Project name: Big Bank Take Little Bank

Names: Jonathan Phouminh, Bao Nguyen, Zachary Chommala

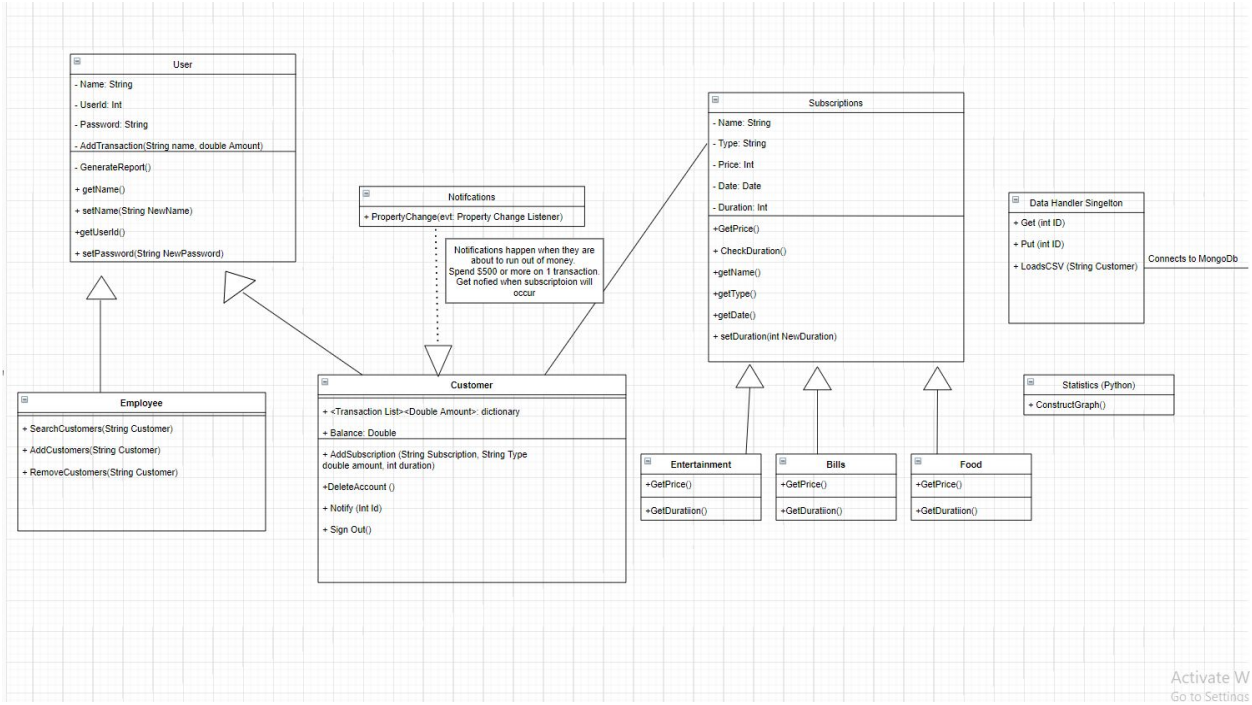
Submission Details: The link to the github we provided is the link for the backend code with all the OOAD patterns. Within that repository will be a link to this PDF as well as instructions to be able to run the .jsp files with Maven. The link to the front end repository will be https://github.com/ZacharyChommala/OopFinalProjectHtmlWork.git

Final State of of the system:

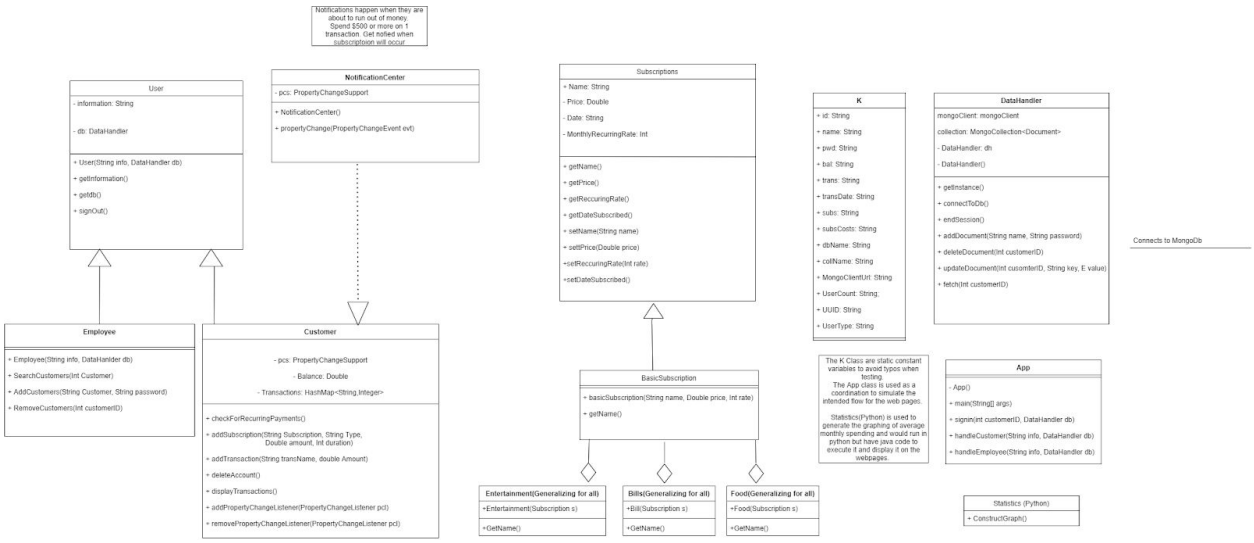
Our final project consists of a simulation of what the program would look like on web pages. We were unable to connect our backend data handling logic with our web pages. So In our presentation we will talk about how the simulation would have corresponded with the web pages, and showed what our web pages would look like and how they would behave. We had an observer pattern used to notify the customers of events like overdrafting. We had a decorator pattern to add subscriptions, or recurring payments for the user to keep track of, and finally we had a singleton pattern to handle all pulling and pushing information to and from our MongoDB database. We primarily used java for our backend and used JSP, and java servlets for our front end. Unfortunately we could not connect the front end with our backend so therefore we will show in the video how they would have corresponded with each other.

Final Class Diagram and Comparison Statement:

Project 4 class Diagram



Final Class Diagram



- Changes of the diagrams:
- **Users (inheritance)**
    - In the first diagram from project 4 we originally had private attributes of Name, User Id and other information about the user, we instead turned that into a String of information, in JSON format, because it was easier to query up the string to pull information from the database.
    - We added more methods for the customer where they were able to display their transactions, worked with the notification center to receive certain notifications like overdrafting
- **Subscriptions (decorator)**
    - In the first diagram we had the classes like Entertainment inherit the Subscriptions, however it was supposed to decorate which it did not show correctly.That was modified in the 5th project to display correctly where we added a Basic Subscription to inherit from Subscriptions, and then classes like Entertainment, Bills, Food, decorated off of the Basic Subscription class.
- **DataHandler (Singleton)**
    - We added more methods for the datahandler and created only one instance of the connection to the mongoDB database. We had methods to handle pulling and pushing information to and from the database. As well we had the deconstructor to end the connection to the database.
- **Notifications (Observer Pattern)**
    - We had the observer pattern of the notification class observe the customer to let them know upcoming events, such as overdrafting or spending large amounts of

money. We had to change the methods to adjust to the new methods in the Customer Class.

Third-Party code vs Original Code:

Sources used for the front end:

- HTML/CSS
  - https://www.w3schools.com/howto/default.asp
    - This website was used to help create visuals such as a hover dropdown, click dropdown, checkboxes, navigation bar, and a login page
- Python
  - https://stackoverflow.com/questions/47372409/how-to-create-chart-or-line-graph-in-python-with-strings-in-csv
    - This website was used in order to convert the indexes of the .csv files in order to be able to use the date in a graph
- Java servlets
  - https://www.tutorialspoint.com/servlets/index.htm
    - For doPost and doGet requests and responses
  - https://www.youtube.com/watch?v=h_qQOVDTxo8
    - To install the tomcat servers on macOS
  - https://www.youtube.com/watch?v=1sfNSpAY9SU
    - A walking tutorial to set up the Maven project to work with a tomcat server and utilize post and get requests
- MongoDB
  - https://www.tutorialspoint.com/mongodb/mongodb_java.htm
    - Able to push and pull from the DB using Java

- Observer Implementation
  - https://docs.oracle.com/javase/7/docs/api/java/beans/PropertyChangeSupport.html
  - Used to implement the observer patter for the Notification Center and Customer

Statement of OOAD process for overall Semester Project

1. One of the negatives that our team experienced was being able to use Java servlets properly. We ran into issues such as the different page interactions weren't working like the way we intended it to, or we weren't able to get the servers to start at all. Additionally, there were the complications of working on different OS's as well. The java servlets seemed to work with Linux better than it did on Windows creating an issue when we tried to either debug or make changes.

2. Another negative was not picking an IDE that was suited for creating a Java web application. We initially decided to develop this project through VS Code, but we came to find out that it wasn't as well suited for this type of application as opposed to using Eclipse which was meant for Java.
3. A positive aspect of this project was that although we didn't achieve a finished product we were able to really see how these object oriented principles tied into real life production applications. Reflecting on previous projects, our old code could have been tremendously improved and cleaner with the use of these patterns. Lastly, planning out the roadmap for the project with the UML diagrams really did make the process of coding smoother as we didn't blindly guess what work needed to be done and ultimately we just had trouble figuring out technology details rather than the general code base.

Code Submission

Comment the code and explain the patterns used