

Project 7

Suresh Nayak
suresh.nayak@colorado.edu

Rajagopal Anandan
rajagopal.anandan@colorado.edu

GitHub Repo - <https://github.com/sureshnnayak/Ludo-Game>

1. Final State of System Statement

Overview:

We have successfully implemented the Ludo game as a mobile application for ios devices. In this game, simultaneously, four players can play. Each player in the game races toward the end position by moving any of their tokens out of 4 available tokens. A player would begin their game by tossing six and taking a token out of the base position. For rolling the dice, we have provided a button that triggers the roll dice feature upon clicking. If there is more than one potential token, the user gets to select the token. The selection of tokens is made user interactive.

Once the token is out, the player can move the token by boxes equivalent to the number obtained on the dice in each upcoming turn. During the game, a player can send other players' tokens back home if they move their token to the same box where an opponent player's token is present. There will be a few safe point boxes where all players are allowed to place their tokens without sending other players' tokens back home. The game will end when all the players, except for one, move all four tokens to the end position.

The board is designed as a 2-dimensional matrix of cells that gets displayed on the screen. Here, the cell is a structure that holds information about the color, what tokens are placed on it, and the next connected cell/cells. Based on these connections, the tokens are moved during the game.

Implementation details:

The complete project is implemented in Swift and SwiftUI. Below are the details of a few essential modules.

Player hierarchy: This is implemented as an interface to hold information about the player's color and tokens. The class is inherited by the Human and Computer class to provide concrete implementation. When a player object is created, the tokens are placed in their respective base position. The classes provide the following public function

- canMove: Check if the given token can move for the given dice value

- move: Moves the given token based on the given dice value
- kill: Finds and kills an opponent token present in the same box as the given token
- player status: returns if the player has won the game or not.

Token Class: To hold the information about an individual token, such as token position, basePosition, and homePosition, and some functions required to update these parameters.

Broadview: A class to render the board on the screen based on the token positions and player status. Some of the functionalities include:

- Provide a touch feature to recognize which token was selected
- Place the tokens in the appropriate cells.

Game Engine: Controls the whole logic of the game. It provides functionalities like initializing the game, initializing the players, and then providing the move functionality based on the input provided by the dice roll.

Layout: defines the Layout of the LudoBoard. Some of the functionalities provided by the class are:

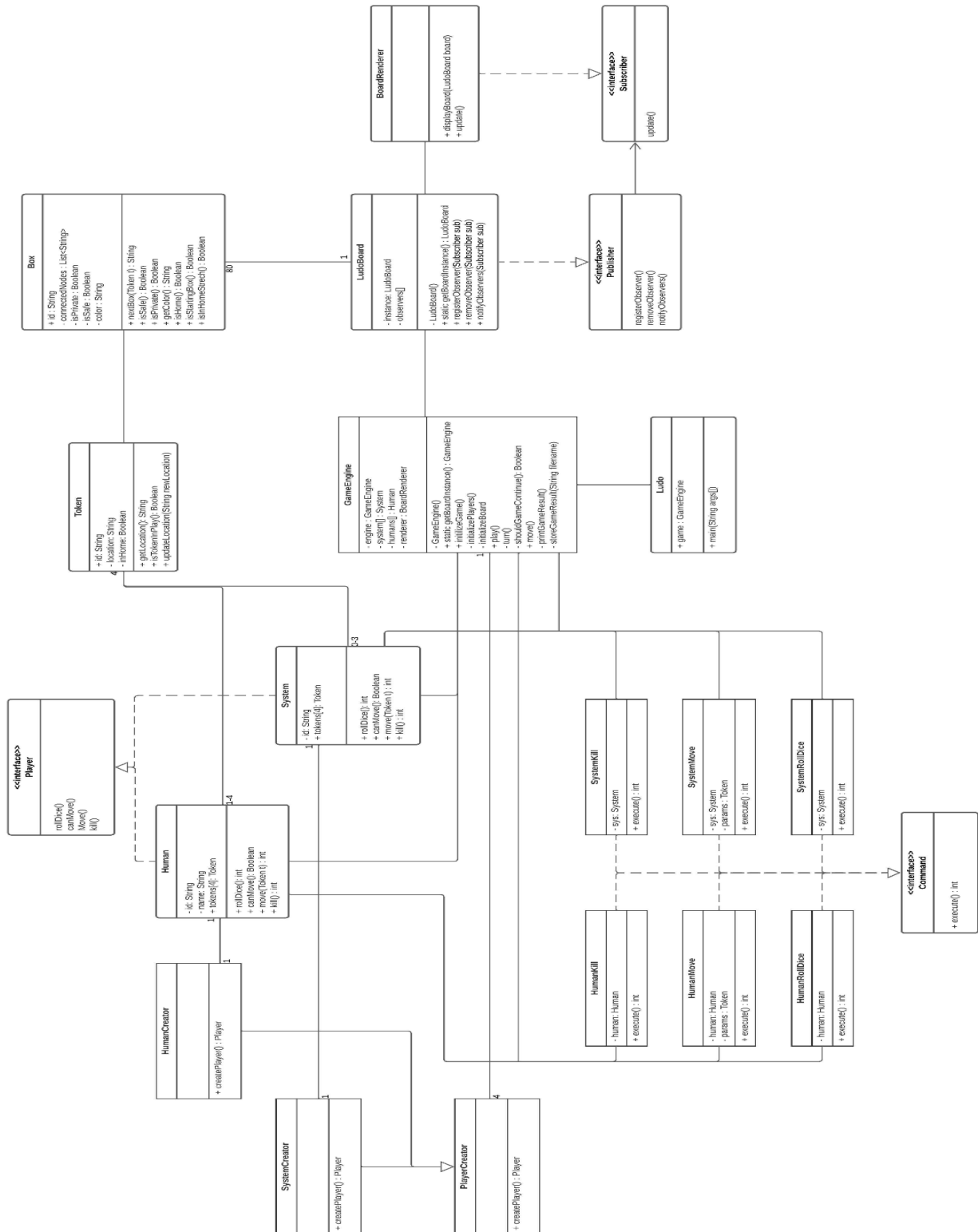
- drawCell: draw the layout by arranging the cells in a 2-Dimensional array.
- drawBoard: decides the dimension of the layout based on the device on which the application is running. These parameters are later used by drawCell.
- Populate Cells: place the tokens on appropriate cells.

Cell: a class that is used to instantiate a cell. The cell object holds information like cell color and connected cell.

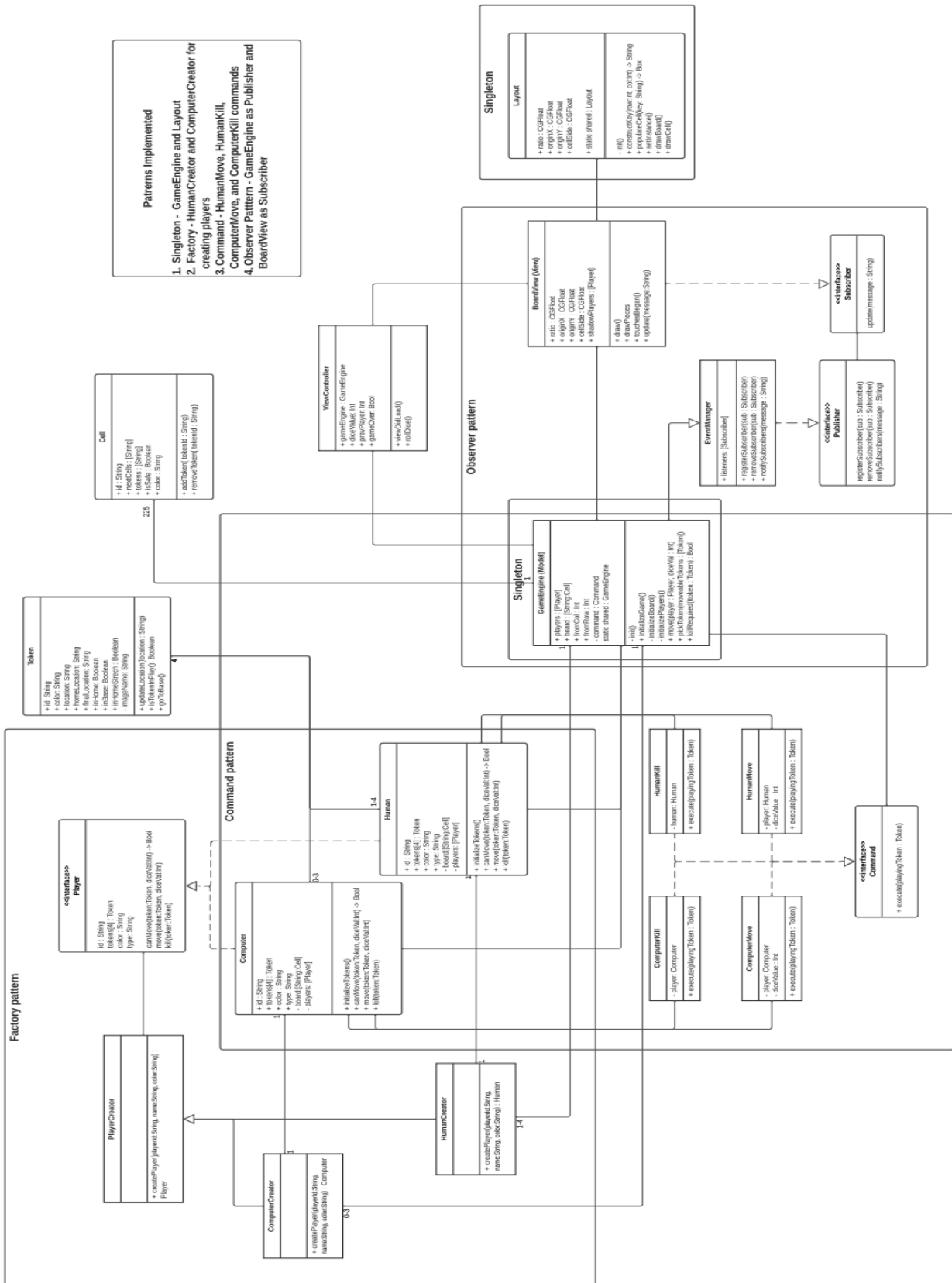
Features not Implemented:

1. Ability to select the number of players. This requires up to multiple pages, which was challenging to implement in the given timeframe.
2. Smoother UI: I think this would still be valid even if we spend a few more weeks coding, given that aesthetics of an application can always be made better. We have provided a minimalistic UI to support the game. Our primary goal was to have a minimal UI and have all the game rules implemented in the application.
3. Ability to select the colors for the players. As per the current implementation, the player would not be able to select the color for their token. This feature was mentioned in the project proposal. However, with time constraints, we were not able to implement this feature.

Initial design proposed



Final System design



Patterns Implemented

1. Factory pattern - Players are created using factory design
2. Command pattern - Player actions are performed using commands
3. Singleton - GameEngine and LudoBoard are singletons
4. Observer pattern - Updates in the model are passed to View using the observer pattern

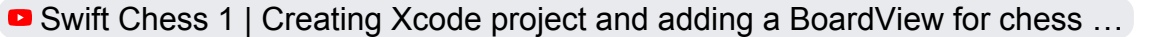
Changes from initial design

- Added ViewController class to communicate between Model and View
- New Layout class to display Ludo board in View
- Observer pattern used by GameEngine and BoardView
- Changes in Cell class to allow easy access to cell properties
- Moved roll dice feature from Player classes to ViewController
- Removed roll dice commands
- New EventManager class for Observer pattern

4. Original code Statement

All the code written for this application is original. However, we have referred to the official Swift documentation and some youtube videos which talked about some of the concepts that were used in our project.

Like:

- Implementation of the grid and providing the feature of selecting the tokens are derived from this Youtube tutorial:

- The screen rendering feature is derived from the swift official documentation:
<https://www.swift.org/documentation/>
- Running a background thread to notify the code about an
 - [activity:https://medium.com/@dhavalkansara51/swift-closures-with-completion-handler-de2d9f2fdd4f](https://medium.com/@dhavalkansara51/swift-closures-with-completion-handler-de2d9f2fdd4f)
 - <https://medium.com/@dhavalkansara51/completion-handler-in-swift-with-escaping-and-nonescaping-closures-1ea717dc93a4>

5. Statement on the OOAD process for your overall Semester Project

Implementing an application in Xcode requires us to follow some guidelines. Which takes away the freedom to design the system as we would design in Java or other programming languages.

For example, when we create a new project, the application comes with default files and classes, which would be treated as the entry point. This affected us while implementing the code, our UML did not consider these constraints. This is partially due to us not having experience in application development. However, having to redesign the UML and in certain cases structuring the code to follow the UML gave us a good experience of the Spiral development model.

So our takeaway from this project is

- The implementation constraints also need to be considered during the design phase. Because every language or platform is different. This requires us to consider some of the constraints the platform and programming language poses.
- Have a thorough understanding of the problem statement and have a well-reviewed design before starting the implementation. This would make life easier as the project grows bigger and more complicated.