Project 6
CSCI4448-OOP
4/28/2020

Project Name

Let Me In - Job Application Tracker

Team Members

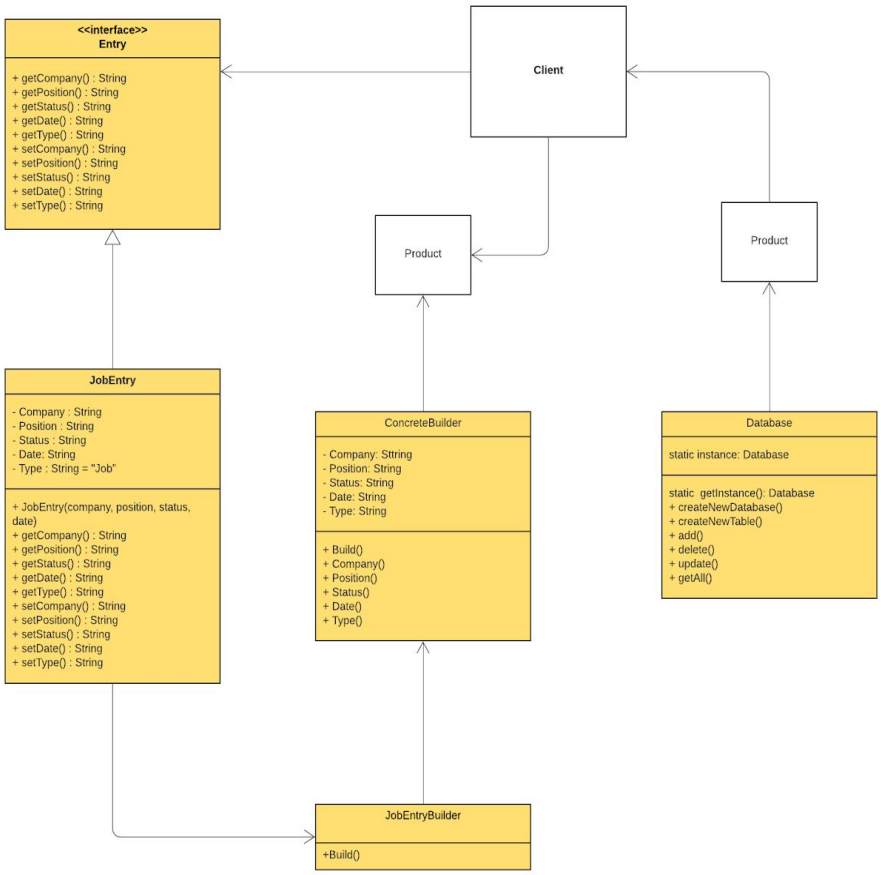
Andy Kim, Justin Vuong, Sabrina Touch

Final State of System Statement

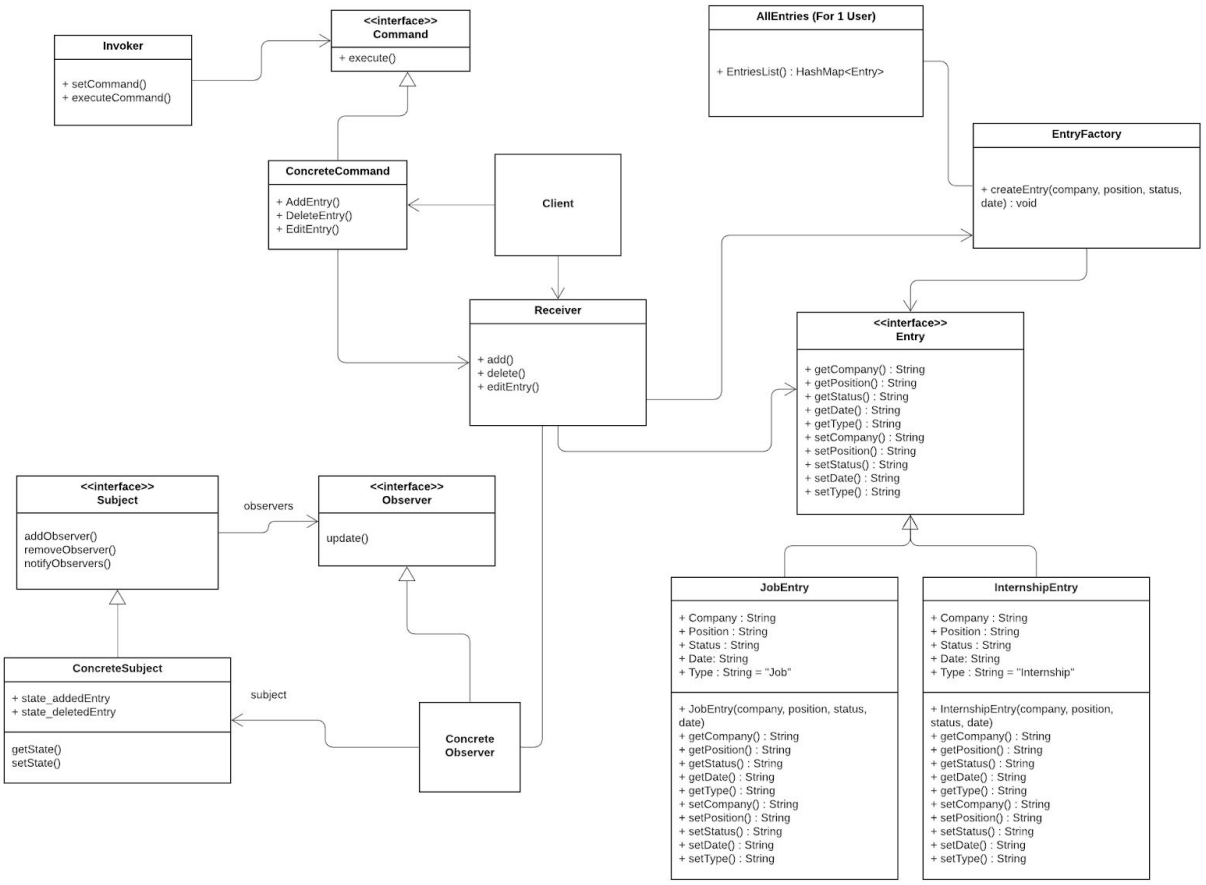
We were able to end up implementing all the features that we wanted for our program. The features that are a part of our system include adding a new job/internship entry, deleting any entry, and updating any entry. Due to the use of our builder pattern, we're able to add a job without the required parameters such as the date.

Final Class Diagram and Comparison Statement

- Final UML Diagram:



- UML Diagram from Project 4:



- We made some key changes in our class diagram for our final project. We originally had factory as one of our creational design patterns, but we ended up using the builder pattern instead. We decided to use the builder pattern, because builder pattern allowed users to leave out some fields blank in our system. For example, if the user did not remember the day he/she applied to a specific job, they have the option to leave it blank. However, the factory pattern provided a stricter format and required users to fill in all of the fields. We also removed the command pattern, because it overcomplicated our system. Since our functionalities are relatively simple, (adding an entry, editing an entry, deleting an entry) we found that forcing patterns into our application was not feasible. We added the singleton pattern to our design, because it allowed for us to create one instance of a database for the user. We don't want to create multiple instances of databases for a given user, so a singleton pattern was an effective way to prevent this issue. We also removed the observer pattern in our design, because we originally thought that observers will allow us to display messages on the front-end side. We realized that print statements on the back-end side cannot be shown through the front-end side, which defeated the purpose of our application using the observer pattern.

Third-Party Code vs Original Code Statement

- **Original Code**

The code for the patterns we implemented were original with some help from the lecture slides.

- **Third-Party Code**
  The code we used for the backend, database, frontend, and API calls were all inspired by code from third-party sources.
  - Full-stack application example using Spring and Maven
    https://www.springboottutorial.com/spring-boot-react-full-stack-crud-maven-application
  - Consuming a RESTful API
    https://pusher.com/tutorials/consume-restful-api-react
  - Using SQLite in Java
    https://www.sqlitetutorial.net/sqlite-java/
  - Material UI for the front-end
    https://github.com/mui-org/material-ui/tree/master/docs/src/pages/getting-started/templates/dashboard
    https://material-ui.com/components/buttons/#button

Statement on the OOAD process for your overall Semester Project

1. Figuring out the difference between the Factory and Builder patterns and when to use each one was something that was important in the design of the semester project.
2. We initially wanted to implement the Command pattern, but had complications with trying to figure out how it would work alongside the use of API calls and our endpoints. We decided to omit the Command pattern and rather just use a central “controller” that handled all the calls coming in from the front-end and direct them towards their specified task in the database.
3. We thought about the usage of the observer pattern wrong since our group imagined that the observer would connect frontend objects to the backend which wasn’t the case. This helped us understand the pattern better since we now understood that it’s all about updating the subscribed backend objects.