# csprojectUML

One author and dev - Sebastian Medina

https://youtu.be/vBN2mzvUMZI

1.
Let's just make a basic microtransaction store front design, with a premium currency, freemium currency, a free starter reward, and some sort of support request which will be a dummy function for now. I don't have a server to invest in.

3.
There are 2 main use cases, namely purchases, and support requests.

Microtransactions, while important, are an entire use-case situation on their own,
based on interactions with various financial institutions and users, and dealing with financial information is complicated enough to not be included for the sake of brevity.

When a purchase is made, there are several cases to be considered. This involves whether the user wants to purchase with coins, gems, or the purchase doesn't have anything for the users need to choose either currency. Then, it will get run by a program which should just deduct the amount for the purchase within the client, since we have no server implementation.

Varying preconditions are the amount of coins, gems, and whether or not their free reward was claimed.

Postconditions such as the success of the purchase, whether to change the currency value, or whether to change the purchase condition on the storefront, are all expected to be rebound through the system.

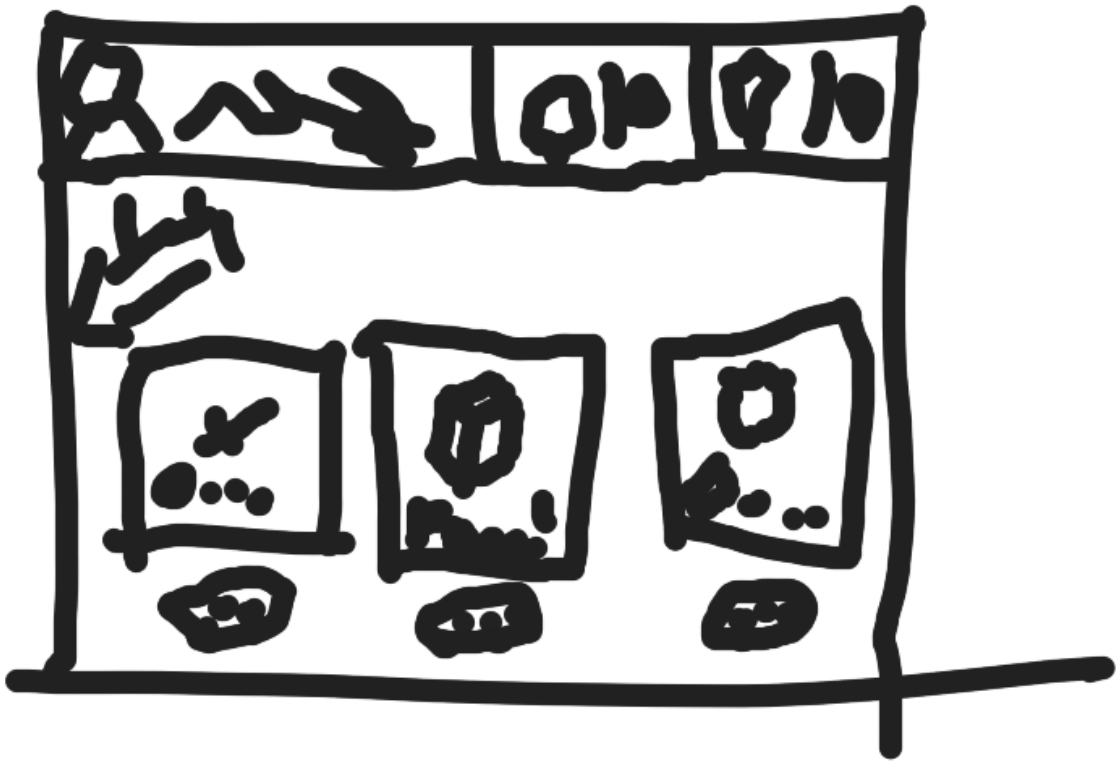
Scenarios A and B are normal currency purchases, coins or gems, that go through and succeed, reducing their respective currency and we tally the purchase in some test counter on the side.

Scenario C is a free purchase, where the user needs to only interact with it for the first time, and claim their purchase, which will get tallied and should disable the option permanently.

Scenario D is a failed purchase, where the user attempts to make any purchase from A through C, but either the amount or flags for the purchase do not line up with the requirements for the purchase to pass, therefore making it invalid, and denying the user with a warning.

Scenario E is an odd one out, just being our dummy support ticket to allow a user to attempt to ask for help with any services or issues in general. Nothing will come of the ticket, for obvious reasons, but it helps to implement for a demonstration.

With no familiarity with the various options in Java GUI libraries or design, we'll just pick JavaFX and see if we can make it work. It already seems to have buttons, text, basic shapes, enough to make what we need for a "developer's" scene.



Post-dev: I really dislike how Java's application setup works, but I managed to make something usable. I would definitely have had a better time in other setups, especially since various version mismatches or simply just bizarre changes thanks to Oracle made it unnecessarily difficult to even set up, but the testing setup works.

There's a lot more I would care to add and polish, but it feels really unnecessary when half of the project that is being graded is the UML. Feels like I more so care about trying to apply the UML and how it works to my efforts rather than anything complicated application wise, especially as a solo dev with a couple other classes.

The UI is not too intuitive, but it works. You can interact with the buttons, and exit whenever. A fake support button where you're guided to enter in, then submit, and purchase buttons that with the allotted currency should do for demonstrating a "working" storefront.

I won't lie, while I would've preferred to have a "manager" of sorts to assist and organize in the purchase system and allow for more dynamic options, it feels completely out of scope for this.

This will *not* be made ready for deployment, it's not applicable to something meant to be used for testing. Since setting up the JavaFX required finicky VM options, here is the command if needed to test:

```
--module-path
"C:\Users\Fux\Documents\hw7\csprojectUML\demo\stress\javafx-sdk-21.0.1\lib"
--add-modules=javafx.base,javafx.controls,javafx.graphics,javafx.media,javafx.fxml
```

Outside of that, there is nothing particularly of note. Oddly enough, I don't hear much about UML at all from peers in the industry.