This is a take-home interview programming assessment. The exercise should be returned within a 24-hour period of receiving it.

Please provide the following as output:

- Java source code
- Appropriate binaries (jar, etc) to run the solution
- Instructions on how to execute the binaries
- Mention any assumptions you made for the solution
- Any other adequate supporting artifacts
- Submitted code should be of production-level quality

---- Start of problem ----

Simulate a "commit log" where you have multiple writers and readers.

The commit log should be written to a text file.

To keep things simple each entry in the commit log will have the following structure:

CID: <Unique-ID>: Data <CRLF>

Where

CID - the commit entry ID
Unique-ID - a unique identifier for a commit log entry
Data - an arbitrary string that represents data

Example entries are:

A: 120: Some sample data was written
B: 99: Sample Entry for CID=B
A: 121: Transaction was committed here
B: 100: Some other sample data being logged

Rules
=====
1. Each writer will only write entries for one type of CID. So you have "A" writers that will add entries with CID="A" and "B" writers for CID="B" and so on.
   E.g.
   An "A" writer will produce entries like:
   A: 1100: Some sample data was written
   A: 1111: <Data/>
2. You should have multiple writers for each commit ID (CID).
3. No two entries for the same CID can be written with the same unique ID.
   E.g. The following is invalid
   A: 100: I was written by writer A1
   A: 100: I was written by writer A2

In this example the Writer A2 should have written an entry with a unique-id that is not 100.

4. Similar to the writers, each reader can only read entries for one type of CID. So you have an "A" reader and a "B" reader and so on.
5. You can have multiple readers for the same code.
6. Readers just output the entries to the console as it reads from the commit logs.
7. Make assumptions for any details that are missing. Clearly document your assumptions.

Program
=======
Implement this solution in java. To keep it simple, write one program to kick off the writers and another program to kick off the readers. Use a property file (or a mechanism of your choice) to configure the readers/writers.
e.g. if you want to use a props file

```
#two writers for CID=A
A=2
#three writers for CID=B
B=3
#Rolling log file name
name=commit.log
```

To easily simulate this you could (don't have to) implement each writer to insert into the rolling log in a loop where it could use the loop counter as the unique ID. (Keep in mind that rule#3 applies to multiple writers)
E.g.

```
A: 0: Data from Thread T1
A: 1: Data from Thread T2
B: 0: Data from Thread T3
```

---- End of problem ----