<<interface>> Animal

- getPicResource(number:int)

ChitCard
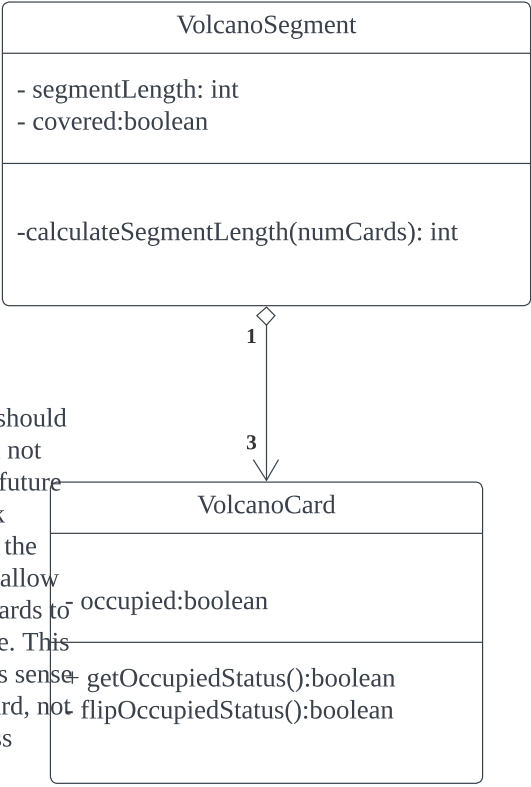
- covered:boolean

- flipCard()

Since total cards to form volcano ring differs, we try to calculate it such that each segment gets an equal amount. By modifying the logic in the method once, it works for all segments

VolcanoSegment

- segmentLength: int
- covered:boolean

-calculateSegmentLength(numCards): int

the getChoice checks if a ChitCard is to be flipped from the choice. if yes, then pass this info to ChitCard for it to be flipped

Player

- covered:boolean

- getChoice(playerChoice)
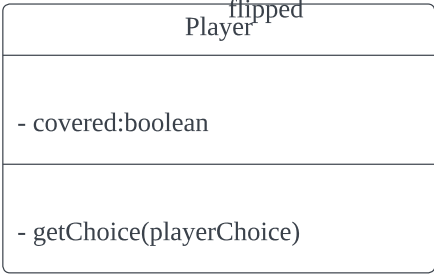
this checking is handled by player, we should not pass the information to ChitCard since it makes ChitCard handle too much
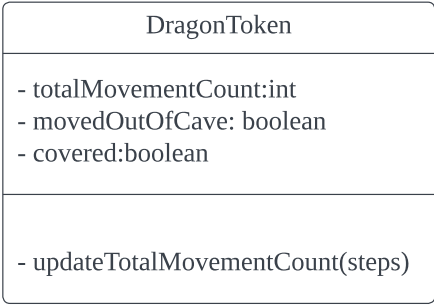
1

3

Why volcano card should be own class and not method: maybe in future want to tweak functionality for the occupation status: allow certain number of cards to be on it, not just one. This responsibility makes sense to be for volcano card, not the player class

VolcanoCard

- occupied:boolean

- getOccupiedStatus():boolean
- flipOccupiedStatus():boolean

DragonToken

- totalMovementCount:int
- movedOutOfCave: boolean
- covered:boolean

- updateTotalMovementCount(steps)

Alternative is to have interface for animal and dragonpirate, then we will have if-else on the enum to determine which resource to fetch

If we have single class, the class can directly point to the resource in the constructor, so this is the best way to do it, however doing it once for every animal is not feasible, so we do one class for animal, once for dragon pirate

However, having a class that only does this is waste of space, so method via interface is best

<<interface>> DragonPirate

- getPicResource(number:int)

ChitCardFactory

- animalCount:int
- animalType: enum