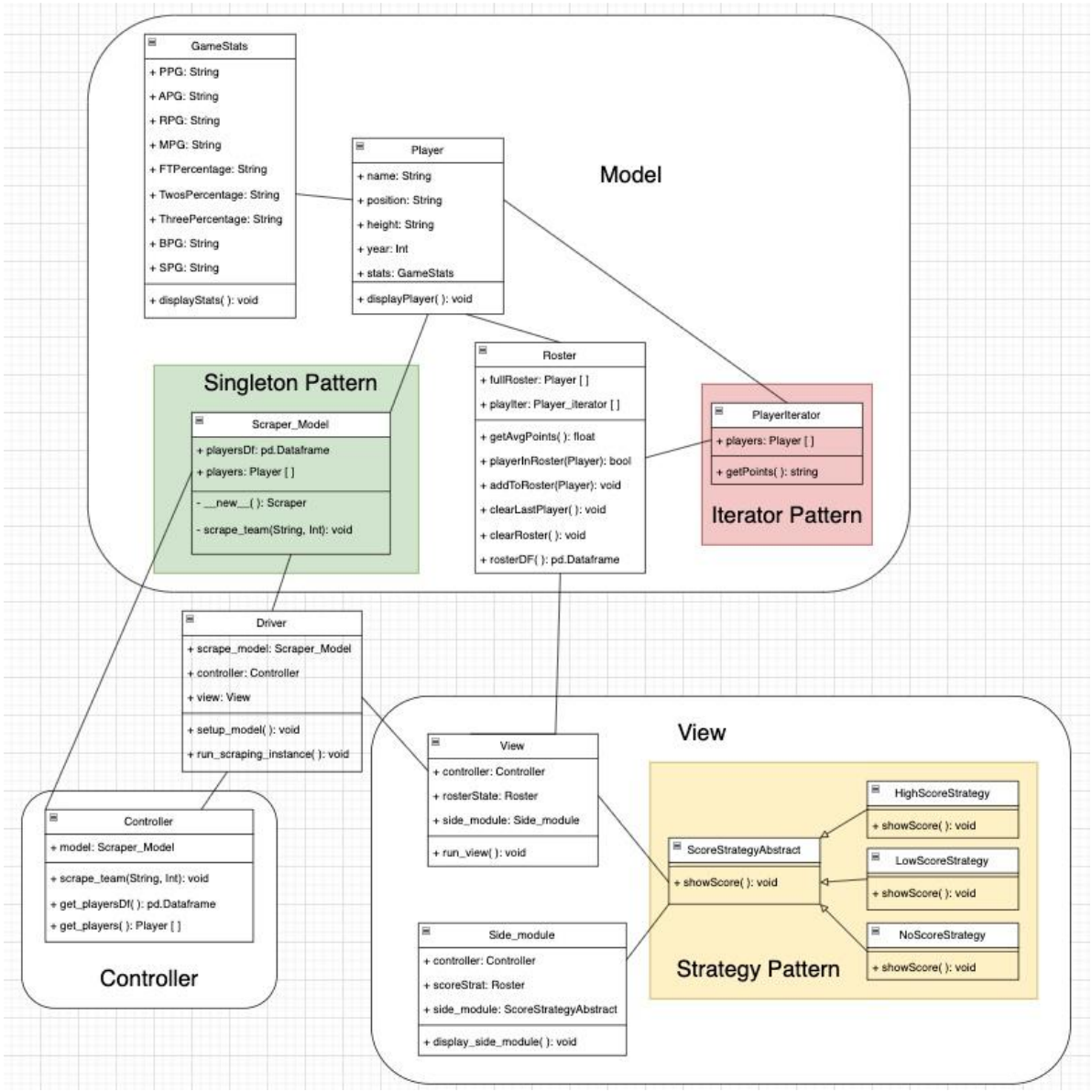
Final Project Report

- Hoop Scraping
    - Joseph Nam
    - Derek Lu
- Final State of System Statement
    - Features implemented:
        - Get data from basketball teams from certain years (from https://www.basketball-reference.com/) and display it in a dataframe.
        - Manage a team roster by adding or deleting roster members
        - Rate the strength of the team.
    - Features not implemented:
        - Starting 5 from the roster
            - This feature was not implemented due to the implementation details of the Streamlit library, the Python library we use to build our GUI. for every change you do on the GUI, it basically reruns the program from scratch. It is somewhat cumbersome to have persistent data for every action done on the GUI, so we decided to only have one instance where this occurs, in the overall roster implementation. The starting 5 feature was a “nice to have,” but not a “need to have.”
        - Color change of UI based off of team rating.
            - This feature was not implemented, as it is somewhat cumbersome to change UI elements already implemented by streamlit. Instead, we implemented a status message that tells you how your team is doing.
        - Changes from Project 5:
            - Was originally conceived as an app that helps with organizing data scraped from class search, until we realised that CU Class Search is not easy to scrape. We instead switched to scraping a basketball website, as scraping it was more straightforward. The overall features were still transferable, instead of picking classes you pick players. Instead of get professor ratings, you are getting player stats.
            - MVC instead of Observer
        - Changes from Project 6:
            - Strategy instead of Decorator
- Final Class Diagram and Comparison Statement
    - Diagrams and comparisons on next pages
- Third-Party code vs. Original code Statement
    - Many of the object patterns were based on the Python examples given in Bruce’s lectures.
        - MVC, Iterator, Strategy, Singleton
    - Referenced a lot of documentation for streamlit for those view elements.
        - https://docs.streamlit.io/

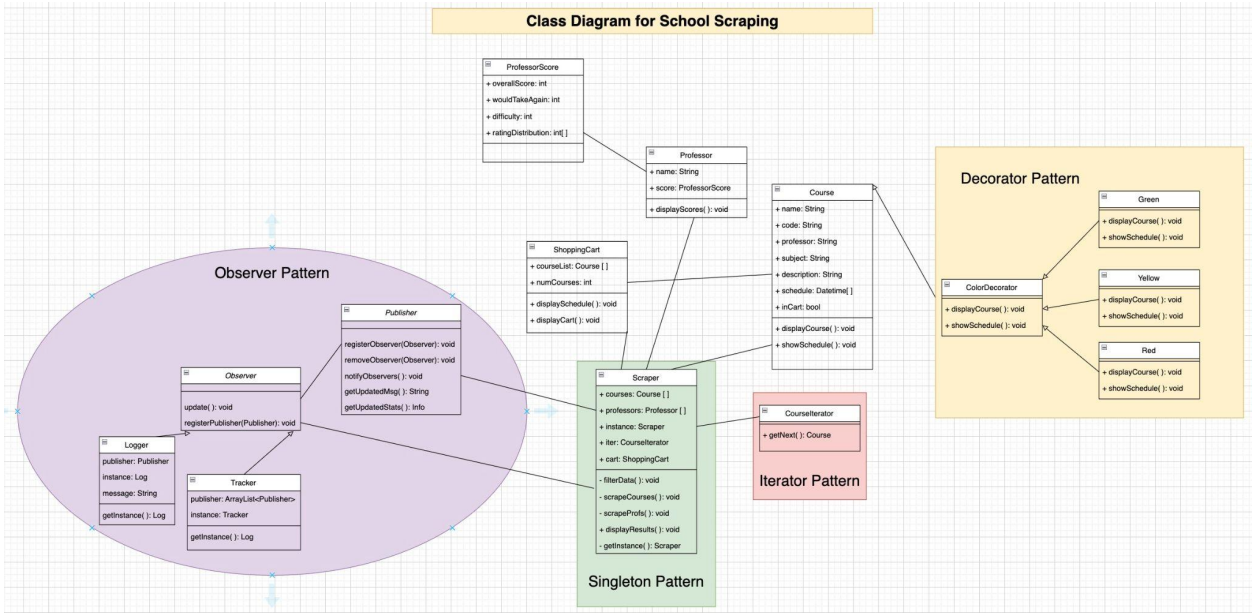
- Statement on the OOAD process for your overall Semester Project
    - One negative issue we ran into when coming up with the project idea was that it felt like we were unnaturally forcing certain design patterns into our project. Because we needed four different patterns, it was difficult to come up with a project idea that seamlessly fit all 4 design patterns and was within our scope.
    - One positive experience we had in the design process was creating the class diagram to organize our thoughts. We think using the class diagram was crucial for us to get on the same page.
    - A key design piece process element we included was using an MVC model when neither of us had used one before. It fit well within the design of our project and made it easier to understand how our code was structured.

**Final Class Diagram**




*More clear jpeg version in the Github repo

## Project 5 Diagram




*More clear jpeg version in the Github repo

## Changes between Project 5 Diagram and Final Report Diagram

From Project 5 to now, our diagram has changed dramatically. We kept the same core idea of building a website that scrapes data but the subject we're scraping changed from CU school courses to basketball teams. This will make all the variable and class names different (eg. "Players" instead of "Course", "Team" instead of "Professor"). We now also have two different design patterns being used from Project 5 using the MVC and Strategy patterns instead of the Observer and Decorator patterns. Using the MVC model actually structured our entire code differently than we originally planned for in Project 5. Our scraper no longer needed a publisher or observer and communication was done through the MVC model. In our original, we didn't have a smooth way in visually displaying our website with a GUI but now in the final class diagram, that is taken care of in the View object.