Queue

DR. SANGA CHAKI

Contents

1. Queues
2. Queue Operations:
   a) Insert,
   b) Delete,
   c) isFull,
   d) isEmpty,
   e) Display
3. Implementation
   a) Array
   b) Linked List

What is a Queue?

1. A queue is an ordered list in which insertions are done at one end (rear) and deletions are done at the other end (front)
2. First element to be inserted is the first one to be deleted.
3. So it’s a FIFO list
4. Enqueue = when element is inserted in queue
5. Dequeue = when element is deleted from queue
6. Underflow = trying to dequeue an empty queue
7. Overflow = trying to enqueue an element into a full queue

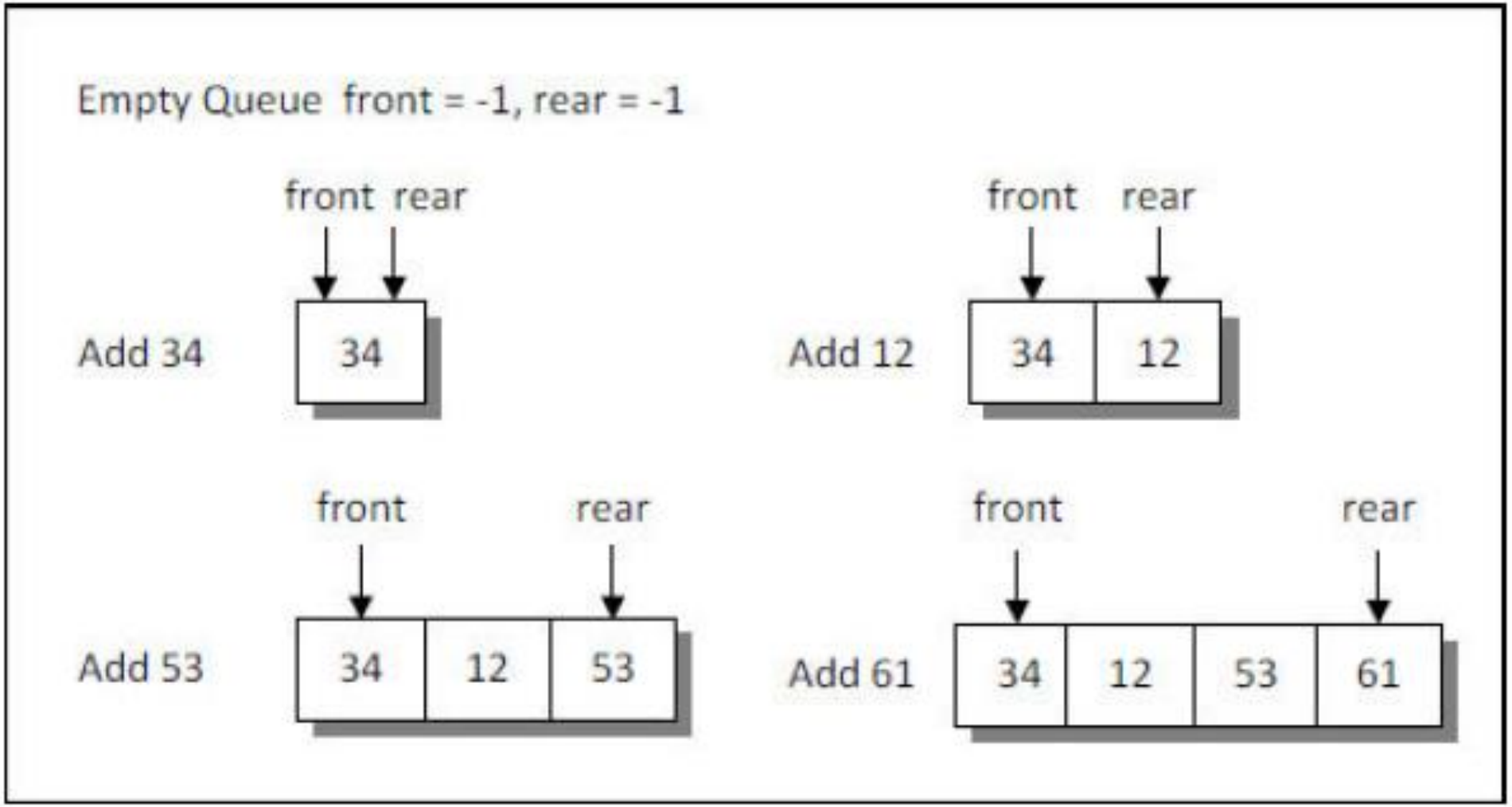
What is a Queue?

Implementations

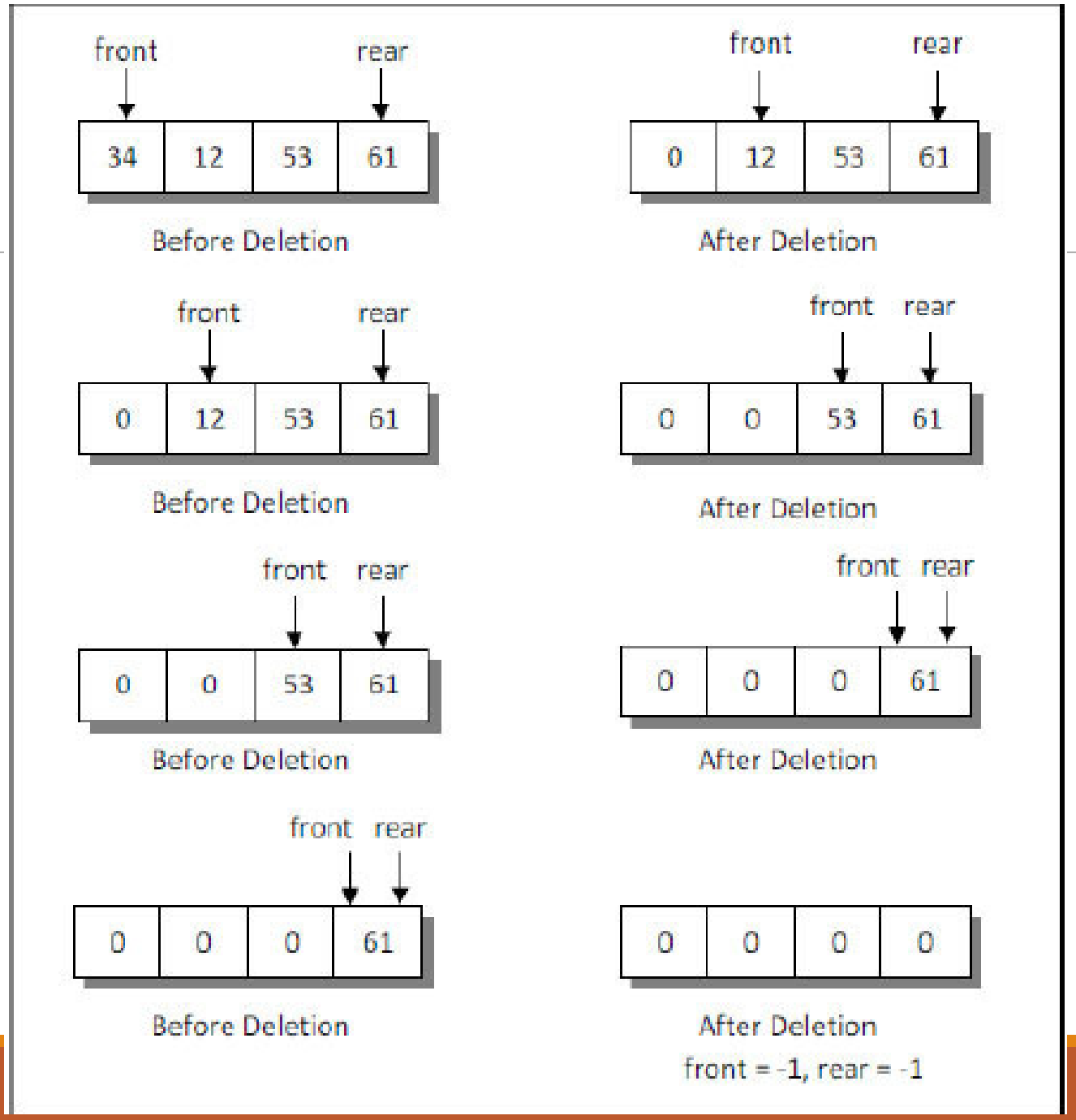
1. Implementations:
   - Array based
   - Linked list based

Array Implementation

Enqueue Procedure



Dequeue Procedure



Enqueue: Array Based Implementation

```
public class AllQueueMethods {
    public static final int MAX=5;
    public static int front=-1;
    public static int rear=-1;

    public static void insert(int Q[])  {
        if(rear==MAX-1) {
            System.out.println("Q is full");
        }
        else {
        if(front == -1)
            front = front+1; //start Q
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the queue element: ");
        int n = sc.nextInt();
        rear = rear+1;
        Q[rear] = n;
        }
    }
```

Dequeue: Array Based Implementation

```
public static void delete(int Q[]) {
    if(front == -1)
        System.out.println("Empty Q");
    else {
        front= front+1;
    }
}
```

Display: Array Based Implementation

```
public static void display(int Q[]) {
    System.out.print("front--->     ");
    for(int i = front; i<=rear;i++) {
        System.out.print(Q[i]+ "     ");
    }
    System.out.print("<--- rear\n");
}
```

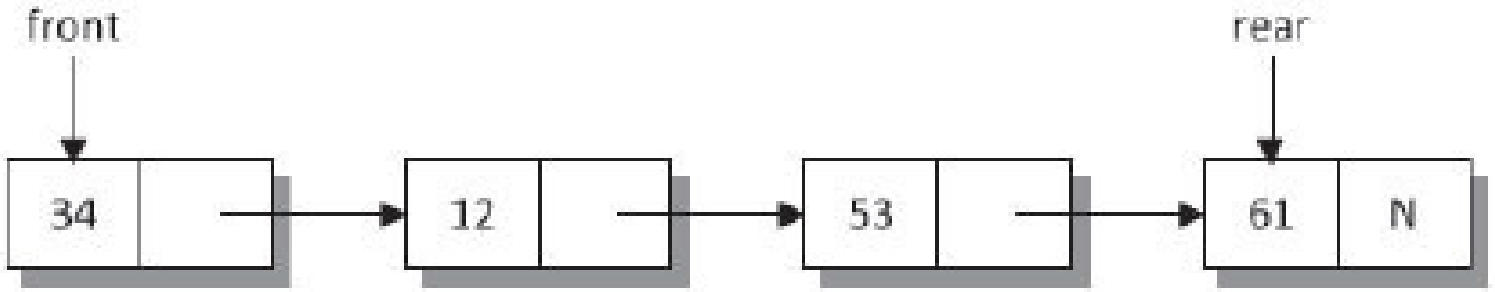
isEmpty and isFull: Array Based Implementation

```
public static void isFull() {
    if(rear==MAX-1)
        System.out.println("Q is full");
}
public static void isEmpty()    {
    if(front==rear && front==-1)
        System.out.println("Q is empty");

}
```

Queue: Linked List Implementation

1. Space for the elements in a linked list is allocated dynamically, hence it can grow as long as there is enough memory available for dynamic allocation.



2. Enqueue operation implemented by inserting element at the end of LL
3. Dequeue operation implemented by deleting an element at the beginning of LL

Enqueue: Linked List Implementation

```
public static void enqueue() {
    Node new_node = new Node();
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter info: ");
    int n = sc.nextInt();
    new_node.info = n;
    if(rear!=null) { //Q already contains data
        rear.next = new_node;
        new_node.next = null;
        rear = new_node;
    }
    else {//Starting with empty Q
        front = rear = new_node;
        new_node.next = null;
    }
}
```

Dequeue: Linked List Implementation

```
public static void dequeue() {
    if(front!=null) {
        front = front.next;
    }
}
```

Display: Linked List Implementation

```
public static void display() {
    Node p = front;
    System.out.print("front --->");
    while(p!=null) {
        System.out.print(p.info+" || ");
        p=p.next;
    }
    System.out.print("<--- rear\n");
}
```