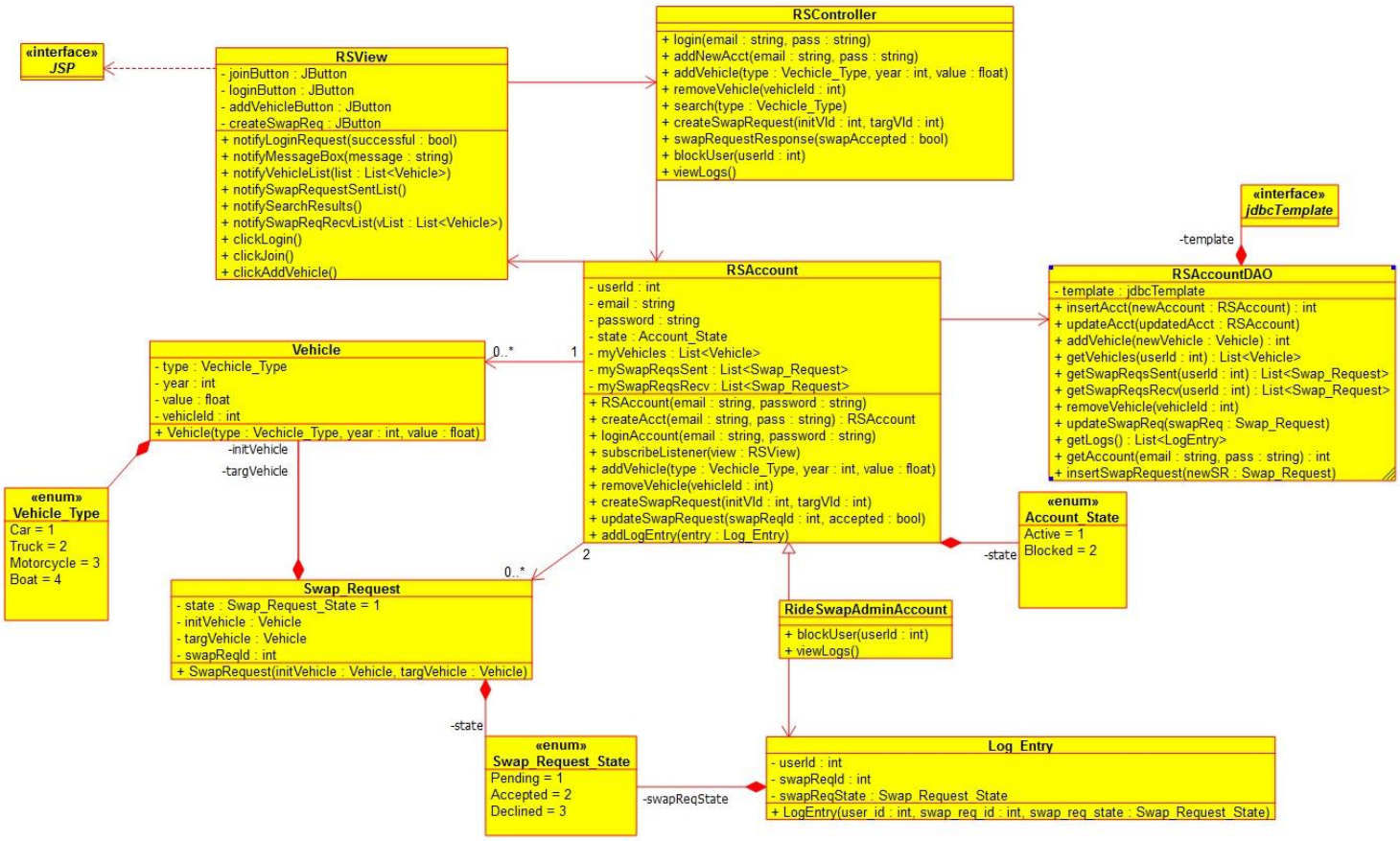
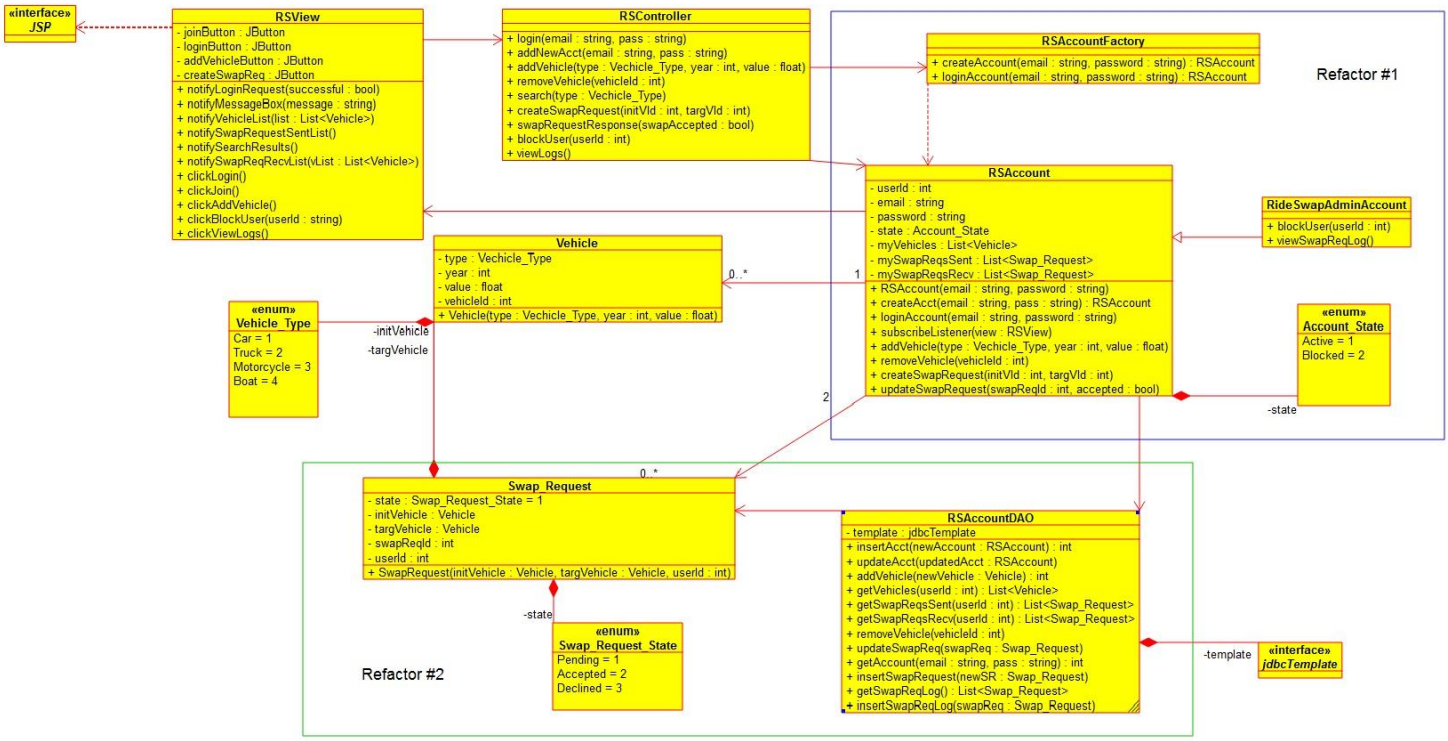
Project: **Swap Your Ride: Part 3 - Refactoring**

Members**: Hanna Loboda, Matt Weber**

ORIGINAL CLASS DIAGRAM:



REFACTORED CLASS DIAGRAM:



Refactoring #1: Added Factory Pattern for Account classes

After some thorough evaluation of the class diagram, we have decided to use the factory design pattern around the RSAccount and RSAdminAccount classes. Previously the RSController class had to query the database and, based on the query result, create one of the two class objects. Using Factory design eliminates the overhead with extra objects and makes the code more scalable when the number of objects in the database grows.

Refactor #2: Eliminate redundant LogEntry class

LogEntry class contained redundant information with the Swap_Request class, the only difference was that LogEntry included a userId attribute. By porting the userId attribute back to Swap_Request, we can then send the Swap_Request object to the log table of the database whenever the state of a Swap_Request object changes. We then modified the log methods, namely viewLogs() in the RSController and clickViewLogs() in the RSView methods enabling admin to view the logs from his account. The information stored in these logs will have the same information that was previously in the LogEntry class that we deprecated.

Other Refactoring Changes:

- RSView Class – Added admin button functionality with clickBlockUser() and clickViewLogs() methods
- Eliminated overheads with viewLogs() signature – the new method does not need any parameters since it is working off the database object, whereas previously we would have to pass all the parameters that we wanted to keep track of in the logs
- Fixed duplicate routines in the code