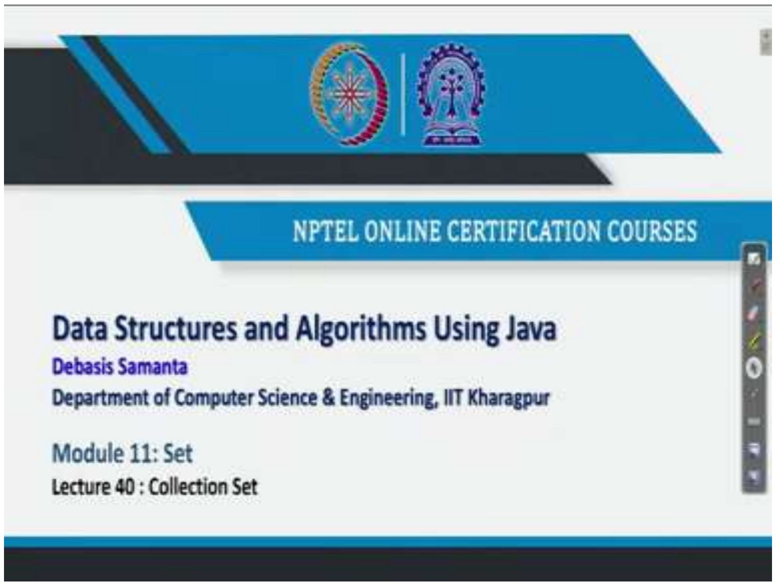
Data Structure and Algorithms using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian institute of Technology, Kharagpur
Lecture 40
Collection Set

(Refer Slide Time: 00:38)


NPTEL ONLINE CERTIFICATION COURSES

Data Structures and Algorithms Using Java
Debasis Samanta
Department of Computer Science & Engineering, IIT Kharagpur

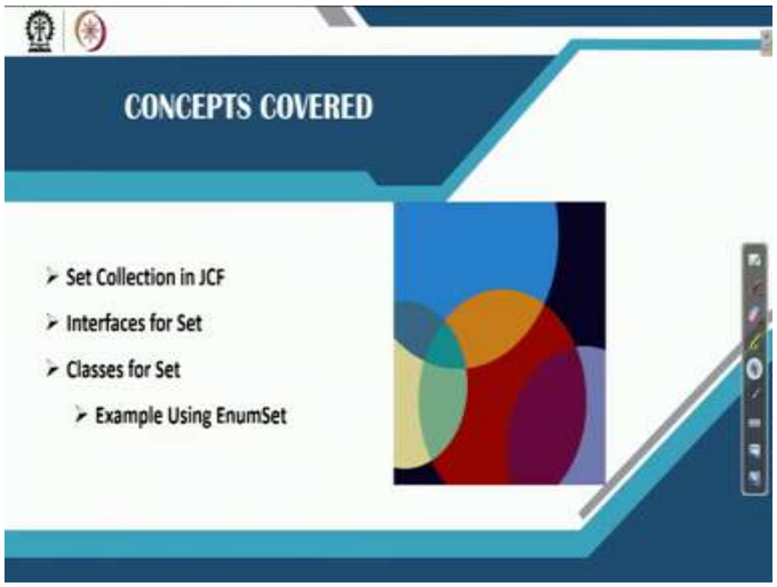
Module 11: Set
Lecture 40 : Collection Set

So in this module, this is a new module and in this module we shall discuss another data structure. This data structure, in fact, is very what is called the familiar to every students in particular all engineering students, it is basically one basic, what is called the topics in mathematics and related to this set, there is an algebra call set algebra. And possibly the student of information technology or computer science you are aware about relational algebra.

Relational algebra is basically one building mechanism, building blocks behind our todays database management system which is also alternatively called relational database management system. Now, this database management system follows the relational algebra to manipulate data. In database management system data is basically a table.

And, you know, table is consist of a set of records. We have also studied that a table is nothing but a collection. In other words if we have the set theory or other relational algebra, then many data manipulation related activities can be done very efficiently.

(Refer Slide Time: 02:20)


CONCEPTS COVERED

- Set Collection in JCF
- Interfaces for Set
- Classes for Set
  - Example Using EnumSet

So, today basically we will start the topic, this topic is related to set, we can say set as a collection. Now, in this lecture, we will try to learn what is the facilities that is provided by the Java developer to manage the set as a collection. As you know Java supports is basically in the form of two things; interfaces and classes.

So, we will try to cover what are the interfaces and classes which a programmer can use in their programs so that the set manipulation is possible. And we will cover one particular set, which is called the enumerated set, so that we can illustrate how the enumerated set can be manipulated using the Java collection framework. So, this is our plan for todays lecture.

(Refer Slide Time: 03:19)

About Set as a Collection

Now let us start about set. Now, I told you that set as a collection. Now possibly you know, set is basically is a collection of homogeneous data. Now, again, you can refer to the array where we define an array is a collection of homogeneous data. For example, if you want to say list of integer numbers, so this is a collection list of names of students, it is another collection. So, list of records of all students in a class or in an university it is again another collection.

So, a collection depends on what type of object that you want to maintain. It can be numeric data, it can be user defined data type abstract data like students object, persons information, employee information like. Now here question that arises, why we should study set differently, although we know that array is also a collection is a homogeneous set basically is a collection of homogeneous data. And set that just now I told you that set is also a homogeneous collection of homogeneous data.

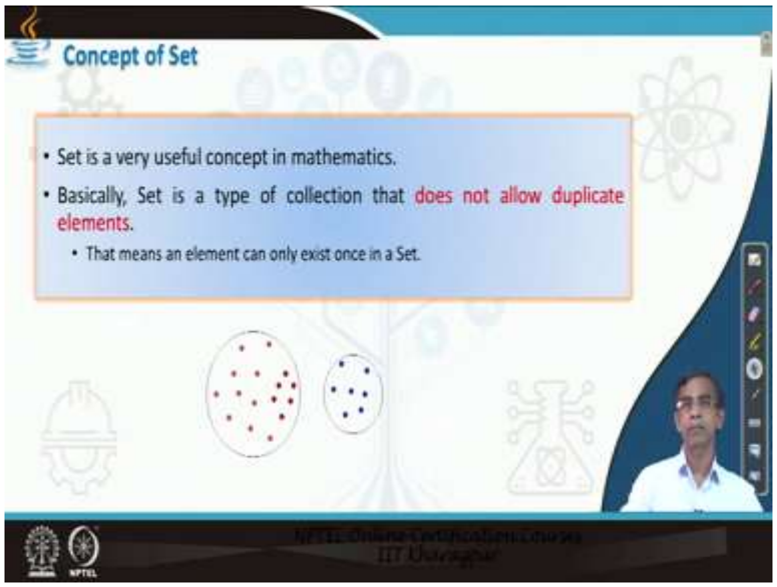
Then why we should take the two things differently array and sets. Now you can recall an array can contains numbers. For example, array of numbers where any numbers can appear one or more times. So, this means that array is a collection which basically allows you to store duplicate elements in it. On the other hand, if you know the set, set is also a collection of elements where all elements are of unique. There is no duplicate elements is allowed.

So, this concept is basically there and this concept makes it as an important aspects. In another way, also we can think about why array is not sufficient whereas the set is required. Now so far the array data structure is concerned usually the operations that is possible which is basically we have studied like insertion operation, deletion operation, traversal operation, merging maybe. So, these are the operations.

But they are few more operations when we have to take into account whenever there is, is basically manipulation when two or more what is called the lists are needs to be managed together. For an example I can tell you possibly know two lists are given to you and you have to find the list of all common elements in the two list or the elements which are not common in both the lists or the list which includes all elements which are there in both the lists.

So, these kind of operations require some different way to be managed the data, these operations popularly called intersection, union, difference, sum addition of two sets all these things are there. They are basically form one new mathematics. It is called the set algebra or it is also called relational algebra. Now, relational algebra, where the elements in this algebra is basically set and different operations related to sets that we can do it. So, set in fact, compared to array much more mathematical or more logical.

(Refer Slide Time: 07:42)



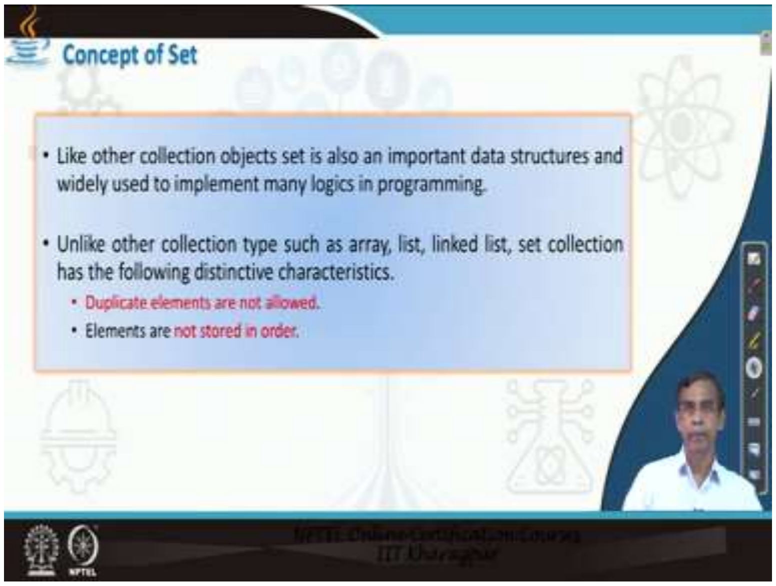
Concept of Set

- Set is a very useful concept in mathematics.
- Basically, Set is a type of collection that does not allow duplicate elements.
    - That means an element can only exist once in a Set.

In fact, now we will study about the set concepts. So, we have understanding about that set is basically is a homogeneous is a collection of homogeneous data, homogeneous elements where no duplicate elements is allowed. So, this means that if an element exist then only it exists once. It cannot exist more than once like.

So, this is the concept of set and pictorially we usually set or represent a set using Venn diagram sort of things say ellipse or a circle where all elements are there. So, that is basically in mathematics we usually follow it.

(Refer Slide Time: 08:23)


Concept of Set

- Like other collection objects set is also an important data structures and widely used to implement many logics in programming.
- Unlike other collection type such as array, list, linked list, set collection has the following distinctive characteristics.
  - Duplicate elements are not allowed.
  - Elements are not stored in order.

Now, here so set in fact is basically collection and as you have already learned about a collection, is basically needs a theory of data structure so that how it can be stored in memory and what are the different operations that we can perform on these data collection rather.

So, like other data structure, like other collections as data structure set is also another collection. And it needs the theory of data structure to be studied. So, this is what is our objectives are there. And another point I want to mention in this regard like array or link list or any other structure that we have studied, the elements those are there in a set is not necessarily to be in order. For example, an array we can think about that all elements are in an ascending order if the array of elements are integers or string, whatever it is there.

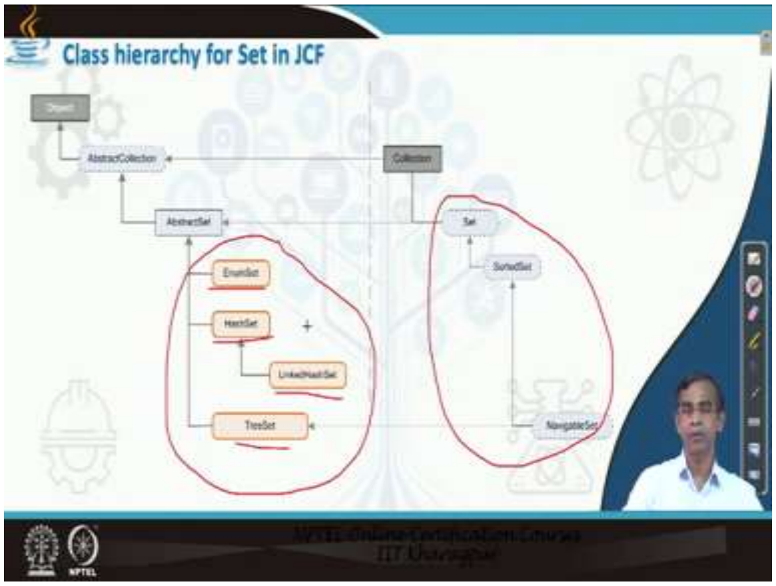
But whenever we consider a set of objects, sometimes ordering is not an issue or is not an important point to be considered. Anyway the set in that sense makes a different than the other data structure where the some majority of the data structure are essentially represents or somehow by default or explicitly can be made ordered. Here if we need a set also can be made ordered also, no way. But in general, set is not necessarily to be an ordered collection.

(Refer Slide Time: 10:14)

Set in JCF

Anyway so we have understood about the set and a, set as a collection rather it is basically set as a as another data structure to be studied. Now, since set is an important one data structure. So, Java developer considered it very seriously and they therefore include this structure in its Java collection framework, as a java dot util package.

(Refer Slide Time: 10:43)


Class hierarchy for Set in JCF

Now, let us see the class hierarchy or supports in Java collection framework facilitating the set related activities. So, set like other collections we have studied, it has a couple of interfaces and abstract class and then major main classes. Now the interfaces which are there in this set collection are listed here. These are the basically set up interfaces. As we see the set is the interface which basically the collection is an abstract class which basically implements this set interface.

And then there are another two sets sorted set, it is basically is a child interface of the interface sets and navigable set is basically is an another is a basically extension of sorted set. Now, there are different interfaces, basically is a design plan which declares many methods which basically necessary to manage sets those methods are only declared but they need to be implement. So, they will be implemented by their corresponding classes.

As I said, the collection is, collection implement set but collection is an again abstract class. So, this basically the collection, again, is an extension of abstract collection actually and this abstract collection is again extended by another abstract class it is called the abstract set. Now, finally, there are many class which basically implements all these abstract class they are by all interfaces. For example, here as we see set is an interface, basically abstract set is basically implements set interface.

Now, this implementation carried out in all these classes. So, there are 4 different classes as we have, as we see here in this figure. So, this is called the Enum Set, this is called Hash Sets, this is Linked Hash Set, this is Tree Set. Possibly this kind of nomenclature we have already familiar while we are discussing about map. There is also Enum map or like say Hash map or it is basically Linked hash map and Tree map. There are some other sorted map sort of things are there.

Now you see so this is map is the one version whereas set is the another version of collection. So, the way we store a map, what is a map? Map basically is a table, we have to maintain table so that faster access is possible. Here also the different way, the set can be stored by which faster accessing is possible. Now Enum set is basically it store the enumerated data type.

You know, exactly when a data type is called enumerated, it basically is a set of finite elements which is fixed. There is no, this is just like a constant storing constants, like a set of constant storing into a group. It is called the Enum so a class can be declared for that using the Enum key word.

Now, like hash basically hashing is basically one technique we have already studied by which a particular element can be accessed in a fastest way. So, a set also can be stored using the hashing technique then such a set is called the hash set. Now similarly a hash can be stored using the tree structure then such set is called tree set. If we store a set using the linked structure, then it is called the linked set, link as well as hashing together. Then it is called the linked hash set.

So, these are the different actually way by which a set can be maintained or a set data structure can be stored. Now again, Java developer has given enough efforts to have the set in a different way to store in memory, those internal representation is actually Java programmer should not bother about. What they should bother about is that, that just simply follow the plug and play procedure. That means what are the different methods or functions are there to manipulate the set element according to a particular representation.

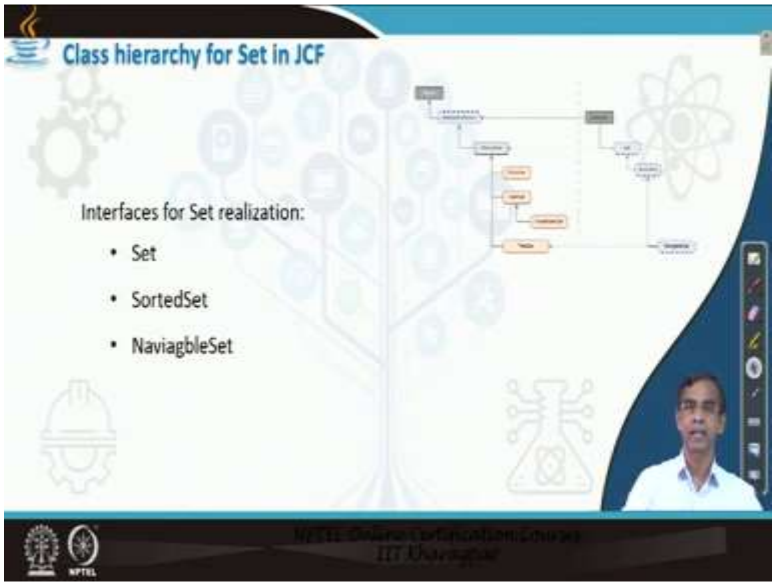
Rather we can say the data structure for which it is stored in memory and whatever the different operations, those are basically bundle as a class. So, we have to just simply use class in your program and then you just use the different methods, the different constructor to create your sets and manipulative sheets. So, this is basically the concept that that is there and it is overall concept basically so far this collection framework is concerned and set is also not a different things than the collection framework.

(Refer Slide Time: 16:31)

Interfaces Defining Set

So, let us first start about what are the different interfaces that we have mentioned about? The methods those are declare there because those methods are to be understood, because we want to use them in our program, because all methods those are declared in interface are implemented in their corresponding classes, like hash set, tree set, linked hash set like this one. So, let us first start about interfaces. Those are there to manage sets.

(Refer Slide Time: 16:49)



Class hierarchy for Set in JCF

Interfaces for Set realization:

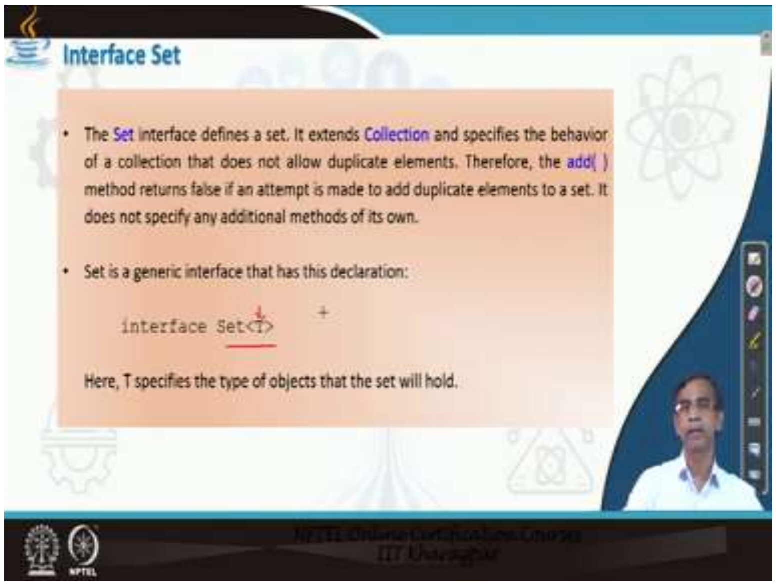
- Set
- SortedSet
- NaviagbleSet

Now there are 3 interfaces; namely set and sorted set and navigable set. Now the 3 different interfaces are to cater to the need of 3 different type of set collection. So, set is basically just ordinary set where the homogeneous data to be stored, not necessarily to be in order. On the other hand, sorted set is basically is an approach to store the elements in the set in a particular order, either ascending order or descending order or some order which is basically applicable to a particular object, set of objects.

Now, navigability set is basically just like other navigable collection if we want to access a particular element, but where the key value does not match exactly. So, it is basically near about, closest match, or we want to access some data which is greater than some values or less than some values or within certain range, then to facilitate this kind of activity, the navigable set has been planned. So, there are 3 different interfaces targeting 3 different ways the set collection can be maintained in computer memory.

(Refer Slide Time: 18:14)

Interface Set



Interface Set

- The Set interface defines a set. It extends Collection and specifies the behavior of a collection that does not allow duplicate elements. Therefore, the add( ) method returns false if an attempt is made to add duplicate elements to a set. It does not specify any additional methods of its own.
- Set is a generic interface that has this declaration:

```
interface Set<T>
```

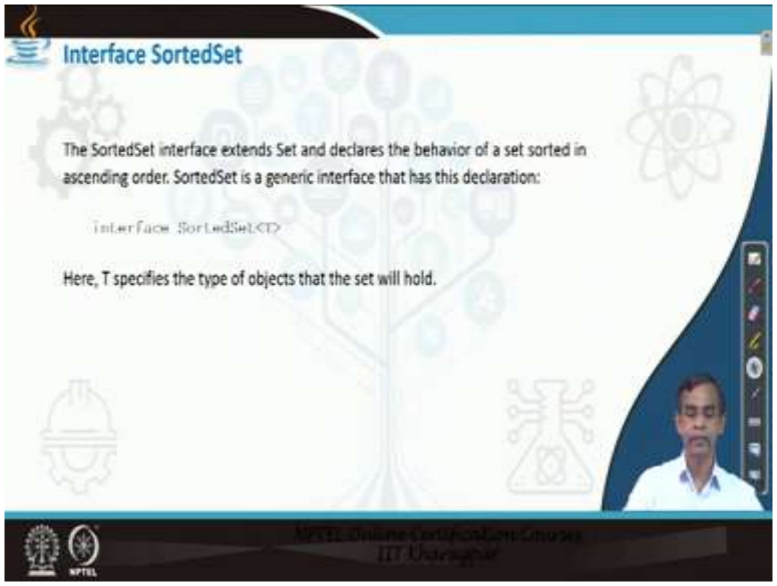
Here, T specifies the type of objects that the set will hold.

Now, let us see what are the different methods are there in these interfaces? Now, first of all, all interfaces like set, sorted sets, navigable sets they allow the genetic type to be maintained. That means with these classic definitions, with this collection framework or any other class if you consider, then it can allow you to store data of any type. Rather, we can say it is an object of any type. So, you can define a set of all integers or a string or floating point number or any complex number like objects or whatever it is there.

But it allow you to store only genetic types and only one genetic specification is required. For example, here an interface set is basically defined in Java by this structure. So, T is basically generic that mean it can store any T type of data in it. So, this is the concept it is there.

(Refer Slide Time: 19:27)

Interface SortedSet



Interface SortedSet

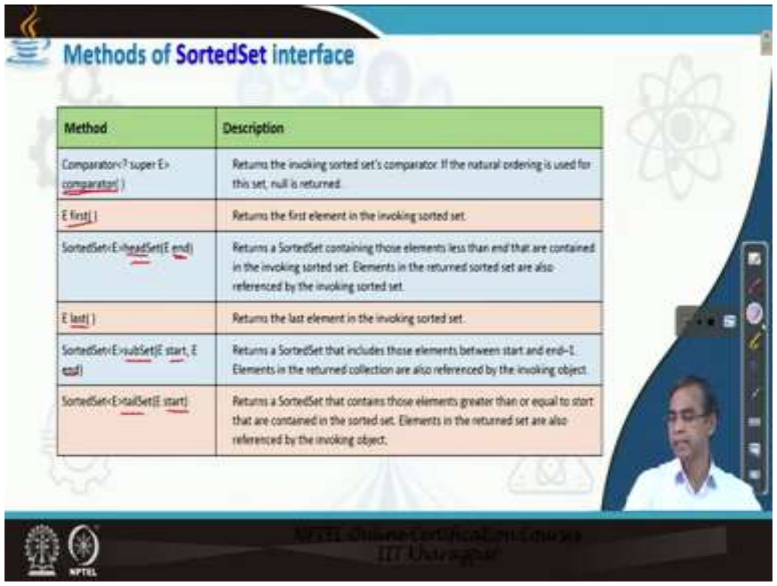
The SortedSet interface extends Set and declares the behavior of a set sorted in ascending order. SortedSet is a generic interface that has this declaration:

```
interface SortedSet<T>
```

Here, T specifies the type of objects that the set will hold.

Now, let us learn about the different methods, okay, different, okay first let us consider about sorted set I have already told about it. So, sorted set is basically an older version of the set and like set it also allowed to store generic type. So, any type of data you can store using the sorted set facilities.

(Refer Slide Time: 19:48)



Methods of SortedSet interface

Method	Description
Comparator<? super E> comparator( )	Returns the invoking sorted set's comparator. If the natural ordering is used for this set, null is returned.
E first( )	Returns the first element in the invoking sorted set.
SortedSet<E> headSet(E end)	Returns a SortedSet containing those elements less than end that are contained in the invoking sorted set. Elements in the returned sorted set are also referenced by the invoking sorted set.
E last( )	Returns the last element in the invoking sorted set.
SortedSet<E> subSet(E start, E end)	Returns a SortedSet that includes those elements between start and end–1. Elements in the returned collection are also referenced by the invoking object.
SortedSet<E> tailSet(E start)	Returns a SortedSet that contains those elements greater than or equal to start that are contained in the sorted set. Elements in the returned set are also referenced by the invoking object.

Now set actually, it does not have any method defined in its of own. All the methods which are declared in the collection are also because set is basically an extension of the collection or rather we can say collection implement set , so it mean all methods which are declared in collection are also accessible to the interface set. Mainly the add method you are familiar to. So, add method is there.

But in this set whenever this add method is implemented, it automatically take care about that if you want to add a duplicate elements, then it will simply ignore. So, that is a special consideration that has been taken while the add method is implemented in their set related classes like say hash set or T set like there. Now let us come to the sorted set. It has certain what is called the methods of its own. I have listed a few methods like so he is a comparator method because sometimes sorted set means we have to store the elements in an order.

So, how you can define an order? For example, two student object, so how you can make them ordered that okay student S1 and then S2. So S1 before S2 or vice versa. So, we have to impose an ordering rule. So, that impose ordering rule basically implemented by comparator. So, you can override method in your program. If you want to use depending on your own define data type, then you can have this ordering.

There are certain data types for example integer all numeric int, sort, long or any string type, string or character ordering is automatically there. So, we do not have to write this. So, comparator method by default it is there. We do not have to think about it. Now, there is another method is called the first, the name implies that in the sorted set it will basically return you the top element or the first elements.

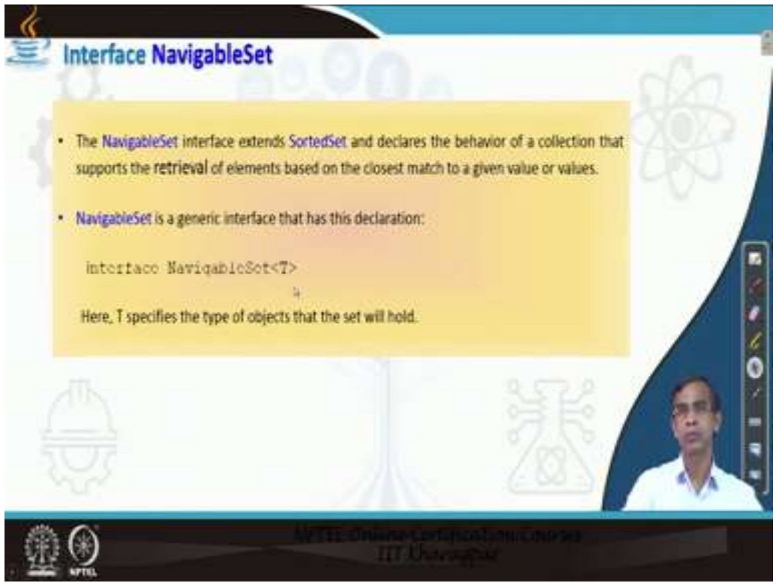
Now head set end it basically gives you some elements, which is in the front part of the set because it is in the sort order. So, it is starting from the top element to the element which is indicated by end. So, it will basically gives you a subset which basically from the starting from some set to a some sets like. The last element is just opposite to the first element. It will return the last element which is stored in the sorted sets.

Now subset also, another in this head set is basically top few elements. But here started starting from the start to end the list of all elements will be inclusive all the start and end elements are basically to be retrieve. So, it is basically gives another method by which you can retrieve a subset of a set. Now tail set is just opposite to head set. It basically give the bottom most element starting from the start.

So, last few elements which is stored there in the sets. Now so these are the different what is called the methods are defined in the class, in the interface sorted set and ultimately they have been implemented corresponding to the different class it is there.

(Refer Slide Time: 23:18)

Interface NavigableSet



Interface NavigableSet

- The NavigableSet interface extends SortedSet and declares the behavior of a collection that supports the retrieval of elements based on the closest match to a given value or values.
- NavigableSet is a generic interface that has this declaration:

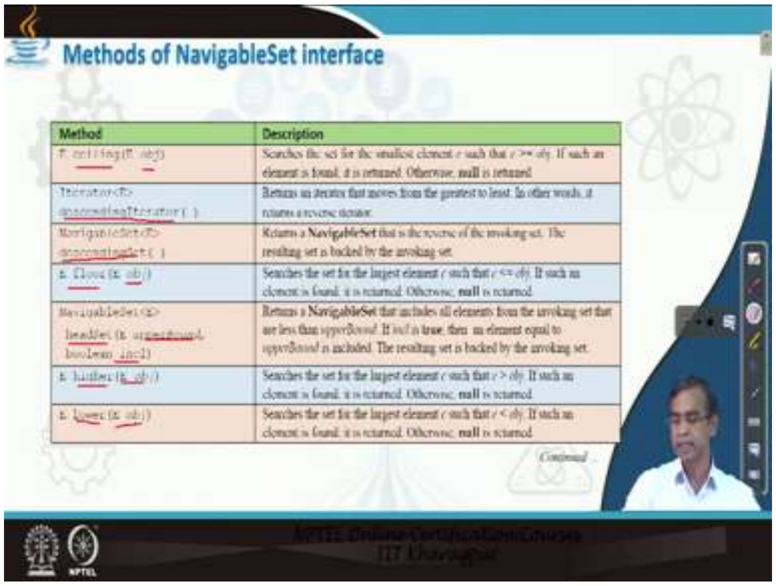
```
interface NavigableSet<T>
```

Here, T specifies the type of objects that the set will hold.

Now let us come to the navigable set. Navigable set, just like set whereas all the sets that we have discussed about set, sorted set is basically they are crip set, crip set means if one particular element matches, then only this element is there.

But here, navigable sets is basically is an extension of sorted sets and it also declares the behaviour of a collection that supports the retrieval of elements based on the closest match to a given value or values. You can provide a set of values and then you can try to retrieve the elements in the set, which basically I mean matches, but not exactly matches it is closest match, nearby matches. Now, like other sets, it also allow you to store any type of data. For example, it is basically generic type and so it can allow you to store any type of data here.

(Refer Slide Time: 24:20)



Methods of NavigableSet interface

Method	Description
`E ceiling(E obj)`	Searches the set for the smallest element *e* such that *e* >= *obj*. If such an element is found, it is returned. Otherwise, **null** is returned.
`Iterator<E> descendingIterator( )`	Returns an iterator that moves from the greatest to least. In other words, it returns a reverse iterator.
`NavigableSet<E> descendingSet( )`	Returns a **NavigableSet** that is the reverse of the invoking set. The resulting set is backed by the invoking set.
`E floor(E obj)`	Searches the set for the largest element *e* such that *e* <= *obj*. If such an element is found, it is returned. Otherwise, **null** is returned.
`NavigableSet<E> headSet(E upperBound, boolean incl)`	Returns a **NavigableSet** that includes all elements from the invoking set that are less than *upperBound*. If *incl* is **true**, then an element equal to *upperBound* is included. The resulting set is backed by the invoking set.
`E higher(E obj)`	Searches the set for the largest element *e* such that *e* > *obj*. If such an element is found, it is returned. Otherwise, **null** is returned.
`E lower(E obj)`	Searches the set for the largest element *e* such that *e* < *obj*. If such an element is found, it is returned. Otherwise, **null** is returned.

Continued ...

Now, there are several methods which are also define in this rather we can say declare in this interface. These methods are like sealing. Sealing is basically if you pass an objects then it will basically retrieves, all the elements which basically above the value of this object.

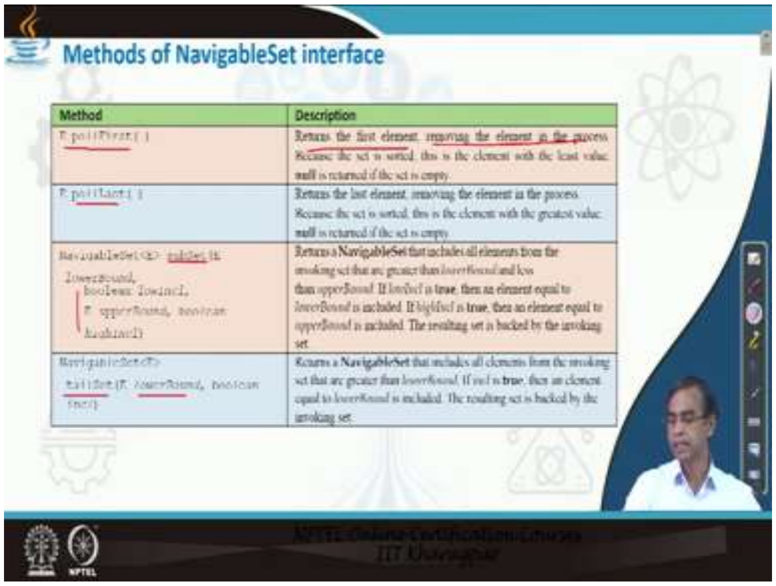
So, it basically you can pass for example we pass 5, then it will retrieve all the elements which is basically 5 and above. Now here is a descending iterator, as you know. So, it basically if your sorted set or some sets stored in some manner, then if you want to have the elements in descending order are like this one, like descending order there is an ascending order. So, all the order by default it is an ascending order.

But if you want to have it in descending order so you can use this method to apply on a particular collection, set collection then it will basically return you in a descending order. So, it basically iterator if you want to traverse a sorted set actually in a bigger series an extension of sorted set, then you can do it. Now say they are also one method is called the descending set. It basically return you all the elements in the descending order of the elements.

The floor is basically just opposite to the ceiling. It will basically gives you some element, which is less than the object that is passed it there. And head set is basically, same as the method we have discussed is basically top few elements starting from upper bound to some boundaries there. And here inclusive means weather upper bound will be included or not. Then higher is basically just like a first like.

So, it is basically return all the elements which is basically higher than the it is just like ceiling also we can say. So (higher) lower is opposite to the higher. It basically returns all the elements which has the value lower than these value passed it. So, there are different methods that is declared there. There are few more methods also there in this interface which I have listed here.

(Refer Slide Time: 26:33)



Methods of NavigableSet interface

Method	Description
E pollFirst( )	Returns the first element, removing the element in the process. Because the set is sorted, this is the element with the least value. null is returned if the set is empty.
E pollLast( )	Returns the last element, removing the element in the process. Because the set is sorted, this is the element with the greatest value. null is returned if the set is empty.
NavigableSet<E> subSet(E lowerBound, boolean lowIncl, E upperBound, boolean highIncl)	Returns a NavigableSet that includes all elements from the invoking set that are greater than lowerBound and less than upperBound. If lowIncl is true, then an element equal to lowerBound is included. If highIncl is true, then an element equal to upperBound is included. The resulting set is backed by the invoking set.
NavigableSet<E> tailSet(E lowerBound, boolean incl)	Returns a NavigableSet that includes all elements from the invoking set that are greater than lowerBound. If incl is true, then an element equal to lowerBound is included. The resulting set is backed by the invoking set.

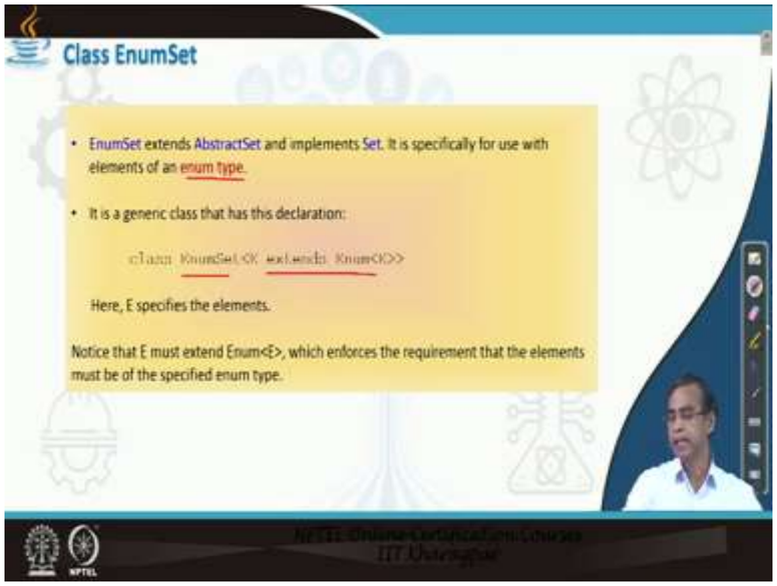
The poll first means it will basically pick that which is the first elements stored in this set. Poll last is basically the last element as it named the poll implies that it will return the first element and while return, removing the element in the process. So, poll first and poll last means, it will not only return the first order, it will basically remove the elements after returning also. That mean it is on sort of delete or removal process we can say.

And this is a subset which we have already understood about. If you want to have a subset between certain bound of values, then we can specify the bound of values and then it will basically return the subsets. And then tail set it is just like head set is a lower parts of the set, starting with the certain value that is passed by the lower bound. So, these are the different elements, methods which are basically defined there in this interface and it includes the different interfaces there.

(Refer Slide Time: 27:35)

Class EnumSet

NPTEL Online Certification Courses
IIT Kharagpur



Class EnumSet

- EnumSet extends AbstractSet and implements Set. It is specifically for use with elements of an enum type.
- It is a generic class that has this declaration:

```
class EnumSet<E extends Enum<E>>
```

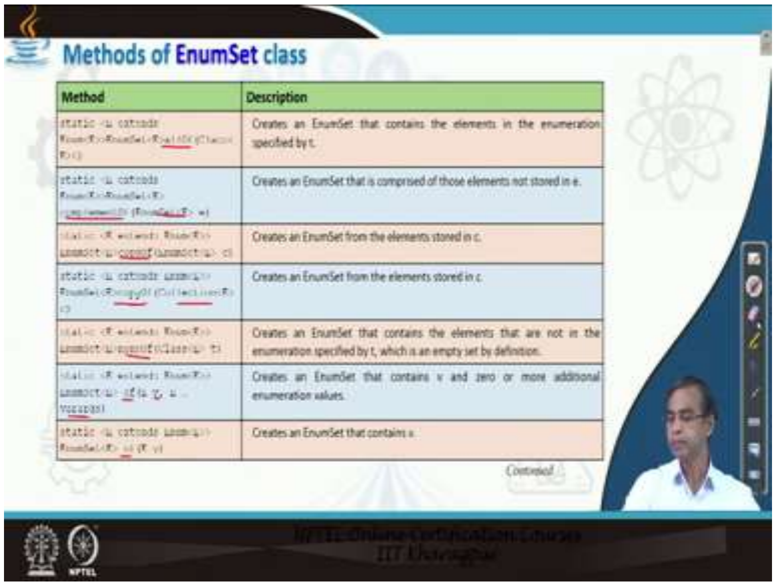
Here, E specifies the elements.

Notice that E must extend Enum<E>, which enforces the requirement that the elements must be of the specified enum type.

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us learn about the EnumSet class. So, this is the one class, very simple class. And it is basically to store the elements, not of all type that are rather we can say the Enum type whereas the Enum type is again, abstract type because you can define your enumerated sets according to your own ways. So, this class basically Enam set store only the enumerated type elements. And now let us see how this type of class or this class can be utilized in our programming endeavour.

(Refer Slide Time: 28:12)



Methods of EnumSet class

Method	Description
static <E extends Enum<E>> EnumSet<E> allOf(Class<E> t)	Creates an EnumSet that contains the elements in the enumeration specified by t.
static <E extends Enum<E>> EnumSet<E> complementOf(EnumSet<E> e)	Creates an EnumSet that is comprised of those elements not stored in e.
static <E extends Enum<E>> EnumSet<E> copyOf(EnumSet<E> c)	Creates an EnumSet from the elements stored in c.
static <E extends Enum<E>> EnumSet<E> copyOf(Collection<E> c)	Creates an EnumSet from the elements stored in c.
static <E extends Enum<E>> EnumSet<E> noneOf(Class<E> t)	Creates an EnumSet that contains the elements that are not in the enumeration specified by t, which is an empty set by definition.
static <E extends Enum<E>> EnumSet<E> of(E v, E ... varargs)	Creates an EnumSet that contains v and zero or more additional enumeration values.
static <E extends Enum<E>> EnumSet<E> of(E v)	Creates an EnumSet that contains v.

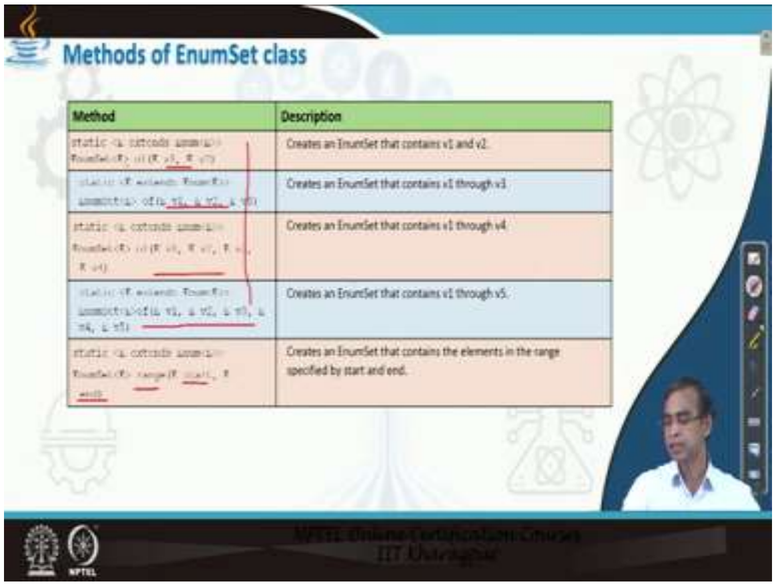
Contined

First of all let us see what are the different methods that this class supports. So, here we have listed few major methods are there and here the method is basically. Here we just listed all of method. So, all of method by name, it implies that it will basically retrieve all the elements which are there in the sets. And then there is another method is called the complement of. So, it is just compliment. That means if you pass a set, which basically return the elements which is here, which is not here, those elements, but there in the given sets.

So, this basically complement set. You know a set A then complement A like this one. Then copy of is basically bulk it is basically return a copy of all the elements and copy of there are many methods are there. Copy of if you give the collection, it basically copy this collection into the Enum set. Here copy is also given another Enum set to copy this one. And none of it is basically it is just exclusion operation like. And then of is basically is a set within certain values then if this of method has many what is called the polymorphic views.

It basically these of basically tell about V and the Vargs. That means basically it represents elements which V and certain argument that it passed actually. So, like this, there are many more of version also there defined.

(Refer Slide Time: 29:49)



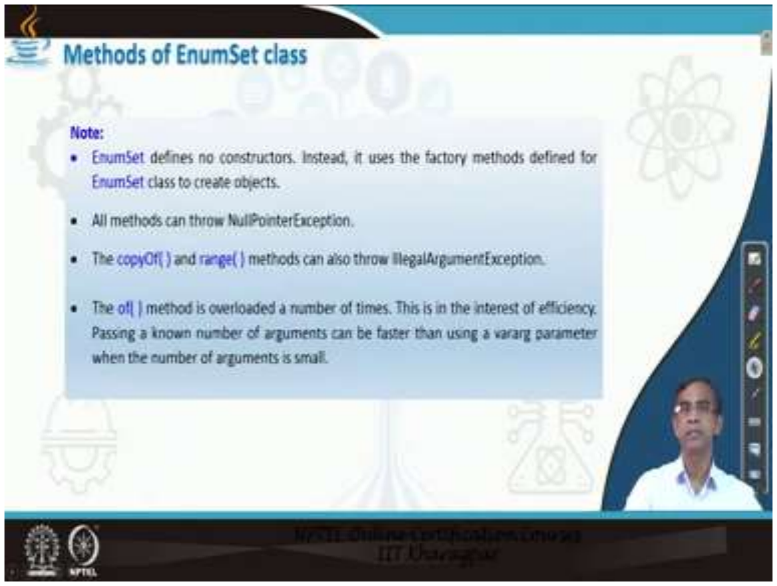
Methods of EnumSet class

Method	Description
static <E extends Enum<E>> EnumSet<E> of(E v1, E v2)	Creates an EnumSet that contains v1 and v2.
static <E extends Enum<E>> EnumSet<E> of(E v1, E v2, E v3)	Creates an EnumSet that contains v1 through v3
static <E extends Enum<E>> EnumSet<E> of(E v1, E v2, E v3, E v4)	Creates an EnumSet that contains v1 through v4
static <E extends Enum<E>> EnumSet<E> of(E v1, E v2, E v3, E v4, E v5)	Creates an EnumSet that contains v1 through v5.
static <E extends Enum<E>> EnumSet<E> range(E start, E end)	Creates an EnumSet that contains the elements in the range specified by start and end.

NPTEL Online Certification Courses
IIT Kharagpur

Here we have listed few more. It basically gives of methods V1 and V2. So, basically it returns the enumerated sets which contains V1 and V2. And then again, if you want to have the set which contain V1, V2, V3 or V1, V2, V3, V4 or V1, V2, V3, V5. So, these are the different what is called the polymorphic function methods are declared there and there is another method also range. If we give the start and end, then it basically retrieves subset in between the start and end both inclusive. So, these are basically different methods are there.

(Refer Slide Time: 30:34)


Methods of EnumSet class

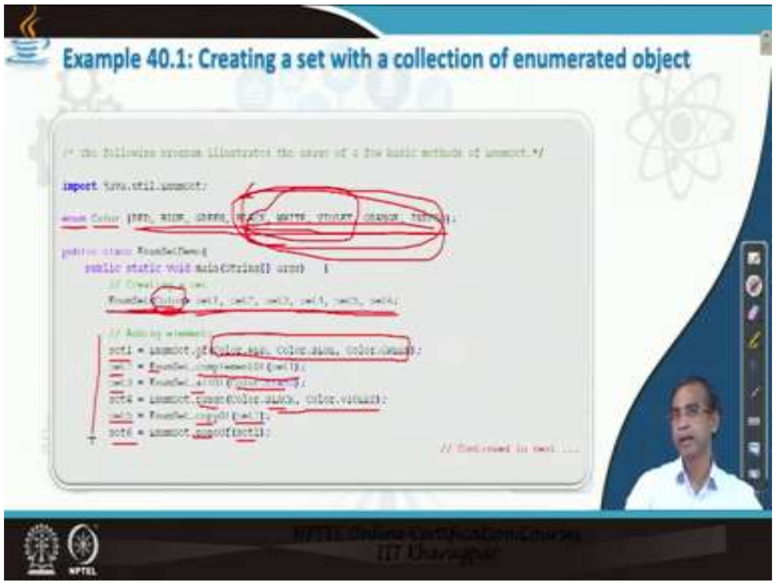
Note:

- EnumSet defines no constructors. Instead, it uses the factory methods defined for EnumSet class to create objects.
- All methods can throw NullPointerException.
- The copyOf( ) and range( ) methods can also throw IllegalArgumentException.
- The of( ) method is overloaded a number of times. This is in the interest of efficiency. Passing a known number of arguments can be faster than using a vararg parameter when the number of arguments is small.

Now, let us study about with some examples how we can use the Enum set and create one set collection of this type.

(Refer Slide Time: 30:41)

Creating a Collection of EnumSet



Example 40.1: Creating a set with a collection of enumerated object

```
/* The following program illustrates the usage of a few basic methods of EnumSet.*/

import java.util.EnumSet;

enum Color {RED, BLUE, GREEN, BLACK, WHITE, VIOLET, ORANGE, YELLOW};

public class EnumSetDemo{
    public static void main(String[] args)   {
        // Creating a set
        EnumSet<Color> set1, set2, set3, set4, set5, set6;

        // Adding elements
        set1 = EnumSet.of(Color.RED, Color.BLUE, Color.GREEN);
        set2 = EnumSet.complementOf(set1);
        set3 = EnumSet.allOf(Color.class);
        set4 = EnumSet.range(Color.BLACK, Color.VIOLET);
        set5 = EnumSet.copyOf(set1);
        set6 = EnumSet.noneOf(set1);
                                                        // Continued in next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

So, I just want to illustrate with a simple example so that you can understand this concept better. I will not be able to cover many more methods to illustrate the utility of this Enum set class. But the same thing is there in the same line you just simply use the method.

Once you have to create the set first and then you call different methods, which you have already learned about it. And then you just see that how they are doing, how this method are returning the results. And then you can learn by this process. So, little bit practice is required. Now, let us have a very introductory program on so that you can understand the usefulness of this set. Rather you can say the enumerated sets. Now first of all we have to define an Enum class.

So, first we declare colour. Colour is an enumerated class. So, we declare this by this keyword is a Enum. Now this is a class, but this class consists of the object. So, we can say these are the fixed object or static object like. So, this means the colour can have any object like red, blue, green, black this one. As enumerated is a colour there is no constructor. We cannot create any object like.

So, if you want to access any elements so colour dot red or colour dot white, that means you just want to access these elements. White or red for a particular sets earlier. So, we have declare only one enumerated set program can allow to define many more enumerated set also no issue. Anyway, so let us see one example first we create Enum set. And again, you know that Enum set, it does not have any constructors.

So, we cannot create any object of this type, only you can define an object. We cannot initialise the object rather we can just define objects of type. So, what we have defined that Enum set colour that means we want to maintain one enumerated set who is basically of type colour. So, this is the generic type in this case that means enumerated set can store any type of enumerated type. So, here we want to store the set of all colours actually that is what we want to say.

Now what we have defined, we define 6 different objects. They are set 1, set 2, set 6. So, all these objects are basically collection we can say. So, these are the 6 different collections are defined and we will now include elements in it. Now, how we can include the elements? Now here you see set 1. Now we want to include the set 1 and we call the of method.

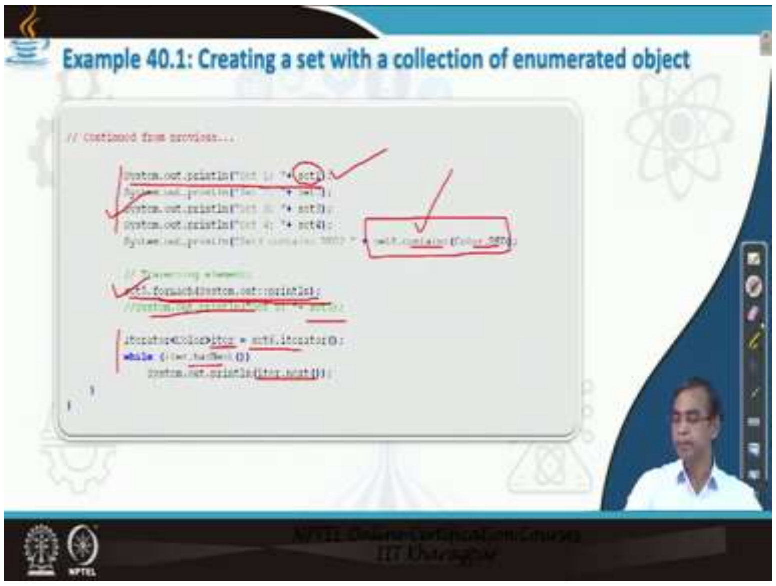
So, here are basically all these elements that we have passed it. So it basically create a set that is the set1 which includes all these colours like red, blue, green like. Now another set, set 2 is initialized with this kind of now we see compliment of set 1. So, set 1 contains this one and then final set is this one compliment means whatever the elements which belong to set 1 except those, it will include this one.

This mean that this will include the blue, blue is there, red, blue, except red, blue, it will green, green is also there. So, it will contain these one basically compliments of set 1 is basically there. So, set 2 will include all these elements in it. Now here again, set 3 all of, colour dot class. You can understand what it does. It basically contains the set 3 is basically all colours which are there. So, the set 3 is basically all colours.

Now range; colour black and violet. Now here with respect to this one the black and violet so this will include all these elements into this set, set 4. Now copy of set 2 so basically set 5 and set 2 are the same. Now, none of set 1 this means that whichever the sets it is there, it will include set 6 will include which none of this one means it basically complement again.

So, it basically, except dot this one, it will include all the sets, all these elements are there. So, now what we have done we have created is a set of sets rather with calling different methods and therefore the sets are ready. Now then we can perform many operations. I will discuss few operations here, but many more operations will be discussed in our next video class.

(Refer Slide Time: 35:26)



Example 40.1: Creating a set with a collection of enumerated object

```
// Continued from previous...

    System.out.println("Set 1: "+ set1);
    System.out.println("Set 2: "+ set2);
    System.out.println("Set 3: "+ set3);
    System.out.println("Set 4: "+ set4);
    System.out.println("Set4 contains RED? " + set4.contains(Color.RED));

    // Traversing elements
    set5.forEach(System.out::println);
    //System.out.println("Set 5: "+ set5);

    Iterator<Color>itor = set6.iterator();
    while (itor.hasNext())
        System.out.println(itor.next());
  }
}
```

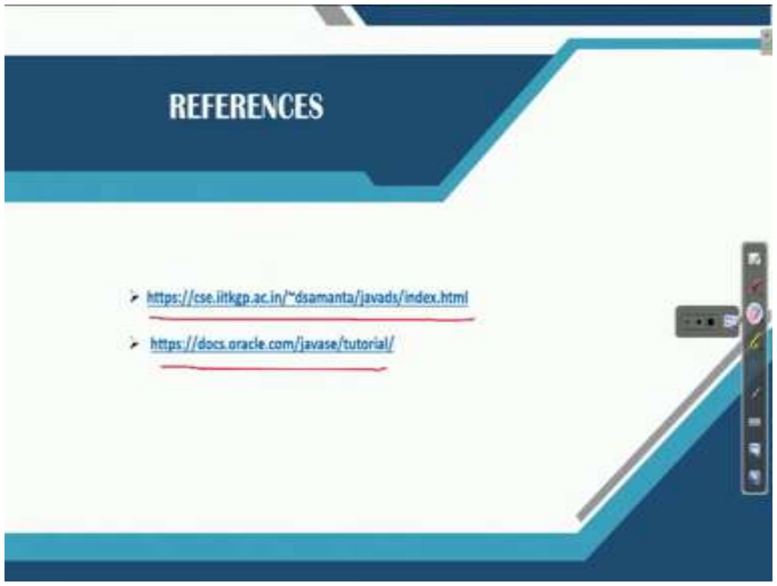
Now, here few operations that I have mentioned here. First of all, printing its sets simple println statement can be used passing the collection name. Then it will print all the elements in it that you can check it. So, here you see how we have printed all the elements which are there in set 1, set 2, set 3, set 4. Now also, we can print this way set 4 contain this one it basically return true or false.

It search weather the set 4 contains this colour red or not, it will return, true or false. So, this is the contents method is that you can. And this content method is declaring collection, right you know. And this way it is also applicable to this Enum set also Now, here is again, for each this method is called the one traversal method. It is basically it will print just like println, set 5 all elements in the set 5.

So, it is basically same as system dot out dot println statement, but it is the new syntax that you are learning for each system dot out dot println means all elements which are there in set 5 it basically redirect to the standard output called the system dot out that is the console or monitor. Now there is another way of the set can be traverse. We have learned about very neat method, but here is an iterator.

So, we want to iterate set 6. So, we can declare an iterator objects and it basically check each objects and print it. So, this way a set can be traverse. So, in this discussion, we have learned few basic things about that how the set collection can be facilitated using Java collection framework. And therefore a simple illustration by which the Enum set can be utilized.

(Refer Slide Time: 37:19)



REFERENCES

- https://cse.iitkgp.ac.in/~dsamanta/javads/index.html
- https://docs.oracle.com/javase/tutorial/

I hope you have understood this concept here. And if you are interested to learn many more things about this set collection, a very nice article is maintained here in this link, actually. So I should suggest to go through this link. This is very important link that you should follow after each lecture coverage then you can understand about in details, because whatever the thing that I am telling you are listening, sometimes you may not follow properly, but some students may think about the document that is available in the writing form.

And then they can go through the documents according to own level of understanding so they can better. So, this link is fine. And in addition to the good pedagogical discussion there is also lot of programs and both slides and everything is bundle there. So, it will really very helpful. And for this is the link for the advanced learner. If you want to learn many more things about the set related collection framework, then this is the ultimate what is called a link that you should traverse.

This is official website that is maintained by the Oracle and which is the basically the master documents for learning anything. So, this is a topic that we have covered for today and we will discuss about more many more other classes are there, particularly programming related issues are there we shall discuss these things in our next slides okay rather next lectures, okay. Thank you.