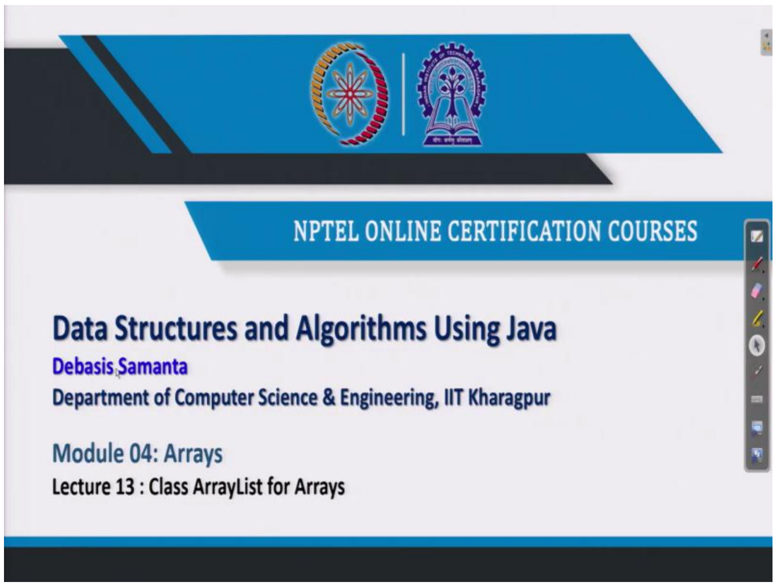
Data Structures and Algorithms Using JAVA

Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Lecture No 13
Class ArrayList for Arrays

(Refer Slide Time: 0:40)



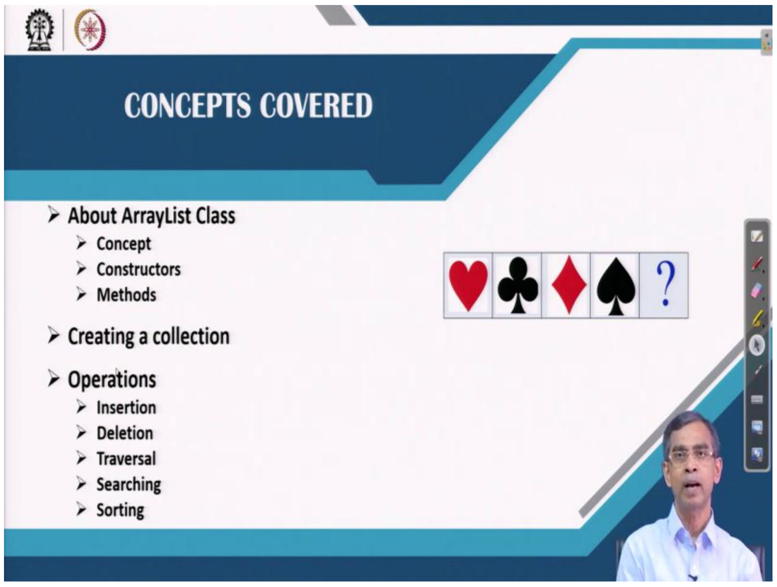
NPTEL ONLINE CERTIFICATION COURSES

Data Structures and Algorithms Using Java
Debasis Samanta
Department of Computer Science & Engineering, IIT Kharagpur

Module 04: Arrays
Lecture 13 : Class ArrayList for Arrays

Now Array being a very important one data structure, Java has enough support to handle it in the way Java can help you. Now today we are going to discuss about the Java facilities for handling Arrays. This is basically one class is there that is defined in Java collection framework, the class is called ArrayList class.

(Refer Slide Time: 0:53)



CONCEPTS COVERED

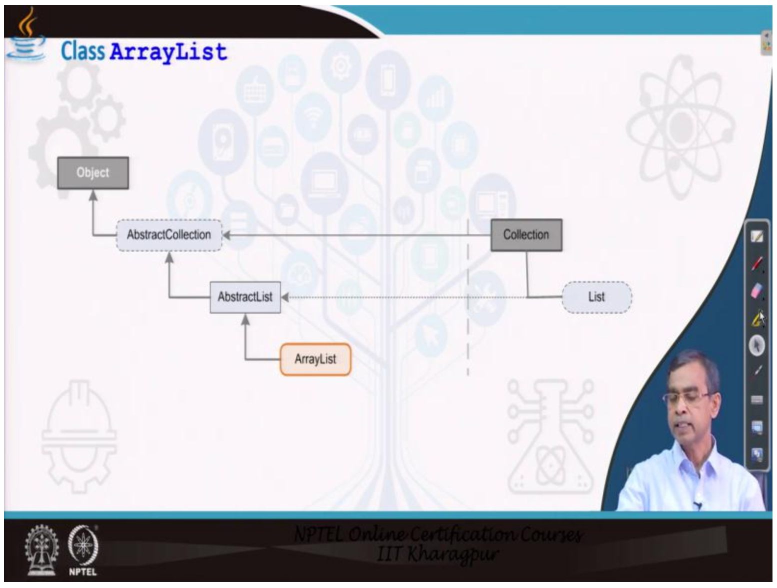
- About ArrayList Class
  - Concept
  - Constructors
  - Methods
- Creating a collection
- Operations
  - Insertion
  - Deletion
  - Traversal
  - Searching
  - Sorting

Now, let us see regarding the ArrayList class, the different constructor that basically use to create the collection of type ArrayList and different methods in this class. And then finally we will check the regular operation which is basically required using these Arrays other than the core programming that we have discussed in the last video. So, this is the utilities that is there in this ArrayList class by which you can do. You do not have to write much more code; the way we have done in the last video.

(Refer Slide Time: 1:27)

ArrayList Class

NPTEL Online Certification Courses
IIT Kharagpur



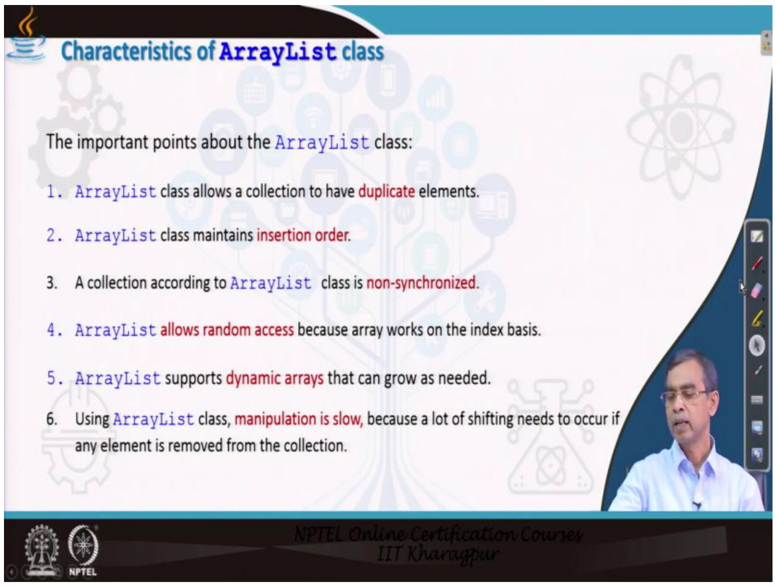
Class ArrayList

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us first discuss about ArrayList class in brief because you have already discussed these things while we are discussing it in our module 3 discussion, the regarding Java collection framework. Now ArrayList class is basically is a class which implements the list interface, the list interface is basically child interface of the super interface called the collection. It also extends the abstract list. Abstract list again extends the abstract collection.

So, these basically the composition of the ArrayList. The different methods which are declared there in list of collection are also by virtue of implementation are there. Or in other words what we can say, ArrayList implements all the methods which are declared either in abstract list or abstract collection or collection interface or list interface. Now we do not have to bother about the implementation. It is already done for you by the Java developer who will only access them in our use usefulness.

(Refer Slide Time: 2:28)



Characteristics of ArrayList class

The important points about the ArrayList class:

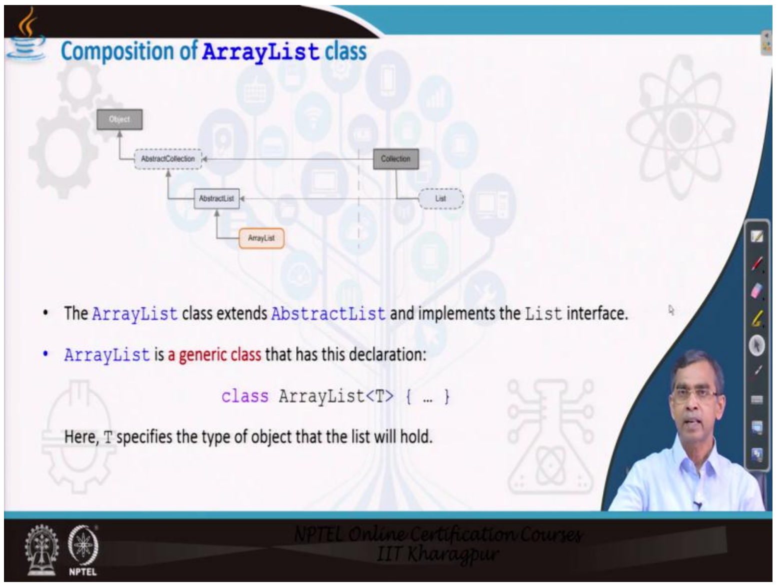
1. ArrayList class allows a collection to have duplicate elements.
2. ArrayList class maintains insertion order.
3. A collection according to ArrayList class is non-synchronized.
4. ArrayList allows random access because array works on the index basis.
5. ArrayList supports dynamic arrays that can grow as needed.
6. Using ArrayList class, manipulation is slow, because a lot of shifting needs to occur if any element is removed from the collection.

Now so far the ArrayList class is concern, it follows few important characteristics. The thing is that ArrayList class like any concept, the concept of array, it allows you to store duplicate means same element can be stored there. However, all elements are of homogeneous type and it maintain insertion order. If you go on inserting one by one without mentioning its index or whatever it is there, it will basically insert starting from the first to next and so on.

And it is again non-synchronized that means, it cannot be done in a synchronous way for threaded programming. It also allow random access because it follows the index type data structure. ArrayList, initially you can declare it, you can store some value in it and later on if you go on adding, automatically its size will be increased. That mean, ArrayList allow a programmer to grow the array dynamically.

Like any arrays for accessing or it basically painting or whatever storing elements, array is the super data structure. It is basically very fast, but if there is no insertion, deletion operations are there. Whenever insertion, deletion operations are there like any array concept, this ArrayList also gives a very slow performance. Otherwise, it is excellent one collection in the Java collection framework.

(Refer Slide Time: 3:58)



Composition of ArrayList class

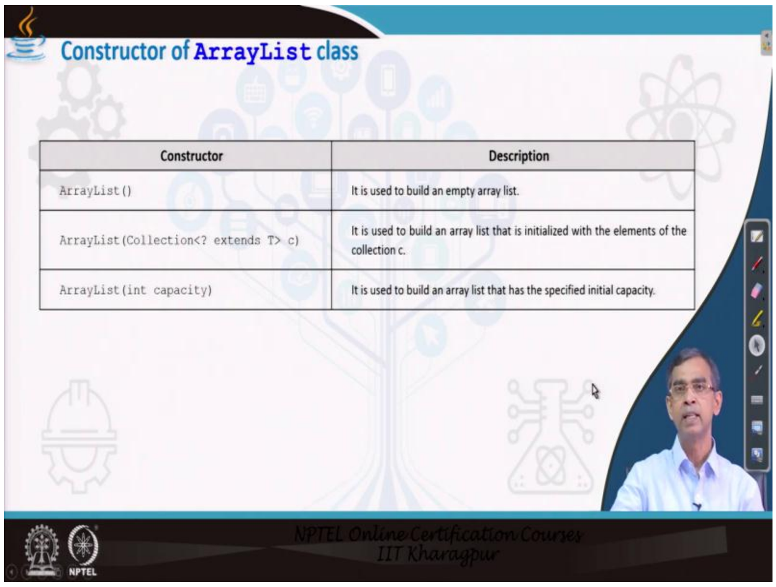
- The ArrayList class extends AbstractList and implements the List interface.
- ArrayList is a generic class that has this declaration:

```
class ArrayList<T> { … }
```

Here, T specifies the type of object that the list will hold.

Now, ArrayList, it is also a generic class that mean ant type of collection you can maintain using this class or constructor.

(Refer Slide Time: 4:09)



Constructor of ArrayList class

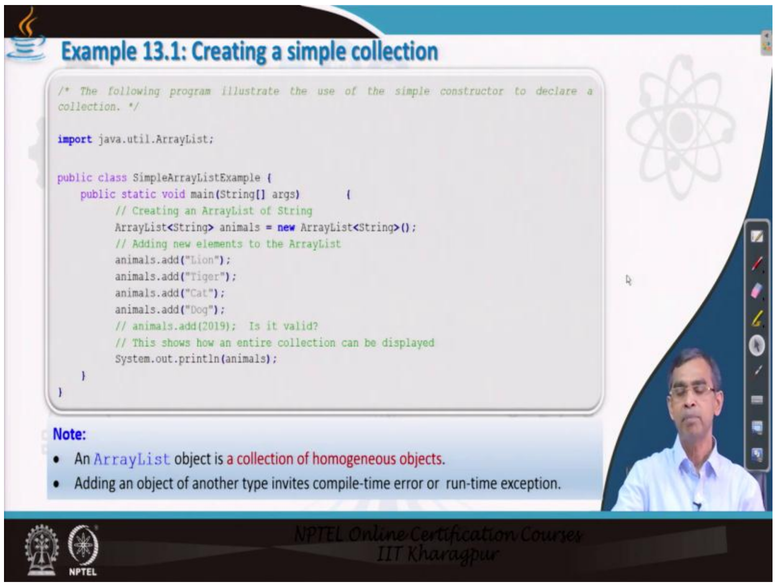
Constructor	Description
ArrayList()	It is used to build an empty array list.
ArrayList(Collection<? extends T> c)	It is used to build an array list that is initialized with the elements of the collection c.
ArrayList(int capacity)	It is used to build an array list that has the specified initial capacity.

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us see what are the different constructors are there in order to create a collection of type ArrayList. There are mainly 3 constructors which I have listed. One is default constructor. It basically create an empty collection with a minimum size, it is 16 in the Java supports and it also can be create a ArrayList collection with an existing collection. So, these are second constructor.

Another example of creating an ArrayList collection is basically specifying the capacity if you know well in advance what should be the size, then you can use the third constructor. So, these are the 3 constructors by which you can create a collection of type ArrayList.

(Refer Slide Time: 4:47)



Example 13.1: Creating a simple collection

```
/* The following program illustrate the use of the simple constructor to declare a
collection. */

import java.util.ArrayList;

public class SimpleArrayListExample {
    public static void main(String[] args)      {
        // Creating an ArrayList of String
        ArrayList<String> animals = new ArrayList<String>();
        // Adding new elements to the ArrayList
        animals.add("Lion");
        animals.add("Tiger");
        animals.add("Cat");
        animals.add("Dog");
        // animals.add(2019);  Is it valid?
        // This shows how an entire collection can be displayed
        System.out.println(animals);
    }
}
```

Note:

- An ArrayList object is a collection of homogeneous objects.
- Adding an object of another type invites compile-time error or run-time exception.

NPTEL Online Certification Courses
IIT Kharagpur

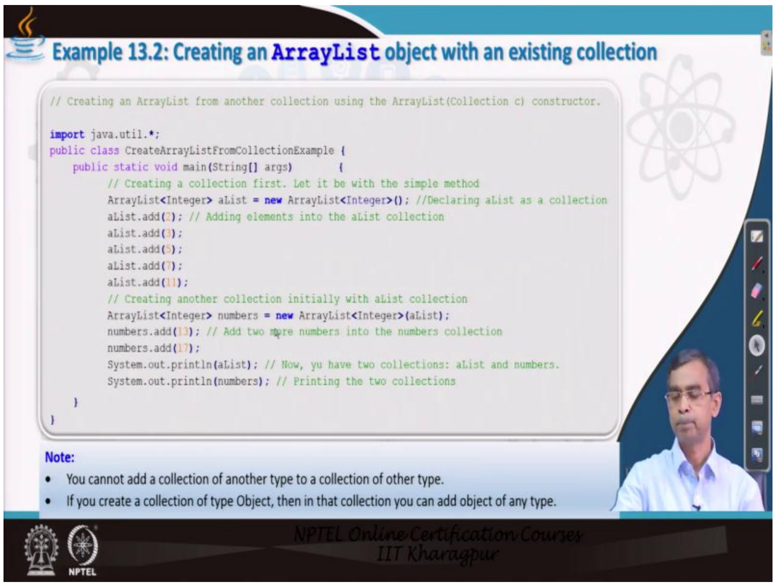
Now, this program you can little bit check it here how we have created a collection of type ArrayList. You note down few important features here. This is the program that we are going to use it and here you should import java dot util dot ArrayList because this is whatever the method that you are going to use, it is basically you there are declared in this one. If you do not do it, your program will give compilation error.

Now, you note down this is the line where we declare is a collection this collection is a type of string that mean, this is the name of the collection animals and it will store a generic class string in this case. And once it is the there, there is a method add, add method is declare in the collection or it is also implements in list and then in the ArrayList class also. So, it basically add one by, elements one by one.

So, in the zeroth position, the lion goes, next one and so on. So, this is the order of insertion that in takes place because we are adding the element into given. So, this add method as we see here is basically to add the number of elements in it. Now here so we can add any number of objects or elements of types string as we have seen here. But what will happen if we, if you want to add suppose this one because it is basically string collection. And this 201 supposed to be a number and then it may give some error actually in your case.

It basically throw some mistype mismatch error in your program if you do it. Now, so that is why when you do it, you have to be little bit careful about that how the array can initialized one by one. But you have to be little bit careful about so that the type mismatch can be avoided.

(Refer Slide Time: 6:47)



Example 13.2: Creating an ArrayList object with an existing collection

```
// Creating an ArrayList from another collection using the ArrayList(Collection c) constructor.

import java.util.*;
public class CreateArrayListFromCollectionExample {
    public static void main(String[] args)        {
          // Creating a collection first. Let it be with the simple method
          ArrayList<Integer> aList = new ArrayList<Integer>(); //Declaring aList as a collection
          aList.add(2); // Adding elements into the aList collection
          aList.add(3);
          aList.add(5);
          aList.add(7);
          aList.add(11);
          // Creating another collection initially with aList collection
          ArrayList<Integer> numbers = new ArrayList<Integer>(aList);
          numbers.add(13); // Add two more numbers into the numbers collection
          numbers.add(17);
          System.out.println(aList); // Now, yu have two collections: aList and numbers.
          System.out.println(numbers); // Printing the two collections
    }
}
```

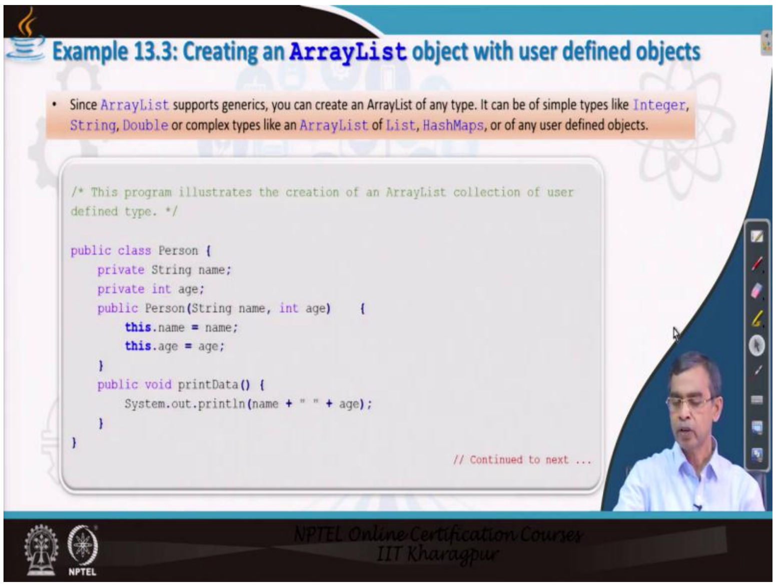
Note:
- You cannot add a collection of another type to a collection of other type.
- If you create a collection of type Object, then in that collection you can add object of any type.

Now, this is another example how with which you can create an ArrayList collection with the existing collection. This example is similar here, we have to declare the collection aList of type integer. Now we add the elements into this aList using some statement here. Then we create another collection this number and you see here, it is also type same integer but in this case it basically empty but here we create the collection with the existing collection aList.

That mean, numbers will store all the elements which is there in the aList here and later on, we can go on adding the number, it will go after this one, 1317. Now, if we print the two list, we can we can see this is the aList and this is basically all the elements which is there in the numbers. So, it basically so how an ArrayList collection can be created with an existing collection in your program.

So, this is a way by which you can create the collections of type array but here one point you can note it, if you declare the initial collection is of one type and you want to use the same collection to store into it with another type, then it can throw again exceptional error because the array type if you want to store, they should be of same type.

(Refer Slide Time: 8:13)

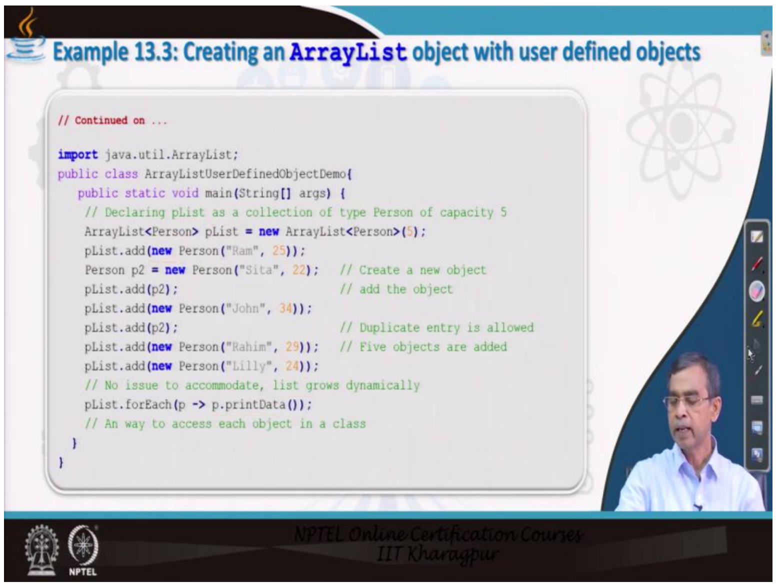


Example 13.3: Creating an ArrayList object with user defined objects

- Since ArrayList supports generics, you can create an ArrayList of any type. It can be of simple types like Integer, String, Double or complex types like an ArrayList of List, HashMaps, or of any user defined objects.

```
/* This program illustrates the creation of an ArrayList collection of user
defined type. */

public class Person {
    private String name;
    private int age;
    public Person(String name, int age)    {
        this.name = name;
        this.age = age;
    }
    public void printData() {
        System.out.println(name + " " + age);
    }
}
                                        // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur


Example 13.3: Creating an ArrayList object with user defined objects

```
// Continued on ...

import java.util.ArrayList;
public class ArrayListUserDefinedObjectDemo{
   public static void main(String[] args) {
    // Declaring pList as a collection of type Person of capacity 5
    ArrayList<Person> pList = new ArrayList<Person>(5);
    pList.add(new Person("Ram", 25));
    Person p2 = new Person("Sita", 22);   // Create a new object
    pList.add(p2);                        // add the object
    pList.add(new Person("John", 34));
    pList.add(p2);                        // Duplicate entry is allowed
    pList.add(new Person("Rahim", 29));   // Five objects are added
    pList.add(new Person("Lilly", 24));
    // No issue to accommodate, list grows dynamically
    pList.forEach(p -> p.printData());
    // An way to access each object in a class
  }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

However, there are certain Java collection framework that can allow you to initialize an ArrayList with any type whatever it is there. These are the way that you can do also alternatively. Now this is an example that we are going to discuss about how a collection can be created using user defined data type. So, first we discussed a class of user is a Person, these are the different elements in it and this is the one simple method to print the elements in a object of the type class.

Now, so this is the one user defined type. Now, we want to create a collection which will include all the elements of type Person. So, our next program is basically shows you how you can do that. If you see it we create this our program that basically shows how the user defined data type using the collection can be created and we discussed a collection pList is the name of the collection and you see this collection is of generic type, Person and this 5 indicate that initial capacity. So, 5 is the size.

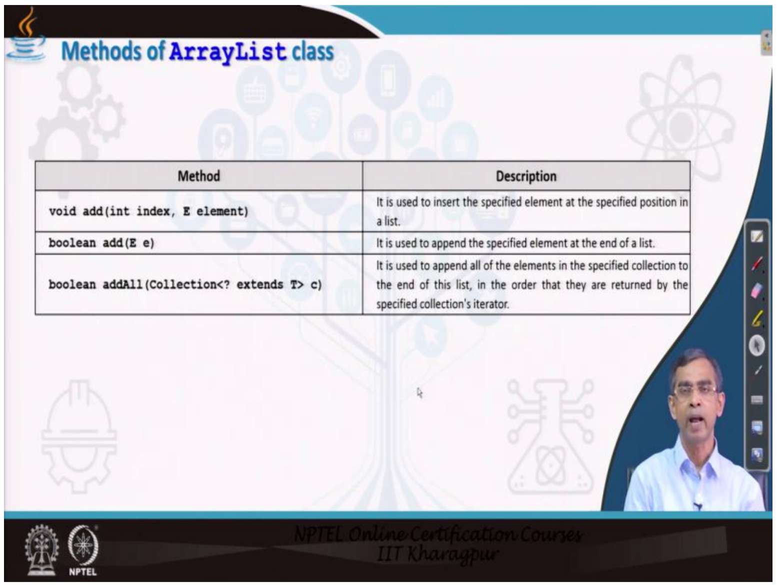
Now here, we create an object and with this object we add this. So, this is our one with a object can be created and it can be added. And this is also one object is created stand alone, it can be added. So, both the syntax is okay for adding elements at the time of creating the object also. So, this is another and this another. So, we have created around 6 objects into the program and then added into this one.

And this basically traversing the entire collection using for each method, that method is there in Java, it is called the for each loop so that for every elements in P in the collection pList is basically print the data actually. So, this way you can create a collection of any type in this case, Person type and then later on you can access each elements, print it or insert, delete whatever it is there the way the logic you know you can do that the same way or the insert method it is there in Java collection framework for this ArrayList class also you can do it.

(Refer Slide Time: 10:36)

Insertion into a Collection

NPTEL Online Certification Courses
IIT Kharagpur



Methods of ArrayList class

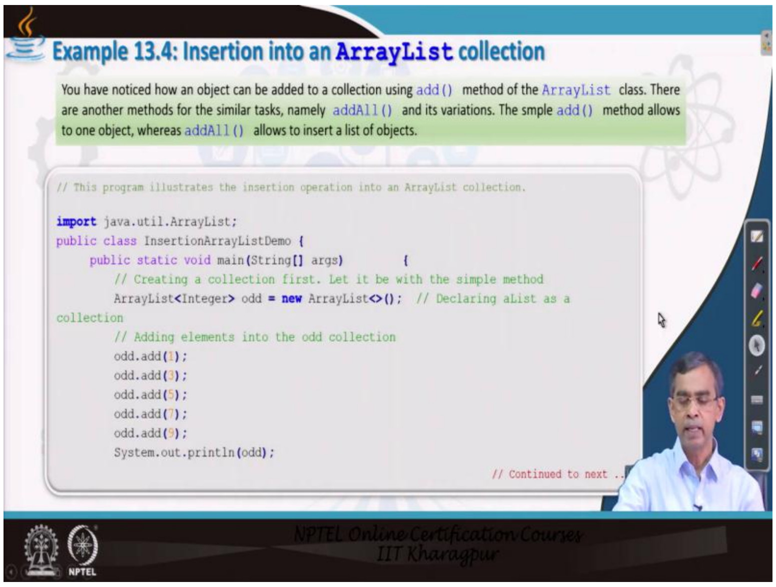
Method	Description
`void add(int index, E element)`	It is used to insert the specified element at the specified position in a list.
`boolean add(E e)`	It is used to append the specified element at the end of a list.
`boolean addAll(Collection<? extends T> c)`	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.

NPTEL Online Certification Courses
IIT Kharagpur

Now here is an example. What are the different methods are there in order to insert an elements into a ArrayList collection. As you see here we have listed the 3 methods. The first is add and two arguments in index it indicated that where you want to insert and e element is a generic method in which, which element you want to insert. So, e is the element to be inserted at the index location the first method is there. The second method is basically add E, it is basically insertion in the order, so it will insert at the end of the array if size permits.

And again I told you absolutely you should not bother about what should be the size or there or not because in case of ArrayList collection size automatically grow. Whenever it reach to the maximum size, if you want to add after that, it will automatically expand it. Then the next method addAll it is basically add a collection into the existing element. It is just like adding a collection into the collection while creating a new collection. But it is a existing collection if you want to add a set of elements into it, then addAll method you can consider.

(Refer Slide Time: 11:52)



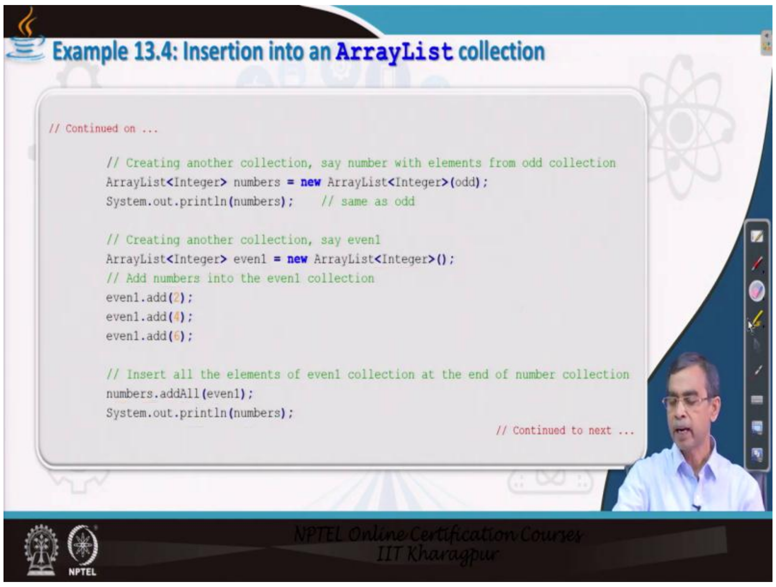
Example 13.4: Insertion into an ArrayList collection

You have noticed how an object can be added to a collection using add() method of the ArrayList class. There are another methods for the similar tasks, namely addAll() and its variations. The smple add() method allows to one object, whereas addAll() allows to insert a list of objects.

```
// This program illustrates the insertion operation into an ArrayList collection.

import java.util.ArrayList;
public class InsertionArrayListDemo {
    public static void main(String[] args)        {
        // Creating a collection first. Let it be with the simple method
        ArrayList<Integer> odd = new ArrayList<>();  // Declaring aList as a
collection
        // Adding elements into the odd collection
        odd.add(1);
        odd.add(3);
        odd.add(5);
        odd.add(7);
        odd.add(9);
        System.out.println(odd);
                                                            // Continued to next ..
```

Now let us have a quick one example showing the difference type of insertion operation and this is a program insertion into the ArrayList demo and this basically create an array collection, the name of the collection is odd here. Initially it is empty, we are going to create the collection odd here adding one by one the element and is basically print the collection.

(Refer Slide Time: 12:24)



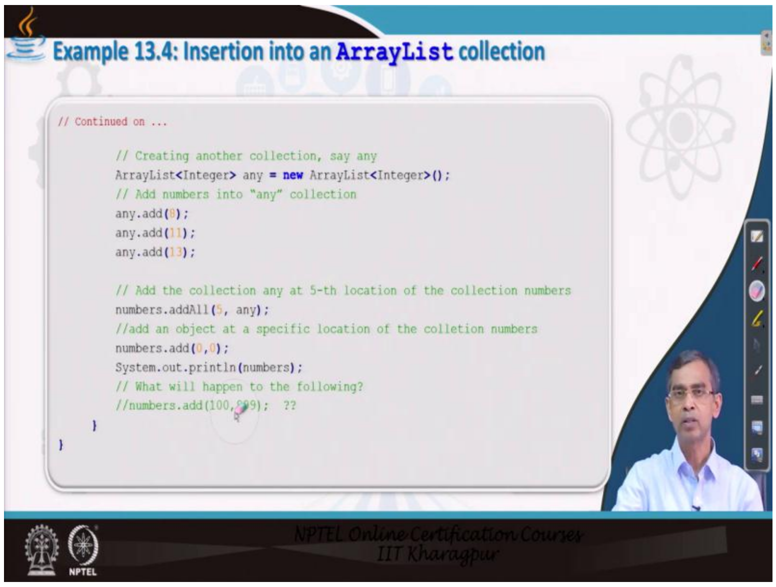
Example 13.4: Insertion into an ArrayList collection

```
// Continued on ...

        // Creating another collection, say number with elements from odd collection
        ArrayList<Integer> numbers = new ArrayList<Integer>(odd);
        System.out.println(numbers);    // same as odd

        // Creating another collection, say even1
        ArrayList<Integer> even1 = new ArrayList<Integer>();
        // Add numbers into the even1 collection
        even1.add(2);
        even1.add(4);
        even1.add(6);

        // Insert all the elements of even1 collection at the end of number collection
        numbers.addAll(even1);
        System.out.println(numbers);
                                                            // Continued to next ...
```

Now after that a list is created, now we can add few more elements into it on here is basically you see how the different element can be added here. Here we can see create a another list, another collection but using the extra collection. Alternatively, we can do it for this odd method addAll and then some other elements into it also you can do it. Now, here you can see another collection even1 which is initially empty and we add all the elements which are already there add 2 into this even1. So, basically add 2 that mean, we add 2 into the even1, add 4 into the even1 also. So, this is the even list is created.

Now, here addAll even 1 into the numbers, numbers is created initially with the odd. Now all these numbers is added so, these basically include all the elements which is the which are there in the collection odd in addition to this extra element. So, this is the usefulness of the addAll method and finally it will print after the addition of the elements or insertion of the elements into it. So, this method shows the different way the insertion can be done. Likewise, you can practice with whatever the methods I have listed to use it and see how they are effect in your program.

(Refer Slide Time: 13:46)



Example 13.4: Insertion into an ArrayList collection

```
// Continued on ...

        // Creating another collection, say any
        ArrayList<Integer> any = new ArrayList<Integer>();
        // Add numbers into "any" collection
        any.add(8);
        any.add(11);
        any.add(13);

        // Add the collection any at 5-th location of the collection numbers
        numbers.addAll(5, any);
        //add an object at a specific location of the colletion numbers
        numbers.add(0,0);
        System.out.println(numbers);
        // What will happen to the following?
        //numbers.add(100,999);  ??
    }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

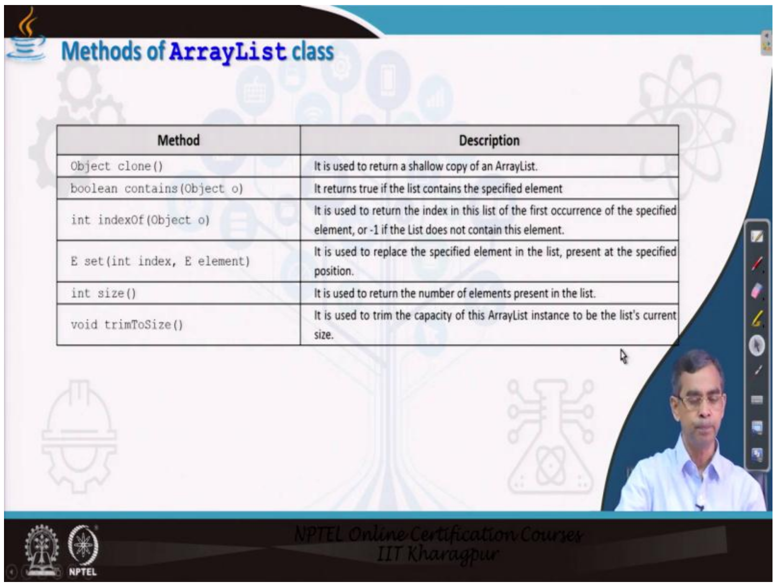
And this is also another example by which the elements can be go on adding into this program. Now here again you should note it if you want to add into the array of say, here for example, here we can add into the number up to say, 15. Now, all of us then we want to add one element say, 99, 999 in the hundredth location. What will happen?

It will not report any error. It automatically grow according to the need of the statement or program and it, it basically allocate the memory for you. So, this way the array can expand it and according to whatever it is there, but in this case mainly elements in between will not be there because it is somewhere up to 14 elements are there. 15 onwards to 99 it remain blank. It is not good of course. So, that is not good but it will do it.

(Refer Slide Time: 14:42)

Accessing Objects in a Collection

NPTEL Online Certification Courses
IIT Kharagpur



Methods of ArrayList class

Method	Description
`Object clone()`	It is used to return a shallow copy of an ArrayList.
`boolean contains(Object o)`	It returns true if the list contains the specified element
`int indexOf(Object o)`	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
`E set(int index, E element)`	It is used to replace the specified element in the list, present at the specified position.
`int size()`	It is used to return the number of elements present in the list.
`void trimToSize()`	It is used to trim the capacity of this ArrayList instance to be the list's current size.

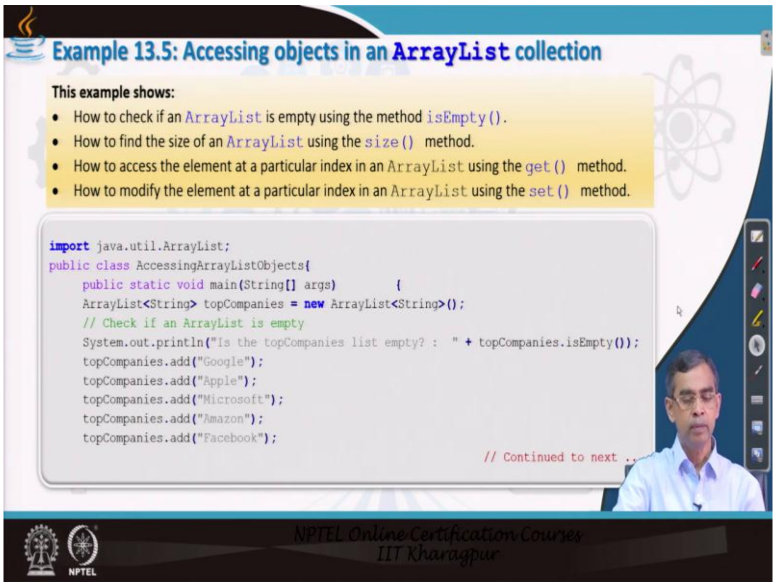
NPTEL Online Certification Courses
IIT Kharagpur

Now, accessing objects in a collection is basically we have a collection if we want to get the first element, last element or any element at the ith position of the array, whatever it is there, java provides many methods for this. Here I have listed few methods. The is the indexOf that means, if you pass an object, that element and if you want to know in which position in the collection it present, so that can be obtained.

Then set method is basically to, to replace one element at an index position with element E and it will overwrite if int index position the element is already there, that element will be replaced by this one. So, set method is there. Now other 2 method clone, clone is already you know it is basically return an object of the same type. That means if you want to create a duplicate of a existing, duplicate of an existing collection, then you can call the clone method for that collection, it will create a replica of that.

Now, contains it basically check whether in that collection an element o is present or not. If it present, it will return true otherwise, false. And size is basically what is the current size of the array that mean, number of elements present in the array. Now trimToSize if you want that it is basically I want to set few elements into it. Initially it has 100, I want dispose of say, 20 so trimToSize we can mention so it basically trim from the last to the trim size element it is there. So, these are the different method that can be applied to handle or manipulate your array.

(Refer Slide Time: 16:32)



Example 13.5: Accessing objects in an ArrayList collection

This example shows:

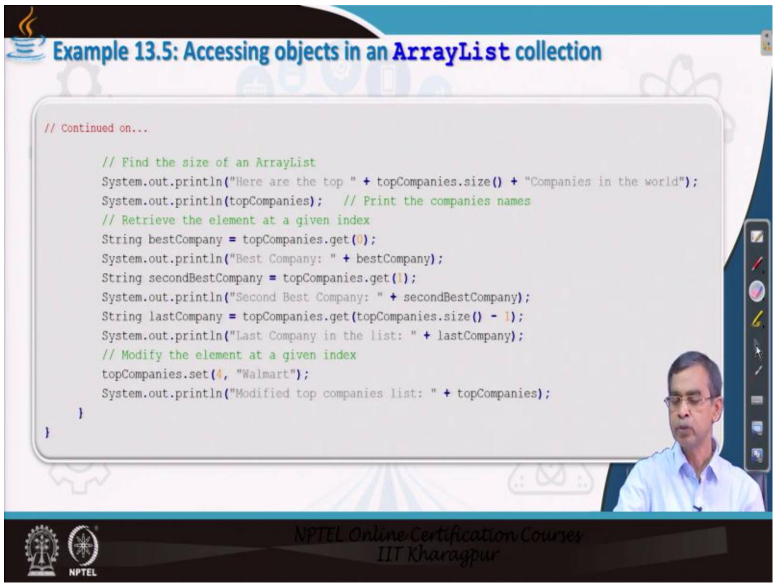
- How to check if an ArrayList is empty using the method isEmpty().
- How to find the size of an ArrayList using the size() method.
- How to access the element at a particular index in an ArrayList using the get() method.
- How to modify the element at a particular index in an ArrayList using the set() method.

```
import java.util.ArrayList;
public class AccessingArrayListObjects{
    public static void main(String[] args)        {
    ArrayList<String> topCompanies = new ArrayList<String>();
    // Check if an ArrayList is empty
    System.out.println("Is the topCompanies list empty? :  " + topCompanies.isEmpty());
    topCompanies.add("Google");
    topCompanies.add("Apple");
    topCompanies.add("Microsoft");
    topCompanies.add("Amazon");
    topCompanies.add("Facebook");
                                                    // Continued to next ..
```

Now, let us few example it is there how we can see the get method and set method in the example. Set method as I told you it basically replace one method by and get method is basically to retrieve a particular element in the ith location. Now, let us see the utilization of all these few methods those we have discussed in the in this program. This program is basically create a collection of type string and then we create the string collection with these are the different elements in it.

So, ones this array element, array ArrayList collection is created, we will be able to apply whether it isEmpty, isEmpty indicates that the whether it contains any elements or not, size, get, set or clear method, clear method also there to remove all the elements from the existing collection which needs to be called with care because it will really remove all the elements and you will not be able to get it later on.

(Refer Slide Time: 17:46)



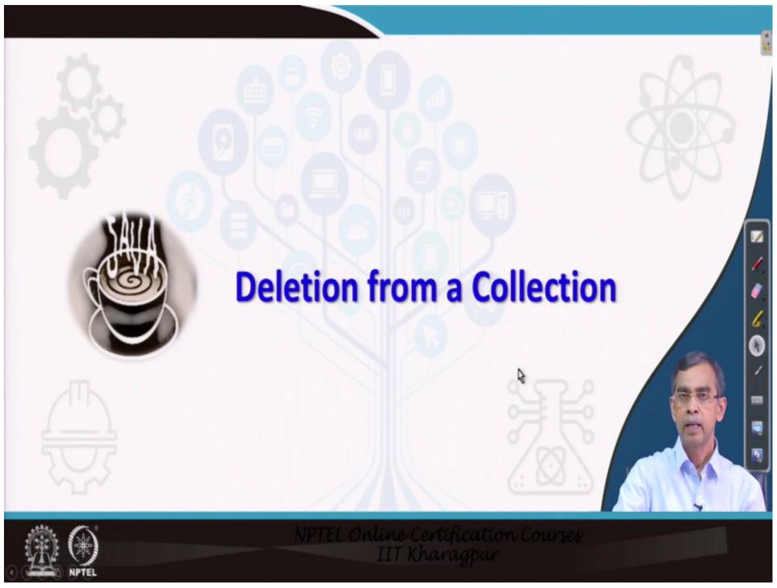
Example 13.5: Accessing objects in an ArrayList collection

```
// Continued on...

        // Find the size of an ArrayList
        System.out.println("Here are the top " + topCompanies.size() + "Companies in the world");
        System.out.println(topCompanies);   // Print the companies names
        // Retrieve the element at a given index
        String bestCompany = topCompanies.get(0);
        System.out.println("Best Company: " + bestCompany);
        String secondBestCompany = topCompanies.get(1);
        System.out.println("Second Best Company: " + secondBestCompany);
        String lastCompany = topCompanies.get(topCompanies.size() - 1);
        System.out.println("Last Company in the list: " + lastCompany);
        // Modify the element at a given index
        topCompanies.set(4, "Walmart");
        System.out.println("Modified top companies list: " + topCompanies);
    }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

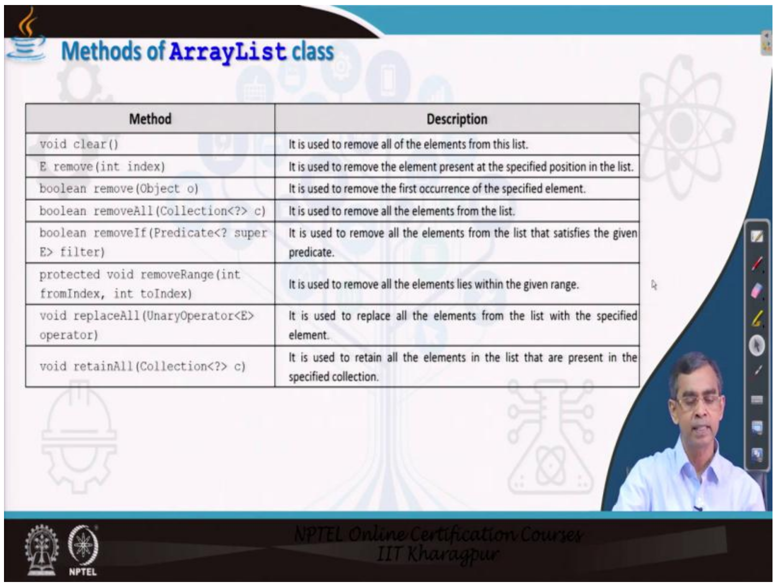
Now, these are the few examples here on the created collection top companies and here you see get 0, it basically return you the first elements in the ArrayList collection. Get 1, the next elements and here get whatever the elements are there with the last elements. And here you see set 4 that mean in the fourth location whatever the elements are there will be replaced by another element Walmart and then it will print the modified list after the get set and everything is there. So, these are the few method which you can see the existing methods those are defined in the ArrayList class can be used in your program to perform certain desired operation.

(Refer Slide Time: 18:37)



Deletion from a Collection

NPTEL Online Certification Courses
IIT Kharagpur



Methods of ArrayList class

Method	Description
`void clear()`	It is used to remove all of the elements from this list.
`E remove(int index)`	It is used to remove the element present at the specified position in the list.
`boolean remove(Object o)`	It is used to remove the first occurrence of the specified element.
`boolean removeAll(Collection<?> c)`	It is used to remove all the elements from the list.
`boolean removeIf(Predicate<? super E> filter)`	It is used to remove all the elements from the list that satisfies the given predicate.
`protected void removeRange(int fromIndex, int toIndex)`	It is used to remove all the elements lies within the given range.
`void replaceAll(UnaryOperator<E> operator)`	It is used to replace all the elements from the list with the specified element.
`void retainAll(Collection<?> c)`	It is used to retain all the elements in the list that are present in the specified collection.

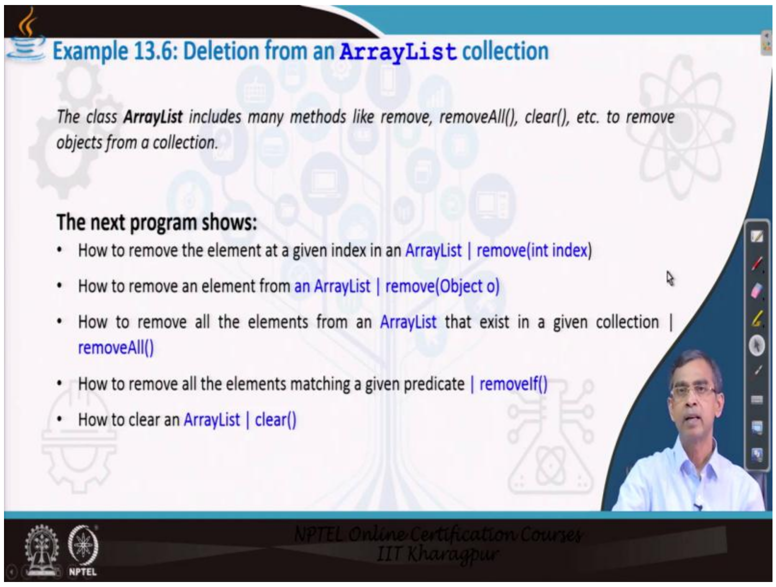
NPTEL Online Certification Courses
IIT Kharagpur

Like creating and insertion of element, there is a deletion operation, in order to do the deletion operation there are many methods like here clear method is basically clean the entire collection, remove, remove a particular element giving the position that from this element once it is removed, it will not be able to recover it. And then object o, if you pass an argument and object, it will remove if the object present in the collection otherwise, it will return false.

removeAll Collection c that means it will basically search the entire collection, and removing all the element which are there in which elements are there. removeIf is a predicate you can specify if a sudden condition is satisfied, then only that element will be removed. It is some way of other than collection it is there. removeRange starting from to end one so, from say, particular index to a particular index, all the elements will be removed for this purpose the removeRange method is there.

The returnAll is basically just like intersection of this one so, it will keep all elements which are there and other than all the elements which is not in this collection will be removed from this one. So, it is little bit different, then removeAll and returnAll, there the two methods are there. Now, you can write your program to understand how these methods work for an existing collection, for this purpose you have to create a collection of your own and then try to apply or call all these methods on that collection and see how they work for you.

(Refer Slide Time: 20:14)



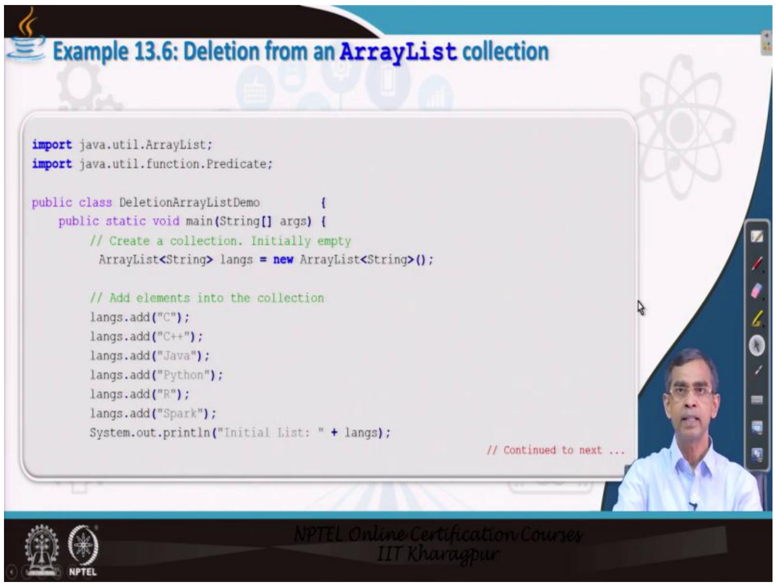
Example 13.6: Deletion from an ArrayList collection

*The class **ArrayList** includes many methods like remove, removeAll(), clear(), etc. to remove objects from a collection.*

The next program shows:

- How to remove the element at a given index in an ArrayList | remove(int index)
- How to remove an element from an ArrayList | remove(Object o)
- How to remove all the elements from an ArrayList that exist in a given collection | removeAll()
- How to remove all the elements matching a given predicate | removeIf()
- How to clear an ArrayList | clear()

NPTEL Online Certification Courses
IIT Kharagpur



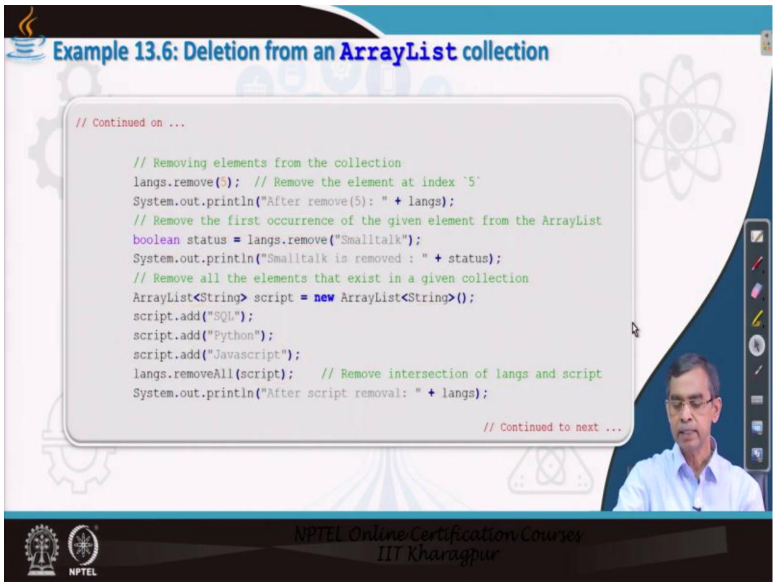
Example 13.6: Deletion from an ArrayList collection

```
import java.util.ArrayList;
import java.util.function.Predicate;

public class DeletionArrayListDemo         {
    public static void main(String[] args) {
        // Create a collection. Initially empty
         ArrayList<String> langs = new ArrayList<String>();

        // Add elements into the collection
        langs.add("C");
        langs.add("C++");
        langs.add("Java");
        langs.add("Python");
        langs.add("R");
        langs.add("Spark");
        System.out.println("Initial List: " + langs);
                                                  // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

I have listed few examples to understand this one. I will highlight few methods those are we have mentioned just now and this example shows you first we create an array, the name of the collection is called langs and add few elements in it. This ArrayList collection is a collection of string type data and after the ArrayList collection is created, we print the existing ArrayList. So, this is basically the first step to create the ArrayList collection and later let us see how the different methods regarding the removal or deletion operation can be performed.

(Refer Slide Time: 20:55)



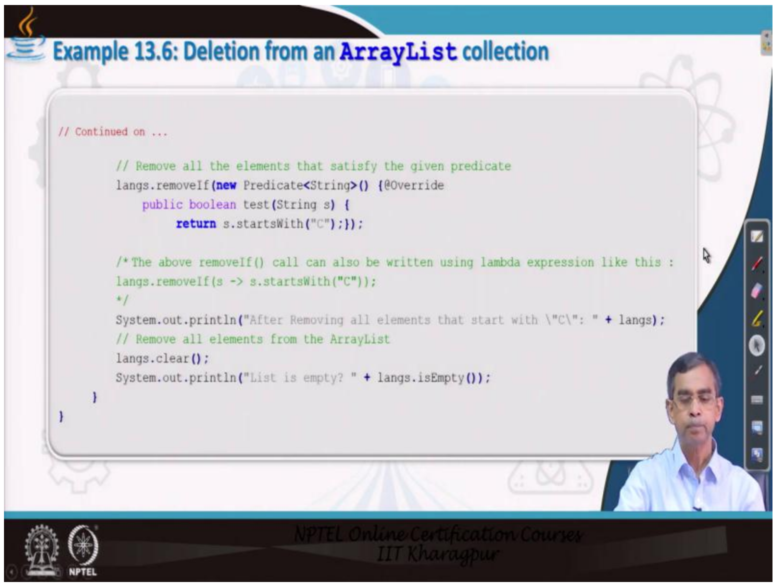
Example 13.6: Deletion from an ArrayList collection

```
// Continued on ...

        // Removing elements from the collection
        langs.remove(5);  // Remove the element at index `5`
        System.out.println("After remove(5): " + langs);
        // Remove the first occurrence of the given element from the ArrayList
        boolean status = langs.remove("Smalltalk");
        System.out.println("Smalltalk is removed : " + status);
        // Remove all the elements that exist in a given collection
        ArrayList<String> script = new ArrayList<String>();
        script.add("SQL");
        script.add("Python");
        script.add("Javascript");
        langs.removeAll(script);    // Remove intersection of langs and script
        System.out.println("After script removal: " + langs);

                                                        // Continued to next ...
```

Now, here is the method remove 5, you can understand what it does for you. It will basically remove the elements which is in the fifth index. And after removing it will print the modified collection. Then status remove Smalltalk. It basically says that if it is successfully remove this object, then status is a Boolean, that means true or not. So, that means you can understand whether Smalltalk was there in the collection and once it is there if then it will remove it and after removal you can print them, list also.

Now, this is another example here. You can see we create another collection script and this one and then the script we create some elements in it and this is the new collection it is there. Now, removeAll script, you can understand that we call them method for the collection langs and script is the argument for this, that means whatever the elements which are present here will be removed from the existing collection langs and then it will print the collection langs after removing if common elements are there.

(Refer Slide Time: 22:10)

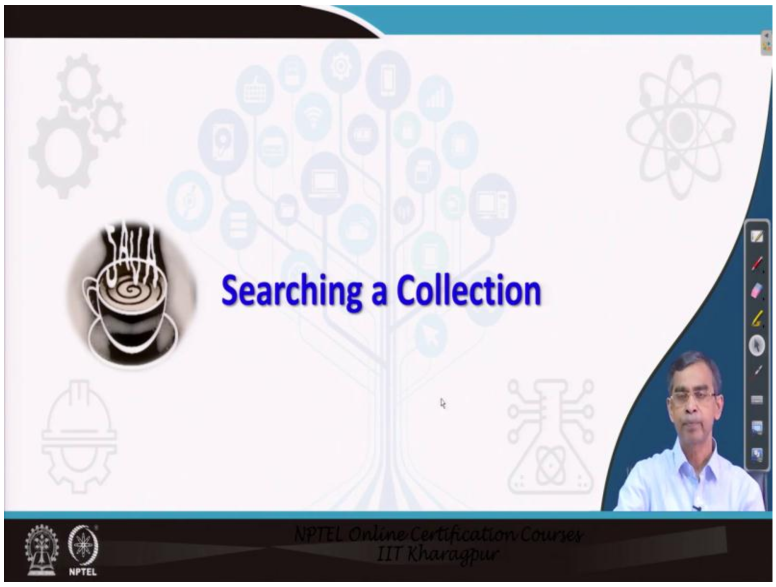


Example 13.6: Deletion from an ArrayList collection

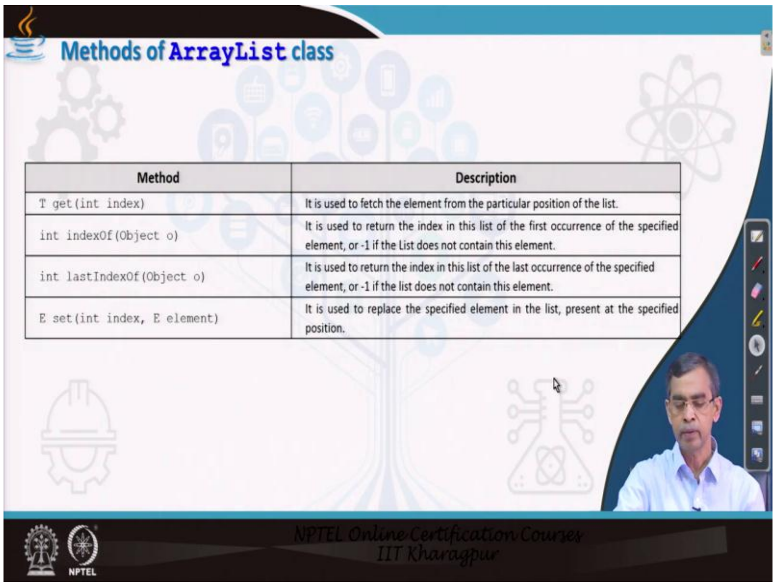
```
// Continued on ...

        // Remove all the elements that satisfy the given predicate
        langs.removeIf(new Predicate<String>() {@Override
            public boolean test(String s) {
                 return s.startsWith("C");});

        /* The above removeIf() call can also be written using lambda expression like this :
        langs.removeIf(s -> s.startsWith("C"));
        */
        System.out.println("After Removing all elements that start with \"C\": " + langs);
        // Remove all elements from the ArrayList
        langs.clear();
        System.out.println("List is empty? " + langs.isEmpty());
    }
}
```

Now there are few more methods likewise, you can see the that can be exercised. For example, this is the one the Predicate method. Here, we want to remove that elements whose first letter is correct c, then we can call this predicate and the we can remove it here in this one. Then clear is the method you can understand what it will does do for you is that it will (crea), it will clean or remove all the elements which are currently present in this collection langs and then print it. It is basically is empty, that means the elements is not, it will basically print false like. So, these are the different methods regarding deletion operations that we have exercised for your program.

(Refer Slide Time: 23:01)



Searching a Collection

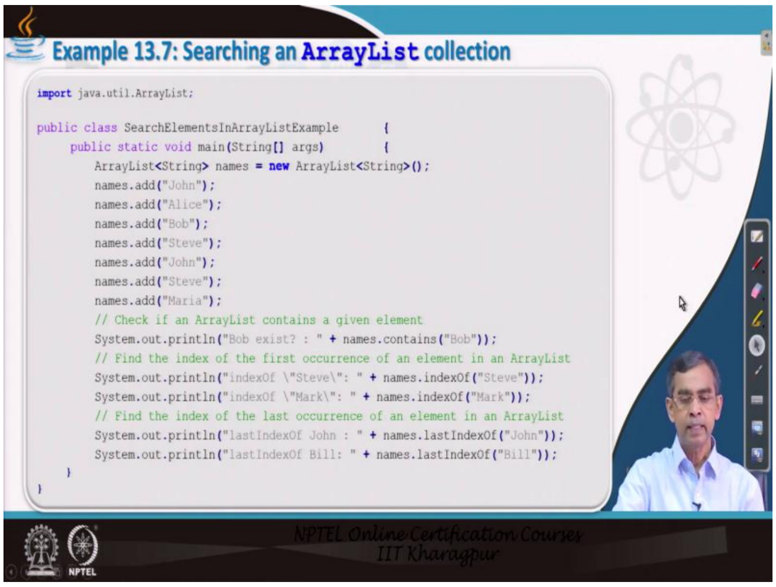

Methods of ArrayList class

Method	Description
T get(int index)	It is used to fetch the element from the particular position of the list.
int indexOf(Object o)	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
int lastIndexOf(Object o)	It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
E set(int index, E element)	It is used to replace the specified element in the list, present at the specified position.

NPTEL Online Certification Courses
IIT Kharagpur

And then there are few methods also there regarding the searching. The search method in fact exclusively there is no method in the collection. Only the search is by virtue of accessing like get method, the index of it basically say that whether object o is there or not. If it is there, then it basically return in which location. So, this one way of searching like. So, there is no exclusive search method like there but by virtue of all these methods you can perform the search method and this one. Now, so there is a one program that you can consider how the searching can be done in your Arrays there.

(Refer Slide Time: 23:38)


Example 13.7: Searching an ArrayList collection

```
import java.util.ArrayList;

public class SearchElementsInArrayListExample      {
    public static void main(String[] args)        {
        ArrayList<String> names = new ArrayList<String>();
        names.add("John");
        names.add("Alice");
        names.add("Bob");
        names.add("Steve");
        names.add("John");
        names.add("Steve");
        names.add("Maria");
        // Check if an ArrayList contains a given element
        System.out.println("Bob exist? : " + names.contains("Bob"));
        // Find the index of the first occurrence of an element in an ArrayList
        System.out.println("indexOf \"Steve\": " + names.indexOf("Steve"));
        System.out.println("indexOf \"Mark\": " + names.indexOf("Mark"));
        // Find the index of the last occurrence of an element in an ArrayList
        System.out.println("lastIndexOf John : " + names.lastIndexOf("John"));
        System.out.println("lastIndexOf Bill: " + names.lastIndexOf("Bill"));
    }
}
```

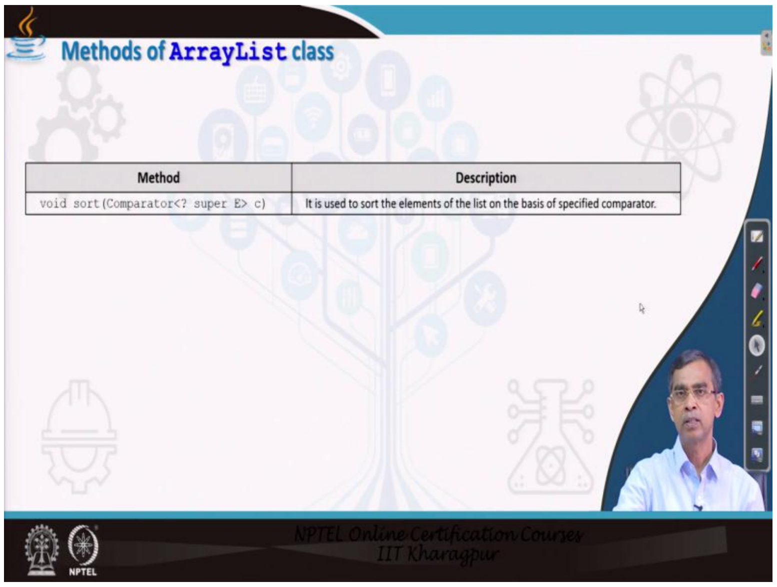
NPTEL Online Certification Courses
IIT Kharagpur

This is a program that you can see we create a collection of the type string here name of the collection is names and then we call the different method that we have discussed one by one.

Index of, last index, first index all these things are there. So, this are the basically regarding searching like distinct collection that you can check with your own time.

(Refer Slide Time: 24:08)

Sorting a Collection

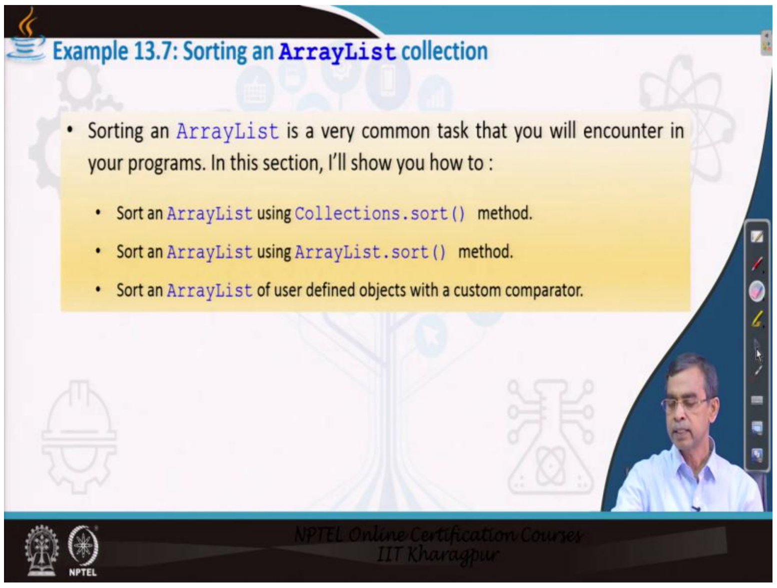


Methods of ArrayList class

Method	Description
`void sort(Comparator<? super E> c)`	It is used to sort the elements of the list on the basis of specified comparator.

Now let us see there is a sorting is a very important one application in any collection involved and here we are going to discuss about sorting collection but you have to little bit careful about when you want to the sorting. Sorting for the number data type is very easy but for other data type user defined data type you have to be little bit careful about it. Anyway let us see how the sorting can be accomplished using array list collection, there is one method is there all the sort method by which you can do the shorting operations on any collection.

(Refer Slide Time: 24:45)



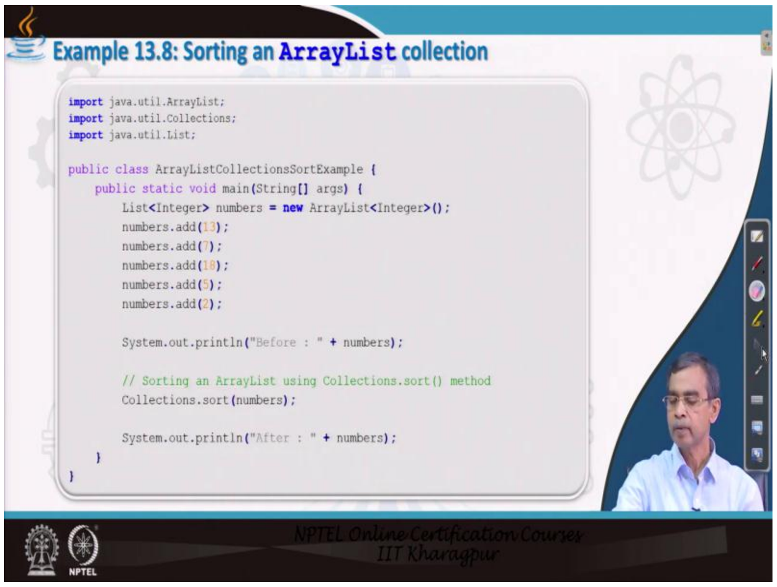
Example 13.7: Sorting an ArrayList collection

- Sorting an ArrayList is a very common task that you will encounter in your programs. In this section, I'll show you how to :
  - Sort an ArrayList using Collections.sort() method.
  - Sort an ArrayList using ArrayList.sort() method.
  - Sort an ArrayList of user defined objects with a custom comparator.

Now, this example also has a two versions of sort one is collection dot sort and ArrayList dot sort. I, you do not have to bother about which sort but you can call any two methods for your collection it is basically the factory method it is already defined in the collection you do not have to bother about it. ArrayList is also available method which is defined in ArrayList collection. Now let us see these are the static method that you can apply for your collection objects and you can get the work done. Now, let us see the program that you can apply it.

(Refer Slide Time: 25:16)

Sorting using Collection.sort()


Example 13.8: Sorting an ArrayList collection

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class ArrayListCollectionsSortExample {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<Integer>();
        numbers.add(13);
        numbers.add(7);
        numbers.add(18);
        numbers.add(5);
        numbers.add(2);

        System.out.println("Before : " + numbers);

        // Sorting an ArrayList using Collections.sort() method
        Collections.sort(numbers);

        System.out.println("After : " + numbers);
    }
}
```

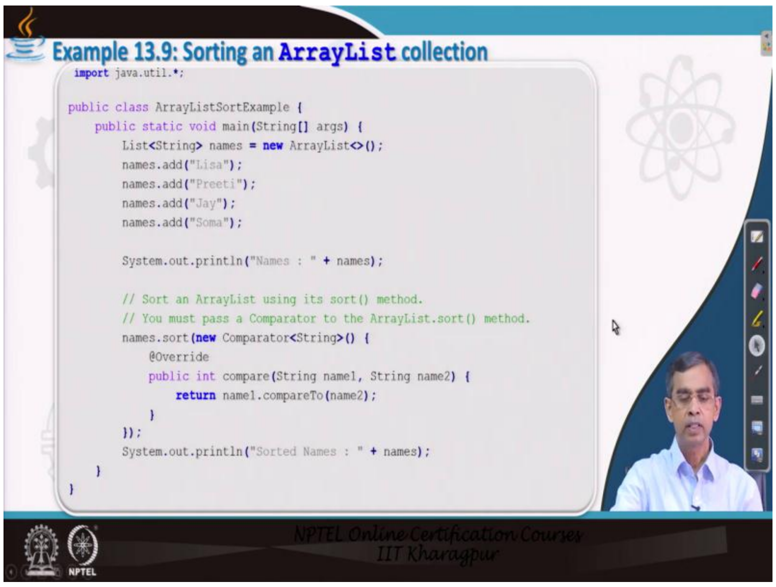
NPTEL Online Certification Courses
IIT Kharagpur

We first give a demo about collection dot sort method. It is as usual just to create a collection and then we can call the collection dot sort method, you can see this is a factory method that you can call, these are numbers. The number is the collection we pass it into this, you say it sort and sorted element will be stored in this collection again, it does not return the sort one. So, numbers element will be modified as a sorted order of the elements and after the sorting you can print the collection, it will print this one.

(Refer Slide Time: 25:54)

Sorting using ArrayList.sort()

NPTEL Online Certification Courses
IIT Kharagpur



Example 13.9: Sorting an ArrayList collection

```
import java.util.*;

public class ArrayListSortExample {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>();
        names.add("Lisa");
        names.add("Preeti");
        names.add("Jay");
        names.add("Soma");

        System.out.println("Names : " + names);

        // Sort an ArrayList using its sort() method.
        // You must pass a Comparator to the ArrayList.sort() method.
        names.sort(new Comparator<String>() {
            @Override
            public int compare(String name1, String name2) {
                return name1.compareTo(name2);
            }
        });
        System.out.println("Sorted Names : " + names);
    }
}
```

Now likewise, there is a ArrayList or sort method also can be called. It is basically as earlier one also. Only one thing that you can declare about here in this particular example, the compare method needs to be implemented if you want to create the compare because comparison is one important concept which is required there in sorting operation. Now, for the different numbers or comparison is less than, greater than or equals, whatever it is there but for the string and others you have to use the compareTo method which is declare the string.

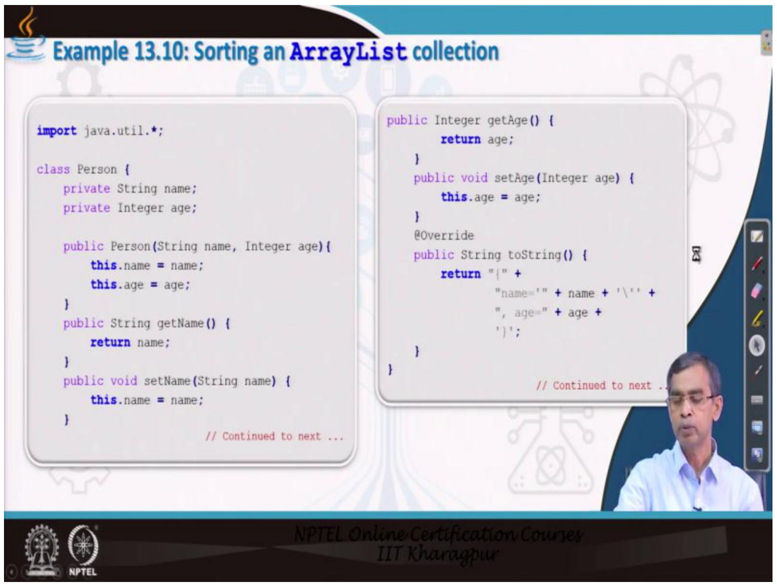
But of you want to apply the same method for your user defined class, maybe string or others, the default can be used otherwise, you have to little bit modify your method accordingly. So, here is an example as you can see create a list name, name of the list or collection is names and it includes initially these are the elements and we modify the sort compareTo method.

We can call this method sort, it is basically it is a ArrayList sort methods and then we pass this compareTo method. In compareTo method is basically called the string compare in name1 dot compareTo name2 it is basically string compare. So, this return and then accordingly sort it and sorted version of the collection will be printed here. So, this is the way that ArrayList can be sorted.

(Refer Slide Time: 27:19)

Sorting using Custom Comparator

NPTEL Online Certification Courses
IIT Kharagpur



Example 13.10: Sorting an ArrayList collection

```
import java.util.*;

class Person {
    private String name;
    private Integer age;

    public Person(String name, Integer age){
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
                                // Continued to next ...
```

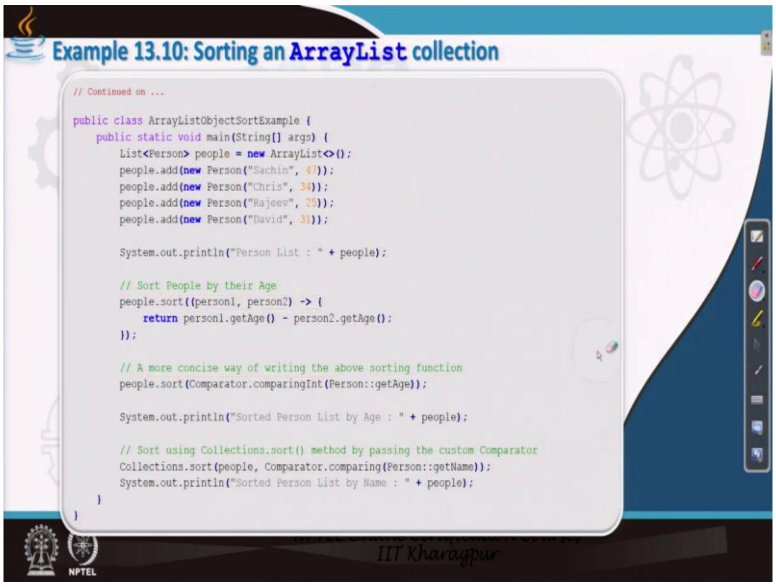
```
public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
                                // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

But for the custom classes, user defined classes you have to really write your own method. Here is an example that you can check and then run it, I declare one class, user defined class public with certain regular method constructor getName, setName and getAge like setAge and then here is an Override method of string toString is basically all the elements which are there, string, integer and everything can be converted into a string in this method and then it return it.

So, basically putting all the elements into store this is a one idea you can follow of converting all the elements that is there in a particular type of class or fields and then they are printing in the forms of converting into the string actually. Now let us see the compareTo method for which we can we can sort according to the age, there are 2 attributes we can do the sorting either with respect to name or age.

(Refer Slide Time: 28:24)



Example 13.10: Sorting an ArrayList collection

```java
// Continued on ...

public class ArrayListObjectSortExample {
    public static void main(String[] args) {
        List<Person> people = new ArrayList<>();
        people.add(new Person("Sachin", 47));
        people.add(new Person("Chris", 34));
        people.add(new Person("Rajeev", 25));
        people.add(new Person("David", 31));

        System.out.println("Person List : " + people);

        // Sort People by their Age
        people.sort((person1, person2) -> {
            return person1.getAge() - person2.getAge();
        });

        // A more concise way of writing the above sorting function
        people.sort(Comparator.comparingInt(Person::getAge));

        System.out.println("Sorted Person List by Age : " + people);

        // Sort using Collections.sort() method by passing the custom Comparator
        Collections.sort(people, Comparator.comparing(Person::getName));
        System.out.println("Sorted Person List by Name : " + people);
    }
}
```

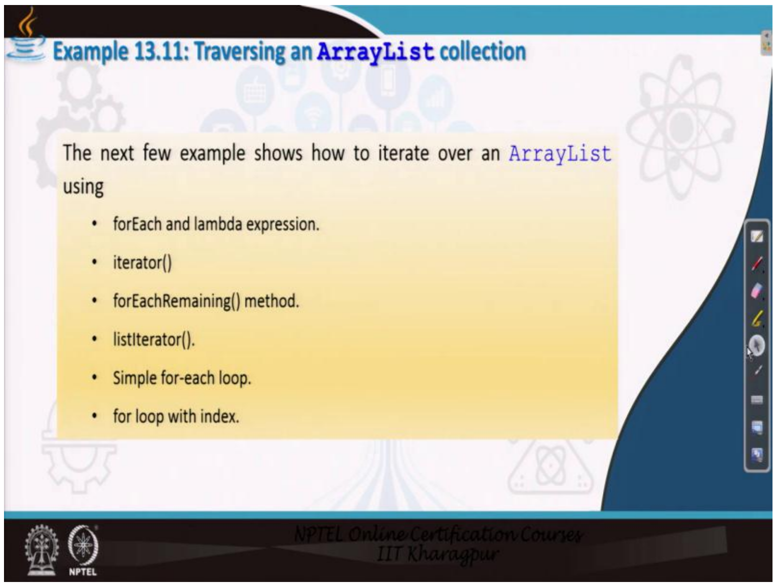
I can show you one example which basically show you with respect to age, the integer how it can be sorted here. So, first we create a collection, the name of the collection is people. We initialize the collection with some elements in it. So, it basically after creating the collection it print and then here, we can see how the sort method can be called using this one. So, it is basically is a compareTo method that which is a getAge and person2 getAge and now here is basically, if this getAge is greater than this one, then it is basically in swap or interchange or it basically shows that in which order it is there.

So, it basically shows you ascending order of sorting and then accordingly comparator method can be defined for you by which you can do it and then it works for you after the sorting elements is there. So, these are the few important methods that you can check and then you see how it works for you so that the user defined method can be there. I just give an idea about it or give a task rather so that just modify this program not for sorting according to the age but for string. You can use again compareTo string method for this purpose and then redefine this program and run it. So, these are the different sorting operations that we have discussed about.

(Refer Slide Time: 29:44)

Traversing a Collection

NPTEL Online Certification Courses
IIT Kharagpur


Example 13.11: Traversing an ArrayList collection

The next few example shows how to iterate over an ArrayList using

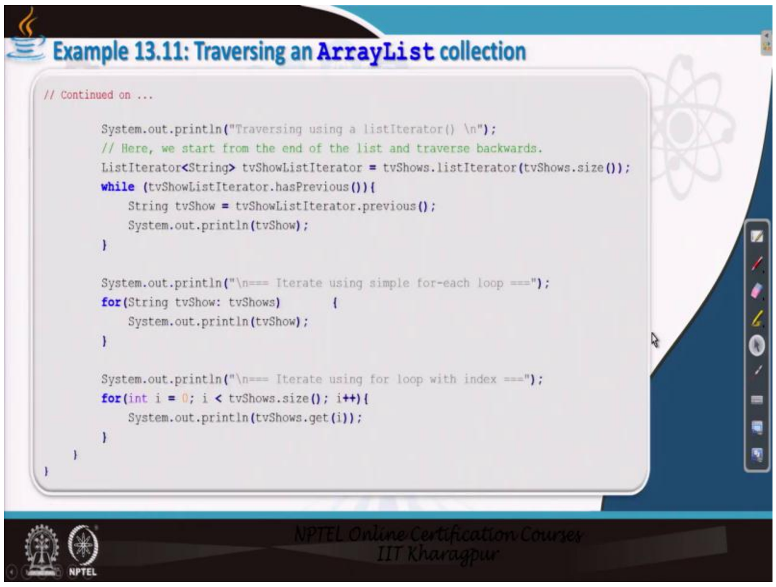
- forEach and lambda expression.
- iterator()
- forEachRemaining() method.
- listIterator().
- Simple for-each loop.
- for loop with index.

NPTEL Online Certification Courses
IIT Kharagpur

Traversing a collection, likewise, many methods are there. Traversing is very simple for any collection there. But there are readymade few sorting traversal methods are there which we have listed there. We shall discuss in details when we will discuss the about Java utility there regarding the iterator, lambda expression, forEachRemaining, listIterators, split iterator, for-each loop, loop with index, while loop and others.

So, these are the different way by which the loop can be controlled over a collection and it can be stored there. However, we will discuss these things later on in details about traversing a collection. This is at the end of this course actually we will discuss about because how the different way the collection can be traversed. Right now we can keep on, keep it on hold so that you can, however if you are interested you can check it from the different materials in the link there were I have given there. I have also mentioned few examples so that you can run those examples there also

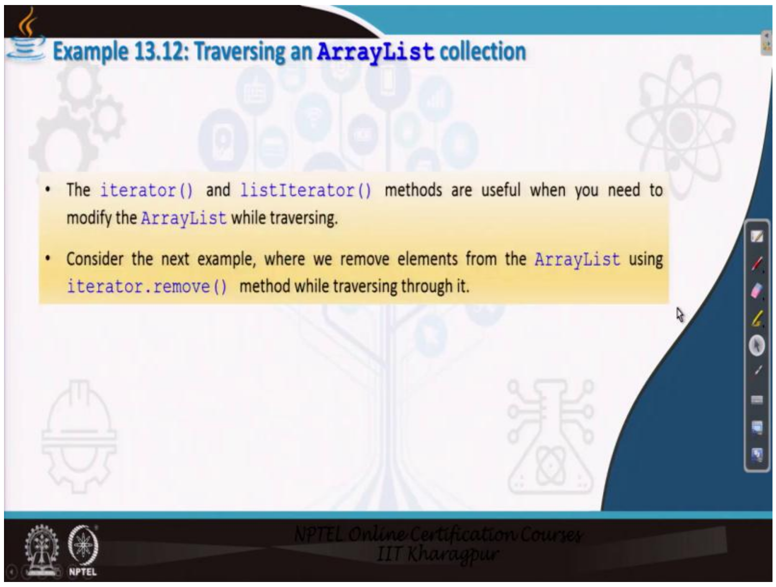
(Refer Slide Time: 30:45)



Example 13.11: Traversing an ArrayList collection

```
// Continued on ...

        System.out.println("Traversing using a listIterator() \n");
        // Here, we start from the end of the list and traverse backwards.
        ListIterator<String> tvShowListIterator = tvShows.listIterator(tvShows.size());
        while (tvShowListIterator.hasPrevious()){
            String tvShow = tvShowListIterator.previous();
            System.out.println(tvShow);
        }

        System.out.println("\n=== Iterate using simple for-each loop ===");
        for(String tvShow: tvShows)        {
            System.out.println(tvShow);
        }

        System.out.println("\n=== Iterate using for loop with index ===");
        for(int i = 0; i < tvShows.size(); i++){
            System.out.println(tvShows.get(i));
        }
    }
}
```

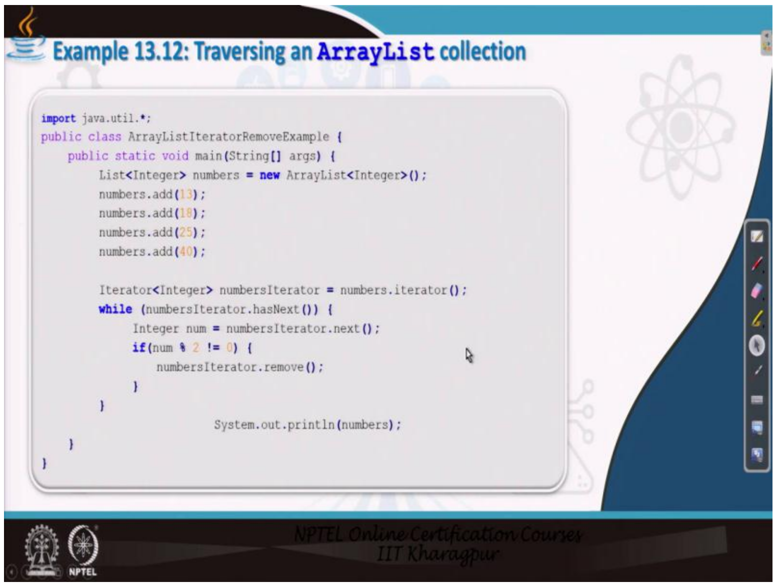
NPTEL Online Certification Courses
IIT Kharagpur



Example 13.12: Traversing an ArrayList collection

- The iterator() and listIterator() methods are useful when you need to modify the ArrayList while traversing.
- Consider the next example, where we remove elements from the ArrayList using iterator.remove() method while traversing through it.

NPTEL Online Certification Courses
IIT Kharagpur



Example 13.12: Traversing an `ArrayList` collection

```java
import java.util.*;
public class ArrayListIteratorRemoveExample {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<Integer>();
        numbers.add(13);
        numbers.add(18);
        numbers.add(25);
        numbers.add(40);

        Iterator<Integer> numbersIterator = numbers.iterator();
        while (numbersIterator.hasNext()) {
            Integer num = numbersIterator.next();
            if(num % 2 != 0) {
                numbersIterator.remove();
            }
        }
                    System.out.println(numbers);
    }
}
```

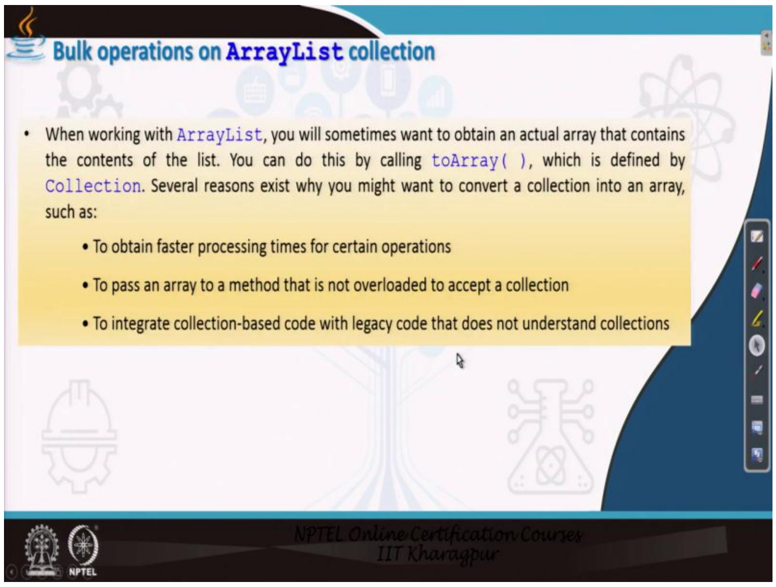
NPTEL Online Certification Courses
IIT Kharagpur

So, this is a one example that you can show it that basically traverse the entire collection using the different approaches. So, this is the one example which the iterator list the iterator method it is there.

(Refer Slide Time: 31:06)

Bulk Operation

NPTEL Online Certification Courses
IIT Kharagpur



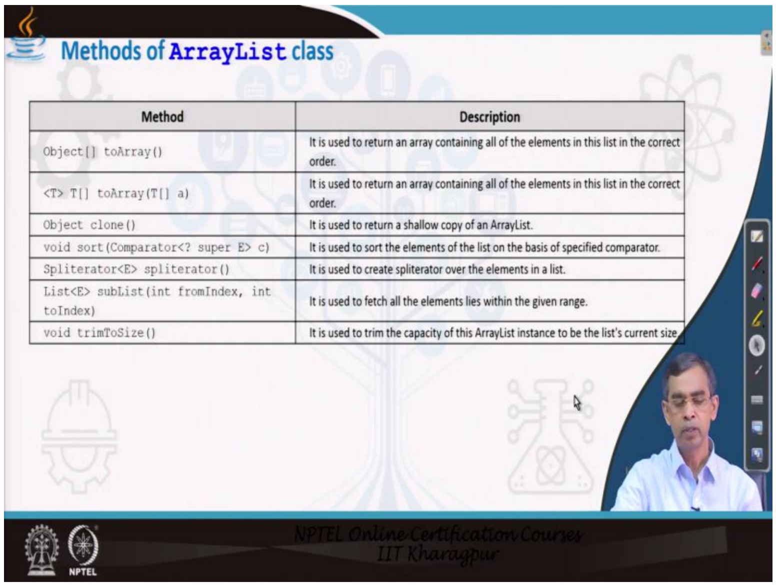
Bulk operations on ArrayList collection

- When working with ArrayList, you will sometimes want to obtain an actual array that contains the contents of the list. You can do this by calling toArray( ), which is defined by Collection. Several reasons exist why you might want to convert a collection into an array, such as:
  - To obtain faster processing times for certain operations
  - To pass an array to a method that is not overloaded to accept a collection
  - To integrate collection-based code with legacy code that does not understand collections

NPTEL Online Certification Courses
IIT Kharagpur

Now, there are certain method called bulk operation which can be applied on a collection. Clone is a one example we have discussed about for the bulk operation.

(Refer Slide Time: 31:17)



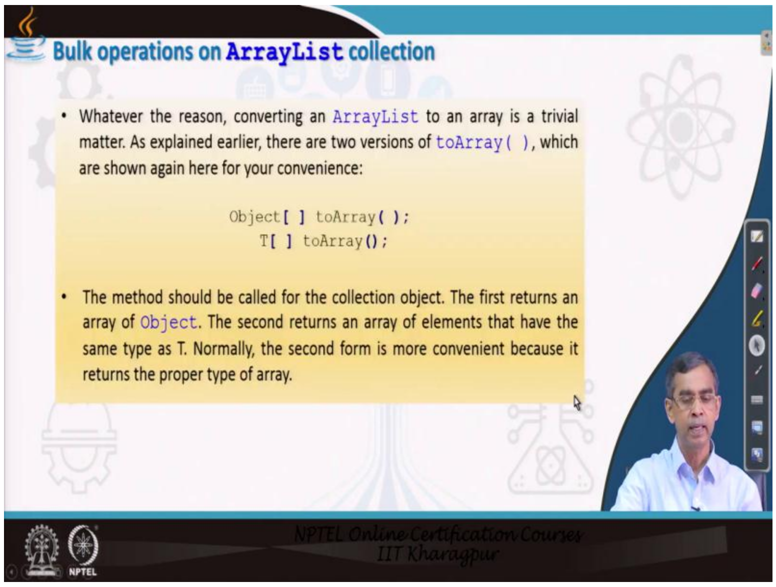
Methods of ArrayList class

Method	Description
`Object[] toArray()`	It is used to return an array containing all of the elements in this list in the correct order.
`<T> T[] toArray(T[] a)`	It is used to return an array containing all of the elements in this list in the correct order.
`Object clone()`	It is used to return a shallow copy of an ArrayList.
`void sort(Comparator<? super E> c)`	It is used to sort the elements of the list on the basis of specified comparator.
`Spliterator<E> spliterator()`	It is used to create spliterator over the elements in a list.
`List<E> subList(int fromIndex, int toIndex)`	It is used to fetch all the elements lies within the given range.
`void trimToSize()`	It is used to trim the capacity of this ArrayList instance to be the list's current size.

NPTEL Online Certification Courses
IIT Kharagpur

But in ArrayList, there are certain methods like here toArray, it basically convert any type of collection to an array. toArray given an array of any type can be converted also, clone is the method already you have familiar to these are there. Now, let us see a simple example to give the example of clone we are already familiar to. I will not give that example; I will give another example about two array method which is there.

(Refer Slide Time: 31:48)



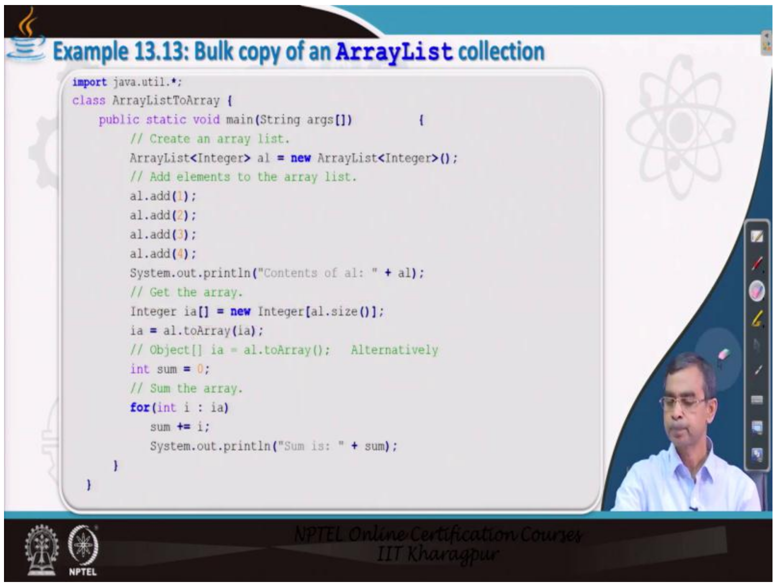
Bulk operations on ArrayList collection

- Whatever the reason, converting an ArrayList to an array is a trivial matter. As explained earlier, there are two versions of toArray( ), which are shown again here for your convenience:

```
Object[ ] toArray( );
   T[ ] toArray();
```

- The method should be called for the collection object. The first returns an array of Object. The second returns an array of elements that have the same type as T. Normally, the second form is more convenient because it returns the proper type of array.

NPTEL Online Certification Courses
IIT Kharagpur



Example 13.13: Bulk copy of an ArrayList collection

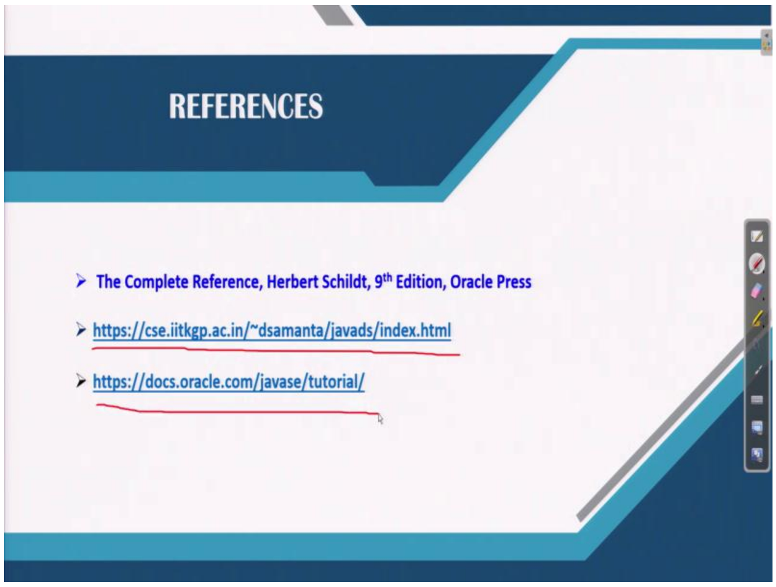
```
import java.util.*;
class ArrayListToArray {
    public static void main(String args[])         {
        // Create an array list.
        ArrayList<Integer> al = new ArrayList<Integer>();
        // Add elements to the array list.
        al.add(1);
        al.add(2);
        al.add(3);
        al.add(4);
        System.out.println("Contents of al: " + al);
        // Get the array.
        Integer ia[] = new Integer[al.size()];
        ia = al.toArray(ia);
        // Object[] ia = al.toArray();   Alternatively
        int sum = 0;
        // Sum the array.
        for(int i : ia)
           sum += i;
           System.out.println("Sum is: " + sum);
     }
  }
```

NPTEL Online Certification Courses
IIT Kharagpur

So, this example, these are example you can check the usefulness of the two array. First we create a1 is an array of type integers, this is basically the generic arrays, generic collection and here now we create another array of cloning this one, integer io a1 dot size, specifying the size of this array is this one and then two array method we call ia that means, it basically create a clone and then this one.

So, ia will be created and then it can be stored into there. So, it is basically copy and then some of the ia elements can be calculated and it can be printed. So, toArray method is basically if you create an array of integers and then the array can be stored into the collection that is basically the application that you can think about.

(Refer Slide Time: 32:45)



REFERENCES

- The Complete Reference, Herbert Schildt, 9th Edition, Oracle Press
- https://cse.iitkgp.ac.in/~dsamanta/javads/index.html
- https://docs.oracle.com/javase/tutorial/

You can have many more programs from the link that we have given here. So, this link you can consult to get many relevant program for your own practice and this is the one good document in the Oracle where you can get the different methods in details and their utility. These are the basically study that you can do it to understand the concept of ArrayList in your program and you can see how the different features those are there in java collection framework is handy in your program writing. Obviously more practice is required in order to have the full confidence on this. All the program that we have given there, you just practice in addition to many programs which is given there in the link also you can try. Thank you.