```java
import java.util.ArrayList;

public class Factor {
    public static int total = 600000;
    public static int numChild = 2;
    public static int range = total / numChild;
    public static int begin = 0;

    public static void main(String args []) throws InterruptedException {
        // variable to keep track of thread count
        int i;
        long start_s = System.nanoTime();
        System.err.println("Run Factor " + total + ":" + numChild);

        //Array list to keep track of threads
        ArrayList<Thread> threads = new ArrayList<>();
        //loop to run thru threads
        for (i = 0; i < numChild; i++) {
            // Creates new thread
            Thread thread = new Thread(new childFactor(begin, begin + range));
            //adds new thread with factor to ArrayList
            threads.add(thread);
            //starts threads
            thread.start();
        }
        for (Thread each :
                threads) {
            each.join();
        }
        long stop_s = System.nanoTime();
        System.err.println("time: " + (stop_s - start_s)/1000000000.0 + " seconds");
    }

    private static class childFactor implements Runnable{
        // starting value for factoring
        int begin;
        int end;

        // Constructor
        public childFactor(int start, int stop){
            this.begin = start;
            this.end = stop;
        }

        @Override
        public void run(){
            //long start_s = System.nanoTime();
            int val, i;
            for(val = begin; val<end; val++){
                for(i=2; i<= val/2; i++){
                    if(val % i == 0){
                        break;
                    }
                    if(i== val / 2){
                        System.out.println("F:" + val);
                    }
                }
            }
            //long stop_s = System.nanoTime();
            //System.err.println("time: " + (stop_s - start_s) + " seconds");
        }
    }
}

```

```java
import java.util.ArrayList;

public class SquareRoot {
    public static int total = 600000;
    public static int numChild = 8;
    public static int range = total / numChild;
    public static int begin = 0;
    public static double global = 0.0;
    public static double memAttr;


    public static void main(String args []) throws InterruptedException {
        // variable to keep track of thread count
        int i;
        long start_s = System.nanoTime();
        System.err.println("Run Factor " + total + ":" + numChild);

        //Array list to keep track of threads
        ArrayList<Thread> threadList = new ArrayList<>();
        //loop to run thru threads
        for (i = 0; i < numChild; i++) {
            // Creates new thread
            Thread thread = new Thread(new childSquareRoot(begin, begin + range));
            //adds new thread with factor to ArrayList
            threadList.add(thread);
            //starts threads
            thread.start();
        }
        for (Thread each :
                threadList) {
            each.join();
        }
        long stop_s = System.nanoTime();
        System.err.println("time: " + (stop_s - start_s)/1000000000.0 + " seconds");

    }

    private static class childSquareRoot implements
```

```java
 Runnable{
        // starting value for factoring
        int begin;
        int end;

        // Constructor
        public childSquareRoot(int start, int stop){
            this.begin = start;
            this.end = stop;
        }
        @Override
        public void run(){
            System.err.println("CPU: " + Runtime.getRuntime().availableProcessors());
            //long start_s = System.nanoTime();
            double totalSum = 0.0;
            for (int local = (int) begin; local < end; local++) {
                double root = Math.sqrt(local);
                //revise lines to do prints
                System.out.println("Number: " + local + " : SquareRoot of Number: " + root + " ");
                if (local % 5 == 0) {
                    totalSum += root;
                }
            }
            System.err.println("   totalSum = " + totalSum + "  global = " + ++global + " memAttr = " + memAttr);
            //long stop_s = System.nanoTime();
            //System.err.println("time: " + (stop_s - start_s) + " seconds");
        }
    }
}
```