

Statement Purpose:

Purpose of this Lab is to familiarize the students with the use of queue data structure in writing simple Java programs. Another aim is to teach the students how to implement queue data structure using linked list. The students are given small tasks related to queue which they complete during the lab session under the supervision of the lab instructor. This helps them understand the concepts well which they learn in their lectures.

Activity Outcomes:

The students will learn how to write methods related to some actions performed on queues. They will understand how to write simple methods to

- delete a specific node from a queue using only enqueue() and dequeue() methods
- copy even nodes of a queue in a second queue and odd nodes in a third queue.
- reverse the elements of a queue by using a stack for this process

Theory Review (10 Minutes):

Queue

In computer science, a **queue** or **FIFO (first in, first out)** is an abstract data type that serves as a collection of elements, with two principal operations: **enqueue**, which adds an element to the end of collection, and **dequeue**, which removes the first element that was added.

In most high level languages, a queue can be easily implemented either through an array or a linked list.

Applications of queue

In computer science, queues are most commonly used for

- Accessing shared resources (e.g. printer)
- Heaps (min-priority queue or max-priority queue)
- Storing packets on network routers
- Breadth-first search in graphs



Commonly used operations on queues

Commonly used operations on queues are as follows:

1. isEmpty()

This method returns true if queue is empty and false otherwise.

2. isFull()

This method returns true if queue is full and false otherwise. If queue is implemented using linked list, this method is not used.

3. enqueue()

This method is used to insert any element at the end of the queue. It takes value to be inserted as a parameter.

4. dequeue()

This method removes one element from the front of the queue and returns it to the calling method.

5. front() or peek()

This method returns the element at the front of the queue without removing it from the queue.

6. Size()

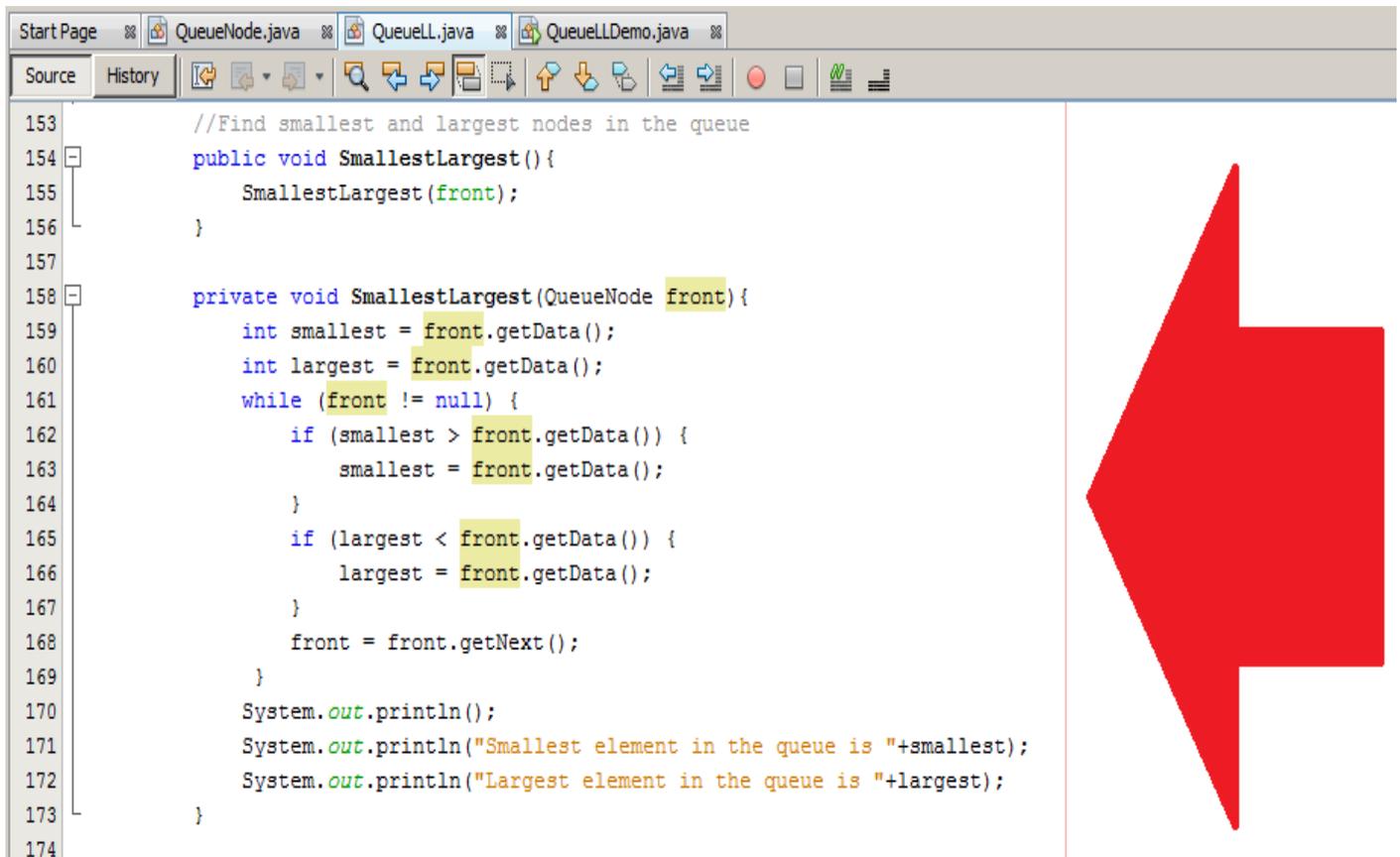
This method returns the number of elements present in the queue.



Practice Activity with Lab Instructor: (10 Minutes)

Queue Implementation using linked list:

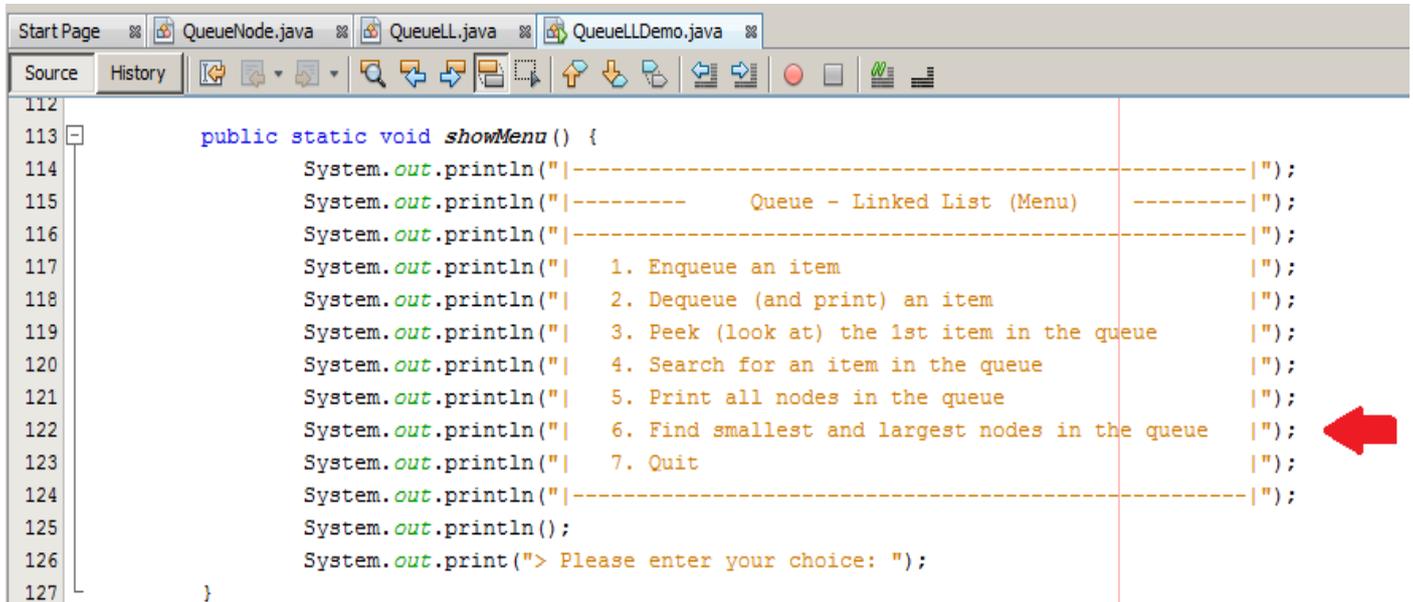
1. Create a project with the name QueueLinkedListProject
2. Create a package with the name queuelldemo within this project
3. Copy QueueNode class in this package
4. Copy QueueLL class in this package and add the following method in this class



```
153 //Find smallest and largest nodes in the queue
154 public void SmallestLargest(){
155     SmallestLargest(front);
156 }
157
158 private void SmallestLargest(QueueNode front){
159     int smallest = front.getData();
160     int largest = front.getData();
161     while (front != null) {
162         if (smallest > front.getData()) {
163             smallest = front.getData();
164         }
165         if (largest < front.getData()) {
166             largest = front.getData();
167         }
168         front = front.getNext();
169     }
170     System.out.println();
171     System.out.println("Smallest element in the queue is "+smallest);
172     System.out.println("Largest element in the queue is "+largest);
173 }
174
```



5. Copy QueueLLDemo class in this package and add the following lines of code in showMenu() method of this main class.

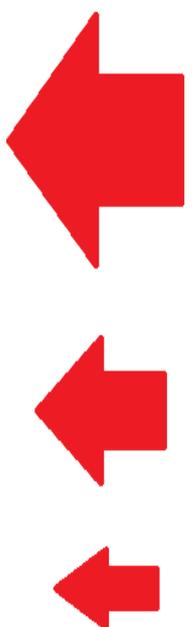


```
112
113 public static void showMenu() {
114     System.out.println("|-----|");
115     System.out.println("|----- Queue - Linked List (Menu) -----|");
116     System.out.println("|-----|");
117     System.out.println("| 1. Enqueue an item |");
118     System.out.println("| 2. Dequeue (and print) an item |");
119     System.out.println("| 3. Peek (look at) the 1st item in the queue |");
120     System.out.println("| 4. Search for an item in the queue |");
121     System.out.println("| 5. Print all nodes in the queue |");
122     System.out.println("| 6. Find smallest and largest nodes in the queue |");
123     System.out.println("| 7. Quit |");
124     System.out.println("|-----|");
125     System.out.println();
126     System.out.print("> Please enter your choice: ");
127 }
```

6. Add the following lines of code in the main() method of this main class



```
Start Page QueueNode.java QueueLL.java QueueLLDemo.java
Source History
99 // Find smallest and largest nodes in queue
100 else if (choice == 6) {
101     if (myQueue.isEmpty()) {
102         System.out.println("> Error: cannot find smallest and largest nodes (the queue is empty)");
103         System.out.println();
104     }
105     else {
106         System.out.println("> Displaying smallest and largest Nodes:");
107         System.out.print("> ");
108         myQueue.SmallestLargest();
109         System.out.println();
110         System.out.println();
111     }
112 }
113 // Quit
114 else if (choice == 7) {
115     System.out.println("> Goodbye!");
116     System.out.println();
117 }
118 else {
119     System.out.println("> Wrong selection. Try again.");
120     System.out.println();
121 }
122 } while (choice != 7);
123
```



7. When you run this program, it will display the following menu on the screen:



```

Output - QueueLinkedListProject (run) %
run:
-----
-----      Queue - Linked List (Menu)      -----
-----
|  1. Enqueue an item                          |
|  2. Dequeue (and print) an item              |
|  3. Peek (look at) the 1st item in the queue |
|  4. Search for an item in the queue          |
|  5. Print all nodes in the queue            |
|  6. Find smallest and largest nodes in the  |
|  7. Quit                                     |
-----
> Please enter your choice:

```

8. After enqueueing some elements e.g. 10, 20, 35, 26, 53, 90 in the queue, if you select option 5, it will display the following output:

```

Output - QueueLinkedListProject (run) %
-----
-----      Queue - Linked List (Menu)      -----
-----
|  1. Enqueue an item                          |
|  2. Dequeue (and print) an item              |
|  3. Peek (look at) the 1st item in the queue |
|  4. Search for an item in the queue          |
|  5. Print all nodes in the queue            |
|  6. Find smallest and largest nodes in the  |
|  7. Quit                                     |
-----
> Please enter your choice: 5
>   Printing All Nodes:
>   10, 20, 35, 26, 53, 90,

```

9. Now if you select option 6, it will display the following output:



```
Output - QueueLinkedListProject (run) ☒
|-----|
|----- Queue - Linked List (Menu) -----|
|-----|
| 1. Enqueue an item |
| 2. Dequeue (and print) an item |
| 3. Peek (look at) the 1st item in the queue |
| 4. Search for an item in the queue |
| 5. Print all nodes in the queue |
| 6. Find smallest and largest nodes in the queue |
| 7. Quit |
|-----|
|
| > Please enter your choice: 6
| >   Displaying smallest and largest Nodes:
| >
| Smallest element in the queue is 10
| Largest element in the queue is 90
```

LAB EXERCISES: (60 Minutes)

1. Delete a node from a queue using only enqueue() and dequeue() methods. You can use another queue for this purpose.

Hint:

You have a queue which contains 100 random numbers between 1 and 1000. Your task is to write a code in main class which deletes a particular node from the queue. However, in this process, you can use only enqueue() and dequeue() methods. In this process, you will dequeue() all elements from first queue, **except the one which you want to delete**, and enqueue() these elements in the second queue. Now all elements of first queue, except the one which you have deleted, are in the second queue. Finally, dequeue all elements of the second queue and enqueue them in the original queue.



Sample Run (for 10 elements):

```
Output - LabActivitiesSolution (run) #2
run:
-----
----- Queue - Linked List (Menu) -----
-----
1. Enqueue an item
2. Dequeue (and print) an item
3. Peek (look at) the 1st item in the queue
4. Search for an item in the queue
5. Print all nodes in the queue
6. Delete node from queue (using 2nd queue)
7. Even and Odd value queues
8. Reverse all elements in a queue
9. Quit
-----

> Please enter your choice: 6
> Before deleting value, elements of the queue are as follows:
751, 481, 438, 520, 263, 92, 139, 211, 83, 854,

> Please enter an INT value (between 1 and 1000) to delete from the queue: 520
> After deleting value 520, elements of the queue 1 are as follows:
751, 481, 438, 263, 92, 139, 211, 83, 854,
```

2. Copy even nodes of a queue in a second queue and odd nodes in a third queue. After the process is over, the first queue should be empty.

Hint:

You have a queue which contains 100 random numbers between 1 and 1000. Your task is to write a code in main class which enqueues all **even elements** of the first queue in the second queue and all **odd elements** in the third queue. During this process, you should use only enqueue() and dequeue() methods. At the end of the process, the first queue should be empty.



Sample Run (for 10 elements):

```
Output - LabActivitiesSolution (run) #2 ✖
run:
-----|
----- Queue - Linked List (Menu) -----|
-----|
| 1. Enqueue an item |
| 2. Dequeue (and print) an item |
| 3. Peek (look at) the 1st item in the queue |
| 4. Search for an item in the queue |
| 5. Print all nodes in the queue |
| 6. Delete node from queue (using 2nd queue) |
| 7. Even and Odd value queues |
| 8. Reverse all elements in a queue |
| 9. Quit |
-----|

> Please enter your choice: 7
Elements of queue 1 are as follows:
348, 168, 744, 123, 120, 510, 605, 799, 91, 798,

Elements of queue 2 are as follows:
348, 168, 744, 120, 510, 798,

Elements of queue 3 are as follows:
123, 605, 799, 91,
```

3. Reverse the elements of a queue. You can use stack for this purpose.

Hint:

You have a queue which contains 100 random numbers between 1 and 1000. Your task is to write a code in main class which reverses all elements of the queue. In this process, you will first **dequeue** all elements of the queue and **push** them onto the stack. Then you will **pop** all elements of the stack and **enqueue** them into the queue.



Sample Run (for 10 elements):

```
Output - LabActivitiesSolution (run) #2 ☒
run:
-----|
----- Queue - Linked List (Menu) -----|
-----|
1. Enqueue an item |
2. Dequeue (and print) an item |
3. Peek (look at) the 1st item in the queue |
4. Search for an item in the queue |
5. Print all nodes in the queue |
6. Delete node from queue (using 2nd queue) |
7. Even and Odd value queues |
8. Reverse all elements in a queue |
9. Quit |
-----|

> Please enter your choice: 8
Elements of queue are as follows:
734, 351, 763, 640, 447, 576, 854, 758, 14, 319,

Elements of queue after reversing are as follows:
319, 14, 758, 854, 576, 447, 640, 763, 351, 734,
```

