```
1 import java.util.ArrayList;
 2
 3 public class Factor {
       public static int total = 600000;
 5
       public static int numChild = 2;
       public static int range = total / numChild;
 6
 7
       public static int begin = 0;
8
       public static void main(String args []) throws
   InterruptedException {
           // variable to keep track of thread count
10
11
           int i;
12
           long start_s = System.nanoTime();
13
           System.err.println("Run Factor " + total +
   ":" + numChild);
14
15
           //Array list to keep track of threads
16
           ArrayList<Thread> threads = new ArrayList
   <>();
17
           //loop to run thru threads
18
           for (i = 0; i < numChild; i++) {</pre>
19
               // Creates new thread
20
               Thread thread = new Thread(new
   childFactor(begin, begin + range));
               //adds new thread with factor to
21
   ArrayList
22
               threads.add(thread);
23
               //starts threads
24
               thread.start();
25
26
           for (Thread each :
27
                    threads) {
28
               each.join();
29
30
           long stop_s = System.nanoTime();
           System.err.println("time: " + (stop_s -
31
   start_s)/1000000000.0 + " seconds");
       }
32
33
34
       private static class childFactor implements
   Runnable{
           // starting value for factoring
35
36
           int begin;
37
           int end;
```

```
38
39
            // Constructor
           public childFactor(int start, int stop){
40
41
                this.begin = start;
42
                this.end = stop;
43
           }
44
45
           @Override
           public void run(){
46
                //long start_s = System.nanoTime();
47
                int val, i;
48
                for(val = begin; val<end; val++){</pre>
49
                    for(i=2; i<= val/2; i++){</pre>
50
                         if(val % i == 0){
51
52
                             break;
53
                         }
                         if(i== val / 2){
54
                             System.out.println("F:" +
55
   val);
56
                         }
                    }
57
58
59
                //long stop_s = System.nanoTime();
                //System.err.println("time: " + (stop_s
60
      start_s) + " seconds");
61
62
       }
63 }
64
```

```
1 import java.util.ArrayList;
 2
 3 public class SquareRoot {
       public static int total = 600000;
 5
       public static int numChild = 8;
       public static int range = total / numChild;
 6
 7
       public static int begin = 0;
       public static double qlobal = 0.0;
8
 9
       public static double memAttr;
10
11
12
       public static void main(String args []) throws
   InterruptedException {
13
           // variable to keep track of thread count
14
           int i;
15
           long start_s = System.nanoTime();
           System.err.println("Run Factor " + total +
16
   ":" + numChild);
17
18
           //Array list to keep track of threads
           ArrayList<Thread> threadList = new
19
   ArrayList<>();
20
           //loop to run thru threads
21
           for (i = 0; i < numChild; i++) {</pre>
22
               // Creates new thread
23
               Thread thread = new Thread(new
   childSquareRoot(begin, begin + range));
               //adds new thread with factor to
24
   ArrayList
25
               threadList.add(thread);
26
               //starts threads
27
               thread.start();
28
           }
29
           for (Thread each :
                   threadList) {
30
31
               each.join();
32
           }
           long stop_s = System.nanoTime();
33
           System.err.println("time: " + (stop_s -
34
   start_s)/1000000000.0 + " seconds");
35
36
       }
37
38
       private static class childSquareRoot implements
```

```
38
    Runnable{
39
           // starting value for factoring
40
           int begin;
           int end;
41
42
43
           // Constructor
44
           public childSquareRoot(int start, int stop
   ){
45
               this.begin = start;
46
               this.end = stop;
47
           }
           @Override
48
49
           public void run(){
50
               System.err.println("CPU: " + Runtime.
   getRuntime().availableProcessors());
51
               //long start_s = System.nanoTime();
               double totalSum = 0.0;
52
               for (int local = (int) begin; local <</pre>
53
   end; local++) {
54
                    double root = Math.sqrt(local);
55
                    //revise lines to do prints
                    System.out.println("Number: " +
56
   local + " : SquareRoot of Number: " + root + " ");
                    if (local % 5 == 0) {
57
58
                        totalSum += root;
59
                    }
60
               }
               System.err.println("
61
                                       totalSum = " +
                 global = " + ++qlobal + " memAttr = "
   totalSum + "
    + memAttr);
62
               //long stop_s = System.nanoTime();
               //System.err.println("time: " + (stop_s
63
    - start_s) + " seconds");
64
65
       }
66 }
```