

DefaultGameBoard._init_: First parameter is set main tile sequence. Second parameter is [starting tile, next tile]. The second parameter (starting tile) will be used to make insertions to main tile sequence property.

DefaultGameBoard.starting_tiles: [starting tiles]

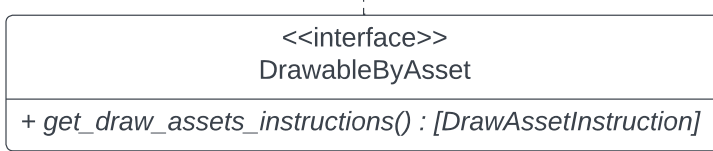
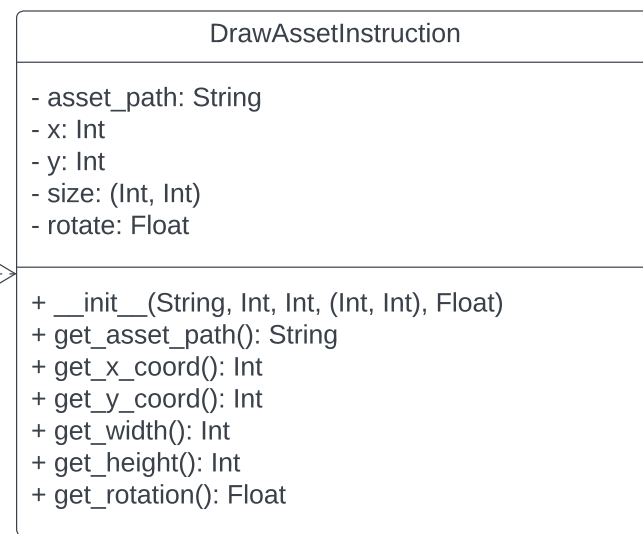
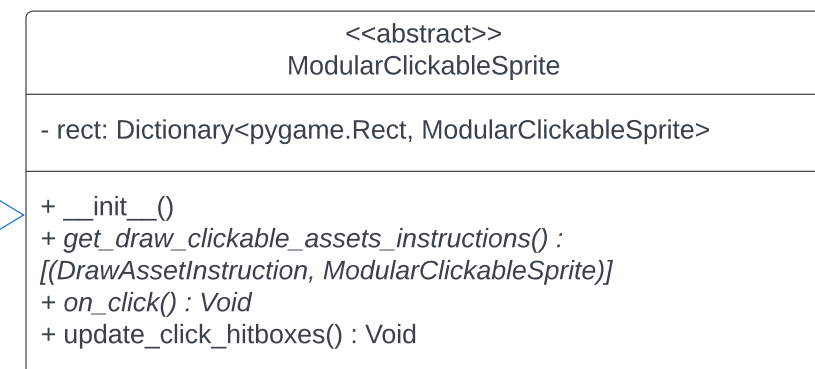
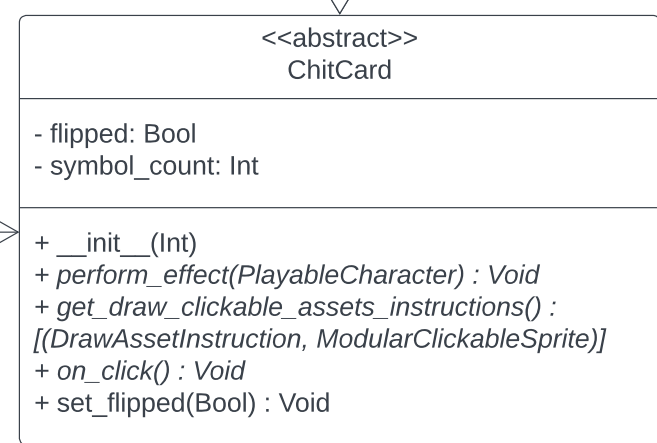
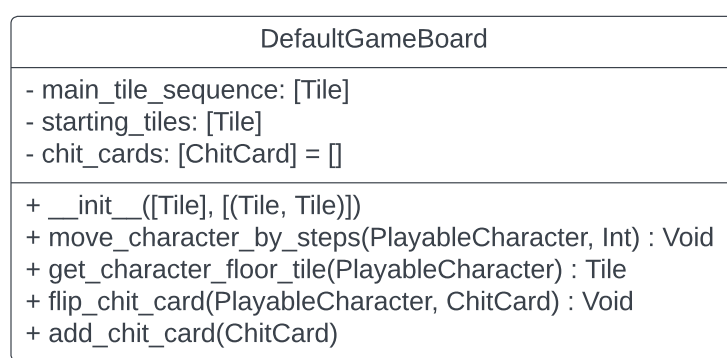
DefaultGameBoard.move_character_by_steps: Moves a dragon from its tile by x number of steps

DefaultGameBoard.flip_chit_card: Flip a chit card on this game board. Chit card object will be obtained by click listener in pygame.

DefaultGameBoard.add_chit_card: Add a chit card to the game board

ChitCard.__init__: Create a chit card with the # number of symbols
ChitCard.perform_effect: Perform the chit card's effect on the dragon.
Run by flipChitCard
ChiCard.set_flipped: Set flipped

ModularClickableSprite.__init__: Run update click hitboxes
 ModularClickableSprite.rect: Dictionary mapping click hit boxes to the object associated with it
 ModularClickableSprite.on_click: Do something to the object on click
 ModularClickableSprite.update_click_hitboxes: Update click hitboxes based on draw instructions (get_draw_clickable_assets_instructions)



`DrawableByAsset.get_draw_assets_instructions`: Gets the drawing instructions for drawing an object onto the pygame screen using assets

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)

PlayableEntity: Represents the playable entity a player interacts with

GameBoard: 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 occurred
Allows for wins from other sources

Singleton: EventBus

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

Todo

Cardinalities

Notes

Upcasts are safe

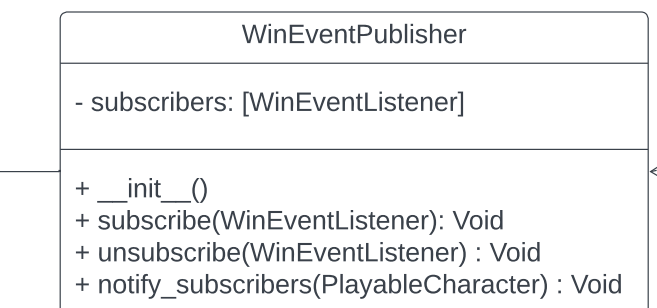
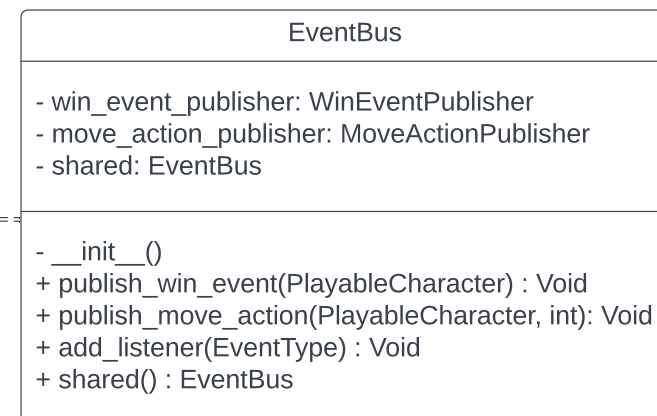
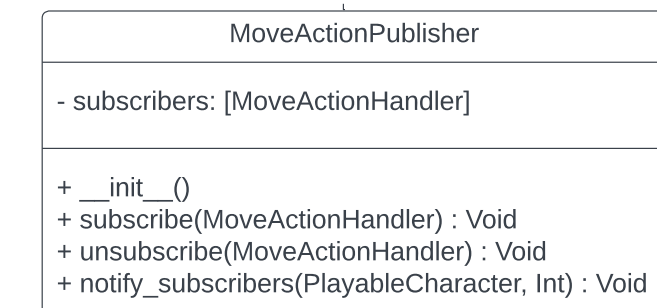
Circular dependencies = too many responsibilities

<https://softwareengineering.stackexchange.com/questions/306483/how-to-solve-circular-dependency>

Java supports circular dependencies

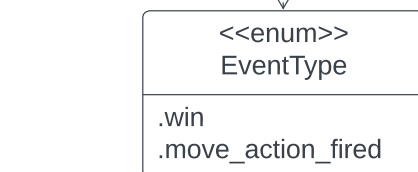
https://www.reddit.com/r/ProgrammingLanguages/comments/yvkysh/languages_which_support_circular_dependency/

MoveActionPublisher.notifySubscribers: Notify subscribers about a move action that needs to be handled. Character and steps to take passed in



WinEventListener.onPlayerWin: On a player win, do something

WinEventPublisher.notifySubscribers: Notify subscribers about the player who won



- EventBus.shared: Get the eventbus singleton
- EventBus.publishWinEvent: Publish win event onto bus
- EventBus.addListener: Add a listener for a certain event type

Dragon.take_turn: Allows a dragon to take its turn & perform actions (e.g. hit card flipping)

CaveTile.place_character: Place dragon onto tile and perform check whether it was return back to own cave, trigger win