10 **Exercise 5: Integers, Part III**
Look at Exercise 2 on Exercise Sheet 1. This program actually performs two vastly different tasks. This is not good program design. Therefore:

1. Separate the two functionalities, i.e. write a first Java program that performs the prime numbers finding task and a second one that does the measuring of the counting time.
2. Change the second program such that it determines and prints the time needed to count from the smallest to the largest integer, when you use a double as counting variable.
   Note: If you have "solved" this task previously by just writing an empty loop, you have NOT solved the previous task. In this case, fix the previous task first. Test your program by letting it count to numbers $n, 2n, 10n$ for sufficiently large n and verifying that the obtained timings roughly show the respective factors.

10 **Exercise 6: Doubles, Part I**
Write a Java program which performs the following tasks:

1. It first inputs three integers n_1, n_2 and n_3 from the user: n_1 must be between 1 and 1 million, n_2 and n_3 may be arbitrary, but different and n_2 must be less than n_3.
2. The program then generates n_1 random floating point numbers and stores them.
3. It computes the sum, the product, the average, the variance, the smallest and the largest value of these numbers.
4. It outputs, in a nicely formatted way, all the numbers input and the statistics computed.

20 **Exercise 7: Doubles, Part II**
Write a Java program which performs the following tasks:

1. The class should have a method to compute the following function:

$$f(o_1, \ldots, o_n) = 1 - \left(1 + \prod_{i=1}^{n} \frac{o_i}{1 - o_i}\right)^{-1}$$

   The function value should a double-precision floating point number.
2. Let us assume $o_1 = 0.5$, $o_i = 0.9$ for all $1 < i < k$ and $o_i = 0.1$ for all $k \leq i$. Compute and print the series of values for f for the following value pairs (k, n): (10, 20), (100, 120), (1000, 1200), and (10000, 10200)
3. Look at the results and make observations. What would happen if we further increase k?
4. Now assume we have to compute all $f(o_1, \ldots, o_n)$ for increasing n. How efficient will your program allow this to be done?
5. After having probably done a computationally suboptimal solution for this task, think of a more efficient way to solve it.
   Hint: Think of an incremental way of computing $f(o_1, \ldots, o_{n+1})$ when you already have computed $f(o_1, \ldots, o_n)$.