Assignment 3: LinkedIn File IO

We will be building a LinkedIn application throughout the rest of this course. I have created an architecture that I expect you to contribute to. Please read the PDF labeled, *The Software Architecture*, that is available on the course website for a thorough description of the architecture and how you can contribute to it.

To start, download the Eclipse project, *java2-lastname-firstname*, from the course website and import this project into your Eclipse IDE. Rename the project by supplying your last name and first name to the name of the project. You will be building on to this project week-by-week.

Import Instructions

1 – Open Eclipse and select the *File -> Import…* menu option.
2 – Select the *General -> Existing Projects into Workspace* option and click the *Next* button.
3 – Click on the *Select archive file* option and browse to the Zip file that you downloaded from eLearn.
4 – Click the *Finish* button to complete the import process.

After you import the project you should immediately notice that the project doesn't compile. That's expected at this point. You are going to write the code to make this project compile.

The Project

This project is made up of two source folders; *src* and *test*. Both source folders contain an *edu.institution* package. The *src* folder is where you will place source code for the assignment. The *test* folder is where you will place your unit tests for the assignment. All code for this assignment should be placed in the *edu.institution.asn3* package.

All of the packages that you will contribute to have been created in this project for you. There is a *lib* folder which contains all of the supporting JAR files that you will need throughout this course. The *lib* folder is already added to your class path, so you need not worry about that.

Task 1 – Copy Assignment Two

To fix some of the compiler errors, copy your code from assignment two into the *edu.institution.asn2* package of this project. You only need the source code from the prior assignment. You don't need to copy the unit tests that you wrote in the prior assignment.

Task 2 – Implement UserRepository

Package: edu.institution

The *UserRepository* interface has been created for you by the architecture. It defines a class which will add, save, remove, and retrieve the user information that we add to our application.

This interface defines the following methods.

```
void init(String filePath, String fileName);
```

This method should deserialize the data stored at the supplied *filePath + fileName* location into a list of LinkedIn users. If there is no previously saved data, then this method should initialize a new list of LinkedIn users. Depending on how you implement this method, you may find that the other methods defined on this interface may need access to the supplied *filePath*, *fileName*, and to the list of LinkedIn users that is created by this method. If that is the case, then it is recommended that you set these values as properties on the implementing class. See the *SerializedUserRepository* code snippet below for an example on setting these values as properties of the class.

```
void add(LinkedInUser user) throws LinkedInException;
```

This method ensures that the supplied user is ready to be added to the user repository and, if so, adds the user to the list of users established from the *init* method.

If the user name and user type are not supplied, throw a new LinkedInException with the message, "The user name and type are required to add a new user."

Valid user types are either 'P' for premier or 'S' for standard. If the user type is invalid, throw a new LinkedInException with the message, "Invalid user type. Valid types are P or S."

If the supplied user already exists, throw a new LinkedInException with the message, "A user already exists with that user name."

If the supplied user is ready to be added, add it to the list and call the *saveAll()* method to save the data.

```
void saveAll();
```

This method overwrites (serializes) the list of LinkedIn users that was established in the *init* method to the file system.

```
void delete(LinkedInUser user);
```

This method removes the supplied LinkedIn user from the list of LinkedIn users that was established in the *init* method. This method should call the *saveAll()* method to persist the deleted data.

```
LinkedInUser retrieve(String username);
```

This method returns the LinkedIn user associated with the supplied user name or *null* if there is no user associated with the supplied user name.

```
LinkedInUser retrieveAll();
```

This method returns all LinkedIn users in the repository or an empty list if there are no users in the repository.

SerializedUserRepository

Package: edu.institution.actions.asn3

You are to create this class and implement the methods described in the *UserInterface*. Your project should compile cleanly once this class is created and properly implementing the interface.

```
public class SerializedUserRepository implements UserRepository {

 @Override
 public void init(String filePath, String fileName) {
  // if you need access to these values in any other method on in this class,
  // then set them as properties of the class
  this.filePath = filePath;
  this.fileName = fileName;

  // deserialize the list of LinkedIn users and set the list as a property of
  // this class
 }

 @Override
 public void add(LinkedInUser user) throws LinkedInException {
 }

 @Override
 public void delete(LinkedInUser user) {
 }

 @Override
 public List<LinkedInUser> retrieveAll() {
  return new ArrayList<>();
 }

 @Override
 public void saveAll() {
```

```
  }

  @Override
  public LinkedInUser retrieve(String username) {
   return null;
  }
}
```

Remember: to serialize and de-serialize the *LinkedInUser* class, it (along with its **super** class) must implement `Serializable`.

Task 3 – List All Users Menu Action

Package: edu.institution.actions.asn3
Class: ListUserAction

```
public class ListUserAction implements MenuAction {
  // implement the process method
}
```

This action should retrieve all of the LinkedIn users from the user repository and print each user's user name to the console. The *process* method for the action should return *true* to keep the user signed in.

Task 4 – Sign Up a New User

Package: edu.institution.actions.asn3
Class: AddUserAction

This action should do the following:

(Use the supplied Scanner instance to prompt the user for information. Do not create your own instance of Scanner in any action)

- Prompt the user for the user name to add.
- prompt the user to enter a password and what type of user he/she is.

```
Enter a username
gwen

Enter a password
gwen

Enter the type of user (P or S)
x

Invalid user type. Valid types are P or S
```

    - construct a new *LinkedInUser,* setting the supplied user name, password, and type, and call the *add* method on the supplied user repository.
        - If the *LinkedInException* is thrown, catch it and display the message on from the exception.
- Return true to keep the user signed in.

The *process* method for the action should return *true* to keep the user signed in.

Task 5 – Delete a User

Package: edu.institution.actions.asn3
Class: DeleteUserAction

(Use the supplied Scanner instance to prompt the user for information. Do not create your own instance of Scanner in any action)

- Prompt for the user name to delete. Check if the supplied user name exists in the user list.
- If the user does NOT exist, display an error message to the console and re-display the menu.
- If the user does exist, prompt for the password of the user being deleted.
- If the password is not correct, display an error message to the console and return out of the action.
- If the password is correct, call the delete method on the user repository passing the user name.
- If the deleted user was the logged in user, then the action should return *false* to tell the Application Controller to sign out the current user. Otherwise, return true to keep the user signed in.

Task 6 – Sign Off

Package: edu.institution.actions.asn3
Class: SignoffAction

This action is easy. Simply *return* false to tell the Application Controller to sign out the current user.

How You Are Graded

1. The application properly serializes and deserializes the entered data (4 points)
2. The application properly lists the users as specified in this assignment (1 point)
3. The application properly signs up a new user as specified in this assignment (2 point)
4. The application properly deletes a user as specified in this assignment (2 point)
5. The user can sign off as specified in this assignment (1 point)

Notes

1. You are not required to write JUnit tests for this assignment.
2. Your project is sent to a service which tests and attempts to grade your assignment. That automated grading process is much easier if you follow the instructions with regards to class names, method names, and output messages. I do evaluate your project after the automated grading occurs to help you understand any mistakes that was pointed out and possibly adjust any points that was deducted from your project.
3. Your project is sent to a service that checks for plagiarism. If the service flags your project for plagiarism, I evaluate it against the student it matched to. If I determine that your project was plagiarized, you will receive a score of zero on the assignment. Continued offenses could result in disciplinary actions taken by this institution.
4. If your program does not compile, you will receive a score of zero on the assignment
5. If your program compiles but does not run, you will receive a score of zero on the assignment.
6. If your Eclipse project is not exported and uploaded to the drop box correctly, you will receive a score of zero on the assignment.
7. If you do not submit code that solves the problem for this assignment, you will not receive any points for the program's compiling or the program's running.