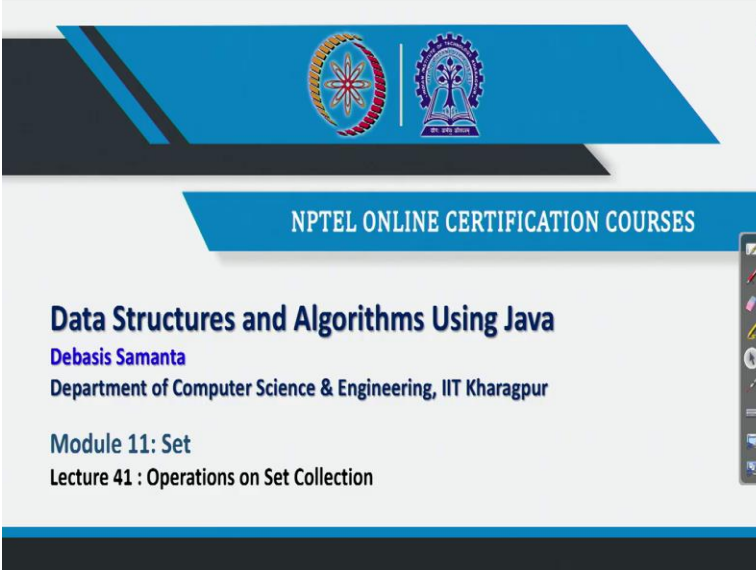


Data Structures and Algorithms Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 41
Operations on Set Collection

(Refer Slide time: 0:33)



The image shows a slide from an NPTEL online certification course. At the top, there is a blue banner with two logos: the Indian Institute of Technology Kharagpur logo on the left and the NPTEL logo on the right. Below the banner, the text reads "NPTEL ONLINE CERTIFICATION COURSES". The main title of the course is "Data Structures and Algorithms Using Java", followed by the instructor's name "Debasis Samanta" and his affiliation "Department of Computer Science & Engineering, IIT Kharagpur". The slide also indicates the current module and lecture: "Module 11: Set" and "Lecture 41 : Operations on Set Collection". A vertical toolbar with various icons is visible on the right side of the slide.

We are discussing the set collection in the last video lecture. Under this module we have discussed about the basic concept of set and particularly the Java collection framework which supports manipulation of sets. So today we will discuss few other classes which are there in the Java collection framework related to the set collection and then we will also illustrate few applications of this set particularly to in the direction of relational algebra actually.

(Refer Slide Time: 1:03)

CONCEPTS COVERED

- Set Using HashSet
- Set Using TreeSet
- Set Using LinkedHashSet
- Bulk Operations on Set Collection

A Venn diagram with three overlapping circles labeled *a*, *b*, and *c*. The regions are numbered as follows: *a* only: ∞; *b* only: 1; *c* only: 13; *a* and *b*: 3; *a* and *c*: 0; *b* and *c*: ∞; *a*, *b*, and *c*: 2. There are also some numbers like 4, 56, and ∞ scattered in the diagram.

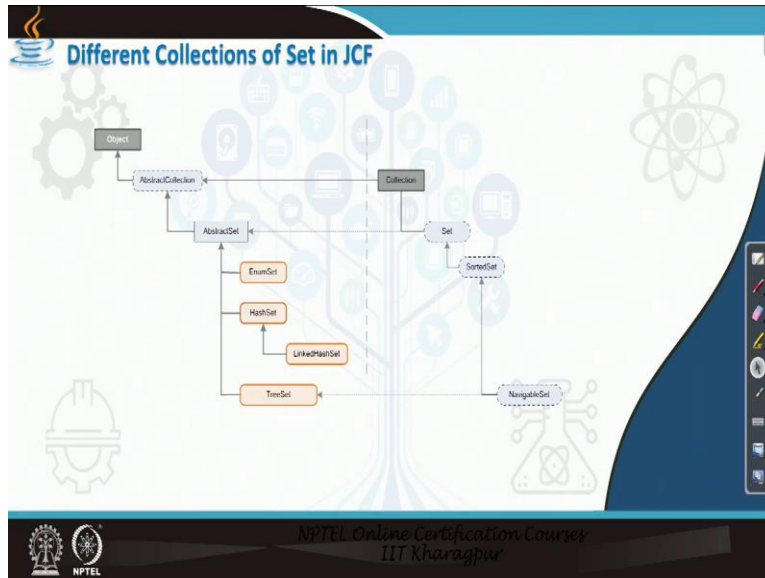
So today, our topics include the hash set and the tree set and linked hash set. These are the three major classes are there. And then we will discuss illustration of each sets I mean using creating sets and their manipulation of each type that mean hash set, tree set and linked hash set. And there are certain bulk operation which is very interesting to learn also will be covered in this lecture class.

(Refer Slide Time: 1:28)

Set Collections

The slide features a central illustration of a coffee cup with steam rising from it. Surrounding the coffee cup are various icons representing technology and science, including gears, a tree with nodes, a hard hat, a circuit board, and an atom symbol. The background is light blue with a dark blue curved border on the right side.

NPTEL Online Certification Course
IIT Kharagpur



So let us first see the set collection which we have already studied about, it has a couple of interfaces and collection. And collections mainly in enum sets which we have already studied in the last video lecture. And in this lecture we will try to cover hash sets, linked hash set and then tree sets.

(Refer Slide Time: 1:47)

Creating a set collection

- Set Interface in Java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to store the duplicate items. We can store a null element in Set.
- Set can be implemented by anyone of the three classes: HashSet, LinkedHashSet, and TreeSet.

```

HashSet<data-type> s1 = new HashSet<data-type>();
LinkedHashSet<data-type> s2 = new LinkedHashSet<data-type>();
TreeSet<data-type> s3 = new TreeSet<data-type>();

```

- Alternatively, you can use the following declaration as all of the class essentially implements Set interface.

```

Set<data-type> s1 = new HashSet<data-type>();
Set<data-type> s2 = new LinkedHashSet<data-type>();
Set<data-type> s3 = new TreeSet<data-type>();

```

NPTEL Online Certification Courses
IIT Kharagpur

Creating a set collection

Note:

- Set being a generic interface, it can contain objects of any class, such as String, Integer, Person (a user-defined class), etc.
- Like an array, a particular set contains only homogeneous types of objects.
- Unlike an array, it cannot include objects in duplicate.

In the next, you will learn how to create sets using the above-mentioned classes.

NPTEL Online Certification Courses
IIT Kharagpur

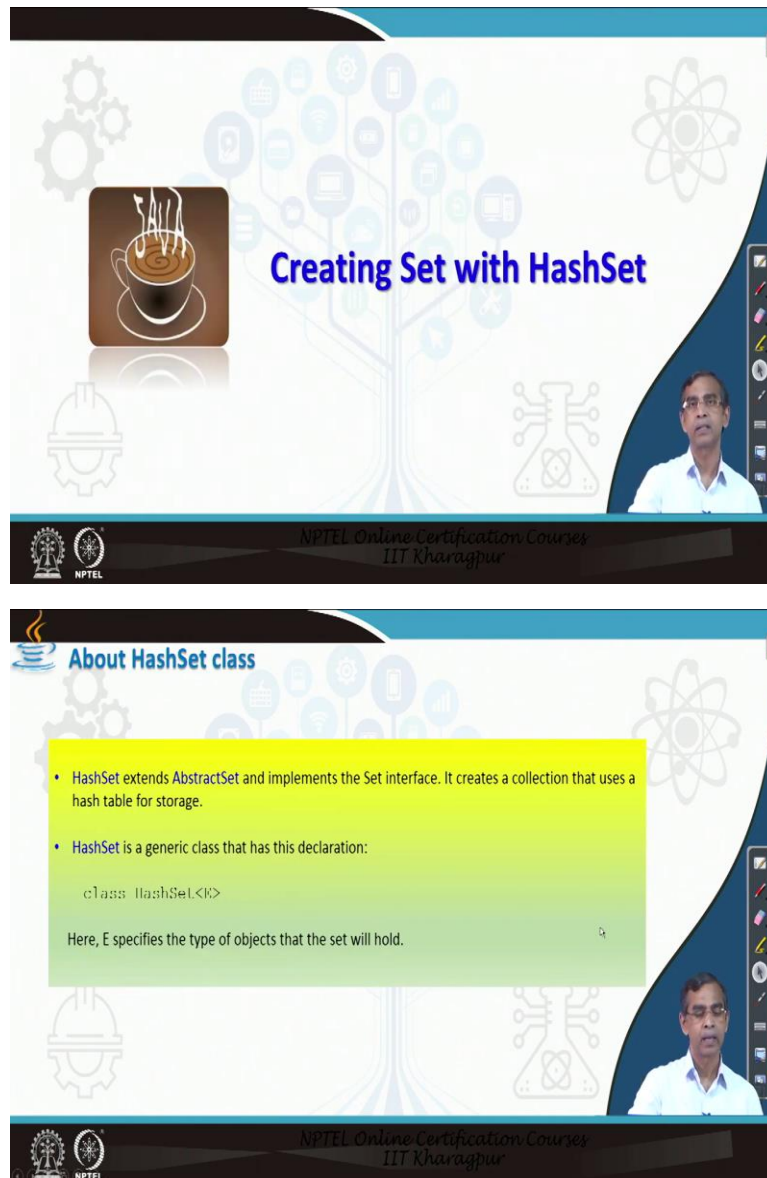
Now here the, so for the creating a set collection I just want to mention one thing, all are basically as a implementation of set interface. So set is basically, is a the super of all classes we can say. Now, I just give you an idea about or syntax that how a particular set, type of set or other, either hash set or navigable set or tree set can be stored there. Now, this is basically the syntax that you can follow. So, I am telling about these are the hash set, linked set and tree sets.

The enum set is not under this character because enum set cannot store the any type of type. It is just only enumerated like. Anyway so if you want to create an hash set collection, so let s1, so data type this is basically generic, that means whether integers, string or user defined and this is the hash set. So this is the one method. Similarly linked hash set if you want to use, this is the, this is basically obvious methods are there. I will discuss about what are the different constructors are there in order to create the different type of set collection of course, but this is the idea about.

So, if you want to create a set using tree set then this is the syntax that you can follow. Alternatively you can use this also, this is basically set because any one type either hash set, link set, tree set or whatever it is they are basically set. So this we can declare this set and then we can create a particular type of the set collection. So this is also the syntax is permissible. It is up to you anyway, no issue it is there.

But whenever it is there the binding respect to that means, you can store any set collection and you can refer to this one. So dynamic binding is possible if you follow this kind of declaration while you create set. Anyway so these are the different syntax that you can follow it. Now every class wise, I mean hash set, linked set, tree set we will discuss the different constructors and then the methods which are there and then how we can use them in our programming.

(Refer Slide time: 4:02)



Creating Set with HashSet

NPTEL Online Certification Courses
IIT Kharagpur

About HashSet class

- HashSet extends AbstractSet and implements the Set interface. It creates a collection that uses a hash table for storage.
- HashSet is a generic class that has this declaration:

```
class HashSet<E>
```


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

NPTEL Online Certification Courses
IIT Kharagpur

So let us see, first let us discuss with hash set first. Now hash set basically is a, is a generic class that we have already discussed about, so it can store any type of elements.

(Refer Slide Time: 4:17)

The slide is titled "Constructors of HashSet class". It features a table with two columns: "Constructor" and "Description".

Constructor	Description
<code>HashSet()</code>	It is a default constructor to create a hash set.
<code>HashSet(Collection<? extends E> c)</code>	It initializes the hash set by using the elements of c.
<code>HashSet(int capacity)</code>	It initializes the capacity of the hash set to capacity.
<code>HashSet(int capacity, float fillRatio)</code>	It initializes both the capacity and the fill ratio (also called load capacity) of the hash set from its arguments. The fill ratio must be between 0.0 and 1.0, and it determines how full the hash set can be before it is resized upward.

Note:

- Specifically, when the number of elements is greater than the capacity of the hash set multiplied by its fill ratio, the hash set is expanded. For constructors that do not take a fill ratio, 0.75 is used.

The slide also includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur" at the bottom.

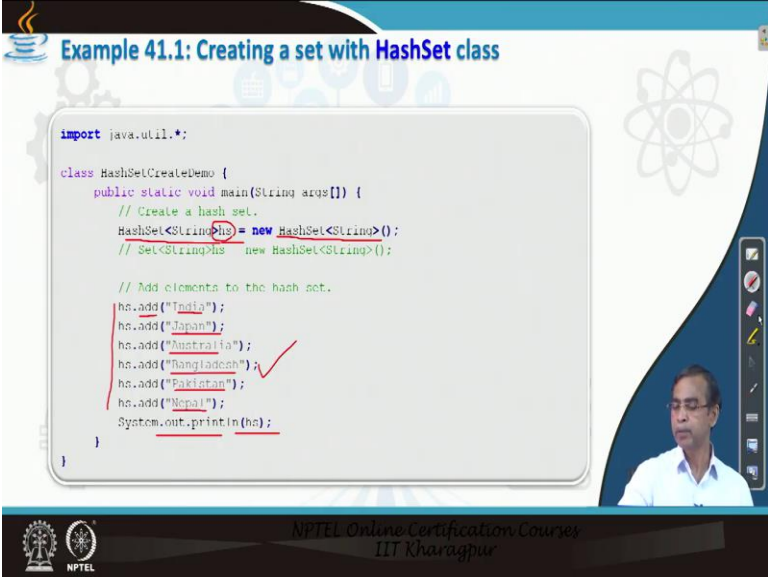
Now here are constructors by which you can create an object that means is a collection object, or the collection object of type hash set. So there are four constructors, this is the default constructor. Default constructor means it will create x set of default size, default size is usually 16 is the number or size there. Now, this is another constructor if you have already a collection of any type, maybe array list or it is other type of collection, you can pass it as an argument and then a hash set can be created. But whenever you pass this array list, now you note that array list (con) may include some duplicate number.

So whenever the hash set is created or rather object is initialized it will remove all the duplicate occurrences. Now, these are another constructor by which you can specify the initial size of the set that you want to have. So capacity is the argument for that. And here is the another is that capacity as well as the fill ratio. By default the fill ratio is 0 point 75 or 75 percent, otherwise you can mention the fill ratio that means if a set initially may be of size 16 and when it is filled automatically it will grow, so automatically or dynamically the size of the set grow to accommodate many more elements to be there.

So that is why the it automatically in which rate it will grow its size, based that basically decides. So if you mention point 5 that means 50 percent, so present size is 32 so it will increase another 16. So 48 will be the next size and so on. So this basically the idea about the different way the

hash set can be created. So this is the constructors in order to create the object of type hash set collection.

(Refer Slide Time: 6:20)



The slide displays a Java code snippet for creating a HashSet. The code is as follows:

```
import java.util.*;

class HashSetCreateDemo {
    public static void main(String args[]) {
        // Create a hash set.
        HashSet<String> hs = new HashSet<String>();
        // Set<String> hs = new HashSet<String>();

        // Add elements to the hash set.
        hs.add("India");
        hs.add("Japan");
        hs.add("Australia");
        hs.add("Bangladesh");
        hs.add("Pakistan");
        hs.add("Nepal");
        System.out.println(hs);
    }
}
```

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, let us see a hash set it does not have any method of its own. All the methods which are declared there in set are basically the method we need and we have already discussed while we are discussing about set interface that which are the methods are there and it again it does not give you that all the elements which should be in order or in the order in which you insert into these elements.

So that ordering is not guaranteed, usually it gives ordering but not necessary that same order that you will get, either not in order means ascending or descending or in the order you have inserted the elements into it. Now, let us start with an example so that we can understand. It is better I have intentionally used a very small example so that you can understand very quickly and then that example can be extended, for that further you can make it more real-life problem solving actually.

So let us first see, in this program we will see how a hash set can be created and we want to create an hash set of some string objects. So, to do these things what we do is that we create the hash set of type string and this is the name of the set, let this the hs. So, this is the default constructor is used, though so initial size of the hs is 16. Now we, we therefore add elements into these sets, so adding this element by calling the add method.

Add method is declared in the collection and it basically also inherited to the set interface because collection implements set and then hash set is basically is an, hash set implements set, that way it is also coming that way. So these are the few what is called the insertion of elements into this collection we can say, these are the elements like these are the elements we have added it. And you see all are basically unique element no duplicates, so no issue.

You can write one add method where you can add again Bangladesh for example. Then you see it will not accept the later one, it will only preserve this one only. So later addition of the same element or duplicate element will be ignored. It will not return any error or exception of course. So this hash set is created, finally you can use any traversal algorithm any iterator. Here I have used println method simply and it will print all the elements those are there in this set collection. So, this is a very simple example by which you can understand about that a default constructor can be used to create a hash sets.

(Refer Slide Time: 9:05)

Example 41.2: Adding a duplicate object

```
/* The following program illustrates that if you add an object, which is already in the set, then that inclusion will be ignored automatically. */
import java.util.*;
import java.util.Set;

public class HashSetDuplicateDemo {
    public static void main(String[] args) {
        // Creating a HashSet
        Set<String> daysOfWeek = new HashSet<>();

        // Adding new elements to the HashSet
        daysOfWeek.add("Monday");
        daysOfWeek.add("Tuesday");
        daysOfWeek.add("Wednesday");
        daysOfWeek.add("Thursday");
        daysOfWeek.add("Friday");
        daysOfWeek.add("Saturday");
        daysOfWeek.add("Sunday");

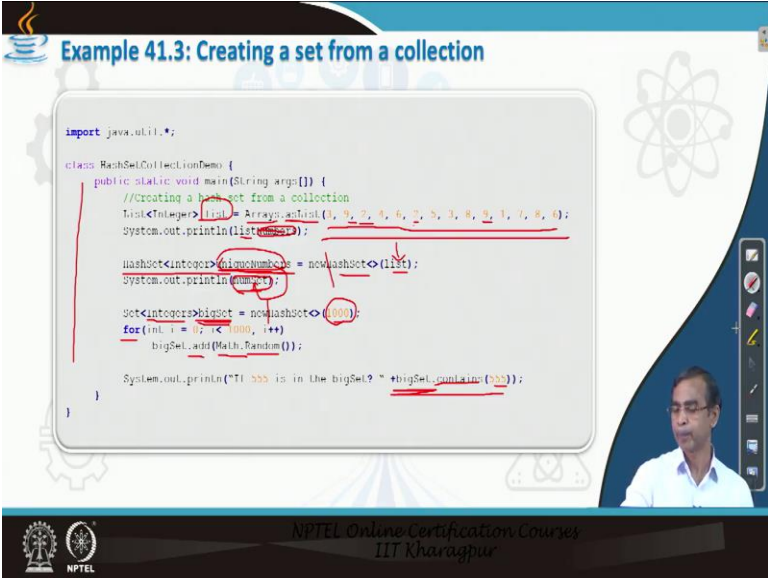
        // Adding duplicate elements will be ignored
        daysOfWeek.add("Monday");

        System.out.println(daysOfWeek);
    }
}
```

Now, here is another example, this example okay illustrate if we add an elements which is already present in the set then what will happen. Now let us see how we declare here hash set of string, days of week and so this is the initial size is 16 the default. And we add seven days into this sets in any order you can do. Here you have used this order, no issue you can use any other order. And that set will store automatically using its own internal mechanism and not necessarily in the same order you have added into the set.

Now, if you add further the element say Monday which is already been added here, then you will see it will not accept it and then if you print this it will print this one. So this, it is not that okay after the Sunday this Monday will be added here in this, it is not like that, so it will not be. So, this means duplicate element automatically taken care, you do not have to bother about it so the set will always include all elements those are unique.

(Refer Slide Time: 10:17)



The slide displays the following Java code:

```
import java.util.*;

class HashSetCollectionDemo {
    public static void main(String args[]) {
        //Creating a list set from a collection
        List<Integer> list = Arrays.asList(1, 9, 2, 3, 6, 7, 9, 3, 8, 2, 1, 7, 8, 0);
        System.out.println(list);

        HashSet<Integer> uniqueNumbers = new HashSet<>(list);
        System.out.println(uniqueNumbers);

        Set<Integer> bigSet = new HashSet<>(1000);
        for (int i = 0; i < 1000; i++)
            bigSet.add(Math.Random());

        System.out.println("11 999 is in the bigSet? " + bigSet.contains(999));
    }
}
```

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

Now here we can consider few more examples regarding how the hash set can be created with an already collection at hand. So with an input collection we want to create a set and that input collection may be any type of collection may be arrays or array list or vector or what type of it. And that again you should note that, that collection made contain duplicate because there is no bar about occurrence of duplicate elements. But whenever we add a collection into as a set then it automatically the duplicate element will be pruned and then only the unique element will be stored into the set.

Now here is an example here we just create a list first. List is a collection of our own we create the list using arrays, an array as list is a method for the arrays class you are already familiar to. And this is basically initial elements. And you can see few elements are duplicates so these are the duplicate elements, as we see 9, 9 here and so. So many elements are in duplicates actually. Now, anyway so this list can permit you to store all this elements.

So this is basically, the list is a collection and what we want to do, we want to create a set with, with the elements taking from this list. Now, what how we can do let us first define a hash set, so we define one set, hash set actually the name of the hash set is unique numbers. And so this is basically list is list of numbers actually. So this actually it is there is not, it is not correct, it is it should be list only, so list. Now so unique numbers we create a set and so it is basically hash sets. Now here you see we pass list as an argument to this constructor.

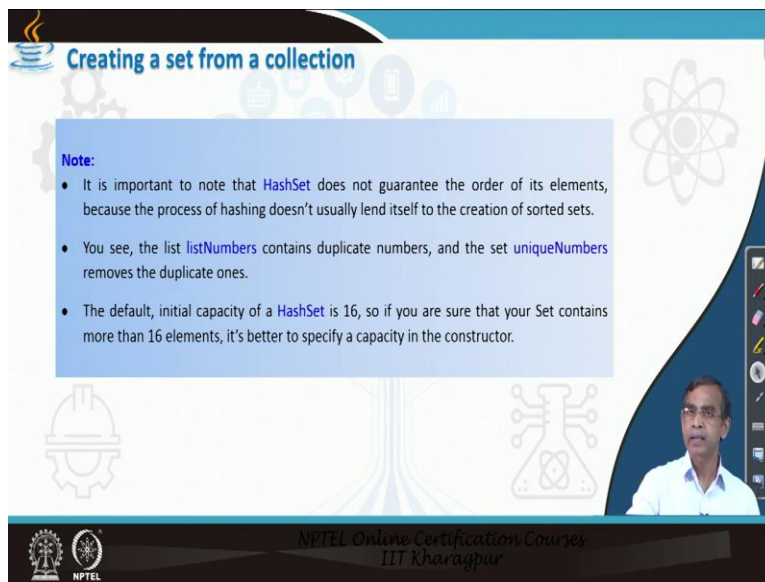
So hash set will take this list that means this list and we create a set, the name of the set is unique numbers. And then we can just simply print these sets here, num set like. It is basically not num set, there is a mistake I will correct this mistake of course. It is basically, it should be unique numbers because this is the set that we have created so this should be passed as an argument here.

So, it is unique numbers. Now we can so this basically you can understand about that all the elements which are there in the list whenever they store in the form of a sets and we print the sets it basically print as an all elements in the set which are there but in a unique manner. Now here another set we are creating this is basically set of integers. So, the set of integers and 1000 is basically as you know it is basically the capacity.

So we just create a set of capacity 1000. Now we can add elements into it. Here again we just use a for loop. And here a random method is called which is defined in the math class which is in the line package. So using this math it basically generate a random number and add this number into the set, big sets. So 1000 random number will be stored into the set, big sets. Now we can use any iterator to print all the elements in this set big sets and then see the results. But here we can just do one more method call, it is called the contents, contents already you have learned about it.

So it is basically check whether 555 elements is already there in this set or not. So this will return either true or false if it present or not respectively. So this basically an example that you can understood that how a set can be created with an existing collection as an input to create it. And then any other method regarding the first element, last element you can call and apply to the set, you can see the result also. So those things left as an exercise. You, you are advised you are recommended to practice with calling many methods, and just rewriting this program and run this so that you can use it and also enjoy it.

(Refer Slide Time: 14:50)



Creating a set from a collection

Note:

- It is important to note that `HashSet` does not guarantee the order of its elements, because the process of hashing doesn't usually lend itself to the creation of sorted sets.
- You see, the list `listNumbers` contains duplicate numbers, and the set `uniqueNumbers` removes the duplicate ones.
- The default, initial capacity of a `HashSet` is 16, so if you are sure that your Set contains more than 16 elements, it's better to specify a capacity in the constructor.

NPTEL Online Certification Courses
IIT Kharagpur

So this is the example that we have illustrated about how a set can be created from a collection.

(Refer Slide time: 14:55)



Traversing a Set

NPTEL Online Certification Courses
IIT Kharagpur

The following example shows different ways of iterating over a `HashSet`

- Iterate over a `HashSet` using simple for-each loop.
- Iterate over a `HashSet` using `iterator()`.
- Iterate over a `HashSet` using Java `forEach` and Lambda expression.
- Iterate over a `HashSet` using `iterator()` and `forEachRemaining()` method.

NPTEL Online Certification Courses
IIT Kharagpur

Now regarding the traversing a set, this is very important. There are many traversal method can be applied and the, I will just give you the syntax of the different traversals method that you can think about. And there are many traversals method that, that can be considered or can be applied for like say for-each loop is a one traversal technique, iterator you are already using it, it also can be, for each loop also I have time to type while I was discussing many collections and print the collection other than `System.out.println` I use many traversals also.

So, by the time you are already familiar about the syntax okay. And so these are the things you can see and there is another lambda expression which has been included from (Java) JDK 7 onwards. This is also one very sophisticated syntax actually that you can consider to print the entire sets. And there is also set using iterator and for each remaining there is another technique.

So four different what is called the method of iteration let us exam, illustrate all these method of iterations in a, in a simple program again, so that you can understand about it, you can practice and the same things you can practice for any other collections again. I am applying it for this collection, set collection but you can apply to any other collection which you have already studied and you can practice it again.


(Refer Slide Time: 16:25)

Example 41.4: Traversing a set using Iterator

```
import java.util.*;

public class SetTraversalDemo {
    public static void main(String[] args) {
        Set<String> pLangs = new HashSet<>();
        pLangs.add("C++");
        pLangs.add("Java");
        pLangs.add("Python");
        pLangs.add("PHP");
        pLangs.add("R");

        // Using simple for-each loop
        for (String pl: pLangs) {
            System.out.println(pl);
        }
    }
} // Continued to next ...
```




Example 41.4: Traversing a set using Iterator

```
// Continued on...

// Using iterator()
Iterator<String> iter = pLangs.iterator();
while (iter.hasNext()) {
    String pl = iter.next();
    System.out.println(pl);
}

// Using forEach and lambda expression
pLangs.forEach(pLangs -> { System.out.println(pLangs); });

// Using iterator() and forEachRemaining() method
iter = pLangs.iterator();
iter.forEachRemaining(pLangs -> { System.out.println(pLangs); });
}
} +
```



So let us first discuss about a simple program. First we have to create a set so we define a set called pLangs, is basically a set of programming languages and we, this is basically string of type and these are the different elements we have added into this set. Now I am just traversing using for each loop. Now this is the syntax that you should note it. So basically, for, this is a syntax for, it is just like a for, and then here basically what element you want to syntax.

So pl is basically any elements which basically belongs to these pLangs will access automatically from the set and it will printed it. So for each elements which is there in this sets one by one it will face it and then print it here. So this is the one very, very smart rather syntax to print

elements, otherwise you can write `system.out.println`, `pLangs` also, this is one. But you see that while this `println` statement, here no customize is there.

Whenever you print then all the elements will be one by one extra pose, so there will be no space, blank and everything but here `println` you can understand one language is printed the next line will be there, so this is better actually. You can just find the difference. So in lieu of this one, in lieu of this one, the usual `system.out.println` `pLangs` also can be. This is very naive methods of printing this one but it is not advisable because it does not have any customization.

So for the printing the elements on the console is concerned, but there are all the iterator you can more customize more systematic way you can display in, that is what it is recommendable actually. Now so these are the method that, this is, this is the one example using the `for` (loop) for each loop. There are many other iterator also, I have already mentioned this is the one let us consider the another iterator is there.

Here if this is iterator actually, this is already you are familiar to so this basically iterator class is there you have to iterate over the string elements object of type string actually. This is the object type that you have to iterate. This is iterator object this is basically the pointer you can say and it, it will iterate over the set `pLangs`. So this is the syntax that you should think about that if you want to iterate over a set and the name of the set dot iterator method should be called.

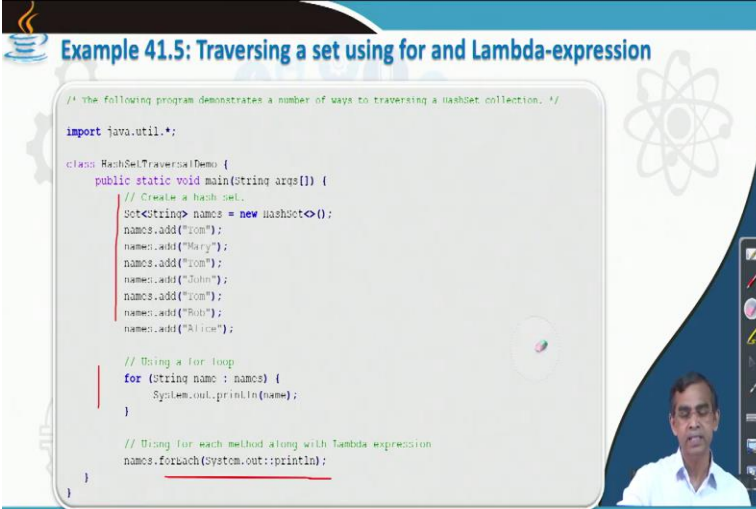
Then has `next` is, basically it will reach, it will check whether this set is empty or not, this collection is empty or not. If it is not empty then it will basically face the next element and print this element. So it is `next` method is there for this iterator this if it is called there is no one actually, so it is the mistake, typing mistake so `iter`, `iter` dot `next` and then return `pl` and it will basically print the elements that is there in the sets.

So this is a one method, it is iterator. Now `next` is the `for each` lambda expression, you note the syntax actually. Again `for each` is the method and for which set, it is the name of the set so this `set` dot `for each` and here is basically we just direct the `pLangs` sets basically redirects to this one. So basically it will accepts an element in from here and then redirect it to this and it will print into this one.

Here you can write X belongs to this and that also you can do it also no issue there. So this way it basically print all the elements which are there in the set. So this is the one way of traversing. The next another way this is called the iterator using for each remaining method. Now let us see first of all we have to I mean create an iterator object for this, so pLangs is the target sets for which you want to iterate and this is for each remaining the same syntax it is like this one also, you can use it to print all the elements.

So, these two are the same of course but here it is using iterator and using it is just only for each lambda expression. So, these are the different ways by which you can traverse any sets or any collection rather. So it is in the context of set but it is also equally applicable to any collection which you have already learned about it.

(Refer Slide Time: 21:02)



The slide displays a code editor window with the following Java code:

```
/* the following program demonstrates a number of ways to traversing a HashSet collection. */
import java.util.*;

class HashSetTraversalsDemo {
    public static void main(String args[]) {
        // Create a hash set.
        Set<String> names = new HashSet<>();
        names.add("Tom");
        names.add("Mary");
        names.add("Tom");
        names.add("John");
        names.add("Tom");
        names.add("Bob");
        names.add("Alice");

        // Using a for loop
        for (String name : names) {
            System.out.println(name);
        }

        // Using for each method along with Lambda expression
        names.forEach(System.out::println);
    }
}
```

The slide also features the NPTEL logo, a small portrait of a man, and the text 'NPTEL Online Certification Courses IIT Kharagpur' at the bottom.

Now this is an another example using lambda expression further. This is the set that we have created and this is the for statement also you can using traverse, and this is the for each also we have covered. So this is the same program again duplicated anyway, so this program has no any other special implications. We have already discussed this program earlier also. So this is example.

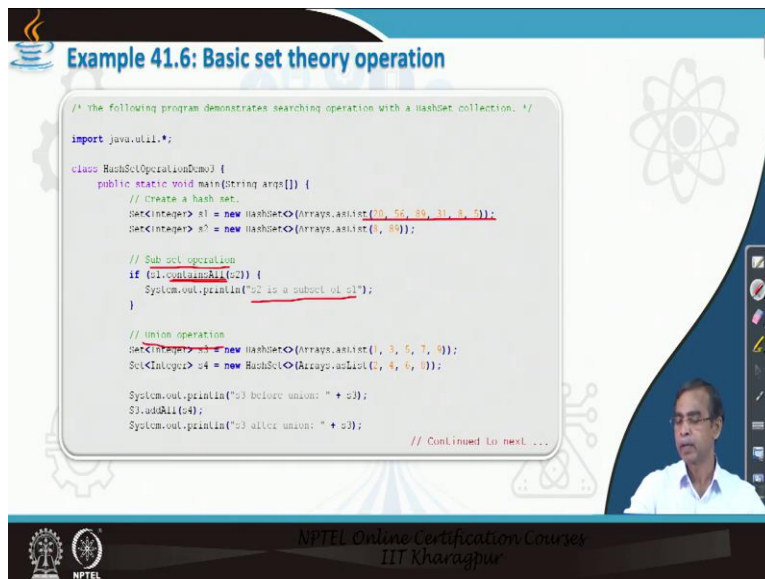
(Refer Slide Time: 21:30)



The slide features a central title "Basic Set Theory Operation" in blue text. To the left is an icon of a coffee cup with steam. To the right is a stylized tree diagram with various icons as branches. The background is light blue with faint icons of gears, a hard hat, and a beaker. A small video inset of a man is in the bottom right corner. The NPTEL logo and "NPTEL Online Certification Courses IIT Kharagpur" are at the bottom.

Now let us consider few more examples regarding the basic set theoretical operations. We already told about that set is mainly important because of its set algebraic operation a set algebra rather and this is also called relational algebra. And in case of relational algebra basic operations like complement and then there are few more operations like union, intersection, then addition, subtraction all those operations are there. Now we will see exactly how all those operations can be carried out for the sets as an operand.

(Refer Slide Time: 22:10)



The slide is titled "Example 41.6: Basic set theory operation". It contains a code block with the following Java code:

```
/* the following program demonstrates searching operation with a HashSet collection. */
import java.util.*;

class HashSetOperationDemo3 {
    public static void main(String args[]) {
        // Create a hash set.
        Set<Integer> s1 = new HashSet<>(Arrays.asList(20, 56, 89, 31, 8, 9));
        Set<Integer> s2 = new HashSet<>(Arrays.asList(1, 89));

        // Sub set operation
        if (s1.containsAll(s2)) {
            System.out.println("s2 is a subset of s1");
        }

        // union operation
        Set<Integer> s3 = new HashSet<>(Arrays.asList(1, 3, 5, 7, 9));
        Set<Integer> s4 = new HashSet<>(Arrays.asList(2, 4, 6, 8));

        System.out.println("s3 before union: " + s3);
        s3.addAll(s4);
        System.out.println("s3 after union: " + s3);

        // Continued to next ....
    }
}
```

The slide also features a small video inset of a man in the bottom right corner and the NPTEL logo and "NPTEL Online Certification Courses IIT Kharagpur" at the bottom.

Example 41.6: Basic set theory operation

```

/* the following program demonstrates searching operation with a hashtable collection. */
import java.util.*;

class HashSetOperationsDemo3 {
    public static void main(String args[]) {
        // Create a hash set.
        Set<Integer> s1 = new HashSet<>(Arrays.asList(10, 56, 89, 31, 8, 9));
        Set<Integer> s2 = new HashSet<>(Arrays.asList(1, 89));

        // Sub set operation
        if (s1.containsAll(s2)) {
            System.out.println("s2 is a subset of s1");
        }

        // Union operation
        Set<Integer> s3 = new HashSet<>(Arrays.asList(1, 8, 9, 10, 31, 56, 89));
        Set<Integer> s4 = new HashSet<>(Arrays.asList(1, 8, 9, 10));

        System.out.println("s3 before union: " + s3);
        s3.addAll(s4);
        System.out.println("s3 after union: " + s3);
    }
}

```

NPTEL Online Certification Courses
IIT Kharagpur

Now here is a very simple example. This example demonstrate searching operation, searching is very frequent operation it is there. Now searching operation, that can be done many ways are there, or we can say searching or finding a subset or what kind of operations are there, or a union operation whether I can say better is not that searching actually we can see in that sense. Now let us see how we can perform certain set theoretical operation like union, intersection here. Now in this slide we can see how the two sets and the union of two sets can be carried out.

So first of all we should have the two sets, so s1 and s2 are the two sets let us consider all sets are of same type that is important. What are the operations that you want to perform here this should be compatible that in they are same type like. So we create, the s1 is the set with these elements as the target I mean collection basically list, This is another list also this one. So, two I mean sets are created, these two sets are s1 and s2.

Now here if you see we are little bit doing some searching like so contains all s2 that means s1, s1 is this one. So this sets actually, if s1 this set contains all s2 that means 889 that means if s, whatever the set s2 contains, it will check whether all elements are there here or not. So it is just like both are common. It is basically there, you can say it is common elements, it is an intersection elements like.

So it then it will print that s2 is a subset of this one. So this way you can check whether one set is a subset or not, and it can be in any order as you can wish also. Now on the other hand, you can call the same method s2 dot containsAll s1, you can see what the containsAll will there, in, in

that case it will give that true, but in the other case it will give false like whatever it is there. That is basically whether a set is a subset or not that can be checked by this method containsAll.

Now, so it is called subset operation. Now let us see union operation how we can perform it. So, again consider the two sets that we have created that is s1 and s2. Now, here we create another two sets, say s3 and s4 and sending these elements belongs to the set s3 and these are the elements belong to the set s4. Now as a union of the two sets is basically include all the sets in addition to the common sets as well as that but it common set will occurs only once.

So it basically 1 2 3 4 5 6 7 8 9, this basically will give you the union of this one. Now s3 set before union it will be there. Now, here addAll method, if we add all the methods of this sets to this sets. then a new set will be added. Then when adding all the sets from here to this sets. So, essentially it is union, so this basically this one union. So you can check the union operation after the set s3 how it basically gives you. So this basically the idea about how the union operations can be performed. Now like union operation, let us consider other operations like intersection and difference operation like.

(Refer Slide Time: 26:03)

The slide displays the following Java code:

```
// Continued on...  
  
// Intersection operation  
Set<Integer> s5 = new HashSet<>(Arrays.asList(1, 2, 3, 4, 5, 7, 9));  
Set<Integer> s6 = new HashSet<>(Arrays.asList(1, 5, 6, 8));  
  
System.out.println("s5 before intersection: " + s5);  
s5.retainAll(s6);  
System.out.println("s5 after intersection: " + s5);  
  
// Set difference operation  
Set<Integer> s7 = new HashSet<>(Arrays.asList(1, 2, 3, 4, 5, 7, 9));  
Set<Integer> s8 = new HashSet<>(Arrays.asList(3, 4, 6, 8));  
  
System.out.println("s7 before difference: " + s7);  
s7.removeAll(s8);  
System.out.println("s7 after difference: " + s7);  
}
```

The slide also features a Venn diagram with two overlapping circles labeled A and B. The intersection of the two circles is shaded in red. A video inset in the bottom right corner shows a man speaking.

So here again these are the sets say s5 and s6. s5 is created from this list, s6 is created from this list, two sets. Now, here we are printing s5 before this intersection and intersection operation is basically by using this retainAll method. So it will basically retain all the elements which is both there in s5 and s6 actually. So you can print the sets that basically if you call retainAll for this s5,

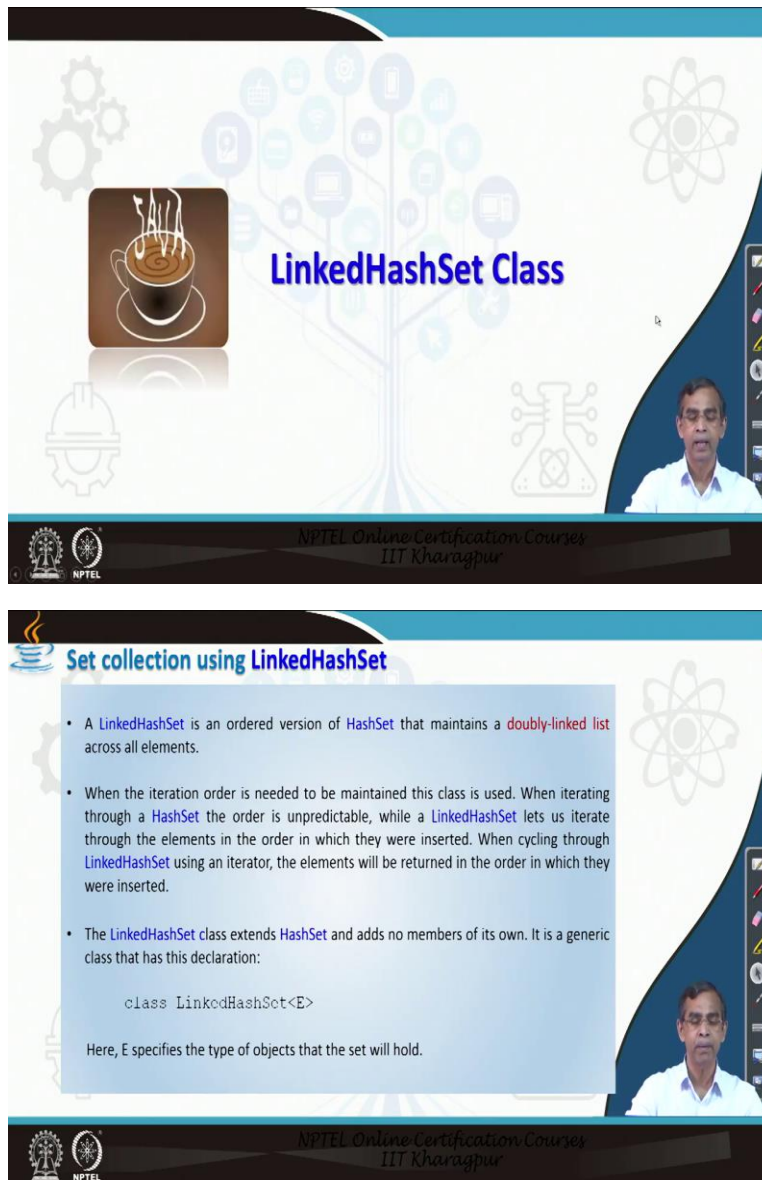
it will basically cut all the elements or remove all the non-common elements and then only common elements, those are in s5 and s6 both will be retained and then it print.

So this is basically the intersection operation that you can understand. Now here is the set difference operation. Let us consider s7 and s8 are the two sets where the elements are there. And you know, difference of the, difference operation A minus B like. So it is basically in the Venn diagram if it is A and if it is B, then it is A set and it is set B. So, these basically intersection. And then difference will be A minus B, will be this is the difference all the elements which are there.

Similarly B minus A will be in this part actually. This is basically common set, I do not just want to teach you the set theory here, I assume that you are already familiar to the set operation. Now let us see how this difference operation can be obtained like. So here, this s7 before the difference operation. And here the removeAll method, you see the different methods are there it is surprisingly not in the form of union, or intersection, or difference, but these method you have to remember somehow that okay this method like this one.

So removeAll method if we call it for this s, so that means it basically removeAll the methods which are basically a set from the set A. So that mean it will only those elements which will be there not in the set B it is there. So this way it will print the set difference A minus B like, or s7 minus s8 here in this case.

(Refer Slide Time: 28:39)



LinkedHashSet Class

NPTEL Online Certification Courses
IIT Kharagpur

Set collection using LinkedHashSet

- A `LinkedHashSet` is an ordered version of `HashSet` that maintains a **doubly-linked list** across all elements.
- When the iteration order is needed to be maintained this class is used. When iterating through a `HashSet` the order is unpredictable, while a `LinkedHashSet` lets us iterate through the elements in the order in which they were inserted. When cycling through `LinkedHashSet` using an iterator, the elements will be returned in the order in which they were inserted.
- The `LinkedHashSet` class extends `HashSet` and adds no members of its own. It is a generic class that has this declaration:

```
class LinkedHashSet<E>
```

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

NPTEL Online Certification Courses
IIT Kharagpur

So, we have already studied about the many set theoretical operations and linked hash set is the next collection. It is very similar to the set that hash map, only the storage mechanism is different. It basically stored in the form of an linked list, otherwise nothing is different there.

(Refer Slide Time: 28:55)

The slide is titled "Constructors of LinkedHashMap class". It features a table with two columns: "Constructor" and "Description".

Constructor	Description
<code>LinkedHashSet()</code>	It is a default constructor to create a hash set.
<code>LinkedHashSet(Collection<? extends E> c)</code>	It initializes the hash set by using the elements of c.
<code>LinkedHashSet(int capacity)</code>	It initializes the capacity of the hash set to capacity.
<code>LinkedHashSet(int capacity, float fillRatio)</code>	It initializes both the capacity and the fill ratio (also called load capacity) of the linked hash set from its arguments. The fill ratio must be between 0.0 and 1.0, and it determines how full the linked hash set can be before it is resized upward.

Note:

- The constructors in the `LinkedHashSet` class are in the similar form that of the constructor in `HashSet` class.
- The `LinkedHashSet` class extends `HashSet` class and implements `Set` interface.
- The `LinkedHashSet` class does not define any exclusive methods of its own. All methods are same as the methods as in `HashSet` class. This implies that whatever the operations we can perform with `HashSet` collections are also possible with the `LinkedHashSet` class.

The slide also includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur" at the bottom.

And then these are the few constructor, this is the self-explanatory. It is a default constructor you can create a set passing a collection or list like. It is basically you can mention set when you create it by mentioning the capacity, the capacity and fill ratio. All these constructor you can see, it is very similar to the hash sets and actually all methods are very similar to hash sets. So whether it is hash set or linked hash set it hardly matters, so far the different operations are concerned.

Only internally they are, they follow the different storage mechanism all methods are implementing of their own which basically programmers should not bother about how they have been implemented and everything. Something is very fast, something is very slow. Linked list definitely it is very fast, in addition to this linked list obviously if compared to hash set, it there are some methods which are of its own also that we have discussed while we are discussing the different methods in line in sets. So that you can consider about it.

(Refer Slide Time: 29:59)

Difference between HashSet and LinkedHashSet

- It is important to note that **HashSet** does not guarantee the order of its elements, because the process of hashing doesn't usually lend itself to the creation of sorted sets.
- In contrast, the **LinkedHashSet** follows the same order of the items as they are added into the set.

NPTEL Online Certification Courses
IIT Kharagpur

I have considered I have only considered or given example of few elements, few methods. There are many methods which basically, it is there in the table that we have already shown to you. You can follow those methods and you can practice in your programming.

(Refer Slide Time: 30:10)

Example 41.7: Difference between HashSet and LinkedHashSet

```
/* The following program illustrates the ordering of elements in two sets
created by HashSet and LinkedHashSet classes. */

import java.util.*;

class DifferentSetDemo {
    public static void main(String args[]) {
        // Create a hash set.
        HashSet<String>hs = new HashSet<String>();

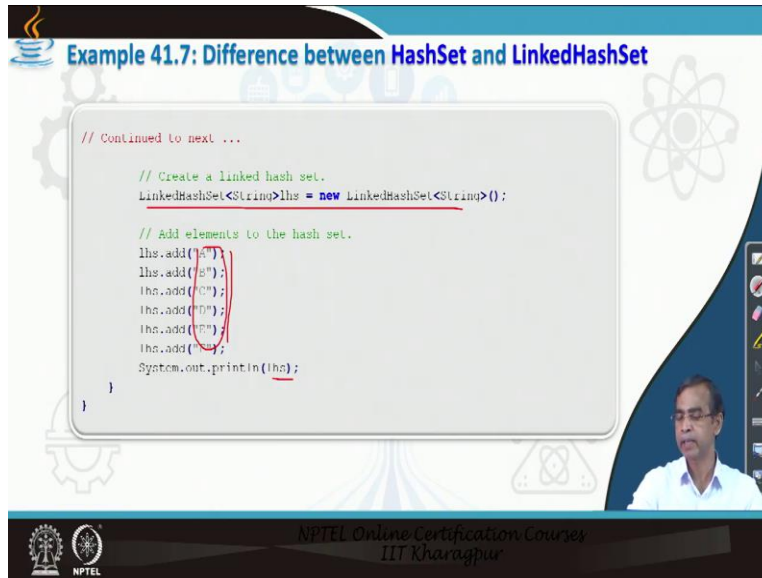
        // Add elements to the hash set.
        hs.add("A");
        hs.add("B");
        hs.add("C");
        hs.add("D");
        hs.add("E");
        hs.add("F");
        System.out.println(hs);

        // Continued to next ...
    }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Example 41.7: Difference between HashSet and LinkedHashSet

```
// Continued to next ...  
  
// Create a linked hash set.  
LinkedHashSet<String>lhs = new LinkedHashSet<String>();  
  
// Add elements to the hash set.  
lhs.add("A");  
lhs.add("B");  
lhs.add("C");  
lhs.add("D");  
lhs.add("E");  
lhs.add("F");  
System.out.println(lhs);  
}
```

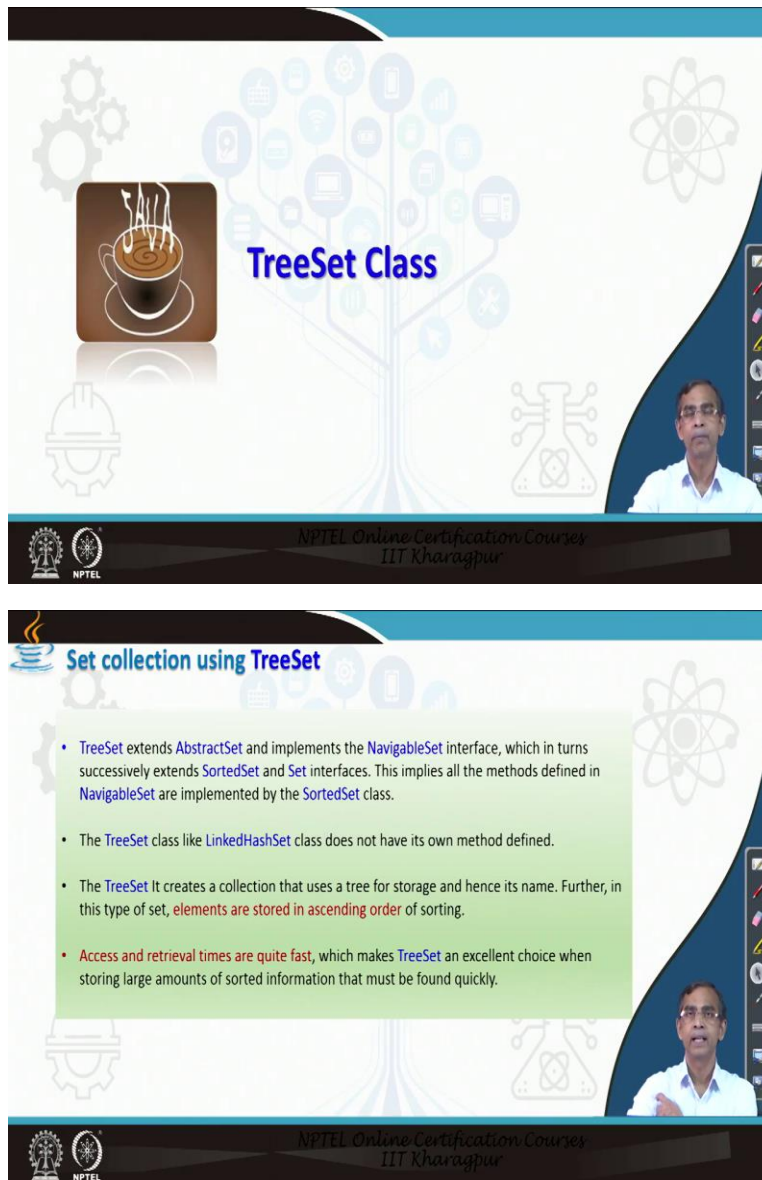


NPTEL Online Certification Courses
IIT Kharagpur

And this is an example this example basically tell about the different set, how we can create it. It is very same this is basically hash set we have created it and the same thing you will see how using the linked hash set can be created. It is basically same, only the thing is that in case of linked hash set, in each order, you add the element it will be preserved. Here in case of hash set this order may not be preserved.

Now, let us see the linked hash set formulation of the same thing we create here, but using the linked hash set actually, any order. If you print it you see that if the printed will be in the same order as it is there, which may not be there in case of hash set. That is the only difference that I want to mention that is the hash set and linked set is concerned, otherwise all the operation that we have applied for the hash set you can again rewrite the code for the linked hash set and check it. It will work.

(Refer Slide time: 31:10)



TreeSet Class

NPTEL Online Certification Courses
IIT Kharagpur

Set collection using TreeSet

- **TreeSet** extends **AbstractSet** and implements the **NavigableSet** interface, which in turn successively extends **SortedSet** and **Set** interfaces. This implies all the methods defined in **NavigableSet** are implemented by the **SortedSet** class.
- The **TreeSet** class like **LinkedHashSet** class does not have its own method defined.
- The **TreeSet** It creates a collection that uses a tree for storage and hence its name. Further, in this type of set, **elements are stored in ascending order** of sorting.
- **Access and retrieval times are quite fast**, which makes **TreeSet** an excellent choice when storing large amounts of sorted information that must be found quickly.

NPTEL Online Certification Courses
IIT Kharagpur

All intersection, union, difference which we have already explained using hash set you can repeat the same thing using linked set or tree set also you can do it. Now tree set is the fastest the set storage mechanism actually. If very large set if you want to maintain and then you need the very first operation then definitely tree set is the solution for that.

(Refer Slide Time: 31:31)

Constructor	Description
<code>TreeSet()</code>	It is a default constructor to create an empty set that will be sorted in ascending order according to the natural order of its elements.
<code>TreeSet(Collection<? extends E> c)</code>	It builds a tree set that contains the elements of <code>c</code> , where <code>c</code> is any collection.
<code>TreeSet(Comparator<? super E> comp)</code>	It creates an empty tree set that will be sorted according to the comparator specified by <code>comp</code> .
<code>TreeSet(SortedSet<E> ss)</code>	It builds a tree set that contains the elements of <code>ss</code> .

Note:

- In all cases, the capacity grows automatically as elements are added with set structure.

NPTEL Online Certification Courses
IIT Kharagpur

Like hash set and linked (has) linked hash set it has all the four constructor very similar manner. So these are the different way the set can be created, only the thing that in case of tree set we have to define the comparator which may not be not there. So all these set basically default for the, where the comparator method need not to be there, but this method if you want to create your own to store your own user-defined type then you can do it. And this is also sorted set you can create and then that can be used as a, as an input to create a tree set for the fast rate operation only. So these are the different way the tree set, the another set collection can be stored, can be maintained.

(Refer Slide time: 32:18)

Example 41.8: Creating a TreeSet collection


```
/* The following program illustrates the creating a tree sets created by TreeSet class. */
import java.util.*;

class TreeSetDemo {
    public static void main(String args[]) {
        // Create a tree set.
        TreeSet<String> ts = new TreeSet<String>();

        // Add elements to the tree set.
        ts.add("D");
        ts.add("E");
        ts.add("B");
        ts.add("A");
        ts.add("C");
        ts.add("F");
        ts.add("G");
        System.out.println(ts);
    }
}
```

Note:

- As explained, because TreeSet stores its elements in a tree, they are automatically arranged in sorted order, as the output confirms.




Example 41.9: Sub set of a TreeSet collection

```
/* Because TreeSet implements the NavigableSet interface, you can use the methods defined by
NavigableSet to retrieve elements of a TreeSet. You can write many programs performing several
operations with the method declared in NavigableSet. In the following, the application of
subSet() is illustrated. The subset() method returns a sub set of a tree set that contains
the elements between elements, say a1 (inclusive) and a2 (exclusive).
*/
import java.util.*;

class SubSetTreeSetDemo {
    public static void main(String args[]) {
        // Create a tree set.
        TreeSet<String> ts = new TreeSet<String>();

        // Add elements to the tree set.
        ts.add("D");
        ts.add("E");
        ts.add("B");
        ts.add("A");
        ts.add("C");
        ts.add("F");
        ts.add("G");
        System.out.println(ts.subSet("D", "S"));
    }
}
```



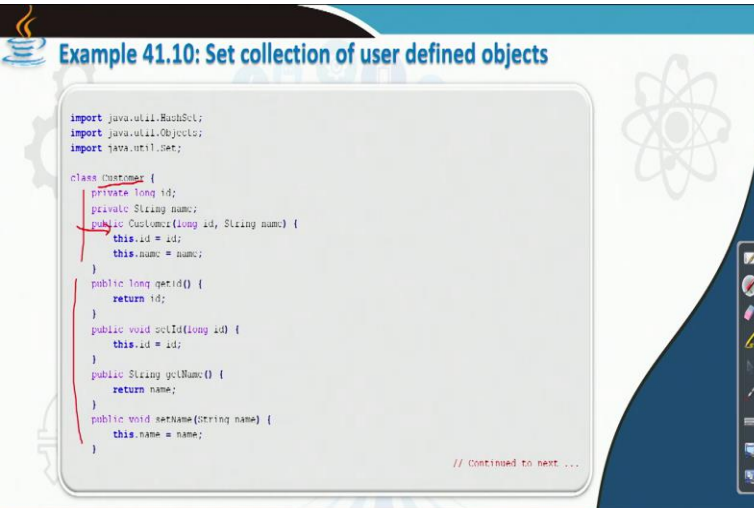
And operation I just do not want to discuss it much more because it is basically same, all those methods that can be called for this set also.

(Refer Slide Time: 32:24)



User Defined Set Collection

NPTEL Online Certification Courses
IIT Kharagpur



Example 41.10: Set collection of user defined objects

```
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

class Customer {
    private long id;
    private String name;
    public Customer(long id, String name) {
        this.id = id;
        this.name = name;
    }
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}

// Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

Now I just want to conclude this lecture by giving one example where a user defined set, user defined object can be stored in the form of a set. First of all we have to first define one user defined type. Here we define a customer, these are the different what is called the property of this customer, name is the field and this is a constructor we define to create the, or initialize this object.

And these are some getter and setter method to just modify the objects if you create and if you want to do so. And so these are called getter method and setter method to modify the different

field values of a given object like. So this is a very simple one object there are few more methods are defined also in this object for printing, printing objects.

(Refer Slide Time: 33:15)

The slide displays the following Java code:

```
// Continued on...  
  
// Two customers are equal if their ids are equal  
@Override  
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
    Customer customer = (Customer) o;  
    return id == customer.id;  
}  
  
@Override  
public int hashCode() {  
    return Objects.hash(id);  
}  
  
@Override  
public String toString() {  
    return "Customer{"  
        + "id=" + id + "  
        + ", name='" + name + "'" + "  
        + "}"  
    };  
}  
  
// Continued to next ...
```

The slide also features the NPTEL logo, the text 'NPTEL Online Certification Courses IIT Kharagpur', and a small video inset of a man in a white shirt.

And there is a method is called a equal method and hash code create method. Equal method means where the two objects are same or not that needs to be compared it because objects are not comparable that in that sense so that is why you have to rewrite a for method. Equal is basically over write method, equal method is defined in the object class. And there is another hash code also over write there, so object hash you can create it, or you can create hash code for this object which I have discussed while I was discussing about the map collection there, using this also you can define.

I have used one very naive one, you can plan your own hash code generation technique also, so that this hash code method can be overwritten here also. And here also, the all object elements, those are the fields of different type can be converting the string. Although string has two string or default method but for object it can be there so you can write it a little bit in a customized way that you can convert the object into a string form. So this is the one method which I have defined here so that object can be converted in the string form. And string is basically platform-independent one type that you can consider to manage or store any order.

(Refer Slide Time: 34:26)

The slide displays a code editor window with the following Java code:

```
// Continued on...  
public class HashSetUserDefinedObjectExample {  
    public static void main(String[] args) {  
        Set<Customer> customers = new HashSet<>();  
        customers.add(new Customer(101, "Rajeev"));  
        customers.add(new Customer(102, "Rashmi"));  
        customers.add(new Customer(103, "Chris"));  
    }  
}  
  
/*  
 * HashSet will use the equals() & hashCode() implementations  
 * of the Customer class to check for duplicates and ignore them  
 */  
customers.add(new Customer(101, "Rajeev"));  
  
System.out.println(customers);  
}
```

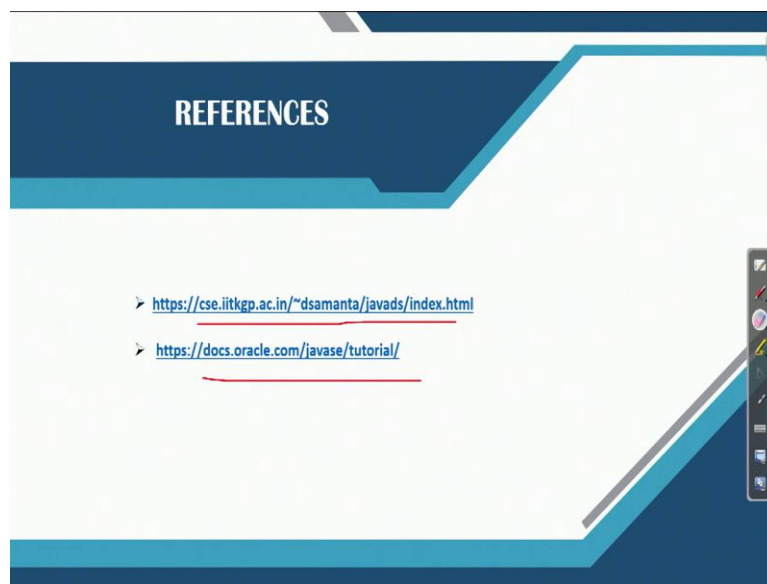
The code is presented in a light gray box with a white background. A red rectangle highlights the three lines where new Customer objects are added to the set. The slide also features a small video inset of a man in the bottom right corner and the NPTEL logo in the bottom left corner.

So these are the object that we have defined now we can create a few objects and then store that those objects in a set. Here we create a set, this set is a here set and few customer objects are created, these are the different elements and then finally we just simply add them. You see we are creating the object, these are the three objects are created and then we add them all these object into the set customers. Later on if we add another customers with the same, you will see that duplicate case will occur so it will not allow.

And finally you can print all the customer it will print the way you can change the string format it will do it and all the elements will be there. You can just store these objects in the array of objects and then process many other operations or that means set to copy and everything also can be done there. That mean, given a set you can convert into, we have learned about that given a list how a set can be stored, alternatively given a set how it can be stored into the form of a list also it can be there.

You can remember in, while I was discussing about map, there also we have given some example how a map can be converted in a set, set of key, set of objects and everything. Here also it can be converted some methods which already we have listed there you can take the help of this method and then you can do it. So these are the different ways the set can be managed we have discussed.

(Refer Slide Time: 35:54)



And these are the important links that you can consider so that you can study further and then complete your understanding rather. So this is basically the topic so far this module is concerned which I have just make it little bit deep actually discussion because set is important in any application development for any IT related application set is coming on the way. So that way it is very important and regarding particularly JDBMS or SQL programming or everything, there is a set thing already implemented.

So that JDBC application there, there are again many methods which have been given there, where you basically have many set utility automatically built in there. So, so those things are there but in addition to this here you can, you can manipulate set as a collection. So that topics is covered here, I hope you have understood it. Thank you. Thank you very much.