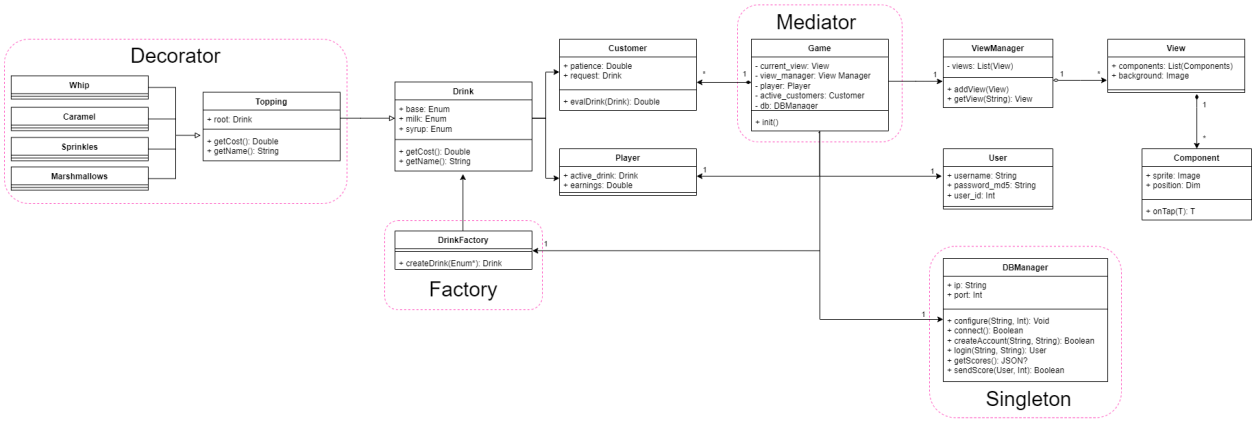
Project 7 Writeup

1. Project Name: LavaJava
Team Members: Jeffery Mitchell & Kaitlyn Huynh

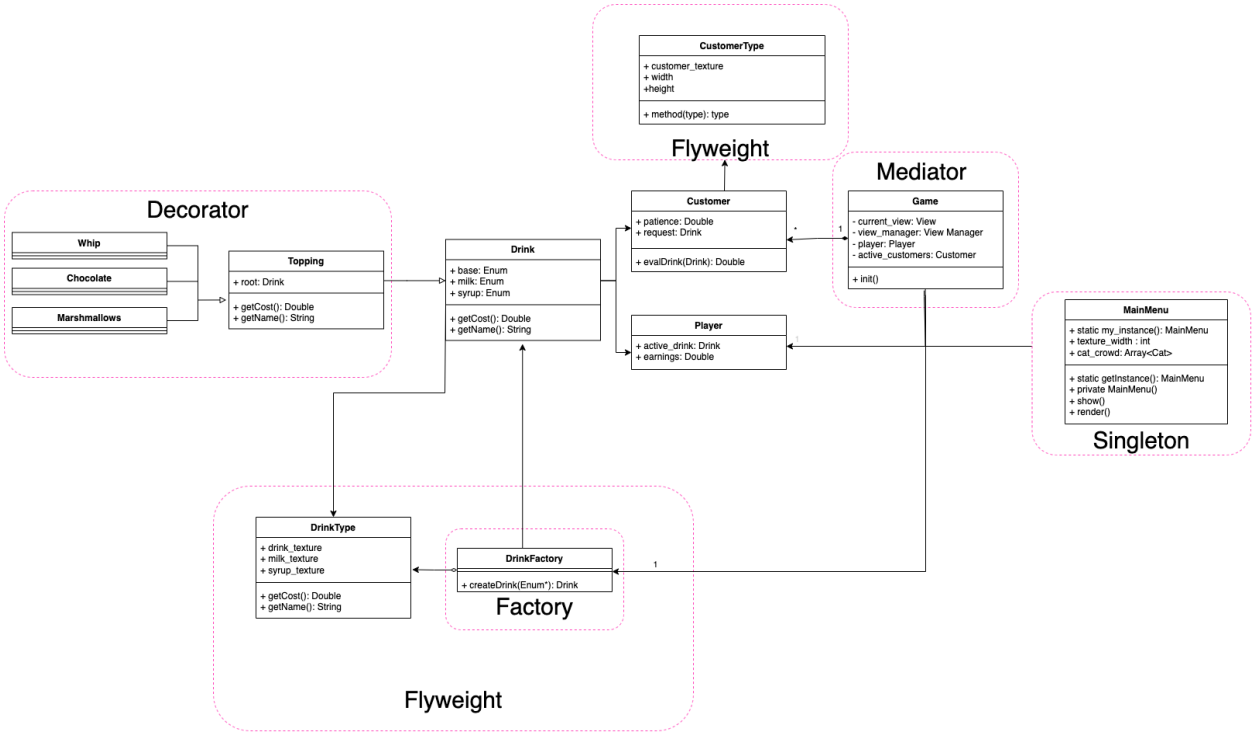
2. Final State of System:
The features that we ended up implementing in the final version of LavaJava include customers coming in and requesting random drinks, users being able to construct those drinks (choose a caffeine, a dairy, a syrup and add toppings to the drink), a timer to build those drinks and a point system that awards players points depending on how fast they can assemble the drink in the timer. Some things that have changed from Project 5 to Project 6 are scrapping the transition of the scene. After play testing, we realized that we could actually just utilize the base Cafe scene that is already used to create the drink. Another thing that changed from Project 5 to Project 6 is the login/logout functionality. Due to time constraints and prioritizing certain features, we wanted to focus on making sure that the core features of the game (creating drinks) were functioning and with the learning curve of LibGDX and unfamiliarity with it, the time we had left was limited and we wanted to make sure that the actual gameplay worked. Despite having to scrap the login/logout feature, the code is fully finished but we weren't able to implement it in the actual game.

3. Final Class Diagram and Comparison Statement:

Project 5 UML Diagram



Project 7 UML Diagram [(Link for better viewing)]



As can be seen from the UML diagram, our initial system and our final system underwent some heavy changes. Initially, we had our DBManager class (handling login/logout and updating high scores) that would be our Singleton & Proxy

pattern. However, we ended up having to scrap that and ended up having our MainMenu class be the Singleton pattern instead. We initially planned on including MVC in our project as well, but that ended up being scrapped as well. We also ended up adding the Flyweight pattern to our system. While developing our application, we realized that the way we're handling the graphics line up with how the Flyweight pattern works. As can be seen in the UML diagram above, our Customer and Drinks class ended up being the ones who used the Flyweight pattern. Classes/Patterns that were consistent from Project 5 to Project 7 were Decorator (Toppings class), Factory (DrinkFactory class) and Mediator (Game class). The ones that ended up changing were Singleton (now in MainMenu instead of DBManager) and the addition of Flyweight (Drinks & Customers).

4. Third Party Code vs Original Code Statement:

When developing our application, we had no prior knowledge of using LibGDX, therefore we ended up using resources to learn how to incorporate LibGDX with our application. Resources that we used are listed below:
LibGDX Documentation:
https://libgdx.com/wiki/articles/java-development-kit-selection
LibGDX Video Tutorial:
https://www.youtube.com/watch?v=rzBVTPaUUDg&list=PLZm85UZQLd2TPXpUJfDEdWTSgszionbJy
Pixel Art in Procreate Tutorial:
https://www.youtube.com/shorts/Jf-YYPur5q0
Pixel Cat Art:
https://www.pngkit.com/view/u2r5w7a9q8u2r5r5_colors-download-settings-cat-pixel-art/

The code and the rest of the pixel assets were created by our team.

5. Statement on OOAD

1. **Trouble with LibGDX**
    a. LibGDX was a framework that we struggled to adapt to since it primarily used Java 8. Since we were unfamiliar with Java 8 and Android Development and using Intellij (we primarily used VSCode for the majority of the semester), it was definitely challenging to create a project in these unfamiliar frameworks/environments let alone learn how to use these environments in a short amount of time.

2. Sizing of Assets
    a. Due to how unfamiliar we were with how to create assets for a game, a lot of trouble popped up when it came to actually putting the assets in the application. When initially creating the artwork for this game, we made all assets different sizes which was shown to be problematic since a lot of offsetting and hardcoding positions for different assets took up a majority of the code. We ended up having to go through many iterations of the same artwork since larger sizes of the artwork would take far too long to load in and some of the artwork wouldn't place well on the scene. If we had more information on how to actually create assets for certain development environments, that would've been a great help but since this is our first time creating an Android game, assets ended up going through multiple iterations to get right.
3. Difficulty of coordinating remote
    a. Since this class is remote and our team members are commuters, setting up meetings proved to be difficult. Also factoring in that all team members are in project heavy courses, setting up weekly meetings were often very tight and would have to be fast since we also had other matters to attend to. Luckily, since we were on campus at the same time for around 2 days a week, it was easy to catch up after class and work on the project. However, due to the remote working environment, we would often get Git Merge errors which would result in multiple hours of frustration just trying to get our branches up to the same version and making sure we had the most up to date code.