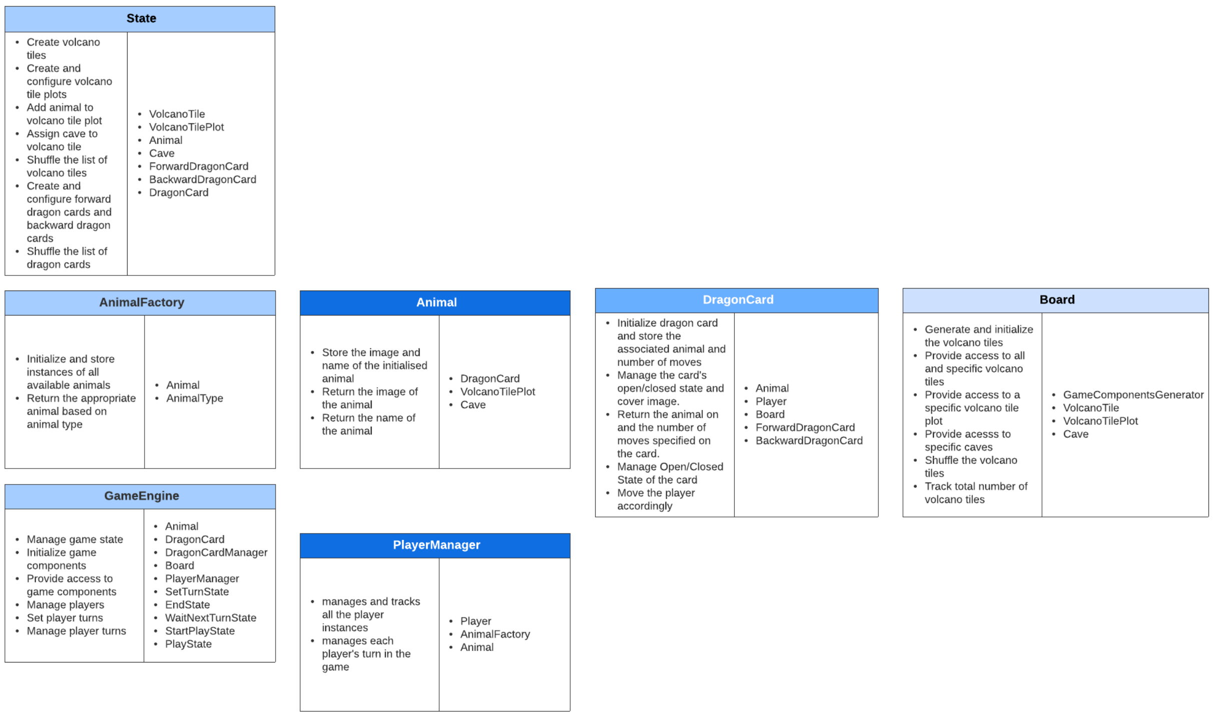

State	
• Create volcano tiles • Create and configure volcano tile plots • Add animal to volcano tile plot • Assign cave to volcano tile • Shuffle the list of volcano tiles • Create and configure forward dragon cards and backward dragon cards • Shuffle the list of dragon cards	• VolcanoTile • VolcanoTilePlot • Animal • Cave • ForwardDragonCard • BackwardDragonCard • DragonCard

AnimalFactory	
• Initialize and store instances of all available animals • Return the appropriate animal based on animal type	• Animal • AnimalType

Animal	
• Store the image and name of the initialised animal • Return the image of the animal • Return the name of the animal	• DragonCard • VolcanoTilePlot • Cave

DragonCard	
• Initialize dragon card and store the associated animal and number of moves • Manage the card's open/closed state and cover image. • Return the animal on and the number of moves specified on the card. • Manage Open/Closed State of the card • Move the player accordingly	• Animal • Player • Board • ForwardDragonCard • BackwardDragonCard

Board	
• Generate and initialize the volcano tiles • Provide access to all and specific volcano tiles • Provide access to a specific volcano tile plot • Provide acesss to specific caves • Shuffle the volcano tiles • Track total number of volcano tiles	• GameComponentsGenerator • VolcanoTile • VolcanoTilePlot • Cave

GameEngine	
• Manage game state • Initialize game components • Provide access to game components • Manage players • Set player turns • Manage player turns	• Animal • DragonCard • DragonCardManager • Board • PlayerManager • SetTurnState • EndState • WaitNextTurnState • StartPlayState • PlayState

PlayerManager	
• manages and tracks all the player instances • manages each player's turn in the game	• Player • AnimalFactory • Animal

Class	Purpose
State	The class is used exclusively by the GameEngine to inform the display (UI) what to render based on the current game state and what actions to perform on user interaction.
GameEngine	Controls the logic of game actions, tracks the current state, and ensures a single instance manages the game state.
PlayerManager	This class is responsible for managing player-related operations in the game.
AnimalFactory	Provides singleton access to shared Animal instances for use in DragonCard, VolcanoTilePlot, and Cave.
Animal	Represents an abstract class that represents an animal that can be used on a DragonCard, in a VolcanoTilePlot, and in a Cave.
Dragoncard	Represents an abstract dragon card that has an associated animal and number of moves, and can move a player on the board.
Board	Manages the collection of VolcanoTile objects that make up the game board.

Discarded Alternatives	Reason
Animal class is not implemented as an abstract class	Led to code duplication and reduced flexibility, as each animal class would have to implement the same methods independently for different animal types
DragonCard class is not implemented as an abstract class	Lacked polymorphism, making it difficult to create and manage different types of dragon cards uniformly.
State class is not implemented	Resulted in complex game logic being spread across multiple classes, leading to a lack of clear separation of concerns and making the code harder to maintain and extend.
PlayerManager class is not implemented	Without a factory pattern, it became cumbersome to manage the creation of Animal instances, increasing the risk of inconsistent object creation and making it harder to manage shared instances.
GameComponentsGenerator class is not implemented	Having the Player class handle everything led to a violation of the single responsibility principle, making the Player class overly complex and difficult to manage.