Program 5: Logging On

CS 617 Winter 2013

1 Goals.

1. To use an object file for output and later for input.
2. To use a cryptographic hash function for password security.
3. To finish the program started in P4.

2 Overview and Use Cases

This program will complete implementation of the logon and authentication process for a group of users. The first user created is "root" and has administrative privileges. All passwords are encrypted before storing them as part of a User. Refer to the demo "Password" for guidance on encryption and related issues. This assignment implements encrypted passwords and object files and adds functionality to P4:

1. A user whose name is in the user list may log in and out. Logging in will automatically log out the prior user.
2. A user may change his own password.
3. The administrator can change anybody's password.
4. The administrator is always the first user created. After creating himself and logging in, the administrator can creates other new users and add them to the user list.

3 The User and Main Classes.

Users.

- At the top of the file, add this: `import java.io.Serializable;`
- Add an interface declaration to the first line of the class definition:
  `public class User implements Serializable { ...`
- In `User.toString()` remove the code that prints the password. Encrypted passwords look like garbage and even so, they should not be printed for security reasons.
- In main(), add a `catch` block for `NoSuchAlgorithmException` Print an error comment about "SHA-1 not available" on the stream `System.err`, then abort.
- Write a second `catch` block to handle all possible exceptions. Print an error comment saying that the exception was caught in `main`. Then execute: `ex.printStackTrace();`

4 The Admin Class: the controller for this application.

Change from text files to object files. Encrypted passwords cannot be read properly by a text-oriented Scanner because the password could contain any bytes, including newlines and nulls. To write such things to a file and read them back in again correctly, you need to use an Object file.

- Using the FileDemo program as an example, remove the Scanner and PrintWriter for the user data file. Replace them by an `ObjectOutputStream` and an `ObjectInputStream`.

- In the Admin constructor, remove the loop that reads the input file one line at a time. Replace it by a 1-line call on `readObject()`, and cast the result of the read operation to `ArrayList<User>`. Store the cast pointer in your ArrayList variable.
- In the finally clause at the end of `doMenu()` remove the loop that writes the Users back to the file. Replace it by a 1-line call on `writeObject()` that writes out your entire user ArrayList.

The Admin class needs more private utility functions.

- `findUser( String logon )` locates the user with the given logon name in the collection of Users and returns the index where it was found. Return -1 if the logon is not in the collection. Use `ArrayList.get( int k ), ArrayList.size(), String.equals()`, and an ordinary for loop. When the loop finds the desired user, the loop variable, k, will be the UID of that user.
- `computeHash( String s )` computes the SHA-1 hash function on the password and returns a byte string. Copy this code very carefully from the password demo on the website, week 5.
- `getPassword()` already prompts the user to enter a password twice and compares the two Strings. Change the code so that, when an equal pair is entered, `computeHash()` is called to encrypt the password. Then convert he result to a String and return it.

Add three more action functions to implement the remaining three menu items.

- `Login()` (about 25 lines of code). Prompt for a logon name and password. The logon succeeds if the name matches one of the Users in the collection, AND the input password matches the one for the selected User. Display a "login failed" message if either condition is not met. However, force the User to enter both logon name and password before you give him any information. If the logon succeeds, set the `activeUser` to point at the logged-in User, and set the `activeUID` to the position of that User in the collection.
- Restrict the use of `displayUsers()` to the system administrator.
- The `changePassword()` function has two cases. If a non-root user is logged in when this menu item is selected, the method is a 1-liner. Using functions you have defined elsewhere, get a new encrypted password and set the password field of the active User.

  If the root user is logged in when this menu item is selected, he is permitted to change anyone's password (about 10 lines). Prompt for the logon name of the person to change, find and get that User, call your function to get a new password, and set the User's password. If the user is not in the users collection, print an error comment.
- Logout is a 2-line function: Set the `activeUser` to null and the `activeUID` to -1.

Testing and Submission. Due February 15 if possible. If not, be prepared with a list of specific problems to ask about. Test all new options thoroughly before you hand this in. Hand in the source code, screen output, and two files, as for P4. However, this time, the files will be object files, not text files. Put the login name and password of your administrator at the top of your main program so that I can test your program if I need to.