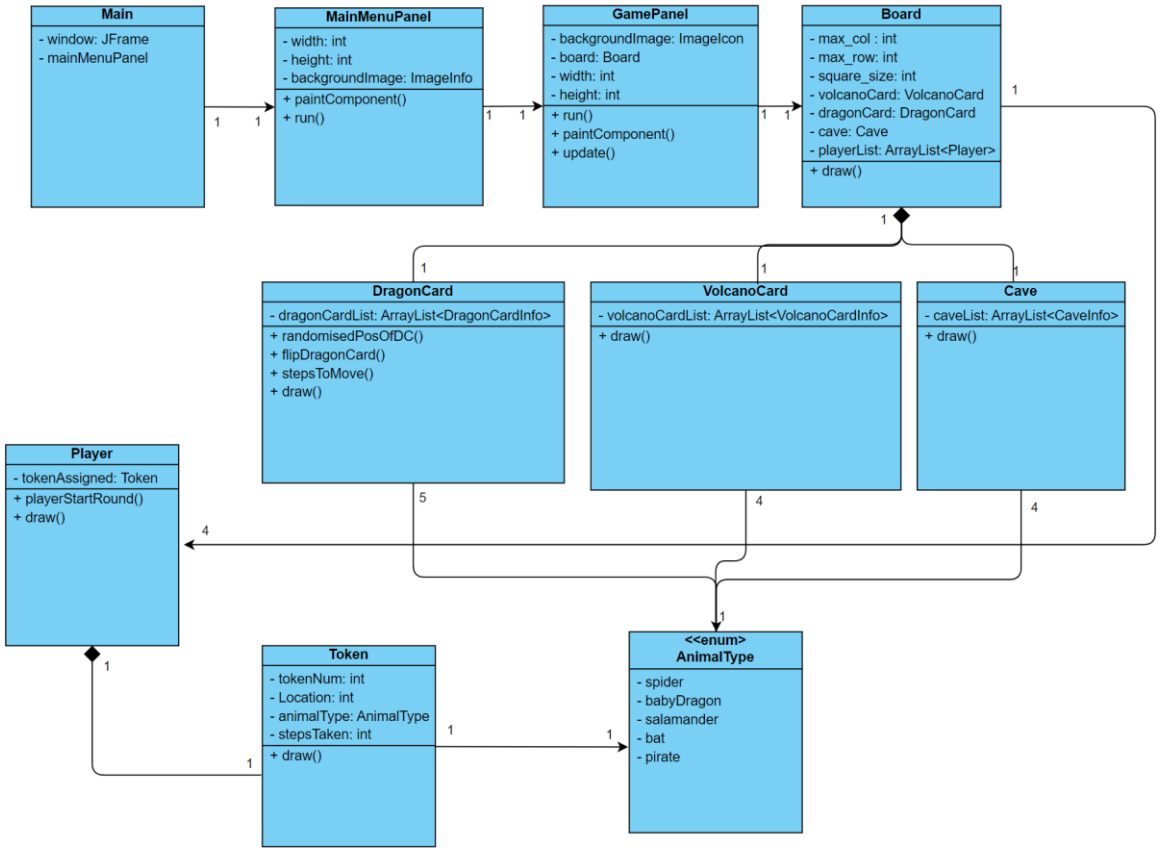
Chen Jac Kern, 32877188

FIT3077 sprint two deliverables

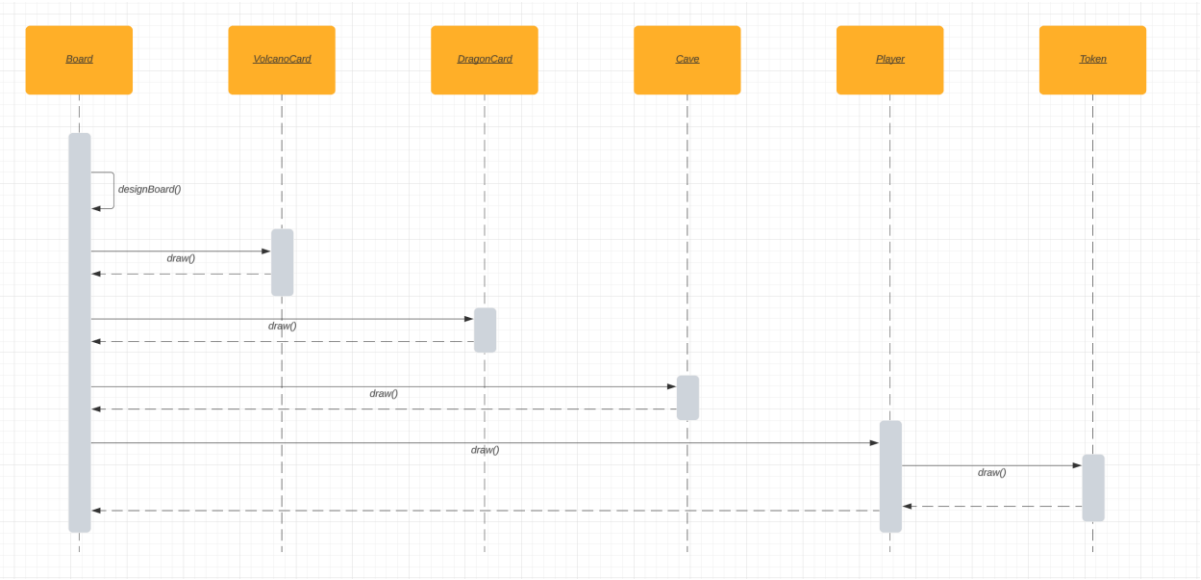
---

1. Class diagram



2. Key game functionalities

--- Set up initial game board: Setting up the game board will be instantiated the moment the game panel has been instantiated. In the GamePanel() constructor method, a list of players and game board will be created. Game board will be created by instantiating the Board class, where in the Board class, the Board() constructor method will instantiate the other classes required. For example, Cave(), DragonCard(), VolcanoCard() and Token(). These classes are instantiated and then placed in their corresponding location to mimic the game in real life.



--- Flipping of dragon cards: The first player is selected from the list of players in playerList and uses the playerRoundStart() function to start their round. When player click on the dragon card, the board class will have a mouse listener where it detects the player clicks on which location (dragon card). After detecting the location, the board class determines which dragon card the player selects and proceed to the dragon card class to instantiate the function flipDragonCards().

--- Movement of dragon tokens based on their current position as well as the last flipped dragon card: After player clicks on the dragon card, if the animal type of the dragon card is same as the token of animal type of the player, the card remains flipped and DragonCard class calculate how many steps (let's call it n) to take and return it to the Board class. Then, the Board class will move the token to the volcanoCardNumber + n.

--- change of turn to the next player: The board class use the mouse listener to detect which dragon card the player chooses and repeat the process until the animal type of dragon card is not the same as the animal type of player's token. Then, the system selects the next player in playerList and start their round using the playerRoundStart() function.

--- winning the game: when player selects a dragon card and the total number of steps travelled stored in token reaches 24, the player currently using the token wins

3. Design Rationales

--- Two key classes:

i. Player class represents individual player in the fiery dragon game. Each player has a token assigned to them and methods, such as starting a round. The player class was created as a class as encapsulating player-specific functionality within the class let the code become more organized and easier to maintain. Also, having a separate class allow multiple instantiations of player object, each object representing a player in the game

ii. Board class were designed as a class due to it creates the entire board, which includes dragon cards, volcano cards and caves. It also has methods to design the board layout and manage player interactions with the board. Similar to player class, the board class encapsulates functionality which improves organization and easier modification. The

board class can also simply be instantiated once again if needed, for example restarting the game.

--- Two key relationships

i. The relationship between player and token class is considered aggregation as a player can possess a token, but a token can exist independently of the player
ii. Board and player class has a relationship of composition as when a board is instantiated, players are also created and board manages how the player proceeds. Players cannot exist independently of the board, for example, when the board is destroyed, the players are destroyed too

--- Inheritance

i. No inheritance are implemented in this project as each classes represents a distinct entity with its own unique of attributes and methods.
ii. Simplicity: the design remains simpler as inheritance can lead to unnecessary complexity
iii. Flexibility: without inheritance, each class can evolve independently without being coupled to a superclass, which allows more flexibility in modification for specific classes

--- cardinalities

i. Cardinality between player and board class will be 4…1 as when a board is instantiated, 4 players with different number will also be instantiated. As this is compulsory, the cardinality will be 4…1 no matter what
ii. Cardinality between dragon card class and animal type class (enum) will be 1…5 as each dragon card is specifically linked with one of the five animals in enumeration class

4. Design Patterns

--- Composite pattern: In the Board class, the design board method creates grid like structure for placement of cards such as volcano cards, dragon cards and caves. Each element of the grid cells behaves uniformly, whether it's a part of a larger structure or not

--- Strategy pattern: All the classes that is required to draw on the board has a similar method of drawing in the draw() method. When the board needs to draw something, it simply delegates the task to the appropriate classes based on what is required. This keeps the drawing logic modular and easy to change.

--- Factory method patter: This pattern was not implemented as instantiation of objects such as players is straightforward and does not require any complex creation logic. Each player is just created with certain parameters and does not require a separate factory class to handle object creation