Project Report: Password Processing System

Overview

This project is a Java-based passwords processing system designed to categorize, index, and manage password files. It processes password files, categorizes them based on their starting character, stores them in an organized structure, and provides functionalities for searching and hashing passwords. The system employs multithreading for efficient file processing and ensures that the processed data is stored in a structured manner.

Components and Functionality

Classes and Methods

1. **'passwordProcess' Class**
    - **Constructor:** Initializes necessary directories ('Processed' and 'Index')
    - **'processPasswords' Method:** Processes all password files int the 'src/Unprocessed-Passwords' directory. Utilizes multithreading for concurrent file processing.
    - **'processFile' Method:** Reads each password file, categorizes passwords, and stores them in appropriate index files.
    - **'writePasswordsToFile' Method:** Writes categorized passwords to index files, ensuring no duplicates.
    - **'writeFile' Method:** Handles the actual writing passwords to files.
    - **'getFileCount' Method:** Determines the number of files in a folder with a specific prefix.
    - **'moveFile' Method:** Moves processed files to the 'Processed' directory.
    - **'hashPassword' Method:** Converts byte arrays to hexadecimal strings.
    - **'notifyUser' Method:** Prints notifications to the console.
    - **'savePassword' Method:** Saves a password to a specified index file.
    - **'searcPassword' Method:** Searches for a password in the index and hashes it if found; otherwise, saves the new password.

Libraries Used

- **'java.io.*':** For file handling (reading, writing, moving files).
- **'java.nio.file.Files':** For reading all lines from a file.

- **‘java.securit.MessageDigest’:** For password hashing using different algortihms.
- **‘java.util.*’ :** For collections and threading utilities.

Indexing Process

The indexing process categorizes passwords based on their first character. The steps include:

1. Reading passwords from files.
2. Determining the index based on the first character (special characters are grouped under “others”).
3. Storing passwords in files named after their starting character in the ‘Index’ directory.
4. Ensuring no duplicate passwords within the same index file.
5. Splitting files into chunks of 10000 passwords if necessary.

Folder Structure

- **‘Processed’ Folder:** Stores files that have been processed. Each file originally located in **‘src/Unprocessed-Passwords’** is moved here after processing.
- **‘Index’ Folder:** Contains subfolders for each index category (a-z, others). Each subfolder contains text files with passwords starting with the respective character.

Results

The project efficiently processes and categorizes passwords into indexed files, supports concurrent file processing to enhance performance, and ensures that passwords are hashed using multiple algorithms for security. The indexing and searching functionalities are well-structured, allowing quick retrieval and management of passwords.

Search Function Implementation

The **‘searchPassword’** method processes user quaries by looking up passwords in the index:

1. **Extract Index:** It determines the subfolder based on the first character of the query.
2. **Search File:** Reads through the relevant file to find the queried password.

3. **Password Found:** If the passwords is found, it hashes the passwords using MD5, SHA-1, and SHA-256, and prints the result.
4. **Password Not Found:** If the passwords is not found, it saves the new password to the index.

Measuring Search Time

To measure the search times for randomly selected passwords, the following steps were followed:

1. **Random Selection:** Randomly select 10 passwords from the index.
2. **Measure Time:** In an IDE, you can measure the time taken to find a password. This can be observed in seconds. By summing the time taken for each search and dividing by the number of searches performed, you can calculate the average search time.

The goals we set for the project were fast indexing and quick password queries. Indexing is achieved quite rapidly thanks to the indexing thread, while password querying takes no more than 1 second, often even less than that. We can say that the desired values are being reached.

Future Enhancements

To further improve the project, the following enhancements can be considered:

1. **Error Handling and Logging:**
    a. Implement a comprehensive logging framework to record errors, warnings, and informational messages.
    b. Enhance error handling to manage exceptional cases more gracefully.
2. **User Interface:**
    a. Develop a graphical user interface (GUI) to interact with system, making It more user-friendly.
3. **Database Integration:**
    a. Integrate a database system to store passwords instead of using flat files, which will improve scalability and retrieval times.

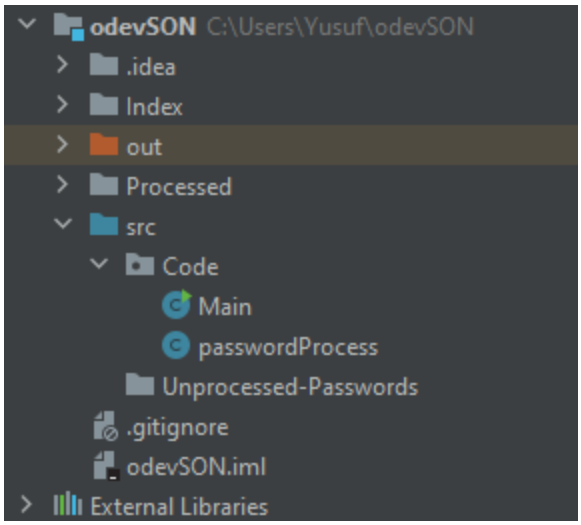
4. **Security Enhancements:**
    a. Implement additional security measures such as encryption for stored passwords.
    b. Ensure secure handling and transmission of passwords.
5. **Performance Optimization:**

    a. Optimize file reading and writing operations to handle larger datasets more efficiently.
    b. Utilize more advanced multithreading techniques to further enhance processing speed.
6. **Advanced Searching Capabilities:**
    a. Implement more sophisticated search algorithms and support for pattern matching.
7. **Automated Testing:**
    a. Develop a suite of automated tests to ensure the robustness and reliability of the system.

By incorporating these enhancements, the password processing system can be made more robust, secure, and user-friendly, thus providing a comprehensive solution for password management and indexing.

Documents:

- https://docs.oracle.com/javase/tutorial/essential/io/index.html
- https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html
- https://www.w3schools.com/java/
- https://www.baeldung.com/java-runnable-vs-extending-thread
- https://www.geeksforgeeks.org/hashing-in-java/
- https://medium.com/




```java
no usages
public class Main {
    public static void main(String[] args) throws IOException {
        File unprocessedFolder = new File( pathname: "Unprocessed-Passwords");
        File processedFolder = new File( pathname: "Processed");
        File indexFolder = new File( pathname: "Index");

        passwordProcess PP = new passwordProcess();
        PP.passwordProcess();
        PP.processPasswords();
        PP.searchPassword( query: "_a7ccpklm8w_", indexFolder);
        PP.searchPassword( query: "123456789", indexFolder);
        PP.searchPassword( query: "ó081jdsh", indexFolder);
        PP.searchPassword( query: "_z_a_q_", indexFolder);
        PP.searchPassword( query: "wollimann55", indexFolder);
        PP.searchPassword( query: "xKkVVCchC1", indexFolder);
        PP.searchPassword( query: "usmc", indexFolder);
        PP.searchPassword( query: "myprofile", indexFolder);
        PP.searchPassword( query: "eltipo", indexFolder);
        PP.searchPassword( query: "=Yh3z#8!C27sbZ4i", indexFolder);
```

```
Password found: _a7ccpklm8w_|7dcdc6c57bdbba19964e9b838e276d71|895c4d5213538b92df8a6ae0fe1a67dc9f741f70|744a69f3dc83e930180c11fa1c8786b039df632afab6ab2da4e627b94f02fd24|_1.txt
Password found: 123456789|25f9e794323b453885f5181f1b624d0b|f7c3bc1d808e04732adf679965ccc34ca7ae3441|15e2b0d3c33891ebb0f1ef609ec419420c20e320ce94c65fbc8c3312448eb225|11.txt
Password found: ó081jdsh|e0b6ea8590242e23acdó6bcb4cf3cb1b|e6208aa6206181f8a8e34673f89eb629221f3a33|258ff2d1cebd31db8411c05d56d827b7310b5f854c33ce4a5e9e815580c98727|62.txt
Password found: _z_a_q_|a4375485df708cdce0f487e8404d3508|b6ad094526274da1fb2e101c4a74974083c4dc4c|a922aad72a4b0df16389e3d2cd5b9aebd5fecde2c73a033b69e6ceeb5c12420e|_1.txt
Password found: wollimann55|5fea7c4ade73d3b6b22b97238c84fe4d|0dddba7d5ef3b09acf85bcf3d638fcf2870daa92|61ff0d8469774840540 2e105e84381f587c2845160a538f4b839a957597c278c|w4.txt
Password found: xKkVVCchC1|760b9b0f3e28efbdefbaef48bd450f04|de8248c9c87a500bfcae29e0223370969ca16a1a|c1241002c26b6c3301abcf79bd4e84c1a856bc1a5f755a4ec555d57acb989449|x3.txt
Password found: usmc|923f21c9d31cf1e356b3e4a3a75670b9|b6c6336736f1647f5636f7c9c0c8ce3afebefde7|91dbdd4a1c1bbe3af464f2618161e115977d02a5d49cc0e23313937c31580143|u1.txt
Password found: myprofile|6bbdb3cdcf42ddb90fdd39dd86aed4d3|f34276d0518706a22b98954f4ef2b0a19b2040f4|829053c1d6a19a6aa07fc48a0397ea0e2511a0ac5baa7189f9c050de54389949|m3.txt
Password found: eltipo|df2eebf5bdc63e8aa6ed4f83b008538d|01975d0efa56115b80e068751cbc576b0d98efa2|3b461fea09db8d1b49321924ca7da9db070d3a45d1e3091e2df3add90f4b4713|e2.txt
Password found: =Yh3z#8!C27sbZ4i|13926b077938b8f0518c4824a29e5c77|9d712da8c2340f11c6f5e26d12d46aeb8c0bb475|6055fa34714e43c1b850e5ada8c3e26e5452681ae5b2685857578bef690dc6da|=1.txt
```

13066220078

Yusuf Asan