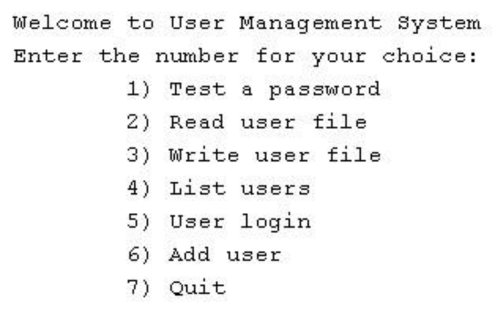
CIS 331 Assignment #4
Working with Methods: Login with Cryptographic Hashing

DESIRED PROGRAM BEHAVIOR

In this assignment, you are to write a program that behaves as described below. It should loop repeatedly, presenting a menu for the user to select from, like this:



```
Welcome to User Management System
Enter the number for your choice:
        1) Test a password
        2) Read user file
        3) Write user file
        4) List users
        5) User login
        6) Add user
        7) Quit
```

The user will enter a choice, and the program should perform the appropriate operation. The following are done:

1) Test a password:

User enters a password. The password must satisfy all these requirements:

- be at least 8 characters long
- contain at least 1 lowercase letter
- contain at least 1 uppercase letter
- contain at least 1 numeric character
- contain at least one special character (such as $, %, &, *, etc. anything on the keyboard that is not a number or letter).
- PW cannot contain a space.

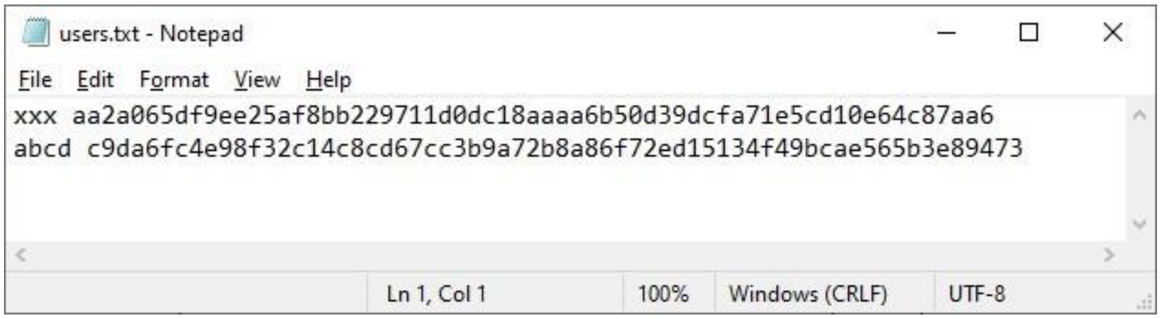
Here are some examples of user entries and the results: Enter password to test abcd abcd is a BAD password

Enter password to test aMx12#37 aMx12#37 is a GOOD password Enter

```
password to test abcD1234
abcD1234 is a BAD password
```

2) Read user file

The user enters the name of a file. As an example, I've provided a sample user file called **users.txt**:



```
xxx aa2a065df9ee25af8bb229711d0dc18aaaa6b50d39dcfa71e5cd10e64c87aa6
abcd c9da6fc4e98f32c14c8cd67cc3b9a72b8a86f72ed15134f49bcae565b3e89473
```

Each line contains a user ID and a hashed password, separated by spaces.

Note that the length of the hashed password is 32 bytes (256 bits). This is shown in the file in hexadecimal (digits in base 16).

When reading this file, the result will be placed in a String array, as described below.

Output will look like this:

```
Enter file name (or full file path) for read users.txt
User file users.txt read.
```

3) Write user file

The user enters the name of a file. The program should write contents of the user array to the file. Output looks like this:

```
Enter file name (or full file path) for write users2.txt
User file users2.txt written.
```

4) List users

When a file is read, the users will be placed in a string array. This option simply lists all users and their hashed passwords in the array. For the sample file provided, the output will look like this:

```
User List:
xxx aa2a065df9ee25af8bb229711d0dc18aaaa6b50d39dcfa71e5cd10e64c87aa6
abcd c9da6fc4e98f32c14c8cd67cc3b9a72b8a86f72ed15134f49bcae565b3e89473
```

5) User Login

Prompt user to enter an id and password. There are four possible outcomes:

1) The password is invalid, for the same reasons as menu option #1 above. Example output:

```
Enter user ID xxx
Enter password baddpw
baddpw is an Invalid password
```

2) The user doesn't exist. Example output:

```
Enter user ID
nobody Enter
password
1Password!
User nobody does not exist
```

3) Login successful:

```
Enter user ID
abcd Enter
password
jmuDukes12!
login successful
```

Note: for the user file provided, there are two users with the following associated passwords (which are hashed in the users.txt file):

xxx with pw **1Password! abcd**
with pw **jmuDukes12!**

You can use these to test the login functionality.

6) Add user

Prompt to get a new user ID and password. Three possible outcomes:

1) The user already exists. Example output:

```
Enter user ID abcd
Enter password xxx
User abcd already exists
```

2) It is a new user, but the password is invalid (same criteria as #1 above). Example output:

```
Enter user ID
newuser Enter
password
badpw
badpw is an Invalid password
```

3) It is a new user with a legitimate password. In this case give a verification message as shown below and add the string containing user ID and the password (separated by a space).

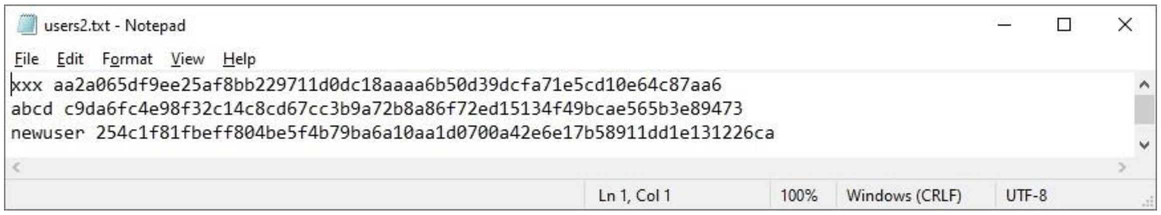
```
Enter user ID
newuser Enter
password
GoodPW1?
User newuser added
```

Note: after successfully adding a user, you should be able to verify by doing another list:

```
User List:
xxx aa2a065df9ee25af8bb229711d0dc18aaaa6b50d39dcfa71e5cd10e64c87aa6 abcd
c9da6fc4e98f32c14c8cd67cc3b9a72b8a86f72ed15134f49bcae565b3e89473 newuser
254c1f81fbeff804be5f4b79ba6a10aa1d0700a42e6e17b58911dd1e131226ca
```

Also, if you write a file, the data should appear in it:

```
Enter file name (or full file path) for write users2.txt
User file users2.txt written.
```



```
xxx aa2a065df9ee25af8bb229711d0dc18aaaa6b50d39dcfa71e5cd10e64c87aa6
abcd c9da6fc4e98f32c14c8cd67cc3b9a72b8a86f72ed15134f49bcae565b3e89473
newuser 254c1f81fbeff804be5f4b79ba6a10aa1d0700a42e6e17b58911dd1e131226ca
```

2) Quit
Say goodbye an exit the program

ERROR HANDLING

Your program must handle any errors that the user has. Anything that might cause an exception must be handled using try/catch, and instead of quitting an appropriate message should appear, and the user should get the menu again. For example, if the user enters an invalid menu choice, the message could look say something like “Invalid choice, please try again”.

If an exception occurs (for example if a user enters a file name that doesn’t exist), you should catch this exception and print an appropriate message.

PROGRAMMING TASKS

For this assignment, you will write a program that includes methods for various user login and password validation tasks. The signatures for these methods and descriptions of the mathematical tasks they perform are listed below:

1) static boolean isValidPassword(String s)

This method verifies that the passed in string is a valid password. Specifically, it ensures the following:

- length is at least 8 characters
- includes at least one upper case character
- includes at least one lower case character
- includes at least one numeric digit
- includes at least one special character
- cannot contain a space

2) static String[] readUserFile(String fname)

This method takes a file name as input and reads the file into a string array. Each line of the file will go into an element of the string array, which is then returned. For this assignment, you can specify a MAX array length, and assume that there will not be more lines in the file than the MAX, similarly to what was done in the previous assignment.

If an error occurs and the file cannot be read, then the method should return **null**.

3) static boolean writeUserFile(String fname, String[] userArray)

This method takes a file name and a string array as parameters, and writes the contents of the array to the specified file. It returns true if the task successfully completes and **false** if not (e.g. if there is an exception because of a bad file name or location).

4) static String listUsers(String[] userArray)

This method simply takes a string array as a parameter and concatenates all values to a string, then returns this string. The values should be separated by end-of-line characters.

5) static String userLogin(String uid, String pw, String[] userArray)

This method takes three parameters: a user ID, a password, and a string array. The string array contains the userIDs and passwords of the currently read file (from the read method above). This method tests to ensure that the login is valid, and returns one of four string messages: the password is invalid, the user doesn’t exist, the user exists but the password does not match, or the login is successful (see outputs shown above).

To test for invalid password, call your already-existing **isValidPassword** method.

It should be easy to test to see if the user exists by searching the array.

If the user does exist, see if the password is correct for that user. For this you can use the sample **GFG** class described in the video lecture, which comes from https://www.geeksforgeeks.org/sha256-hash-in-java/. GFG has two useful static methods. The first, called **getSHA**, takes a string, performs a hash algorithm on it, and produces a byte array of the hashed value. The second, called **toHexString**, converts the byte array to a string with hexadecimal notation, which how the password is stored in each element of the array.So, by first calling getSHA, then passing its result to toHexString, you will have a hash for the password and can compare this to the value for the specified user in the array.

If the passwords match, the login is successful, otherwise you should give a message that the password doesn't match.

6) static String addNewUser(String uid, String pw, String[] userArray)

This method takes three parameters: a user ID, a password, and a string array. The string array contains the userIDs and passwords of the currently read file (from the read method above).

This method tests to ensure the following: (1) the user doesn't already exists and (2) the password is a valid password (call **isValidPassword** for this). If both of these criteria are met, add the user to the next available element in the array. String will contain userid followed by a space followed by hashed password.

Note: similar to the userLogin, you can use the GFG methods to take the password and convert it to a hash string.

NOTE: **NONE** OF THESE METHODS SHOULD INPUT VALUES FROM THE USER OR DISPLAY DATA TO THE USER. They are all to receive their values as formal parameters, and they are all to return values to the caller of the method. All user-input and user display should be done in the **main** method.

NOTE ALSO: Some of these methods are quite simple and I suggest starting with these. Specifically, #2, #3, and #4 are very easy. #1 is a bit more challenging, but shouldn't be too difficult. #5 and #6 will be the most difficult and time-consuming.

NOTE ALSO: Some methods make use of others. If the functionality is already there, use it. Don't reinvent the wheel by coding something you already coded.

THE *main* METHOD

Your **main** method should have a loop that presents a menu of options to the user, and then performs the desired option. For each option, your **main** method will need to ask the user to input the data that will be sent as arguments to the methods, call the appropriate method, and then will display the results based on what gets returned from the method. Again, as stated above, the other methods DO NOT involve any user interaction. They merely take in parameters, perform the appropriate computations, and return results.

The overall structure of the main method will be a loop with a nested decision structure (either an IF statement or a SWITCH). A structure similar to this is shown in the very last slide of the ControlStructures notes.

Also, make sure to do any necessary exception and error handling in the main method.

PROGRAM STYLE

In addition to correct functionality of the program, I expect you to adhere to sound programming style. Use descriptive variable names. Be sure to use proper indentation in your code and supply enough comments to make it clear what the program is doing. Use the camel case notation for your variable, method, and class names.

At the top of your program listing, you need to include comments with:

1) your name and peoplesoft number
2) the course and section
3) the assignment number
4) a statement assuring me that this work was done in accordance to the JMU Honor Code.