1. Viết chương trình nén dữ liệu time-series lấy từ api bằng phương pháp kiểm tra biên độ giao động. Nếu data thay đổi lớn (vượt ngưỡng so với trước đó) thì sẽ ghi lại ngay giá trị đó, nhưng nếu data không có sự thay đổi lớn nào thì định kỳ sau một thời gian nhất định vẫn phải ghi data đó xuống.
   - Lưu ý: mỗi lần lấy data mới thì so sánh với dữ liệu cuối cùng được ghi nhận gần nhất (vượt ngưỡng hoặc sau thời gian), không phải dữ liệu query của lần trước đó.
   - Tạo maven project, sau đó add thêm 2 thư viện logback, và json để parse một dữ liệu json từ một webservice tự viết (giả lập để trả lại một object json thống kê, trong đó có field `user` là số ngẫu nhiên từ 4000 đến 4500). Phân tích dữ liệu json để lấy field `user`. Lưu ý, dữ liệu query từ webservice trên phải thay đổi sau mỗi lần gọi.

   Thực hiện phương pháp nén dữ liệu time-series khi lấy được từ api trên với ngưỡng biến đổi 2% theo phương pháp sau:
   a. Sử dụng ScheduledExecutorService lập lịch lấy dữ liệu mỗi 2s một lần
   b. Nếu số user trong field `user` ở json trả ra mà lớn hơn 2% so với **số user lần cuối cùng ghi vào file log** thì sử dụng logback ghi số user vào file log với level là INFO, nếu không thì sau 12s ghi lại số user đó vào file log với level DEBUG.
   c. Khi truy vấn chỉ cho phép API trả data trong vòng 10ms. Nếu quá 10ms api không trả kết quả hoặc resp bị lỗi thì ghi log level ERROR với nội dung lỗi tương ứng (timeout hoặc content error).
   d. Setup file log rolling sau mỗi 1 phút ghi ra một file mới. Một file dung lượng tối đa là 1MB.
   e. Tổng dung lượng lưu của cả thư mục log không quá 20MB, và thời gian lưu trữ file không quá 1 ngày.

   Ví dụ data lấy được như sau kèm với action ghi log theo level nào:

0s	2s	4s	6s	8s	10s	12s	14s	16s	18s	20s	22s
1032	1030	1092	1091	1092	1090	1089	1090	1132	1032	1232	1232
Info	Ko ghi	Info	Ko ghi	Ko ghi	Ko ghi	Ko ghi	Ko ghi	Debug	Ko ghi	Info	Ko ghi

2. Viết chương trình đơn giản để crawl data từ một kho url. Các Url này là các bài báo của báo Thanh Niên, Tuổi trẻ, Kênh 14, Cafef. Hoặc là một sản phẩm của điện máy xanh. Tạo maven project với project encode là utf-8. Sử dụng thư viện lấy nội dung html của các url trên và nội dung của bài báo hoặc thông số kỹ thuật của sản phẩm được ghi lại. Dữ liệu kho URL và nội dung sau khi crawl về sẽ lưu ở SQLite, data được backup trên disk.

Tiếp theo build toàn bộ project thành một file jar duy nhất chứa tất cả các thư viện đi kèm (sử dụng plug-in *jar-with-dependencies* của maven).

Tạo một máy ảo linux sử dụng Docker hoặc subsystem (lưu ý không cài các bài linux có giao diện). Tiến hành cài openssh trên máy ảo, Sử dụng một tool SSP (như xmanager, PuTTY, powershell…) trên máy thật và remote ssh vào máy ảo sử dụng ssh-key mà **không sử dụng phương pháp remote bằng password**.

Copy file jar vừa build vào trong máy ảo sử dụng câu lệnh SCP. Sau đó dùng shell script của linux (file .sh) để chạy file jar mỗi 5 giây một lần.

3. Viết một webservice đơn giản với việc trả lại danh sách số nguyên tố từ 0 đến số truyền vào. Tạo một project viết một restful webservice (có thể sử dụng sparkjava.com) trả lại dãy số nguyên tố từ một đến n nhập vào từ param trên url của request (Ví dụ: http://localhost:8080/prime?n=10000 sẽ trả lại danh sách các số nguyên tố từ 1 đến 10000).

LƯU Ý: Project của 7 bài tập nếu có upload lên git thì phải để ở dạng private.

Hãy chạy chương trình trong chế độ debug, giả sử request vào api với số n=12571, sử dụng đặt điều kiện vào breakpoint để dừng chương trình tại các số nguyên tố tìm ra mà giá trị của nó nằm trong khoảng 1.000 đến 3.000. **Lưu ý:** không thêm các câu lệnh rẽ nhánh vào code để thực hiện debug.

4. Hãy tự design một cache riêng cho webservice ở bài 3 để giúp giảm số lần tính toán nếu người dùng thường xuyên truy vấn các số n giống nhau. Sử dụng cache bằng dạng Map<> để trả lại ngay kết quả cho số muốn n nhập vào nếu số n đã tồn tại trong cache. Ngoài ra, Cache này có thêm TTL (thời gian sống cho từng phần tử):

- sau khi ghi một phần tử mới vào Cache thì sau *m* giây thì sẽ tự xóa phần tử này đi
- nếu sau *n* giây không có request đọc phần tử nào đó trong Cache (sử dụng hàm get vào phần tử đó) thì Cache cũng tự xóa đi.

Viết cache này bằng cách tạo ra một Class mới là `CacheTTL<K, V> implements Map<K, V>`, trong đó đảm bảo những phương thức sau:

- `CacheTTL(int n, int m)` : hàm khởi tạo với 2 giá trị tham số *n, m*
- `V get(K key)` : hàm trả lại value tương ứng với key
- `void put(K key, V value)`: hàm đẩy cặp giá trị tương ứng vào cache
- `Map<K,V> getMap()`: hàm trả lại tất cả các thành phần còn lại trong cache
- Int getHitRate(): hàm trả lại tỷ lệ hit khi sử dụng cache.

5. Thay thế cache trên bài 4 bằng thư viện Guava cache, vẫn xóa sau 10s không có request và 20s sau khi ghi vào cache. Upload project lên github.com, đồng thời cùng lúc làm 2 việc sau để tạo confict code:

- Vào project trên github, mở một file trên trình duyệt, sửa nội dung file thêm dòng /* Chinh sua tren server github */, sau đó lưu lại
- Vào project trên máy, cũng vào file đó nhưng thêm dòng /* Chinh sua tren server may client */, sau đó Commit và Push lên git server

Hãy merge và edit conflict 2 đoạn code trên và sửa 2 dòng trên thành dòng comment sau: /* Chinh sua hop nhat giua server & client */ NOTE: PROJECT ĐẨY LÊN GIT KHÔNG NÊN ĐỂ PUBLIC.

6. Deploy project bài 5 thông qua git clone và build file jar bằng lệnh của maven trên máy ảo (docker hoặc subsystem). Hãy set heap size tối đa khi chạy chương trình là 512MB, và heap size khi mới khởi tạo là 125MB.

Sửa webservice trên, yêu cầu phải có bước đăng nhập trước khi sử dụng. Sau khi đăng nhập thành công mới có thể sử dụng api. (http://localhost:8080/prime?n=10000 Lưu ý: Không viết giao diện đăng nhập). Yêu cầu mỗi user không request quá 2 lần mỗi 5s và 10 lần mỗi 1 phút.

LỰA CHỌN LÀM 1 TRONG 2 BÀI SAU:

7.1. Viết một ứng dụng client – server để đồng bộ dữ liệu giữa các client với nhau dựa vào http restful.

Mỗi client khi tạo ra sẽ cài đặt để monitor một folder chỉ định từ lúc start client. Nếu folder đó có bất kỳ khi thay đổi nào thì thay đổi đó sẽ được đồng bộ cho tất cả các client khác đang connect đến Server.

Server có trách nhiệm truyền tải thông tin giữa các client với nhau. Các client cần đăng ký với server để đồng bộ hóa dữ liệu. Khi client ngắt kết nối đến server trong một khoảng thời gian, khi nó connect lại thì phải tự đồng bộ các thay đổi mới nhất.

Lưu ý: Client ko được mở port hay chứa một webserver của riêng nó. Webserver chỉ được tạo trên server. Folder mà client monitor không chứa các thư mục con khác. Và mỗi file trong thư mục không quá 1MB.

LƯU Ý: Project của 7 bài tập nếu có upload lên git thì phải để ở dạng private.

Sau khi viết code xong, thì build ra 2 file jar: server.jar và client.jar. các file này sẽ được chạy lên trực tiếp từ dòng lệnh từ hệ điều hành với đối số truyền vào của server.jar là server_name và đối số truyền vào của client.jar là client_name (tên task client) và đường dẫn đến folder mà client đó cần phải đồng bộ. Định nghĩa message types: `HELLO`, `FILE_CHANGE`, `DELTA_REQUEST`, `HEARTBEAT`, `RECONNECT`.

Ví dụ việc build để khi khi chạy server và client lần lượt như sau:

- `Server: java -jar server.jar server_name`
- `Client 1: java -jar client.jar client1_name path/folder1`
- `Client 2: java -jar client.jar client2_name path/folder2`
- `Client 3: java -jar client.jar client3_name path/folder3`

7.2. Viết ứng dụng instance chat đơn giản.

Server có tính năng (api) sau đây:

- Đăng nhập
  - Method: POST
  - Input:
    - Username (body)
    - Password (hashed) (body)
  - Output: Access Token
    - Access token được lưu trữ trên database của user, với thời gian tạo và thời gian hết hạn. Nếu Access token bị hết hạn sẽ không thể sử dụng được các api phía dưới.
- Lấy danh sách bạn bè:
  - Method: GET
  - Input:
    - Access Token (header) (không gửi thêm user name)
  - Output:
    - Json: list<Username> là bạn của user đang request
- Send message:
  - Method: POST
  - Input:
    - Access Token (header)
    - Username: là username của người nhận (body)
    - Message (body)
      - Body có thể là text hoặc ảnh hoặc file bất kỳ.
      - Nếu message là dạng file thì lưu file đó vào thư mục storage/
  - Output: Status:
    - Nếu người nhận online (đang chờ long polling) thì trả lại là 1
    - Nếu người nhận không online thì cho message vào hàng chờ của người nhận, và trả lại là 2.
    - Nếu người gửi không nằm trong danh sách bạn bè của người nhận thì sẽ trả lại là 3 và không cho message vào hàng chờ của người nhận.

LƯU Ý: Project của 7 bài tập nếu có upload lên git thì phải để ở dạng private.

- Get new messages:
    - Method: GET (long polling) trả lại khi có message mới, nếu không 10s sau sẽ trả lại danh sách rỗng.
    - Input:
        - Access Token (header)
    - Process:
        - Trả lại ngay danh sách message trong hàng chờ của user request lên, nếu user đó có ít nhất 1 message trong hàng chờ.
        - Đợi tối đa 10s, nếu không có message mới thì sẽ resp lại một danh sách rỗng
    - Output:
        - List<Time, Sender, Message>: nếu dữ liệu người gửi là dạng file, thì chỉ trả lại một link download file tương ứng.
- Get file
    - Method: GET
    - Input:
        - Access Token (header)
        - File name (url param)
    - Output:
        - File
        - Nếu file không tồn tại, thì trả lại status 404. Lưu ý: các user không thể truy cập vào file của nhau.

>> Database trên server sẽ được lưu trữ dưới dạng các file json. Hãy tạo sẵn 5 user, mỗi user có ít nhất 1 bạn.

Client: Sử dụng Postman collection, tạo đủ các request cho bài này để giả lập cho việc 2 user đăng nhập, lấy danh sách bạn bè, gửi message và nhận message mới.

LƯU Ý: Project của 7 bài tập nếu có upload lên git thì phải để ở dạng private.