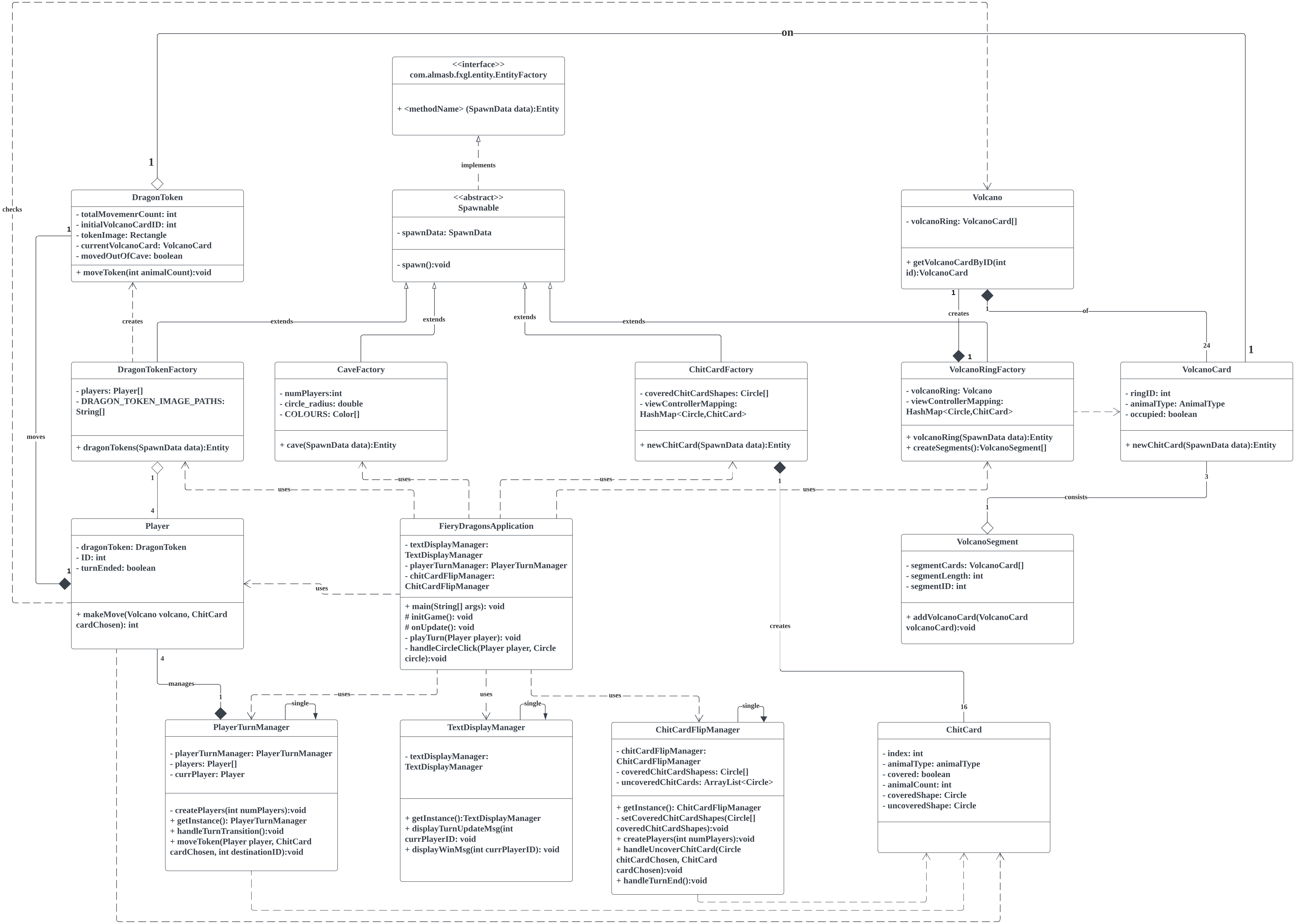


<<interface>>
com.almasb.fxgl.entity.EntityFactory

+ <methodName> (SpawnData data):Entity

DragonToken

- totalMovemenrCount: int
- initialVolcanoCardID: int
- tokenImage: Rectangle
- currentVolcanoCard: VolcanoCard
- movedOutOfCave: boolean

+ moveToken(int animalCount):void

<<abstract>>
Spawnable

- spawnData: SpawnData

- spawn():void

Volcano

- volcanoRing: VolcanoCard[]

+ getVolcanoCardByID(int id):VolcanoCard

DragonTokenFactory

- players: Player[]
- DRAGON_TOKEN_IMAGE_PATHS: String[]

+ dragonTokens(SpawnData data):Entity

CaveFactory

- numPlayers:int
- circle_radius: double
- COLOURS: Color[]

+ cave(SpawnData data):Entity

ChitCardFactory

- coveredChitCardShapes: Circle[]
- viewControllerMapping: HashMap<Circle,ChitCard>

+ newChitCard(SpawnData data):Entity

VolcanoRingFactory

- volcanoRing: Volcano
- viewControllerMapping: HashMap<Circle,ChitCard>

+ volcanoRing(SpawnData data):Entity
+ createSegments():VolcanoSegment[]

VolcanoCard

- ringID: int
- animalType: AnimalType
- occupied: boolean

+ newChitCard(SpawnData data):Entity

Player

- dragonToken: DragonToken
- ID: int
- turnEnded: boolean

+ makeMove(Volcano volcano, ChitCard cardChosen): int

FieryDragonsApplication

- textDisplayManager: TextDisplayManager
- playerTurnManager: PlayerTurnManager
- chitCardFlipManager: ChitCardFlipManager

+ main(String[] args): void
# initGame(): void
# onUpdate(): void
- playTurn(Player player): void
- handleCircleClick(Player player, Circle circle):void

VolcanoSegment

- segmentCards: VolcanoCard[]
- segmentLength: int
- segmentID: int

+ addVolcanoCard(VolcanoCard volcanoCard):void

PlayerTurnManager

- playerTurnManager: PlayerTurnManager
- players: Player[]
- currPlayer: Player

- createPlayers(int numPlayers):void
+ getInstance(): PlayerTurnManager
+ handleTurnTransition():void
+ moveToken(Player player, ChitCard cardChosen, int destinationID):void

TextDisplayManager

- textDisplayManager: TextDisplayManager

+ getInstance():TextDisplayManager
+ displayTurnUpdateMsg(int currPlayerID: void
+ displayWinMsg(int currPlayerID): void

ChitCardFlipManager

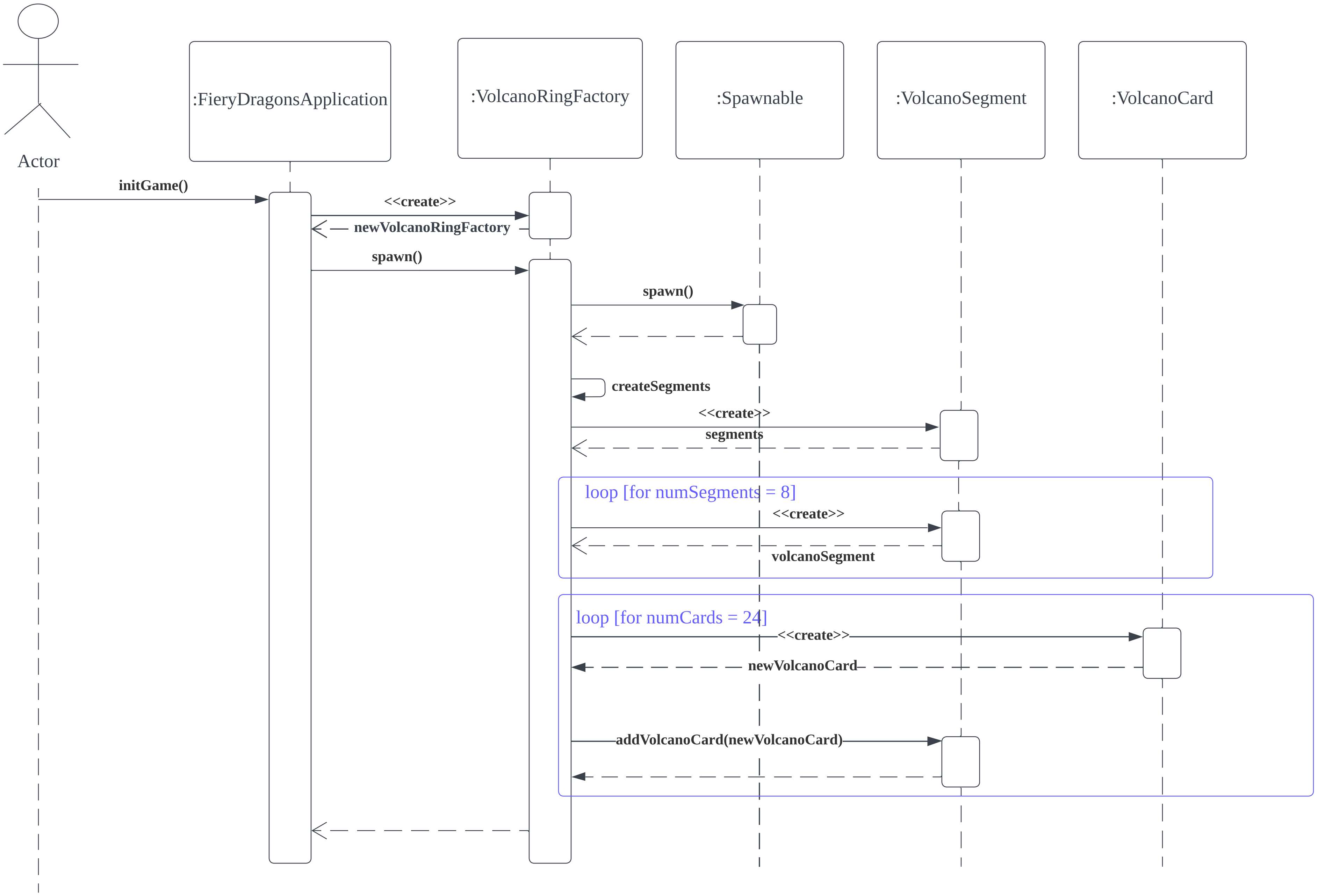
- chitCardFlipManager: ChitCardFlipManager
- coveredChitCardShapess: Circle[]
- uncoveredChitCards: ArrayList<Circle>

+ getInstance(): ChitCardFlipManager
- setCoveredChitCardShapes(Circle[] coveredChitCardShapes):void
+ createPlayers(int numPlayers):void
+ handleUncoverChitCard(Circle chitCardChosen, ChitCard cardChosen):void
+ handleTurnEnd():void

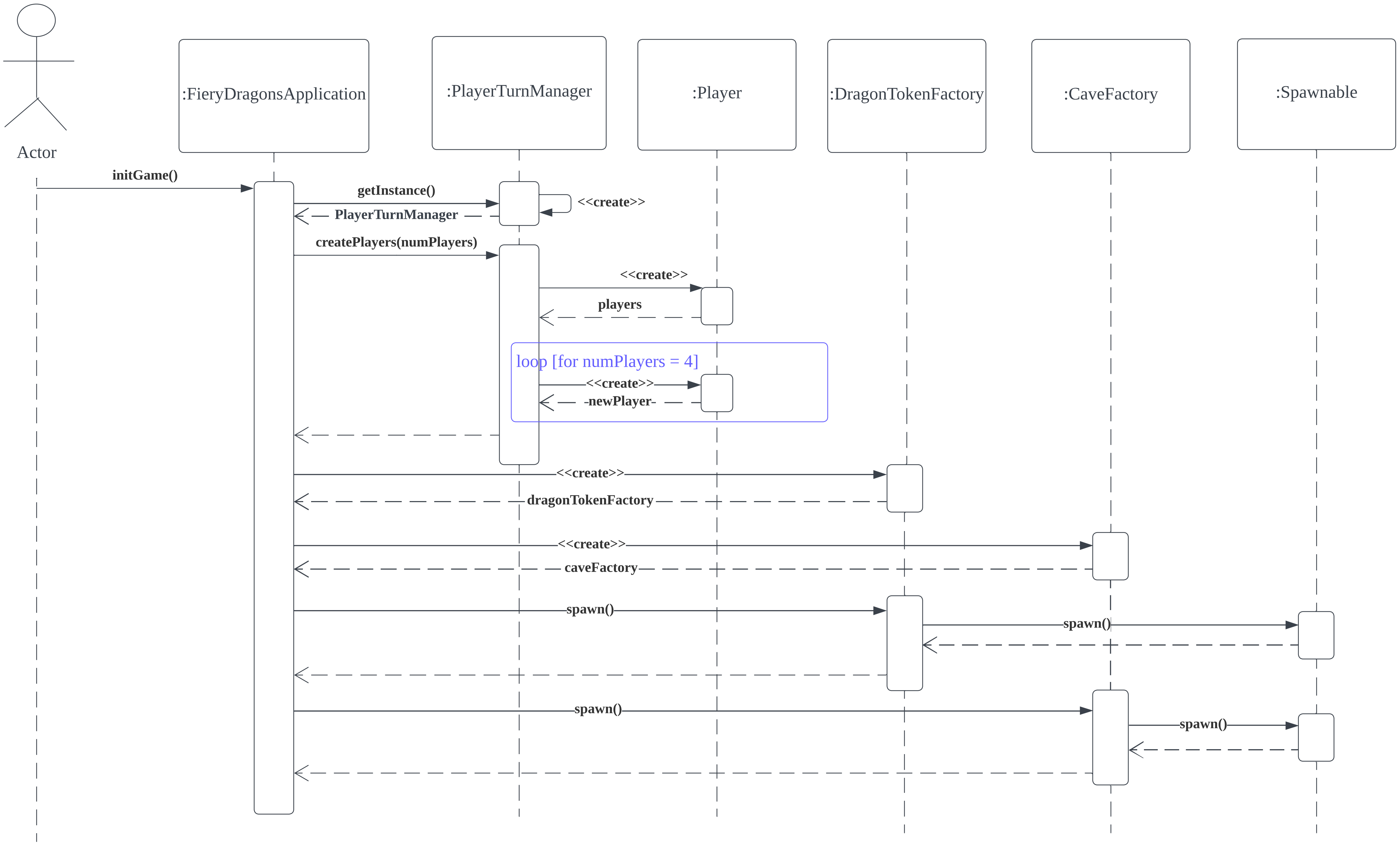
ChitCard

- index: int
- animalType: animalType
- covered: boolean
- animalCount: int
- coveredShape: Circle
- uncoveredShape: Circle

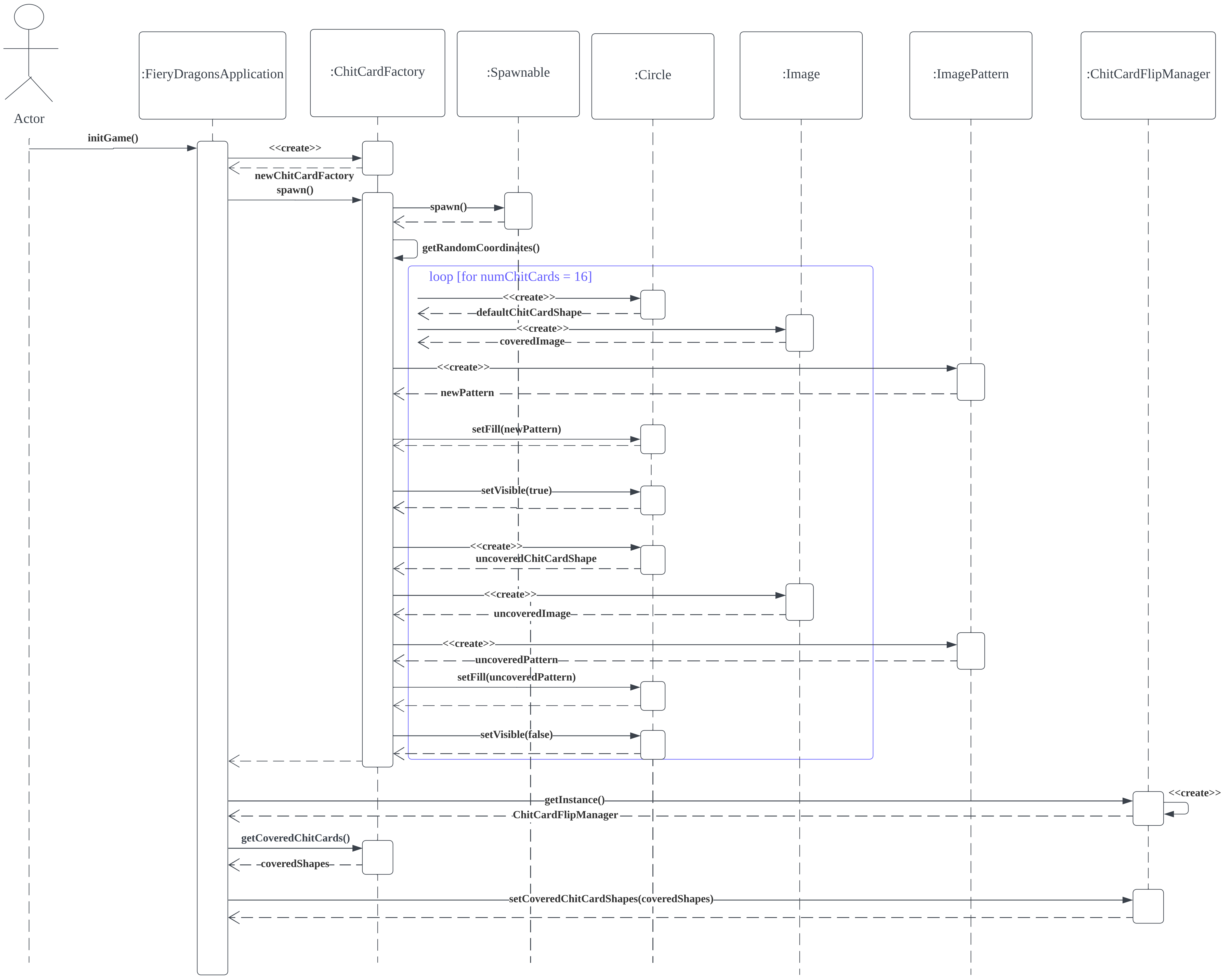
Sequence Diagram for setup of VolcanoRing



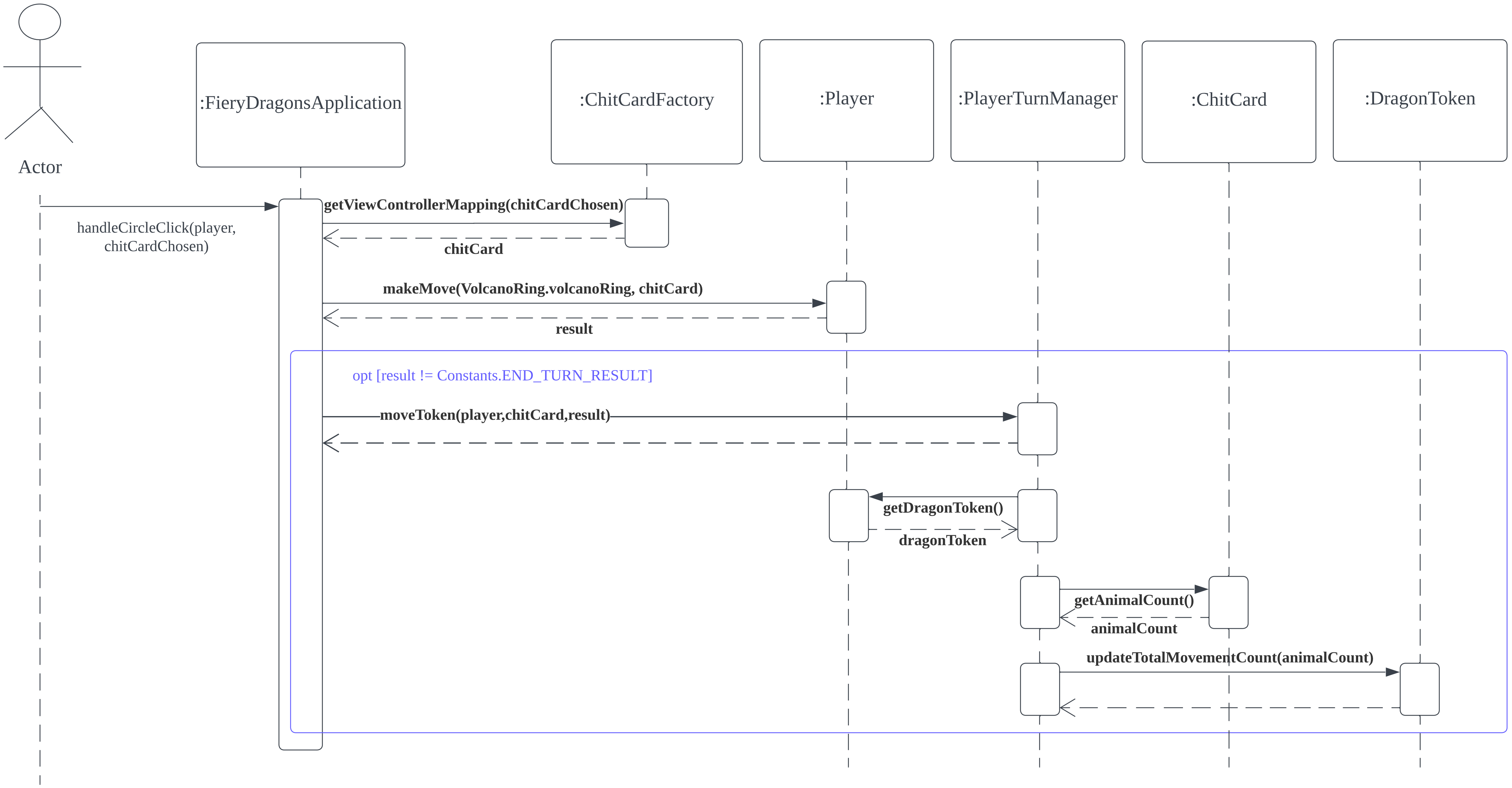
Sequence Diagram for setup of DragonToken and Cave, which are directly related to Player



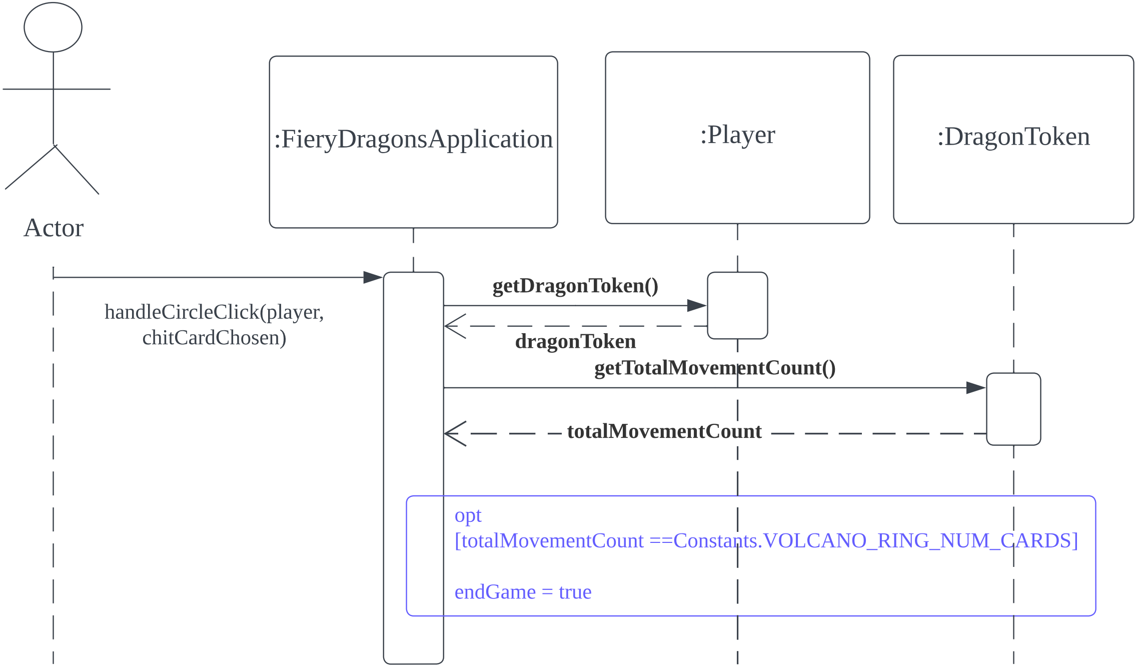
Sequence Diagram for ChitCard-related setup



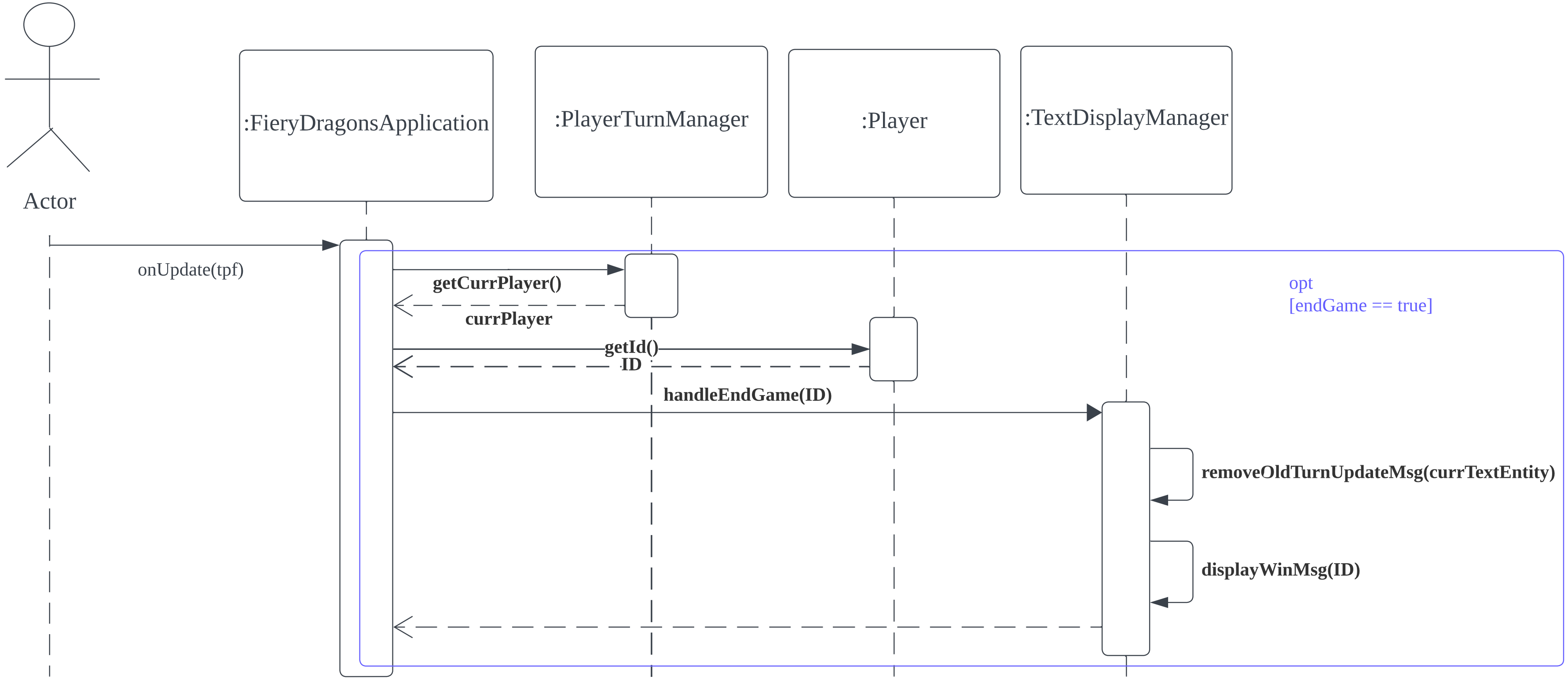
Sequence Diagram for updating key variable(DragonToken. totalMovementCount) used for detecting win condition


Actor
:FieryDragonsApplication
:ChitCardFactory
:Player
:PlayerTurnManager
:ChitCard
:DragonToken
handleCircleClick(player, chitCardChosen)
getViewControllerMapping(chitCardChosen)
chitCard
makeMove(VolcanoRing.volcanoRing, chitCard)
result
opt [result != Constants.END_TURN_RESULT]
moveToken(player,chitCard,result)
getDragonToken()
dragonToken
getAnimalCount()
animalCount
updateTotalMovementCount(animalCount)

Sequence Diagram for Detecting Win Condition



Sequence Diagram for Displaying Win Message



Object Oriented Design and Design Rationales

- Class Diagram(including attributes, methods, cardinalities)
- Sequence diagrams (setup initial board, all functionalities: flipping chit cards, move dragon token, change of turn to next player, winning game)
- Design rationales
  - 2 classes (why not methods)
  - 2 relationships (why aggregation not composition)
  - Inheritance why used or not used
  - 2 cardinalities
- Minimum 3 Design patterns used(or why not used)

Video demonstration (Add timestamps for references to game rules)

Talking points:

- Setup of board. All chit cards covered, indistinguishable. The volcano cards segments are random. Message indicating player's turn.
- Flipping chit card (explaining current animal versus chit card animal, dragon token movement). Encounter mismatch, so go to next player's turn (see displayed message)
- Winning the game when reach cave
- If have time
  - cannot go back further from cave, i.e. only move back further if have moved out of cave
  - if destination exceeds the initial cave position, turn ends
  - cannot use a dragon card that has already been flipped