CSCI 4448 Project 6 Writeup

Status Summary:

- **Project Title:** Sorry! Board Game
- **Group Members:** Brian Noble, Sidhant Puntambekar, Isaac Pyle
- **Work Completed:**
    - For our group project we are 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 the progress we have made on the application, we as a team have acquainted ourselves with the JavaFX framework as well as creating a preliminary database connection to store game data and information. For our model view controller pattern, we have created four primary views for the game which include the home view, game rules view, actual game view, and the leaderboard view. We also have created separate controllers for managing each of these views and are also in the process of building our model database for the leaderboard view. We also have a singleton observer pattern implementation with eager and lazy instantiation to monitor the game state as well as send leaderboard data to the model database for the MVC pattern. As for the game itself, we have constructed a board using a linked list data structure with a factory pattern for creating the game tiles which are responsible for moving each player’s pawns around the board. Additionally, we have created card and deck classes responsible for determining the player actions throughout the course of the game. We are also in the process of implementing a command pattern for moving the respective player pawns along the board which eventually will make it from the starting home tile position to the end of the goal position tile.

- **Changes/Issues Encountered:**
    - One of the issues we encountered at the start of the project was to use Hibernate as a potential ORM between our application and a small SQL database to store player data. We would additionally have used the tracker instance of the observer pattern to facilitate the connection between application data and the Hibernate ORM, but Hibernate seems to require additional dependencies on the project, which in some cases, are deprecated. Therefore, we made the decision to bypass Hibernate and instead continue on with the standard MySQL database connection protocols for Java and JavaFX applications
    - We are currently also implementing a command pattern to facilitate the movement of pawns along the board given the cards the player can pull from the random card deck. Since there are a large number of potential cards that can be drawn, implementing a command pattern for each of the specific behaviors each card has according to the Sorry! rules are taking some time. For the moment, we

are planning on keeping card behaviors as simple as possible by drawing a card and then moving the specified number of spaces indicated by the card value. We hope to address the various specific behaviors of each potential card in the second project sprint.

- **Pattern Usage:**
    - **Model View Controller:** We currently have a MVC pattern set up for managing the various application views that the user can interact with and query data from while playing the Sorry! game.
    - **Singleton/Observer:** We also have managed to implement an observer pattern with a pair of singleton pattern instances of a logger and tracker which are responsible for writing all game actions to text files and writing a summary of the game to the database. In our application, the logger instance is eagerly instantiated while the tracker instance is lazily instantiated. We hope to have this functionality close to finished by the end of this sprint.
    - **Factory:** We have implemented a factory pattern for creating the various game tiles that comprise the Sorry! board and are useful for differentiating between the home tiles, gateway tiles into the safe zone, the safe zone itself, and the slide tiles. Additionally, we have delegated responsibilities to each tile to handle the respective movement of pawns along the board.
    - **Object Pool:** We have implemented object pool patterns through the use of our card deck which is composed of cards that the users will pull from as the game progresses and through a player pool which is responsible for maintaining the state of whose turn it is during the gameplay.
    - **Command:** We are in the process of implementing a command pattern for the movement of the pawns around the board when the player presses the “Draw Card” button. For the time being, we are planning on having each player’s pawn move around the board as designated by the pulled card’s value, but we hope to integrate card specific behavior in the next project sprint. In the case of our application, we have the card object as the specific command component, the board object as the receiver and the invoker/client as the overarching game controller class.

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

Plan for the Next Iteration:

- In terms of goals for the next iteration, we want to:
    - Finish the command pattern by implementing the card specific behavior as outlined by the rules of Sorry!

    - Finish the observer and MVC patterns by having a small SQL database running to track various game statistics and display a leaderboard
    - Have the player select which specific pawn they want to move around the board following the command pattern implementation.
    - Add send pawn home behavior, where if a pawn ends its move on a tile occupied by an enemy pawn, the enemy pawn is sent to their home tile.
    - Add sliding behavior, which would allow one player to send the pawn of another player back to home if they are slid through. Or simply slide further than their card number stated.

- For the end of the project 7 sprint iteration, we hope to have a fully functional game and board states where other human users can play against each other. At the end of the game the player should be able to view relevant game statistics and have the option to play again.