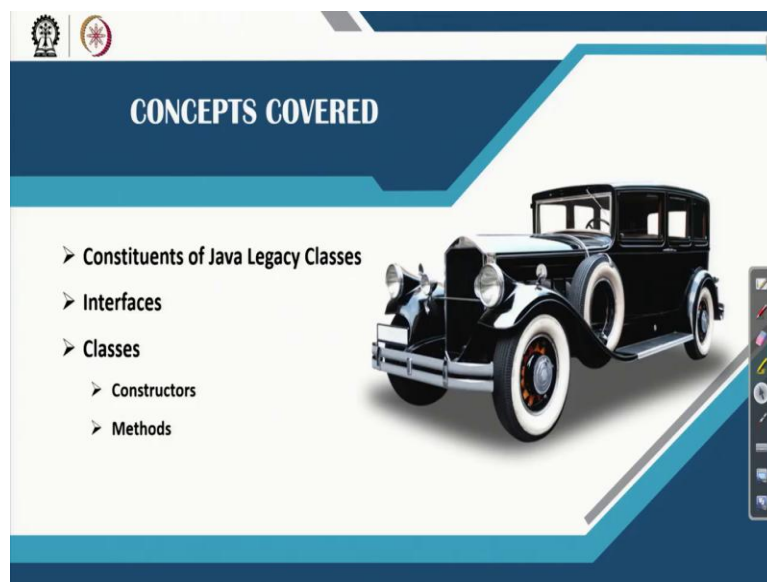


**Data Structures and Algorithms Using Java**  
**Professor Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture 10**  
**Topic: Java Legacy Classes**

In the discussion of collection framework, the discussion will not be complete if we do not include Java Legacy Classes. Actually Java Legacy Classes, I understand that it is not so much useful at the moment because the more better facilities is basically provided by the Java collection framework there, but Java Legacy Classes still many programmers like it.

(Refer Slide Time: 0:59)



There are two reasons for this, first of all it is fast, this means that it is very useful but second thing is that it has a more, what is called the sophistication in the context of the synchronized versus non-synchronized manner. What exactly we want to point out is that all the collection frame, collections that we have discussed under the categories of Java collection framework and Java map framework, they are basically not synchronized whereas all the collections if you maintain using Java Legacy Classes then they are synchronized.

Now, what is the difference between synchronized collection and non-synchronized collection? The synchronized collection means as you know the parallel execution so that means if two or more programs access the same collection, which is very much common in distributed

programming because if you store your collection in a server and then there are request from the different client that collection needs to be access, either modify or whatever it is there, then parallelism or concurrent execution is an important issue.

So, if you want to have your concurrent execution for your collection then definitely the collection framework, the latest collection framework is not suitable because they are non-synchronized, whereas the Java Legacy Classes, the collection to according to this Java Legacy Classes are synchronized that means the parallel program execution is possible. Now, let us, we have a brief idea about the different constituents in Java Legacy Classes.

(Refer Slide Time: 3:06)

The image shows a presentation slide with a white background and a blue header. The title "Java Legacy Classes: Background" is written in blue text. The slide is decorated with several icons: a coffee cup, a gear, a hard hat, a beaker, and a molecular structure. A small video inset in the bottom right corner shows a man speaking. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

**The background**

- Prior to the JCF (Java 2 and onward), the classes were known to meet the need as the JCF do for us are termed as **Java legacy classes**.
- The **Java legacy classes** are mentioned in the following.

Dictionary   Hashtable   Properties   Stack   Vector

- In addition, there is **one legacy interface** called **Enumeration**.

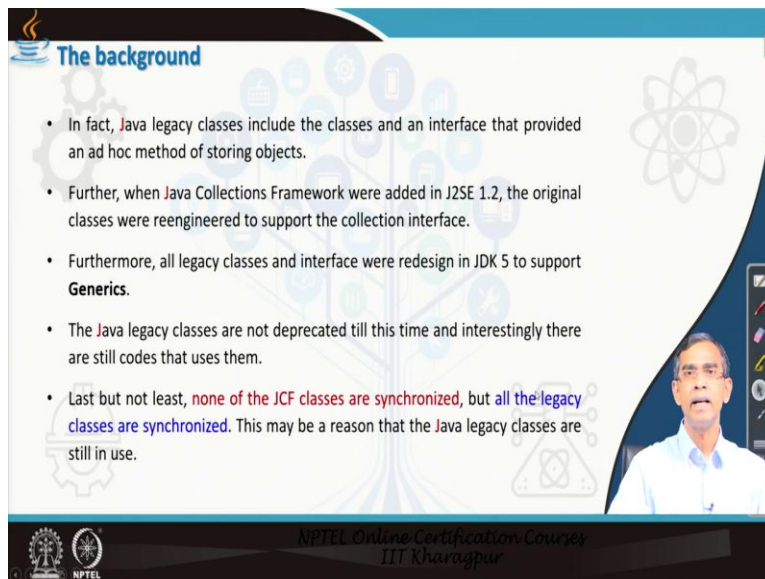
NPTEL Online Certification Courses  
IIT Kharagpur

So, here in this lecture we will little bit elaborate the different compositions. Now, before going to know about the different elements in it, I just want to say that what exactly the history of creating Java Legacy Classes is there. Now, prior to Java 2, actually Java Legacy Classes is the only ways handle a large collection and to facilitate the older collection sets. They proposed many classes, the Dictionary, Dictionary is very similar to the map that we have discussed.

Then Hashtable, it is basically is an another, what is call the way of viewing, it is just like a hash concept, that means if you want to access a particular object, then you have to have the key value for that object and then hash. It is just like map only. Properties is also similar map, so Dictionary, Hashtable, Properties, basically same as map only. And then Stack and Vector, Stack is basically the concept of just Queue and then Stack concept those are there, and Vector is basically a simple the index collection.

And here you can note that there is no more collection view using linkeds, index or sequential or tree form or whatever it is there. So, in that sense Java Legacy Class was not so exhaustive or elaborative or we can say it is not so much efficient compared to this a current the collection framework facilities is there. And now interface, again it is very simple most, it does not have many interfaces, only all classes are there, only one interface is there, this interface is called enumeration.

(Refer Slide Time: 5:08)



### The background

- In fact, Java legacy classes include the classes and an interface that provided an ad hoc method of storing objects.
- Further, when Java Collections Framework were added in J2SE 1.2, the original classes were reengineered to support the collection interface.
- Furthermore, all legacy classes and interface were redesign in JDK 5 to support **Generics**.
- The Java legacy classes are not deprecated till this time and interestingly there are still codes that uses them.
- Last but not least, **none of the JCF classes are synchronized, but all the legacy classes are synchronized**. This may be a reason that the Java legacy classes are still in use.

NPTEL Online Certification Course  
IIT Kharagpur

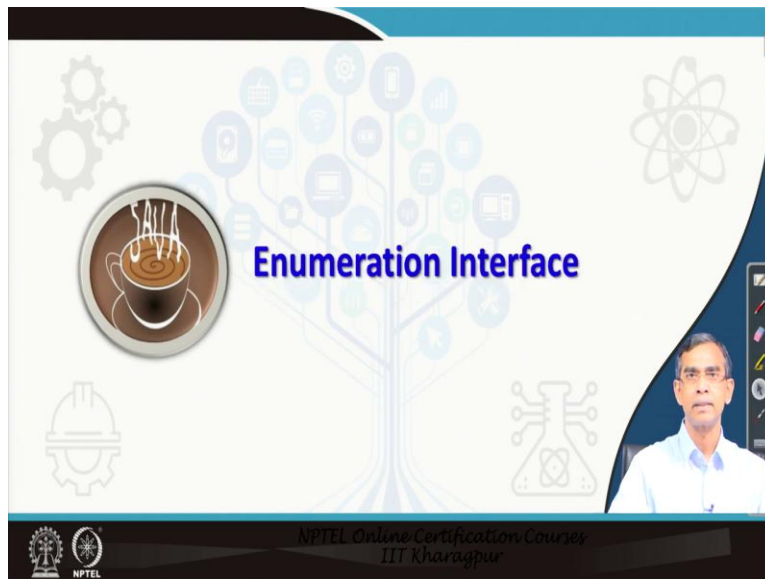


## Java Legacy Classes

NPTEL Online Certification Course  
IIT Kharagpur

So, this is basically the idea about this one and this is basically main important that user still likes not only because of its simplicity, but because of that it allows you to access the collection in a synchronized manner. And as it is synchronized manner, so complexities those are there, in case of collection framework is not here, or in other word, collection framework is so complex, so different mechanism, or so different structures or views are there, so the synchronization implementation a bit difficult that is why Java developer carefully ignore it but it retains in its Java Legacy Classes.

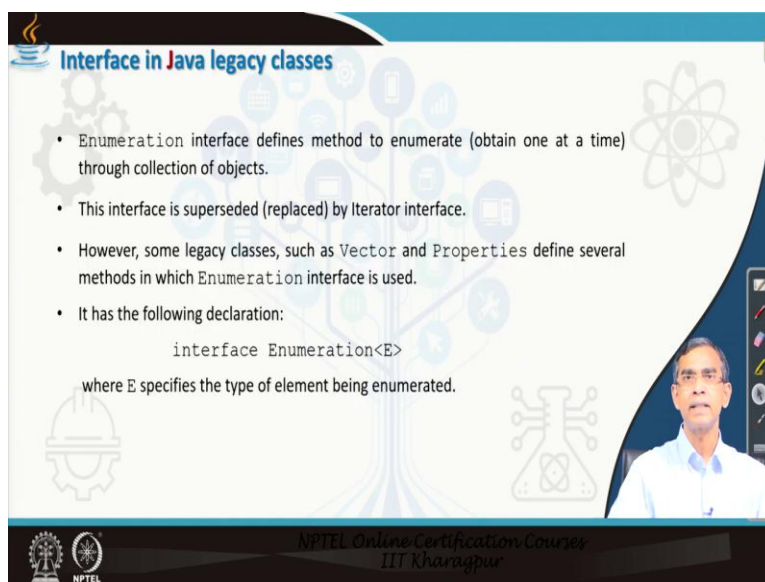
(Refer Slide Time: 5:48)



## Enumeration Interface

NPTEL Online Certification Course  
IIT Kharagpur

The slide features a central graphic of a tree where the branches are composed of various technology-related icons such as a smartphone, a laptop, a Wi-Fi symbol, and a gear. To the left of the tree is a circular icon of a steaming coffee cup. The background is light blue with faint icons of a gear, a hard hat, and a chemical flask. A small video inset of a man in a white shirt is visible in the bottom right corner.



## Interface in Java legacy classes

- Enumeration interface defines method to enumerate (obtain one at a time) through collection of objects.
- This interface is superseded (replaced) by Iterator interface.
- However, some legacy classes, such as `Vector` and `Properties` define several methods in which Enumeration interface is used.
- It has the following declaration:

```
interface Enumeration<E>
```

where E specifies the type of element being enumerated.

NPTEL Online Certification Course  
IIT Kharagpur

The slide features a central graphic of a tree where the branches are composed of various technology-related icons such as a smartphone, a laptop, a Wi-Fi symbol, and a gear. The background is light blue with faint icons of a gear, a hard hat, and a chemical flask. A small video inset of a man in a white shirt is visible in the bottom right corner.

### Methods declared in Enumeration Interface

Method	Description
boolean hasMoreElements()	It returns true while there are still more elements to extract, and returns false when all the elements have been enumerated.
Object nextElement()	It returns the next object in the enumeration i.e. each call to nextElement() method obtains the next object in the enumeration. It throws NoSuchElementException when the enumeration is complete.

Table 10.1: The methods declared by Enumeration interface

NPTEL Online Certification Courses  
IIT Kharagpur

Now, let us first discuss about the enumeration interface. It basically create an enumerated collection, enumerated collection means we have that user define datatype that enumerated type, and it basically useful for creating a new enumerate, it is an interface so no object of this type can be created but if you can create any collection that can be stored in a enumeration list actually, that is the way that you can do it.

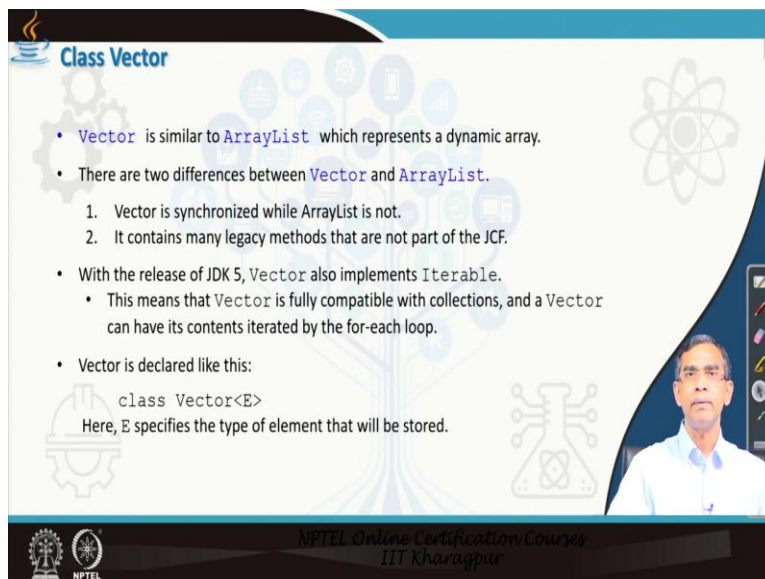
And it has basically main purpose about traversing a particular collection. So, for this traversing there is a methods, those are there, is an interface method, all these interface methods are basically available to other methods, so it has more elements and next element are the two methods are there.

(Refer Slide Time: 6:36)



**Class Vector**

NPTEL Online Certification Course  
IIT Kharagpur



**Class Vector**

- **Vector** is similar to **ArrayList** which represents a dynamic array.
- There are two differences between **Vector** and **ArrayList**.
  1. **Vector** is synchronized while **ArrayList** is not.
  2. It contains many legacy methods that are not part of the JCF.
- With the release of JDK 5, **Vector** also implements **Iterable**.
  - This means that **Vector** is fully compatible with collections, and a **Vector** can have its contents iterated by the for-each loop.
- **Vector** is declared like this:

```
class Vector<E>
```

Here, E specifies the type of element that will be stored.

NPTEL Online Certification Course  
IIT Kharagpur

Now, let us see the class vector. It is a very important one on concept, this concept later on has been incorporated in Java collection framework as the name there is called the ArrayList. So, which is ArrayList in Java collection framework is basically the vector in Java Legacy Class. Now, vector has very simplicity, vector is basically an array, that means a indexed mechanism to store the data.

(Refer Slide Time: 7:08)

**Constructors declared in Vector class**

Constructor	Description
Vector ()	This creates a default vector, which has an initial size of 10.
Vector (int size)	This creates a vector whose initial capacity is specified by size.
Vector (int size, int incr)	This creates a vector whose initial capacity is specified by size and whose increment is specified by incr. The increment specifies the number of elements to allocate each time when a vector is resized for addition of objects.
Vector (Collection c)	This creates a vector that contains the elements of collection c.

**Table 10.2:** The constructors defined by Vector class

NPTEL Online Certification Course  
IIT Kharagpur

You can store any store any type of objects into this vector and it has only few simple what is called the constructors, the default constructor you do not have to mention anything, it will create a collection of type vector and then if you can mention the size of the vector, initial size. You can also mention initial size and by which the increment, that automatically the vector will grow. Also a vector can be created, a collection of type vector can be created having an existing collection say C. So, if you can give an input of the existing collection then a vector can be created then. So, these are the different constructors under this vector class.



(Refer Slide Time: 7:55)

### Methods defined in Vector class

Method	Description
<code>void addElement(E element)</code>	The object specified by <i>element</i> is added to the vector.
<code>int capacity()</code>	Returns the capacity of the vector.
<code>Object clone()</code>	Returns a duplicate of the invoking vector.
<code>boolean contains(Object element)</code>	Returns <b>true</b> if <i>element</i> is contained by the vector, and returns <b>false</b> if it is not.
<code>void copyInto(Object array[])</code>	The elements contained in the invoking vector are copied into the array specified by <i>array</i> .
<code>E elementAt(int index)</code>	Returns the element at the location specified by <i>index</i> .
<code>Enumeration&lt;E&gt; elements()</code>	Returns an enumeration of the elements in the vector.
<code>void ensureCapacity(int size)</code>	Sets the minimum capacity of the vector to <i>size</i> .
<code>E firstElement()</code>	Returns the first element in the vector.
<code>int indexOf(Object element)</code>	Returns the index of the first occurrence of <i>element</i> . If the object is not in the vector, -1 is returned.
<code>int indexOf(Object element, int start)</code>	Returns the index of the first occurrence of <i>element</i> at or after <i>start</i> . If the object is not in that portion of the vector, -1 is returned.
<code>void insertElementAt(E element, int index)</code>	Adds <i>element</i> to the vector at the location specified by <i>index</i> .
<code>boolean isEmpty()</code>	Returns <b>true</b> if the vector is empty, and returns <b>false</b> if it contains one or more elements.

Table 10.3: The methods defined by Vector class (continued...)

NPTEL Online Certification Course  
IIT Kharagpur

### Methods defined in Vector class

Method	Description
<code>E lastElement()</code>	Returns the last element in the vector.
<code>int lastIndexOf(Object element)</code>	Returns the index of the last occurrence of <i>element</i> . If the object is not in the vector, -1 is returned.
<code>int lastIndexOf(Object element, int start)</code>	Returns the index of the last occurrence of <i>element</i> before <i>start</i> . If the object is not in that portion of the vector, -1 is returned.
<code>void removeAllElements()</code>	Empties the vector. After this method executes, the size of the vector is zero.
<code>boolean removeElement(Object element)</code>	Removes <i>element</i> from the vector. If more than one instance of the specified object exists in the vector, then it is the first one that is removed. Returns <b>true</b> if successful and <b>false</b> if the object is not found.
<code>void removeElementAt(int index)</code>	Removes the element at the location specified by <i>index</i> .
<code>void setElementAt(E element, int index)</code>	The location specified by <i>index</i> is assigned <i>element</i> .
<code>void setSize(int size)</code>	Sets the number of elements in the vector to <i>size</i> . If the new size is less than the old size, elements are lost. If the new size is larger than the old size, null elements are added.
<code>int size()</code>	Returns the number of elements currently in the vector.
<code>String toString()</code>	Returns the string equivalent of the vector.
<code>void trimToSize()</code>	Sets the vector's capacity equal to the number of elements that it currently holds.

Table 10.3: The methods defined by Vector class

NPTEL Online Certification Course  
IIT Kharagpur

Now, so far method is concerned I again reiterate that all the methods are very similar to the method that we have discussed in the context of Java collection framework that is there in collection interface, like here add element, capacity, then cloning. Cloning means if you want to create from the vector to new array. If you given an array as an object, you can copy all these object into a vector collection that methods is there.

And then like insert, indexOf, knowing the status, whether a particular vector collection is empty or if you want to remove an element, if you want to see what is a current size, all these things are

basically are there. So, all these operations related to the insertion, deletion, traversal, and modification and knowing the status, so these methods are defined there in vector class.

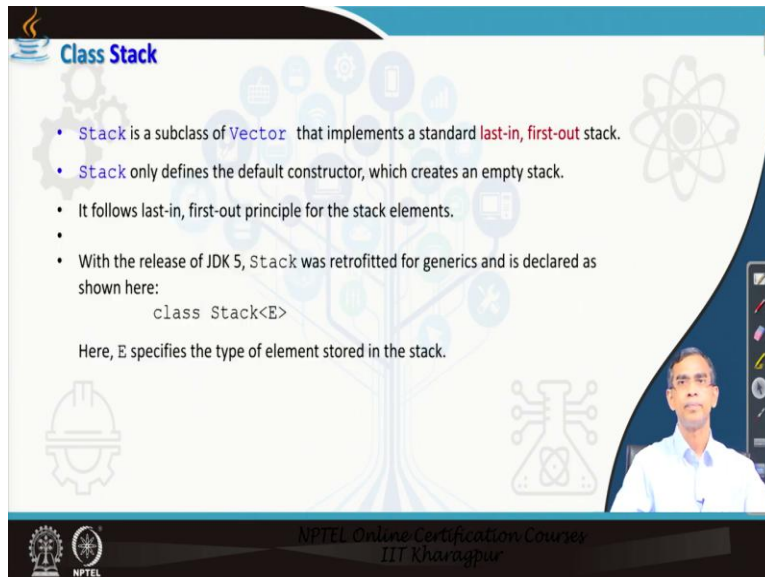
(Refer Slide Time: 8:51)



Now, you can note one thing the stack. Stack is very important data structure but if you recall the Java collection framework, there is no explicitly define any collection type stack, only queue is there, there is no stack like there in there. This is because queue is basically planned in such a way that using this queue as a collection you can use the queue as a stack, you can queue as a queue, you can queue as a priority queue, you can queue, use queue as a double ended queue or deck, so that is one form, but it can be used to facilitate many other what is called the requirements.

But here the same thing it is stack, here stack is the only one representation, is a collection, but it can be used to use it as a stack as well as queue, and then priority queue. So, that is why here queue is not mentioned explicitly, whereas in Java collection framework stack was not mentioned explicitly.

(Refer Slide Time: 9:59)



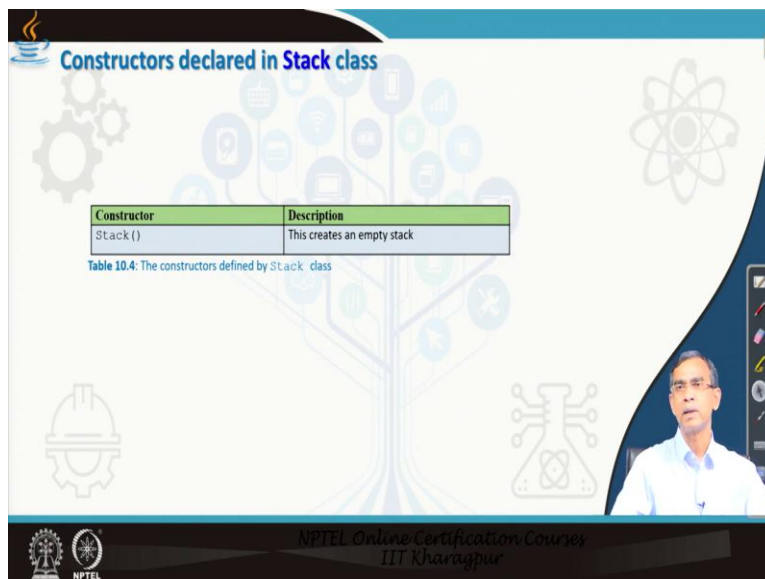
**Class Stack**

- `Stack` is a subclass of `Vector` that implements a standard **last-in, first-out** stack.
- `Stack` only defines the default constructor, which creates an empty stack.
- It follows last-in, first-out principle for the stack elements.
- With the release of JDK 5, `Stack` was retrofitted for generics and is declared as shown here:  

```
class Stack<E>
```

Here, `E` specifies the type of element stored in the stack.

NPTEL Online Certification Course  
IIT Kharagpur



**Constructors declared in Stack class**

Constructor	Description
<code>Stack()</code>	This creates an empty stack

Table 10.4: The constructors defined by `Stack` class

NPTEL Online Certification Course  
IIT Kharagpur

Now, stack is basically to, I mean, enforce one policy, it is called 'last-in, first-out' policy, that mean the element which will be insert last will be deleted first, you cannot do the deletion in any order, it is thus, order is the order of insertion deletion like and then it can also includes any type of objects in it irrespective of whatever it is there. And it has only one constructor, the default constructor to create a stack, we do nor have any other constructor there.

(Refer Slide Time: 10:32)

**Methods defined in Stack class**

Method	Description
<code>boolean empty( )</code>	Returns <b>true</b> if the stack is empty, and returns <b>false</b> if the stack contains elements.
<code>E peek( )</code>	Returns the element on the top of the stack, but does not remove it.
<code>E pop( )</code>	Returns the element on the top of the stack, removing it in the process.
<code>E push(E element)</code>	Pushes <i>element</i> onto the stack. <i>element</i> is also returned.
<code>int search(Object element)</code>	Searches for <i>element</i> in the stack. If found, its offset from the top of the stack is returned. Otherwise, <b>-1</b> is returned.

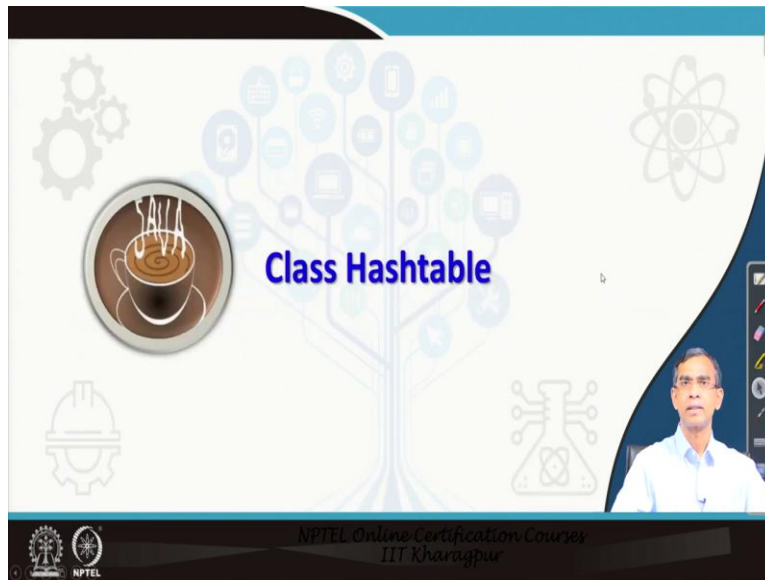
**Table 10.4:** The methods defined by `Stack` class

**Note:**  
**Stack** includes all the methods defined by `Vector` and adds several of its own, shown in Table 10.4 below.

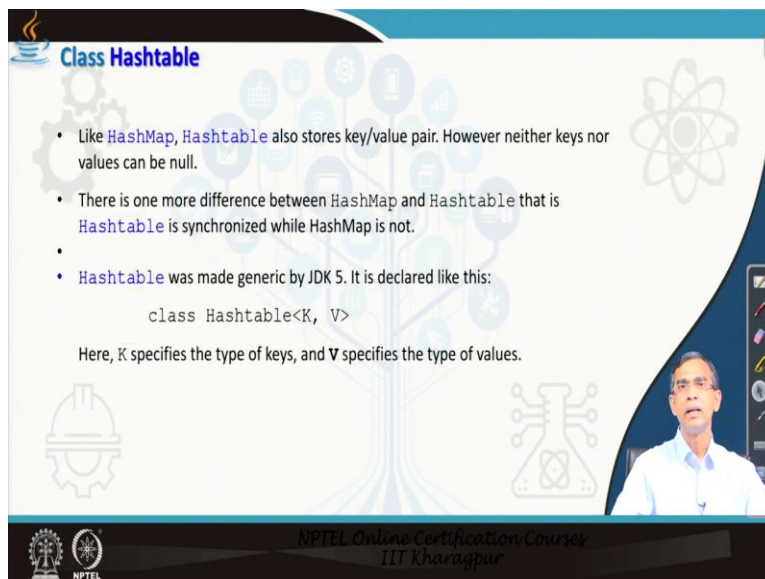
NPTEL Online Certification Course  
IIT Kharagpur

And so fact method is concerned, very simple method, the push method is basically to add an element into the stack and then pop element is basically return the element at the top and it also remove the element once the pop operation is carried out. Now, peek operation just like is pop, it basically you see the value but it will not remove the element from there and it will also check whether stack is currently empty or not by means of empty method it is there. And obviously search method is there by which we can say whether particular element in the stack or not. So, these are the only few simple most methods are there in the stack class.

(Refer Slide Time: 11:18)



The slide features a central title "Class Hashtable" in blue. To the left is a circular icon of a coffee cup with steam. To the right is a stylized tree diagram with various icons as leaves. The background is light blue with faint icons of gears, a hard hat, and a beaker. A small video inset of a man in a white shirt is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".



The slide features a central title "Class Hashtable" in blue. Below the title are three bullet points:

- Like `HashMap`, `Hashtable` also stores key/value pair. However neither keys nor values can be null.
- There is one more difference between `HashMap` and `Hashtable` that is `Hashtable` is synchronized while `HashMap` is not.
- `Hashtable` was made generic by JDK 5. It is declared like this:

```
class Hashtable<K, V>
```

Here, `K` specifies the type of keys, and `V` specifies the type of values.

The slide also features the same background elements as the first slide, including the coffee cup icon, tree diagram, and video inset. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

Now, class `Hashtable` and subsequently `Dictionary`, `Properties`, whatever are there in the Java Legacy Classes, they are basically the hash content I means the objects along with the key values are to be mentioned there. So, `Hashtable` has the two different type `K` and `V`, and here `K` can be of any type and `V` can be of any, `V` is of any objects, so basically the concept is the same that is there in `map`, it is also useful here in `Hashtable` also.

(Refer Slide Time: 11:56)

**Constructors declared in Hashtable class**

Constructor	Description
Hashtable ( )	This is the default constructor. The default size is 11.
Hashtable(int size)	This creates a hash table that has an initial size specified by size.
Hashtable(int size, float fillRatio)	This creates a hash table that has an initial size specified by size and a fill ratio specified by fillRatio. This ratio must be between 0.0 and 1.0, and it determines how full the hash table can be before it is resized upward. Specifically, when the number of elements is greater than the capacity of the hash table multiplied by its fill ratio, the hash table is expanded. If you do not specify a fill ratio, then 0.75 is used.
Hashtable(Map<? extends K, ? extends V> m)	This creates a hash table that is initialized with the elements in m. The capacity of the hash table is set to twice the number of elements in m. The default load factor of 0.75 is used.

Table 10.5: The constructors defined by Hashtable class

NPTEL Online Certification Course  
IIT Kharagpur

Now, if you want to create a Hashtable according Java Legacy Class, there are few constructors that you can follow, the default constructor without specifying any arguments and it basically create a collection of Java Legacy Class of default size 11. And then size also you can mention, whatever the size for the better memory utilization and also at the same time size and feel ration that mean in which rate the Hashtable collection can grow automatically and also it can be created using upper bound specifying the keys and as well as object. So, upper bounded argument object can be there, so this is the last constructor.

(Refer Slide Time: 12:43)

### Methods defined in Hashtable class

Method	Description
<code>void clear( )</code>	Resets and empties the hash table.
<code>Object clone( )</code>	Returns a duplicate of the invoking object.
<code>boolean contains(Object value)</code>	Returns <b>true</b> if some value equal to <i>value</i> exists within the hash table. Returns <b>false</b> if the value isn't found.
<code>boolean containsKey(Object key)</code>	Returns <b>true</b> if some key equal to <i>key</i> exists within the hash table. Returns <b>false</b> if the key isn't found.
<code>boolean containsValue(Object value)</code>	Returns <b>true</b> if some value equal to <i>value</i> exists within the hash table. Returns <b>false</b> if the value isn't found.
<code>Enumeration&lt;V&gt; elements( )</code>	Returns an enumeration of the values contained in the hash table.
<code>V get(Object key)</code>	Returns the object that contains the value associated with <i>key</i> . If <i>key</i> is not in the hash table, a <b>null</b> object is returned.

Table 10.6: The methods defined by `Hashtable` class (continued..)

NPTEL Online Certification Courses  
IIT Kharagpur

### Methods defined in Hashtable class

Method	Description
<code>boolean isEmpty( )</code>	Returns <b>true</b> if the hash table is empty; returns <b>false</b> if it contains at least one key.
<code>Enumeration&lt;K&gt; keys( )</code>	Returns an enumeration of the keys contained in the hash table.
<code>V put(K key, V value)</code>	Inserts a key and a value into the hash table. Returns <b>null</b> if <i>key</i> isn't already in the hash table; returns the previous value associated with <i>key</i> if <i>key</i> is already in the hash table.
<code>void rehash( )</code>	Increases the size of the hash table and rehashes all of its keys.
<code>V remove(Object key)</code>	Removes <i>key</i> and its value. Returns the value associated with <i>key</i> . If <i>key</i> is not in the hash table, a <b>null</b> object is returned.
<code>int size( )</code>	Returns the number of entries in the hash table.
<code>String toString( )</code>	Returns the string equivalent of a hash table.

Table 10.6: The methods defined by `Hashtable` class

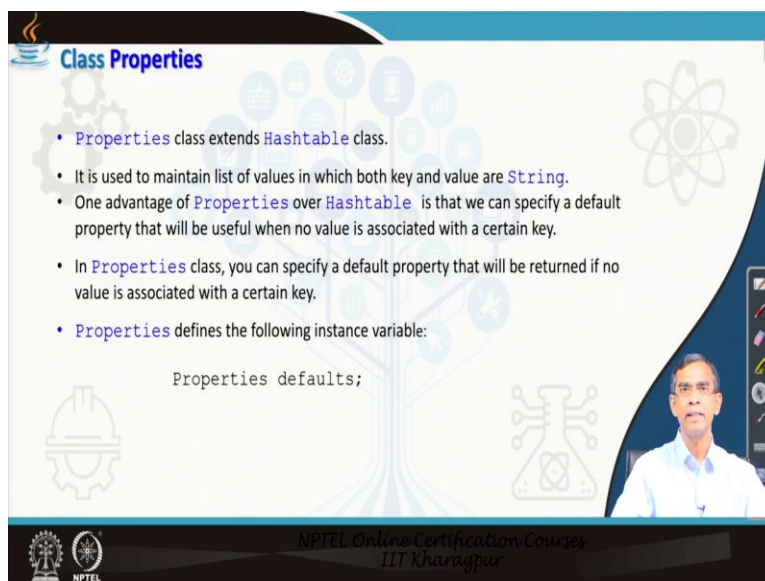
NPTEL Online Certification Courses  
IIT Kharagpur

Now, the methods are very similar to the map methods those are there or we have already discussed like `clear`, `clone`, `contains`, `containsKey`, `containsValue`, and then `get`, and then `put`, then the method is basically called `isEmpty` of whatever it is there. So, those are the very standard method by which you can access elements, can modify the entries in the collection and then know the different status or traversing the collection.

(Refer Slide Time: 13:22)



The slide features a central tree diagram with various icons (gears, a coffee cup, a hard hat, a beaker, and a molecular structure) as nodes. The title 'Class Properties' is prominently displayed in blue text. A small video inset of the presenter is visible in the bottom right corner. The footer includes the NPTEL logo and the text 'NPTEL Online Certification Course IIT Kharagpur'.



The slide contains a list of bullet points and a code snippet. The title 'Class Properties' is at the top left. The bullet points are:

- `Properties` class extends `Hashtable` class.
- It is used to maintain list of values in which both key and value are `String`.
- One advantage of `Properties` over `Hashtable` is that we can specify a default property that will be useful when no value is associated with a certain key.
- In `Properties` class, you can specify a default property that will be returned if no value is associated with a certain key.
- `Properties` defines the following instance variable:

```
Properties defaults;
```

The slide also includes a video inset of the presenter and the same footer as the previous slide.

Now, Class Properties, Properties very similar to Hashtable, it is very similar in the sense that it basically use the key and value pair but there is only one difference is that for the Properties key values as well as object values are to be string type only. So, that is the only difference otherwise it is there. So, which you can implement using properties can be utilized using Hashtable but opposite is not possible.

This is because the Java initially sees everything as a string, so if you can represent a number as a string and user define object also as a form of a string, a floating point values also as a string



and there is a facilities by which Java can allow a programmer to convert a string to a integer values or a floating point values or a Boolean values or any other type of values.

So, string is basically in the sense transfer, convertible from one from to another. Likewise an integer number also can be converted to string and whatever it is there. So, this mechanism is basically exercised here so the Properties is basically has stable, but with the constant that both key and value are of string type.

(Refer Slide Time: 14:43)

The slide displays the title "Constructors declared in Properties class" at the top. Below the title is a table with two columns: "Constructor" and "Description".

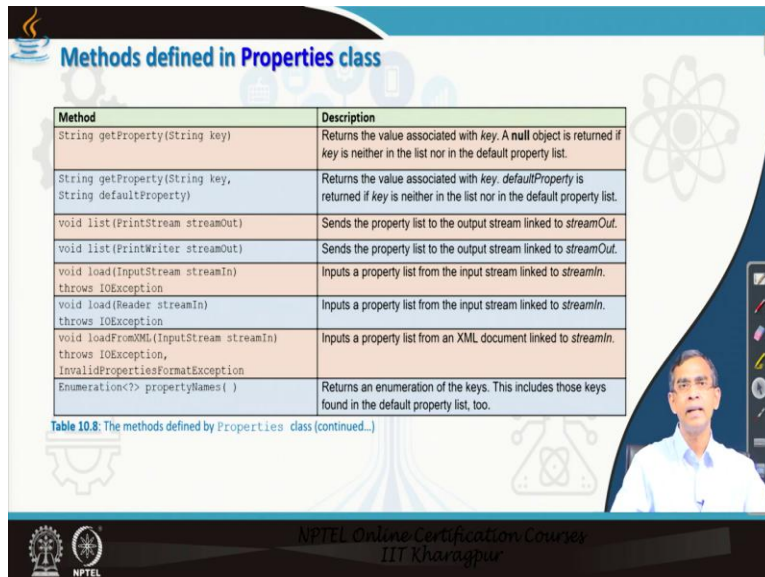
Constructor	Description
Properties( )	This creates a Properties object that has no default values
Properties(Properties propDefault)	This creates an object that uses propdefault for its default values.

Below the table, there is a caption: "Table 10.7: The constructors defined by Properties class".

The slide also features a video feed of a presenter in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

Now, so there are two constructors only, one is the default constructor, so create a collection without any values in it and then there is a another is that given an existing properties if you create a new properties then you can create it, so these are the two ways the constructors are there in for the Properties class.

(Refer Slide Time: 15:06)



Method	Description
String getProperty(String key)	Returns the value associated with <i>key</i> . A null object is returned if <i>key</i> is neither in the list nor in the default property list.
String getProperty(String key, String defaultProperty)	Returns the value associated with <i>key</i> . <i>defaultProperty</i> is returned if <i>key</i> is neither in the list nor in the default property list.
void list(PrintStream streamOut)	Sends the property list to the output stream linked to <i>streamOut</i> .
void list(PrintWriter streamOut)	Sends the property list to the output stream linked to <i>streamOut</i> .
void load(InputStream streamIn) throws IOException	Inputs a property list from the input stream linked to <i>streamIn</i> .
void load(Reader streamIn) throws IOException	Inputs a property list from the input stream linked to <i>streamIn</i> .
void loadFromXML(InputStream streamIn) throws IOException, InvalidPropertiesFormatException	Inputs a property list from an XML document linked to <i>streamIn</i> .
Enumeration<String> getPropertyNames()	Returns an enumeration of the keys. This includes those keys found in the default property list, too.

Table 10.8: The methods defined by Properties class (continued...)

NPTEL Online Certification Course  
IIT Kharagpur

And the methods are basically almost same as the Hashtable methods in addition to its there are few methods, obviously those methods are for internal conversion from string to values and values to string and vice versa, so those methods are mentioned here. So, one is basically get property, it basically return a string, that means what is the property of a key values it is there and then there is a list also, it will list all the key values that is stored in there.

Similarly, load, and there is another is that, so also from the file you can create objects and then store them as a map like means Properties, so there are few methods in which they are using inputStream and outputStream. Those things are basically the advance concept well the file input output or data input output or inputStream outputStream needs to be considered, so it is basically is versatile.

Properties are very huge so far compared to other classes those are there in Java Legacy Classes because initially Java gives enough importance to string only, therefore, storing an efficient manner according the older version of Java is basically using the properties only.

(Refer Slide Time: 16:21)

**store() and load() methods**

Note:

1. One of the most useful aspects of `Properties` is that the information contained in a `Properties` object can be easily stored to or loaded from disk with the `store()` and `load()` methods.
2. At any time, you can write a `Properties` object to a stream or read it back. This makes property lists especially convenient for implementing simple databases.

NPTEL Online Certification Course  
IIT Kharagpur

And there is another store and load methods also allows the programmer to store and load data, I mean storing data and loading from data from the external memory also, store and load is very useful method for external, whatever the other methods are basically from the internal memory or primary memory but store and load for the secondary memories.

(Refer Slide Time: 16:44)

**Class Dictionary**

NPTEL Online Certification Course  
IIT Kharagpur

Now, there is another class Dictionary. Dictionary class is very similar to the map concept that we have discussed, is a map only unlike the Hashtable and properties that we have discussed, it also like the discuss, it is unlike the properties of Hashtable, it is basically is an abstract class. What is the concept? That means all the methods those are there defined in, defined actually you can create a customize class extending Dictionary class to be in your own.

So, Java gives, it is little bit a blank cheque like, so whatever the things that you want to do, you just simply implement Dictionary, that means the compatibility issue can be restore and it can ensure it. So, there is no much about so far the current programming situation is concerned. The Dictionary is really hardly used by any programmer because the Dictionary which is now, which is earlier and the map is which is now is much more, so map gives more facilities than the Dictionary, so people prefer maps only, the Dictionary is highly overlooked here.

(Refer Slide Time: 18:02)

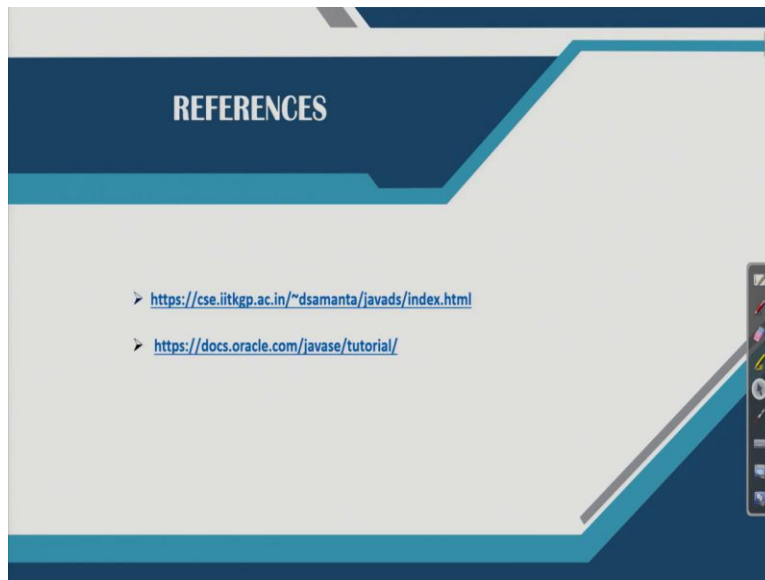
Method	Description
Enumeration<V> elements( )	Returns an enumeration of the values contained in the dictionary.
V get(Object key)	Returns the object that contains the value associated with key. If key is not in the dictionary, a null object is returned.
boolean isEmpty( )	Returns true if the dictionary is empty, and returns false if it contains at least one key.
Enumeration<K> keys( )	Returns an enumeration of the keys contained in the dictionary.
V put(K key, V value)	Inserts a key and its value into the dictionary. Returns null if key is not already in the dictionary; returns the previous value associated with key if key is already in the dictionary.
V remove(Object key)	Removes key and its value. Returns the value associated with key. If key is not in the dictionary, a null is returned.
int size( )	Returns the number of entries in the dictionary.

Table 10.9: The methods declared by Dictionary class

But Dictionary is an interface, it gives design rule that what are the different methods that a programmer, if you want to implement a Dictionary can utilize here. So, there are the few basic methods I have mentioned here, elements is a method, get is a method, isEmpty, and then put method, remove method, know the size of a Dictionary collection, size method, those are the few methods are there in the Dictionary.

Again I want to say is that if you want to implement Dictionary, it is your responsibility to implement all these things. Now, why you can go for implementation? This is because implementation enables you to access all those collection in a synchronized or parallel execution otherwise why you should go for having this one when the map, the better supports is available to you.

(Refer Slide Time: 18:52)



So, with these things I just want to conclude the discussion about Java collection concept and its facilities. Definitely, learning will be complete only when we can utilize all those methods in some program based, so programming issues will be discussed next. Now, our plan of the discussion for the remaining course is basically to cover one by one the different data structure. Different data structure means those are the theoretically data structures are available like array, linked list, tree and graph, many things are there.

Then we will discuss about how all those data structures without using JCF can be implemented. So, realization of this data structure from the core, that mean without taking the help of java dot util. Then we shall discuss about how a particular data structures can be exercised using the Java collection framework facilities or map framework facilities. So, this is basically the plan from the next week onwards, and I hope you are enjoying this course and be involved. What are the link has been given, you should go through the link, study materials and then enjoy it. Thank you.