

Data Structure and Algorithms Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 26
Queue Using JCF

So, welcome Java collection framework is a fascinating facilities repository for implementation with implementation of many data structures. Queue is another important data structure is useful in many applications. So, Java collection framework has its own provision to support how the Queue can be used in your program.

(Refer Slide Time: 00:53)



So, today in this video, we are going to learn about how a Queue collection can be created using Java collection framework and thereby different operations, namely insertion, deletion, printing, and accessing, all these things can be done. Now, Java collections framework in addition to the simple Queue implementation has many variations.

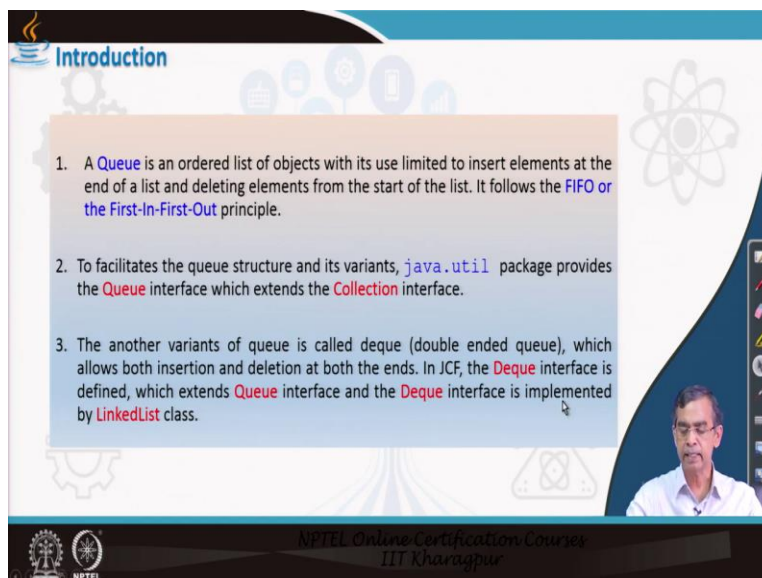
So, there are many variations, we will cover about 2 important variations, the Priority Queue and then Blocking Queue it is there. And then Array Dequeue is also 1. Anyway, so we will discuss about all these variations those are there.

(Refer Slide Time: 01:36)



So, let us start about first the collection framework concept and it is basically in the context of Queue. There are basically what are the different classes and interface who is basically Java developer considers and the Java developer define them, and so that programmer can import this and use it.

(Refer Slide Time: 01:56)

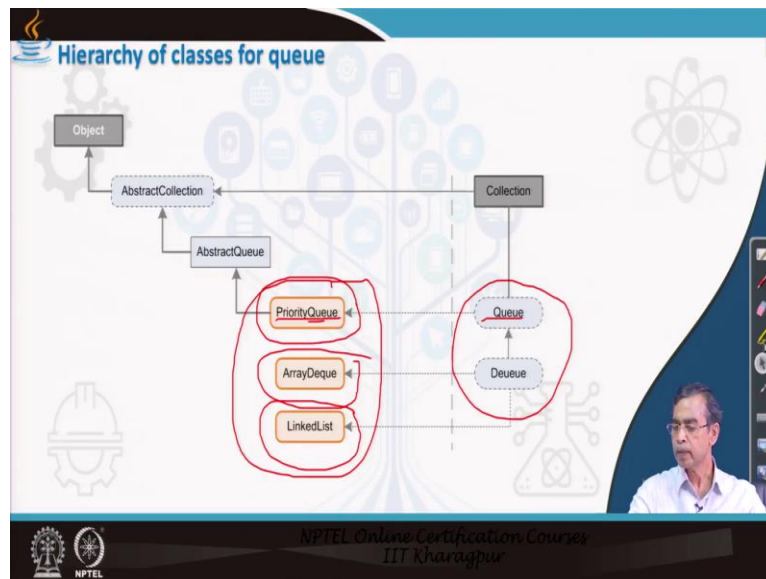


Now, here the Queue is basically a collection like the collections as you have learned about stack then array and then array list, vectors really like this. So, this is also a collection like other

collections we have already learned in this course till time. Now, this collection obviously is a different with different characteristics in the context that.

For example, stack is a collection which implements Last-In-First-Out basically policy, but Queue is basically the collection characterized with First-In-First-Out or FIFO characteristics. Now, here the Queue the Java collection framework has defined it in form of 2 classes, 2 interface rather Queue and the Deque.

(Refer Slide Time: 02:50)



Now, let us see exactly the details about the organization of this concept is there. Now, so as I told you Queue and Deque are the 2 interfaces are which are defined in the Java collection framework. These Queue and Deque interfaces mean they basically have that design plan for the different methods which basically should be there in order to implement Queue and Deque.

So, as you know interface needs to be implemented. So, in Java collection framework, this Queue is implemented by using a stacks are called Priority Queue. And Deque is implemented by using 2 classes the array Deque and linked list. Now, here you can one things. So, linked list basically can help you to implement Queue if you can limit insertion deletion in a restricted way, this is because you have already learned about the linked list Queue implementation using link list like.

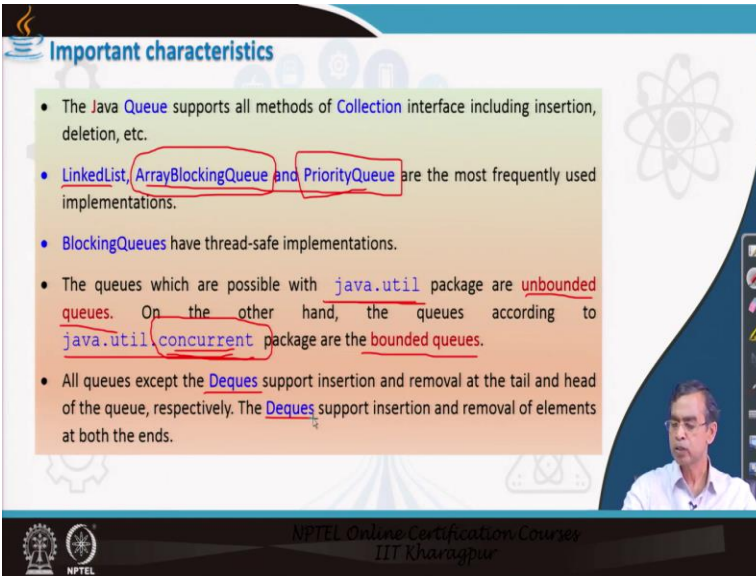
So, that the deletion should take place at the front and then insertion should take place at end like, so if we can use the link list and then accordingly we can restrict our insertion deletion operation, it basically implement, I mean achieve your Queue policy actually. So, that is why link list is basically is our implementation for that Deque. Now, here, so, these are the basically the 3 classes, which are there, which programmer can import into their program to take advantage of these.

Now, Priority Queue, Array Deque, link list as you see Priority Queue implements Deque means all the methods which are defined declared here is implemented here. And similarly, Deque all the methods which are declared here are defined in Array Deque and LinkedList. Now, in this video lectures we will try we will try to understand what are the different methods related to this Queue implementation and Deque implementation is there.

So, this is basically the Java collection framework class hierarchy. So far, the Queue and Deque is concerned. Now, you know Queue is basically simple queue, Deque is basically double ended Queue. This means that you can perform insertion or deletion at both end like that we have already discussed while we are discussing theory about the Queue structure. And Priority Queue is another variation that basically is that it can allow to remove an element according to the Priority of a element of an element.

So, Priority is there now we will see exactly how Priority Queue can be realized in this Java collection framework. So, it is basically interesting topics to learn and the topics is to adjust it I will try to cover as much as possible in a half hour duration like but anyway it will give a little bit firm point only so, that you can starting, you can start your I mean programming, what is called the learning objective here and then you can exercise them in full length, so it is obviously matter of is a more practices required other. Anyway so, this is you can consider the starting point so for that learning Java collection framework in regard to Queue.

(Refer Slide Time: 06:22)



Important characteristics

- The Java Queue supports all methods of Collection interface including insertion, deletion, etc.
- LinkedList, ArrayBlockingQueue and PriorityQueue are the most frequently used implementations.
- BlockingQueues have thread-safe implementations.
- The queues which are possible with java.util package are unbounded queues. On the other hand, the queues according to java.util.concurrent package are the bounded queues.
- All queues except the Deques support insertion and removal at the tail and head of the queue, respectively. The Deques support insertion and removal of elements at both the ends.

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us first start about that few important characteristics that the Queue is basically helped you. Now, as I have already told you so link list then Array Blocking Queue and Priority Queue, ok. We have discussed about the Priority Queue, the recent version of Java, they add another Array Blocking Queue for the thread safe application because if you want to write your program using multi threading, then the thread running the thread in a safe way or in a synchronized way.

So, Blocking Queue is basically allow out of many threads who are competing for a common Queue, so, that it can allow only 1 application to enter into the Queue here the Queue is a critical resource. So, it basically allows this one and this is basically Java dot util is basically your collection framework repository. So, in all programs whatever we will give a demo here, so you have to import Java dot util dot star or Java dot util dot link list or whatever class you want to have in your program.

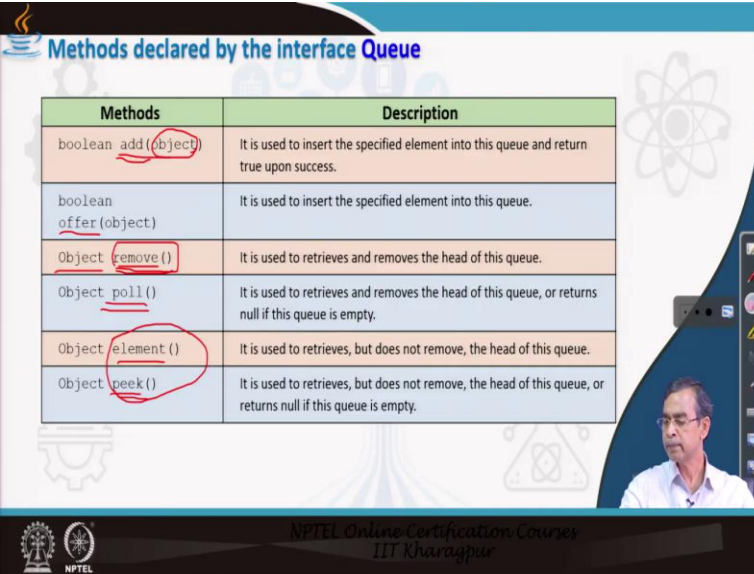
Now, here Java dot util is basically all the Queue implementation likes say Priority Queue, then Array Deque or LinkedList, they are basically called unbounded Queues and they are basically all implements using LinkedList concept. No array use because all the collections is basically to grow dynamically that is the philosophy behind any Java collection framework actually.

And so, these are unbounded. In contrast to unbounded there is a bounded Queues those are declared in another what is called a sub package called concurrent. Concurrent package will not

be covered in this course actually, but it is another part of this 1. So, it is a recent addition were there. But you can use the concept that is there in the same way all using the same method, but concurrent and this is a related to the multi threading programming.

Anyway, so these are the concept and then Deque also it is there. Deque as I told you, it supports in unison Java supports or GCF supports you for the Deque to be used in your program as well as. So, these are the basically total story so, far the Java collection framework and for the Queue implementation is concerned.

(Refer Slide Time: 08:40)



Methods	Description
boolean <u>add</u> (object)	It is used to insert the specified element into this queue and return true upon success.
boolean <u>offer</u> (object)	It is used to insert the specified element into this queue.
Object <u>remove</u> ()	It is used to retrieves and removes the head of this queue.
Object <u>poll</u> ()	It is used to retrieves and removes the head of this queue, or returns null if this queue is empty.
Object <u>element</u> ()	It is used to retrieves, but does not remove, the head of this queue.
Object <u>peek</u> ()	It is used to retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

Now, let us first discuss about the Queue. Queue is an interface but it has been implemented as a LinkedList or as a Priority Queue and then Array Blocking Queue also add a Deque also. Anyway, so we will see exactly how we can create an instance of a Queue and then add element into the Queue and perform different operations. Now, for the Queue structure. So, Queue is basically our data structure and according with their shape, the Queue data structures are basically defined with many methods, I have added I have discussed, add is basically the common method.

So, it has an argument of element to be added here is just like an Enqueue operation we can see. Now, you can note that in theory or in our programming implements our own who use Enqueue Deque concept, but here in the Java collection framework, they universally use add remove and

peek pole all these things are the common method for removing or in different operation, performing different operation.

So, add is basically Enqueue related and then here offer is also similarly, it is basically to Enqueue. So, there are many methods are defined for different purposes the same thing, add and offer actually mean both method will return false if it fails to insert elements otherwise return true. And then here remove is the one element. So, it basically equivalent to the Deque operation I mean delete from Queue and it returned the element deleted.

And it also ok if it fails to delete, then it returns null. Now, poll operation just like remove. It basically same as the poll, but in this case, it the remove if fails then it returns an exception. And in case of a poll, it also removed the elements which deletes from the Queue, but in that case, it should not throw any exception rather it should throw a null value. Null value indicates that nothing is deleted from the Queue.

And these are the 2 arrays for the accessing element without removing only to take which is the element at the front. So, element and peek now here again element and peek are the same operations because it will just simply access the topmost element, but they will not remove the element from the Queue.

Here element is basically if we know he does not find any elements in the Queue that means, if you apply elements for an empty Queue then in that case this element method will return an exception it is called a null pointer exception and in case of peek if it does not find any element that means Queue is empty it will return null.

So, these are that methods as you can see related to Enqueue Deque and simply accessing and these are the total stores for the Queue and if you can use and you can maintain a Queue and call all these methods and then perform your Queue strategic Queue application in your program. So, it is very simple and as such as no difficulty only you have to just see exactly how we can apply it.

(Refer Slide Time: 12:00)

Priority Queue with JCF

NPTEL Online Certification Courses
IIT Kharagpur

Constructors defined in the class PriorityQueue

Being an interface the queue needs a concrete class for the declaration and the most common classes are the `PriorityQueue` and `LinkedList`. The `PriorityQueue` class includes six constructors.

Constructors	Description
<code>PriorityQueue()</code>	The default constructor builds an empty queue. Its starting capacity is 11.
<code>PriorityQueue(int capacity)</code>	This constructor builds a queue that has the specified initial capacity.
<code>PriorityQueue(Comparator<? super E> comp)</code>	This constructor specifies a comparator.
<code>PriorityQueue(int capacity, Comparator<? super E> comp)</code>	This constructor builds a queue with the specified capacity and comparator.
<code>PriorityQueue(Collection<? extends E> c)</code>	This constructor creates a queue which can hold a generic collection.
<code>PriorityQueue(PriorityQueue<? extends E> c)</code>	This constructor create queues that are initialized with the elements of the collection passed in c

NPTEL Online Certification Courses
IIT Kharagpur

Now, this is about the Queue same as Priority Queue, we can define a Queue as a Priority Queue as I told you Priority Queue means it will not follow the insertion and deletion operation strictly in the last in the first in first order rather it will according to the Priority of the process. Now, here Priority in order to judge it, whenever you add automatically this Priority Queue collection take care about to being to the element which has the highest Priority at the front so, that it can be removed first.

So, it basically violates the FIFO or in that sense actually, and then internally the Java collection framework can take care about whenever you add element and being this element at the front.

So, what I can say is that if it basically inserts an element at the front, depending on if it is the highest priority is there or basically in the order depending on its priority of value it is there. So, how do you can say all elements in Priority Queue, they are in that descending order of their Priority Queue values, it is all now here you see, in order to maintain the order, we have to compare.

Now, for the different types of structure, I mean element or object, obviously comparable on method needs to be implemented, we will discuss about whenever, if you want to use the Priority Queue, you have to define a comparable method. For some types, comparable methods are already defined like integer, float or say string. But for some many user defined types a book, student that comparable method is not defined.

Anyway, let us start. First of all, what is a constructor that is there to define or to create an instance of a Priority Queue that means to create a Queue, so this is basically default constructor. You can create a Priority Queue mentioning a capacity but you should not be worried about you can start with initial capacity. And then you can then your Queue structure will grow automatically, but in case of default Priority Queue a default size you will be taken, it will take place will take place it is basically the size is 11.

Now, in the Priority Queue you can define a comparator I will discuss about how your comparator method can be declared and here is also capacity and then comparator both if you want to add it is basically default Priority Queue of size 11 but with comparator declaration here are no comparator because if you think that for your data structure or element that you want to maintain a Queue no need to defend any comparator because then you can use these other constructors, but if you need a comparator then these are the things.

Now, you can create a Priority Queue initialize is a collection that means the existing an array can be added into the PriorityQueue and with this array you can add it and here also Priority Queue can be used to create another Priority Queue. These are the different constructs that you can think for creating a Queue structure in your program, we will learn about how all those Queue structure can be created.

(Refer Slide Time: 15:09)

Constructors defined in the class PriorityQueue

Being an interface the queue needs a concrete class for the declaration and the most common classes are the `PriorityQueue` and `LinkedList`. The `PriorityQueue` class includes six constructors.

Constructors	Description
<code>PriorityQueue()</code>	The default constructor builds an empty queue. Its starting capacity is 11.
<code>PriorityQueue(int capacity)</code>	This constructor builds a queue that has the specified initial capacity.
<code>PriorityQueue(Collection<E> c)</code>	This constructor creates a queue which can hold a generic collection.
<code>PriorityQueue(PriorityQueue<E> c)</code>	This constructor create queues that are initialized with the elements of the collection passed in c

Note:
In all cases, the capacity grows automatically as elements are added with queue structure.

NPTEL Online Certification Courses
IIT Kharagpur

And as I said that Queue structure in case of Java collection framework will grow dynamically because it follows `LinkedList` implementation in it.

(Refer Slide Time: 15:16)

Methods under the class PriorityQueue

- All the methods declared in the `Queue` interface are fully defined in the `PriorityQueue` class.
- Further, since it is a subtype of `Collections` class, it inherits all the methods of it, namely `size()`, `isEmpty()`, `contains()`, etc.

NPTEL Online Certification Courses
IIT Kharagpur

Now, fine all the methods in addition to the method that we have discussed alike so for `PriorityQueue` size is empty content, those methods are basically declared in a collection framework all are applicable to the, because they are basically they extents collection framework `PriorityQueue`, `Queue`, `Dequeue`, everything is there. Anyway, so, there are many methods out there, but those methods are most relevant to the `Queue` operation we have mentioned here.

(Refer Slide Time: 15:46)

Basic operations for queue collection

Queue of JCF supports all operations necessary to maintain a queue structure, which are summarized as:

Operation	Throws exception	Special value
Insert	add(e)	offer(e)
Remove	remove()	poll()
Examine	element()	peek()


NPTEL Online Certification Courses
IIT Kharagpur

Now, let us understand about some application where we can use the Java collection framework supports to write our own program. Now, we will come into the programming issues. Now, so far the programming study is concerned said that regarding the Queue, you have to just, majorly these are the 3 operations to be carried out, insert, remove and examine. Insert means we have to add.

So, here add elements and offer are the 2 methods are available to perform insertion operation. Now, the difference between the 2 operation is that add throw exception in case of add fails to perform it maybe my type is not proper or LinkedList is not able to allocate memory for your Queue whatever it is there, but in case of offer it basically null. Similarly, remove for the Remove operation.

There are 2 methods remove and poll remove basically fails if it is an empty or supple like this then it throw exception but poll through return null. For the examine that means only for accessing there are 2 methods element and peek. Like element, it throw exception and peek special value. So, these are the few methods that we will going to use in our program and using the JCF Java Collection Support facilities in our program. So, let us see how the program can be returned utilizing the Java collections framework it is there.

(Refer Slide Time: 17:21)



Creating a Queue

NPTEL Online Certification Course
IIT Kharagpur

Example 26.1: Creating a queue

```
/* To create a queue using PriorityQueue class along with some common operations.
Queue of string objects. */

public class QueueCreateExample1 {
    public static void main(String[] args) {
        Queue<String> queue = new PriorityQueue<>();
        queue.add("one"); // Adding some elements into the queue
        queue.add("two");
        queue.add("three");
        queue.add("four");
        System.out.println(queue); // Display the contents in queue
        queue.remove("three");
        System.out.println(queue);
        System.out.println("Queue Size: " + queue.size());
        System.out.println("Queue Contains element 'two' or not? : " +
            queue.contains("two"));
        queue.clear(); // To empty the queue
    }
}
```

NPTEL Online Certification Course
IIT Kharagpur

Example 26.1: Creating a queue

```
/* To create a queue using PriorityQueue class along with some common operations.
Queue of string objects. */

public class QueueCreateExample1 {
    public static void main(String[] args) {
        Queue<String> queue = new PriorityQueue<>();
        queue.add("one"); // Adding some elements into the queue
        queue.add("two");
        queue.add("three");
        queue.add("four");
        System.out.println(queue); // Display the contents in queue
        queue.remove("three");
        System.out.println(queue);
        System.out.println("Queue Size: " + queue.size());
        System.out.println("Queue contains element 'two' or not? : " +
            queue.contains("two"));
        queue.clear(); // To empty the queue
    }
}
```

Now, first let us start about how we can create a Queue and then perform any operation on it. And this is the program you can look at little bit carefully. So, this is a program the objective of the program is to give a demo create Queue. And here you see we create a Queue. This is our Queue.

But using Priority Queue, you can understand that if we can add the element will be added into the Queue according to priority. And here we just want to create the Queue of string that means elements those are to be added into the Queue or Queue elements are string type, fine. This is the way that we can create a Queue in this case we created a Queue like. So, you can see this is a Priority Queue concept and this Queue will store string as element in it.

Now, we can use many methods, So, far the insertion is concerned as I told you add and offer. So, here we are using add method only. Now, these are the 4 statement we are adding the method 1, 2, 3, 4. Now, if you add it then it basically Queue will resolve that, which has the highest priority to be added here because it is the Queue is a Priority Queue implementation. So, it will according the Priority Queue it will add.

Now, it is basically in the descending order as I said maybe the t will come first and then t w o, 3 then o, f, whatever it is basically the lexicographic order you can see this is basically ordering in the dictionary order anyway. So, Queue will be added automatically if you give a print Queue here for example, you can see, which is the order? The element that you have added it basically in the Queue.

So, this basically the simple println statement called for the collection Queue to print all the elements that is there, but you can note here that they are not necessarily in the same order as you have added it. Now, here after creating after entering or adding some elements into the Queue, we perform a remove operation it basically delete a Queue and whenever you pass an element it basically delete a particular element from the Queue you can see that, remove operation allow you to remove a particular elements or simply remove without any argument it will remove that top element actually that mean element at the front.

Now, after removing you can print the Queue you can say that how whether this method works for you or not. And then you can print the Queue size, size is defined for each collection framework, it basically says how many total elements are there in this case 3 elements are there. So, this will print 3 like and then now, here contents on method we are already familiar to basically search for an element, those are defined in collection and as they are defined in collection Queue is a child class.

So, that method also you can use it so, it check that whether 2 is there in the Queue or not. And here you see clear method we are familiar to this clear method in several occasions we have used you can understand what this method will do, this method will delete all elements from the Queue. So, Queue will be now after that if you print Queue it will basically will not print anything. So, so this is the idea about how we can create a Queue and then using Priority Queue and then different operations can be applied to the Queue.

(Refer Slide Time: 21:09)

```
/* To create a queue using LinkedList class along with some common
operations. Queue of integer numbers. */

import java.util.LinkedList;
import java.util.Queue;

public class QueueCreateLinkedList {
    public static void main(String[] args) {
        // Declaration of a queue using LinkedList class
        Queue<Integer> q = new LinkedList<Integer>();
        // Adds elements {0, 1, 2, 3, 4} to queue
        for(int i=0; i<5; i++)
            q.add(i);
        // Display contents of the queue.
        System.out.println("The queue contains :"+q);
    }
}

// Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

```
/* To create a queue using LinkedList class along with some common
operations. Queue of integer numbers. */

import java.util.LinkedList;
import java.util.Queue;

public class QueueCreateLinkedList {
    public static void main(String[] args) {
        // Declaration of a queue using LinkedList class
        Queue<Integer> q = new LinkedList<Integer>();
        // Adds elements {0, 1, 2, 3, 4} to queue
        for(int i=0; i<5; i++)
            q.add(i);
        // Display contents of the queue.
        System.out.println("The queue contains :"+q);
    }
}

// Continued to next ...
```

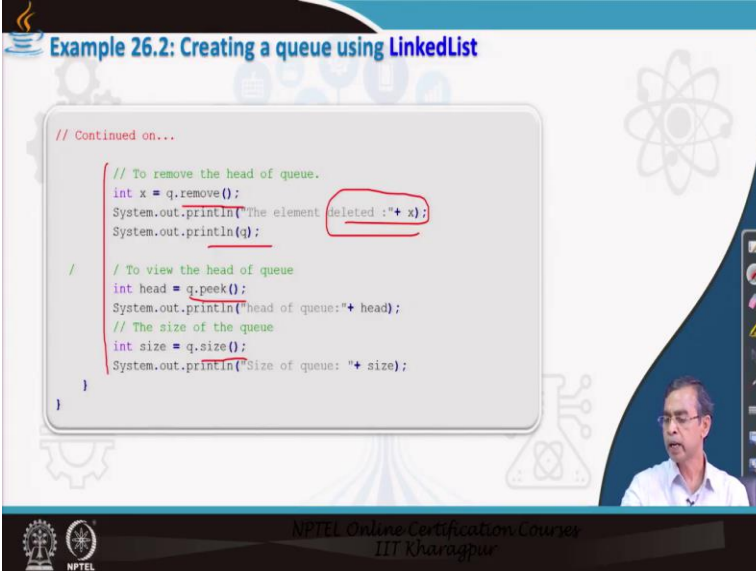
NPTEL Online Certification Courses
IIT Kharagpur

Now, let us consider another one example. In this, this is another example demonstration Queue create LinkedList, we have to use a LinkedList implementation, earlier we used the Priority Queue to create a Queue. Now, in this case, we create another Queue say q. And this we declare the Queue because it is basically interface, but we want to create an object of type LinkedList. So, this basically and then integer is basically the type of the element that we want to store into our Queue. So, now we are going to have another Queue, the name of the Queue is q and then we want to initialize the Queue with some numbers say 0, 1, 2, 3, 4 in the same order as 0 to this 1.

So, we can use just add method call the add method within a for loop integer i equals 0, i less than 5, and then we can add each numbers into the Queue. So, what you can see that Queue will be loaded or the elements will be entered into the Queue, the 5 different is it starting from 0 and the last number is 4. And finally, the Queue is created Queue is loaded or initialized we can say and if we print this is a print statement if we call and then it basically clean the Queue.

So, entire Queue element will be printed on the screen. So, these basically example gives you how Queue can be created using LinkedList earlier example was creating a Queue using Priority Queue. Now, so this program has, fine, we have created a Queue and added some element on the same Queue, we can remove or you can delete some elements from there also.

(Refer Slide Time: 23:02)



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

```
// Continued on...  
  
// To remove the head of queue.  
int x = q.remove();  
System.out.println("The element deleted :"+ x);  
System.out.println(q);  
  
/ / To view the head of queue  
int head = q.peek();  
System.out.println("head of queue:"+ head);  
// The size of the queue  
int size = q.size();  
System.out.println("Size of queue: "+ size);  
}
```

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

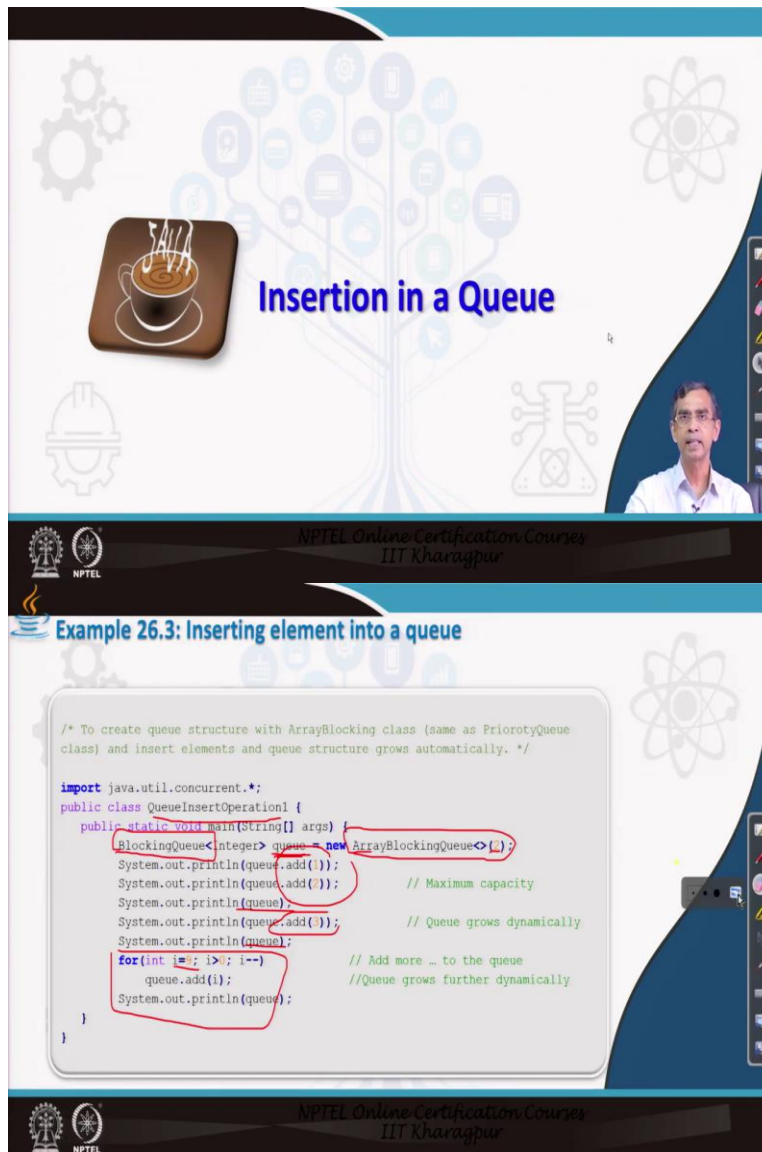
Here is a continuation of the same program here we can just apply a remove method. And then after removing you can see which elements has been removed from the Queue and then you can clean the Queue and so on. So, peek is also not removing only check the front most element size also can be called and so on. So, here actually the objective is that how the different operations can be performed. Whatever be the way you can create any content in the Queue no issue.

All methods are basically unified method like. So, that is why Java developer has taken many years to define all methods in the respective class in that way. So, the programmer just need not to be bothered about what is a additional method for this structure or for this collection. It is

basically name is same, but operation according to the different class. So, that operation is basically at the back the Java developer has done for you as a facility.

Anyway, so, we have learned about how the Queue can be created and then perform the different operations on the Queue. So, far we have learned about creating a Queue using 2 structures like Priority Queue and LinkedList.

(Refer Slide Time: 24:18)



The image shows a presentation slide with a blue and white theme. At the top, there is a title "Insertion in a Queue" next to an icon of a coffee cup. Below the title, there is a small video inset of a man speaking. The slide is part of an NPTEL Online Certification Course from IIT Kharagpur. The main content is a code example titled "Example 26.3: Inserting element into a queue". The code is in Java and demonstrates how to create a queue using the ArrayBlockingQueue class and insert elements. Red circles highlight the queue creation and the insertion loop.

```
/* To create queue structure with ArrayBlocking class (same as PriorityQueue
class) and insert elements and queue structure grows automatically. */
import java.util.concurrent.*;
public class QueueInsertOperation {
    public static void main(String[] args) {
        BlockingQueue<Integer> queue = new ArrayBlockingQueue<>(4);
        System.out.println(queue.add(1));
        System.out.println(queue.add(1)); // Maximum capacity
        System.out.println(queue);
        System.out.println(queue.add(1)); // Queue grows dynamically
        System.out.println(queue);
        for (int i=0; i<> i--); // Add more ... to the queue
            queue.add(i); //Queue grows further dynamically
        System.out.println(queue);
    }
}
```

Example 26.3: Inserting element into a queue

```

/* To create queue structure with ArrayBlocking class (same as PriorityQueue
class) and insert elements and queue structure grows automatically. */

import java.util.concurrent.*;
public class QueueInsertOperation1 {
    public static void main(String[] args) {
        BlockingQueue<Integer> queue = new ArrayBlockingQueue<>( );
        System.out.println(queue.add( ));
        System.out.println(queue.add( )); // Maximum capacity
        System.out.println(queue.add( )); // Queue grows dynamically
        System.out.println(queue);
        for(int i=0; i>0; i--) // Add more .. to the queue
            queue.add(i); //Queue grows further dynamically
        System.out.println(queue);
    }
}

```

Note: add() returns true if the element is inserted into queue successfully else it return false.

Now, let us study few more programming experience here. First of all, we are again repeating insertion into a Queue with a different method. Here is another example. So, these basically program give a demo about the insertion operation. First of all we have to create. Again here in this example, we create another Queue, but using this we are using Array Blocking Queue that is another class it is declaring the latest person. Initially we define the size 2. If you do not mention the size, default size 11 will be used. And here we are using Blocking Queue.

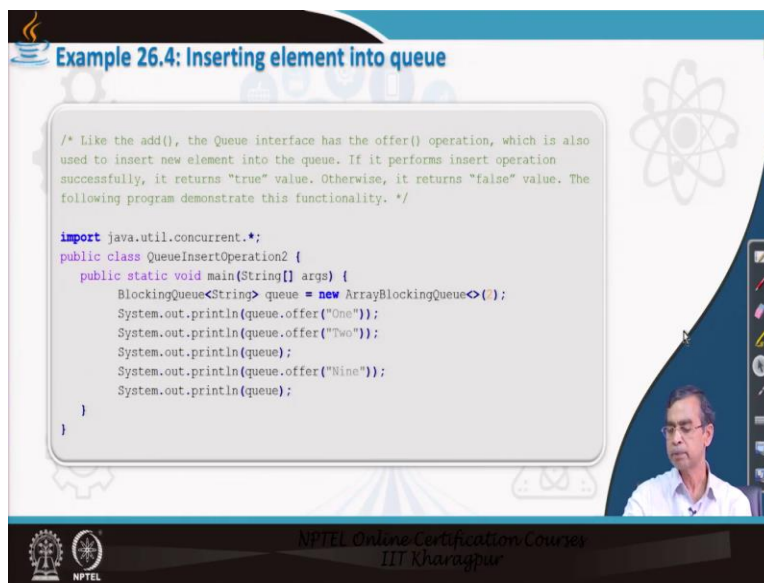
This particular Queue you can use if you want to write a threading program or multi threading program for concurrent execution. We will not do any threading program I am here, but I am just giving for a simply an illustration that how that kind of structure can be created. So, here Queue is basically the Blocking Queue; Blocking Queue means that if 2 or more threads attempt to enter into the Queue or try to operate on the same Queue means 2 threads on to insert same Queue or delete from the same Queue at the same time, then which will express first and second.

So, Blocking Queue is basically control the execution. So, it basically enforce the mutual exclusion in the sense that if the 2 competing methods are applying for a particular structure, then it will allow only 1 method to enter into the structure that means it will execute only 1 method at a time, ok. Anyway, so this concept is basically a matter of multi threading concepts are there.

Now, whatever we so Queue is basically the Blocking Queue in this case, now in this Blocking Queue, we are performing add operation print, it is as usual earlier and this 1 and so we are

adding few more elements and if you can go in case of 9, you can say 100 also no problem Queue will grow automatically. So, this way Queue will give will insert the Queue will be maintained and we can see that Queue will be used and many operations on Queue structure can be exhibited.

(Refer Slide Time: 26:33)



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

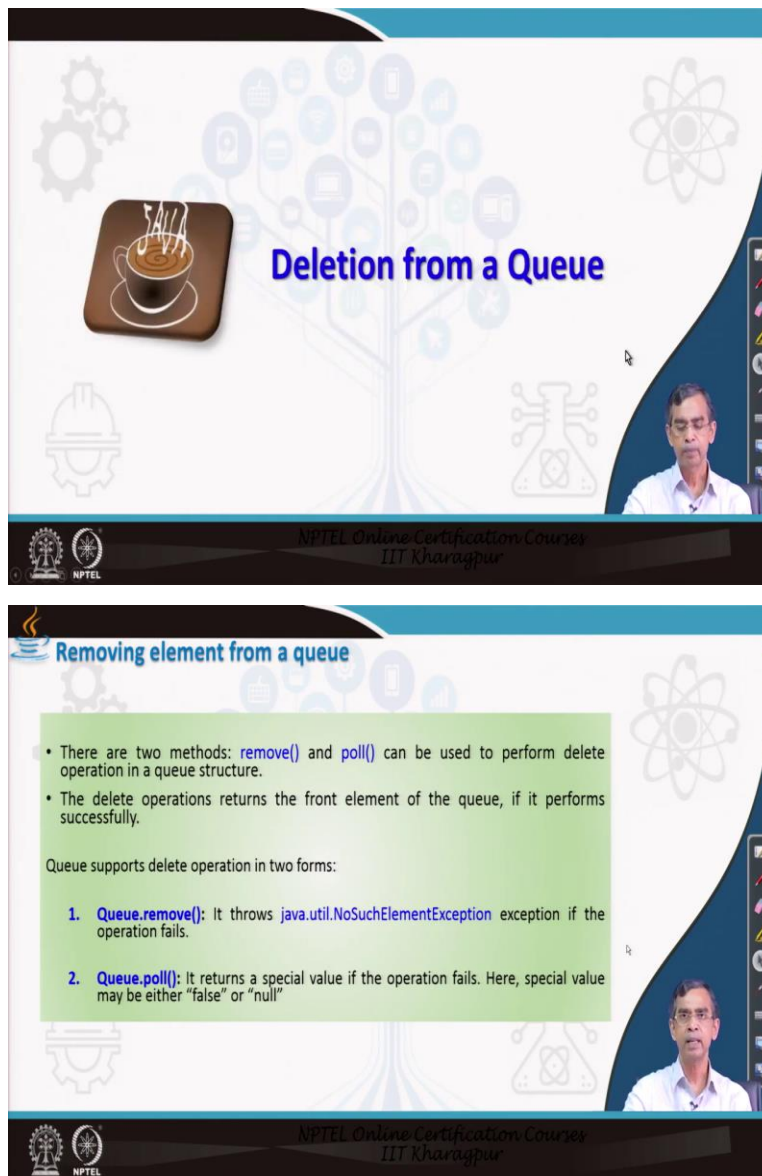
```
/* Like the add(), the Queue interface has the offer() operation, which is also
used to insert new element into the queue. If it performs insert operation
successfully, it returns "true" value. Otherwise, it returns "false" value. The
following program demonstrate this functionality. */

import java.util.concurrent.*;
public class QueueInsertOperation2 {
    public static void main(String[] args) {
        BlockingQueue<String> queue = new ArrayBlockingQueue<>(10);
        System.out.println(queue.offer("One"));
        System.out.println(queue.offer("Two"));
        System.out.println(queue);
        System.out.println(queue.offer("Nine"));
        System.out.println(queue);
    }
}
```

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

Now, let us see few more operations there, this is again insertion operation again Blocking Queue and this is for the string, the same thing that we have just take a have used this program for your practice only. So, different way how the Queue can be created that has been shown here actually.

(Refer Slide Time: 26:51)



Deletion from a Queue

NPTEL Online Certification Courses
IIT Kharagpur

Removing element from a queue

- There are two methods: `remove()` and `poll()` can be used to perform delete operation in a queue structure.
- The delete operations returns the front element of the queue, if it performs successfully.

Queue supports delete operation in two forms:

1. **Queue.remove():** It throws `java.util.NoSuchElementException` exception if the operation fails.
2. **Queue.poll():** It returns a special value if the operation fails. Here, special value may be either "false" or "null"

NPTEL Online Certification Courses
IIT Kharagpur

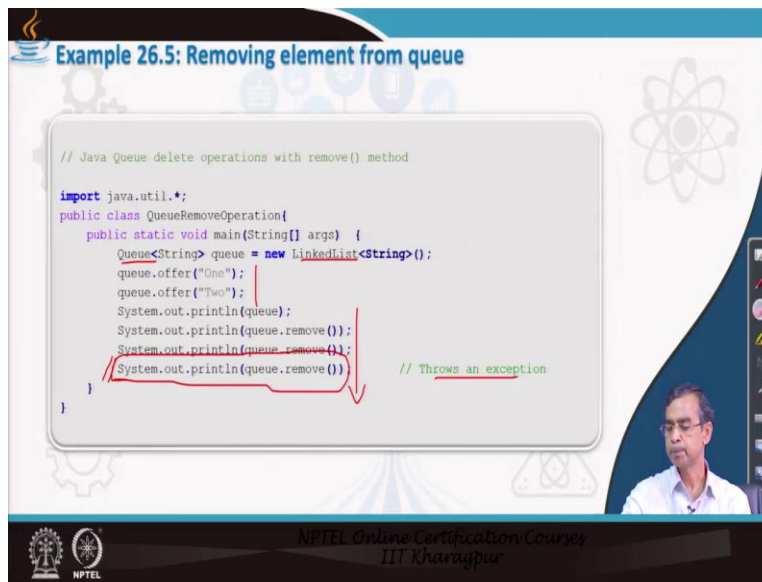
And deletion similar to the insertion, deletion of partition by means of 2 methods as I have already discussed remove and poll. The 2 methods are same that means it they will remove the first a topmost element or front element from the Queue, but the difference is that 1 method return an exception whereas the other method return simply a Boolean flag.

(Refer Slide Time: 27:16)

Example 26.5: Removing element from queue

```
// Java Queue delete operations with remove() method

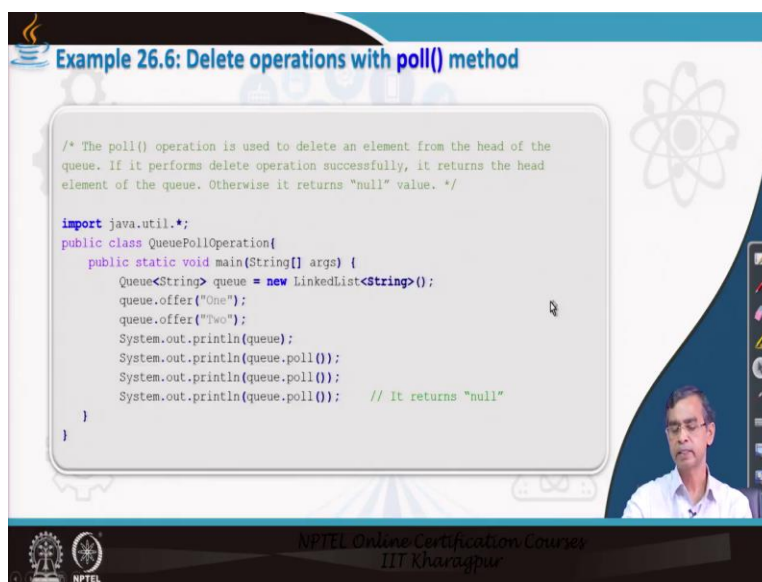
import java.util.*;
public class QueueRemoveOperation{
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<String>();
        queue.offer("One");
        queue.offer("Two");
        System.out.println(queue);
        System.out.println(queue.remove());
        System.out.println(queue.remove());
        System.out.println(queue.remove()); // Throws an exception
    }
}
```



Example 26.6: Delete operations with poll() method

```
/* The poll() operation is used to delete an element from the head of the
queue. If it performs delete operation successfully, it returns the head
element of the queue. Otherwise it returns "null" value. */

import java.util.*;
public class QueuePollOperation{
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<String>();
        queue.offer("One");
        queue.offer("Two");
        System.out.println(queue);
        System.out.println(queue.poll());
        System.out.println(queue.poll());
        System.out.println(queue.poll()); // It returns "null"
    }
}
```



Now, here is a program for you actually the Queue removal operation and we create a Queue using LinkedList here which is a Queue we add few elements and then remove elements and then you can see at this point, whenever you call the remove method, because after we have only 2 elements, and then third elements we are attempting to remove so it will throw an exception for this.

If you are not, if you do not uncomment then you can it will run without any exception, but if you give a comment, then it will run without any exception if you uncomment then you can see. Subprogram you can just check that, how it is working for you. So, this is about deletion

operation. And they are... And this is the another illustration for the same, but using the pole method, you can check it.

(Refer Slide Time: 28:18)

The image shows two slides from an NPTEL presentation. The top slide is titled "Traversing a Queue" and features a coffee cup icon. The bottom slide is titled "Example 26.7: Queue with user-defined objects" and displays Java code for a "Book" class that implements "Comparable<Book>". The code includes fields for bookId, name, author, publisher, and quantity, along with a constructor and a comment indicating that methods are defined next.

```
import java.util.*;

class Book implements Comparable<Book> {
    int bookId;
    String name;
    String author;
    String publisher;
    int quantity;

    public Book(int id, String name, String author, String pub, int qty) {
        bookId = id;
        this.name = name;
        this.author = author;
        publisher = pub;
        quantity = qty;
    }

    // Methods under this class are defined next

    // Continue to next ...
}
```

And then traversing a Queue. Traversing a Queue is basically printing the entire Queue. It is many methods out there, we will just simply give an example about. So, traversing method I just do not want to discuss simply println works for you. There are, again many standard traversal method, for which we will discuss separately, when all collections are covered. That is, I came to detail it is right now.

Anyway, so those things again, after learning those topics, actually you can come back to this 1 and then perform many, what is called the traversal operations on different Queue structure the different way you can create it, it will work for you. Anyway, let us come to on good example here, as I told you, so far the Priority Queue is concerned, if you want to maintain a Queue with your own data type for which no comparable method is there.

Because, comparable method is must to define for a Priority Queue in order to add because that Priority Queue will take this comparable method to check which is in basically order that mean which is in higher than the other sort of sort of things is there. Now, here we first take an example, first of all, we define 1 class of our own and let the name of the class be book. So, this is basically user defined class that we are going to implement it.

And this class has the different field it is here. These are different attributes of the class object. And this is the constructor that we use. So, here are basically the different fields, member elements and that the constructor is there. And finally, we will give some methods to define the different operation for example, print book, read book, whatever it is here. Now, here I just want to mention what so, book implements comparable here the comparable is a one class which is defined in Java in the object class it is basically it is defining util.

So, we just a comparable interface. So, this method have to be declared for this class so, that this comparable method can complete that if 2 books are given, then how the 2 books are comparable. So, 2 books book 1 and book 2, so, which should mean first order then next order and whatever it is there. Now, let us see other methods particularly the comparable method we need to define another method like print book or read book or delete book all those things you can define. I just do not want to discuss it here again.

(Refer Slide Time: 31:01)

Example 26.7: Queue with user-defined objects

```
// Continue on...  
  
// Defining the compareTo() method  
public int compareTo(Book b) {  
    if (bookId > b.bookId)  
        return 1;  
    else  
        if (bookId < b.bookId)  
            return -1;  
        else  
            return 0;  
}  
} // End of class Book  
  
// Continue to next page ...
```

NPTEL Online Certification Course
IIT Kharagpur

Example 26.7: Queue with user-defined objects

```
// Continue on...  
  
// Defining the compareTo() method  
public int compareTo(Book b) {  
    if (bookId > b.bookId)  
        return 1;  
    else  
        if (bookId < b.bookId)  
            return -1;  
        else  
            return 0;  
}  
} // End of class Book  
  
// Continue to next page ...
```

NPTEL Online Certification Course
IIT Kharagpur

Now, let us come to the discussion about the comparable method here is basically I declare the comparable because the comparable method actually we required to define the compareTo method actually in the comparable class. So, it is basically compareTo is a method is basically for we are basically implementing the compareTo method which is there in the comparable class actually.

Now, it basically reads an argument, so, this is the if we call these compareTo method for the current object with argument is another object in basically compare the 2 books actually here. Now, you see what we need here. The comparable method we can declare and you see in the

book that different attributes are there, we can compare 2 books with 1 attribute say bookId. So, here is basically we define that ok how it can be comparable.

So, bookId of the current book so we can write this bookId is greater than b dot bookId means the first bookId then we can say that it is returned 1. 1 means 2 otherwise it returned false and it is minus 1. So, it is minus 1. And if they are not comparable it returned 0, these kinds of things or so 1 and minus 1 is required for returning and that is the defined about the comparable class concept it is there and we just have to give a sip to the compareTo method and that we did here.

So, this completes the book class which comparable declaration now. Once it is declared, we will be able to create a Priority Queue if we do not extend the comparable but we will not be able to keep use the Priority Queue for this object it will return an exception error. But any simple Queue can be created for that object because for the Queue object, no need to have the comparable method there.

(Refer Slide Time: 33:00)

Example 26.7: Queue with user-defined objects

```
public class QueueCreateDemo {
    public static void main(String[] args) {
        PriorityQueue<Book> queue=new PriorityQueue<Book>();

        Book b1=new Book(111,"Joy with Java","Debasis Samanta","Elsevier",1);
        Book b2=new Book(222,"Complete Java","Herbert Schildt","Oracle",4);
        Book b3=new Book(333,"Headstart Java","Feroozan","O'Reilly",4);

        //Adding books to the queue
        queue.add(b1);
        queue.add(b2);
        queue.add(b3);
        System.out.println("Traversing the queue elements:");

        //Traversing queue with for-each loop
        for(Book b:queue) {
            System.out.println(b.bookId+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
        }
    }
}
```

// Continue to next ...

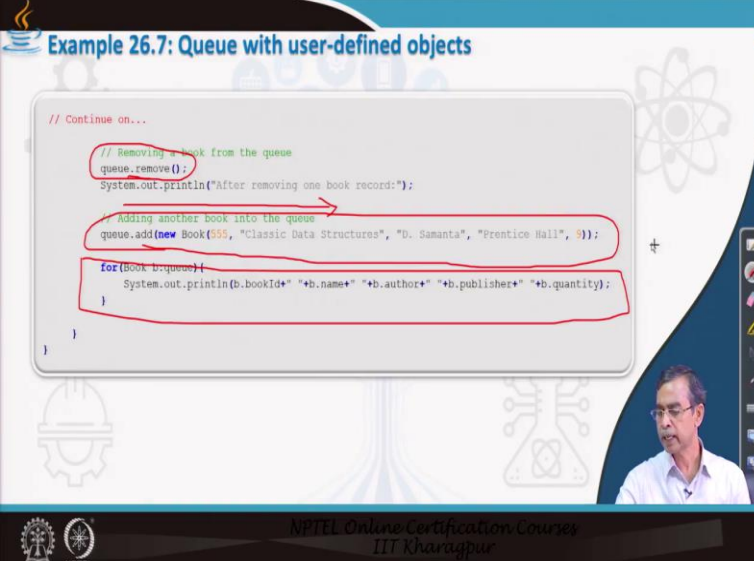
NPTEL Online Certification Courses
IIT Kharagpur

Anyway, so for the Priority Queue it is important, otherwise it is not so important. Now, let us come to again, using this book I want to, we want to maintain a Queue with a number of books in it, so, this is an example demonstration propped up. We can create a Queue, but this Queue we create within our own class the book. So, this is a Priority Queue and Priority Queue initially the default size 11.

Now, we add we create 3 book object as it is here. So, b1, b2 and b3 are the 3 book objects are created. And then the 3 book objects are added in this order you can add in the other order also and you can check it how it works. So, I gave I left is an exercise to do the same, but in the different order and check the program. So, here you can again print the method. Here is basically the for-each loop we have used to print the Queue.

So, is a book B in Queue and then print the different attributes. So, this is the way that you can print the entire Queue also. Now so, so in this program what do you have done is that we have created 3 objects all 3 objects are created and then we have print the Queue, fine. Now, let us see how the different operations that we can apply on these Queue objects and how they work it.

(Refer Slide Time: 34:20)



Example 26.7: Queue with user-defined objects

```
// Continue on...  
// Removing a book from the queue  
queue.remove();  
System.out.println("After removing one book record:");  
// Adding another book into the queue  
queue.add(new Book(55, "Classic Data Structures", "D. Samanta", "Prentice Hall", 9));  
for(Book b:queue){  
    System.out.println(b.bookId+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);  
}
```

The slide features a code editor window with the above code. Red annotations highlight the `queue.remove()` call, the `queue.add()` call, and the `for` loop. A red arrow points from the `queue.remove()` line to the `queue.add()` line. The slide also includes a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

So, here we just simply call the remove method. If it is empty then or definitely throw an exception it is not empty. And after removing again you can call the println method to print it and you can check it so print. Then you can add another book I have used another book into the Queue structure and again, you can print the Queue structure and you can see after adding this one, so you now I hope you are a little bit ready to take the program and twist a twist it in different way and check that how it is working to accompanies many operations related to Queue like insertion, deletion and accession. So, this is the Queue structure that you can think for this concept and this is the idea about it.

(Refer Slide Time: 35:16)

Copying a Queue

Example 26.8: Copying an array of objects to a queue

```
/* To convert a Java array to Queue using addAll() method defined in the
Collection class. */

import java.util.*;

public class ArrayToQueue {
    public static void main(String[] args) {
        // Create an array of String objects
        String city[] = {"CCU", "DEL", "BLR", "MUM", "GAU"};
        // Declare a queue
        Queue<String> queue = new PriorityQueue<String>();
        // Copying the array to Queue
        Collections.addAll(queue, city);
        // Printing the array
        System.out.println("Array : " + city);
        System.out.println();
        // Printing the queue
        System.out.println("Queue : " + queue);
    }
}
```

Now, I will quickly cover few more bulk operation that JCF supports for the Queue structure. They are copying a Queue. Here basically we will see how a collection in this case how a collection that is maybe an array or vector or whatever it is there can be added to a Queue. Now, let us see this example.

First of all, we have to create a collection, it is basically an array is a collection and we declare a Queue here and then you see we call addAll method. So, addAll method is declared in collection objects. And that method can be defined for the Queue. With this method. So, you can call that

addAll method passing to argument Queue and city, what it will do is that it will basically copy all the array into this Queue and then we can print the 2 array as well as Queue and see that that Queue has been created.

Now, here onwards you can perform many operations related to addition I mean insertion deletion or accessing for this Queue you will see it is working for you, but it is initialized with these are the elements there, ok. So, I just advise you to start writing few more quotes here related to insertion, deletion, size, capacity, so many things are there and you can test it. So, this basically example shows that given an collection given a collection, maybe any type how using addAll method, you can initialize a Queue and the same Queue can be used or that is ready for your program.

(Refer Slide Time: 37:08)

Example 26.9: Copying a queue to an array

```
/* To copy a Queue object to an array using toArray method defined in the
Collection class. */
import java.util.*;
public class QueueToArray {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<String>(); // Creating a queue
        queue.add("Jio");
        queue.add("123");
        queue.add("John");
        queue.add("Jaya");
        queue.add("Jim");
        queue.add("!@#%");
        // Copying the queue to an array of string objects
        String friends[] = queue.toArray(new String[queue.size()]);
        System.out.println(Arrays.toString(friends)); // Printing the array
        System.out.println();
        System.out.println("Queue : " + queue); // Printing the queue
    }
}
```

NPTEL Online Certification Courses
IIT Kharyapur

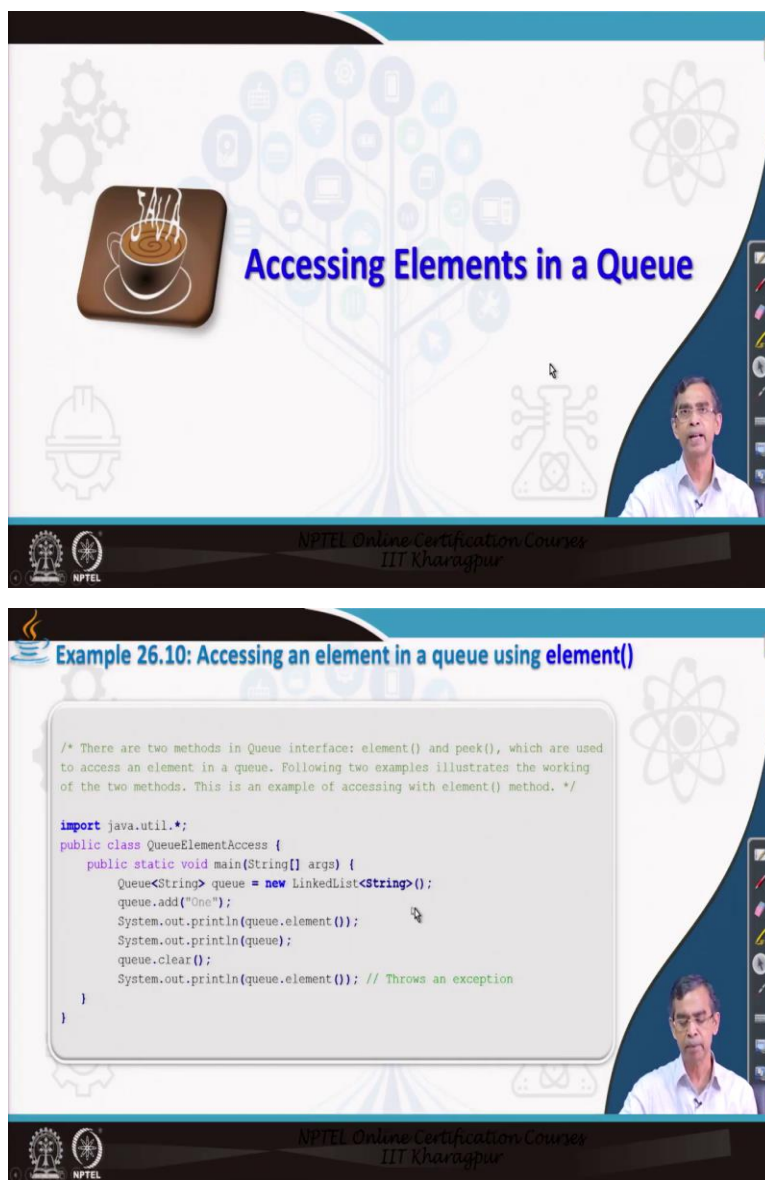
Now, so, this is one example now let us consider the opposite example given a Queue, I want to initialize the elements of the Queue and store them into an array or into any other collection. So, for this purpose first of all we have to have a Queue and let us this is the Queue that we have created with this declaration. Then, if you want to create a collection of type arrays, then toArray a method is there we already know about this method.

Now, so toArray and then total size, that size these queue dot size and then it basically copy all the elements which are there in the Queues will be copied into this array. So, this array is basically obviously match should be there otherwise it will throw an exception that in means this

Queue and the array should same structure. So, here you see we create a Queue of string and then array also to be string if you use integer then it should be integer like, ok.

So, this is an array and we can see that from the Queue toArray how all elements can be copied. And finally, once it is copied, we can print both arrays as well as the Queue. You can see and it is just make a duplicate so that Queue will return its element, but it will make a copy into other collection. So, this is the example that we have considered. Two example from array to Queue and Queue to array these are the very good examples to understand about how it works.

(Refer Slide Time: 38:37)



The image shows two slides from a video lecture. The top slide is titled "Accessing Elements in a Queue" and features a coffee cup icon. The bottom slide is titled "Example 26.10: Accessing an element in a queue using element()" and displays a Java code snippet. Both slides include the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

Accessing Elements in a Queue

NPTEL Online Certification Courses
IIT Kharagpur

Example 26.10: Accessing an element in a queue using element()

```
/* There are two methods in Queue interface: element() and peek(), which are used
to access an element in a queue. Following two examples illustrates the working
of the two methods. This is an example of accessing with element() method. */


import java.util.*;
public class QueueElementAccess {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<String>();
        queue.add("One");
        System.out.println(queue.element());
        System.out.println(queue);
        queue.clear();
        System.out.println(queue.element()); // Throws an exception
    }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Example 26.11: Accessing an element in a queue using peek()

/* Queue.peek(): The peek() operation is used to retrieve an element from the head of the queue, without removing it. If it performs the operation successfully, it returns the head element of the queue. Otherwise, it returns null value. */

```
import java.util.*;
public class QueuePeekOperation {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<String>();
        queue.add("One");
        System.out.println(queue.peek());
        System.out.println(queue);
        queue.clear();
        System.out.println(queue.peek()); // Returns null value
    }
}
```



NPTEL Online Certification Courses
IIT Kharagpur


Example 26.11: Accessing an element in a queue using peek()

/* Queue.peek(): The peek() operation is used to retrieve an element from the head of the queue, without removing it. If it performs the operation successfully, it returns the head element of the queue. Otherwise, it returns null value. */

Note:

- If we try to call **peek()** method on empty queue, it returns null value, but does NOT throw an exception as shown above.

```
System.out.println(queue);
queue.clear();
System.out.println(queue.peek()); // Returns null value
}
```



NPTEL Online Certification Courses
IIT Kharagpur

Now, accessing different elements and Queue that we have discussed about the method that is the element and peek. I just want program is ready for you, you can practice it, it is as obvious it is earlier, I do not want to repeat it again. And this is a peek and then earlier 1 is basically element is there.

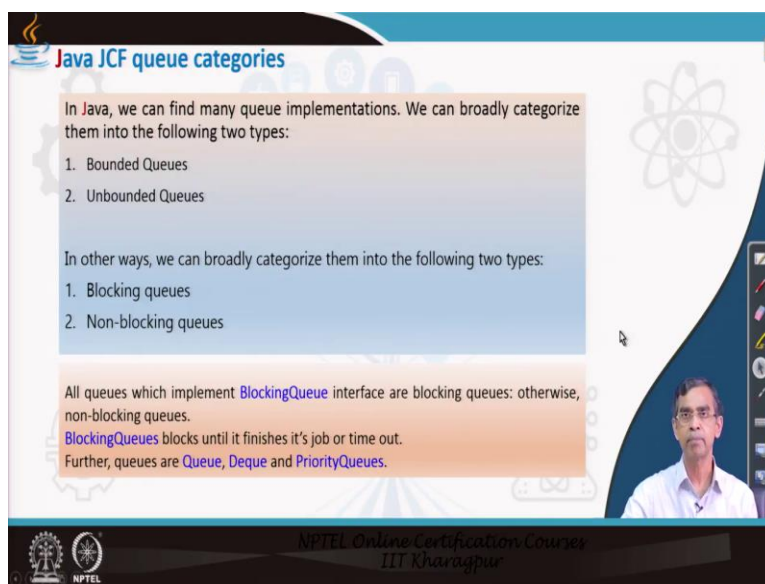
(Refer Slide Time: 38:58)



The slide features a central graphic of a tree with various icons as leaves, including a gear, a lightbulb, a document, and a person. To the left of the tree is a brown coffee cup with steam rising from it. The title 'Categories of Queue' is written in blue text to the right of the coffee cup. The background is white with faint icons of a gear, a hard hat, and a chemical flask. A small video inset of a man is in the bottom right corner.

Categories of Queue

NPTEL Online Certification Courses
IIT Kharagpur



The slide has a white background with a faint tree graphic. The title 'Java JCF queue categories' is in blue. The text is organized into three sections: an introductory paragraph, a numbered list of two types, another introductory paragraph, a second numbered list of two types, and a concluding paragraph. A small video inset of a man is in the bottom right corner.

Java JCF queue categories

In Java, we can find many queue implementations. We can broadly categorize them into the following two types:

1. Bounded Queues
2. Unbounded Queues

In other ways, we can broadly categorize them into the following two types:

1. Blocking queues
2. Non-blocking queues

All queues which implement `BlockingQueue` interface are blocking queues: otherwise, non-blocking queues.
`BlockingQueues` blocks until it finishes its job or time out.
Further, queues are `Queue`, `Deque` and `PriorityQueues`.

NPTEL Online Certification Courses
IIT Kharagpur

Blocking queue: operations

In addition to Queue's two forms of operations, BlockingQueue's supports two more forms as shown below.

Operation	Throws exception	Special value	Blocks	Times out
Insert	add(e)	offer(e)	put(e)	offer(e, time, unit)
Remove	remove()	poll()	take()	poll(time, unit)
Examine	element()	peek()	N/A	N/A

NPTEL Online Certification Courses
IIT Kharagpur

And then categories of Queue as I told you Blocking Queue, Queue, Dequeue, Priority Queue have exercised in many ways of creating Queue of course. Blocking Queue some sense that there are Array Dequeue is also we can do it and this is basically blocking Queue. If you want to have the multi threading programming then you should consider and in case of Blocking Queue 2 additional methods are defined there is called the put and take is basically in for the mutual exclusion because these 2 methods are called synchronized method in this 1. Anyway so, this is for the multi threading program. If you are interested for multi threading program then you can think about creating a Queue viable Queue.

(Refer Slide Time: 39:37)

Array DEQueue with JCF

NPTEL Online Certification Courses
IIT Kharagpur

The ArrayDeque class

The `ArrayDeque` class extends `AbstractCollection` and implements the `Deque` interface. It has no method of its own. `ArrayDeque` creates a dynamic array and has no capacity restrictions.

`ArrayDeque` is a generic class that has this declaration:

```
class ArrayDeque<E>
```

Here, E specifies the type of objects stored in the collection.

NPTEL Online Certification Courses
IIT Kharagpur

Now, Array Deque is another collection that is a Queue collection is a double ended Queue you can add or remove from any point as it is a little bit confusing so many, many experts recommend that it should not be used and this is also a Java collection framework for (39:59) in importance it has added it but really practically it has very less utilization.

(Refer Slide Time: 40:02)

The ArrayDeque class constructors

Constructors	Description
<code>ArrayDeque()</code>	Builds an empty deque. Its starting capacity is 16.
<code>ArrayDeque(int size)</code>	Builds a deque that has the specified initial capacity.
<code>ArrayDeque(Collection<? extends E> c)</code>	Creates a deque that is initialized with the elements of the collection passed in c.

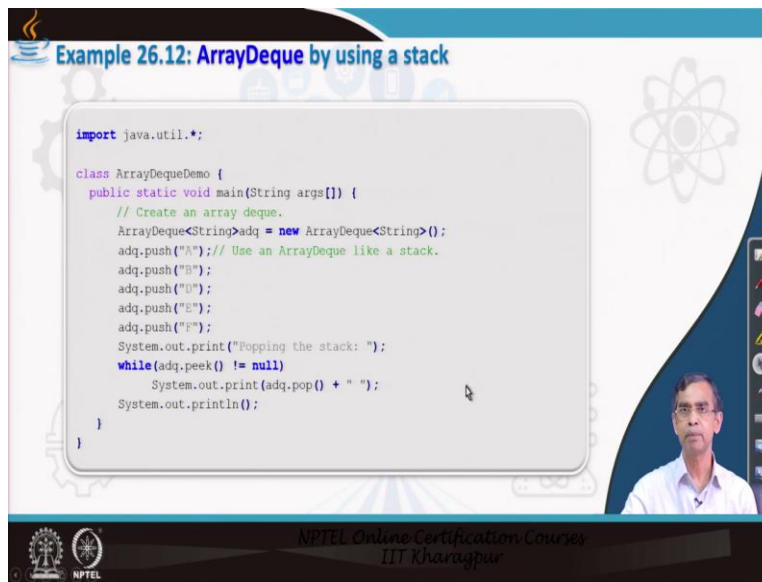
Note:

- In all cases, the capacity grows as needed to handle the elements added to the deque.

NPTEL Online Certification Courses
IIT Kharagpur

So, I just mentioned for the sake of continuity that this is the one class and you can take the different methods all methods are basically same for the other structure also like say add, remove of offer, poll, peek like this 1 same thing is applicable to here, but only the different name and you can perform the different operation.

(Refer Slide Time: 40:27)



Example 26.12: ArrayDeque by using a stack

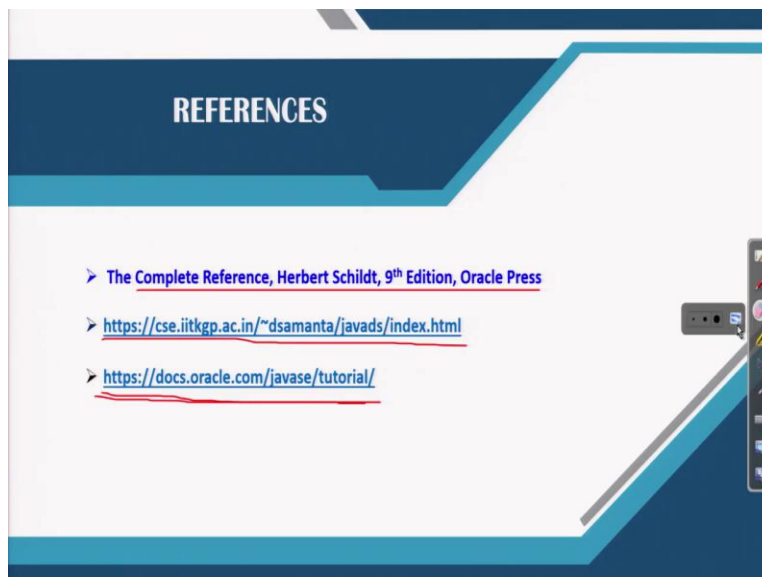
```
import java.util.*;

class ArrayDequeDemo {
    public static void main(String args[]) {
        // Create an array deque.
        ArrayDeque<String> adq = new ArrayDeque<String>();
        adq.push("A"); // Use an ArrayDeque like a stack.
        adq.push("B");
        adq.push("D");
        adq.push("E");
        adq.push("F");
        System.out.print("Popping the stack: ");
        while (adq.peek() != null)
            System.out.print(adq.pop() + " ");
        System.out.println();
    }
}
```

NPTEL Online Certification Course
IIT Kharagpur

Here is an example you can think about it.

(Refer Slide Time: 40:29)



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/>

Anyway so, we have given a lot of programs and we have demonstrated how the different way Queue can be created using the Java collection framework. All Programs you can load from the link that is given here. In this link all programs you can find and see the discussion has no limit actually to know more. So, this is the tutorial, the documentation, the official documentation from the Oracle site. If you want to know more many things particularly the Blocking Queues,

Array Dequeues and others, which have not been covering so much in details in this video, you can go through and you can learn many things from here.

And this book also has a good deal of discussion about different classes, particularly Queue, LinkedList class, you can have. You can go through the book and then plan many more any topics there. And this link also have a good discussion about Queue and defined algorithms, that is fine. I just advise you to go through the different programs, practice is vigorously, so, that your learning can be considered a little bit complete like. Thank you very much.