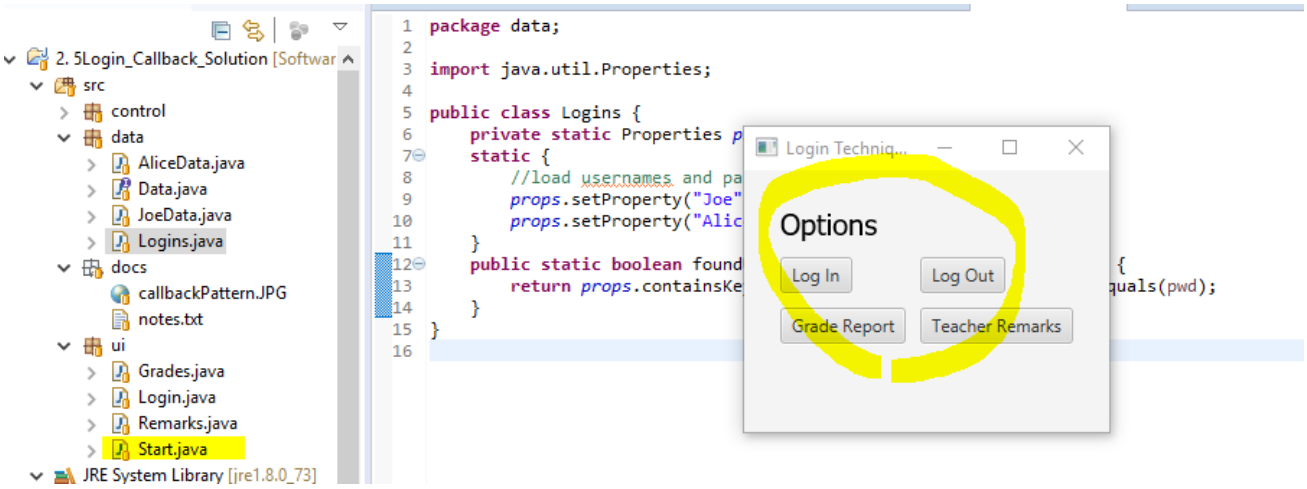
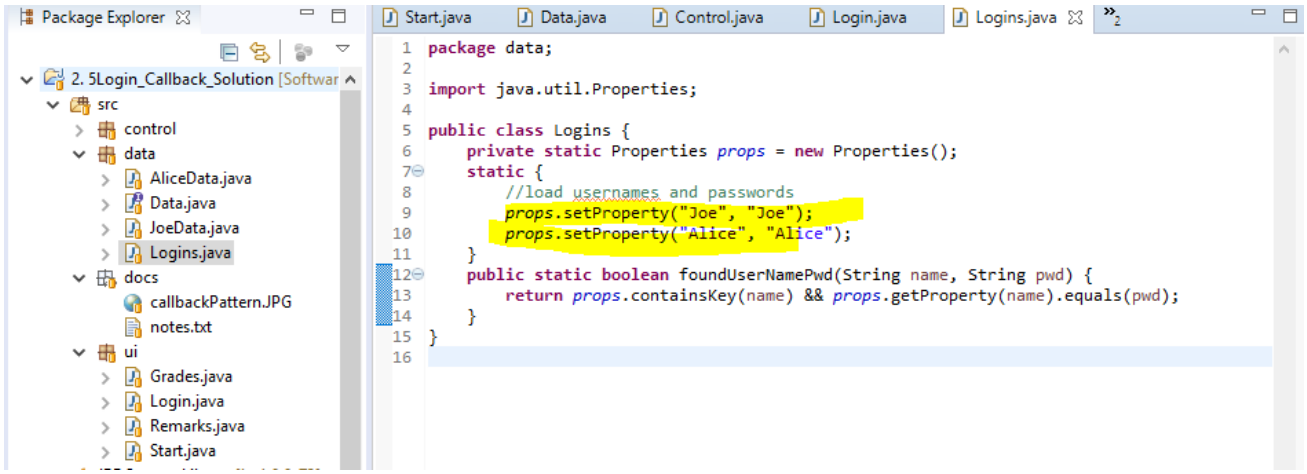
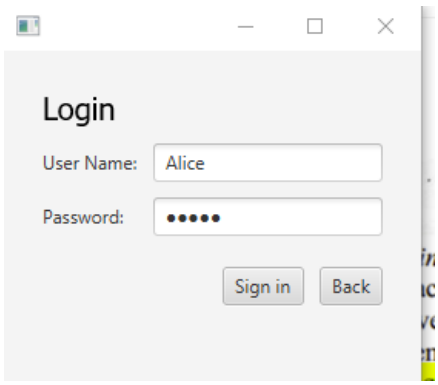
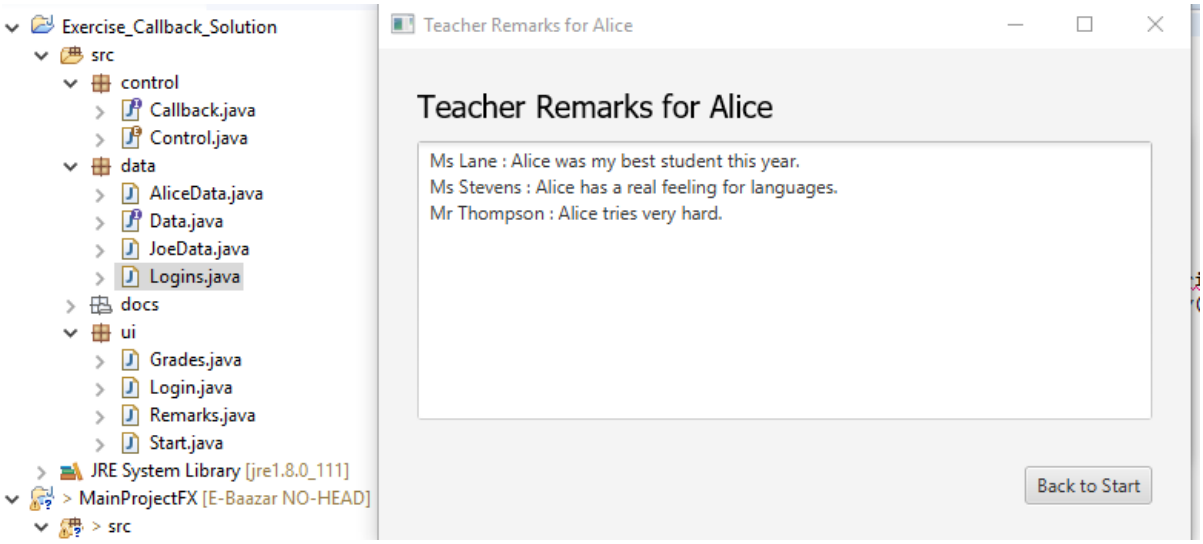
# Callback function

B. *Handling Logins.* In Exercise_LoginAndAuthorization, we learned a technique for keeping track of a user-requested target window even if user's navigation needs to be diverted to a login screen. The exercise in Exercise_Callback has a similar requirement, but now you are required to provide, after the user logs in, not only his requested target screen, but also data, which is specific to the user, that needs to be displayed on his target screen. Instructions are in the docs folder of this project.






About the Solution:

This solution accomplishes 4 things:

(1) User is forced to login if he tries to go to one of the protected screens

(2) User needs to login only once

(3) Application remembers actor's target window when redirected to Login

(4) Application populates target window with correct data after Login

Technique for (1) & (2). When user logs in, a flag isLoggedIn, in Control, is set to true. Whenever user attempts to go to a window, this flag is checked; if false, user is redirected to the Login window.

Technique for (3) & (4). When user attempts to go to window A but has not logged in, he is re-directed by A's handler to the Login window, and a reference to A's handler is passed in. This reference is typed as Callback so that the login use case does not need to know where the request came from. After Login succeeds, it calls the performAction method on the Callback that was passed in. Control returns to A's handler, which includes an implementation of performAction. The performAction method knows which window to create since A's handler always creates this kind of window. The performAction method also knows how to populate the window because it has access to the username (obtained when user logged in), which is used as a key to obtain the necessary data for the window.