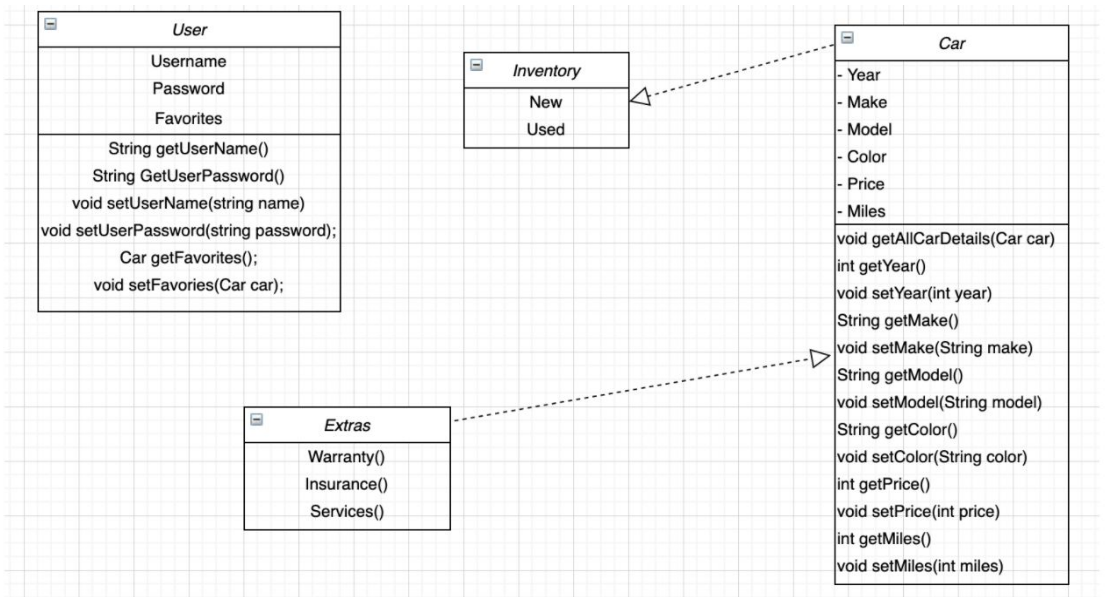
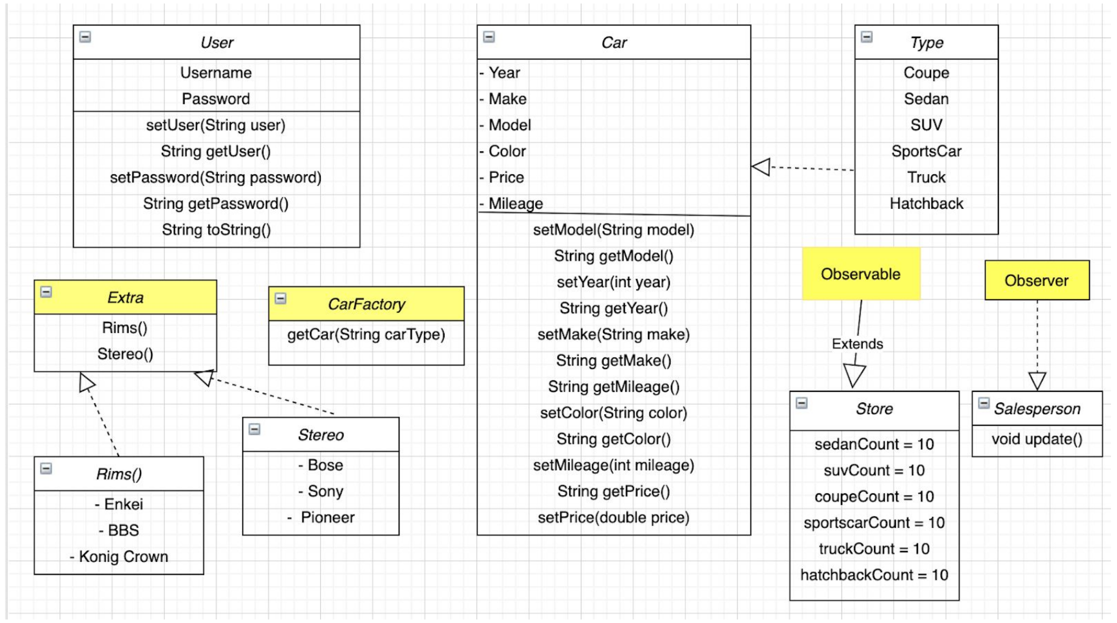
Project Six

- Project name: Search Engine
- Team Member names: David Tsing, Lina Larson, Brett Schneider
- Final State: For the final project we got rid of the favorites section and inplace changed the search function to work primarily with dropdown bars used to sort through the cars. We changed variables such as the warranties to now be add on options for the cars and added the Observer pattern to keep track of our inventory.
- Final diagrams:
  - Diagram from project 4:



  - Current diagram:



For our final diagram, we’ve changed many things since the diagram in project 4. We implemented more design patterns rather than just one. The highlighted boxes represent the design patterns used. For the “Extra” box, we used the decorator pattern, “CarFactory” box was the factory pattern mixed with the adapter pattern, and the last two boxes represent the observer pattern. Some functions we planned to use were deleted since project 4, like the favorites function. In the “Extra” category, we made that feature more specific to the car, rather than the service for the car. The name “inventory” was changed to “Store” to keep track of our inventory, and we’ve also initialized our inventory. We added a “type” of car that inherited all of the car methods.

- Third party code:
  - Springboot - https://spring.io
  - Maven - https://maven.apache.org
  - Postgres - https://www.postgresql.org
  - Heroku - https://dashboard.heroku.com/apps
  - Lecture notes, and Design First book
- Statements:
- Design patterns used:
  - Decorator method:
    - Used this method to add extra features for the car such as rims, and stereo systems.
    - The positive thing about this pattern is that it was pretty straightforward to implement, and we had no issues getting around to it.
  - Factory method:
    - We used this method to create our cars. We have 6 types of cars, and each of them have 6 attributes.
    - The positive thing about this pattern was that this approach was efficient in car creation.
    - The negative thing about this pattern was that whenever we made a change to the car types, we had to change all of the types.
  - Observer method
    - Keep track of inventory, and notify the salesperson if inventory runs out.
    - The negative aspect of this pattern's use was the implementation. Sometimes the inventory doesn’t update when we create a car object of the type.
- Outside of the design patterns we encountered issues with connecting the database. We also faced issues with certain parts of the code not working the same once we deployed it to Heroku.