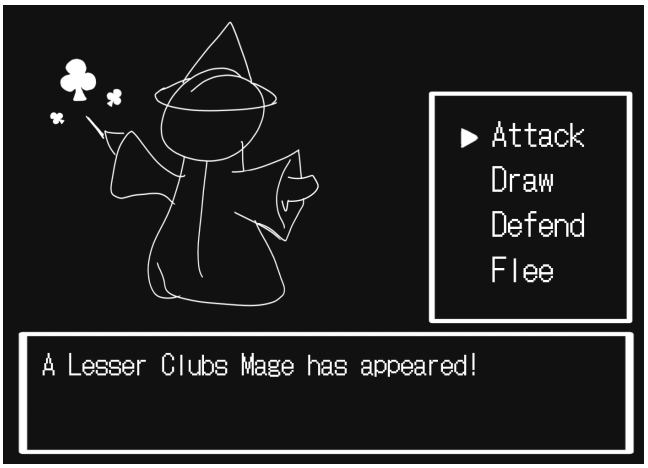
First is the TurnSelection view, where the player chooses the action they want to take for this turn (Attack/Draw/Defend/anything else we implement).

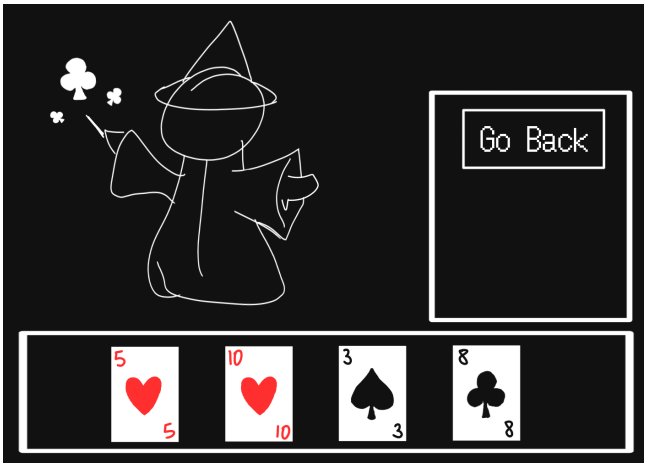
- For the sake of making things easier to code, we can just make the game entirely mouse-based. So each action could correspond to a button that the player can press, rather than moving a cursor up and down with the WASD keys
    - The mock up image still shows a cursor though because I didn't feel like making the options look like buttons lmao



Pressing any of the buttons leads into their own Use Case. Attack is the most complicated, with multiple views, so I'll talk about that one first.

First, the Use Case just calls and returns all the cards the player has in their inventory, so they can pick what card they want to attack with. For now, I'll call this the "AttackSelect" view.

- Each of the cards *look* like images, but actually function as buttons (I know this is doable since I've seen it in those building-block type of "coding" websites, we'd just need to figure out how to do it in Java)
- I've also included a "Go Back" button, for if the player decides that they'd rather do a different action for this turn (e.g., they don't have any good cards, so they've changed their mind and decided that they want to Draw from the deck this turn).
    - This button would just take them back to the TurnSelection view with no values changed



If the player *doesn't* pick Go Back, then they click on the card they want to attack with. The card's values are sent out into the program, and somewhere in the Clean Architecture parts, the enemy is called to perform its random turn.

In the Output Data(?), the player's and the enemy's respective actions are analysed.

- If anyone chose to Attack, then damage is dealt to their opponent (i.e., damage is dealt to the player if the enemy attacked, and vice versa).
- If *both* the player and enemy Attacked, then the first thing that happens is the suits of their cards are compared to see if one suit has an elemental effectiveness against the other. A damage multiplier is added to the Character that used the effective card if so, and *then* damage is dealt to the player and the enemy

While all this is happening, we also use the API to remove the played card from the player's hand and put it into the discard pile. (And the same is done for the enemy, if it also attacked).

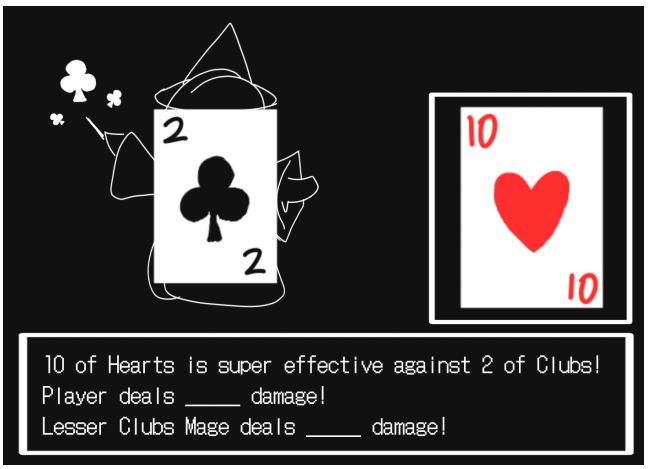
This then gets presented as what I'll call the "TurnResults" view:

- I don't remember if Hearts beating Clubs was actually what we agreed upon, but I wanted to show that text so just go with it for now
- The "____" would be where an actual damage value goes. Since the player is the one who used the 10 of Hearts, they would get a bonus Super Effective damage multiplier
- This shows an example where both the player and the enemy Attacked, and where one card had an elemental effectiveness against the other card.
    - If only one Character attacked or the two cards are neutral against each other, the "super effective" line does not show.
    - If the player does not attack, the "Player deals ___ damage" line does not show, and that box in the corner is empty.
    - If the enemy does not attack, the "[Enemy Name] deals ___ damage" line does not show, and there is no card on top of the enemy's image
    - If a Character's action is not attack, whatever action they *did* take is stated in the text box. For example:
        - "[Enemy Name] drew a card!"
        - "Player defended!"

      It would probably be written in the very top line of the box, before the damage is mentioned. e.g.:
        - "Lesser Clubs Mage defended!"
          "Player deals ___ damage!"

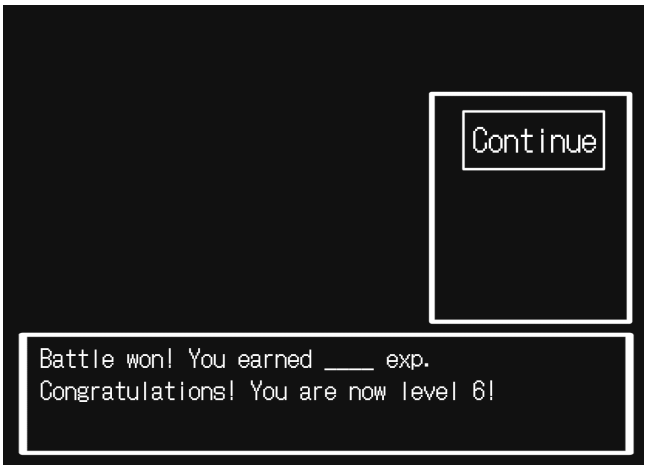
      The hierarchy for the lines would go (from highest priority to lowest):
        - Super Effective Damage mention
        - Non-Attacking Player Action
        - Non-Attacking Enemy Action
        - Player deals damage
        - Enemy deals damage

      Higher priority lines show up first in the text box, and only two or three lines are shown at one time.
- There's probably also a button somewhere for the player to hit "continue", but I don't know where that would be



If the enemy has been defeated, then the player is awarded exp and we proceed to the "BattleWon" view.

- If the player has earned enough exp to level up, there is an additional line telling them that, as shown here.
- If the player has levelled up and reached a "milestone" level where they also unlock something (e.g., more inventory space), that is mentioned in a new line below the level up text.



If the player has been defeated, then it's Game Over, and we move to the "BattleLost" view or "GameOver" view or something (or maybe a BattleLost view that then moves to a GameOver screen/view?). I don't have a mock up image for that though.

Finally, if both Characters are still alive, then we go back to the TurnSelection view with the updated properties and start a new turn.

Of course, that's just for the battle section. The explore section would have its own separate set of views, but I think it would be much less complicated.

I don't think the presenters for the TurnSelection use cases would have a "prepareSuccessView" and a "prepareFailView", like the week5ca program? Instead they'd have:

- "prepareNextTurnView", for when both the enemy and the player are still alive (so it goes back to the TurnSelection view, now with all the updated values (health, inventory, etc) where the player can once again choose their turn.
- "prepareEndBattleView" for when either the enemy or the player is defeated and the battle is done.
    - Maybe this would be split up into "prepareWinBattleView" and "prepareLoseBattleView"?