UML Class Diagram: Hungry Hungry Cow

model

Entity
- position: Position
- + getPosition()

control

Position
- - x: int
- - y: int
- + getX()
- + setX()
- + getY()
- + setY()

factory

HungryCowAnimateFactory
- + INITIAL_PLAYER_SCORE = 0: int
- + makePlayer()
- + makeEnemy()

HungryCowInanimateFactory
- + REGULAR_REWARD_VALUE = 10: int
- + BONUS_REWARD_VALUE = 20: int
- + PUNISHMENT_VALUE = 10: int
- + makeRegularReward()
- + makeBonusReward()
- + makePunishment()

BoardFactory
- + makeBoard(int[][] boardData, animateFactory: HungryCowAnimateFactory, inanimateFactory: HungryCowInanimateFactory)

animate

<<interface>> Animate
- + move(direction: Move)

Player
- - score: int
- + getScore()
- + setScore()

Enemy

unanimate

<> Unanimate
- - value: int
- + getValue()

BonusReward
- + randomizePosition()

RegularReward

Punishment

<<Enum>> Direction
- + UP
- + DOWN
- + LEFT
- + RIGHT

driver

Board
- - dimension: int
- - startSpace: Position
- - endSpace: Position
- - barriers: Set<Position>
- - player: Player
- - enemies: List<Enemy>
- - objectives: List<RegularReward>
- - punishment: List<Punishments>
- - bonuses: List<BonusReward>
- + generateBoardInstance()
- + tickBoardState(input: Direction)
- - movePlayer(input: Direction)
- - moveEnemies()
- - isTheGameOver()
- - checkIfPlayerWin()
- - collectObjectives()
- - collectBonusRewards()
- - collectPunishments()
- - randomizeBonusRewards()
- - validMove()

Classes will be kept arbiturary.

The actual implementation of class will be further extended into real sprites for the game.

i.e. The player will be extended into a Cow class which will run around eating Grass (RegularReward).

<<import>>

<<imports>>

Use