Feature Demo

- show any data files that have been created and describe how they are used within your program (focus on the file *content* rather than the code that uses it)

CardGame Files

1. By directory (in XML) - independent components assemble code
    a. Deck - specifies card value, suit, quantity
        i. Cs308 deck
    b. Game - game logic, competition (group vs. adversary - design decisions)
        i. Poker xml file
        ii. General card game abstracted to (alternating) sequence
            1. EntryBet, dealer series of actions, alternate player actions (rounds)
    c. Hands - added new types, currently simplified
        i. pokerHands
    d. Players - little changed
        i. 3 players
    e. View - corresponds to game
        i. Languages, Stylesheets, Errors, Dimensions
2. LobbyView - Dimensions of Lobby View, Error Stylesheets, Bundles (CSS, Language

- run the program from the master branch through a ***planned*** series of steps/interactions (it will look *much* better if this demo shows everyone's parts working together as a single program rather than multiple separate demos)

Lobby Demo

VERSION_1

1. Custom game - Show Error Handling
    a. Omit a PlayersFile to view Errors, then pick a correct one
2. Construct a Custom Game
    a. Create Blackjack
        i. Hands.xml, blackjackGame_v2, Standard, view.xml, 3 players
        ii. ,Stay 3x to show a splash screen.

VERSION_V2

3. Show new lobbyview_XML file
    a. Swap v1→ v2 in Main.java
        i. New starting dimensions, new game bundle, new default stylesheet
    b. New icon (new bundle), preset game

GAME

1. PokerXBlackjack
    a. Blackjack against the table
        i. New silly deck
        ii. Dealer action like blackjack
        iii. Evaluated on blackjack hands
2. Toggling language and stylesheet in game

- show a variety of JUnit or TestFX tests (including both **happy and sad paths**) and discuss how writing tests helped or affected the way the code was written

1. UI - TestFX
    a. LobbyView FailCustomGame()
    b. LobbyView runBlackJack()
    c. GameView Good/Bad wagerInput()
2. XML Reader - LobbyViewTest
    a. No printouts/debug mode needed
    b. All assertions
    c. Data driven nature, ease of extension (e.g. stylesheet/bundle check)

- if asked, everyone should be able to describe how their work relates to specific demoed features as they are shown

1. UI Features
    a. LobbyView Icons -> GameView generation via Bundle + FileChooser
    b. Exceptions are handled via an ExceptionDisplayer
    c. Languages and CSS are toggled via a Controller call

2. Abstracted controller
    a. Group game profile
    b. Design decisions below
3. XML Parsing (effective targeting by doc and tags, interfaces are good)

Overall Team

- describe how much of the planned features were completed this Sprint and what helped or impeded progress
    - UI Side:
        - All new features were added, but many nuances need to be fixed
            - LobbyView CSS + Language
            - Multi-Language Exceptions
            - Ready Button Nested Loop
            - General CSS Styling could be improved
    - Backend
        - Goal 1- Poker
            - Strategy profile → reflect on controller
            - Error handle player actions (lambdas)
            - Actions for new games (reflection)
        - Goal 2 - General data driven
            - Segmented files (5 independent)
            - Robust parsing (easy new deck)
- describe a specific significant event that occurred this Sprint and what was learned from it

Significant Event (max)

- Refactoring Controller by game
- Abstract concepts that vary
    - Entry bet
        - Blackjack = wager, poker = ante/blinds, roulette = bet on event
    - ACTIONS
        - Lambdas in various actions (group needs ability to modify rest of table flags)
    - *Evaluation*
        - *Add communal cards or not*
        - *Number of cards in hand*
    - *Payoffs*
        - *Pot, vs. one one one vs. dealer*

- describe what worked, what did not, and something specific that is planned for *improvement* next Sprint
    - Worked:
        - Contract by Design (by calling interface methods)
            - Explicit interfaces (GameView interface)
        - XML Reading went smoothly
            - LobbyReaderTest
        - Data-driven nature being extended to the UI + Game Generation
        - Functional Interface + Lambdas
            - GameCaller()

            - Enabled MV separation
                - group.BetAction (all three lambdas act on view/table)
        - Exception Handling improved (was not a *major* focus of our sprint)
            - ActionException (many constructors)
    - Did not work:
        - API Changes (new features came up)
        - “Mutable” UI now gets rerendered
        - Creating a smoother *.showAndWait()* functionality
        - Did not get to all of GameView nuances (Such as color setting for losing players)
        - Did not get to extend full data-drive to exceptions or LobbyView, but the shell is in place.
        - UI Full Composition means passing through
    - Improvement
        - API talk — be as specific and proactive as possible
        - Use Trello More
        - More robust code

- what features are planned to be completed during the next Sprint (taking into account what was done this Sprint), who will work on each feature, and any concerns that may complicate the plan

Next Sprint

1. General refactoring!!
2. Increased robustness
    a. XML Validation, defaults
3. New features
    a. In-game editor for specific data files
    b. Different visual representation of cards/actions, if time permits.
4. View only google doc (help)
    a. Game manual
    b. Coder manual (extension)
        i. Mini analysis
        ii. New hands, simple as new object