Entities & Responsibilities

GameWorld: Manages game initialisation & admin tasks (e.g game config, game setup, main game loop, player turn.)

Tile: Represents the tiles which the dragons will interact with (stand on)

DrawProperties: Data class for organising the drawing properties required for drawing any element

PlayableEntity: Represents the playable entity a player interacts with

GameBoard: Represents the game board. It initialises the game board & runs the interactions with the game board by the players (e.g performing movement, flipping chit card)

ChitCard: Represents the chit cards and their effects

EventBus: Handles registration of listeners, and notification of appropriate listeners on event fire

WinEventXxxx....: Publisher and listeners for win event

MoveActionXxxx....: Publisher and listeners for a move action (for characters) that is fired

DrawableByAsset: Indicates that the object is drawable by pygame using assets

DrawAssetInstruction: A data class for organising data required for drawing an asset

ModularClickableSprite: Allows classes to be represented as a sprite that is clickable on a screen.

Patterns Used

Observer: WinEventPublisher, WinEventListener
- Why?: Don't have to check all starting tiles to see if win occured. Allows for wins from other sources

Singleton: EventBus
- Why?: Should be one central event bus managing all events

Todo

Cardinalities

Starting tiles must be winning tiles for DefaultGameBoard (need to make the typing more strong, extra class inheriting from Tile [WinnableTile])

Notes

Upcasts are safe
Circular dependencies = too many responsibilities
https://softwareengineering.stackexchange.com/questions/306483/how-to-solve-circular-dependency
Java supports circular dependencies
https://www.reddit.com/r/ProgrammingLanguages/comments/yvkysh/languages_which_support_circular_dependency/