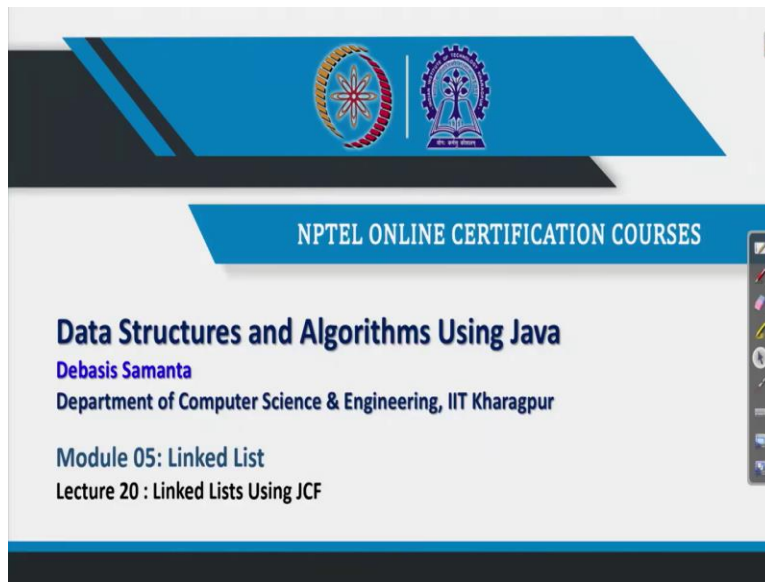


**Data Structures and Algorithms using Java**  
**Professor Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture 20 - Linked Lists using JCF**

(Refer Slide Time: 00:46)



Programming for linked list using Java language, we have experienced that we can do many things of our own. Now, today let us see as an alternative way how Java can support as a built-in things where many things are available readily and you just simply plug and play. Now so today we will cover about the application of linked list in your program using Java Collection Framework.

(Refer Slide Time: 01:02)

**CONCEPTS COVERED**

- Class LinkedList in JCF
- Defining Linked List
- Creation of Linked List
- Operations
  - Insertion
  - Deletion
  - Traversals

The slide features a blue header with the title 'CONCEPTS COVERED'. Below the header, there is a list of topics. To the right of the list is a photograph of a blue and white train traveling on a track through a green, hilly landscape. The slide also includes logos of IIT Kharagpur and NPTEL in the top left corner.

Now today's topic include, first we will learn about where exactly which class or package in which portion of the package that you can use to implement linked list, and with this package how you can create your linked list, and how the different operations, mainly insertion, deletion, and traversals can be accomplished. So, this is our today's plan.

(Refer Slide Time: 01:30)

**Linked List Class**

- The `LinkedList` class is a member in the `Java Collections Framework` to support **sequential access** of a list of items unlike `ArrayList`, which provides indexed access.
- It inherits the `AbstractSequentialList` class and implements the `List` and `Deque` interfaces.

The class implementation hierarchy of `LinkedList` class

```
graph TD; Collection[Collection] --> List[List]; Collection --> Queue[Queue]; AbstractSequentialList[AbstractSequentialList] -.-> List; AbstractSequentialList -.-> Dequeue[Dequeue]; LinkedList[LinkedList] -.-> AbstractSequentialList; LinkedList -.-> Dequeue;
```

The slide features a blue header with the title 'Linked List Class'. Below the header, there are two bullet points explaining the class's role and inheritance. A class hierarchy diagram is shown below the text, illustrating the relationships between `Collection`, `List`, `Queue`, `AbstractSequentialList`, `Dequeue`, and `LinkedList`. The slide also includes logos of IIT Kharagpur and NPTEL in the top left corner and the text 'NPTEL Online Certification Courses IIT Kharagpur' at the bottom.

Now `LinkedList`, there is a class. There is a class in the `Java Collection Framework` which basically supports whatever the linked list theoretically you can give. So actually, there linked list has been implemented in a very nice and efficient way. And you can see there is a hierarchy

of the class, how it is there. As you know, in the Java Collection Framework there is a collection class which basically is an abstract class sort of thing, which basically design what are the different methods that can be, can be there in any subclass of this collection class. Now, as we can see, this collection class is basically is a class where there is, there are two interfaces namely, list and queue.

So, list and queue is basically extend the collection class. And there is another interface deque also, deque extends the queue class. Deque is an interface which extends queue. Now, there is another class, it is an abstract class called the AbstractSequentialList. The LinkedList is a subclass of AbstractSequentialList and which implements deque. So, by the process actually if you see, LinkedList is an implementation of all the methods which are defined either in collection or list, queue, or deque or AbstractSequentialList.

These are basically the class, these are the classes or interfaces which has its own methods and fields and everything and all those methods are basically the abstract methods, we can say. Ultimately, all these abstract methods are implemented in LinkedList class. And this LinkedList class basically is a support for you to implement or realize linked list related different activities. So, today we shall start about what exactly the contents in linked list, what are the different methods that can be there and fields and everything.

(Refer Slide Time: 03:38)

**Features of linked list class**

Following are the few salient features of this collection:

- It provides a linked-list data structure
- The class can **contain duplicate** elements.
- The class uses a **doubly linked list** to store the elements.
- The class maintains **insertion order**.
- The class is **non-synchronized**.
- Manipulation is **faster than array** because no shifting is required.
- The **LinkedList** class can be used to maintain a collection as a **linked list, stack** or **queue**.

NPTEL Online Certification Course | IIT Kharagpur

Okay, so this is basically summary about the linked list class. As I told you, linked list class is mainly to cater the need of linked list data structure, and this essentially is basically is a double linked list actually. So, because double linked list can serve the purpose of both the list and both way direction, movement direction. That is why the developer has planned it as a double linked list only. So, it is not a single linked list.

Now, as that linked list like an array, it can contain duplicate elements. Linked list can be grow in the same order as in the insertion order. So it is basically, if you add, it will add at the end of the list actually. So, insertion order can be maintained. So, if you add in the order 1, 2, 3, 1 is the first element, second element is 2, and the last element is 3, in that order. So, it maintains insertion order. And one important observation or I mean point regarding the linked list is a it is non-synchronized.

That means if you use this structure in your multi-threading programming, then you may not get the synchronized manner, so it is (not) non-synchronized. And, this is the one limitation. But it is actually not a limitation, to improve the speed up and everything, Java developer carefully thought about that, it should not be non-synchronized. If you want the synchronized data structure then only the vector class you have to do it.

Otherwise, the linked list implementation that we have discussed and programming, the customized programming or programming in Java, you can do and then is a synchronized programming you can achieve. And that is obviously, programming of your own have the added advantage in this regard. Manipulation is faster compared to array because this linked list is planned in such a way the insertion at any point or deletion from any position is faster than it is in case of array. And LinkedList also, this class purpose, many other purpose to serve many other data structure like stack, queue, deque.

And you know in the recent version of the Java, there is no class like stack but we have to rely on only the legacy class where the stack class is defined, otherwise, queue and deque, if you want to have. These are the other data structure that we will learn shortly after a few lecture in this continuation of the module. We will discuss about stack and queue, there we will learn about these other data structure. Now, if you want to implement all these data structures, then only the linked list is the solution. There is no other class that is there to implement this.

(Refer Slide Time: 06:26)

**Creating Linked List**

NPTEL Online Certification Courses  
IIT Kharagpur

**Constructors in the LinkedList class**

- The `LinkedList` class consists of two constructors, which are as follows:

Constructor	Description
<code>LinkedList ()</code>	It is used to create an empty list.
<code>LinkedList(Collection&lt;? extends E&gt; c)</code>	It is used to construct a list containing the elements of the specified collection. The ordering of the element in the list and collection is same.

NPTEL Online Certification Courses  
IIT Kharagpur

Now, let us start first how you can create a linked list with the help of Java Collection Framework or in other word, the `LinkedList` class that is there in JCF. Now, to create a linked list, the `LinkedList` class has two constructors. One is a default constructors, it creates an empty list. And, another constructor is there which can create a linked list if you pass an argument, a collection. A collection can be an array list or is an array or any other list of objects like any other collection. So these are the two constructors that can be used to create a linked list using Java Collection Framework.

(Refer Slide Time: 07:18)

**Example 20.1: Create a linked list**

```
// This program define a linked list and add some items into it

import java.util.*;
public class CreateLLandAddItems {
    public static void main(String args[]) {
        // Creating an empty ll of class LinkedList
        LinkedList<String> ll = new LinkedList<String>();
        // Adding elements to the linked list using a number of add methods
        ll.add("Mumbai");
        ll.add("Chennai");
        ll.add("Kolkata");
        ll.add("Delhi");
        ll.add("Bangalore");
        ll.add("Gurgaon");
        ll.add("Hyderabad");

        // System.out.println("Linked list : "+ ll); // Simple printing
        // Printing the list using an iterator
        Iterator<String> itr = ll.iterator();
        while(itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

Now, let us see how a program you can think about using this LinkedList class that is there in Java Collections Framework. Now first of all, you should import java dot util dot star because this is the location of class where linked list is defined. If you do not import, definitely your program will not run successfully, it will basically give you the compile time error because they will not be able to resolve many methods that is there.

Now let us come to the main method which we are going to discuss. This program basically demonstrate how a list can be created and how some elements can be added into the list. Obviously we, we shall use the LinkedList class in this case. Now, the first step that you have to follow is that you have to just create an object of type linked list. And as the linked list is a generic class, so you have to tell is the collection of what type. Now, so this is basically LinkedList, this is the class for which we want to create an object of this type.

And, this basically the template of the type that we want to use. That means we want to create a linked list of type string, that means it will store string elements. And so this is basically the constructor call string and this is a default constructor we have called. That means initially with this we will be able to create a linked list the name of this linked list is ll, which is initially empty. Now, once the list is created, we (sh), we will be able to add elements into it.

In order to add element, there are many methods we shall discuss, but one universal method that is declared there in a collection class is called add. So, you can, we can use the add method. This

method is basically defined fully how an element can be added into a linked list following the insertion order. So, that method is already implemented for you, you can just call. You can recall how we can implement method of our own and you can call in a master program or driver program because the same thing, we are writing the driver program only because no need to define methods, no need to define class for linked list or node structure.

Nothing is there here, or everything is implicit. Now, so here you can see next few statement which you have mentioned here to add few string into the linked list namely, Mumbai, Chennai, Kolkata, etcetera. So, list is now loaded. Once the list is loaded, we can print the list. So there are many ways the list can be printed. One simple most way that you can print the list is system dot out dot print ln and then ll, ll means list. It will automatically print all the elements which are there in the list.

Otherwise, we can use one another method which is many programmer prefers is called the iterator. So, you have to define an iterator objects and these iterator objects, itr. We define that this iterator data object to iterate a string collection. And so, this is a method that is defined in the iterator class. And then this is a statement by which iterator will scan each element in the list and print it. So, this basically printing each element and it will check that whether the collection is (fini) exhausted or not.

So hasNext is basically check whether the next element is present or not, if not present, it will quit the loop. So this is basically using iterator the code you have to use all the time. So this once you are habituated this code, it become very easy. So, this is called the iterator. This is the one way to traverse a linked list. And here we have used, during the process of traversal only printing but you can use many thing. Like, so you can count what is the length of each string is or whether it is a palindrome or not.

So, you can call some methods, define some method of your own and can add it. So, this is a very simple one demonstrable program that, that is for just only an illustration and with this illustration therefore, I hope you have to, you are able to understand about how a linked list can be created and how the same list can be loaded with some elements of your own. So, this is a simple way that we can do it.

(Refer Slide Time: 11:41)

**Example 20.2: Create a linked list with user defined objects**

```
/* This program shows how to create an array of objects of type Students and then add the array
into a linked list and then printing the same. */

import java.util.*;
//Declaration of a user defined class
class Student {
    String name;
    double marks;
    // Constructor
    Student(String s, double m) {
        name = s;
        marks = m;
    }
    void printdata () { // to print a record
        System.out.print("Name : " + name);
        System.out.println(" Marks : " + marks);
    }
}

// Continued to next ...
```

Now, there are many more things in the store. Now, let us see how a linked list can be maintained using user-defined object. User-defined object means there are some standard object like integer or string, double, float. These are the classes are there, we can create a collection of all these objects of this type. Now, in this example, we shall study about how our defined class and according to this class a number of objects can be stored in a linked list. Here, let us consider a program which basically define a class called Student.

This is our own defined class, we are defining a class. This class has two fields, one is called name and another is called marks. Name is of type string and marks is of type double. And, this is a constructor to create an instance of this class. So, it is basically if you want to create an object, you have to pass what is the name and what is the marks, and accordingly object will be created. And this is another method we have declared which basically print an object.

So, it will print the name and the marks. So, this includes the definition of a user-defined class called, Student. Now, what we want to do is that, we want to create a number of Student's object and all those Student objects we want to store into linked list. So, this is basically our objective. Now for this, let us see what is the program that you can do it. So the program is that, we have to create a linked list of type student, and then we can go on adding and then finally if you want to print you can print, that is all.



(Refer Slide Time: 13:29)

**Example 20.2: Create a linked list with user defined objects**


```
// Continued on...

// The main class is defined below.
public class CreateLofCollection {
    public static void main(String args[]) {

        // Create an array of objects
        Student sArray[] = new Student[5]; // To store 5 objects

        // Create the array sArray
        sArray[0] = new Student("Ram", 79.6);
        sArray[1] = new Student("Rahim", 85.5);
        sArray[2] = new Student("John", 90.1);
        sArray[3] = new Student("Lisa", 69.4);
        sArray[4] = new Student("Ana", 59.8);

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




**Example 20.2: Create a linked list with user defined objects**

```
// Continued on...

// Creating a linked-list with sArray collection
LinkedList<Student> ll = new LinkedList<Student>(Arrays.asList(sArray));

Student temp;
// Printing the list using an iterator
Iterator<Student> itr = ll.iterator();
while(itr.hasNext()){
    temp = itr.next();
    temp.printData(); // Print the current record.
}
}
```



So this is basically the program, you can check it. This is the one simple program which basically create a number of Student's object and all the Student object will be stored in a linked list. Now to do these things, first this is the main method, and here we first declare an array. This array basically is declared to store the number of Student's objects. So here we have declared an array of size 5. That means we can accommodate maximum 5 Student's object in this array.

Now here, once the array object is defined, we just okay insert the 5 different Student's objects into the array, starting from 0 to 4. So in the zeroth location of this array, we store one object,

Ram is the name and marks is 79 point 6 and so on, so on. So, five different objects are created and those objects are stored in the array. The name of the array is sArray. Now our next objective is basically using this sArray. That means we want to put all the objects which are there in sArray into a linked list.

Now for this things, we should refer to the second constructor which basically allow to, create a linked list passing a collection as an argument to the linked list. Now, so the next part of the program, so this is basically in continuation of this, next part of the program which basically looks like this. You just check it. We create a LinkedList of type Student. And this basically you see when we create this list, we pass the collection. Now, this collection is basically sArray, so we collection as an array list.

So, there is a method that is defining class arrays that you have already know that arrays, array list and everything. And for this arrays class, there is a asList. That means, this array will be considered an asList and then this is basically a collection of arrays. Now, we pass this collection into this constructor, this constructor. Then this constructor will automatically load all the nodes into the list with the objects that is there in sArray. So, that means that in this case, the linked list will in contain 5 nodes and in these nodes the objects namely Ram, Rahim, John and everything will be stored there.

So, this way you can see how easily and how very efficiently you can create a linked list having an, having a collection already available. Now, after these things, this is basically the code. This code is basically traversing the linked lists that you just have created. And this basically, the way that I can create an iterator, this iterator of type Student objects. So iterator object is created for this linked list, so ll dot iterator and hasNext. And then it basically read an object from the list. That object is temporarily stored in temp.

And we finally call the printData method. printData method basically for an object how we can print. Alternatively, without these things also we could write system dot out dot print ln and then temp, that also you can do it, so that it will print also. But it is not the good idea to how to print a particular object. But this is a nice idea that how the using the printData method of the objects itself and we can print it.

Anyway, so this is an example by which you can understand that how a linked list can be created. That linked list not necessarily for a particular type, it can be of any type, and then the linked list can be manipulated. Manipulation whenever, manipulation here, so for the manipulation we only create a LinkedList and then print it. But other manipulation are there, it is basically regarding insertion, deletion and other operations.

(Refer Slide Time: 17:39)

The image displays two sequential slides from a video lecture. The top slide is titled "Operations with LinkedList" and features a coffee cup icon with steam rising from it. The bottom slide is titled "Operations with LinkedList collection" and lists the following operations:

- The following operations with LinkedList are usually frequently.
  - Creating a linked-list
  - Insertion of an item in a linked-list
  - Deletion of an item from a linked-list
  - Traversing a linked-list

Both slides include the NPTEL Online Certification Courses IIT Kharagpur logo and a video feed of the presenter in the bottom right corner.

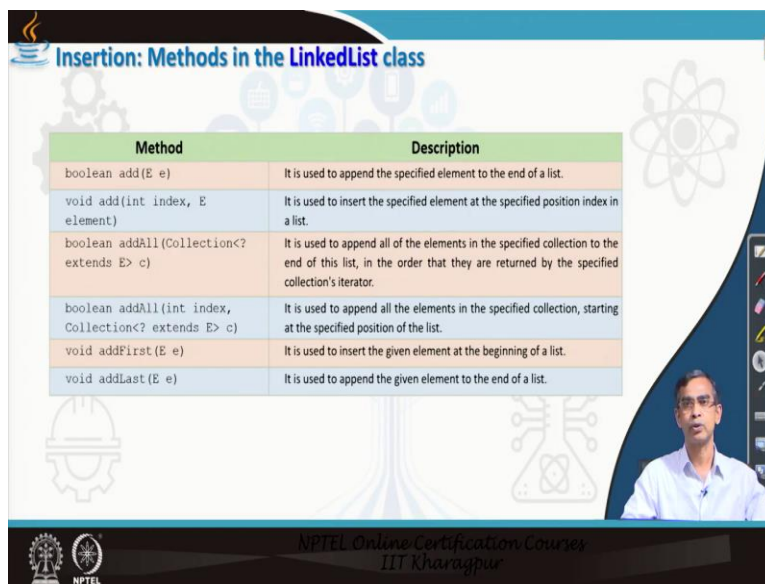
Now, let us see how the different operations, we the LinkedList class that is there in JCF can be done. So, the operation basically that we are going to discuss, traversing we have learned about

little bit. Mainly we will focus about insertion and deletion related operation. And there are some other miscellaneous operations are there, that also try to cover.

(Refer Slide Time: 18:05)



The slide features a central title "Insertion into LinkedList" in blue text. To the left is an icon of a coffee cup with steam. The background is a light blue tree-like structure with various icons in its branches. A small video inset of a man in a white shirt is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".



The slide is titled "Insertion: Methods in the LinkedList class". It contains a table with two columns: "Method" and "Description".

Method	Description
<code>boolean add(E e)</code>	It is used to append the specified element to the end of a list.
<code>void add(int index, E element)</code>	It is used to insert the specified element at the specified position index in a list.
<code>boolean addAll(Collection&lt;? extends E&gt; c)</code>	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
<code>boolean addAll(int index, Collection&lt;? extends E&gt; c)</code>	It is used to append all the elements in the specified collection, starting at the specified position of the list.
<code>void addFirst(E e)</code>	It is used to insert the given element at the beginning of a list.
<code>void addLast(E e)</code>	It is used to append the given element to the end of a list.

A small video inset of a man in a white shirt is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

Now, let us first consider how the insertion can be done in the LinkedList. Now, in order to this insertion, LinkedList stores many methods for the purpose. Now, we have already experienced with one add method. But add is not the only method, there are some other methods also. Now, whatever the other method which are there, which are defined there in the LinkedList class is listed in this table. Now, you can see the add method, already we are familiar to. So add method is there.

Now, add method has another form also. In the first form, add and then element, which element you want to add into the list. Now, the next method, add at a particular location in the linked list. So, it is basically insert at, at a given position, so index you can specify. So 10th index or 5 index or whatever it is there; and there is, okay. So, this is basically mentioning the index in the, in the LinkedList, you can add an element which will be passed here. And, then add all. So, add all basically if you want to add a collection into an existing linked list so it will add, it is just like merging.

So, collection is basically is another list you can say, not necessarily in the linked list form. It can be linked list itself, because linked list is also a collection. So, these collections imply any other linked list or other collection like array list, arrays or something else so or vector, whatever it is there. So, any collection can be considered here as an argument and if you pass this value or object as an argument, then it will add. So, it is essentially merging operation you can see. Now, here the addAll has another version. It basically, you can add a set of elements into the linked list at a particular location.

So here in this method the addAll will add at the end, but you can add at any locations in the linked list. Now, so, these are the basically bulk operation you can say, add as a group, right. Now, there are again addFirst and addLast, it is just like insert at front and insert end like. Insert front, insert end we have already experienced. So it is basically in JCF, they are called addFirst and addLast element. So, these are the basically methods which are there in, in LinkedList class to help you to insert elements into the list.

(Refer Slide Time: 20:43)


### Example 20.3: Insertion at different locations

```
/*This program shows how items can be inserted at different locations in a linked list. For
this purpose, there are methods like add(), addFirst(), addLast() are defined in the LinkedList
class. */

import java.util.*;

public class LLinsertionDemo {
    public static void main(String args[]) {
        // Creating an empty ll of class LinkedList
        LinkedList<String> ll = new LinkedList<String>();
        // Adding elements to the linked list using a number of add methods
        ll.add("Mumbai"); // Add an initial item
        ll.add("Chennai"); // Add another item
        ll.addLast("Kolkata"); // Add at the end
        ll.addFirst("Delhi"); // Add at the front
        ll.add("Bangalore"); // Add in the specific location
        ll.add("Guwahati"); // Sequential add goes at the end
        ll.add("Hyderabad"); // Another sequential insertion


        // Continued to next -
    }
}
```



### Example 20.3: Insertion at different locations

```
// Continued to next -

// Printing the list using an iterator
Iterator<String> itr = ll.iterator();
while (itr.hasNext()) {
    System.out.println(itr.next());
}
// Alternatively:
System.out.println("Linked list : " + ll); // Simple printing
}
```



Now, let us have some example which basically okay we can see how the different methods can be utilized in a program. So, this is one example. I gave the name as a linked list insertion demo. So, this program tell you, how the different way you can add elements into a linked list. So, first what we should do? We should create a linked list, right. And here in this example, we created a linked list of type string, name of the linked list is ll in this case.

Now, here we can call many methods which are defined there in LinkedList class and which we have just now learnt. For example, add method, add is an object, addLast you can understand. So, this will add at end. addFisrt, so Delhi will add before Mumbai, Chennai, Kolkata, like. Add

2, this basically tell that it will be added into the 2nd location, so 0,1, 2nd. That is the 3rd location we can say and like this. So there are few more things also we can go on.

Now, so there are few more lines in this code. Now okay, so we have done the adding and after adding you can, you may be interested to see exactly how your linked list look like. There is a simple print or printing the list or traversing the linked list, again using the iterator and you can see. Alternatively, as I said that system dot out dot print ln linked list ll, you can see. This is a very compact one, that way also you can print. But you can find the two printing in a different ways, but you can see.

You can just have an experience about how the two print work for you, one using iterator and one using simple print ln as the entire list also. Now anyway, so our objective was not to print actually, our objective is to how the different way the element can be insert, insert into a linked list. We have experienced few, not addAll and everything. That again you can think for writing programming, you can create a linked list and then pass an argument as an linked list. addAll can be done at any specific location. All these things you can try of your own. Practice, right and you can check that how the different methods are working for you.

(Refer Slide Time: 23:06)

**Example 20.4: Insertion of a list into a LinkedList**

```
/* A sub list can be inserted into a linked list in addition to a single item. This program
shows how a sub list can be inserted at different locations in a linked list. For this
purpose, the addAll() method is used.*/

import java.util.*;

public class InsertSubListToLL{
    public static void main (String args[]){
        LinkedList<String> l11 = new LinkedList<String>();
        System.out.println("Initial list of elements: " + l11);
        l11.add("MP Allahabad");
        l11.add("MP Lucknow");
        l11.add("MP Varanasi");
        System.out.println("Initial list: " + l11);

        // Create another linked list, say l12
        LinkedList<String> l12 = new LinkedList<String>();
        l12.add("MLA Nadia");
        l12.add("MLA Kharagpur");

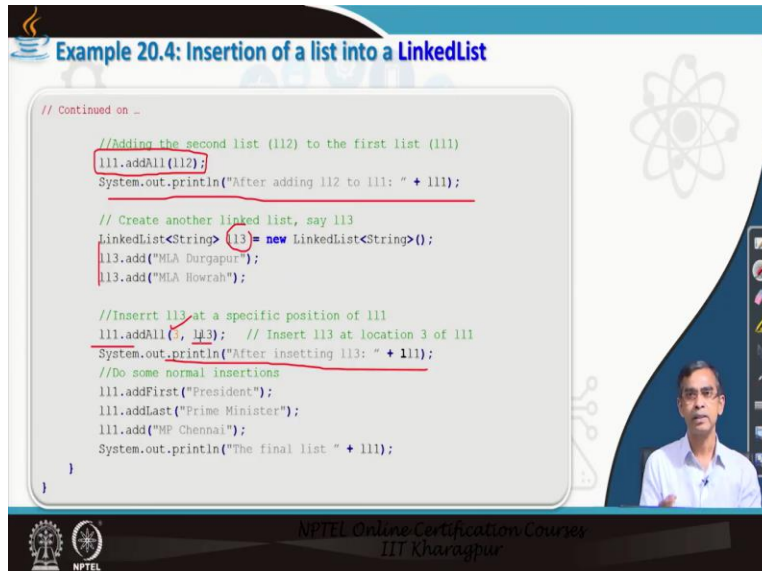
        // Continued to next ...
    }
}
```

NPTEL Online Certification Courses  
IIT Kharagpur



### Example 20.4: Insertion of a list into a LinkedList

```
// Continued on ...  
  
//Adding the second list (l12) to the first list (l11)  
l11.addAll(l12);  
System.out.println("After adding l12 to l11: " + l11);  
  
// Create another linked list, say l13  
LinkedList<String> l13 = new LinkedList<String>();  
l13.add("MLA Durgapur");  
l13.add("MLA Howrah");  
  
//Insert l13 at a specific position of l11  
l11.addAll(3, l13); // Insert l13 at location 3 of l11  
System.out.println("After inserting l13: " + l11);  
  
//Do some normal insertions  
l11.addFirst("President");  
l11.addLast("Prime Minister");  
l11.add("MP Chennai");  
System.out.println("The final list " + l11);  
}
```



NPTEL Online Certification Courses  
IIT Kharagpur

Okay, this is an example actually. I made it ready for you, how a list can be added. It is just like merging sort of thing. Now, let us, okay consider this program first. So, here we create one LinkedList of type string and the linked list is l11. And we add some elements into the LinkedList where 3 nodes are inserted into the LinkedList of like this. Next, we create another LinkedList. Let the name of the LinkedList is l12 and we add few elements into the list. Okay, so two linked lists are created.

Now, we can do many things here. We can merge, l12 can be merged after l11 or vice versa. Or l12 can be merged in l11 in a specific location, addAll is mentioning where you want to add. That, that kind of things are there. Now, let us see the next of the programming, how we can do it? Now, here you can see l11 add l12. It is basically you see equivalent to merge, l11 merge l12 so l12 list will be merged after the list l11. So l12 will be appended after l11 like. And, after merging you can print.

Now, here create another LinkedList, say l13. Now, this linked list again of type string, so we can create another linked list, we can add some element into it. Now, here insert l13 at a specific position of l11. l11 right now, this is basically merging of l11, l12 both. Now we want to insert this l13 at the index of 3, that means the 4th location of the list l11, and after adding at a particular specific location, if you can print you can see how linked list grows automatically.

Now, after the LinkedList, so this way LinkedList is basically expanding, and this is basically the maintenance of a LinkedList by performing several insertion operation. And then you, you can



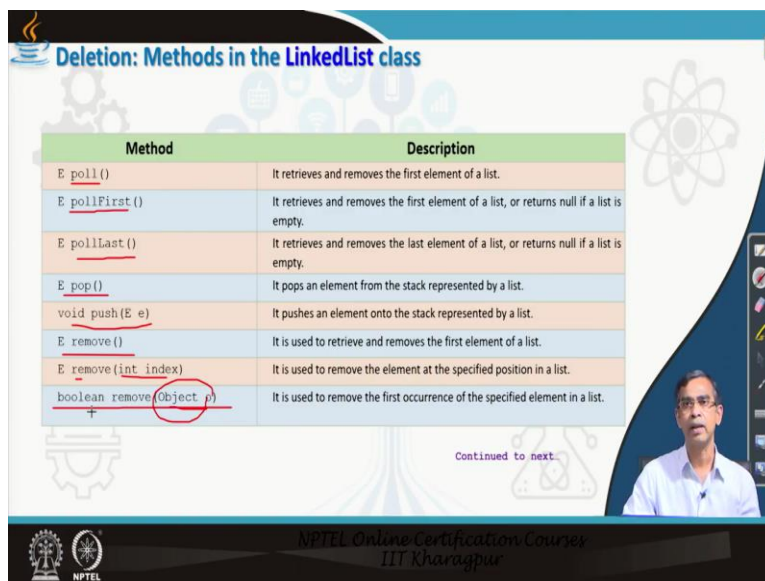
can add or delete or you can perform any operations on to the ongoing linked list that you are processing. So, this is the idea about how the insertion as a whole can be achieved using LinkedList class in Java Collection Framework.

(Refer Slide Time: 25:33)



## Deletion from LinkedList

NPTEL Online Certification Courses  
IIT Kharagpur



## Deletion: Methods in the LinkedList class

Method	Description
<u>E poll ()</u>	It retrieves and removes the first element of a list.
<u>E pollFirst ()</u>	It retrieves and removes the first element of a list, or returns null if a list is empty.
<u>E pollLast ()</u>	It retrieves and removes the last element of a list, or returns null if a list is empty.
<u>E pop ()</u>	It pops an element from the stack represented by a list.
<u>void push (E e)</u>	It pushes an element onto the stack represented by a list.
<u>E remove ()</u>	It is used to retrieve and removes the first element of a list.
<u>E remove (int index)</u>	It is used to remove the element at the specified position in a list.
<u>boolean remove (Object o)</u>	It is used to remove the first occurrence of the specified element in a list.

Continued to next.

NPTEL Online Certification Courses  
IIT Kharagpur

Continued from previous.

Method	Description
E <code>removeFirst()</code>	It removes and returns the first element from a list.
boolean <code>removeFirstOccurrence(Object o)</code>	It is used to remove the first occurrence of the specified element in a list (when traversing the list from head to tail).
E <code>removeLast()</code>	It removes and returns the last element from a list.
boolean <code>removeLastOccurrence(Object o)</code>	It removes the last occurrence of the specified element in a list (when traversing the list from head to tail).
void <code>clear()</code>	It is used to remove all the elements from a list.

NPTEL Online Certification Courses  
IIT Kharagpur

Now, surely we have learnt about insertion and now we should learn about deletion operation, how the node can be deleted from a LinkedList. Now, for the purpose of deletion, like insertion there are many methods are defined. Here I have listed those are the important methods. Now, here you can see the poll method. Now, poll method is basically it, it removes the first element of the list. Now, pollFirst is basically similar of the poll. Actually, poll, pollFirst both are defined, they are in collection and they are declared in collection and implemented in LinkedList class.

That is why both implementation is there, that is why they are in 2 different form. But their, their functionality is same, they, they will do same thing. Absolutely, no difference between poll and pollFirst, by default poll is, I mean removing the first element itself. Now, in contrast to pollFirst, there is a method called pollLast. As the name implies, you can understand. It basically deletes the last element from the list. And, pop is a one method. It basically, pop means it is also similar to the poll. It remove the first element.

Now, you may be a little bit surprised about why so many methods and whatever it is there, you can recall this LinkedList class also implements deque. Deque is an interface which extends queue. And for the queue or stack implementation everything, all those methods are required. So that is why they have implemented. Otherwise, absolutely no problem, you can use only poll. You may be limited with only using poll, pollFirst or whatever it is there. Now, there is again push method. push method is similar to insertion method actually.

It basically adds some element into a list. So, this is not related to deletion of course, it is related to rather insertion. Then, remove is very similar to the poll again. It is used to retrieve and remove the first element of a list. So poll, the pop, remove, they are basically same functionality. Now, remove int index. You can understand what it will do. It will basically remove an element in a, from a particular location. So, it is just like delete key like loop, okay? So, delete at any position like.

And, there is also boolean remove Object o. That means you can specify a particular object which you want to eliminate or remove from a list. So, you have to pass the element and then remove. And after the successful operation, it will return true or false. True implies that successfully the element has been deleted. And if false, that it did not because either the element does not exist or some error occurs here. So, these are the few methods. There are few more methods in this context regarding the deletion are there.

These are a little bit okay risky method I can say because it will, I mean delete as a whole sort of thing like clear. So, clear method as you can understand, it will basically remove all the elements from the LinkedList. Once it is cleared, means you will not be able to get back it. Otherwise, removeFirst and the removeLast, you can understand that it will remove the first element from the LinkedList, it will remove the last element from the LinkedList. Now, there are two different versions also, removeFirstOccurrence.

As you know, a linked list can contain duplicate elements. So, same element may occur several times. So, if you call this method, it will remove the first occurrence. And, opposite to this first occurrence, LastOccurrence will remove the last occurrence of the things. So, these are the few methods related to delete, deletion are defined there in the class LinkedList. Now, let us have a quick demo of a method, a program which basically exercise the different methods just we have learnt.

(Refer Slide Time: 29:30)


### Example 20.5: Deletion of an object from a LinkedList

```
/* Like insertion, deletion operation on a linked list can be carried out many ways.
Following few examples illustrates the deletion operation with methods remove(),
removeFirst(), removeLast(), etc.*/

import java.util.*;

public class DeletionFromLL {
    public static void main(String [] args) {
        // Creating a linked list
        LinkedList<String> l1 = new LinkedList<String>();
        l1.add("A");
        l1.add("E");
        l1.add("I");
        l1.add("O");
        l1.add("U");
        l1.add("H");
        System.out.println("List of vowels: "+l1);

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



NPTEL Online Certification Course  
IIT Kharagpur

### Example 20.5: Deletion of an object from a LinkedList

```
// Continued on ...


//Removing specific element from the linked list
l1.remove("H"); // Remove the vowel H
System.out.println("After deletion of H : "+l1);

//Removing element on the basis of specific position
l1.remove(1); // This will remove A from the list
System.out.println("After invoking remove(index) method: "+l1);

// Let's create another list of semi-vowels
LinkedList<String> l2 = new LinkedList<String>();
l2.add("W");
l2.add("Y");
// Adding new elements to the list of vowels
l1.addAll(l2); // Append l2 after l1
System.out.println("Updated list : "+l1);

//Removing first element from the list
l1.removeFirst();
System.out.println("After invoking removeFirst() method: "+l1);

// Continued to next ...
```



NPTEL Online Certification Course  
IIT Kharagpur

**Example 20.5: Deletion of an object from a LinkedList**

```
// Continued on ...

//Removing last element from the list
ll1.removeLast();
System.out.println("After invoking removeLast() method: "+ll1);
// Removing all elements from ll2
ll1.removeAll(ll2);
System.out.println("After removing semi-vowels: "+ll1);
ll1.add("A");
ll1.add("B");
ll1.add("A");

//Removing first occurrence of element from the list
ll1.removeFirstOccurrence("A");
System.out.println("After removing first occurrence of A: "+ll1);
//Removing the last occurrence of B
ll1.removeLastOccurrence("B");
System.out.println("After invoking removeLastOccurrence() method: "+ll1);
//Removing all the elements available in the list
ll1.clear();
System.out.println("After invoking clear() method: "+ll1);
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

Now, this is the one example is basically deletion from LinkedList demo. First, we have to create a linked list and in this case we create a linked list 1 of type string and the number of elements are added into the linked list which is like this one. So finally you can print the linked list that we have created. Once, the linked list is created then you can perform all operations related to all sort of insertion and deletion, but here we will limit our discussion related to only deletion operations.

Now, let us see ll1, the list which has been created with 6 elements in it, how the different deletion can be done here. We have done several deletion operation. Now first we call remove H. It basically removed the first element, remove the objects that means H. So if H is present there, it will remove and it will return boolean like, okay. So, it basically removes the element H. And after removing, you can print it. You can see that whether removal is successful or not.

Then remove 0, here we are passing index. So, this means that it will remove the element which is in the zeroth location, in there is a 0 or 5 or whatever it is there. Now, here by mistake if you give a remove 100, then it will return false or it can throw an exception that okay this is not a valid one. So you have to be a little bit careful about whether the element is present or not. If it is present, then you try to remove it. Otherwise, it throw exception.

Now, here you can see we create another linked list of again same type string. Let the name of linked list say ll2. We add few elements, add all the elements into it. This is just okay as I said

that you can perform any operations of your, as you want, as you wish. And then here you see ll1 removeFirst, you can understand. It is basically similar to removeFirst, poll or pollFirst, like this one. And these basically okay, after the performing this operation, it will print the list. So, there are few more again, you can write.

You can see exactly how the different other operations can be carried out. For illustration, I have added many more, many more method call; you can do also. This is just for your practice that you can see the essence of the Java LinkedList collection that is there in JCF, so that how it can be done. So removeLast application, removeAll, removeAll means it will delete all elements, it is equivalent to clean. Then you can add again, into this list is now empty. You can add the more element into the LinkedList 1. And here you see, ll1 removeAll ll2.

That means it basically remove, now this is a little bit okay, peculiar operations here. It will remove from the list ll1 which those are the objects which are basically present in the LinkedList 2 that is for the objective. Then you can check it, how it is working. Then, removeFisrtOccurence, removeLastOccurence, this means multiple or duplicate elements are there. It will remove the first or last occurrences there. Now, clear is basically another very dangerous operation to do because it will remove all the elements. So these examples show, how the different way the deletion operation can be done in a program.

(Refer Slide Time: 30:57)





### Example 20.6: Traversal operation on a linked List collection

We have learned how to print a linked-list in sequential order starting from the first item in the list. The `LinkedList` class allow you to traverse a linked in reverse order as well. For this purpose, you should use the method `descendingIterator()`. This can be applied to a list storing of any type of items.

```
/* The following program illustrates how to traverse two different type of lists in
reverse order, that is, from the end to the front. */

import java.util.*;
public class TraverseReverseLL{
    public static void main(String args[]){
        // Case 1: a linked list of countries
        LinkedList<String> lCountries = new LinkedList<String>();
        lCountries.add("Australia");
        lCountries.add("India");
        lCountries.add("South Africa");
        lCountries.add("Zimbabwe");

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





### Example 20.6: Traversal operation on a linked List collection

// Continued on...

```
//Traversing the list of countries in reverse order
Iterator itr1 = lCountries.descendingIterator();
while(itr1.hasNext()) {
    System.out.println(itr1.next());
}

// Case 2: a linked list of numbers
LinkedList<Integer> lNumbers = new LinkedList<Integer>();
lNumbers.add(123);
lNumbers.add(345);
lNumbers.add(567);
lNumbers.add(789);

//Traversing the list of numbers in reverse order
Iterator itr2 = lNumbers.descendingIterator();
while(itr2.hasNext()) {
    System.out.println(itr2.next());
}
}
```



And regarding traversal of a `LinkedList`, we are already familiar to that how the different traversals can be done. Mainly iterator or simply even for loop also you can add it, for loop just like traversing one this one. But iterator is the most convenient way to traverse a `LinkedList` actually. Now, here we can see we create a `LinkedList` called `lCountries`, add some element into it and then finally, we can see how the list can be traverse using iterator.

We can create another list, add some number, we can again traverse. But here you see traversal can be done in different ways, 2 different way mainly. Traversal can be done from front to end. Oppositely, the traversal also can be done from end to front, in the reverse order. Now see there



is no reverse method defined for the class LinkedList. Now there is another added advantage shows that some methods which are not there, you can define of your own, if you go for custom programming means programming of your own.

Now, but descending iterator in some sense can okay, can facilitate that methods that how we can iterate in a opposite order, in the order the elements are inserted, opposite to that one. So, this is the one example that you can think about you (insert) in a, visiting the elements or visiting the list in a descending order. So, for this the method is like this, descendingIterator, descending. If you write simply iterator, it will by default is a ascending order or in the same order from front to end.

(Refer Slide Time: 34:39)



The image shows a presentation slide titled "Miscellaneous Operations" in blue text. The slide has a white background with a blue border at the top and bottom. On the left, there is a brown coffee cup icon with white steam and musical notes. To the right of the coffee cup is the title "Miscellaneous Operations". The background is decorated with various icons: gears, a tree with circular nodes, a hard hat, and a chemical flask. In the bottom right corner, there is a small video inset showing a man in a light blue shirt speaking. At the bottom of the slide, there is a black bar with the NPTEL logo on the left and the text "NPTEL Online Certification Course IIT Kharagpur" on the right.



### Example 20.7: Miscellaneous operations

```

/* The LinkedList class is loaded with several other methods like get(), contain(),
size(), set(), etc. The following program illustrates those methods and their utilities
in Java programming. */

import java.util.*;
public class OtherMethodsOfLL{
    public static void main(String args[]){// Creating a linked list
        LinkedList<String>lLetters = new LinkedList<String>();
        lLetters.add("J");
        lLetters.add("O");
        lLetters.add("Y");
        lLetters.add("W");
        lLetters.add("I");
        lLetters.add("T");
        lLetters.add("H");
        lLetters.add("U");
        lLetters.add("A");
        lLetters.add("V");
        lLetters.add("A");
    }
}
// Continued to next ...

```



### Example 20.7: Miscellaneous operations


```

// Continued on ...

lLetters.add("2020");
System.out.println("List : "+lLetters);
// Finding an elements in the linked list
boolean status = lLetters.contains("Z");
if(status)
    System.out.println("List contains the element 'Z' ");
else
    System.out.println("List doesn't contain 'Z'");

// Finding the number of elements in the linked list
int size = lLetters.size();
System.out.println("Number of letters = "+ size);
// Get and set elements from the linked list
String element = lLetters.get(1);
System.out.println("Element returned by get() : " + element);
lLetters.set(1, "The fun");
System.out.println("Linked list after change : " + lLetters);
}
}

```



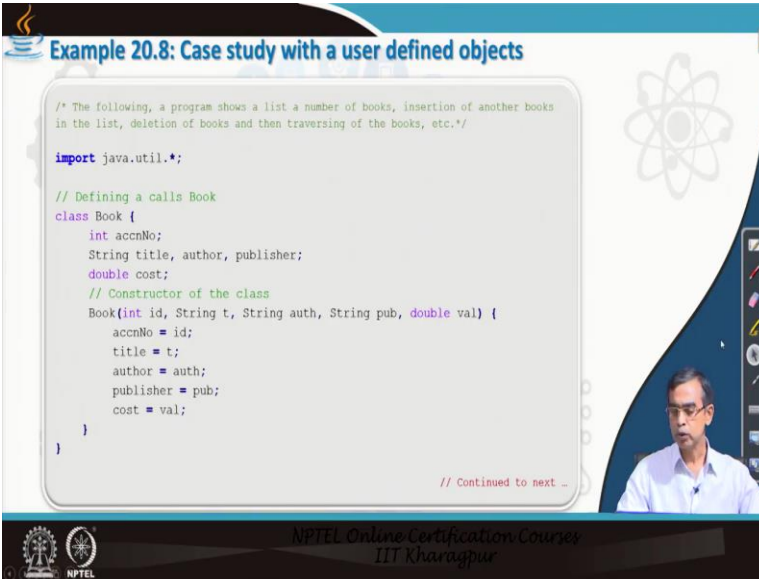
And there are certain miscellaneous operation. I will quickly go for, we have little bit short time right now. Miscellaneous operation means we can copy, add or whatever it is there. Let us first take a quick review demo of this one. We create first LinkedList Letters and add some element into it. After adding some element, we can perform many operations, the few operations that we have mentioned here. Okay, add operation, you are already familiar to.

Now, there is another contains. It is basically search. That means, if you want to search a LinkedList with specifying a particular object that means, it will search the LinkedList that you have created whether Z contains or not, it basically gives you. And, then size also another

method that you can call. It basically says that how many elements are there in the LinkedList at present. Now, get 11, it basically says that whether 11 is present there in the LinkedList or not. Like get, you can say set.

Set means the 11, the element which is present there, you can set either different. So you can modify particular objects by the set method also. So, these are the few methods which are there defined in the class. It is called the collection. They are declared and in the LinkedList class they have been implemented are useful. So for the detailed description of the class, you should refer to the Oracle tutorial document where you can find, or the many methods are there.

(Refer Slide Time: 36:02)



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

```
/* The following, a program shows a list a number of books, insertion of another books
in the list, deletion of books and then traversing of the books, etc.*/

import java.util.*;

// Defining a class Book
class Book {
    int accnNo;
    String title, author, publisher;
    double cost;
    // Constructor of the class
    Book(int id, String t, String auth, String pub, double val) {
        accnNo = id;
        title = t;
        author = auth;
        publisher = pub;
        cost = val;
    }
}
```

At the bottom right of the code editor, it says: `// Continued to next ...`

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

### Example 20.8: Case Study with a user defined objects

```
// Continued on ...  
  
// The main class maintaining a library of books  
public class LibraryLL {  
    public static void main(String[] args) {  
        List<Book> library = new LinkedList<Book>(); //Creating list of Books  
        //Creating a list of Books  
        Book b1 = new Book(101,"Oracle Java","Leslie Lamport","Oxford",88.5);  
        Book b2 = new Book(102,"Complete Java","McGraw Hill", "TMH" , 94);  
        Book b3 = new Book(103,"Joy with Java","Samanta","Prentice Hall",69.6);  
        library.add(b1); //Adding Books to list  
        library.add(b2);  
        library.add(b3);  
  
        //Traversing the list  
        for(Book b:library) {System.out.println("Book ID: " +b.acenNo);  
            System.out.println("Title: "+b.title+" Author: "+b.author+" Publisher: "+b.publisher+" Cost: "+b.cost);  
            System.out.println();  
        }  
    }  
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

Now, these are the final example. With this example, it is basically similar to the Student. We have created a user-defined class called Book and we can create some objects of the class Book and then we can add the Book objects into the list. So, this program you can practice again. We create a list called Library of type Book, add some Book into it and we are going on adding and then finally we are printing. Here you can see printing using for loop. So, this is an alternative to iterator, how the for loop can be used to print an object like. So, this is the, the usual way by which the another, another programmers' convenient way by which a list can be traverse also.

(Refer Slide Time: 36:48)

## REFERENCES

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

Okay, so these are the different methods that we have discussed. Those are there in LinkedList class and it is useful for your, okay program so that you can use. It is an handy because readily available, so the readymade things are there. If you want to have more discussion about LinkedList class, I should recommend you to go through this document, the Oracle official websites. There are tutorial is there.

And all the programs that I have used in these slides, you can get it from there and many more discussion also there which is basically easy to read. I should advise you to check the both the links and learn many more things from there. Okay, with these things I would like to remain here. We shall discuss about other data structure in the next class. Thank you very much.