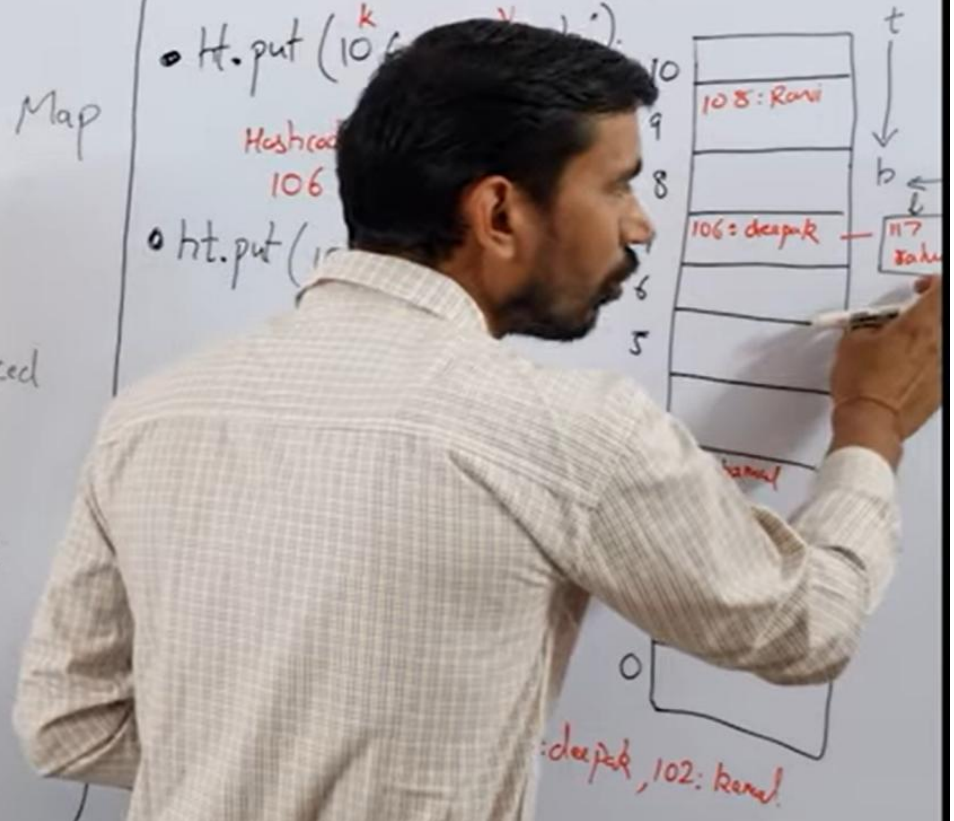
```
package java.util;
class Hashtable implements Map
{
    // constructors
    // methods
}
```

③ It is a legacy class & was introduced in JDK 1.0 version

④ Hashtable underline DS is hashtable

⇒ Working of Hashtable

- Hashtable ht=new Hashtable();
- ht.put(10 k, ...);
- ht.put(...);



⇒ Hashtable:- ① Hashtable is the direct implemented class of Map interface

wa.util;

shtable implements Map

structors

ethods

class & was introduced

ersion

DS is hashtable

⇒ Working of Hashtable

- Hashtable ht=new Hashtable();
- ht.put(106, "deepak");

Hashcode % 11 = rem

106 % 11 = 7

- ht.put(102, "kamal");

102 % 11 = 3

- ht.put(108, "Ravi");

108 % 11 = 9

- ht.put(117, "rahul");

117 % 11 = 7

- 108:Ravi, 117:rahul, 106:deepak, 102:kamal