

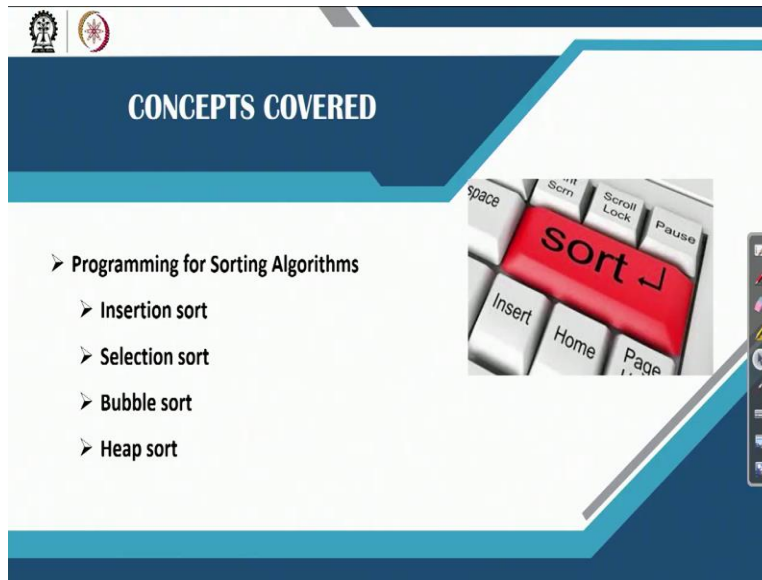
Data Structure and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 53
Programs for Sorting (Part-1)

So, we are discussing about sorting techniques and sorting is used to sort a collection, collection consist of any type of data. Now today we will discuss about programing aspect of different sorting techniques. We have learned about the different sorting techniques, their concepts and then algorithms there in a complexity analysis but now we will just try to learn about how to program those sorting algorithm.

Now before going to start this discussion, I want mention one important thing, programming that I will discuss it just for a hint but if you should not just only run the program, rather best idea would be that if you understand the logic and then follow some algorithms whatever algorithm I have discuss in this course that you can follow or the algorithm those that they are in books whatever the things if you learned that this algorithm is you are more comfortable you can follow this algorithm no issue, then try to write the code of your own.

So, here whatever the course that I am going to explain or illustrate this is for your hint only. So, you can check you see that program is running but you will not be happy for that only the thing is that if you can write the code of your own and then run it and then you should feel happy it is. Now so this code that I am giving you this is just for your hints only. So, that in case suppose you find it difficult to write the code you can follow that how it is being implemented.

(Refer Slide Time: 2:19)



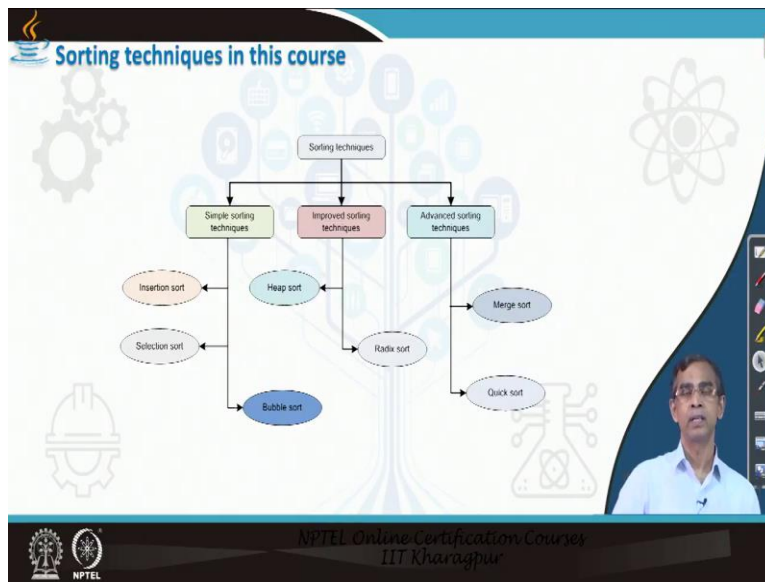
The slide features a dark blue header with the text 'CONCEPTS COVERED' in white. Below the header, there is a list of topics under the heading 'Programming for Sorting Algorithms'. To the right of the list is an image of a computer keyboard with a prominent red key labeled 'sort'. The slide also includes two logos in the top left corner and a vertical toolbar on the right side.

- Programming for Sorting Algorithms
 - Insertion sort
 - Selection sort
 - Bubble sort
 - Heap sort

So, this part that is important, now let us see what are the codes that we are going to discuss today. This is related to 4 different sorting algorithms shall discuss namely; insertion sort, selection sort, bubble sort, and heap sort, and here again all these sorting that will implement considering the generic only. Generic means that you write the code once and then apply this codes to sort any type of data so that is a generic programming we will consider.

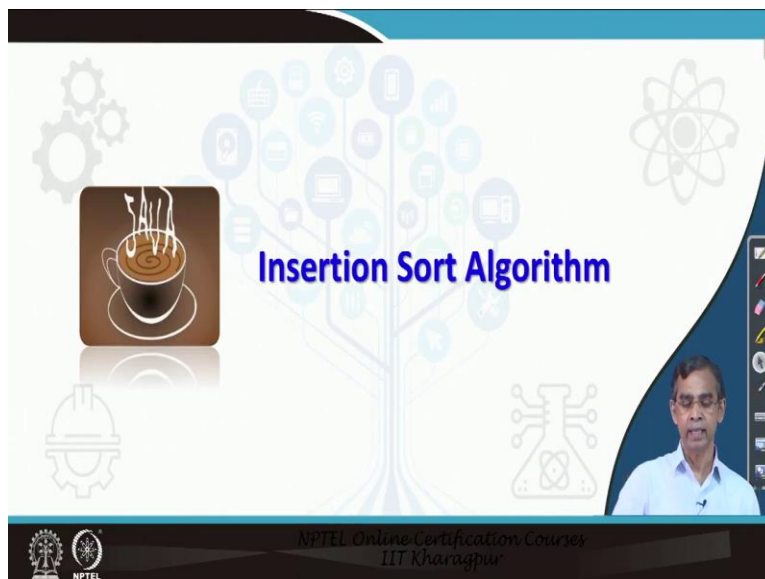
In many of the books you can find only the simple sorting technique to start to sort integer data, but here you should run the program. Whatever the program I have given or you can write the code then you should try to run the codes for any type of data including your user defined data type also that will be the most advantageous from this course if you can do that.

(Refer Slide Time: 3:10)



So, let us start about first insertion sort technique.

(Refer Slide Time: 3:14)



Insertion sort: Concept

The diagram illustrates the concept of insertion sort. It shows an input list with elements $K_1, K_2, \dots, K_j, \dots, K_n$. The element K_j is selected and inserted into the output list. The output list is scanned from right to left starting from the element immediately to the right of K_j . The position of K_j is found, and the elements to its right are shifted one place to the right. Finally, K_j is inserted at its correct position in the output list.

NPTEL Online Certification Courses
IIT Kharagpur

The program of the insertion sort I just do not want to discuss the sorting algorithm again because I understood that you have already covered it, so it the pre requisite for this.

(Refer Slide Time: 3:26)

Insertion sort: Algorithm

```

1  B[1]=A[1] // Initially first element in the input list is the first element in the output list
2  For j = 2 to N do // Continue taking one key at a time from the input list
3      K = A[j] // Select the key K from the input list
4      // K is the key under insertion
5      // Search the final location i in the output list
6      flag = TRUE // To control the number of scan in the output list
7      i = j-1 // Points the rightmost element in the output list
8      While (flag = TRUE) do // Scan until we get the place for K
9          If (K < B[i]) then
10             i = i-1 // Stop if the list is exhausted
11             If (i = 0) then
12                 flag = FALSE // Stop here
13             EndIf
14         Else
15             flag = TRUE // Stop here
16         EndIf
17     EndWhile
18     // Move to right one place all the keys from and after i+1
19     p = i
20     While (p >= 1) do
21         B[p] = B[p+1]
22         p = p-1
23     EndWhile
24     // Insert the keys at i+1 th place
25     B[i+1] = K // Insert the key at the (i+1)-th place
26 EndFor
27 Stop

```

NPTEL Online Certification Courses
IIT Kharagpur

So, this is the insertion concept that is their algorithm I have already mention you can follow this algorithm.

(Refer Slide Time: 3:30)

```
/* This program sorts an array of data using Insertion sort technique. */  
  
public class InsertionSort <T extends Comparable<T>> {  
    public static void insertionSort(T[] arr, int len){  
        for (int i = 0; i < len; i++){  
            int j = i;  
            while (j > 0 && arr[j].compareTo(arr[j-1]) > -1){  
                T temp = arr[j];  
                arr[j] = arr[j-1];  
                arr[j-1] = temp;  
                j = j-1;  
            }  
        }  
    }  
}
```

// Continued to next...

And I have followed this algorithm of course in this programming practice. Now here is the code you just follow the code how we have to write the write. We are declaring a class the name this class is declared as generic class as we have said so this is the t that mean this class can take any type of data but here we have to take the extends as you know this is basically limited to data which basically comparable. Now here the t the comparable type there are many datatypes those are comparable like all numeric datatypes those are basically sub class of the class numbers like say integer, then float and then long sort, double whatever it is there.

And also string type string class is also comparable. So, this t basically any sub class of the class number as well as string is basically you can use it. Otherwise if you want to, if you want to use this sorting algorithm for you own datatype say student, person or whatever it is there, then you have to implement this comparable method, the compareTo method or this class comparable. So, you have, in this class you can write that over write compareTo.

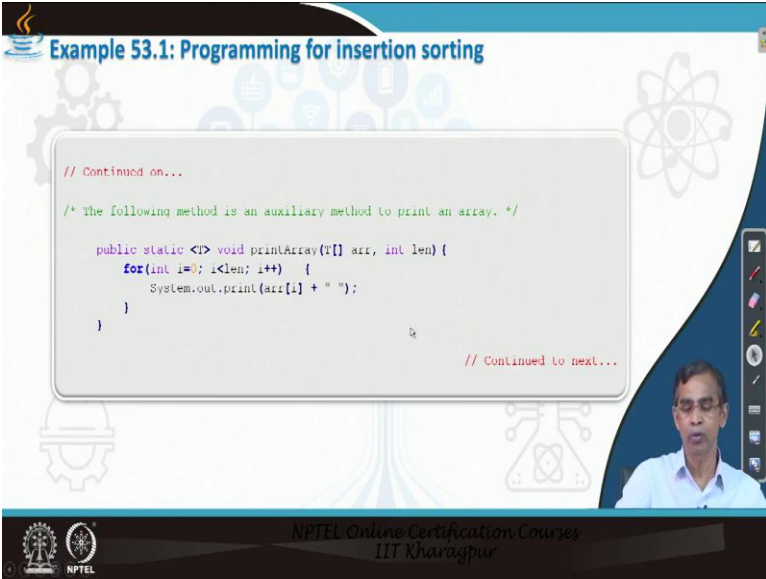
So, method compareTo can be written for this class then over write that procedure I will discuss also. I have discussed in earlier discussion also I have used that kind of concept but here again I will give some idea about how it can use it. Anyway so this, this is basically the concept you should follow that means I that you should write generic programming that is good only and this is the class that we this is the method which belong to this class this is the method is basically the implementation of the insertion technique algorithm.

This method takes an array of elements, elements of type `t` and this required another argument it is called the what is the size of this array that is needs to be passed. Then this is the simple logic it is called the logic that is discussed in case of insertion algorithm. It basically starts from the current location to check that whether the elements to be inserted in which position according to the partially sorted list actually this logic is implemented here.

And that you can write you can check it and here we are using `compareTo` in this scale number already the `compareTo` method is implemented but you can overwrite this `compareTo` method so that for any other datatype other than all those comparable type we can apply them. And so this is the procedure it is just simply whenever we found that the element needs to be placed here, so we just simply swap the element and that is the main logic of insertion sort.

So, algorithm we have discussed and then implementation logic is so simple that you have checked it. So, it is very simple so you can follow it. The code is written in a little bit proportional ways so that it is most compact and then reverse code and you can use it.

(Refer Slide Time: 6:53)



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

```
// Continued on...

/* The following method is an auxiliary method to print an array. */

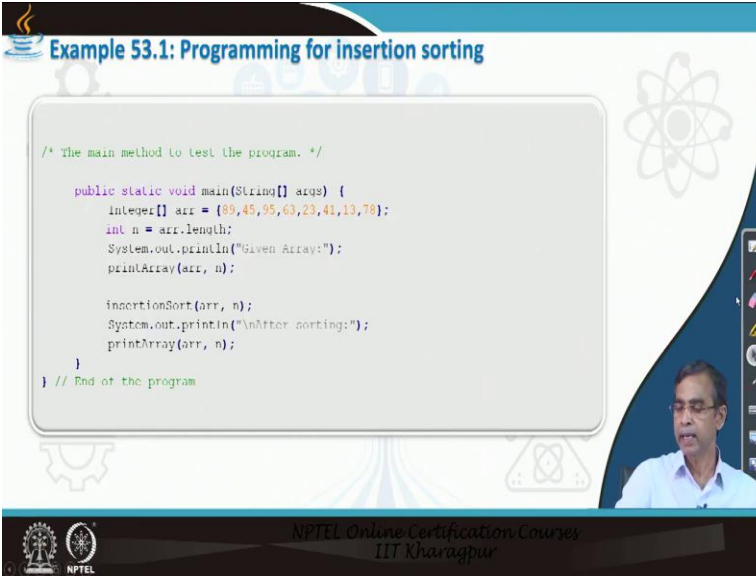
public static <T> void printArray(T[] arr, int len) {
    for(int i=0; i<len; i++) {
        System.out.print(arr[i] + " ");
    }
}

// Continued to next...
```

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 then finally you can write a driver program so that you can test this code writing a main method and this basically is the code to print the array that is simply for loop only you have used it.

(Refer Slide Time: 7:01)



Example 53.1: Programming for insertion sorting

```
/* The main method to test the program. */  
  
public static void main(String[] args) {  
    Integer[] arr = {89,45,95,63,23,41,13,78};  
    int n = arr.length;  
    System.out.println("Given Array:");  
    printArray(arr, n);  
  
    insertionSort(arr, n);  
    System.out.println("\nAfter sorting:");  
    printArray(arr, n);  
}  
} // End of the program
```

NPTEL Online Certification Courses
IIT Kharagpur

And here is the main method, the main class actually who is basically to test whether this program is working or not. So, here the main method which we discuss in within the same class actually otherwise you can write another class and say it is insertion sort demo and then you can call this method including that class declaration here.

Anyway, so here we consider an array of object so this is array and you just note that integer array we have declared because it is a generic so we have to pass the object only so that is a integer array. So, if you declared int array, it may not over for you because they generic so integer array it to be declare as an object like so this a wrapper method you have to call. So, this is basically the elements which is stored in into this array, array is an array of integer objects.

So, these elements, n is the array dot length because automatically complete how many elements are there you can print array the method we have already declared there to print the elements and then call the method insertion sort passing this array as an argument n is the length it basically sort the elements and then print the array. Now so this is the program that okay the simple few lines of code and that occurs that basically that matters how it sorts according the insertion sort other sorting algorithm in the same line actually only the logic is different.

But it is moreover the same actually and here I want to give some hints about or the tips that you can test it using different type of array here I have used a random array. You can think about array where all elements are same or you can consider an array of in ascending order or

descending order. So, different way you can think about it. And then you can run the program for different inputs and another consideration that you should do is that you can generate some random numbers, very large set of random numbers and store random numbers in a file.

You have already cover about file handling and then you can read the data from the file, put into the array and then stored into the integer array and then you can call this, so that will be the good way that you can run your sorting algorithm for very large set of numbers that is the good idea I hope you have understood it so that you should follow for all sorting techniques that you have write programs so this is the insertion sort.

(Refer Slide Time: 9:43)

Selection Sort Algorithm

NPTEL Online Certification Courses
IIT Kharagpur

Selection sort: Concept

(a) The list prior to the i -th iteration

(b) Select step

(c) Swap step

NPTEL Online Certification Courses
IIT Kharagpur

Now let us consider selection sort the algorithm we have already discussed in details with illustration is basically select the minimum elements and then take place in the minimum elements in the ith position if it is in the it is in ith alteration.

(Refer Slide Time: 10:01)

Selection sort: Algorithm

Algorithm SelectionSort (A, n)

1. For i = 1 to (n-1) do // (n-1) iterations
2. j = SelectMin(i, n) // Select the smallest from the remaining part of the list
3. If (i ≠ j) then // Interchange when the minimum is in remote
4. Swap(A[i], A[j])
5. EndIf
6. EndFor
7. Stop

Algorithm SelectMin (L, R)

1. min = A[L] // Initially, the item at the starting location is chosen as the smallest
2. minLoc = L // minLoc record the location of the minimum
3. For i = L+1 to R do // Search the entire part
4. If (min > A[i]) then // New the smallest is updated here
5. min = A[i]
6. minLoc = i // New location of the smallest so far
7. EndIf
8. EndFor
9. Return minLoc // Return the location of the smallest element
10. Stop

Algorithm Swap (X, Y)

1. X = X + Y // X holds total of both the values
2. Y = X - Y // Y now holds the previous value of X
3. X = X - Y // X now holds the previous value of Y
4. Stop

NPTEL Online Certification Courses
IIT Kharagpur

And this is the algorithm that we have given to you for your understanding it requires some few codes like swap because it require to be interchange and the select mean. How to find a minimum this is the algorithm very simple algorithm, finding minimum is a very simple task and then it is basically the main selection sort that you can do it.

(Refer Slide Time: 10:21)

Example 53.2: Programming for Selection sorting

```

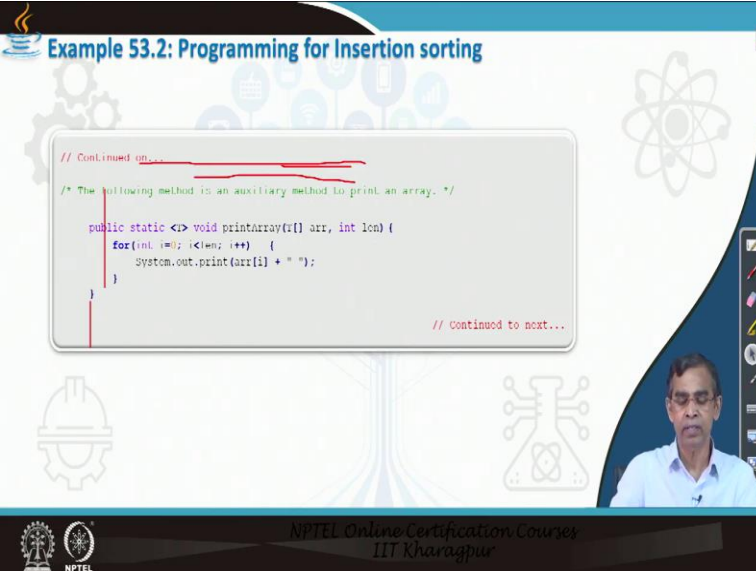
/* This program sorts an array of data using Selection sort technique. */
public class SelectionSort <T extends Comparable<T>> {
    public static void selectionSort(T[] arr, int len) {
        for (int i=0; i<(len-1); i++) {
            int minIndex = i;
            for (int j=i+1; j<len; j++) {
                if (arr[minIndex].compareTo(arr[j]) > 0) {
                    minIndex = j;
                }
            }
            T temp = arr[minIndex];
            arr[minIndex] = arr[i];
            arr[i] = temp;
        }
    }
}
    
```

// continued to next...

NPTEL Online Certification Courses
IIT Kharagpur

And here is the code that you can follow so again we just declare one generic class name of the class is selection sort. It is comparable so that any type of data and this is the method that we are going to discuss. It is basically same ways that the insertion sort we have done. And this is the logic of the selection sort that we have discuss about it and this is basically the method that we can find about the swapping. And that is okay in the finding the minimum and swapping it takes place in the same line actually here.

(Refer Slide Time: 11:00)



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

```
// Continued on...  
/* The following method is an auxiliary method to print an array. */  
  
public static <E> void printArray(E[] arr, int len) {  
    for(int i=0; i<len; i++) {  
        System.out.print(arr[i] + " ");  
    }  
}  
  
// Continued to next...
```

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

And then next part of the code that you can think about it is basically how to find a print, how to print the array. So, that is the sub method or auxiliary method you can write.

(Refer Slide Time: 11:09)

Example 53.2: Programming for Selection sorting

```
// Continued on...

/* The main method to test the program. */

public static void main(String[] args) {
    Integer[] arr = {89,45,95,23,41,13,63};
    int n = arr.length;
    System.out.println("Given Array:");
    printArray(arr, n);

    selectionSort(arr, n);
    System.out.println("\nAfter sorting:");
    printArray(arr, n);
} // End of the program
```

NPTEL Online Certification Courses
IIT Kharagpur

And then finally the main method that you can discuss about here, here is the main method so this is the main method in the same way with this state the insertion sort it is the same way you can create an initial array and then you can just sort using selection sort here. Here again we have considered integer array, you can consider some other types of array like long, float, or double, sort.

Even you can consider the string array also where the string can be putted and then you see that this algorithm what for any type of data including string also and if you over write the compareTo method or your user defined like student, person, and everything then you can create the array of objects and then you can call the selection sort for this array in the same. Only this part will different and then defining the compareTo method will be different. Then you can use this algorithm to sort any type of data that you want to sort so this is the selection sort.

(Refer Slide time: 12:19)

The image shows two slides from an NPTEL presentation. The top slide is titled "Bubble Sort Algorithm" and features a coffee cup icon. The bottom slide is titled "Bubble sort: Concept" and contains a diagram illustrating the sorting process. The diagram shows two vertical arrays representing the state of the array during the sorting process. The left array shows elements being compared and swapped, with arrows indicating the movement of elements. The right array shows the result of the sorting process, with elements being placed in their sorted positions. Labels in the diagram include: "Smaller elements bubble up in inner loop", "i-th largest elements sinks and placed here", "i-th outer loop", and "(i-1) elements are sorted and stored here". The NPTEL logo and "NPTEL Online Certification Courses IIT Kharagpur" are visible at the bottom of both slides.

Then let us discuss about the bubble sort, bubble sort you know exactly how you can do it, n number of passes it is required in each pass the largest element will go to the bottom direction and then smaller number will try to pop up little bit upwards and finally after n minus 1 alteration, all elements will be in sorted order.

(Refer Slide Time: 12:44)

Bubble sort: Algorithm

Algorithm BubbleSort (A, n)

```
1. For i = 1 to n-1 do // Outer loop with n-1 passes
2.   For j = 1 to n-i do // Inner loop: in the i-th pass
3.     If (A[j] > A[j+1]) then // Check if two elements are out of order
4.       Swap(A[j], A[j+1]) // Interchange A[j] and A[j+1]
5.     EndIf
6.   j = j + 1 // Move to the next elements
7. EndFor
8. EndFor
9. Stop
```

Algorithm Swap (X, Y)

```
1. temp = X // Store the value in X in a temporary storage space
2. X = Y // Copy the value of Y to X
3. Y = temp // Copy the value in temp to Y
4. Stop
```

NPTEL Online Certification Courses
IIT Kharagpur

We have discussed the algorithm and this is the algorithm that you can follow swap method that is required for interchanging that you can do. And another thing that you should consider here we have mentioned that the bubble sort can be little bit modified it to sort the element more efficient way for an input list when the elements define in an already in sorted order. Otherwise, whatever with that base case what case ever is case sit usually takes order being kept but with this modification the best case can take only order of n time. That means with n number of comparing that is require so little modification that we flag that you can use it.

(Refer Slide Time: 13:25)

Example 53.3: Programming for Bubble sorting

```
/* This program defines a generic class to store a collection. */
class GenericArraySorting<T extends Comparable<T>>{
    T a[];

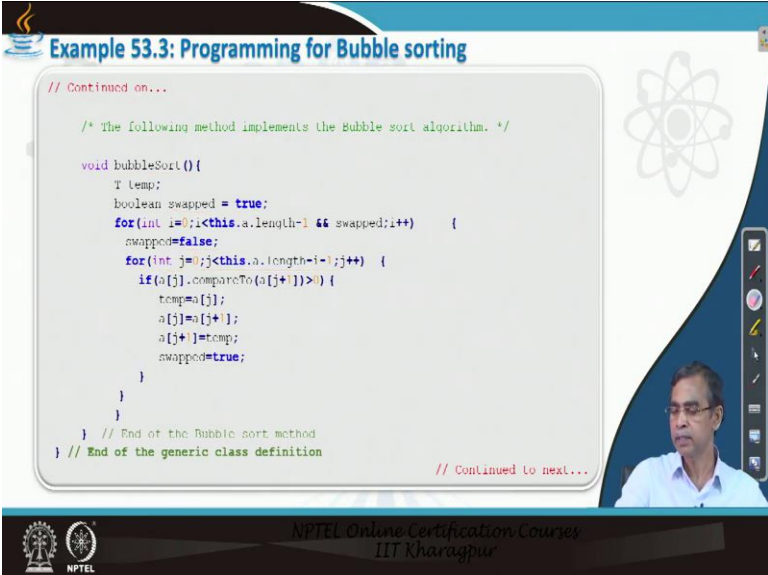
    GenericArraySorting(T x[]) { // Define a constructor
        a = x;
    }
    T getData(int i) { // To return the element stored in the i-th place
        return a[i];
    }
    void printData() { // A method to print the elements in array a
        for(int i = 0; i < a.length; i++)
            System.out.print(this.getData(i) + " "); // Print the i-th element
        System.out.println();
    }
} // Continued to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

So, we have implemented this version of this that means modified version of the bubble sort where we use the flag here. So, now let us see there is again generic implementation this implementation I have done in the same way in the previous one. So, we declare a class the same the array we declare it here and this basically we will just create a constructor this a little bit different way that basically same way we can do earlier also but it is basically we create the array, initialize the array and then read the data that is okay reading the data. The get data method you can call integer i and it basically return ai this basically if you want to print the data like.

So, in case in print data method you will call it the get data. And this is the print method so this a little bit in different way the algorithm you can follow the previous method also to define it. I just make it twist little bit. So, this basically defined the generic class and finally I will include the method here the method is the bubble sort.

(Refer Slide Time: 14:33)



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

```
// Continued on...

/* The following method implements the Bubble sort algorithm. */

void bubbleSort(){
    T temp;
    boolean swapped = true;
    for(int i=0; i<this.a.length-1 && swapped;i++) {
        swapped=false;
        for(int j=0; j<this.a.length-i-1; j++) {
            if(a[j].compareTo(a[j+1])>0) {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                swapped=true;
            }
        }
    } // End of the Bubble sort method
} // 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.

The method is declared here this are bubble sort is included and here we used flag this indicates that the flag is initially true that mean you have to continue the loop. But whenever we see that there is no interchange so you can make the swap false. And then it can here if we do not enter write false, then if we find any interchange you can make it true, initially it is true that we have to repeat it and we just enter the loop we can make it false.

If there is no thing, nothing occurs here when it remain false that means that your sorting is done all elements are already in sorted order. So, we have to quit the method quit the loop actually, then it over. So, this basically the bubble sort, the logic is basically the same way that we have discussed about that means we have to compare two adjacent elements. And then swap them if they are not in order that concept is basically followed.

(Refer Slide Time: 15:35)

```
// Continued on...  
  
/* The main method to test the program. */  
public class TestBubbleSort {  
    public static void main(String[] args) {  
        Integer[] i = {109, 44, 79, 74, 98};  
        // Store the data into generic array  
        GenericArraySorting<Integer> arrayInt = new GenericArraySorting<Integer>();  
        // Printing the data..  
        System.out.println("Array Before Sorting : ");  
        arrayInt.printData();  
        arrayInt.bubbleSort();  
        System.out.println("Sorted Array is : ");  
        arrayInt.printData();  
        System.out.println();  
    }  
} // End of the program
```

And this code is an implementation of the same logic here and this is the master program that you can consider in this master program just this is the one code that we have written there and this is the main method this a input list and this is the integer array created. Then we print the array, then call the bubble sort and then print the array. So, we can test the program again for the different arrangement of the elements in ascending order all same, in descending order random number will are set of numbers and then again try with different types integer, long, sort, float, string, even user defined also you can think about it. So, this these basically the idea that the bubble sort algorithm works for you.

(Refer Slide Time: 16:33)

Heap Sort Algorithm

NPTEL Online Certification Course
IIT Kharagpur

Now let us another sorting technique, the heap sort heap sort is bit lengthy procedure but again it is also good one and algorithm that we have a logic you have hope you have understood it.

(Refer Slide Time: 16:46)

Heap sort: Concept of heap tree

A

1	2	3	4	5	6	7	8	9	10
95	85	25	65	50	05	15	45	55	35

NPTEL Online Certification Course
IIT Kharagpur

Heap sort: Concept of sorting using heap tree

Create heap

- Create the initial heap tree with n elements stored in the array A .


For $i = n$ down to 1 do

Remove max

- Select the value in the root node (this is the maximum value in the heap). Swap the values (that is $A[1]$) and value at the i -th location in A .

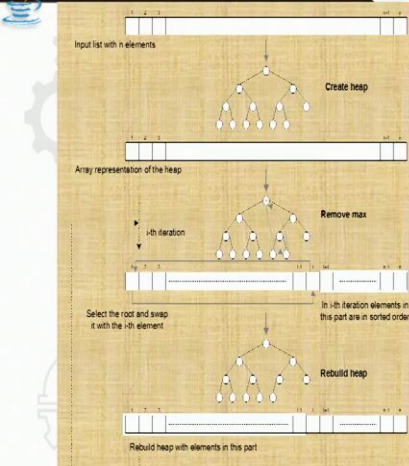
Rebuild heap

- Rebuild the heap tree for elements $A[1, 2, \dots, i-1]$.



Here array of integers needs to be stored as an input and this the procedure that we have to consider, first you have to create heap so on routine can be written for that, then remove max is basically deleting the root node.

(Refer Slide Time: 17:01)



Input list with n elements

Create heap

Array representation of the heap

i -th iteration


Remove max

Select the root and swap it with the i -th element

In i -th iteration elements in this part are in sorted order


Rebuild heap

Rebuild heap with elements in this part



Heap sort: Algorithm (Heap sort)

1. Read n elements and stored in the array A
2. **CreateHeap**(A) // Create the heap tree for the list of elements in A
3. **For** $i = n$ down to 2 **do** // Repeat $n-1$ times
4. **RemoveMax**(B, i) // Remove the element at the root and swap it with the i -th
5. **RebuildHeap**($B, i-1$) // Rebuild the heap with the elements $B[1, 2, \dots, (i-1)], i-1$
6. **EndFor**
7. **Stop**




And then rebuild the heap. So, this algorithm is given there, this is basically the main sorting method which basically call this are the sub method are there.

(Refer Slide Time: 17:10)

Heap sort: Algorithm (Create heap)

1. $i = 1$ // Initially, the heap tree (B) is empty and start with first element in A
2. **While** ($i < n$) **do** // Repeat for all elements in the array A
3. $x = A[i]$ // Select the i -th element from the list A
4. $B[i] = x$ // Add the element at the i -th place in the array B
5. $j = i$ // j is the current location of the element in B
6. **While** ($j > 1$) **do** // Continue until the root is checked
7. **If** $B[j] > B[j/2]$ **then** // It violates the heap (max) property
8. $temp = B[j]$ // Swap the elements
9. $B[j] = B[j/2]$
10. $B[j/2] = temp$
11. $j = j/2$ // Go to the parent node
12. **Else**
13. $j = 1$ // Satisfy heap property, terminates this inner loop
14. **EndIf**
15. **EndWhile**
16. $i = i + 1$ // Select the next element from the input list
17. **EndWhile**
18. **Stop**



And this is the create heap method.

(Refer Slide Time: 17:12)

Heap sort: Illustration (Remove max)

```
1. temp = B[i] // Swap the elements
2. B[j] = B[i]
3. B[i] = temp
4. Stop
```

NPTEL Online Certification Courses
IIT Kharagpur

And the remove max method is there.

(Refer Slide Time: 17:14)

Heap sort: Illustration (Rebuild heap)

```
1. If (i = 1) then
2. Exit // No rebuild with single element in the list
3. j = i // Else start with the root node
4. flag = TRUE // Rebuild is required
5. While (flag = TRUE) do
6. leftChild = 2*i, rightChild = 2*i+1
   /* Check if the right child is within the range of heap or not */
   /* Note: If right child is within the range then also left child
7. If (rightChild < i) then
8. /* Compare whether left or right child will move to up or not */
9. If (B[j] > B[leftChild] AND B[leftChild] > B[rightChild]) then
10. Swap(B[j], B[leftChild]) // Parent and left violate heap property
11. j = leftChild // Swap parent and left child
12. Else
13. If (B[j] > B[rightChild] AND B[rightChild] > B[leftChild]) then
14. Swap(B[j], B[rightChild]) // Parent and right violate heap property
15. j = rightChild // Swap parent and right child
16. Else // Move down to node at the next level
17. flag = FALSE // Heap property is not violated
18. EndIf
19. EndIf // Check if the left child is within the range of heap or not
20. Else
```

Continue...

NPTEL Online Certification Courses
IIT Kharagpur

Heap sort: Illustration (Rebuild heap)

```

21     If (leftChild < i) then
22         If (B[i] > B[leftChild]) then // Parent and left violate heap property
23             Swap(B[i], B[leftChild]) // Swap parent and left child
24             j = leftChild // Move down to node at the next level
25         Else // Heap property is not violated
26             flag = FALSE
27         EndIf
28     EndIf
29 EndIf
30 EndWhile
31 Stop

```

And rebuild heap method is there.

(Refer Slide Time: 17:17)

Example 53.4: Programming for Heap sorting (defining generic class)

```

/* This program defines a generic class to store a collection. */
public class MinHeap<T extends Comparable<T>> {
    private T[] Heap;
    private int size;
    private int maxsize;

    private static final int FRONT = 0;

    public MinHeap(T[] arr , T node)
    {
        this.maxsize = arr.length;
        this.size = 0;
        Heap = arr;
        Heap[0] = node;
    }
}
// Continued to next...

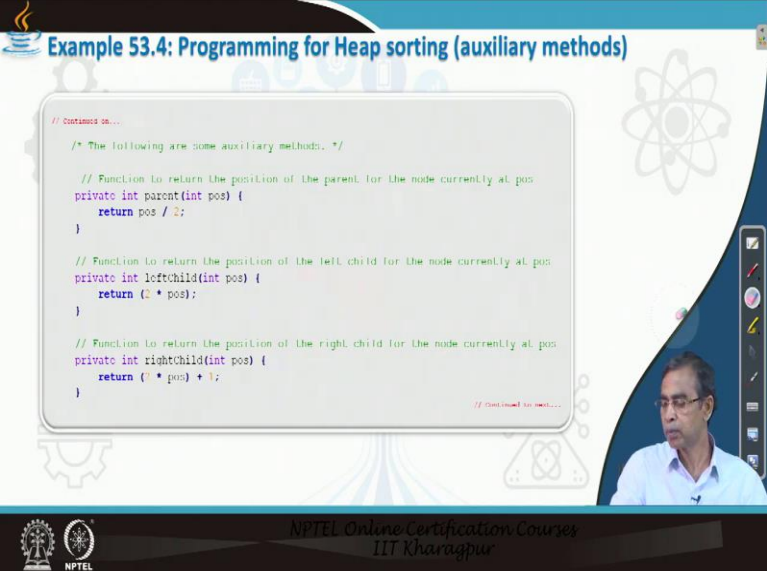
```

Now here is a programming you can just take based on the concept that we have discussed this is the programming. So, programming again generic programming as you can see here this is the declaration of generic class. This is a few fields we have declared, one is basically array where the heap will store so this is basically data. This are total number of elements the maximum size of the array actually that is fine.

And this are is the location is a point of front 0. Now we are considering min heap the logic that we have given is the max heap, again you can rewrite this code or rewrite this code so that it can

work for the max heap. I have intension give you that mean heap because in many of the book only max heap is discussed. Now let us see the mean heap and how it works so this basically the logic that you can think about how it works actually and that basically.

(Refer Slide Time: 18:20)



The slide displays the following C++ code for auxiliary methods in heap sorting:

```
// Continued on...  
  
/* The following are some auxiliary methods. */  
  
// Function to return the position of the parent for the node currently at pos  
private int parent(int pos) {  
    return pos / 2;  
}  
  
// Function to return the position of the left child for the node currently at pos  
private int leftChild(int pos) {  
    return (2 * pos);  
}  
  
// Function to return the position of the right child for the node currently at pos  
private int rightChild(int pos) {  
    return (2 * pos) + 1;  
}  
  
// continued to next...
```

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

This is a next part of the code here is the next part of the code the parent, this are auxiliary code left child, right child it is required because the next part of the code that basically required which is the parent which is the left child which is the right child is basically we are storing the heap tree in the from front array as you have to move from any node to the lowest I mean towards the children node or sometimes we have from the children node to each parent node so this kind of method can be call for them and here is the logic.

(Refer Slide Time: 18:55)

Example 53.4: Programming for Heap sorting (auxiliary methods)

```
// Continued on...  
/* The following are some auxiliary methods. */  
  
// function that returns true if the passed node is a leaf node  
private boolean isLeaf(int pos) {  
    if (pos >= (size / 2) && pos <= size) {  
        return true;  
    }  
    return false;  
}  
  
// function to swap two nodes of the heap  
private void swap(int lpos, int rpos) {  
    int tmp;  
    tmp = heap[lpos];  
    heap[lpos] = heap[rpos];  
    heap[rpos] = tmp;  
}  
  
// Continued to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

And here is the program that is the next part of the program that you can think about and there are few more code the is leaf and then swap this a very simple code is a very logic is so simple. You will be able to follow it.

(Refer Slide Time: 19:10)

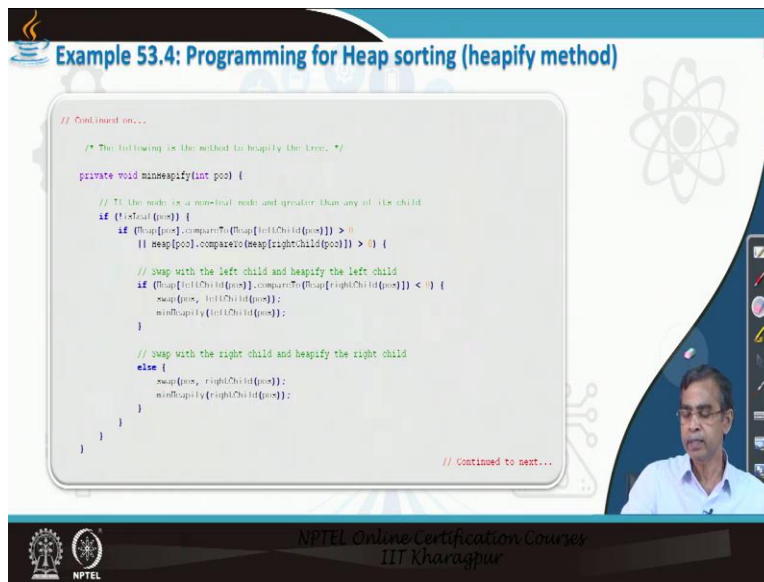
Example 53.4: Programming for Heap sorting (auxiliary methods)

```
// Continued on...  
/* The following is the method to print a heap tree. */  
  
public void print() {  
    for (int i = 1; i <= size / 2; i++) {  
        System.out.print(" PARENT : " + heap[i] + "  
            + " LEFT CHILD : " + heap[2 * i] + "  
            + " RIGHT CHILD : " + heap[2 * i + 1] );  
        System.out.println();  
    }  
}  
  
// Function to build the min heap using the minHeapify  
public void minHeap() {  
    for (int pos = (size / 2); pos >= 1; pos--) {  
        minHeapify(pos);  
    }  
}  
  
// Continued to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

And here is the this is the print method how to print the heap, it is just printing an array that is all nothing else. And this is the main code the mean heap, mean heap is basically just see the minheapify for all elements so removing heap and then minheapigying is there and so this basically the code works like that way.

(Refer Slide Time: 19:32)



Example 53.4: Programming for Heap sorting (heapify method)

```
// Continued on...  
  
/* The following is the method to heapify the tree. */  
  
private void minHeapify(int pool) {  
    // If the node is a non-leaf node and greater than any of its child  
    if (!isLeaf(pool)) {  
        if (Heap[pool].compareTo(Heap[leftChild(pool)]) > 0  
            || Heap[pool].compareTo(Heap[rightChild(pool)]) > 0) {  
            // swap with the left child and heapify the left child  
            if (Heap[leftChild(pool)].compareTo(Heap[rightChild(pool)]) < 0) {  
                swap(pool, leftChild(pool));  
                minHeapify(leftChild(pool));  
            }  
            // swap with the right child and heapify the right child  
            else {  
                swap(pool, rightChild(pool));  
                minHeapify(rightChild(pool));  
            }  
        }  
    }  
}  
  
// Continued to next...
```

NPTEL Online Certification Course
IIT Kharagpur

And here is the detail scored about how the minheapify the routine is defined so miheapify routine is basically the it basically called the heap tree and then it compare the elements and then again call the minheapify that means it is basically rebuild t is there and if require there is swap is there basically the logic that we have defined. So, this logic is basically a programming. So, it is programming according to my style programming according to your own style something else you can follow the algorithm better and then write the code of your own.

Now here compareTo method we are using so you can just overwrite the compareTo method depending on how you want to compare say for new student. So, compareTo method can overwrite there and student maybe one numeric field can be used to compare the two objects in order so that automatically take care. So, you can take the previous discussion where I overwrite the compareTo method for some student, the user defined object you can follow it here you can add the code there and then you can run the programming again so that test it.

(Refer Slide Time: 20:38)

Example 53.4: Programming for Heap sorting (insert method)

```
// Continued on...  
/* The following is the method to insert a node into the heap tree. */  
  
public void insert(T element) {  
    if (size >= maxSize) {  
        return;  
    }  
    Heap[++size] = element;  
    int current = size;  
  
    while (Heap[current].compareTo(Heap[parent(current)]) < 0) {  
        swap(current, parent(current));  
        current = parent(current);  
    }  
}  
  
// Continued to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

And so this is basically building the tree inside there.

(Refer Slide Time: 20:43)

Example 53.4: Programming for Heap sorting (remove and sort methods)

```
// Continued on...  
/* function to remove and return the minimum  
element from the heap */  
  
//  
public T remove() {  
    //System.out.println(size);  
    T popped = Heap[FRONT];  
    Heap[FRONT] = Heap[size--];  
    minHeapify(FRONT);  
    return popped;  
}  
  
// Continued to next...
```

```
// Continued on...  
/* Sorting with the same heap. */  
public T[] sort(T sorted[]) {  
    //System.out.println(maxSize);  
    int i = 0;  
    while (size > 0) {  
        T a = remove();  
        sorted[i] = a;  
        i++;  
    }  
  
    for (int j = 0; j < i; j++) {  
        System.out.println(sorted[j] + " ");  
    }  
    return sorted;  
}  
  
// Continued to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

And remove code and all these things are there and this is the main method that is the sorting method which basically remove the root node and then heapify it is there. Now this code is basically in a fragmented way written in a basically. So, modular way the we have written the code but again I am telling the, the code is choice of your own.

(Refer Slide Time: 21:09)

Example 53.4: Programming for Heap sorting (main method)

```
// Continued on...  
/* The main method to test the program. */  
public static void main(String[] args)  
{  
    System.out.println("The Min Heap is ");  
    Integer[] A = new Integer[10];  
    MinHeap minHeap = new MinHeap(A, 5);  
    //minHeap.insert(5);  
    minHeap.insert(3);  
    minHeap.insert(17);  
    minHeap.insert(10);  
    minHeap.insert(9);  
    minHeap.insert(0);  
    minHeap.insert(4);  
    minHeap.insert(7);  
    minHeap.insert(9);  
    minHeap.minHeap();  
  
    minHeap.print();  
}  
// end of the program
```

NPTEL Online Certification Course
IIT Kharagpur

So, you can write the code according to your own choice. And here is the master program this program basically see here what we are doing, we are defining the main method and this is basically we call the routine here that means we have to create the minHeap of say size say 5 here. And then we consider an array of maximum size is 15 then what we are doing is that we are storing we are inserting elements so this basically build the heap this basically building a heap starting from the empty heap and initially inserting one element at a time so initially when we create the node with array a.

So, basically we insert 5 and then we are inserting each element one by one. So, these are the elements are inserted so the heap contains all this element 5 to 5, 3, 17, 10 in random order so heap is created and then we just after create on heap we call the mean heap that basically the heap sort method actually. And then if we print the heap it will basically print the elements in the sorted order. So, this basically the idea about here the we store the array in the form of a minHeap actually.

So, this is the code that you can follow it may take some time to understand the code but anyway so you can run the code for different type of data. Here we have given the demonstration for integer type you can took the running with long, double, sort whatever it is there and it will work.

(Refer Slide Time: 22:54)

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 purple book cover for 'Classic Data Structures, Second Edition' by Debasis Samanta, published by Prentice Hall. The slide also includes a vertical toolbar on the right side with various icons and a bottom navigation bar.

And for many other discussions you can follow this book so that you can try to implement some other sorting techniques those are there in this book to practice more. In our next part we will discuss few more techniques there so that you can follow it, thank you.