CSE218 HW 2 User Account Management

User Account Management is one of the most widely used features in computer applications. You will be building a simple user account management system in this assignment. When you design the module, keep in mind that some of the fields such as Gender and GPA may not be needed in some applications and some other fields such as credit card number or email address could be used instead. So, write your module as flexible as possible to allow easy modifications. This module will be reused now and then in future assignments, and especially in the final project of this course.

1. The module must use an array big enough to hold at least 5000 user accounts in a bag.
2. You will use the raw data from the last assignments to create 3000 accounts (i.e. 3000 **UserAccount** objects) and place them into the array inside the bag.
3. Each of the 3000 accounts must contain the following information:
    - First Name (not unique)
    - Last Name (not unique)
    - ID (uniformly 8 digits long, unique, String type)
    - Gender (Just male or female for now ☺, based on girl or boy name in the data files)
    - User name. It must be unique, made of the first 4 letters of the last name + the first initial + the last digit of the ID. If the last name is shorter than 4, then use the entire last name for the first part of the user name.
    - Password. Randomly generated. It does not have to be unique but it must at least have the following:
        - (1) a capital letter
        - (2) a lowercase letter
        - (3) one of the following symbols (between double quotes): " !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"
        - (4) a minimum length of 8
    - GPA. Randomly generated between 0.0 and 4.0
4. The array must then be sorted. It must be sorted by user name with the algorithm we discussed in class. When you insert 3000 user accounts objects into the bag, evaluate the efficiency of the following two approaches: (1) sort the array each time you insert an object and (2) sort it only once after all 3000 objects have been inserted. You should verify your evaluation by timing the two different approaches. Write a separate method to do so.
5. Use binary search so any user name and its matching password can be searched as fast as possible.
6. Build GUI so you may test your module. When you enter a user name and password, the GUI will either say “Success!”, when both the user name and password are correct, or “Failure”, when either the user name is not found or it does not match the password.
7. If you are interested, you may also implement a “Sign Up” feature so new user accounts can be created by a user on the GUI. Still, the user name must be unique. If not, your program should know to reject it. And the password must meet the requirements as stated above. Also, the ID must be automatically generated and the GPA must be within the same usual range.

Use abstraction and data encapsulation. It is the foundation of reusability and scalability. Create short and pure methods: one for each task and each task only. Name them all descriptively using proper nouns and verbs if applicable.

Here are some suggestions for the assignment:

You may need a **name-warehouse**, as you did in the last assignment, to import and store names into arrays.

You may need a **user-account-factory**. The factory will contain an instance of the name-warehouse and a bunch of methods to allow you to emit individual data items such as first name, last name, id, gender, user name, password, and GPA, all separately and one at a time. Obviously, the emitUserName() method relies on data provided by methods that emit first name, last name and id. The user account factory will have a method (**emitUserAccount()**) that uses the individual data items emitted from the above methods to create and return **ONE** user account **object**, which is the ultimate goal of the factory.

You may also need a **user-account-bag**. The bag contains a user-account array and will allow you to insert user account objects emitted from the account factory. It will also provide a method to sort the array and two overloaded methods – one searching for any combination of user name and password during log-in, and the other one for searching for duplicate user name during new user account creation.

The bag will take objects from the user account factory to fill the first 3000 accounts. It is done one at a time for 3000 times. The bag may also take data from the GUI to create new account objects if you decide to implement the optional portion of the assignment this time. In this case, information will first be collected from the GUI and sent to the account factory to create an individual account object. The individual account object will then be sent to the account bag to be stored, sorted and searched.

Here is an incomplete list of possible methods you could potentially build, some in the factory and some in the bag. You decide on the parameters to use for each method. Feel free to use them or not to use them at all.

1. public String emitFirstName()
2. public String emitLastName()
3. public boolean isMale(String firstName)
4. public String emitID()
5. public double emitGPA()
6. public String emitUserName()
7. public String emitPassword()
8. public void sortAccounts()
9. public boolean searchAccount(String username, String password)