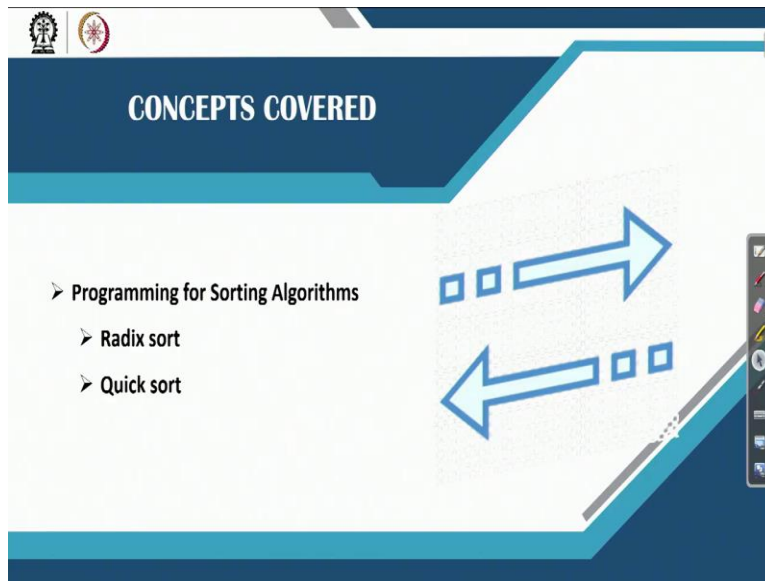


Data Structure and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 54
Programs for Sorting (Part-2)

We are discussing about writing programs for sorting techniques in the last video lectures we have discuss few sorting techniques programming for 4 sorting algorithms.

(Refer Slide Time: 0:33)



Here we will discuss 2 very good sorting method called radix sort and quick sort.

(Refer Slide Time: 0:41)

The slide features a central title "Radix Sort Algorithm" in blue text. To the left is an icon of a coffee cup with steam. The background is decorated with various icons including gears, a tree with nodes, a hard hat, and a molecular structure. A small video inset of a man is visible in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

Now radix sort algorithm as you know for different type of data the different radix to be consider.

(Refer Slide Time: 0:52)

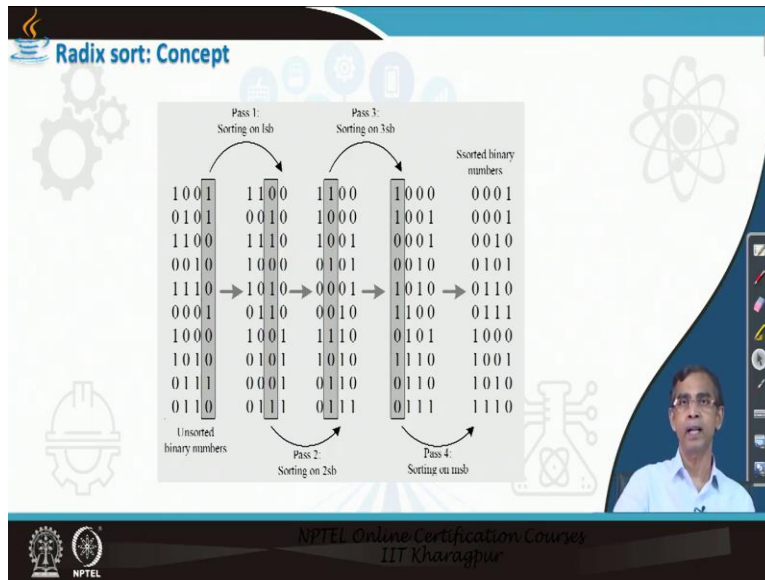
The slide is titled "Radix sort: Concept" and includes a bullet point: "A sorting technique which is based on *radix* or *base* of constituent elements in keys is called radix sort. The radixes and bases in few important number systems are listed in the table given below." Below the text is a table with the following data:

Number system	Radix	Base	Example
Binary	0 and 1	2	0100, 1101, 1111
Decimal	0,1,2,3,4,5,6,7,8,9	10	326, 12345, 1508
Alphabetic	a, b, ..., z A, B, ..., Z	26	Ananya, McGraw
Alphanumeric	A, ..., Z, a, ..., z, and 0...9	36	06116014 05CSS201

The slide also features a small video inset of a man in the bottom right corner and the NPTEL footer.

So, writing generic program is really very difficult for this because the bucket can varies an accordingly memory can be different.

(Refer Slide Time: 1:00)



But here we will discuss about the radix sort. I will give you the radix sort for integer numbers and again the radix sort while you write the code you have to think about how many maximum number of elements rather we can say digits in case of decimal number system or bits in case of binary number systems are there. Now this is one example of binary number system that we have discussed while we have discussing this radix sub technique.

You can implement fast this is binary sort is easy. So, only 2 buckets are there logic is also very simple only the important thing that you have to think about is that how to find the considered element starting from the LSB MSB and everything. I will give you a logic for that so that you can follow the same logic but you can try with other different type of data.

(Refer Slide Time: 2:02)

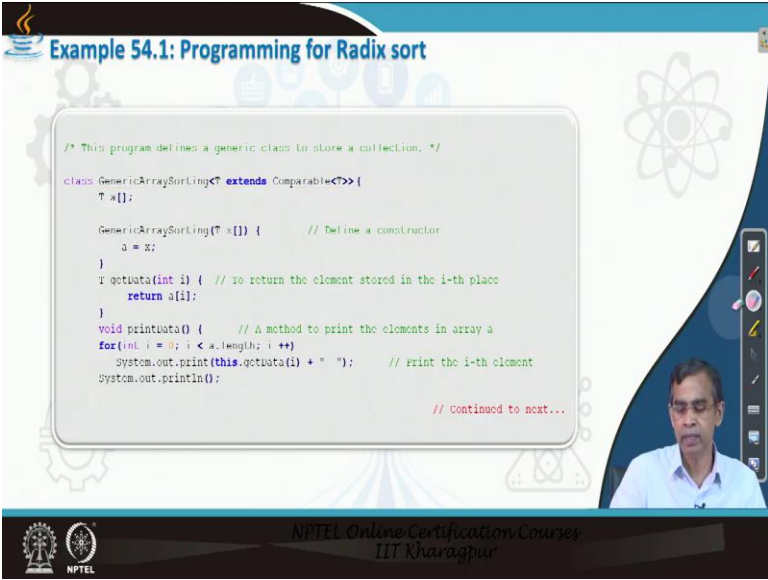
```
* For each base in the number system *
1. For i = 1 to c do // c denotes the most significant position
2.   For j = 1 to n do // For all elements in the array A
3.     x = Extract(A[j], i) // Get the i-th component in the j-th element
4.     Enqueue(Qi, A[j])
5.   EndFor // Combine all elements from all auxiliary arrays to A (assume A is empty*)
6.   For k = 0 to (b-1) do
7.     While Qk is not empty do
8.       y = Dequeue(Qk) // Dequeue of y from the queue Qk
9.       Insert(A, y) // Insert y into A
10.    EndWhile
11.  EndFor
12. EndFor
13. Stop
```

So, here is basically logic extract is required because we have to extract the i th element in the i th pass for a number actually. And in case of binary bits whatever be the total number of string size is there absolute no problem it will run it. So, the number of alteration will depends on how many total number of bits or number of digits are there in a system. And the number of buckets is basically depends on algorithm radix of the system.

For the decimal number system number of buckets or you can say number of arrays rather that needs to be stored in case of this one is the 10. In case of binary system only 2 and size of the arrays that is bucket actually should be same of the total number of element that is there in their input list. So, if you want to sort n elements of list of n numbers, then the size of the bucket or each bucket should be of n actually that is the problem.

So, that is why this sorting algorithm radix sort is okay not good for memory constant write procedure environment because you have to have huge amount of memory if n is say suppose 1000, then a decimal number system is used, then definitely 10 into 1000 extra memory is required to do the sorting technique. Otherwise, it is very easy actually because implementation is very easy.

(Refer Slide Time: 3:33)



Example 54.1: Programming for Radix sort

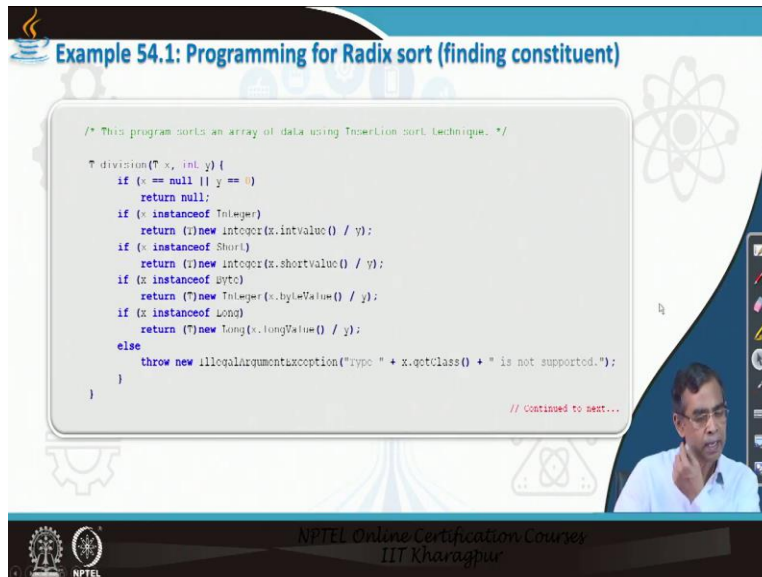
```
/* This program defines a generic class to store a collection. */  
class GenericArraySorting<T extends Comparable<T>> {  
    T a[];  
  
    GenericArraySorting(T a[]) { // Define a constructor  
        a = a;  
    }  
    T getdata(int i) { // To return the element stored in the i-th place  
        return a[i];  
    }  
    void printdata() { // A method to print the elements in array a  
        for(int i = 0; i < a.length; i++)  
            System.out.print(this.getdata(i) + " "); // Print the i-th element  
        System.out.println();  
    }  
}
```

// continued to next...

Now here is the code that you can think about I am giving a code for you hint you try to see how I have solved this problem. And it may take some time to understand if you want to understand according to your own level of understanding but I am giving highlighting how I am I have written the code here. So, as usual I have declared this as a generic so that it can take any type of number or string or long, float, long whatever it is there I am not using the float number because floating point is not good actually to you use it, but it can take floating point number also starting from the lowest significant digit to the most significant digit including decimal also but here decimal needs to be eliminated there.

So, we are declaring the generic type and then print data to auxiliary method to print the array of element that is a as usual for other so it is not so difficult to understand.

(Refer Slide Time: 4:34)



Example 54.1: Programming for Radix sort (finding constituent)

```
/* This program sorts an array of data using Insertion sort technique. */  
  
T division(T x, int y) {  
    if (x == null || y == 0)  
        return null;  
    if (x instanceof Integer)  
        return (I)new Integer(x.intValue() / y);  
    if (x instanceof Short)  
        return (I)new Integer(x.shortValue() / y);  
    if (x instanceof Byte)  
        return (I)new Integer(x.byteValue() / y);  
    if (x instanceof Long)  
        return (I)new Long(x.longValue() / y);  
    else  
        throw new IllegalArgumentException("Type " + x.getClass() + " is not supported.");  
}
```

// Continued to next...

Now next part is basically how I am just extracting each component of a given elements here you can see how I do it so it is basically x and y, y is depending upon radix if you use the binary system it will be 2 otherwise, it is 10 depending on number system decimal if it is hexadecimal then so I hexadecimal sixteenth and depending on this one and then you just see how exactly we are dividing each elements and then getting each component in the elements.

And so this basically to take about for the different sort integer bits and long so this means that this program will work for integer, sort byte and long to 3 different. Byte is basically for binary numbers it will work. Anyway this basically implementation that you can think about.

(Refer Slide Time: 5:34)

Example 54.1: Programming for Radix sort (finding constituent)

```
/* this program find a constituent element. */
T modulus(T x, int y) {
    if (x == null || y == 0)
        return null;

    if (x instanceof Integer)
        return (T)new Integer(x.intValue() % y);
    if (x instanceof Short)
        return (T)new Integer(x.shortValue() % y);
    if (x instanceof Byte)
        return (T)new Integer(x.byteValue() % y);
    if (x instanceof Long)
        return (T)new Long(x.longValue() % y);
    else
        throw new IllegalArgumentException("type " + x.getClass() + " is not supported.");
}
// continues to next...
```

NPTEL Online Certification Course
IIT Kharagpur

Now here is the next part of the code that you can think about how I am giving you the modular suppression is require in our to find the modular value. So, that you can find a so the different constituent you can find it. So, this is the part and then we can go to the bucketing.

(Refer Slide Time: 5:55)

Example 54.1: Programming for Radix sort (auxiliary methods)

```
/* this program find a constituent element. */
T getMax() {
    T max=a[0];
    for (int i=1; i<a.length; i++) {
        if (a[i].compareTo(max)>=0)
            max=a[i];
    }
    return max;
}

private static <T> T[] copyArray(T[] source) {
    return source.clone();
}
// continues to next...
```

NPTEL Online Certification Course
IIT Kharagpur

So, this part is basically being the bucketing here so depending on the value that we can write and then the bucketing is there and then you can copy the bucket. And finally we can store the element and then sort it.

(Refer Slide Time: 6:08)

Example 54.1: Programming for Radix sort (auxiliary methods)

```
/* This program find a constituent element. */  
  
void countSort(int exp) {  
    int i, size=a.length, count[]=new int[10];  
    T[] output=new T[size];  
    for(i=0;i<size;i++)  
        count[modulus(division(a[i],exp),10).intValue()]++;  
  
    for(i=1;i<10;i++)  
        count[i]+=count[i-1];  
  
    for(i=size-1;i>=0;i--) {  
        output[count[modulus(division(a[i],exp),10).intValue()]-1]=a[i];  
        count[modulus(division(a[i],exp),10).intValue()]--;  
    }  
    for(i=0;i<size;i++)  
        a[i]=output[i];  
}  
  
// Continue to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

So, this basically the sorting that I follow it and it basically give one each element then put into the bucket and then sort it there. So, this is basically the main sorting activities

(Refer Slide Time: 6:24)

Example 54.1: Programming for Radix sort (auxiliary methods)

```
/* This program find a constituent element. */  
  
void radixSort() {  
    int exp=1;  
    T max=this.getMax();  
    Long maxLong=new Long(division(max,exp).longValue());  
    for(;>exp=exp*10){  
        this.countSort(exp);  
        maxLong=division(max,exp).longValue();  
    }  
} // End of the generic class definition  
  
// Continue to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

And then finally it basically perform the sorting technique here. So, call all the method that we have discussed so this is the main method whether radix method actually you can say. So, getting the elements just taking the bits or digits and then perform hexa bucketing. It basically solves the program. So, this is the code I have written for you. You can check it you can run this code for

the beginning you can see whether it is working or not giving the input. And how to run the code master program is given that master program you can think about it.

(Refer Slide Time: 7:09)

Example 54.1: Programming for Radix sort (auxiliary methods)

```
/* This program find a constituent element. */  
  
public class TestRadixSort{  
    public static void main(String[] args) {  
        Integer i[] = {20, 85, 79, 18, 88};  
        // Store the data into generic array  
        GenericArraySorting<Integer> arrayInt = new GenericArraySorting<Integer>(i);  
        // Printing the data...  
        System.out.print("Array before sorting : ");  
        arrayInt.printData();  
        arrayInt.radixSort();  
        System.out.print("Sorted array is : ");  
        arrayInt.printData();  
        System.out.println();  
    }  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Here is the main method that basically giving the demo of the radix sort. So, this is the main method this is the data that you have given, 2 digit each and you can see the agitation of the algorithm you can again run this program with varieties of input combinations, you can take some is 1 digit number, some is 2 digit numbers, some is 3 digit number also you can check it also will work and also you can take with different type of data here integer long, sort, byte and then integer also you can take it.

String also you can take it also you can see how it works. Because you convert into this and then it can work if convert in the Unicode and then Unicode to this one some modification you may needed to if you want work with string, but radix sort is for number system only. If you want to apply sorting algorithm for your user defined type it may not work. So, it is not advisable to follow this one. So, it has this that limitation. So, this algorithm is you can run for different type of input, different input cases or different test cases and check that it is working.

But running but other than running you can see how the logic has been implemented and what is the code it is there. And that you can follow otherwise you can see this is the logic that you have to implement. If you understand then you better write your own methods own sharps methods

and call them in your own program better that is the good idea that you can think about it. So, first you try to understand logic and then the algorithm you can follow.

Whatever the algorithm that is there algorithm of higher level programming and then you can think the programming according to your own understanding that is the best way that you can write programs. So, this that is more advisable, so this is the radix sort.

(Refer Slide Time: 9:11)



Now quick sort algorithm as we have discussed it is based on the dived and conquer principle. This is one very what is call the popular sorting algorithm you can find readymade code in many books also, but practicing code and writing the program of own following the logic that is the most advantageous and that you should do.

(Refer Slide Time: 9:45)

The slide illustrates the partitioning step of the quick sort algorithm. It shows an array with indices 1, 2, ..., n. A pivot element is chosen, and the array is divided into a 'Left sub-list' and a 'Right sub-list' based on the pivot. The diagram shows the pivot element being moved to its correct position, with elements less than the pivot to its left and elements greater than the pivot to its right.

NPTEL Online Certification Courses
IIT Kharagpur

So, let us see what is the first logic of this algorithm. Here divide and conquer mean the fast you have to write the partition method again there are many algorithms you can find implementing this partition, you can follow any algorithm, follow the algorithm write your own code do not copy code from here and there.

(Refer Slide Time: 10:02)

```
Algorithm Partition(left, right)
1. loc = left // The left most element is chosen as the pivot element
2. While (left < right) do // Repeat until the entire list is scanned
3.   While(A[loc] < A[right]) and (loc < right) do // Scan from right to left
4.     right = right - 1 // No interchange. Move from right to left
5.   EndWhile
6.   If (A[loc] > A[right]) then
7.     Swap(A[loc], A[right]) // interchange the pivot and the element at right
8.     loc = right // The pivot is not at the location right
9.     left = left + 1 // Next scan (left to right) begins from this location
10.  EndIf
11.  While (A[loc] > A[left]) and (loc > left) do // Scan from left to right
12.    left = left + 1 // No interchange. Move from left to right
13.  EndWhile
14.  If (A[loc] < A[left]) then
15.    Swap(A[loc], A[left]) // interchange the pivot and the element at left
16.    loc = left // The pivot is not at the location left
17.    right = right - 1 // Next scan (right to left) begins from this location
18.  EndIf
19. EndWhile
20. Return (loc)
21. Stop
```

Continue ...

NPTEL Online Certification Courses
IIT Kharagpur

I am giving you the algorithm here you can follow this algorithm also this is the one algorithm. There are plenty of algorithms are available from any book in the book of (10:13) structure also several algorithms is given. How to implement partition algorithm? So, best idea would be this one

(Refer Slide Time: 10:20)

```
1. Loc = Partition(left, right) // left and right are two pointers to locate partitions
// at left and right, respectively
2. If (left < right) then // Check for the termination condition
3. QuickSort(A, left, loc-1) // Perform quick sort over the left sub list
4. QuickSort(A, loc+1, right) // Perform quick sort over the right sub-list
5. EndIf
6. Stop
```

And this is the quick sort algorithm is a recursive (10:23) and if you do not know recursion, then definitely it will first you should practice about how to write recursive program. In the book of (10:34) there are many example is given how to practice recursion also that you can follow.

(Refer Slide Time: 10:38)

Example 54.2: Programming for Quick sort (Generic class)

```
/* This program defines generic class and many utility methods in it. */  
  
class GenericArraySorting<T extends Comparable<T>> {  
    T a[];  
    GenericArraySorting(T x[]) { // define a constructor  
        a = x;  
    }  
    T qctuata(int i) { // To return the element stored in the i-th place  
        return a[i];  
    }  
    void printuata() { // A generic method to print the elements in array a  
        for(int i = 0; i < a.length; i++)  
            System.out.print(this.qctuata(i) + " "); // Print the i-th element in a  
        System.out.println(); // Print a new line  
    }  
}
```

// Continued to next...

NPTEL Online Certification Courses
IIT Kharagpur

And here is the quick sort techniques, again I am defining is using the generic facility generic programming. In all my programming practices or demonstration, I gave the generic implementation and that is the most important thing as an advance programmer you should follow it. Now here is the generic implementation of the class. So, this is the generic the simple method get, print all this things are standard here.

(Refer Slide Time: 11:15)

Example 54.2: Programming for Quick sort (Generic class)

```
// Continues on...  
  
/* This method for partition is defined below. */  
  
int partition(T arr[], int low, int high) {  
    T pivot = arr[high];  
    int i = (low-1); // index of the left most element  
    T temp;  
    for (int j=low; j<high; j++) {  
        if (arr[j].compareTo(pivot)<0) {  
            j++;  
            temp = arr[j]; // swap arr[i] and arr[j]  
            arr[i] = arr[j];  
            arr[j] = temp;  
        }  
    }  
  
    // swap arr[i+1] and arr[high] (or pivot)  
    temp = arr[i+1];  
    arr[i+1] = arr[high];  
    arr[high] = temp;  
    return i+1;  
}
```

// Continued to next...

NPTEL Online Certification Courses
IIT Kharagpur

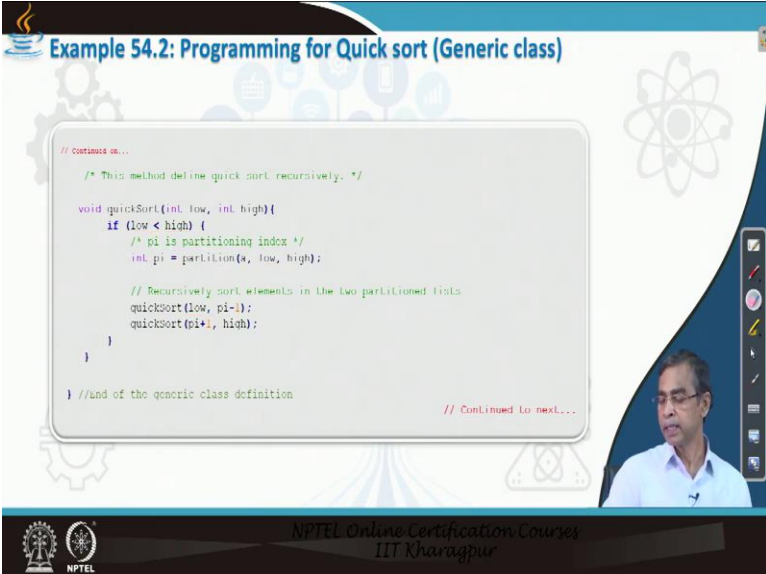
Now next is the code for the different routine so first of all I write to write the partition routine here. I just gave an idea about an implementation of the logic partition. So, this partition is

basically including swapping and everything that is works. You can just simply test that how this partition works for given input, so partition means you take the first element and place into its least into its position so that any element in the right part of this right element is basically greater than and any element in the left part of this element is lower.

And again you can try partition with many elements are of the same that mean duplicate two or three numbers are the same you can see that it also still working or not because this are the test case that you have to some student write code it cannot work for many cases. Many situation that is not good so you have to check it and again you can check that whether giving the order in ascending order, descending order, whether it is working or not that you can check it.

And then finally, after giving some string data that also you can check so that it works, you can check whether it is working or not. So, this are the different situation that only partition can be practice more extensively that I should say suggest you to do to it partition is the heaviest part in the quick sort algorithm. So, this is the logic that I have given this algorithm that you can follow you can check that it is working. Then you try to understand the logic it is follows or you can follow any other algorithm and then you can implement it so this is the partition technique and then recursive version of this algorithm we can write it.

(Refer Slide Time: 13:01)



The slide displays the following C++ code for a recursive quick sort algorithm:

```
// continue on...  
/* This method define quick sort recursively. */  
void quickSort(int low, int high){  
    if (low < high) {  
        /* pi is partitioning index */  
        int pi = partition(a, low, high);  
  
        // Recursively sort elements in the two partitioned lists  
        quickSort(low, pi-1);  
        quickSort(pi+1, high);  
    }  
}  
//End of the generic class definition  
// Continued to next...
```

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

This are the following part of the code is basically quick sort technique as we see here we call partition and then we call the quick sort or the left part of the list and this the right part of the list

and this is the boundary condition, whenever you write any record sheet condition the boundary condition is there. So, this is the boundary condition how long the quick sort technique should be call recursively.

So, this is important this is the quick sort technique as a whole as you see the partition is only method that you have to think about it. So, here after partition you can print the list you can see how it is printing the elements. So, this is the implementation of the algorithm.

(Refer Slide Time: 13:43)

```
// Continued on...  
/* This method apply quick sort. */  
  
public class TestQuickSort {  
    public static void main(String[] args) {  
        Integer i[] = {29, 41, 29, 74, 99};  
        // Store the data into generic array  
        GenericArraySorting<Integer> arrayToObj = new GenericArraySorting<Integer>();  
        // Printing the data...  
        System.out.print("Array before Sorting : ");  
        arrayToObj.printData();  
        arrayToObj.quickSort(0, i.length-1);  
        System.out.print("Sorted Array is : ");  
        arrayToObj.printData();  
        System.out.println();  
    }  
}
```

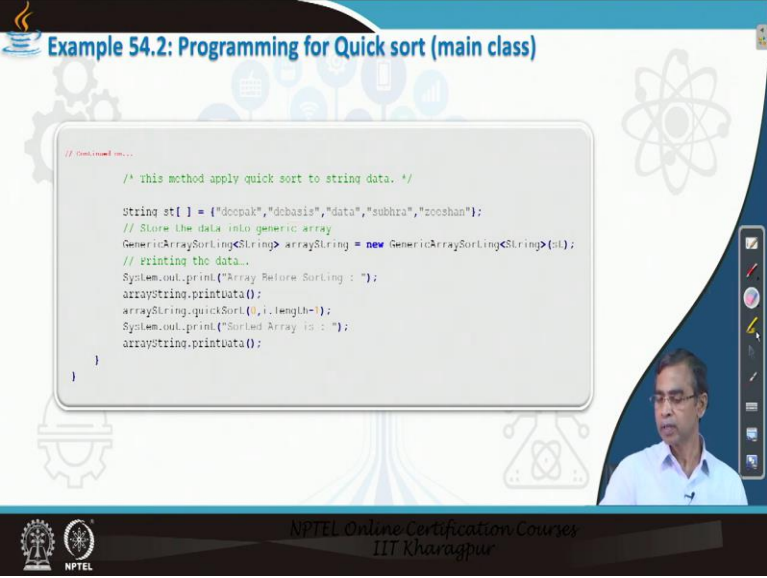
NPTEL Online Certification Courses
IIT Kharagpur

And then finally let us come to the discussion of testing the program. Here, this is basically the demo, which basically gives basically test the program that you have develop. We consider input array integer and then we just create the generic arrays and then call the method print data for the initial list. And then call the quick sort and you check that how you call the quick sort method in this case. And finally we print the data it right. So, here we have considered integer of all size like this one this the standard input we are considering to give the demo for all.

But you should consider different elements of different size different length and the different type and again different pattern arrangement. Either ascending, descending or whatever it is there. Again the best idea would be that you can generate some random numbers store in a file you can write open the file get the data put into array and then that array can be pass to the sorting technique that you can try and practice that will be the best idea to do that.

And here again I have used the integer data you can use the string some names and then pass this there you declare the string and then you can call the method it also will work. Next is basically user defined data type that if you want to do then you have to implement compareTo method. So, compareTo method you can add into the end of the generic class declaration as the user defined declaration the compareTo method it works for that. So, you can define the class student all the constructor that you can defined and finally compareTo how to compare. And it will work for you. So, this is the idea about quick sort implementation.

(Refer Slide Time: 15:38)



The slide displays a Java code snippet for a quick sort algorithm applied to a string array. The code is as follows:

```
// Example 54.2: Programming for Quick sort (main class)

/* This method apply quick sort to string data. */

String st[] = {"doopak","dobasis","data","subhra","sooshan"};
// Store the data into generic array
GenericArraySorting<String> arrayString = new GenericArraySorting<String>(st);
// printing the data...
System.out.println("Array Before Sorting : ");
arrayString.printData();
arrayString.quickSort(0, arrayString.getLength()-1);
System.out.println("Sorted Array is : ");
arrayString.printData();
}
```

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

And here actually I gave the idea about how the sort can be executed using some string so that algorithm you can check and you see that it is also working so that is the one hint it is given there. So, I hope you have understood about how to implement different sorting algorithm

(Refer Slide Time: 15:57)

The slide features a dark blue header with the word 'REFERENCES' in white. Below the header, the text reads: 'For details of the discussed sorting algorithms', 'See the book', 'Classic Data Structures' (in green), 'Chapter 10' (in blue), and 'Prentice Hall of India'. To the right of the text is a book cover for 'Classic Data Structures' by Debasis Samanta, Second Edition, published by Prentice Hall. The book cover is purple and white with a stylized 'D' logo. The slide also includes a vertical toolbar on the right side with various icons for navigation and editing.

I have only consider few sorting algorithms in this programming demonstration and it gives an idea about how to write code for programming. Now sorting and searching two types of algorithms usually programmer consider to practice their programming and they can learn many things from right implementing all those sorting algorithm.

So, there are many more sorting algorithms also known like counting sort, sales sort and then there are many other sorting like techniques that you can implement (thou) in this book cover many sorting techniques and many searching techniques also there I could not cover all sorting and searching techniques into the code, but you can write practice that will help you I think fine. We can stop it here, thank you.