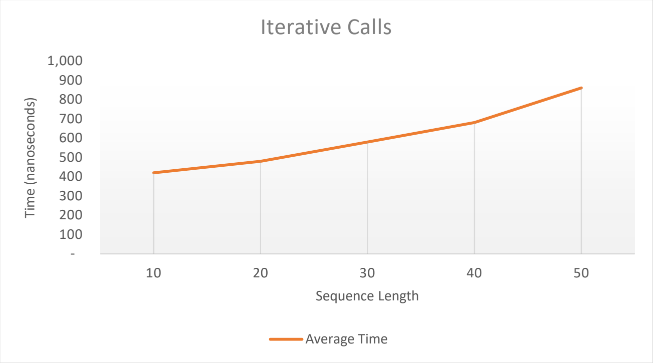
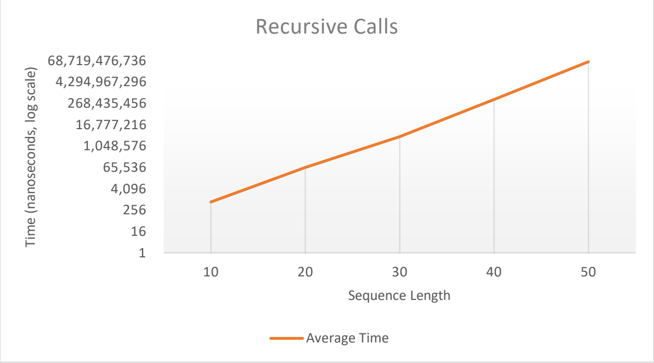
Call Type	Term Length	Run #1	Run #2	Run #3	Run #4	Run #5	Average Time
Iterative	10	1,100	300	200	300	200	420
Iterative	20	1,000	400	400	300	300	480
Iterative	30	900	500	500	500	500	580
Iterative	40	900	700	600	600	600	680
Iterative	50	1,100	800	800	800	800	860
Recursive	10	700	1,700	500	400	400	740
Recursive	20	66,100	103,900	69,900	45,400	43,900	65,840
Recursive	30	3,902,000	3,372,200	3,490,900	3,371,400	3,617,900	3,550,880
Recursive	40	436,355,800	442,960,700	448,699,700	444,652,700	448,958,700	444,325,520
Recursive	50	54,956,136,000	61,174,416,500	61,872,311,500	60,105,766,400	60,322,967,700	59,686,319,620

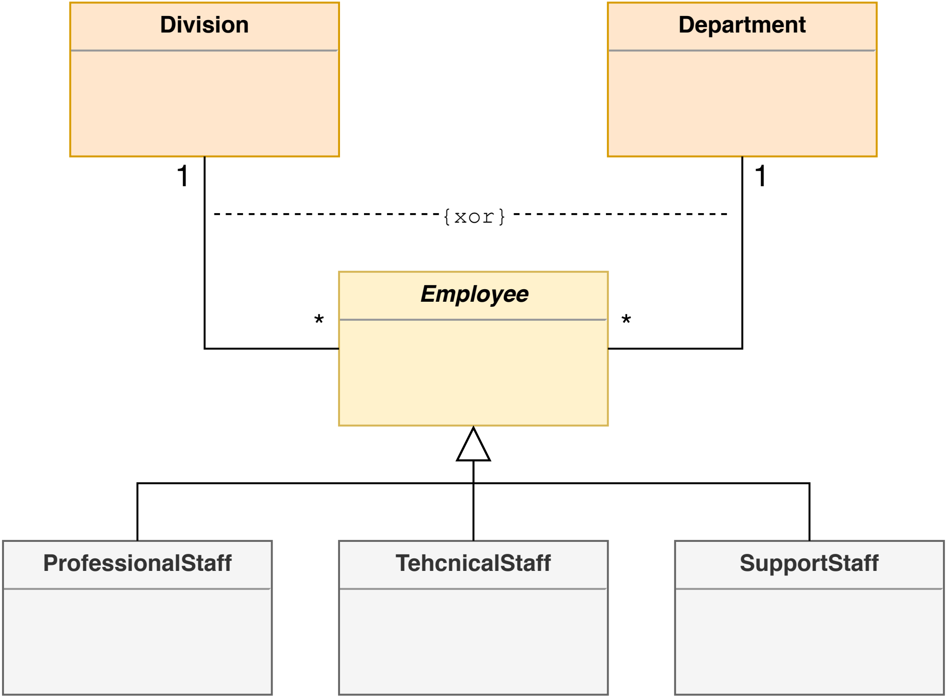


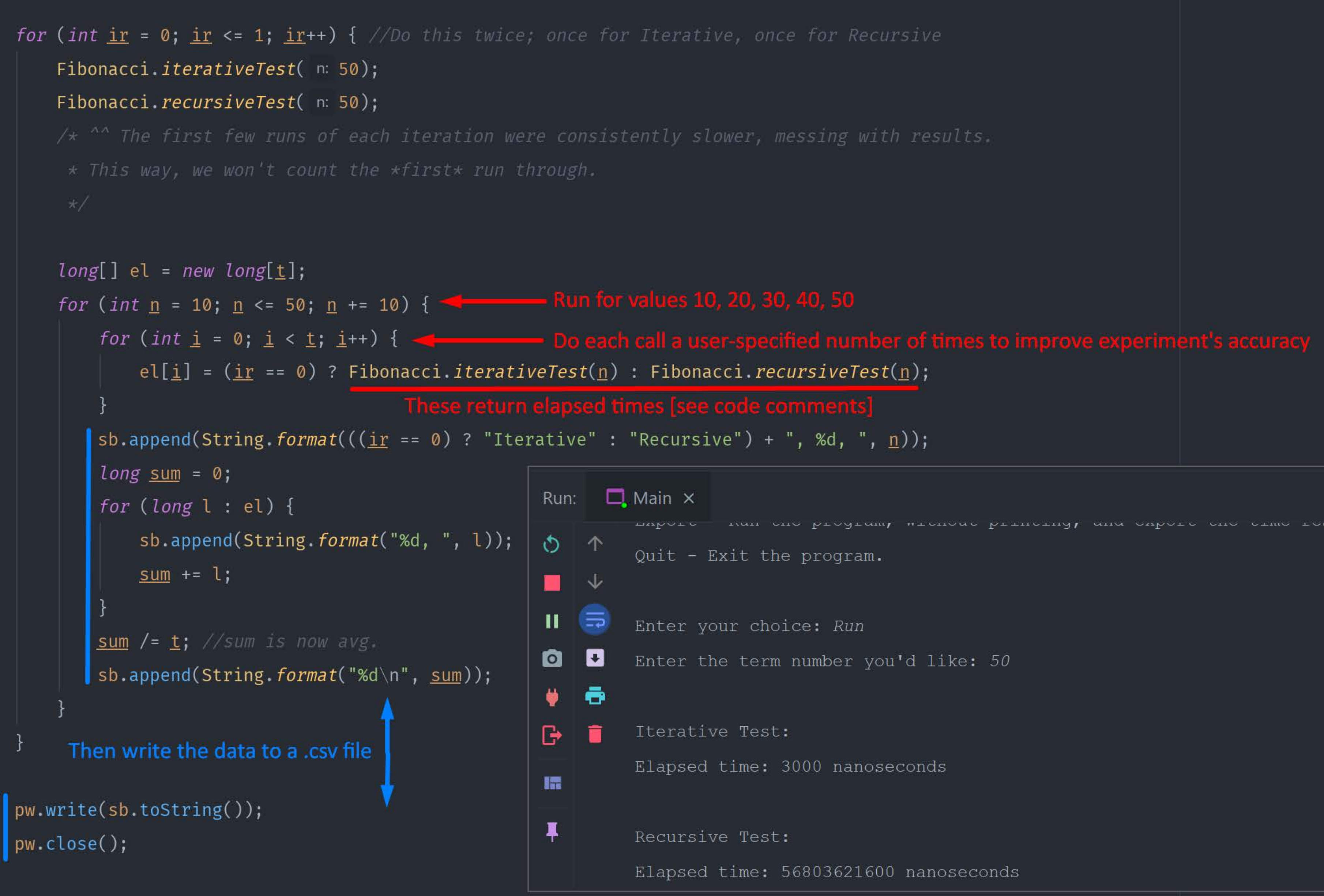


In all cases, the iterative call was faster than the recurisve ones. As the axes of the two graphs show, the iterative method increases in time linearly, but the recursive method increases exponentially. This is because the iterative method requires a single call, each with n runs through a loop, for any sequence of length n. For example, an n-value of 2 requires two runs through a for-loop; an n-value of 4 requiers four runs through a for-loop. The recursive method, however...

For an n-value of 2, it must calls itself once using $n-1=1$ and $n-2=0$, for a total of two calls. For an n-value of 4, it calls itself once using $n-1=3$ and $n-2=2$. The call which has $n=3$ passed into it will call itself with $n-1=2$ and $n-2=1$. Then, the two calls with $n=2$ with, as we calculated before, each call two more times. This means we have, for an n-value of 4, calls itself six times.

One of these is far more complex than the others. The iterative method is $O(n)$ complexity, and the recursive method is $O(n^2)$ complexity[https://goo.gl/BhfqUZ]. This is why one increases in time linearly, and the other exponentially.


Division
Department
1
1
{xor}
Employee
*
*
ProfessionalStaff
TehcnicalStaff
SupportStaff



```
for (int ir = 0; ir <= 1; ir++) { //Do this twice; once for Iterative, once for Recursive
    Fibonacci.iterativeTest( n: 50);
    Fibonacci.recursiveTest( n: 50);
    /* ^^ The first few runs of each iteration were consistently slower, messing with results.
     * This way, we won't count the *first* run through.
     */


    long[] el = new long[t];
    for (int n = 10; n <= 50; n += 10) {
        for (int i = 0; i < t; i++) {
            el[i] = (ir == 0) ? Fibonacci.iterativeTest(n) : Fibonacci.recursiveTest(n);
        }
        sb.append(String.format(((ir == 0) ? "Iterative" : "Recursive") + ", %d, ", n));
        long sum = 0;
        for (long l : el) {
            sb.append(String.format("%d, ", l));
            sum += l;
        }
        sum /= t; //sum is now avg.
        sb.append(String.format("%d\n", sum));
    }
}

pw.write(sb.toString());
pw.close();
```

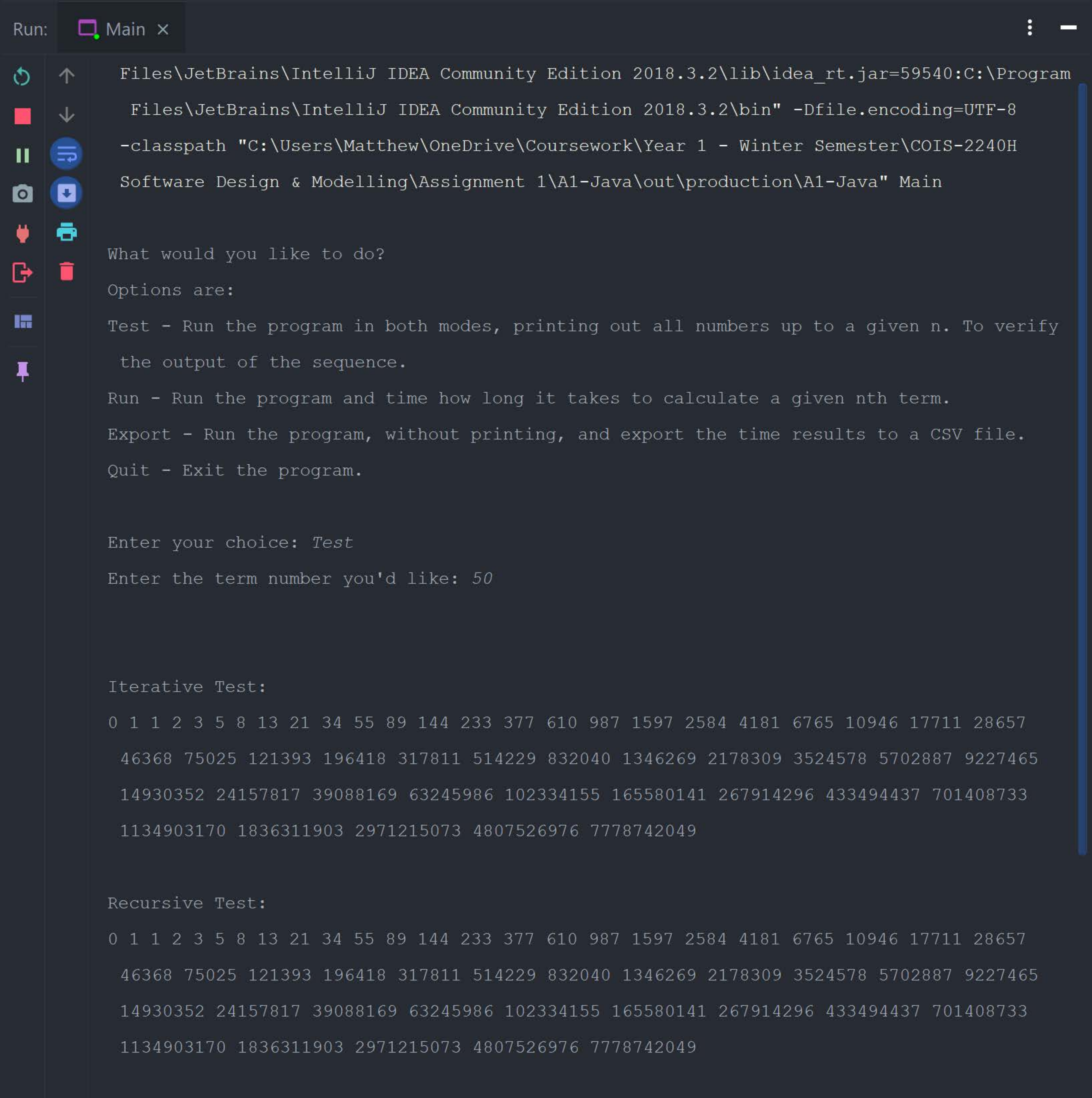
Run for values 10, 20, 30, 40, 50

Do each call a user-specified number of times to improve experiment's accuracy

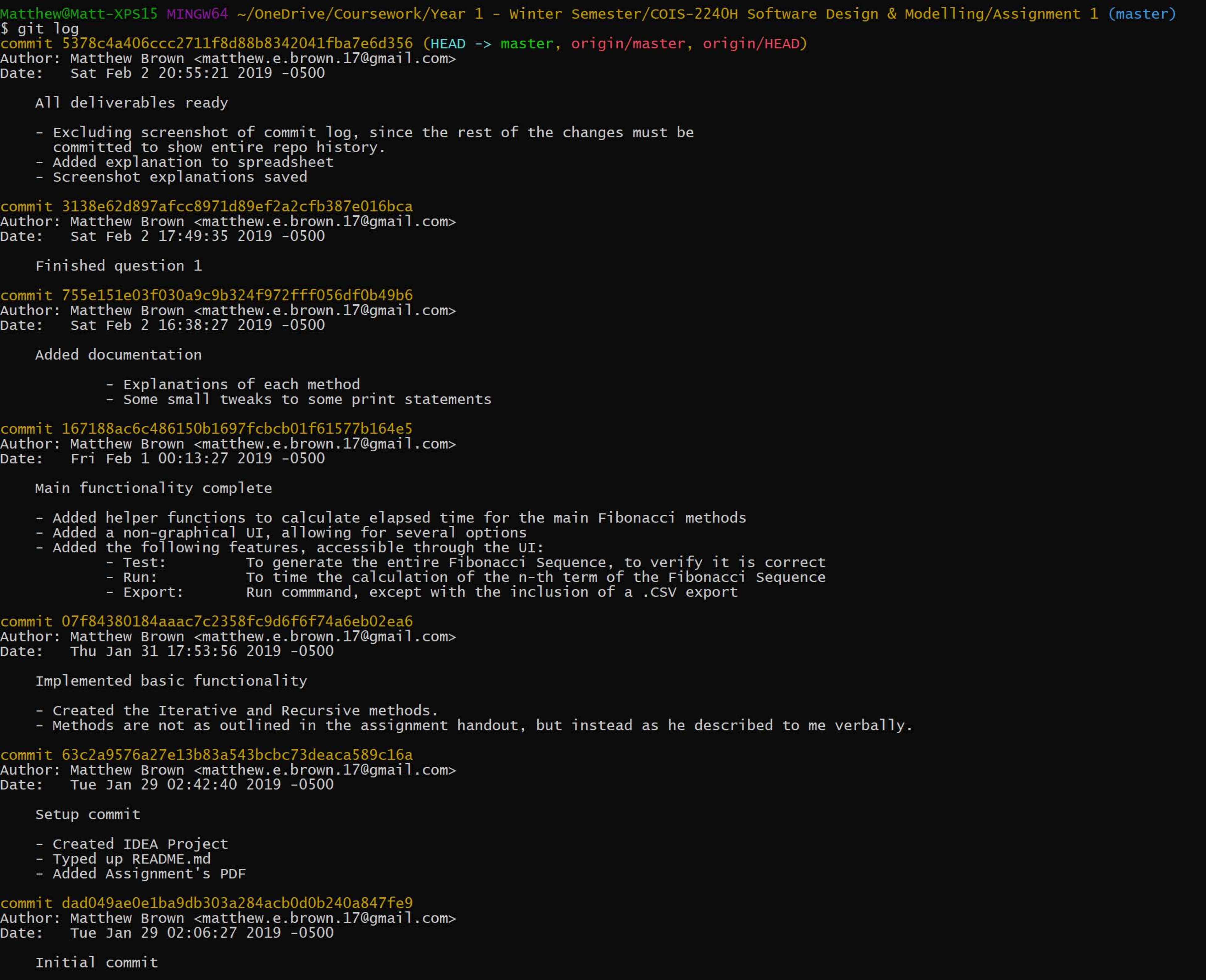
These return elapsed times [see code comments]

Then write the data to a .csv file

```
Export - Run the program, without printing, and export the time fo
Quit - Exit the program.


Enter your choice: Run
Enter the term number you'd like: 50


Iterative Test:
Elapsed time: 3000 nanoseconds


Recursive Test:
Elapsed time: 56803621600 nanoseconds
```



```
 Files\JetBrains\IntelliJ IDEA Community Edition 2018.3.2\lib\idea_rt.jar=59540:C:\Program
  Files\JetBrains\IntelliJ IDEA Community Edition 2018.3.2\bin" -Dfile.encoding=UTF-8
 -classpath "C:\Users\Matthew\OneDrive\Coursework\Year 1 - Winter Semester\COIS-2240H
 Software Design & Modelling\Assignment 1\A1-Java\out\production\A1-Java" Main

What would you like to do?
Options are:
Test - Run the program in both modes, printing out all numbers up to a given n. To verify
 the output of the sequence.
Run - Run the program and time how long it takes to calculate a given nth term.
Export - Run the program, without printing, and export the time results to a CSV file.
Quit - Exit the program.

Enter your choice: Test
Enter the term number you'd like: 50



Iterative Test:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657
 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465
 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733
 1134903170 1836311903 2971215073 4807526976 7778742049

Recursive Test:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657
 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465
 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733
 1134903170 1836311903 2971215073 4807526976 7778742049
```



```
Matthew@Matt-XPS15 MINGW64 ~/OneDrive/Coursework/Year 1 - Winter Semester/COIS-2240H Software Design & Modelling/Assignment 1 (master)
$ git log
commit 5378c4a406ccc2711f8d88b8342041fba7e6d356 (HEAD -> master, origin/master, origin/HEAD)
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Sat Feb 2 20:55:21 2019 -0500

    All deliverables ready

    - Excluding screenshot of commit log, since the rest of the changes must be
      committed to show entire repo history.
    - Added explanation to spreadsheet
    - Screenshot explanations saved

commit 3138e62d897afcc8971d89ef2a2cfb387e016bca
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Sat Feb 2 17:49:35 2019 -0500

    Finished question 1

commit 755e151e03f030a9c9b324f972fff056df0b49b6
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Sat Feb 2 16:38:27 2019 -0500

    Added documentation

            - Explanations of each method
            - Some small tweaks to some print statements

commit 167188ac6c486150b1697fcbcb01f61577b164e5
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Fri Feb 1 00:13:27 2019 -0500

    Main functionality complete

    - Added helper functions to calculate elapsed time for the main Fibonacci methods
    - Added a non-graphical UI, allowing for several options
    - Added the following features, accessible through the UI:
            - Test:         To generate the entire Fibonacci Sequence, to verify it is correct
            - Run:          To time the calculation of the n-th term of the Fibonacci Sequence
            - Export:       Run commmand, except with the inclusion of a .CSV export

commit 07f84380184aaac7c2358fc9d6f6f74a6eb02ea6
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Thu Jan 31 17:53:56 2019 -0500

    Implemented basic functionality

    - Created the Iterative and Recursive methods.
    - Methods are not as outlined in the assignment handout, but instead as he described to me verbally.

commit 63c2a9576a27e13b83a543bcbc73deaca589c16a
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Tue Jan 29 02:42:40 2019 -0500

    Setup commit

    - Created IDEA Project
    - Typed up README.md
    - Added Assignment's PDF

commit dad049ae0e1ba9db303a284acb0d0b240a847fe9
Author: Matthew Brown <matthew.e.brown.17@gmail.com>
Date:   Tue Jan 29 02:06:27 2019 -0500

    Initial commit
```