Implementation of Queue using Arrays

```
package queue;

public class Uqueue {
        private int maxSize;
    private int front;
    private int rear;
    private int[] queueArray;

    public Uqueue() {
        maxSize = 5; // One extra space to differentiate between front and rear
positions
        queueArray = new int[maxSize];
        front = 0;
        rear = -1;
    }

    public void enqueue(int value) {
        if (rear==maxSize-1) {
          System.out.println("Queue is full. Cannot enqueue " + value);
          return;
          }
        else {
            rear = (rear + 1);
            queueArray[rear] = value;
            System.out.println(value + " enqueued.");
        }

        }

    public int dequeue() {
        if (front>rear) {
          System.out.println("Queue is empty. Cannot dequeue.");
            return -1;
        }
    else {
        int dequeuedValue = queueArray[front];
        front = (front + 1);
        System.out.println(dequeuedValue + " dequeued.");
        return dequeuedValue;
    }
         // Or throw an exception
    }


public int peek() {
    if (front<=rear) {
        return queueArray[front];
    } else {
        System.out.println("Queue is empty. Nothing to peek.");
        return -1; // Or throw an exception
    }
}

public void display1() {
      int i;
      if(isEmpty()) {
                System.out.println("Empty Queue");
      }
      else {
                System.out.println("Items in queue");
                for(i=front;i<=rear;i++) {
                          System.out.println(queueArray[i]);
                }
      }
}

public boolean isEmpty() {
    return (rear + 1)== front;
}

public boolean isFull() {
    return rear== maxSize-1;
    }

    public static void main(String[] args) {
        Uqueue queue = new Uqueue();

        queue.enqueue(10);
        queue.enqueue(20);
        queue.enqueue(30);
        queue.display1();

        System.out.println("Peek: " + queue.peek());

        queue.dequeue();
        queue.dequeue();
        queue.dequeue();
        queue.enqueue(40);
        queue.display1();

        System.out.println("Is empty? " + queue.isEmpty());
        System.out.println("Is full? " + queue.isFull());
    }


}
```

[OR]

```
public class Queue {
  private int maxSize;
  private int front;
  private int rear;
  private int[] queueArray;

  public Queue(int size) {
    maxSize = size; // One extra space to differentiate between front and rear
positions
    queueArray = new int[maxSize];
    front = 0;
    rear = -1;
  }

  public void enqueue(int value) {
    if (rear == maxSize - 1) {
      System.out.println("Queue is full. Cannot enqueue " + value);
      return;
    } else {
      rear = (rear + 1);
      queueArray[rear] = value;
      System.out.println(value + " enqueued.");
    }

  }

  public int dequeue() {
    if (rear == front-1) {
      System.out.println("Queue is empty. Cannot dequeue.");
      return -1;
    } else {
      int i = front;
      int t=rear;
      int dequeuedValue = queueArray[i];
      int d = dequeuedValue;
      for (i = front; i <= t; i++) {
        System.out.println("Front is .: "+front+"  Rear is : "+rear);
        System.out.println("ith pos : "+queueArray[i]+"i+1th pos :
"+queueArray[i+1]);
        queueArray[i] = queueArray[i + 1];
        t--;
            }
      rear=rear-1;
      System.out.println("Rear after loop is"+rear);
      System.out.println(d + " dequeued.");
      return d;
    }
  }

  public int peek() {
```

```
  if (front <= rear) {
    System.out.println("Peek:" + queueArray[front]);
    return queueArray[front];
  } else {
    System.out.println("Queue is empty. Nothing to peek.");
    return -1; // Or throw an exception
  }
}

public void display() {
  int i;
  if (isEmpty()) {
    System.out.println("Empty Queue");
  } else {
    System.out.println("Items in Queue");
    for (i = front; i <= rear; i++) {
      System.out.println("Front is"+front+"Rear is"+rear+"i value is"+i);
      System.out.println(queueArray[i]);
    }
  }
}

public boolean isEmpty() {
  return rear == front - 1;
}

public boolean isFull() {
  return rear == maxSize - 1;
}

public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
  System.out.print("Enter the size of the queue: ");
  int size = scanner.nextInt();
  Queue queue = new Queue(size);
  while (true) {
    System.out.println("\nQueue Operations:");
    System.out.println("1. Enqueue");
```

```
      System.out.println("2. Dequeue");
      System.out.println("3. Peek");
      System.out.println("4. Display");
      System.out.println("5. Is Empty");
      System.out.println("6. Is Full");
      System.out.println("7. Exit");
      System.out.print("Enter your choice: ");
      int choice = scanner.nextInt();
      switch (choice) {
        case 1:
          System.out.print("Enter element to insert: ");
          int insertItem = scanner.nextInt();
          queue.enqueue(insertItem);
          break;
        case 2:
          queue.dequeue();
          break;
        case 3:
          queue.peek();
          break;
        case 4:
          queue.display();
          break;
        case 5:
          System.out.println("IS EMPTY:" + queue.isEmpty());
          break;
        case 6:
          System.out.println("IS FULL:" + queue.isFull());
          break;
        case 7:
          scanner.close();
          System.exit(0);
        default:
          System.out.println("Invalid choice. Please try again.");
      }
    }
  }
}
```