Lab 2

This first part of the assignment will require you to build the Java objects to be used in an online banking application, and the second part will add the online servlet component with authentication and web components. These first requirements for the Java objects should set you up to simply focus on the Servlet code and how the pages interact with each other once the Servlet is deployed.

Requirements for this assignment are as follows:

1. Create Java objects to contain:
    a. Banking users with usernames and unique identifiers
    b. 3 different types of accounts that can be added to the banking user's object/profile (keeping in mind that a user could have more than one of the same type of account, like 2 savings accounts) that can hold funds as dollars and cents.
2. Methods to (at a minimum)
    a. View account history
    b. View account balances
    c. Sum all balances
    d. Transfer funds between accounts (also not allowing a user to transfer more from an account than they have in the account)
    e. Add accounts (must have a type and a name)
    f. Remove accounts (must have no funds in them at the time)
    g. Logging to show transaction state (could be used for account history)
        i. Think about how you want to store logging data, could be per user or could be mapped using a java.util Class to hold in a RandomAccessFile, or a logfile per user.
3. Maintaining state in a Java File
    a. Storing objects in a Java File
    b. Once a transaction is complete, write new object to file
    c. Used on startup to bring records into memory

Based on the first part of this lab , you will be adding the ability to run this application via a Java Servlet through a web browser. To accomplish this you will first need to download and install a Java web server that supports Servlets such as Apache Tomcat, Jetty, Glassfish, or any other Java web server. I would suggest Tomcat, but any of them will do. Once you have a running

Java web server you will then need to create the appropriate Java Servlets to work with your already created Java objects and methods to be used through a web browser. Since you have objects and methods to manipulate the objects and their data, you will only need to create the Servlets to present information to the browser and then have the information from the browser sent to the Java Servlet for processing. One main addition to this lab will be the ability to “login” to your servlet to identify the user to the server and create state through sessions/cookies. You will only need to have a single user logged in at any time, and it is up to you on “how” the user will login. You can simply have them type in their name, you can add a password if you like, or you can have all the users presented to the “login” screen as a dropdown box in a HTML FORM object. Really any method you prefer to identify the user to the initial Servlet so that the correct object can be identified (if it exists, you should still be able to add “NEW”).

Once you have “logged in” you then will have the ability to manipulate your banking transactions through different servlets. You have the option to have one or more servlets in your application, but I suggest many, and use redirects (sendRedirect or RequestDispatcher).

Since you already have the code to manage your objects in your Java file (serialized), there will not be a need to add code to manage the “file” unless you are doing all of your “writes” to the file at the end of the program, then you will need to either modify writes per transaction, or you will need to have a logout routine to kick off the write of the object to the file. One issue to keep in mind is that you need to manage the “write” / update to the file when it makes sense, and since we do not have state in knowing when a user is “done”, you should write to the file anytime there is a change to the user object. Anything where you are reading data from the user/file, no need to update, only when something changed like an account amount / name / etc.

For the Servlet(s) you will not need to have 100% functionality of a banking application, so here is what I am looking for:

1. Login screen – to identify the user (or add new)
2. Transfer screen – have the ability to utilize a transfer method in one of your objects between accounts. Does not matter which accounts, but you will need to present the user with their accounts, choose which one to transfer from and which one to transfer to. After the transfer the user should be able to see their accounts updated to reflect the change.
3. Add / Delete accounts – per user, be able to add and delete accounts
4. Balance screen – show the user a listing of all of their accounts and balances in the accounts
5. View history – ability to show what was done when with the account. If you add new, you can hard code accounts

What to hand in

You will need to hand in your code in a .war file (Web Archive File) so that I can simply drop the war file into my Tomcat directory for deployment. Please don't hard code any paths/etc for data as it will be relative to my environment. Please name your head directory for deployment your U of M x500 login name (i.e. langa008) so that there are unique applications.

Grading Criteria

- Ability to login and identify the user
- Functionality of the 4 screens listed above
- Logging of transactions
- Must have <META> tags for BACK button Pragma, Cache-control, and Expires

Grading Topic	Description	Points / 100
Login / identify	**User** can be identified and added to session to create state of user data	10
Add Account	**Add** an account per user that will be used with a name, type and $ amount	10
Delete Account	**Delete** account and remove it from the user object	10
View balances	**View** data around account's (name, $ amount, and optional ID/key)	10
Logging	**Log** data around transactions	10
History	Show **history** of transactions to screen	10
Transfer	Ability to **transfer** data between accounts - NOTE: must only allow sufficient funds from one account to another	15
File storage	Hold user objects in **Java file** for offline state	15
Meta tags	Ensure META tags are added to response to client	10