Group 3

Andrew Kim

Luke Brown

Sebastian Jagiella

Job Search App Project

Initial Design

Seeker
This class takes the personal data of a Job Seeker and generates a unique text file for that user, named after their username. It stores separate information about the Seeker as separate instance variables. This information includes the Seeker's name, date of birth, and their highest level of education, as well as any previous Jobs the Seeker had, and any relevant skills they possess. The information can be viewed using the toString method. Besides these variables, the Seeker class also stores the user's username and password. The class also contains getter and setter methods for each variable. Any time a setter method is used to change user information, the updatePersonalFile method is called to change the corresponding data in the user's unique text file.

Employer
This class takes the data of a Job Employer and generates a unique text file for that user, named after their username. It stores separate information about the Employer as separate instance variables. This information includes the company name, a description of the company, and a list of any Jobs they have posted. The Employer's username and password are also stored as instance variables in the class.

Job
This class stores the details of a single Job listing as separate instance variables. The information stored includes the Job title, location, a description of the Job and its requirements, and its status, such as open, filled, etc. The class also contains getters and setters for each of these variables.

Seeker Driver
This class contains the main method where the options for the user are implemented. The user is prompted to enter their username (the program catches if they aren't a previous user) and they are asked to make an account. If they already have an account, they are asked to enter their

password which is tied to the username via a HashMap, and they are redirected based on a Master File for both Seekers and Employers which recognizes the identity of the user based on where the username is located. If they are a new user, they are asked if they are a Job Seeker or an Employer. The appropriate class is generated based on the previous input. The menu created is tailored for either the Seeker or the Employer with the specified options for both. The user has the option to enter in the number associated with each option in order to "go into" that particular option. A sub-menu is created for each option where the user can accomplish certain tasks. Viewing the profile displays the profile of the individual (using the getters tied to the class). Updating the profile updates the specific attribute of the user (using setters tied to the class) and automatically updates their file. Both classes have an option to see a list of the other class (Seekers can see open Jobs, and Employers can see the Seekers). This is implemented by creating an ArrayList of either type and saving it to a file after creation. The user is able to sign out, which returns them to the home page. Finally, the user has an option to delete the account. In this case, the associated files that will be tied to that individual's unique username are removed and the program is terminated.

Final Design

Changes
A static ArrayList of applicable Jobs was added to the Employer class to keep track of each of them. A toString() method was added to the Job class to more easily print the ArrayList<Job>. A deletePersonalFile() method and a deleteEmployerFile() method were added to the Seeker and Employer classes, respectively, in order to clear data when the program terminates. A new static final long variable, serialVersionUID, was added to the Job, Seeker, and Employer classes, because each class implements Serializable in order to save the files in case of termination.

Testing

The program was tested alongside its creation in order to minimize confusion and debug more efficiently. We decided to use only one main driver - "JobSeekerDriver" - instead of three separate drivers for each class. This was chosen because of how inefficient it would be to pull in drivers specifically for each class since bits and pieces would have to be placed in specific locations due to the choices from the user. It made more sense to have one main driver and test as we implemented. Some errors included an unhandled NullPointerException when searching for a Job from an Employer that does not exist. This was fixed by ensuring that the Employer exists beforehand. Another error was discovered when a user does not terminate the program after deleting their account or a Job listing where the information does not update, and therefore is still displayed. This was rectified by ensuring the user terminates after such actions. Further testing was performed by multiple trial runs, none of which encountered errors.