Class-Responsibility-Collaboration (CRC) Cards

FieryDragonsApplication	
• Knows and tracks whether a player has won • Initialises the game by requesting the Factory and Manager classes to create the view components and the state • Requests the manager classes to handle updates when a player's turn ends or when the game ends • Requests the chit card view components to listen for user input(clicks), and asks the Manager classes to respond to them. • Allows the next player to play their turn if the game has not ended	Player PlayerTurnManager TextDisplayManager ChitCardFlipManager ChitCardAdapter VolcanoRingFactory DragonTokenFactory CaveFactory ChitCardFactory ButtonFactory TextFactory MainMenu

Description:
It is the main entry point of the game, extending the GameApplication class provided by the FXGL framework. It provides game settings like the background, application window dimensions, and the MainMenu, which is what the Player sees when the game is first launched. It serves as the central hub through which game components are initialised and managed. The onUpdate() method that is called by FXGL every frame allows custom logic to be implemented to handle different events.

PlayerTurnManager	
• Knows the current player's turn • Creates and tracks the player instances • Instructs DragonToken to move (valid move) • Sets the dragon token's destination Volcano Tile to "occupied" • Updates DragonToken's state: current Volcano tile and total movement, whether it has moved out of the cave • Transitions turn to the next player	DragonToken Player VolcanoRingFactory

Description:
The PlayerTurnManager creates the Players during initial setup, assigning them a random starting animal (will be shown on the cave). It continues to track the players throughout the game, and tracks which player is currently playing its turn.

As the controller, whenever a successful move is made, it asks DragonToken to move (view) and updates the state of VolcanoCard(Tile) and DragonToken (model). It also manages the change of turn to the next player.

Player	
Knows the animal on the random cave it starts on Controls its dragon token Checks whether its turn should end (invalid move) Identifies the destination volcano card tile (valid move)	PlayerTurnManager CaveFactory ChitCardAdaptor DragonToken

Description:
The Player class is created by the PlayerTurnManager during the initial setup, where it is assigned a random animal. This animal type will be passed on to the CaveFactory so that it can render the required animal in the cave.

To fulfill its responsibilities, it requires information from its
(i) DragonToken (e.g. whether the dragon token has moved out of the cave, the animal on the tile the dragon token is currently on)

(ii) the ChitCardAdaptor, which returns the ChitCard instance(model) bound to the UI shape (view), for the player to access the animal type on it.

ChitCardFlipManager	

• Tracks all the view components (Circle) of the ChitCards(uncovered and covered)
• Animates flipping chit cards clicked by Players
  o Set the covered side to invisible and uncovered side to visible
• Toggles the click listeners of the ChitCard's uncovered view component (Circle)
  o Toggle off once the card has been uncovered
  o Toggle on when player's turn ends
	ChitCard
	ChitCardAdapter
	Circle

Description:
The ChitCardFlipManager performs the rotation animation

(i) whenever a player clicks on a covered chit card shape

(ii) on all uncovered chit card shapes whenever a player's turn ends

It maintains an array of covered and uncovered chit card shapes to help with this. To get the chit card view components for modifying this array, it collaborates with the ChitCard model, since the ChitCard model contains references to both its covered and uncovered form. The ChitCardAdapter also helps to get the covered form of the ChitCard at the end of a Player's turn, so that its visibility can be set to 'True'.

Animal	
• Knows the base image path of the resource used for rendering animals on ChitCards/VolcanoCard tiles. • Knows how many animals are used • Helps to create and fill a shape with the ImagePattern (with the specified number of animals) • Contains a factory method which returns a new instance of a specific animal based on the AnimalType enum.	AnimalType BabyDragon Bat DragonPirate Salamader Spider

Description:
The animal class helps create custom animal image patterns that can be applied as the 'fill' property of Shapes. The customisation involves the specific animal type and the number of animals to be displayed in the image.

VolcanoRingFactory	
• Manages a single reference to the volcano ring of VolcanoCards • Creates volcano segments from the configuration of segments. • Initialises the view (Rectangle) component from randomly arranged volcano segments. It is arranged as a Group in the shape of a ring, and the randomisation is provided by the Utils class • Initialises the model(VolcanoCard component) from the randomised volcano segments.	VolcanoSegment VolcanoCard Util Rectangle Group Animal

Description:
VolcanoSegment is a static helper class within VolcanoRingFactory to help with the creation of segments. The Animal class helps to fill the Rectangle with the specific animal image. Here we make use of the Liskov Substitution Principle, since the abstract fillUIShape() method will work

with any inheritor of the Animal class. In summary, this class helps initialise the view and model for the volcano ring.

DragonToken	
• Executes the movement animation whenever a valid move is made. • Knows its position on the circumference of the circular path where it is free to move around the volcano ring (measured by an angle) • Knows the view component (Rectangle) bound to it and the text label that travels with it • Knows the current volcano card tile (and the AnimalType of it) it is on • Knows its starting volcano card tile • Tracks and manages its total movement count • Tracks whether it has moved out of the cave	VolcanoCard Text Rectangle Timeline KeyValue KeyFrame

Description:

The dragon token represents the player's piece on the game board and is moved according to the game's rules. It tracks the current position and movement count, which are essential for determining the game's progress and outcome. When the total movement count reaches the number of volcano cards, a win condition is detected.

It has two main responsibilities: animating the movement (Using Timeline, KeyValue, KeyFrame) and providing a public interface for classes to read/write to its properties.

Moving the token could be the responsibility of a GamePieceMovementController, which would handle the logic of moving any game piece, including dragon tokens. This would allow DragonToken to act more as a data holder for the token's state, with all movement logic externalised. However, in the current game, there are no other moving pieces, so it is reasonable for it to handle this responsibility.