```
package stacksqueues;

// Java Program to find the maximum for
// each and every contiguous subarray of size k.
import java.util.Deque;
import java.util.LinkedList;

 class SlidingWindow
{

    // A Dequeue (Double ended queue)
    // based method for printing
    // maximum element of
    // all subarrays of size k
    static void printMax(int arr[], int n, int k)
    {

        // Create a Double Ended Queue, Qi
        // that will store indexes of array elements
        // The queue will store indexes of
        // useful elements in every window and it will
        // maintain decreasing order of values
        // from front to rear in Qi, i.e.,
        // arr[Qi.front[]] to arr[Qi.rear()]
        // are sorted in decreasing order
        Deque<Integer> Qi = new LinkedList<Integer>();

        /* Process first k (or first window)
        elements of array */
        int i;
        for (i = 0; i < k; ++i)
        {

            // For every element, the previous
            // smaller elements are useless so
            // remove them from Qi
            while (!Qi.isEmpty() && arr[i] >=
                        arr[Qi.peekLast()])

                // Remove from rear
                Qi.removeLast();

            // Add new element at rear of queue
            Qi.addLast(i);
        }

        // Process rest of the elements,
        // i.e., from arr[k] to arr[n-1]
        for (; i < n; ++i)
        {

            // The element at the front of the
            // queue is the largest element of
            // previous window, so print it
            System.out.print(arr[Qi.peek()] + " ");

            // Remove the elements which
            // are out of this window
            while ((!Qi.isEmpty()) && Qi.peek() <=
                        i - k)
                Qi.removeFirst();

            // Remove all elements smaller
            // than the currently
            // being added element (remove
            // useless elements)
```

```java
            while ((!Qi.isEmpty()) && arr[i] >=
                        arr[Qi.peekLast()])
                Qi.removeLast();

            // Add current element at the rear of Qi
            Qi.addLast(i);
        }

        // Print the maximum element of last window
        System.out.print(arr[Qi.peek()]);
    }

    // Driver code
    public static void main(String[] args)
    {
        int arr[] = { 12, 1, 78, 90, 57, 89, 56 };
        int k = 3;
        printMax(arr, arr.length, k);
    }
}
// This code is contributed by Sumit Ghosh
```