Classes in the UML
CaveTile - Is the class that represents the cave tiles that occupy the game board
NonCaveTile - The class that represents the tiles that the players move along on the regular game board
Dragon - Dragon class is the class that represents the player object. The dragon's act as the player interacting with the game board and performing actions based on the results of other actions with the rest of the game's components
GameBoard - GameBoard class is the class that acts as a representation of the gameboard. It consists of the tiles that make up the game board, the players playing the game and the chit cards arrangement in the game board.
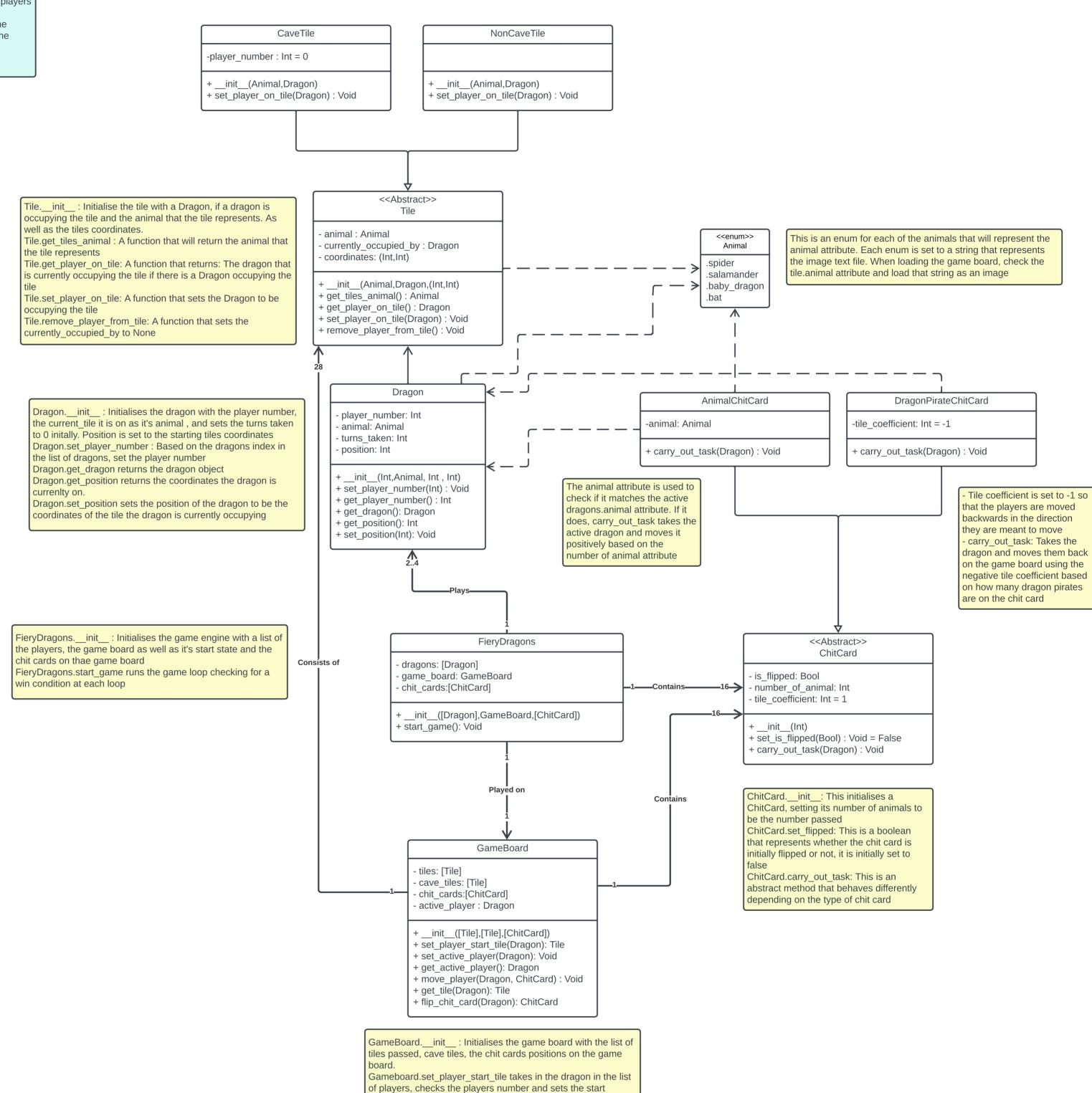AnimalChitCard is the chit cards that represent animals which players have to match to move forward on the game board. Based on the number of animals as well as if the dragons animal matches the chit cards animal.
DragonPirateChitCard is the chit cards that represent the dragon pirates, it's interaction with players is to move them back instead of forward based on whichever player has flipped the chit card
FieryDragons is the game engine that runs all of the game loops and interactions between the currently active player and the elements on the game board. It keeps executing the loop till the game is quit or a player has won the game.

CaveTile.set_player_on_tile: This method has a different behaviour than NonCaveTile as we want to check if the player has returned to the cave. Place the dragon on the tile, check if the Dragon.player_number == Cave_Tile.player_number. If it matches, end the game and trigger win condition. If it doesn't match, dont allow the player to be placed on the tile
• Player Number is initially 0 as there are always 4 caves but can be 2,3 or 4 players. This allows the caves to not be possessed by a player

NonCaveTile.set_player_on_tile: This method has a different behaviour than CaveTile as we just want to set the dragon to be tiles.dragon attribute, we also want to set dragons.animal attribute to be the tiles.animal attribute. We don't need to check if the tile is matching a player_number, we do however need to check if the tile is currently, occupied, if it is don't place the player and end the active players turn, move to the next players turn. If not occupied, carry out the normal action of placing a player on the tile

CaveTile

-player_number : Int = 0

+ __init__(Animal,Dragon)
+ set_player_on_tile(Dragon) : Void

NonCaveTile

+ __init__(Animal,Dragon)
+ set_player_on_tile(Dragon) : Void

Tile.__init__ : Initialise the tile with a Dragon, if a dragon is occupying the tile and the animal that the tile represents. As well as the tiles coordinates.
Tile.get_tiles_animal : A function that will return the animal that the tile represents
Tile.get_player_on_tile: A function that returns: The dragon that is currently occupying the tile if there is a Dragon occupying the tile
Tile.set_player_on_tile: A function that sets the Dragon to be occupying the tile
Tile.remove_player_from_tile: A function that sets the currently_occupied_by to None

<<Abstract>>
Tile

- animal : Animal
- currently_occupied_by : Dragon
- coordinates : (Int,Int)

+ __init__(Animal,Dragon,(Int,Int))
+ get_tiles_animal() : Animal
+ get_player_on_tile() : Dragon
+ set_player_on_tile(Dragon) : Void
+ remove_player_from_tile() : Void

<<enum>>
Animal

.spider
.salamander
.baby_dragon
.bat

This is an enum for each of the animals that will represent the animal attribute. Each enum is set to a string that represents the image text file. When loading the game board, check the tile.animal attribute and load that string as an image

Dragon.__init__ : Initialises the dragon with the player number, the current_tile it is on as it's animal , and sets the turns taken to 0 initally. Position is set to the starting tiles coordinates
Dragon.set_player_number : Based on the dragons index in the list of dragons, set the player number
Dragon.get_dragon returns the dragon object
Dragon.get_position returns the coordinates the dragon is currenlty on.
Dragon.set_position sets the position of the dragon to be the coordinates of the tile the dragon is currently occupying

Dragon

- player_number: Int
- animal: Animal
- turns_taken: Int
- position: Int

+ __init__(Int,Animal, Int , Int)
+ set_player_number(Int) : Void
+ get_player_number() : Int
+ get_dragon(): Dragon
+ get_position(): Int
+ set_position(Int): Void

AnimalChitCard

-animal: Animal

+ carry_out_task(Dragon) : Void

The animal attribute is used to check if it matches the active dragons.animal attribute. If it does, carry_out_task takes the active dragon and moves it positively based on the number of animal attribute

DragonPirateChitCard

-tile_coefficient: Int = -1

+ carry_out_task(Dragon) : Void

- Tile coefficient is set to -1 so that the players are moved backwards in the direction they are meant to move
- carry_out_task: Takes the dragon and moves them back on the game board using the negative tile coefficient based on how many dragon pirates are on the chit card

FieryDragons.__init__ : Initialises the game engine with a list of the players, the game board as well as it's start state and the chit cards on thae game board
FieryDragons.start_game runs the game loop checking for a win condition at each loop

FieryDragons

- dragons: [Dragon]
- game_board: GameBoard
- chit_cards:[ChitCard]

+ __init__([Dragon],GameBoard,[ChitCard])
+ start_game(): Void

<<Abstract>>
ChitCard

- is_flipped: Bool
- number_of_animal: Int
- tile_coefficient: Int = 1

+ __init__(Int)
+ set_is_flipped(Bool) : Void = False
+ carry_out_task(Dragon) : Void

ChitCard.__init__: This initialises a ChitCard, setting its number of animals to be the number passed
ChitCard.set_flipped: This is a boolean that represents whether the chit card is initially flipped or not, it is initially set to false
ChitCard.carry_out_task: This is an abstract method that behaves differently depending on the type of chit card

GameBoard

- tiles: [Tile]
- cave_tiles: [Tile]
- chit_cards:[ChitCard]
- active_player : Dragon

+ __init__([Tile],[Tile],[ChitCard])
+ set_player_start_tile(Dragon): Tile
+ set_active_player(Dragon): Void
+ get_active_player(): Dragon
+ move_player(Dragon, ChitCard) : Void
+ get_tile(Dragon): Tile
+ flip_chit_card(Dragon): ChitCard

GameBoard.__init__ : Initialises the game board with the list of tiles passed, cave tiles, the chit cards positions on the game board.
Gameboard.set_player_start_tile takes in the dragon in the list of players, checks the players number and sets the start position based on the number of players participating in the run of the game
Gameboard.set_active_player sets the current active player to be moved on the game board
Gameboard.get_active_player returns the active player to target interactions with chit cards and to check the win condition for
Gameboard.move_player moves the dragon that is currently active player a number of steps based on the chit card flipped
Gameboard.get_tile returns the tile that the dragon is currenlty occupying.
Gameboard.flip_chit_card flips a chit card over if it hasn't been flipped and unflips all the flipped chit_cards at the end of the players turn

28

2..4
Plays

1
Contains
16

16

Consists of
1

Played on
1

1
Contains
1