CSCI 4448 Project 5 Writeup

Project Summary:

- **Project Title:** Sorry! Board Game
- **Group Members:** Brian Noble, Sidhant Puntambekar, Isaac Pyle
- **Project Summary:** For our group project we will be recreating the popular board game “Sorry!”. The main goal of the application is to play the game for four players (at first four human players, and then eventually for a single human player and three other computer generated players). In terms of implementation, our goal is to accurately represent the “Sorry!” board game through a JavaFX UI framework and implement several object oriented design patterns for user input (command pattern), monitoring the game state (observer pattern), and creating card objects on the fly (decorator/factory). When we finish development of the project, we expect to have a fully functional “Sorry!” board game where you can either play with other human players (we intend to implement this first) or a series of random and human players (we intend to implement this if time permits). Overall, we want to have a user interface that displays the game state to the main player and game logic that can delegate functionality to several objects representing the pawns and playing cards associated with the game. Once we have the primary game functionality, we want to extend the game by creating our own separate rules (through new card objects implemented through factory and decorator patterns), random computer-controlled players, and a potential point scoring system where each player has a selected hand of cards from the deck instead of randomly drawing from it.

Project Requirements:

Requirements
Goals of the system:

- Main player/user can play Sorry! game with 3 other real players
  - Players are able to draw cards and do the corresponding actions
  - Players can move pieces around the board based on card values and instruction
  - A winner is decided when one of the players gets all of their pieces to the ‘home’ tile of their color.
- Main player/user can view previous game results from the home screen
- Main player/user can display the rules page, as well as an about section.
- Logger is required to log all interactions that happen on the board
- Tracker is required to track game data and saves it to a separate table in the database

Responsibilities

- Card objects move the corresponding pawn on the board

- Players are able to get a card object from the deck class
- Game identifies when 4 pieces of the same color are in their respective home tile, thus ending the game and saving the results to a database.
- Previous results page connects to the results database, pulls the table and displays to the user
- Logger saves game data to text files to be used for bug fixing
- Tracker saves game data to the database, which the previous results page uses to display more extensive data to the user

Users and Tasks: Use Cases

- **Number and types of users**
  - For the initial iteration of our project, we will only have one type of user who is the main player. They will play Sorry! against three other random players who will be generated by the application.
- **Tasks users perform**
  - The tasks for the main player user to perform while playing the Sorry! board game application are to navigate to and around the home page for the game, initiate a game from the homepage, peruse the rule set and previous game history, and most importantly, play the game while interacting with other human players and eventually, random computer generated players.
- **Use Cases**

Use Case	Actor	Basic Program Path(s)	Alternative Program Path	Final Goal
Read Rules	Player	Opens the rule dialogue box	N/A	The rule dialogue box should display
About Game	Player	Opens the about game dialogue box	N/A	The about game dialogue box should display
Select Card	Player	Player clicks the “Select Card” button and a card is selected.	Deck is empty after card is picked, so deck is reshuffled	Selected card should be displayed and move pawn should begin
New Game	Player	Player begins a new game	N/A	New game state is initialized
Exit	Player	Exits the Sorry!	N/A	Application is

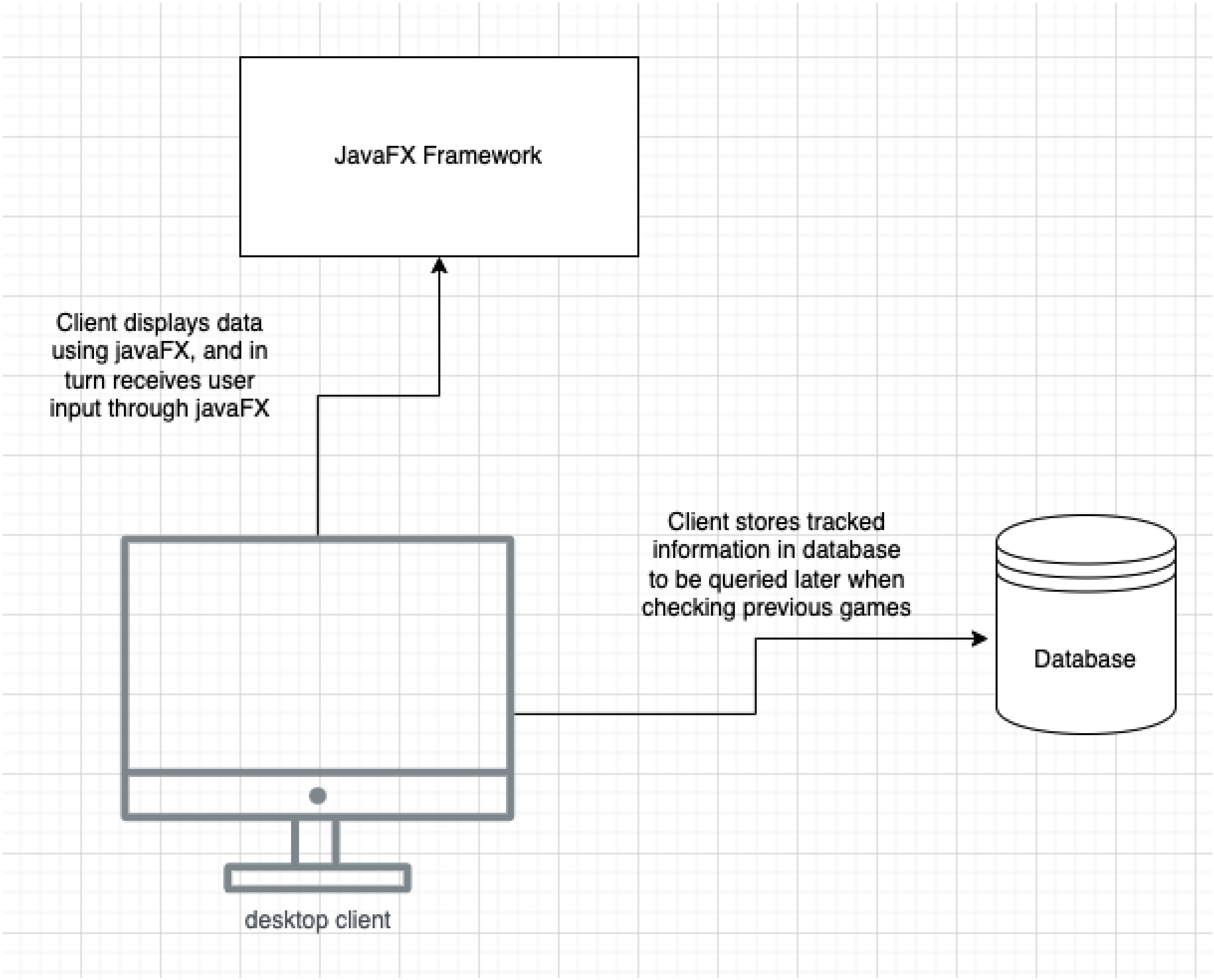
		Game application		closed
Move Pawn	Player	Moves the selected pawn by the amount displayed on the card	● Split the upcoming move between two pawns ● No pawns available for move, turn ends ● Kick enemy pawn from tile landed on by player pawn	Pawn move event will be handled

- Here is a link to an alternate case diagram: https://lucid.app/lucidchart/2c9b6271-e306-45ce-99ce-231cab0739dd/edit?invitationId=inv_4833295e-2899-4155-9ffe-2e9f7179460d

UML Activity Diagram:

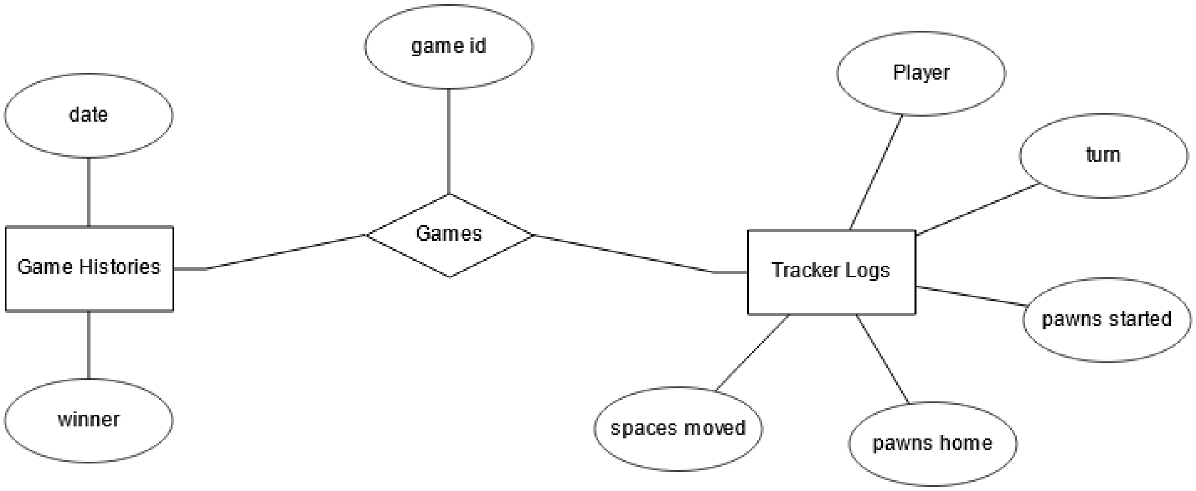
- Here is a link to the activity diagram: https://lucid.app/lucidchart/6a2974ce-fce7-43a9-9877-7a8fef9848fe/edit?invitationId=inv_09bf7755-51e0-4996-887c-b053cf6d6d6d
-

UML Architecture Diagram:


JavaFX Framework
Client displays data using javaFX, and in turn receives user input through javaFX
Client stores tracked information in database to be queried later when checking previous games
Database
desktop client

Data Storage:

- We are planning on using a MySQL database to store information related to the previous game history and the tracker observer object. After a full revolution around the board (each player in the game will play on their turn), the tracker will write relevant player data (such as the number of pawns the player has started, number of cumulative spaces moved, number of sorry cards drawn, etc.) to the main game view (handled by the game controller) and will update a database table such that in the main menu view, there will be detailed statistics about player actions after the game is over. The following diagram is an ER model of the database.



UI Mockup:

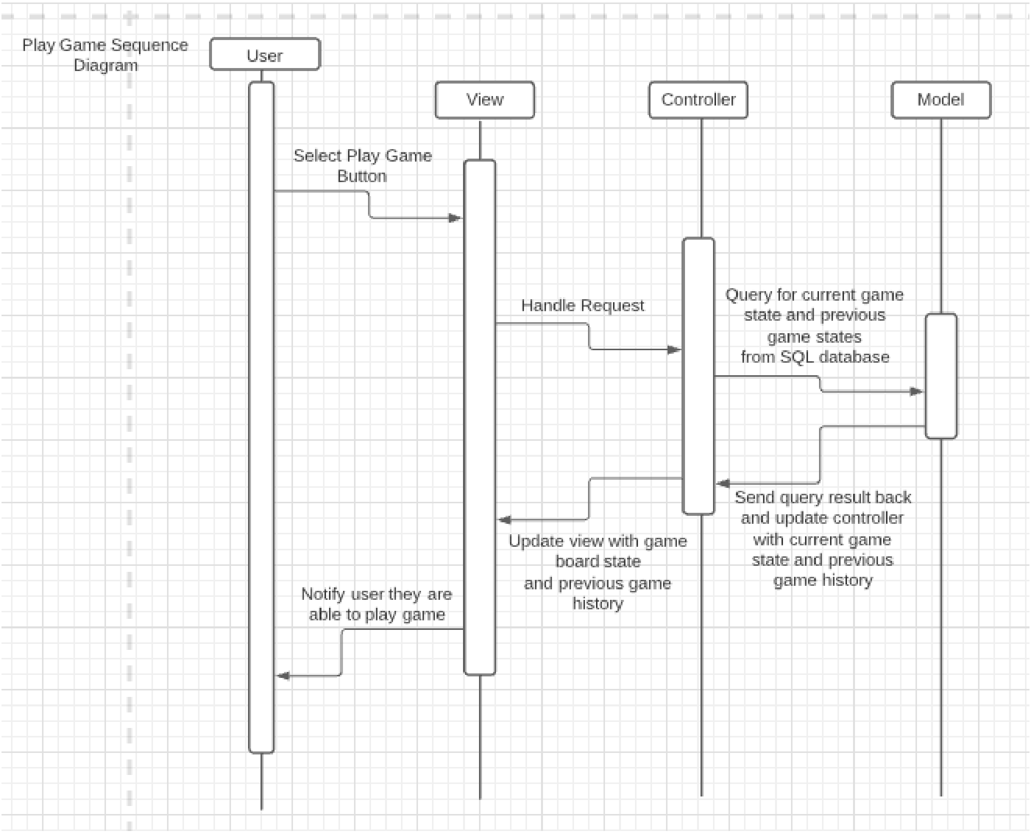
- Here is a link to the UI mockup: https://balsamiq.cloud/s5lj48j/prww5mb

UML Class Diagram:

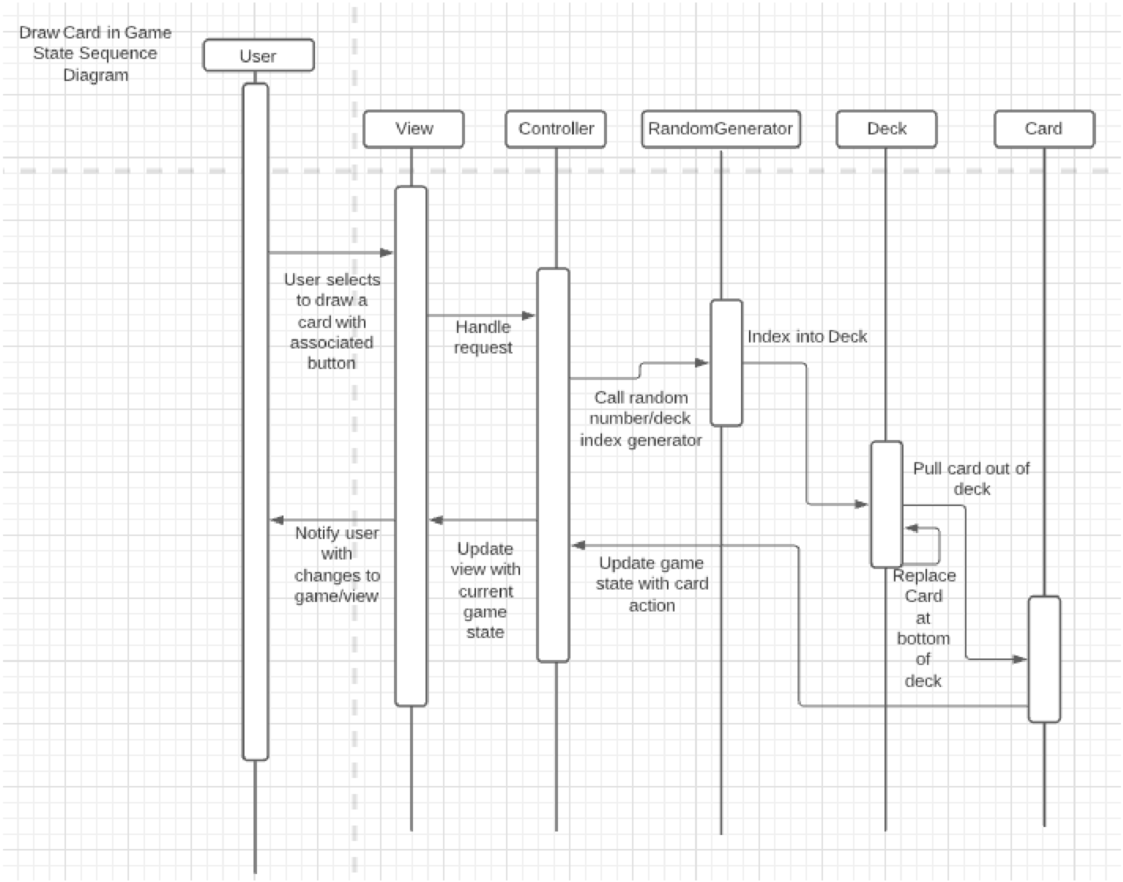
- Here is a link to the UML class diagram with design patterns highlighted in red: https://lucid.app/lucidchart/3f0f6c9b-ceba-4be0-b844-adc06bc564d6/edit?invitationId=inv_7a6a468c-e7a3-4e99-8feb-9b286ebb16c4

User Interactions / UML Sequence Diagrams:

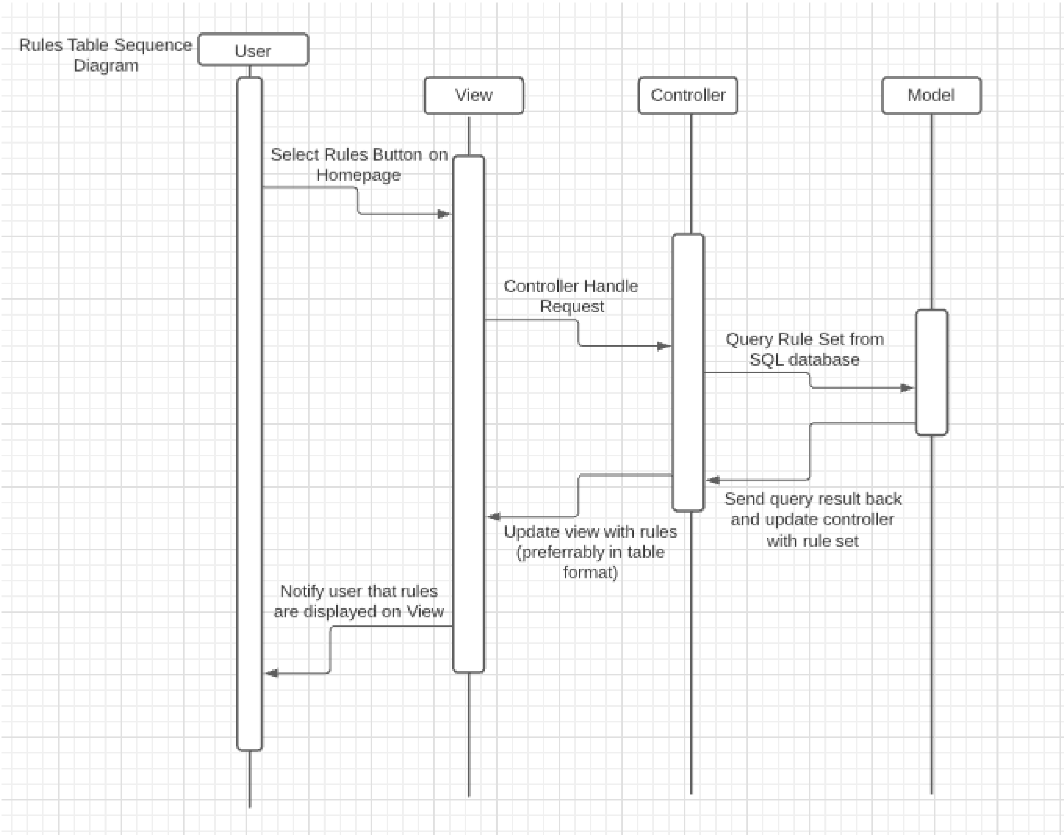
- **Possible User Interactions:**
  - **Select Play Option:** For this user interaction, the main user/player will first land on a home screen and have the option to either begin a new game, read the rules, browse previous game outcomes, and read about the game/developers. For the primary option of playing the game, the application will need to update the view through the Model-View-Controller pattern where the controller will be responsible for mediating the retrieval of previous game history from the model data store in memory. Once this data is retrieved, the controller will then notify the view to update to the game board and the main player/user will be able to play the game.


Play Game Sequence Diagram
User
View
Controller
Model
Select Play Game Button
Handle Request
Query for current game state and previous game states from SQL database
Send query result back and update controller with current game state and previous game history
Update view with game board state and previous game history
Notify user they are able to play game

- **Draw a Card:** For this user interaction, the main user/player will be in the game with three other human players (or eventually random players) and from there will need to interact with the card deck in order to move their pawns around the board. The user will simply need to press a button visible on the main game board screen to draw a random card out of the deck. Depending on the card, the user will have a choice to select which options to take if it is a special card, or will continue with the default behavior if the card is standard/generic. Following the draw of a card, the move order among players will be clockwise where the next player will follow the same process by drawing a card and playing their turn.


Draw Card in Game State Sequence Diagram
User
View
Controller
RandomGenerator
Deck
Card
User selects to draw a card with associated button
Handle request
Index into Deck
Call random number/deck index generator
Pull card out of deck
Notify user with changes to game/view
Update view with current game state
Update game state with card action
Replace Card at bottom of deck

- **Select Rules Option:** For this user interaction, the main player/user will want to read the rules of Sorry! before playing the game since they are inherently complicated. This is a similar setup to the sequence diagram for selecting the option of playing the game where the controller will be responsible for facilitating interpretation of the rule display button action from the view to retrieving the rule set from the model data store. The controller will then feed this information back to the view and the view will notify the user that the rule set is displayed.


Rules Table Sequence Diagram
User
View
Controller
Model
Select Rules Button on Homepage
Controller Handle Request
Query Rule Set from SQL database
Send query result back and update controller with rule set
Update view with rules (preferrably in table format)
Notify user that rules are displayed on View

- Here is a link to our three sequence diagrams: https://lucid.app/lucidchart/157cd543-115d-447b-8d60-817125637f36/edit?invitationId=inv_f3230154-025b-4879-9cd9-415c4328710f

References:

- https://en.wikipedia.org/wiki/Sorry!_(game)
- https://www.hasbro.com/common/instruct/sorry.pdf