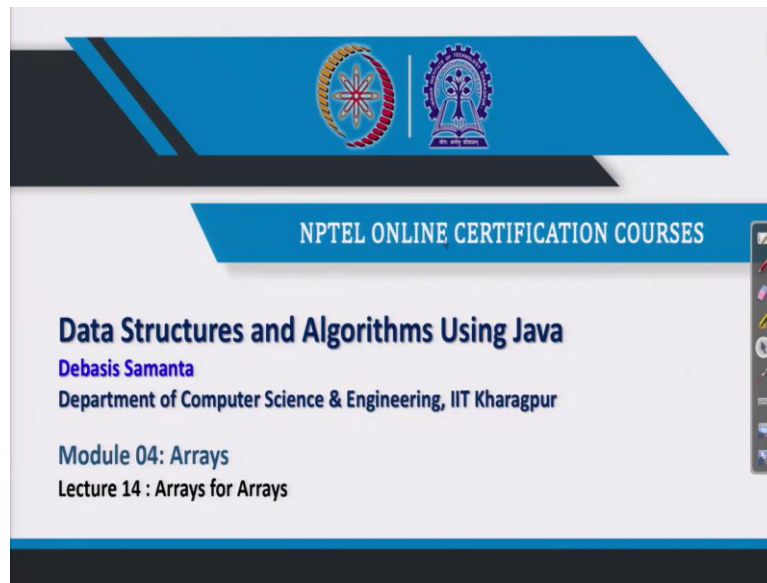


**Data Structures and Algorithms Using JAVA**  
**Professor Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Kharagpur**  
**Lecture No 14**  
**Arrays for Arrays**

Now, let us learn one another unique things in Java.

(Refer Slide Time: 0:29)

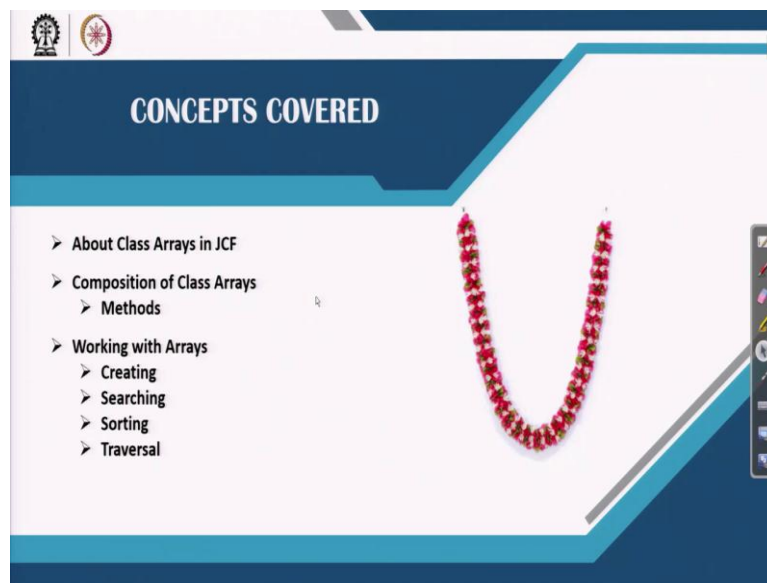


This is about arrays, you may be little confused about that we have enough programming efficiency about writing code to deal with array and we have done it one of our lecture classes where basically core programming. Then again Java supports then some time it is little bit confusing that which you want to use and when and whatever it is there.

So, obviously some programmer wants that everything can be from the ground level that means of their own code. So, they may be one reason that it can be better customized. On the other hand, the idea about the Java collection framework is to speed up in your software development effort. So, if you use it, definitely your productivity will increase without knowing about how those things are there and moreover the Java collection framework they claim that their implementation is most efficient or faster.

Sometimes there is a need to breach a gap or breach the two things together that means, something that is in your core, something that is there is for a JCF or Java Collection Framework to have the two blend together. No in order to provide these two blend, java provides one concept, the name is arrays, is a plural form.

(Refer Slide Time: 2:13)



The concept of arrays comes there so that you can handle many concept those are there so far core programming is concern as well as the java collection framework facilities are there. Now, in this lecture we will try to learn the concept of class Arrays which is supported by the Java developer, the class is again defined in java dot util package. We will see exactly what are the different methods are there.

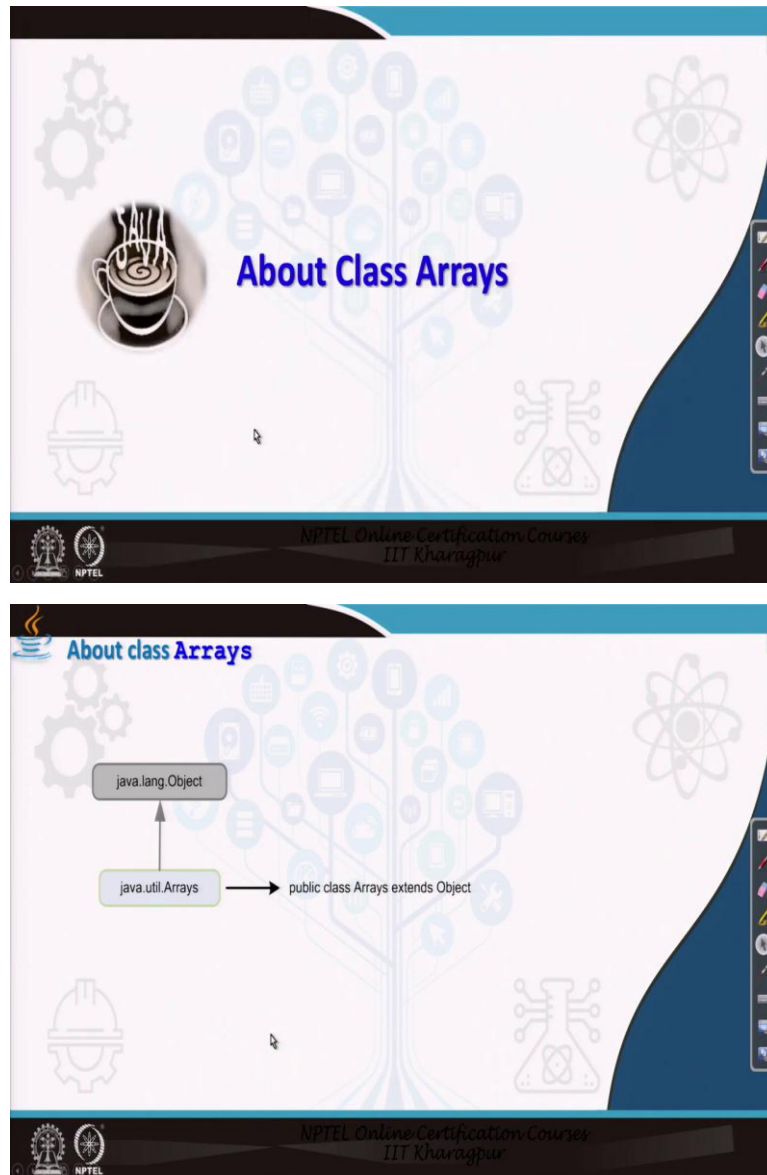
Now again you can note that like ArrayList class, there are many methods are defined there in class Arrays. But unlike there is no any constructor defined there in Arrays. That mean you can call the methods those are there in Arrays dot class for the class itself. For example, a method in which is defined in arrays so, if you want to invoke this method, then arrays dot in, that is the procedure.

Now, we will see exactly the utility of this arrays class and the different operations how you can perform using this arrays rather we can say how we can create arrays from the existing whatever the array that you can maintain or you can create using core programming or you can create using the JCF array list. Mainly arrays is very much acceptable or any much popular among programmer because of its search searching and sorting methods.

Now there are many sorting and searching algorithms have been implemented in this class arrays for any type of, of data, any type of elements which is the array can contain for Boolean, for integer, for float, for double, for string, whatever it is there for regular data type of codes not for user defined data type. If user defined data type you if you want to use, then my suggestion is that you should think for code programming that is more better instead of

collection framework there. Anyway we will discuss about the different operations with this arrays class; sorting, searching, traversal, creating and all these things in this lecture.

(Refer Slide Time: 4:40)



Now let us learn about class arrays as I told you it is defined in java dot util package, it is just defined in this there is no dependencies that it derives or it implements like it just simply standalone class one whose super class is object class.

(Refer Slide Time: 4:59)

**About class Arrays**

- This class name is in plural form “**Arrays**” because it can handle java arrays of different primitive types like *byte, char, short, int, long, float, double, and universal type Object*.
- This class **Arrays** has been designed to bridge the gap between collections and Java arrays. More specifically, there are often times when loops are used to do some tasks on an array like fill an array with a particular value, sort an array, search in an array, etc.
- This class provides several static methods to dynamically create and access Java arrays. In fact, it consists of **only static methods** and **the methods of Object class**.
- The methods of this class can be used by the class name itself. That is, no object of the class **Arrays** need to be created to access a method in it.

NPTEL Online Certification Courses  
IIT Kharagpur

Now it has only static methods it does not have any other non-static methods and the methods are of object class actually.

(Refer Slide Time: 5:11)

**About class Arrays**

- The methods of this class can be used by the class name itself. That is, no object of the class **Arrays** need to be created to access a method in it.
- The following is the syntax to access a method in **Arrays**.

```
Arrays.<methodName ( . . . )>;
```

Here, <T> denotes a type parameter. This type parameter includes *boolean, byte, char, short, int, long, float, double, and Object*.

NPTEL Online Certification Courses  
IIT Kharagpur

Now let us see how and arrays method can be accessed I told you if the, this method is defined in this arrays class the static method like unlike this one for this class you can declare it this way and here T, basically it is a, this basically we can say it is T is there this method is basically generic method all this T is applicable to the primitive data type that is the one important thing that you should note it here. Now let us see some illustration of the different methods of the class arrays.

(Refer Slide Time: 5:55)

The top screenshot shows the title "Methods of Class Arrays" in blue text. Below the title is a large tree diagram where the branches are composed of various icons representing different fields of study or technology. A small inset video of the presenter is visible in the bottom right corner of the slide.

The bottom screenshot shows a table with the following content:

Method	Description
<code>&lt;T&gt;ListAsList(&lt;T&gt; array)</code>	It returns a List that is backed by an array of type T. Both the list and array refer to the same location.
<code>int binarySearch(&lt;T&gt; array[], &lt;T&gt; value)</code>	It searches for the specified element in the array with the help of Binary Search algorithm. It returns the index of an array location, if found else NULL. Array of type boolean is not applicable to such a method.
<code>&lt;T&gt;[] copyOf(&lt;T&gt;[] source, int len)</code>	It returns a copy of an array source and len is the length of the copy. If the copy is longer than source, then the copy is padded with zeros (for numeric arrays), nulls (for object arrays), or false (for boolean arrays). If the copy is shorter than source, then the copy is truncated.

The first method that you can see ListAsList that means if any existing array which you can create using the primitive data type like int or float or char whatever it is there and then you can convert this into a collection. So, it is basically conversion of the primitive arrays using the collection arrays. The binary search is an algorithm very popular for searching an element in a collection, so this method you can do it here is the collection you have to send and then the element that needs to be searched.

Now here again this T is any type int, char, long, short, for not for that object it is basically regular data type. The copy of just like this one also it basically a sub list can be created from the given list and then it can be stored as a collection, so this are the few important methods

those are there, so for the array collection is concern in addition there are few more methods also.

(Refer Slide Time: 7:10)

Method	Description
<code>&lt;T&gt;[] copyOfRange(&lt;T&gt;[] source, int start, int end)</code>	It returns a copy of a range within an array in from start to end-1.
<code>boolean equals(&lt;T&gt; array1[], &lt;T&gt; array2 [])</code>	It returns true if two arrays are equivalent, otherwise, it returns false. Here, array1 and array2 should be comparable for equality.
<code>boolean deepEquals(Object[] a, Object[] b)</code>	The method can be used to determine if two arrays, which might contain nested arrays, are equal.
<code>void fill(&lt;T&gt; array[], &lt;T&gt; value)</code>	It fills an array with a specified value.
<code>void fill(&lt;T&gt; array[], int start, int end, &lt;T&gt; value)</code>	It fills an array with a specified value to a subset of an array from position start to end-1.
<code>void sort(byte array[])</code>	It sorts an array so that it is arranged in ascending order.
<code>void sort(&lt;T&gt; array[], int start, int end)</code>	It sorts a subarray in ascending order between the position start and end-1.
<code>void parallelSort(byte array[])</code>	It sorts, into ascending order, portions of an array in parallel and then merges the results.
<code>void parallelSort(&lt;T&gt; array[], int start, int end)</code>	It sorts a subarray in ascending order between the position start and end-1 in parallel.
<code>&lt;T&gt;Spliterator spliterator(T array[])</code>	It returns a spliterator to an entire array. Here, array is the array that the spliterator will cycle through.

And here are few more methods that you can check quickly copyOfRange it is basically similar to the copy of a sub list equals it basically check whether two arrays are equals or not. Equals in the sense that they contain same set of elements. Now deepEquals actually within an array, an array so, nested array if it is there it can check if an array contains nested array or not. Then fill array, it is basically if we want to copy or make a fill some elements in an array with a value you can do it so, filling. So, suppose you want to initialize all the elements by 1, so it is basically you can call this fill, this array which needs to be initialize by all a default value, say 1. It is like this

And also you can do partially fill from a start to end, there also sub list can be filled. Now these are the sort method by which it can be applied to any type of data in order to sort it and this second method is basically sorting a sub list, not the entire list needs to be sorted, only a part. Now, it also has a facility to accomplish the parallel execution of sorting. There are many parallel sorting algorithms are there which can be applied to the whole array or is part array. And this is the one method is basically for traversing an array using split iterator, split iterator.



(Refer Slide Time: 8:51)

Methods of class Arrays

Method	Description
<code>&lt;T&gt; Spliterator spliterator(T array[], int start, int end)</code>	It enables you to specify a range to iterate within the array.
<code>&lt;T&gt; Stream stream(T array[])</code>	It supports a Stream interface. Here, array is the array to which the stream will refer.
<code>&lt;T&gt; Stream stream(&lt;T&gt; array[], int start, int end)</code>	It supports a Stream interface within a specific range from the position start to end-1.
<code>void setAll(double array[], IntToDoubleFunction&lt;? extends T&gt; genVal)</code>	It assigns values to all of the elements of the array. Other overloading methods exist to deal with arrays of int, long, and generic.
<code>void parallelSetAll(double array[], IntToDoubleFunction&lt;? extends T&gt; genVal)</code>	It does the same thing as <code>setAll(...)</code> but in parallel.
<code>void parallelPrefix(double array[], DoubleBinaryOperator func)</code>	It modifies an array so that each element contains the cumulative result of an operation applied to all previous elements. For example, if the operation is addition, then on return, the array elements will contain the values associated with the running sum of the original values. Many other versions are provided, including those that operate on types int, long, and generic, and those that let you specify a range within the array on which to operate.

NPTEL Online Certification Courses  
IIT Kharagpur

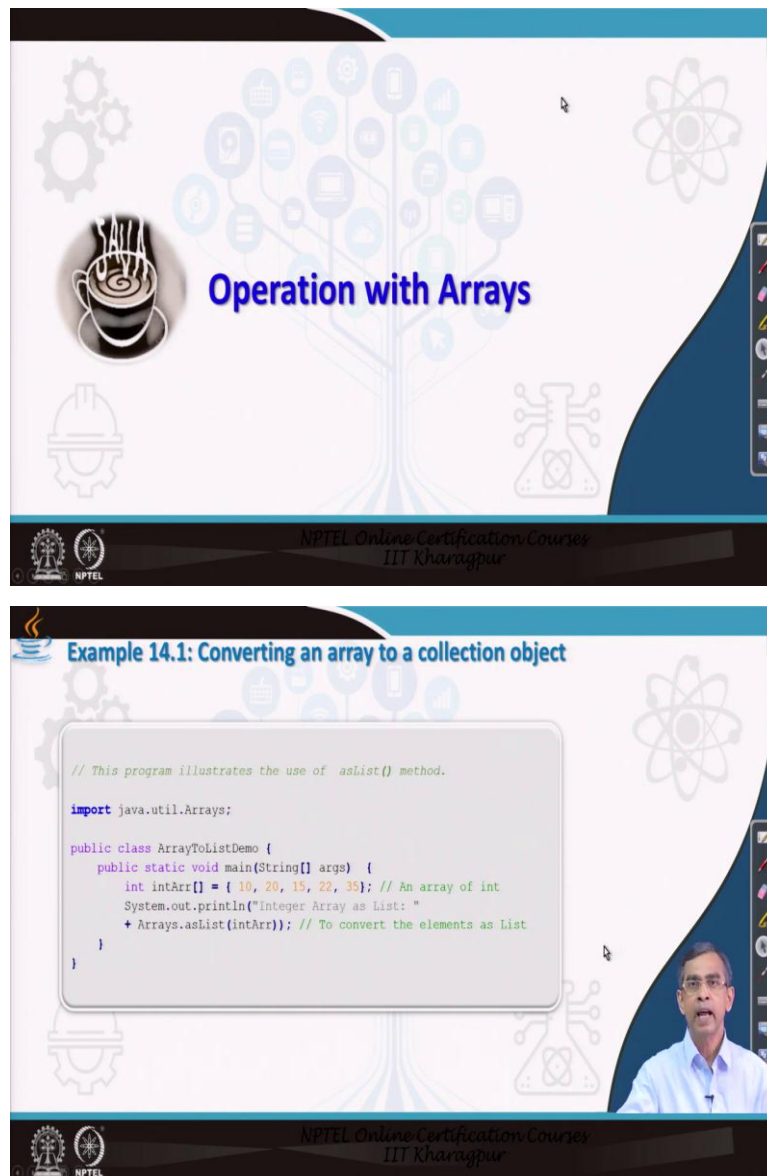
Methods of class Arrays

Method	Description
<code>&lt;T&gt; Spliterator spliterator(T array[], int start, int end)</code>	It enables you to specify a range to iterate within the array.
<code>&lt;T&gt; Stream stream(T array[])</code>	It supports a Stream interface. Here, array is the array to which
<p><b>Note:</b> Arrays also provides <code>toString()</code> and <code>hashCode()</code> for the various types of arrays. In addition, <code>deepToString()</code> and <code>deepHashCode()</code> are provided, which operate effectively on arrays that contain nested arrays.</p>	
<code>DoubleBinaryOperator func)</code>	cumulative result of an operation applied to all previous elements. For example, if the operation is addition, then on return, the array elements will contain the values associated with the running sum of the original values. Many other versions are provided, including those that operate on types int, long, and generic, and those that let you specify a range within the array on which to operate.

NPTEL Online Certification Courses  
IIT Kharagpur

So, there are many methods like this one and also it allow you to create a string or convert an array into a string, all these kind of are there. So, few methods in this case also included here. Now, there are many methods also very much useful where we can apply it. We can create an array elements to a string elements also that conversion is possible and also we can create hash code of each elements. Regarding hash code, we will learn it in details but just for timing, you assume that hash code is a method which basically convert a particular value into another form. So, those things are basically quite common so far the arrays manipulation is concern.

(Refer Slide Time: 9:44)



The top slide, titled "Operation with Arrays", features a background with a tree-like structure of icons representing various technologies and concepts. The bottom slide, titled "Example 14.1: Converting an array to a collection object", displays the following Java code:

```
// This program illustrates the use of asList() method.  
  
import java.util.Arrays;  
  
public class ArrayToListDemo {  
    public static void main(String[] args) {  
        int intArr[] = { 10, 20, 15, 22, 35}; // An array of int  
        System.out.println("Integer Array as List: "  
        + Arrays.asList(intArr)); // To convert the elements as List  
    }  
}
```

Now, let us have a quick view of different operations with arrays and try to understand its supports to a programmer. Now, this is a very simple example. Now, in order to have access the facility that is defined in arrays, you should import this package that mean, this class, arrays class which is defined again java dot util. Otherwise, this your program may not run if you invoke many methods are there.

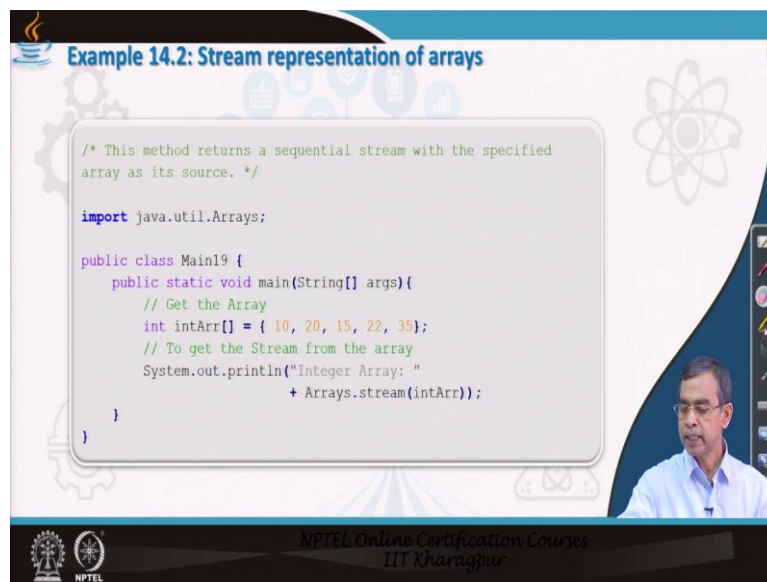
Now, this is a simple program to give an example of how the arrays can be created. Now let us see what we do it here, we create according to our own way an array of integer. So, we create an array of integer, declare it as a type and initialize this array element few numbers. It is declared here and here you see this is the one statement that you can check it. So, basically we pass an argument as the name of the array, it is array.



That means, we passed the whole array into this method asList. What the asList will do? asList will convert this array into a collection. So, this collection is an integer collection. So, it is basically it converts an integer array into a collection. This collection is of type list. Now, the list is a collection after which the ArrayList can be obtained or you can see or view this list as a ArrayList or any other type of list.

Later on we will see how this can be viewed as a linked list or set or tree whatever it is there. Anyway so this example explain, how from a primitive array like this one, how from a primitive array like this one, a list can be created using the method which is defined there. So, arrays dot asList. Now, we should call although, asList is a static method, but it is better to reserve the scope by calling or specifically mentioning this method in your program. That is why arrays dot asList, it is a customer your convention to call all the methods this way.

(Refer Slide Time: 12:15)



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

```
/* This method returns a sequential stream with the specified
array as its source. */

import java.util.Arrays;

public class Main19 {
    public static void main(String[] args){
        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35};
        // To get the Stream from the array
        System.out.println("Integer Array: "
            + Arrays.stream(intArr));
    }
}
```

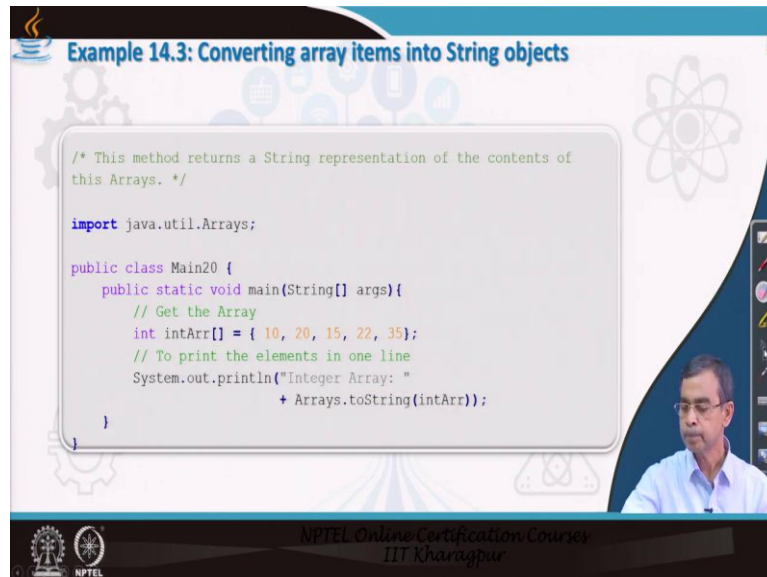
The slide also features a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

Now, so this example explain how using the arrays method a simple array can be converted into this one. We have given an example for integer likewise, float, string, everything you can call it. Now, this is another example as we can see, we can convert using the stream method to an integer array into a stream class. So, stream is an important concept is basically converting the byte values of the arrays and byte is many program it require for a faster execution or data transmission.

Regarding the stream and everything we will discuss about the handling different stream, io stream, byte stream, character stream all these things. So, for timing we can keep up to these much only. So, it is basically using the stream method in arrays class, we can convert any

type of data into a stream class. Now, there is another also, we can convert using the method they are toString method which basically convert into a string type.

(Refer Slide Time: 13:18)



The slide displays a Java code snippet for converting an integer array to a string. The code includes a comment explaining the purpose of the Arrays.toString() method, an import statement for java.util.Arrays, and a main method that creates an integer array [10, 20, 15, 22, 35] and prints it using Arrays.toString(). The slide also features a small video inset of a presenter and logos for NPTEL and IIT Kharagpur.

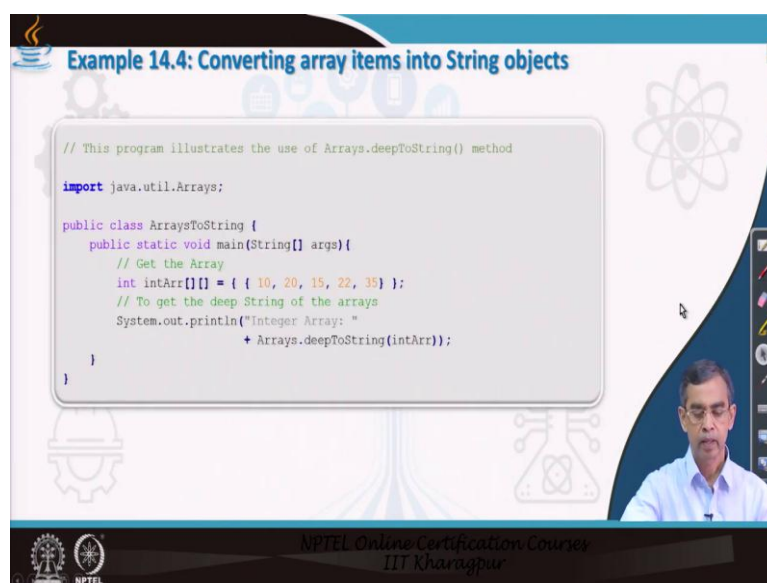
```
/* This method returns a String representation of the contents of
this Arrays. */

import java.util.Arrays;

public class Main20 {
    public static void main(String[] args){
        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35};
        // To print the elements in one line
        System.out.println("Integer Array: "
            + Arrays.toString(intArr));
    }
}
```

String is a another type, here you can see the same program we create an integer array and then convert this integer array into string collection. So, basically all these things will be viewed as a string and from the string we can convert into integer or any other desirable format if you required and so this is basically the toString method which is there in the class, it is basically object class defined there and we can call for it and then you can see how the elements in a regular array can be converted into string type.

(Refer Slide Time: 14:05)



The slide displays a Java code snippet for converting a 2D integer array to a string using the Arrays.deepToString() method. The code includes a comment explaining the use of the method, an import statement for java.util.Arrays, and a main method that creates a 2D integer array and prints it using Arrays.deepToString(). The slide also features a small video inset of a presenter and logos for NPTEL and IIT Kharagpur.

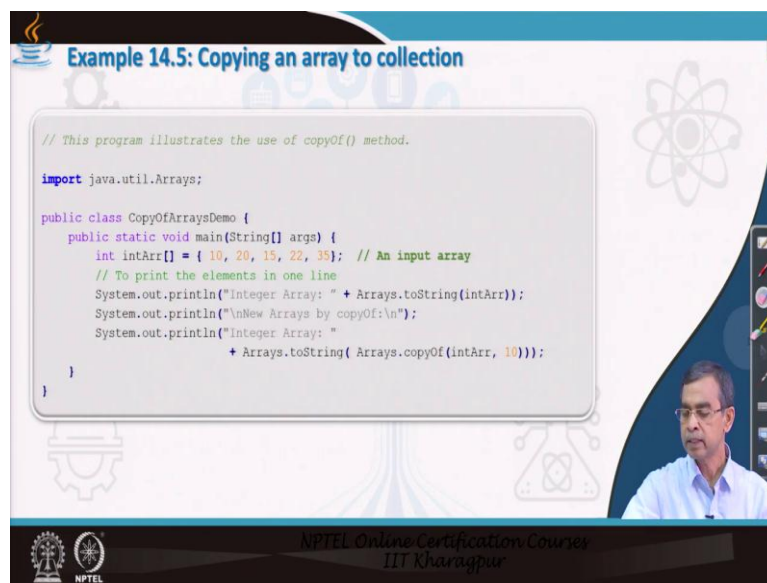
```
// This program illustrates the use of Arrays.deepToString() method

import java.util.Arrays;

public class ArraysToString {
    public static void main(String[] args){
        // Get the Array
        int intArr[][] = { { 10, 20, 15, 22, 35 } };
        // To get the deep String of the arrays
        System.out.println("Integer Array: "
            + Arrays.deepToString(intArr));
    }
}
```

Now, deepToString is basically similar to the string only but if it is a nested array within array and array, you can see it is basically within array and array, but there is also another sub list of array we can create within this one so, deepString means it is basically within the string, a string can be converted and then another sub list can be string converted and everything. So, are deepString method, it is similar to toString but it is a nested array actually that you can apply. So, we have learned about few methods by which the regular array can be converted into a arrays, different types actually, different collections of different types.

(Refer Slide Time: 14:38)



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

```
// This program illustrates the use of copyOf() method.

import java.util.Arrays;

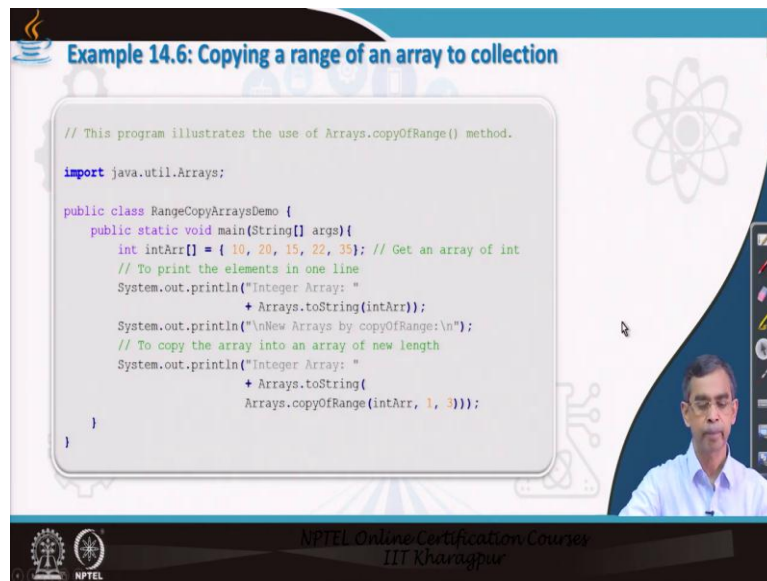
public class CopyOfArraysDemo {
    public static void main(String[] args) {
        int intArr[] = { 10, 20, 15, 22, 35}; // An input array
        // To print the elements in one line
        System.out.println("Integer Array: " + Arrays.toString(intArr));
        System.out.println("\nNew Arrays by copyOf:\n");
        System.out.println("Integer Array: "
            + Arrays.toString( Arrays.copyOf(intArr, 10)));
    }
}
```

The slide also features a video inset of a man in a light blue shirt, the NPTEL logo, and the text "NPTEL Online Certification Course IIT Kharagpur" at the bottom.

Now here is an another example. This example says how an existing array collection can be made a copy to another collection like. Here, in this example as we see, this is an integer array, we declare and loaded with these elements. These basically convert into toString. Now, here I can make a copy, copy of these arrays and it basically give the length, that mean, how much size. Now, here although 5, but remaining will be assigned with 0 values, so the total array will be all these array plus 0, 0, 0 and all these things.

So, it basically copy of this elements and a copy will be created and this copy will pass to this toString method that means, the entire copy of this class will be there. So, what exactly the net result is that, this element will remain intact, a copy of it will be created with 0 elements at the tail end and then the whole arrays will be converted toString will be printed. So, this is a one example by which the copy method can be utilised in many occasion. So, this is a example of copyOf method.

(Refer Slide Time: 16:00)



**Example 14.6: Copying a range of an array to collection**

```
// This program illustrates the use of Arrays.copyOfRange() method.

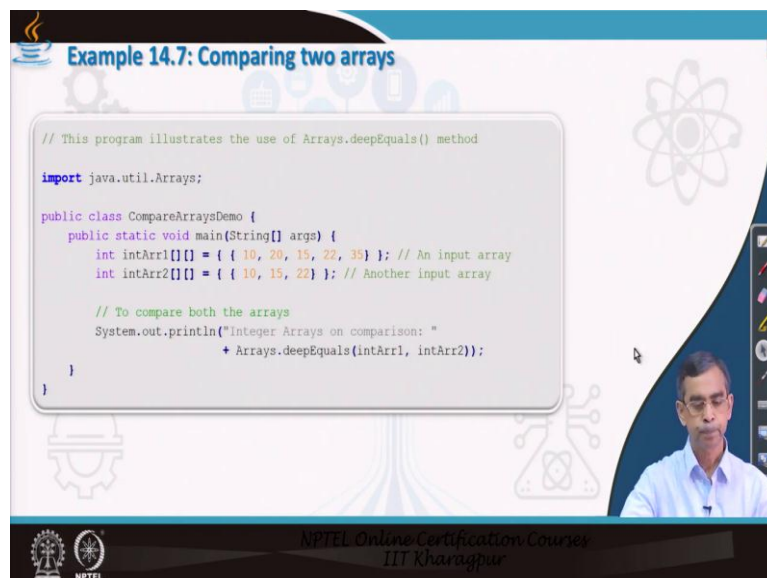
import java.util.Arrays;

public class RangeCopyArraysDemo {
    public static void main(String[] args){
        int intArr[] = { 10, 20, 15, 22, 35}; // Get an array of int
        // To print the elements in one line
        System.out.println("Integer Array: "
            + Arrays.toString(intArr));
        System.out.println("\nNew Arrays by copyOfRange:\n");
        // To copy the array into an array of new length
        System.out.println("Integer Array: "
            + Arrays.toString(
                Arrays.copyOfRange(intArr, 1, 3)));
    }
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

Now, copyOf method can be applied to the whole arrays or it can be a part array. This example indicates how the same copyOf method can be applied to a part. That mean, it basically copy not the whole but is a sub list of the array. In this case, from 1 to 3 both inclusive a copy can be applied and then it can be converted to string. The same thing only the part method actually, part of sub list of a regular array can be copied and a copied can be done using arrays method.

(Refer Slide Time: 16:37)



**Example 14.7: Comparing two arrays**

```
// This program illustrates the use of Arrays.deepEquals() method

import java.util.Arrays;

public class CompareArraysDemo {
    public static void main(String[] args) {
        int intArr1[][] = { { 10, 20, 15, 22, 35} }; // An input array
        int intArr2[][] = { { 10, 15, 22} }; // Another input array

        // To compare both the arrays
        System.out.println("Integer Arrays on comparison: "
            + Arrays.deepEquals(intArr1, intArr2));
    }
}
```

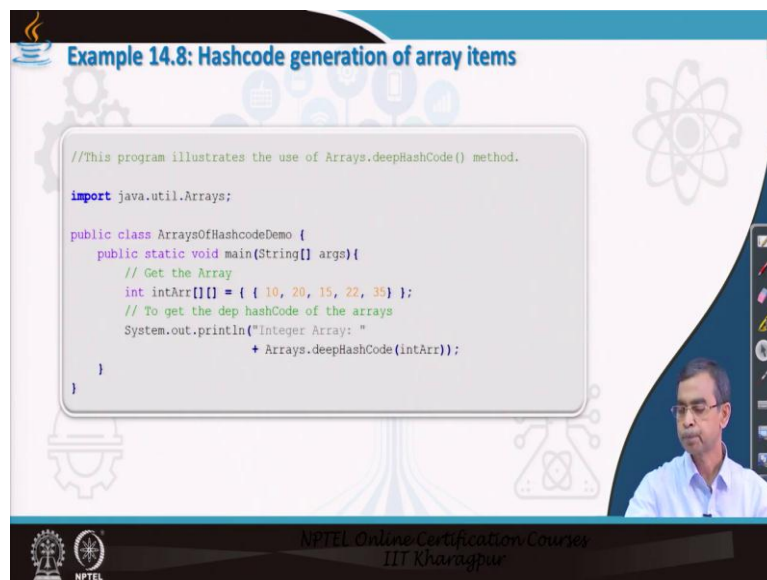
NPTEL Online Certification Courses  
IIT Kharagpur

Now, there is a method in arrays dot class to compare two arrays. Now, whatever be the order of elements, if they contain same elements in both the arrays, if duplicacies are there they will consider only one occurrence of it and then compare. If they found the two elements, two

arrays contain the same sets of elements, then it returns true otherwise, it returns false. So, here this is an example as you can see the two arrays we have declared and it is the two-dimensional array we declare and using the string and whatever it is there, it is basically deepEquals. Like deepEquals only equals method can be applied.

If it is equals methods, then it is better to apply to one-dimensional array. The deepEquals basically or row-wise it will check and then for every row, then all columns will be checked and then it will find if the two arrays contain the same. As we see in this case, two arrays are not of equal elements stored in it, so this definitely returns false and you can run this program and you can check that how it can decide. You can take some other input here and then run it again, then you can see how it works. So, I should advise you to do many experiments over this simple basic program and see the different results that you can, you can see from the execution of the program.

(Refer Slide Time: 18:10)



The slide displays the following Java code:

```
//This program illustrates the use of Arrays.deepHashCode() method.  
  
import java.util.Arrays;  
  
public class ArraysOfHashCodeDemo {  
    public static void main(String[] args){  
        // Get the Array  
        int intArr[][] = { { 10, 20, 15, 22, 35 } };  
        // To get the dep hashCode of the arrays  
        System.out.println("Integer Array: "  
            + Arrays.deepHashCode(intArr));  
    }  
}
```

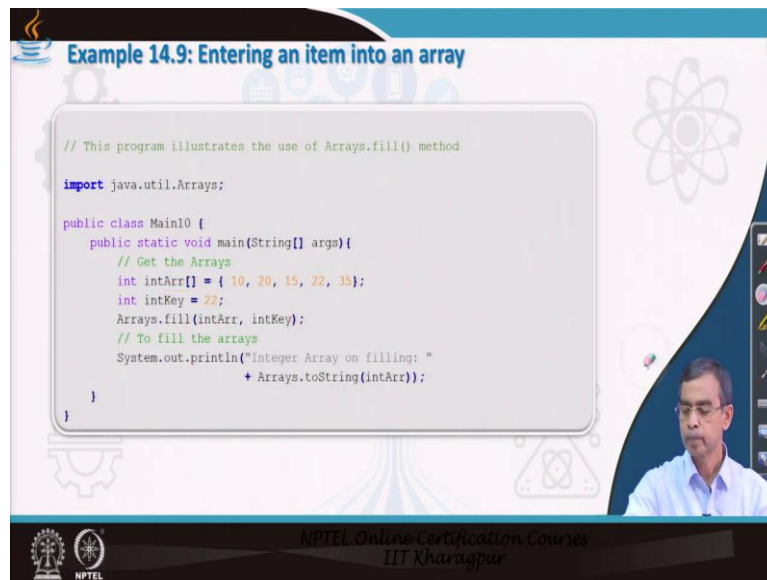
The slide also features a video inset of a speaker in the bottom right corner and the NPTEL logo in the bottom left corner.

Now, Hashcode generation I told you I will discuss in details about Hashcode concept but hashcode is basically is a transformation from one type of value to a particular elements. So, this program, given an integer array, is a two-dimensional array, no issue, now deepHashCode if it is two-dimensional array you should call otherwise, if it is a one-dimensional array, you write hashCode only.

So, this in this case, h and then this one. So, this is for the simple one-dimensional array. What (()) (18:44) Hashcode? This means that for every element in, it will give a hash value say, h1 unique. So, for this also another, for this also another, and so on so. So, basically if we call this method for this arrays, it will return hash values of all the elements which are

there in this arrays. So, this basically is a Hashcode conversion from a given array. It has many application. We will discuss about mapping of the different records into tables and everything. So, there we will see how adjustably the hashcode concept is applicable there.

(Refer Slide Time: 19:25)



**Example 14.9: Entering an item into an array**

```
// This program illustrates the use of Arrays.fill() method

import java.util.Arrays;

public class Main10 {
    public static void main(String[] args){
        // Get the Arrays
        int intArr[] = { 10, 20, 15, 22, 35};
        int intKey = 22;
        Arrays.fill(intArr, intKey);
        // To fill the arrays
        System.out.println("Integer Array on filling: "
            + Arrays.toString(intArr));
    }
}
```

Now, inserting an element into an array elements also it is possible. Here is an example. This is an existing arrays of elements, integer numbers and say, 22 is the one value we want to insert. Now, here insertion by means of the method called the fill method. So, it is int array and intKey the value that needs to be stored into this array. So, basically it will it will add this element at the end and duplicacy is allowed.

So, 22 is there so after 35, I will basically add 22. It is there. So, this is a fill array method that can be there and one more one more important thing I should advice you to note it, if you want to print a collection according to the arrays or any elements, so better that it can be converted, it it is can be converted into two string and then print it.

For example, here after adding this 22, entering this value 22 into this array, we can print the entire array using print in statement, you could do int array, this is there also but better way that it can be converted to the string and then that string, resultant string can be pass can argument to this println method to print it. That also you can do.



(Refer Slide Time: 21:04)

The slide features a central graphic of a tree where the branches are composed of various icons representing different fields of study or technology. The title "Searching with Arrays" is prominently displayed in blue text. A small inset video shows a male speaker in a light blue shirt. The footer includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

The slide is titled "Example 14.10: Binary search on an array". It displays a code editor window with the following Java code:

```
// This program illustrates the use of Binary Search method.  
  
import java.util.Arrays;  
  
public class BinarySerachArraysDemo {  
    public static void main(String[] args){  
        // Get the Array  
        int intArr[] = { 10, 20, 15, 22, 35};  
        Arrays.sort(intArr);  
        int intKey = 22;  
        System.out.println(intKey  
            + " found at index = "  
            + Arrays.binarySearch(intArr, intKey));  
    }  
}
```

A small inset video shows the same male speaker. The footer includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

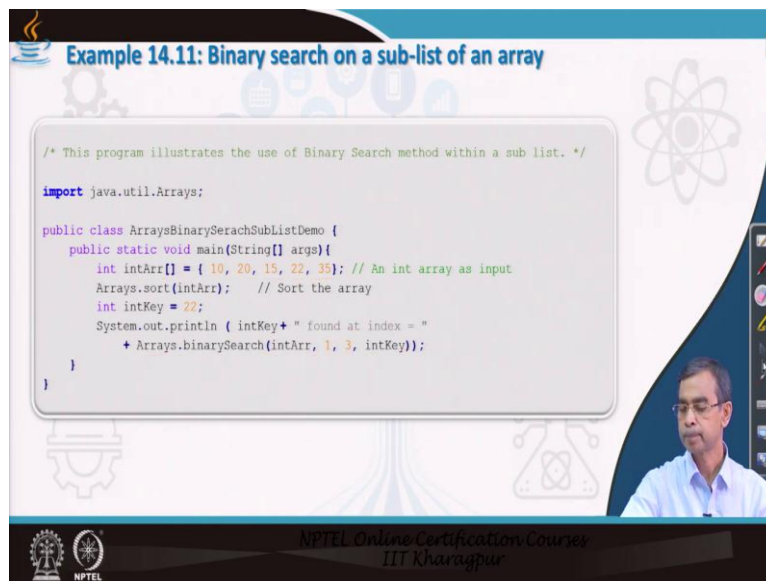
Now, searching and sorting are the two important operations which we can do with the arrays and here is some example. The binary search is the algorithm which is known as the best algorithm to search for a any item in a list and there is a method defined there and this method is defined in this arrays class for any type of primitive data like int, boolean, long, short, char, float, double, etc.

Now, here this is an example as we can see. We first create an array of integers. And then binary search algorithm that whatever the elements it is there, it should be first sorted. Now, this is not in sorted order so we call a sort method which is there in the arrays class. So, we call first sort method and then sort the array and ones it is sorted there, suppose you want to search 22, then we can call this method binary search, search to be performed on this array

which is sorted now, after this method the elements will be ordered in sorted order in ascending order.

And then this is the key value to be sorted and then binary search method can be called for this elements and then it will return in which location this element is present. So, this is the one example of the sort, searching method that is there in this arrays. Now, these are binary search method also can be applied to the search over a sub list.

(Refer Slide Time: 22:54)



The slide displays a Java code snippet for a binary search on a sub-list. The code is as follows:

```
/* This program illustrates the use of Binary Search method within a sub list. */
import java.util.Arrays;

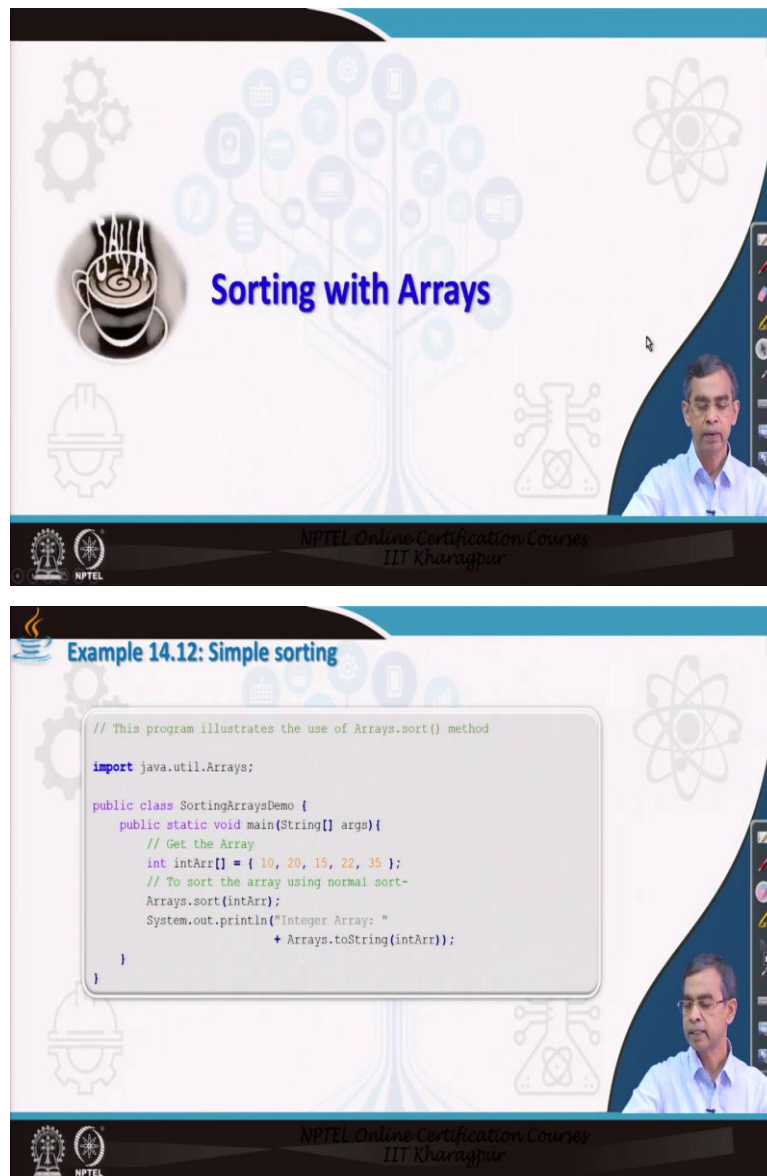
public class ArraysBinarySerachSubListDemo {
    public static void main(String[] args){
        int intArr[] = { 10, 20, 15, 22, 35}; // An int array as input
        Arrays.sort(intArr); // Sort the array
        int intKey = 22;
        System.out.println ( intKey+ " found at index = "
            + Arrays.binarySearch(intArr, 1, 3, intKey));
    }
}
```

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

Now, here is an example as we can see, same example but we have little modify. The modification you can see in this call of binary search there. In the previous example, we call this for the entire array, but in this example we call this binary search method to search the item within this element 1 and 3 both inclusive. So, it basically searching over a sub list if a particular element in this case, intKey is present in the list or not.

So, this can be applied to this one. Now, again I should suggest you to test all these program, not for on integer type, you can declare is a double or string and then initialize with some element and then search it. We will see you do not have to bother about what type of data that it has to process and how it has to be processed. It will do for you including byte type, including any other type those are the standard built in data type but not for user defined data type. You cannot do all these operations for user defined data type. So, this is a searching algorithm.

(Refer Slide Time: 24:14)



The top slide, titled "Sorting with Arrays", features a central tree diagram with various icons representing different fields of study. The bottom slide, titled "Example 14.12: Simple sorting", displays a Java code snippet for sorting an array.

```
// This program illustrates the use of Arrays.sort() method
import java.util.Arrays;

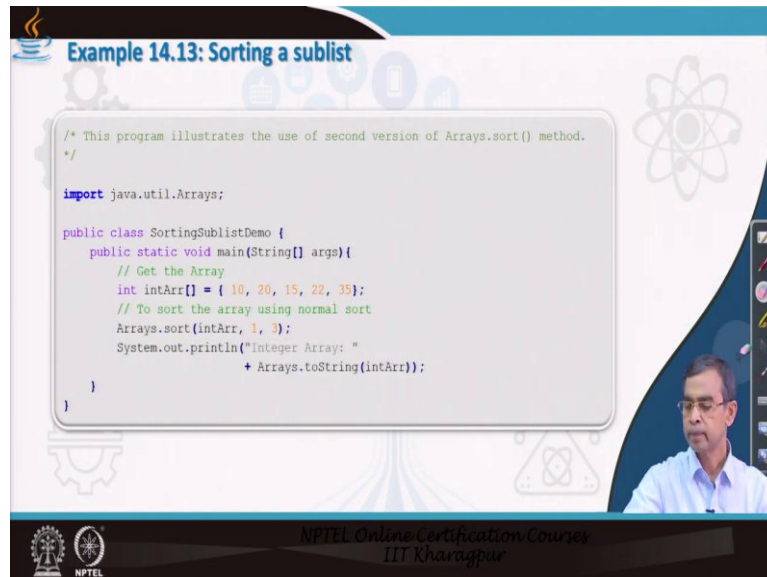
public class SortingArraysDemo {
    public static void main(String[] args){
        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };
        // To sort the array using normal sort-
        Arrays.sort(intArr);
        System.out.println("Integer Array: "
            + Arrays.toString(intArr));
    }
}
```

Now, let us come to the sorting with arrays. There are many sort methods including the parallel sorting and others which are the speciality in this class. Now, let us see the sort method that can be applied. We have little bit glimpse of, idea about sort method, we have called ones, we see the binary search here exactly the same thing, the sort and then the array that has to be sorted. It can be any type as I told you. It can be any type, either long, float, short or string, whatever it is there.

Now, if you call this method, it will sort that mean, all are of the element or this element will be replaced in a sorted fashion of this element. So, this if it is the input, it will give 10, 15, 20, 22, 35 like this one and this is basically the regular way by which it can be printed. So, this

element can be printed. Now, again you can try this sorting method calling over it the different type of array elements and see the result.

(Refer Slide Time: 25:19)



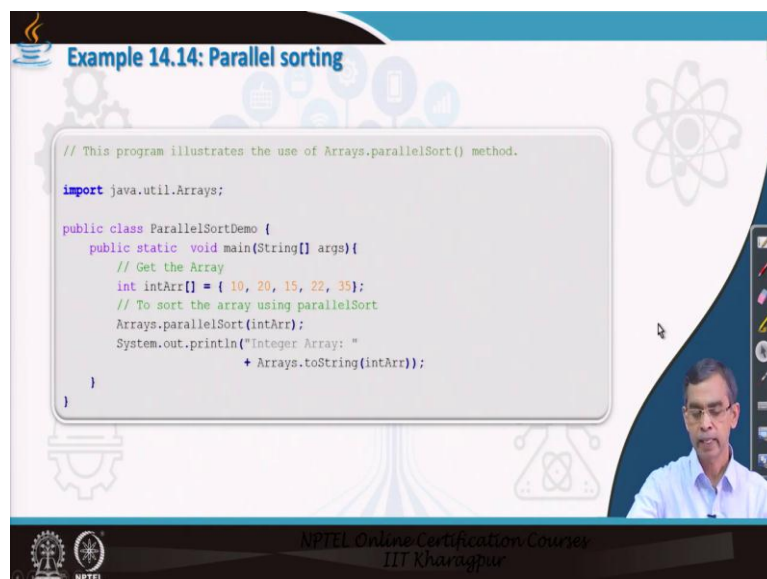
The slide displays a Java code snippet for sorting a subarray. The code uses the `Arrays.sort()` method with two arguments: the array and the start index (1). The array contains the values {10, 20, 15, 22, 35}. The output is expected to be "Integer Array: [10, 20, 15, 22, 35]".

```
/* This program illustrates the use of second version of Arrays.sort() method.
 */
import java.util.Arrays;

public class SortingSublistDemo {
    public static void main(String[] args){
        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35};
        // To sort the array using normal sort
        Arrays.sort(intArr, 1, 3);
        System.out.println("Integer Array: "
            + Arrays.toString(intArr));
    }
}
```

Now, these are sort, sort can be performed again on the sub list also like binary search array. Here is the same composition, you can see here the sorting not for all only so, 0, 1. So, up to this it will basically sort and if we apply it and you will see this is basically 10, 35 and it will sort this order. The rest of the part will not be sorted here, for example, if it is 35, if it is 10, you will see up to this it will not be sorted but only this part will be sorted. You can check the program with different input to it and then run this program and see the working of this methods in how it is, it gives output.

(Refer Slide Time: 26:03)



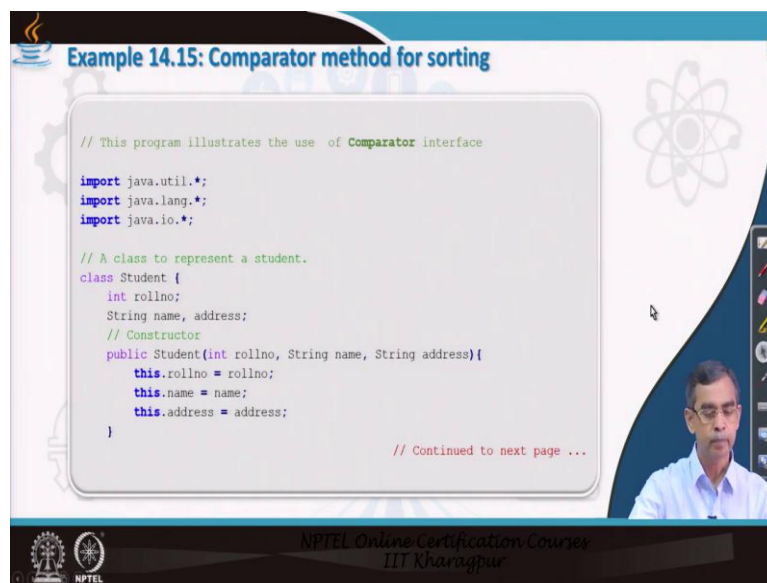
The slide displays a Java code snippet for parallel sorting. The code uses the `Arrays.parallelSort()` method. The array contains the values {10, 20, 15, 22, 35}. The output is expected to be "Integer Array: [10, 20, 15, 22, 35]".

```
// This program illustrates the use of Arrays.parallelSort() method.
import java.util.Arrays;

public class ParallelSortDemo {
    public static void main(String[] args){
        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35};
        // To sort the array using parallelSort
        Arrays.parallelSort(intArr);
        System.out.println("Integer Array: "
            + Arrays.toString(intArr));
    }
}
```

And parallel sort is just same as sort but this internal mechanism how the sorting can be done. The best idea to see the parallel sort only for few elements it is not so good, but if you can exercise this program with very large set of data and then if you can calculate the time of sorting, then you can understand parallel sort gives the fastest output in a smallest time actually. So, because of it use the multiple thread to sort it and an efficient way of doing what. So, if you want to sort very large set of data and time is an important factor, then you can call the arrays parallel sort method to test it.

(Refer Slide Time: 26:50)



```
// This program illustrates the use of Comparator interface

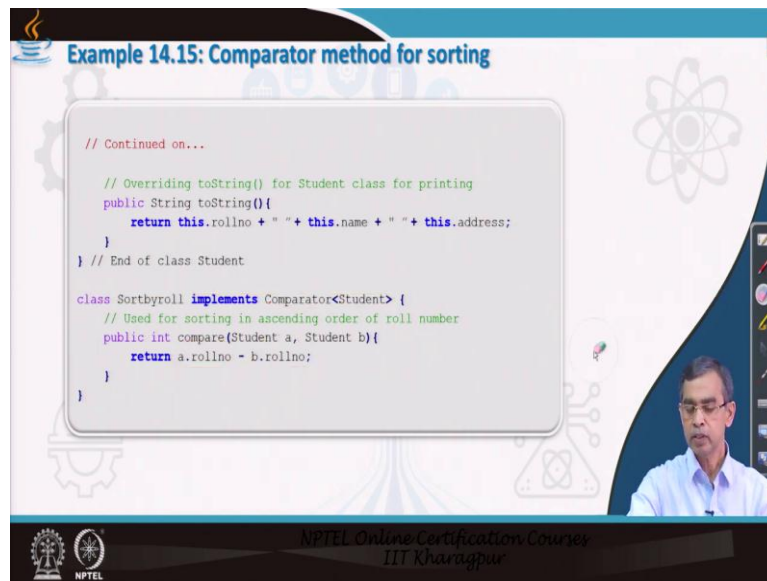
import java.util.*;
import java.lang.*;
import java.io.*;

// A class to represent a student.
class Student {
    int rollno;
    String name, address;
    // Constructor
    public Student(int rollno, String name, String address){
        this.rollno = rollno;
        this.name = name;
        this.address = address;
    }

    // Continued to next page ...
```

Now, comparator is one method, I have already given an idea about while I was discussing ArrayList for user defined data type. Now, arrays also can be extended to user defined data type. And here is an example. First let us define one class called the student with these constructors and few methods in it, the regular method to print.

(Refer Slide Time: 27:17)



The slide displays the following Java code:

```
// Continued on...  
  
// Overriding toString() for Student class for printing  
public String toString(){  
    return this.rollno + " " + this.name + " " + this.address;  
}  
} // End of class Student  
  
class Sortbyroll implements Comparator<Student> {  
    // Used for sorting in ascending order of roll number  
    public int compare(Student a, Student b){  
        return a.rollno - b.rollno;  
    }  
}
```

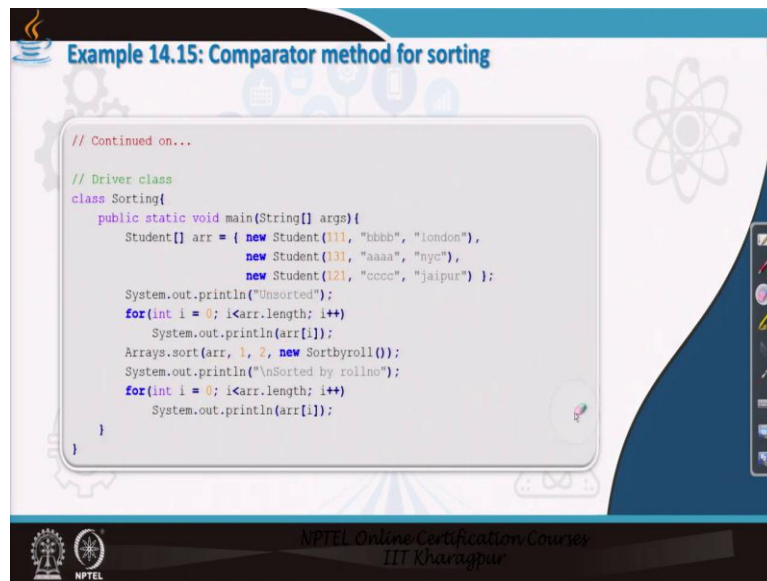
The slide also features the NPTEL logo and the text "NPTEL Online Certification Course IIT Kharagpur" at the bottom.

So, here is another method which basically convert the all elements of the class to string for better printing and again defining this comparator method you check and here we return. So, it basically return either true or false depending on if this is greater than this is. So, it will return true or it is false. Now, this comparator method is basically needs to be pass while we call a sort method so that it basically know on which field it to be sorted. In this case, we sort on the field roll number. Roll number where it is an integer type.

Now, likewise, we can call this method for a particular field because whenever you have to sort a user defined data type, it cannot be perform the sorting on the entire object. It can be performed only sorting based on a particular field in it. So, here is roll number for example. Likewise, it can be perform into other in this case, address or name which are the other type of attributes it is there. So, these are the method that you can consider for sorting, better sorting using comparator method, it is discussed.



(Refer Slide Time: 28:40)



**Example 14.15: Comparator method for sorting**

```
// Continued on...  
  
// Driver class  
class Sorting{  
    public static void main(String[] args){  
        Student[] arr = { new Student(111, "bbbb", "london"),  
                          new Student(131, "aaaa", "nyc"),  
                          new Student(121, "cccc", "jaipur") };  
  
        System.out.println("Unsorted");  
        for(int i = 0; i<arr.length; i++)  
            System.out.println(arr[i]);  
  
        Arrays.sort(arr, 1, 2, new Sortbyroll());  
        System.out.println("\nSorted by rollno");  
        for(int i = 0; i<arr.length; i++)  
            System.out.println(arr[i]);  
    }  
}
```

NPTEL Online Certification Course  
IIT Kharagpur

And this is basically demo of the program. Again I should suggest you the create the objects, only here 3, you can create at least 10 objects and the run the program to see how it basically sort and while we are doing, you see call calling the sort method over the array, here we want to sort on the sub list from 1 to 2 and here we pass the comparator method which we have discussed in the previous slides in the comparator sort by roll method it is there.

So, we have defined a method of our own how the comparator can work to sort it. Because for sorting, comparison operation needs to be defined for some data which is custom type. For regular data integer, number or everything not an issue, but for others it is really an issue.

(Refer Slide Time: 29:33)

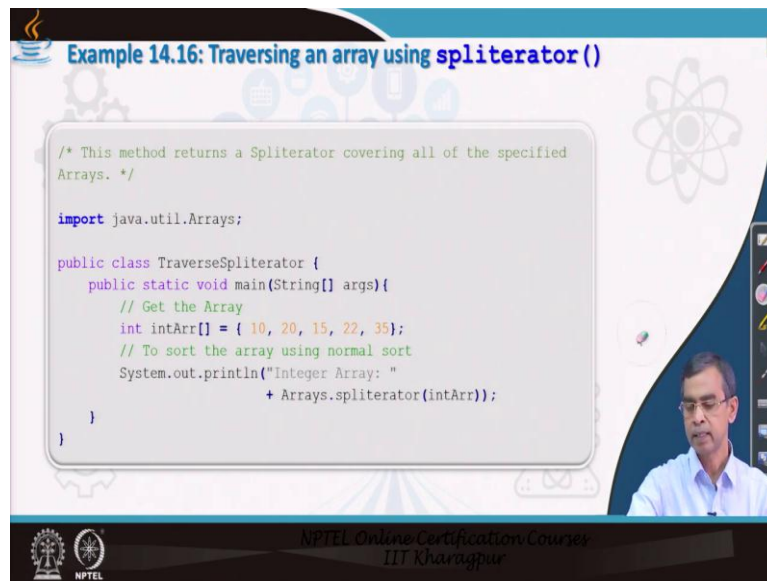


**Traversing Arrays**

NPTEL Online Certification Course  
IIT Kharagpur

### Example 14.16: Traversing an array using spliterator ()

```
/* This method returns a Spliterator covering all of the specified Arrays. */  
  
import java.util.Arrays;  
  
public class TraverseSpliterator {  
    public static void main(String[] args){  
        // Get the Array  
        int intArr[] = { 10, 20, 15, 22, 35};  
        // To sort the array using normal sort  
        System.out.println("Integer Array: "  
            + Arrays.spliterator(intArr));  
    }  
}
```



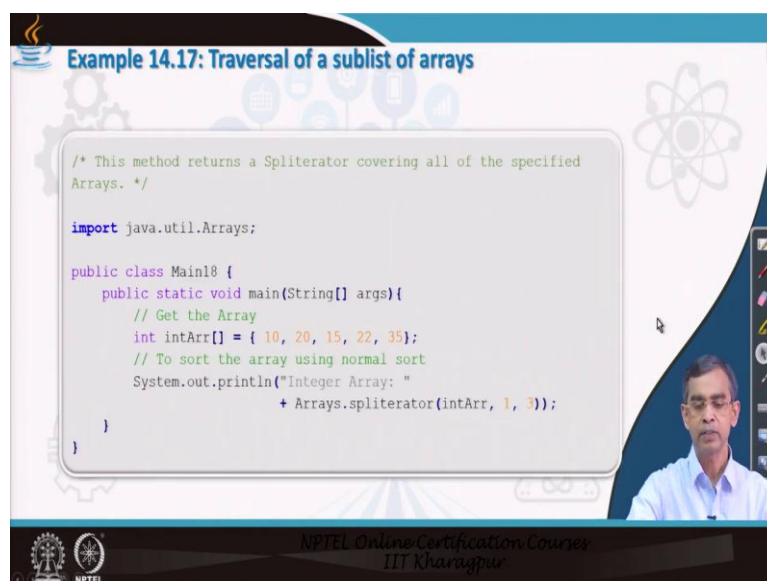
NPTEL Online Certification Courses  
IIT Kharagpur

Now, traversing arrays. It is basically same. There are different methods of traversing. This example, use one spliterator which is declared in the collection object, you just call it, it will see how one by one this will print f this array for you. So, there are many iterator methods are there which we will discuss not now actually because this is an advanced topic, so, we will discuss once the all collection, all type of collection because we are now at the very beginning of discussing about only two collection, ArrayList and Arrays. There are many more collections. My plan is to discuss this iterators, the different iterators ones all collections are covered. Then we will be able to have a good view of this iterators later.

(Refer Slide Time: 30:23)

### Example 14.17: Traversal of a sublist of arrays

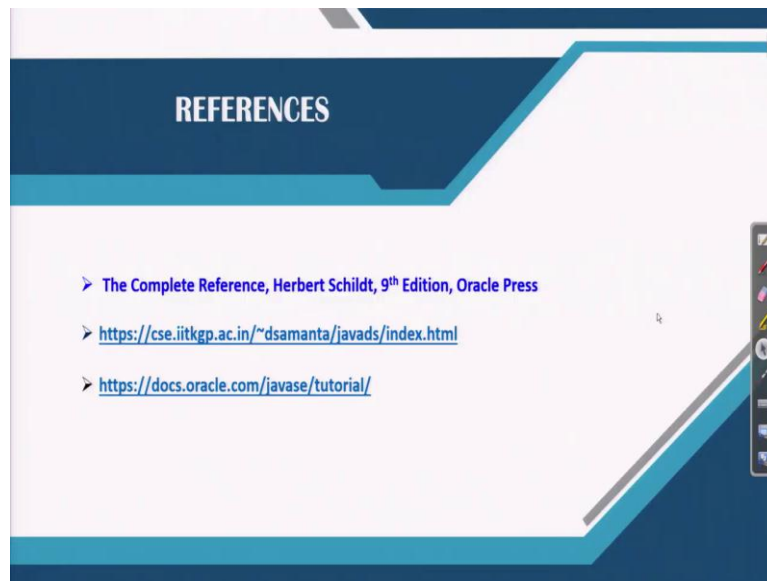
```
/* This method returns a Spliterator covering all of the specified Arrays. */  
  
import java.util.Arrays;  
  
public class Main18 {  
    public static void main(String[] args){  
        // Get the Array  
        int intArr[] = { 10, 20, 15, 22, 35};  
        // To sort the array using normal sort  
        System.out.println("Integer Array: "  
            + Arrays.spliterator(intArr, 1, 3));  
    }  
}
```



NPTEL Online Certification Courses  
IIT Kharagpur

So, this is the method by which you can traverse it. And these are traversal or sub list also you can do it.

(Refer Slide Time: 30:26)



For more studies, details about this Arrays class, you can consult these link here. this is an important link where all the materials I have used in the slides including all the slides also you can get it from this link also and this is just about data structure, you have already have the idea, I hope in the lecture we have discussed the theory of data structures.

And regarding this arrays class, a details account of this class you can get it from the Oracle tutorial slides. So, this is for the extra material for your studies. My advice again is that you should study all these programs that we have mentioned here, every lecture videos at least you can see 10 to 15 programs are there. I could not include much more program because of the time limits.

There are few more programs you can find from these java dot the docs oracle dot com, java c tutorial files there and from the given link also you can find many programs. So, my suggestion is that you should download all the programs, test from your own side according to your own speed of understanding. Thanks for your attention.