APIT, Concurrency in Swing, Primes Lab

Dr. Simon Rogers

11/11/2014

Task

You will write a Java program that will compute the prime numbers (numbers only divisible by one and themselves). The program will start from 0, and check each integer for whether it is a prime or not. If it is a prime. It should maintain a list of primes that have been found, as well as text fields showing the number it is currently at, and how many primes it has found. There should be buttons for starting the search, stopping the search and resetting the search. If the search is stopped and then the user clicks the start button, the search should restart from where it was stopped. Reset should return to zero and blank out all of the output.

Update: 10th Feb 2015: To make this feasible in the 1hr session, I have created a skeleton `Primes.java` to which you just need to add the SwingWorker object and any other objects that are required for `publish` and `process`. You can find this on Moodle: `Primes.java`.

An example screenshot can be seen in the figure below.

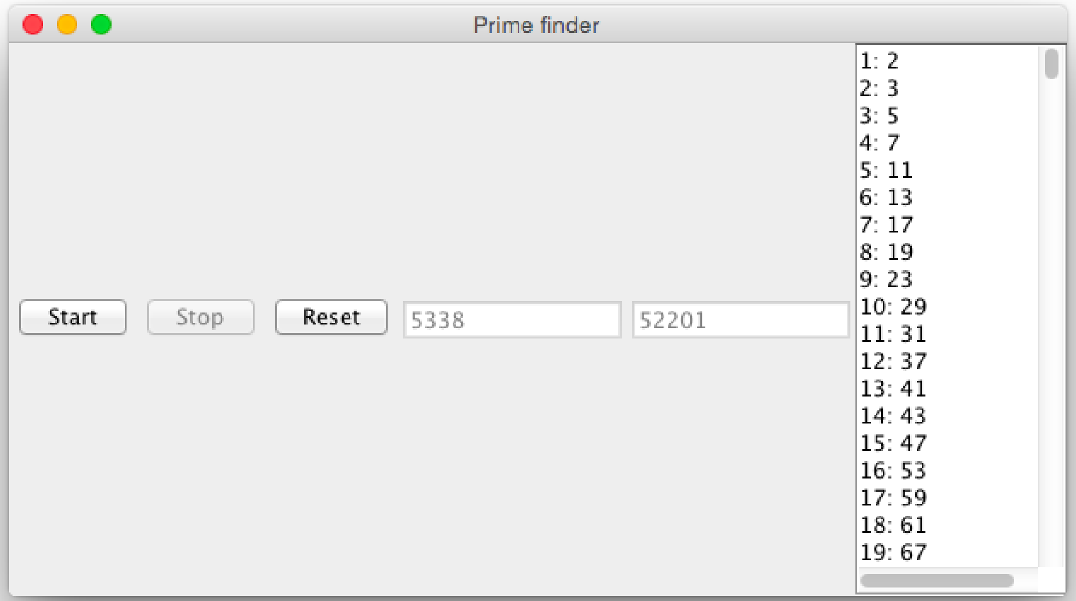
Some hints:

- You will need a `SwingWorker` to do the prime finding.
- To search if N is a prime, you should loop i from 2 to $N/2$ and check whether or not `N%i==0` (this asks: 'is the remainder after division equal to zero?'). If this is the case for any i, then N *is not* a prime.
- 1 and 2 *are* primes.
- If using `process` and `report` with `SwingWorker` and you wish to report more than one value, you'll need to create some kind of custom object.
- To make a `JTextArea` scrollable, put it inside a `JScrollPane` and add the scroll pane to your frame:

```
a = new JTextArea(20,10); % 20 rows, 10 columns
s = new JScrollPane(a);
getContentPane().add(s);
```

- You can add text to the `JTextArea` using the `append()` method. Remember to also append a newline character: `\n`.

To aid with your debugging, the 3907th prime is 36,847. There is a list of the first 10,000 primes here.



```
1: 2
2: 3
3: 5
4: 7
5: 11
6: 13
7: 17
8: 19
9: 23
10: 29
11: 31
12: 37
13: 41
14: 43
15: 47
16: 53
17: 59
18: 61
19: 67
```

Figure 1: Example application