```
import java.util.Vector;
import static java.lang.Math.sqrt;
import static java.lang.Math.floor;

class EthCode
{
      // This methid finds all primes smaller than 'limit'
      // using simple sieve of eratosthenes. It also stores
      // found primes in vector prime[]
      static void simpleSieve(int limit, Vector<Integer> prime)
      {
            // Create a boolean array "mark[0..n-1]" and initialize
            // all entries of it as true. A value in mark[p] will
            // finally be false if 'p' is Not a prime, else true.
            boolean mark[] = new boolean[limit+1];

            for (int i = 0; i < mark.length; i++)
                  mark[i] = true;

            for (int p=2; p*p<limit; p++)
            {
                  // If p is not changed, then it is a prime
                  if (mark[p] == true)
                  {
                        // Update all multiples of p
                        for (int i=p*p; i<limit; i+=p)
                              mark[i] = false;
                  }
            }

            // Print all prime numbers and store them in prime
            for (int p=2; p<limit; p++)
            {
                  if (mark[p] == true)
                  {
                        prime.add(p);
                        System.out.print(p + " ");
                  }
            }
      }

      // Prints all prime numbers smaller than 'n'
      static void segmentedSieve(int n)
      {
            // Compute all primes smaller than or equal
            // to square root of n using simple sieve
            int limit = (int) (floor(sqrt(n))+1);
            Vector<Integer> prime = new Vector<>();
            simpleSieve(limit, prime);

            // Divide the range [0..n-1] in different segments
            // We have chosen segment size as sqrt(n).
            int low = limit;
            int high = 2*limit;

            // While all segments of range [0..n-1] are not processed,
            // process one segment at a time
            while (low < n)
            {
                  if (high >= n)
                        high = n;

                  // To mark primes in current range. A value in mark[i]
                  // will finally be false if 'i-low' is Not a prime,
                  // else true.
                  boolean mark[] = new boolean[limit+1];

                  for (int i = 0; i < mark.length; i++)
                        mark[i] = true;

                  // Use the found primes by simpleSieve() to find
                  // primes in current range
                  for (int i = 0; i < prime.size(); i++)
                  {
                        // Find the minimum number in [low..high] that is
                        // a multiple of prime.get(i) (divisible by prime.get(i))
                        // For example, if low is 31 and prime.get(i) is 3,
                        // we start with 33.
                        int loLim = (int) (floor(low/prime.get(i)) * prime.get(i));
                        if (loLim < low)
                              loLim += prime.get(i);

                        /* Mark multiples of prime.get(i) in [low..high]:
                              We are marking j - low for j, i.e. each number
                              in range [low, high] is mapped to [0, high-low]
                              so if range is [50, 100] marking 50 corresponds
                              to marking 0, marking 51 corresponds to 1 and
                              so on. In this way we need to allocate space only
                              for range */
                        for (int j=loLim; j<high; j+=prime.get(i))
                              mark[j-low] = false;
                  }

                  // Numbers which are not marked as false are prime
                  for (int i = low; i<high; i++)
                        if (mark[i - low] == true)
                              System.out.print(i + " ");

                  // Update low and high for next segment
                  low = low + limit;
                  high = high + limit;
            }
      }


      public static void main(String args[])
      {
            int n = 100;
            System.out.println("Primes smaller than " + n + ":");
            segmentedSieve(n);
      }
}
```