Program 4: A User Database

CS 617 Winter 2013

1 Goals.

1. To use a text file for output and later for input.
2. To use exceptions and write an exception handler that does more than abort execution.
3. To implement phase 1 of a well-structured project with three classes.
4. To use an ArrayList.

2 Overview

This section discusses a finished application that you will construct in two phases. When all phases are finished, it will simulate a logon and authentication process for a group of users. The first user created is "root" and has administrative privileges. Later users have fewer privileges. Eventually, all passwords will be encrypted before storing them in a file.

This will be a menu-driven application allowing for these actions (in the final version):

1. The administrator creates the first user, himself. Then he logs in.
2. The administrator creates other new users and adds them to the Users collection. To do this, he must be logged in.
3. The administrator can display a list of all users in the database. For this assignment, display the full name, logon name, and password of each user. In P5, the passwords will not be displayed.
4. The administrator can change anybody's password.
5. A user who is logged in may change his own password.
6. A user whose name is in the collection may log in and out. Logging in will automatically log out the prior user.
7. Finally, the menu has an option to quit.

Program 4 implements only part of the menu and does not encrypt the passwords.

3 The User Class: Part of the Model for this application.

- A `User` will have three data members, all type String. The real name and logon name are both final variables. The password can be changed.
- Provide a constructor with three parameters that initializes all three data members.
- Implement get functions for all three. (accessors)
- Implement a set function for the password. (mutator)
- Implement a `toString()` function that will print the real name and logon name. P4 should display the password also, but this will be dropped from P5. Put each user on a single line of the output. The format should be readable but does not need to line up in neat columns.

4 The Main Class for P4

This class should contain only a very brief main function. Print an identifying line (program name and your name), instantiate an `Admin` object and call its `doMenu()` function. Surround the code with a `try` block and catch `IOException`s. When caught, print a comment, a stack trace, and abort.

5 The Admin Class for P4: the controller for this application.

- The Admin class should contain an ArrayList of Users, and variables to store the current user (a reference to a User in the ArrayList), the UID of the current user (an int, the position of the user in the User collection), a menu, a Scanner for the keyboard, a Scanner for the user-file, and a `PrintWriter` for the user file.
- You may use this to define the menu:

```
private static String menu = " 1. New User\n 2. Log in\n"
" 3. Change Password\n  4. Log off\n  5. List users\n  6. Quit";
```

- In this phase of the program, we will implement menu options 1, 5, and 6.

The Admin constructor. The first time this program is run, the file of user data (userFile) will not exist. The Admin constructor should print a clear comment about the missing file, then continue with normal execution. On the second and later executions, the file will exist, and the Admin constructor must read it, create a series of User objects, and store them in the ArrayList, then close the Scanner (which closes the input file) and reopen it for output. Write the following code:

- Write a `try` block. It should do three things:
  - Open the userFile file and a Scanner to read it. Save the Scanner pointer in the appropriate class member.
  - Read all the data using a normal loop and hasNext().
  - Close the stream.
- Write a `catch` block for FileNotFoundException. Display a comment about not finding the file. DO NOT abort execution; this is normal the first time you run this application.
- Note: IOException should never happen because of the way things are being done. If it does happen, let the exception pass up to main. You don't need to handle it here.
- At this point, the data from the file has been read, if it existed. The next step is to open an output file with the same name as the now-closed input file.
  - Put this code in a try block that immediately follows the catch block from the earlier try. Control will come here after the first try-block or its catcher finishes its work. Notice that we are reentering execution after an exception.
  - For the second try block, write an IOException catcher. Any IOException that happens here is because of a serious problem with reading the file. Print a comment and re-throw the exception to main.

At the end of the constructor, the program and its user list are ready for use.

The doMenu() function. Write a main loop that displays the menu forever, until the user selects "Quit".

- Prompt the user for a menu choice and process that choice with a switch. Do not use an if. . . else structure.

- In the switch, do NOTHING except call one of the functions below. When the program is converted to Swing, this switch will be replaced by a bunch of JButtons.
- Put the entire menu loop in a `try` block with a `finally` clause.
- In the finally clause, write the Users in the user list to the output file, close the file, and print a message on the screen.
- Break out of both the switch and the `for(;;)` statement if the user selects "Quit". Leaving the loop will end the try block, which will send control directly to the finally block.
- After the finally block, control should return to main, where it hits the end of another try block and ends the program.

The Admin class needs a private utility function.

- `getPassword()` prompts the user to enter a password twice and compares the two Strings. When an equal pair is entered, it returns the validated password.

Admin needs public functions for two menu options.

- `newUser()`: Create a new user (about 20 lines of code, including whitespace and comments). Prompt for and read the user's full name (assume that it contains space characters) and the logon name (no whitespace). Call the `getPassword()` function to get a password. Make a new User and put it into the ArrayList of Users.
- `displayUsers()`: Use a for-each loop to display the Users in the users-collection. Let the `toString()` function in the User class format the user data.
- No function is needed for the "Quit" menu item. Just break out of the loop.

6 Testing and Submission.

Due January 31, 2013 Hand in your three source files and the screen output from two test runs, pasted into a comment below your main program. Hand in copies of the output file from two consecutive runs; the first run should start with no input file and end with two users. The second run should add another user. Use these two runs to test errors in entering the password. The screen output that you hand in should correspond to the file output.