Unit 2 Program 6: System Users

We are all familiar with the process of logging into computer systems. The purpose of this project is to create a program that would simulate the behaviors of login control.

For this project you will be creating a pair of interacting classes. The first represents the `User`, and the second represents the user's `Password`. Note that the `Password` will be a separate class, but a `Password` object will be used as a variable within the `User` class. The program is structured this way to demonstrate how we might increase system security by keeping the `Password` separate from other `User` details.

The `Password` class (which should be written FIRST) contains a single `String` instance variable that represents a user's password for accessing a computer system. In addition to the constructor (which accepts a single String as its input parameter), the only methods which need to be written for this class are the `toString` method and a getter for the value of the instance variable. The `toString` method should generate the following output:

```
Password: yourPassWordHere
```

The `User` class should include the following instance variables:

1. String representing the user's ID (`userID`)
2. A `Password` object
3. String indicating this user's `accessLevel` ("Admin", "SuperUser", or "Standard")
4. Boolean value indicating whether the user is currently `loggedIn` (`True` = logged in)

The following items will be required for the `User` class. You should ONLY write the items and methods indicated below.

1. A parameterized constructor which takes three strings representing the user ID, password and access level (in that order) as inputs. Note, the input for the password is a `String`, NOT a `Password` object. The constructor will create the Password object.

   Newly created accounts are logged out by default. In addition, if the input for `accessLevel` is not one of the allowable values ("Admin", "SuperUser", or "Standard"), the user should be assigned "Standard" access rights. Do NOT write a default constructor.

2. A `login` method as described below:
   a. @param `String` – the ID of the user trying to log in
   b. @param `String` – the password of the user trying to log in
   c. @return `boolean` – `True` if log in was successful, `False` otherwise
   d. Description: If the user ID and password provided match the user ID and password of this user, the user is logged in and the `loggedIn` variable is updated. Otherwise, nothing happens.

3. A `logout` method as described below:
   a. @param: none
   b. @return: none
   c. Description: This method logs out the current user (update instance variable!).

4. A `toString` method as described below:
    a. Uses the standard `toString` signature block.
    b. If the account is currently logged out, the `toString` method should generate the following output:

```
Access Denied
```

    c. If the account is currently logged in, the `toString` method should generate the following output:

```
User: defined user ID
Password: defined password
Access Level: defined access level
```

Finally, to validate your program, you need to create a runner class (which contains the main method) which will exercise all of the functions within the project. Your main method should do the following.

1. Create a `User` with the following details:
   ID: Babbage
   Password: 2Bo!2b?Shkspr
   Access Level: Master

2. Print out the `User` just created. (should say Access Denied)

3. Attempt to log in using incorrect credentials

4. Print out the `User` (should say Access Denied)

5. Attempt to log in using the correct credentials

6. Print out the `User` (should print out user ID, password, and an access level of "Standard".

7. Logout

8. Print out the `User` (should say Access Denied)

9. Create a second `User` with the following details:
   ID: Lovelace
   Password: ttls,hI1derwUr!
   Access Level: Admin

10. Log into the account (correct credentials)

11. Print out the `User` (should display details as above in #9)

Your MUST turn in the following items for this project to be complete: the entire project directory (zipped) that includes all classes and the runner described above and screen captures of your tester results and runner results.

Screenshot on Mac: command-shift-3 puts the image on your desktop

Screenshot on PC: hit Print Screen button, open Paint, paste into a new file, and save as a jpg; or use Snip tool