As discussed, here is the New Relic code challenge. In general, we'd like to get this back within roughly 48 hours, and you can expect to spend 3-6 hours total on it.  Please send your solution back to me within a couple of days.

I've linked [a copy of a template project](https://drive.google.com/drive/folders/1vECqI4asvp66mCCui3xsGbBizj8tK-Oc) that you may wish to use to bootstrap the build process, but there's absolutely no requirement to use it if you don't wish to. Please note though, that within this project there is an implementation started that should help you get going.

From that link, you should be able to download, but not view, a zip archive.

When you are done, please submit your solution in one of the following ways:

1. You may zip up a directory and send that to me. Note that .zip files can sometimes be problematic in email, though a .7z, for instance, works in our filters currently.
2. If you have a GitHub account or similar and wish to share your solution via a private repo that you've created, that is fine. Alternatively, if you prefer to use a cloud storage solution and can share a zip that way, that works.

Let me know if you have any questions and good luck!  
  
**New Relic Code Challenge**  
  
Write a server (“Application”) in Java that opens a socket and restricts input to at most 5 concurrent clients. Clients will connect to the Application and write any number of 9 digit numbers, and then close the connection. The Application must write a de-duplicated list of these numbers to a log file in no particular order.  
  
**Primary Considerations**

* The Application should work correctly as defined below in Requirements.
* The overall structure of the Application should be simple.
* The code of the Application should be descriptive and easy to read, and the build method and runtime parameters must be well-described and work.
* The design should be resilient with regard to data loss.
* The Application should be optimized for maximum throughput, weighed along with the other Primary Considerations and the Requirements below.

**Requirements**

1. The Application must accept input from at most 5 concurrent clients on TCP/IP port 4000.
2. Input lines presented to the Application via its socket must either be composed of exactly nine decimal digits (e.g.: 314159265 or 007007009) immediately followed by a server-native newline sequence; or a termination sequence as detailed in #9, below.
3. Numbers presented to the Application must include leading zeros as necessary to ensure they are each 9 decimal digits.
4. The log file, to be named "numbers.log”, must be created anew and/or cleared when the Application starts.
5. Only numbers may be written to the log file. Each number must be followed by a server-native newline sequence.
6. No duplicate numbers may be written to the log file.
7. Any data that does not conform to a valid line of input should be discarded and the client connection terminated immediately and without comment.
8. Every 10 seconds, the Application must print a report to standard output:
   * The difference since the last report of the count of new unique numbers that have been received.
   * The difference since the last report of the count of new duplicate numbers that have been received.
   * The total number of unique numbers received for this run of the Application.
   * Example text for #8:

Received 50 unique numbers, 2 duplicates. Unique total: 567231

1. If any connected client writes a single line with only the word "terminate" followed by a server-native newline sequence, the Application must disconnect all clients and perform a clean shutdown as quickly as possible.
2. Clearly state all of the assumptions you made in completing the Application.
3. Tests are provided to exercise all of the primary considerations and requirements.

**Notes**

* You may use common libraries in your project such as Apache Commons and Google Guava, particularly if their use helps improve Application simplicity and readability. However the use of large frameworks, such as Akka, is prohibited.
* Your Application may not for any part of its operation use or require the use of external systems, for example Apache Kafka or Redis.
* At your discretion, leading zeroes present in the input may be stripped—or not used—when writing output to the log or console.
* Robust implementations of the Application typically handle more than 2M numbers per 10-second reporting period on a modern MacBook Pro laptop (e.g.: 16 GiB of RAM and a 2.5 GHz Intel i7 processor).