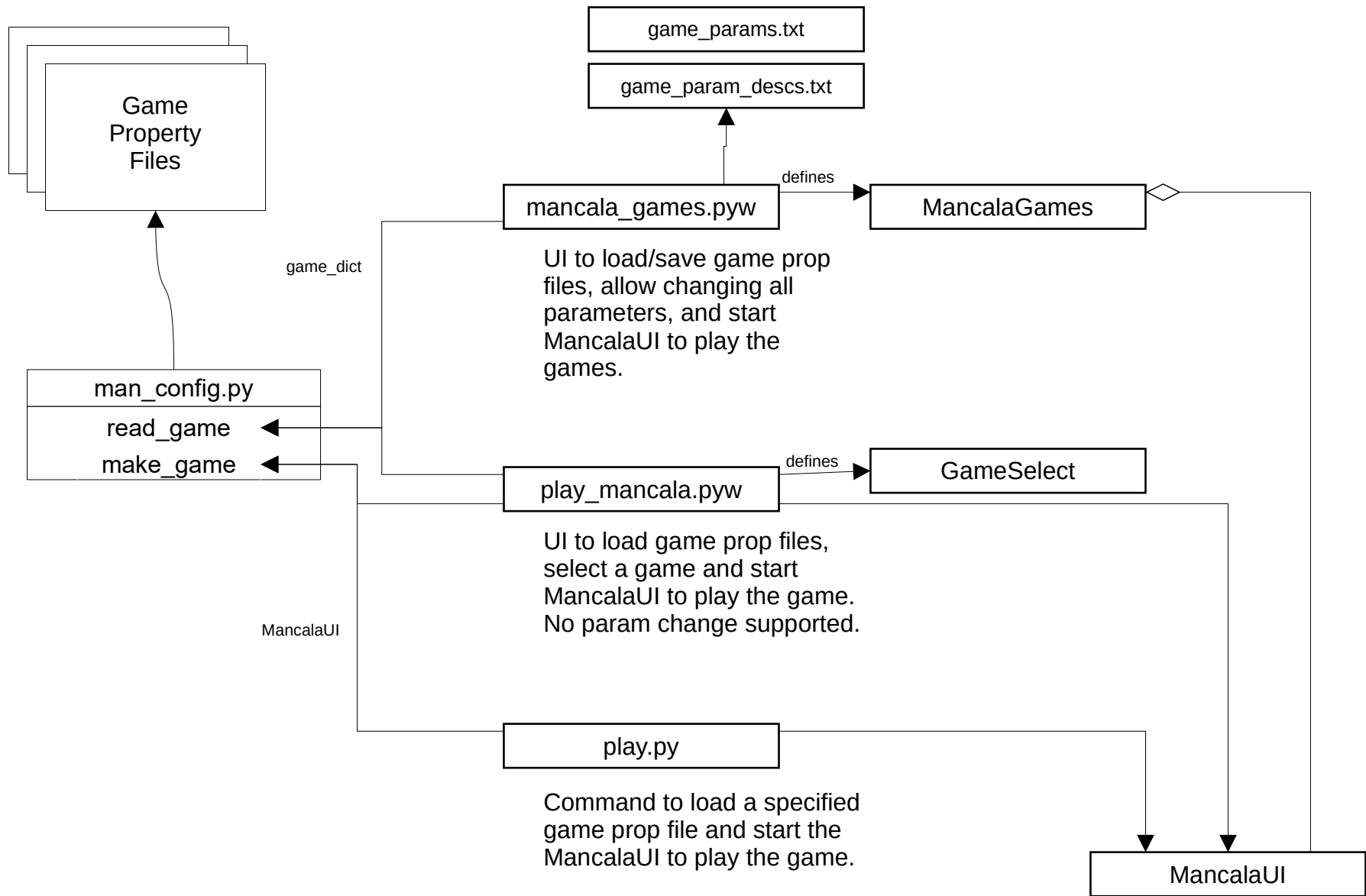
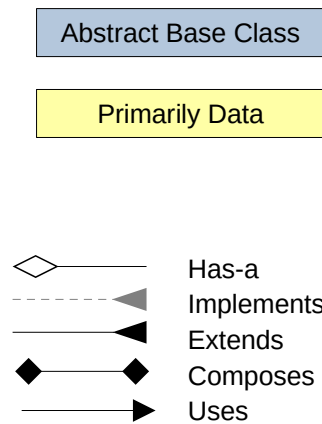


# Mancala Games



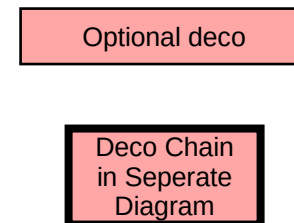
# Notation Conventions

## Class Diagram Conventions



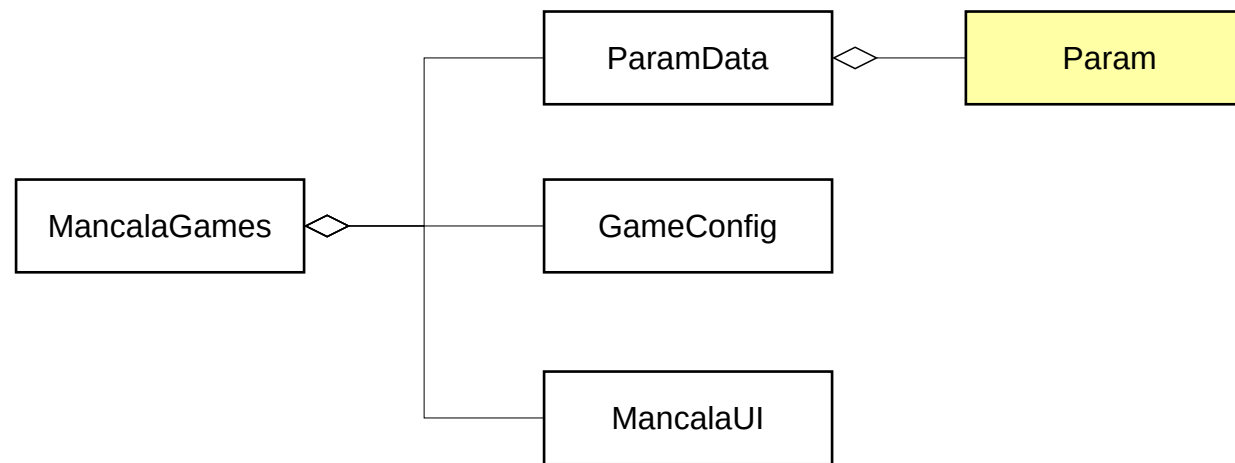
## Deco Chain Conventions

- One path down the deco chain is used.
- Intersecting arrows are decision points.
- Shown in **call order** from start dot (constructed in reverse order).
- Calls down the deco chain maybe at any point in each deco's processing.
- All paths shown might not be possible (see ginfo\_rules).



# MancalaGames

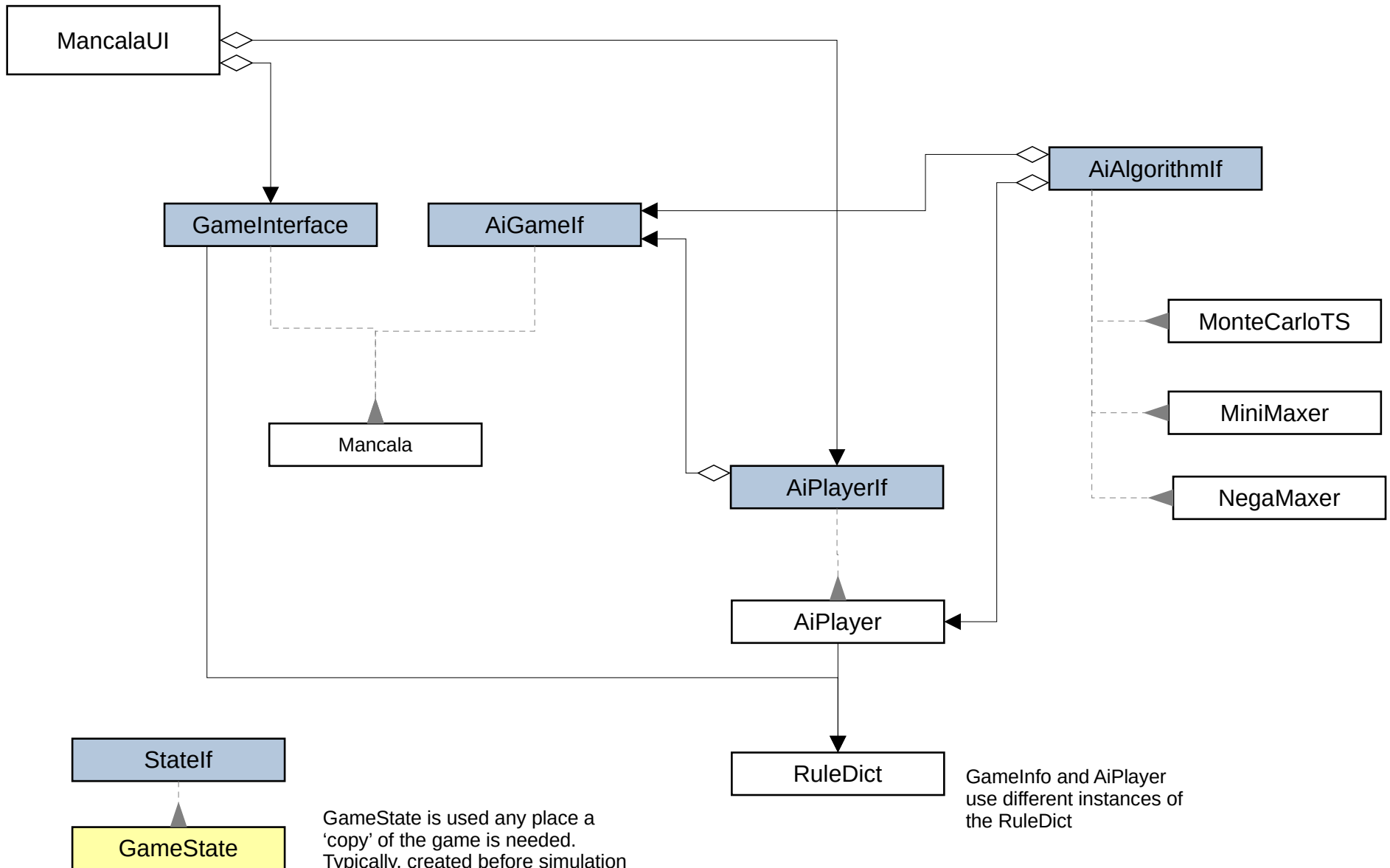
(the Mancala Games UI class)



game\_params.txt and game\_param\_descs.txt are loaded into the ParamData class.

Management of the game configuration file is done with the GameConfig class in MancalaUI

# Mancala, GameState, AIPlayer and AIAlgorithm

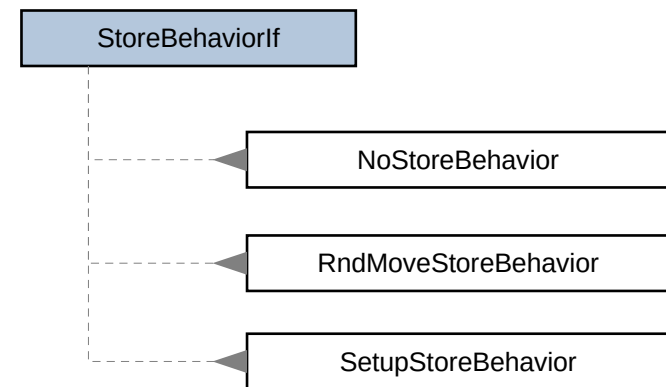
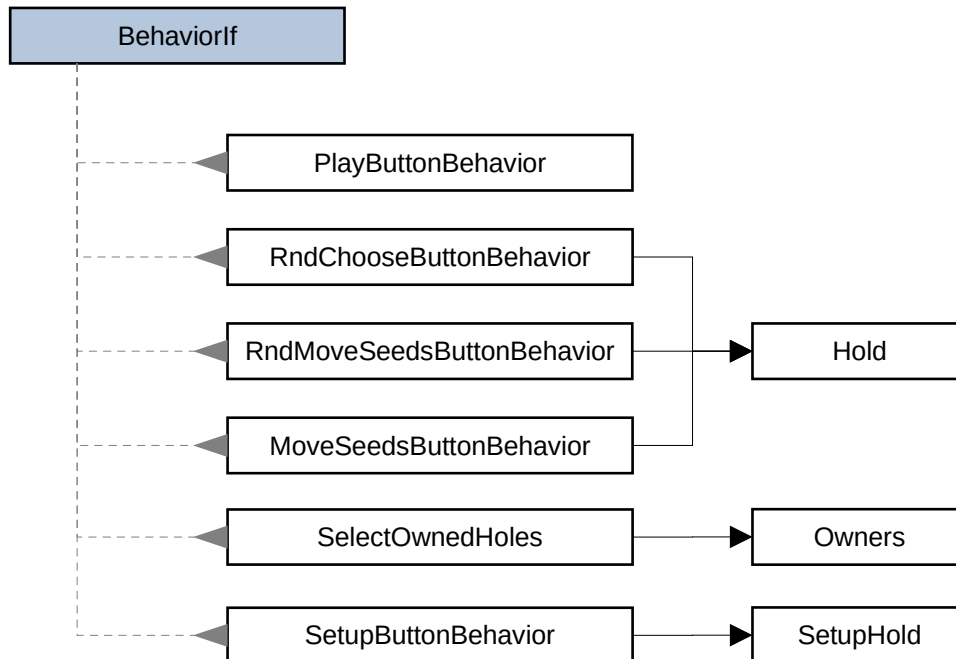
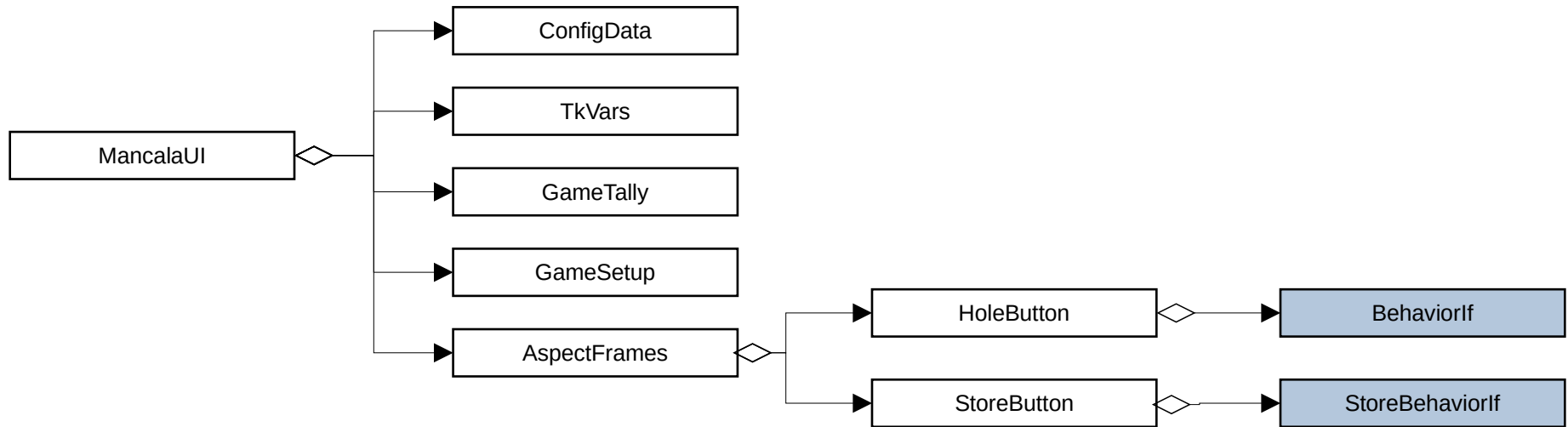


GameState is used any place a 'copy' of the game is needed. Typically, created before simulation and restored after.

Created and set via the state property of the Mancala class.

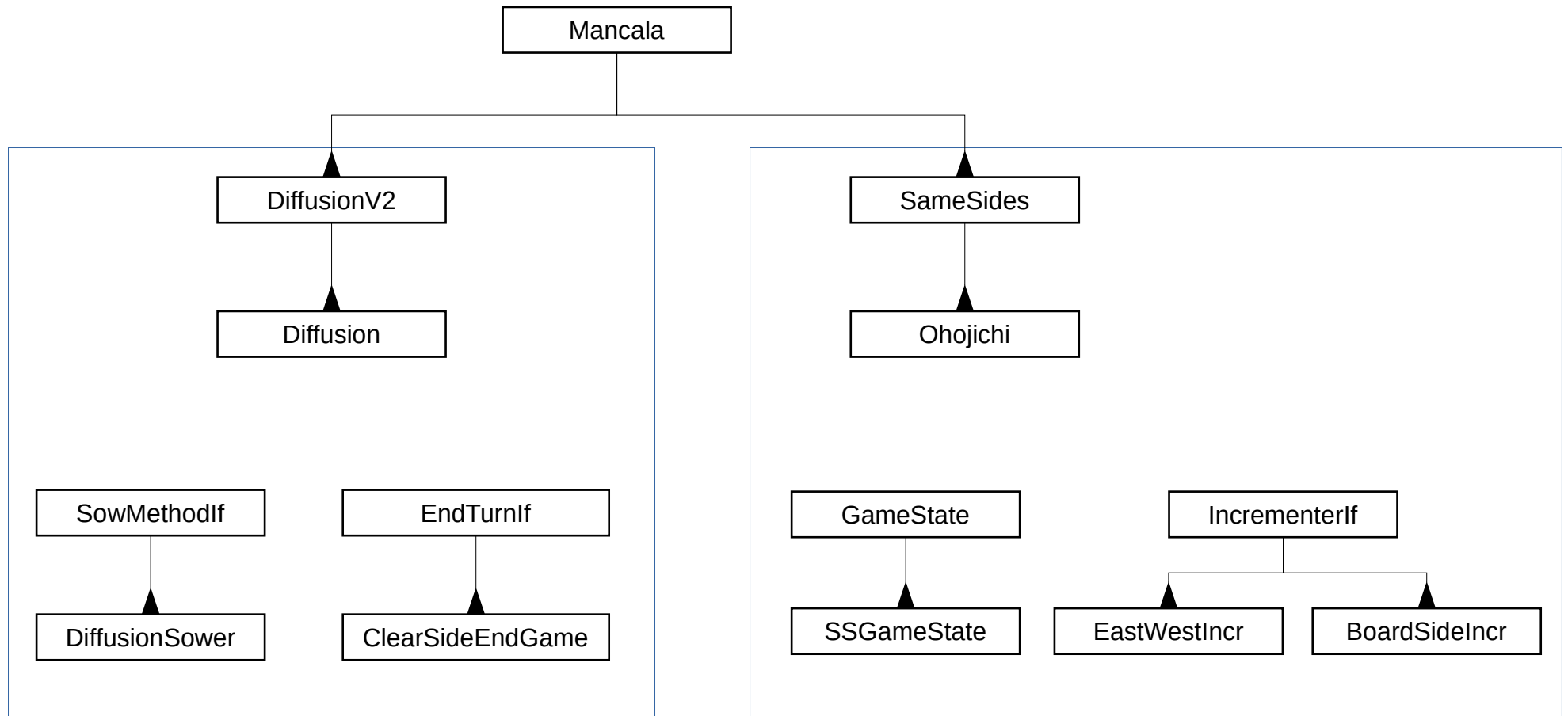
GameInfo and AiPlayer use different instances of the RuleDict

# Mancala UI Classes

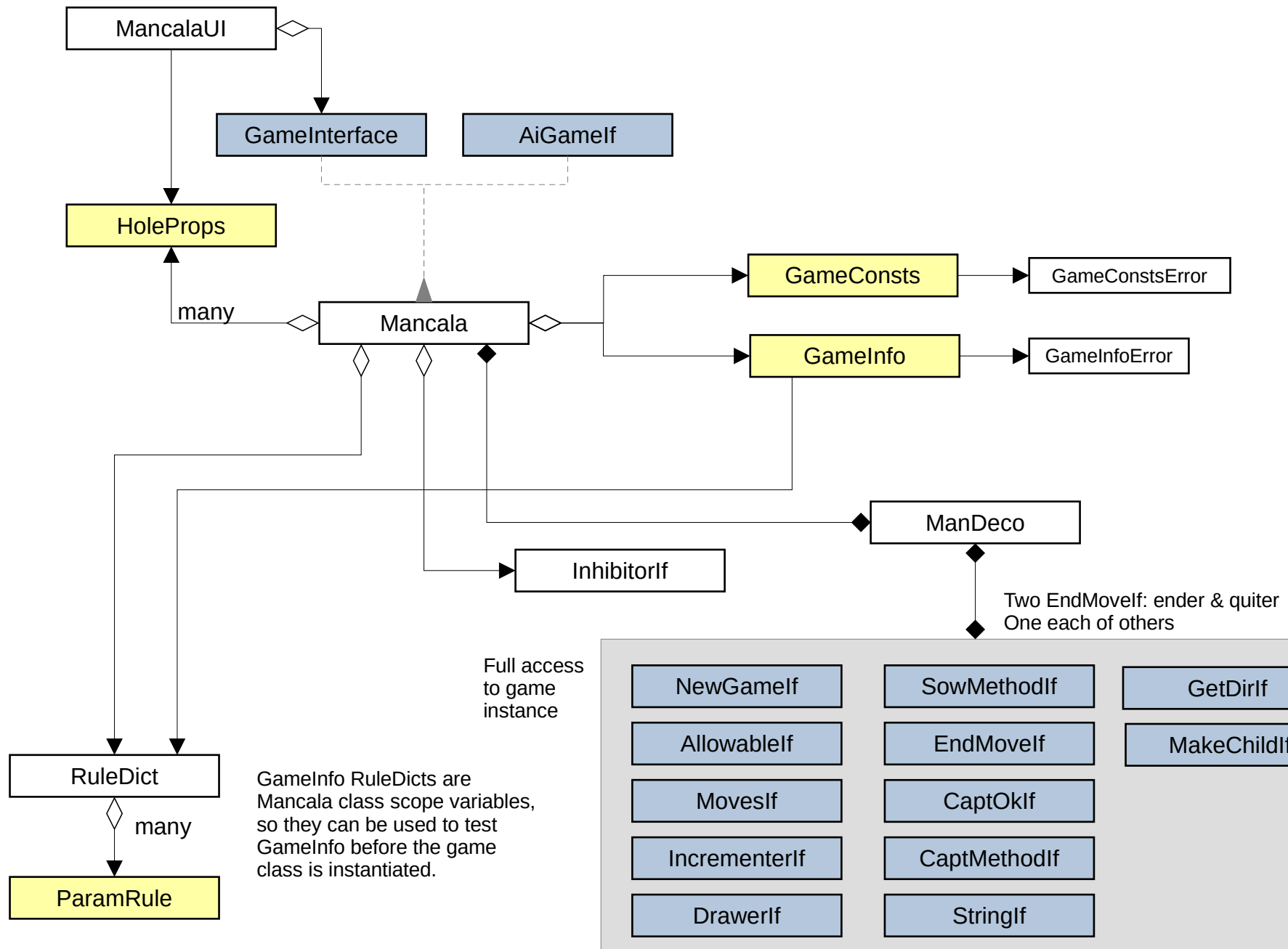


Hold, Owners and SetupHold are used to collect a few global variables and operations. One global instance of each is created.

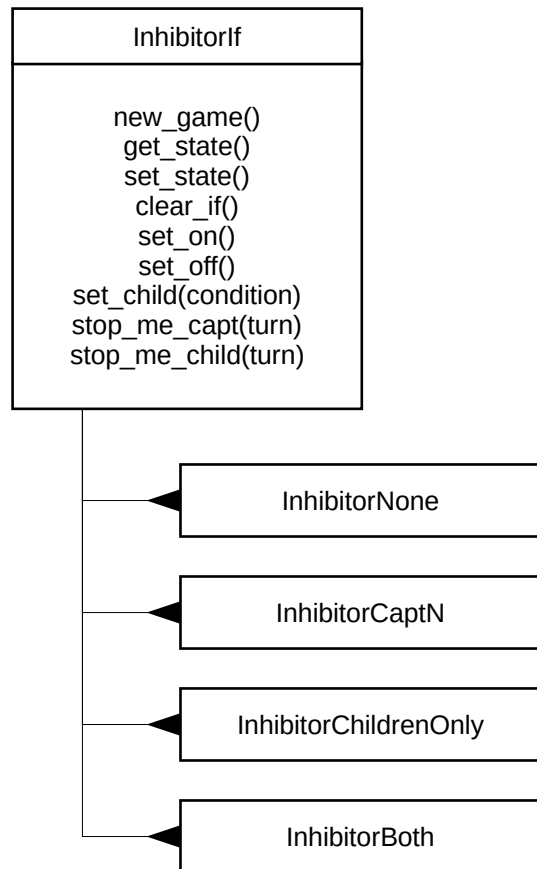
# Mancala Classes



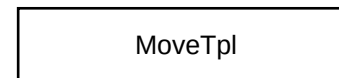
# Mancala Classes



# Import Classes for Moves



The decorator chains and button behaviors use and control the inhibitor.



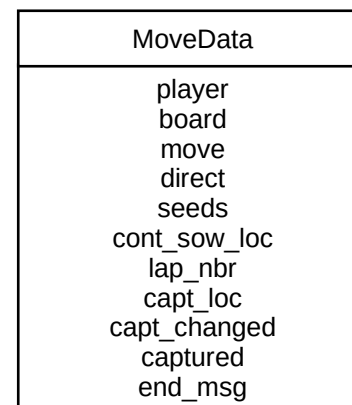
Moves are one of (based on game parameters):

1. position
2. (position, direction)
3. (row, position, direction)

MoveTpl prints the moves nicely.

Row is in terms of the UI, that is Top/True is 0 and Bottom/False is 1. This is the “not” of the game.turn.

Moves are created when initializing the HoleButtons for the human players and via the `get_moves` deco chain for the AI player.



MoveData is used to communicate information about each move between the deco chains and individual decorators.

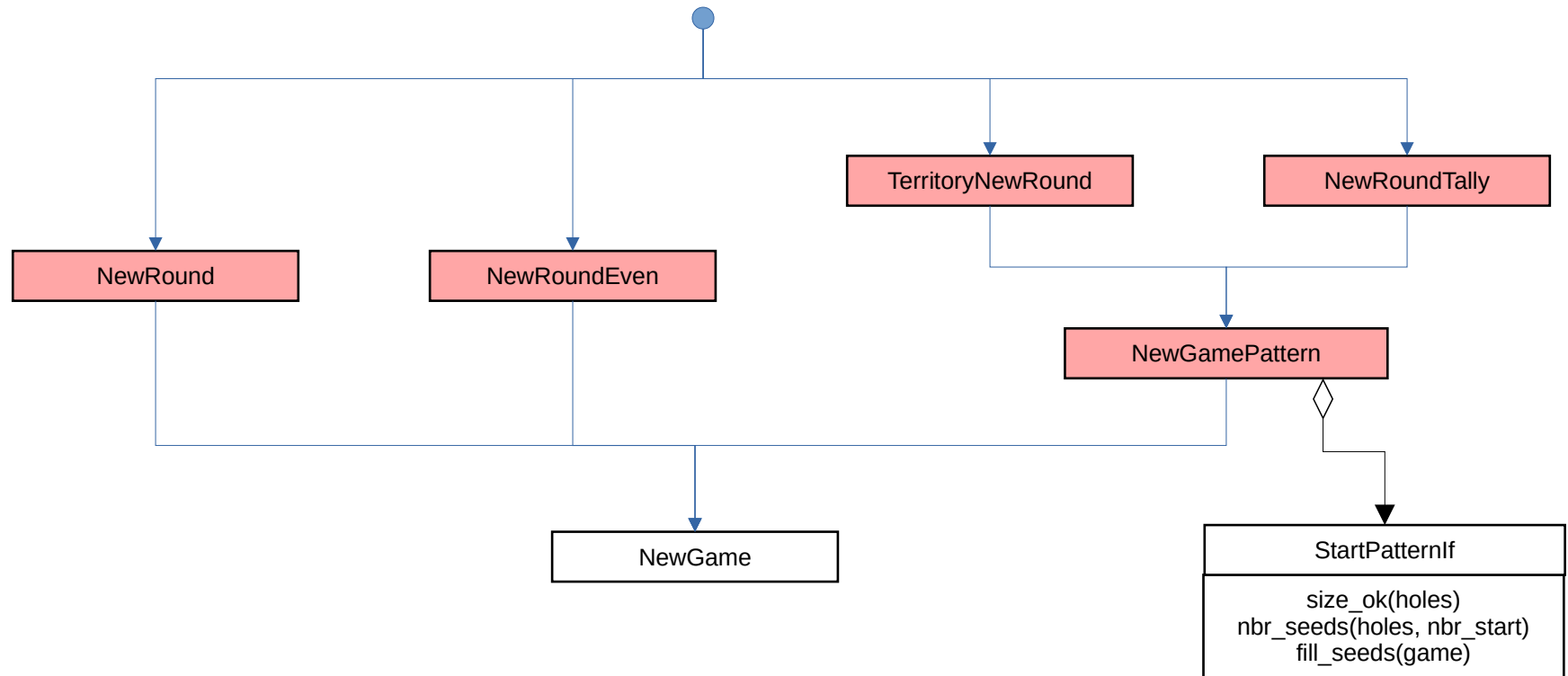
See class comment for where each field is set and/or updated.



# Decorator Usage

Game Op/Step	Primary Decorator	Other Classes & Decorators Used	Description
New Game	new_game	StartPattern, inhibitor	Setups the game for initial play. Applies any prescribed moves.
Determine Drawable Holes	allow		Return a list of holes that are playable.
Collect Moves	get_moves		Return a list of possible moves.
Draw seeds to start a move	drawer		Parse the move, determine number of seeds to sow, possibly leave one seed
Determine sow direction	get_direction		Convert the move & location into an actual sowable direction: clockwise or counter-clockwise.
Sow	sower	MoveData, incr, make_child, inhibitor	Drop the seeds into the board holes.
Capture seeds	capturer & capt_ok	MoveData, incr, make_child, inhibitor	Perform any captures.
Evaluate end of game	ender	MoveData	At the end of each move determine if the game is over: game has been won, no more moves, game outcome can't change, etc.
Logging	get_string		Creates an ASCII string for the game.
Force end of game	quitter		The game needs to end either because of endless sow or user selection. Something fair will be done.

# New Game Decorators and Chain



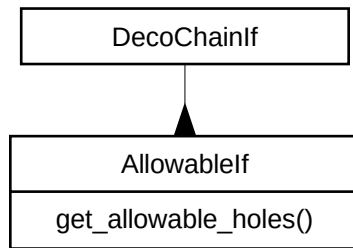
State variables changed:

blocked  
board  
owner  
starter  
store  
turn

Parameters:

blocks  
goal  
min\_move  
round\_starter  
round\_fill  
rounds  
start\_pattern

# Allowables Decorators and Chain

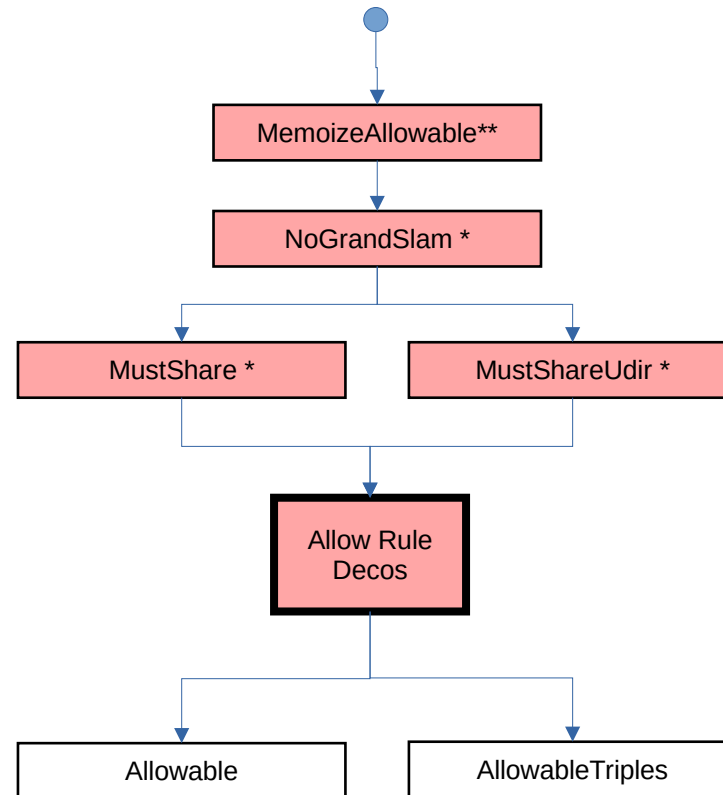


State variables read:

turn  
board  
store  
blocked  
owner  
child  
mcount

Parameters:

min\_move  
allow\_rule  
mlength  
mustshare  
grandslam  
udir\_holes

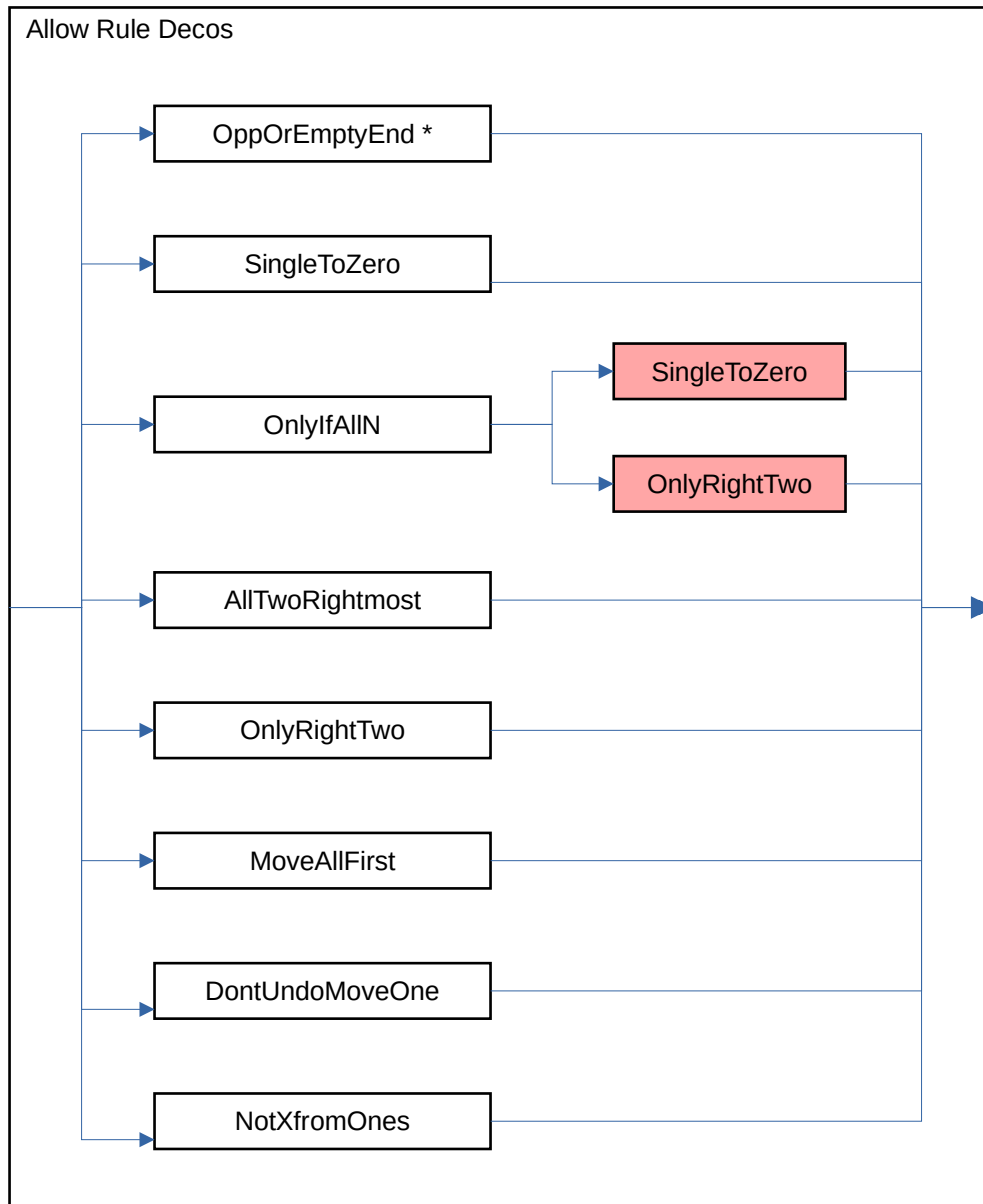


Notes:

\* Simulates some portion of moves to determine allowables

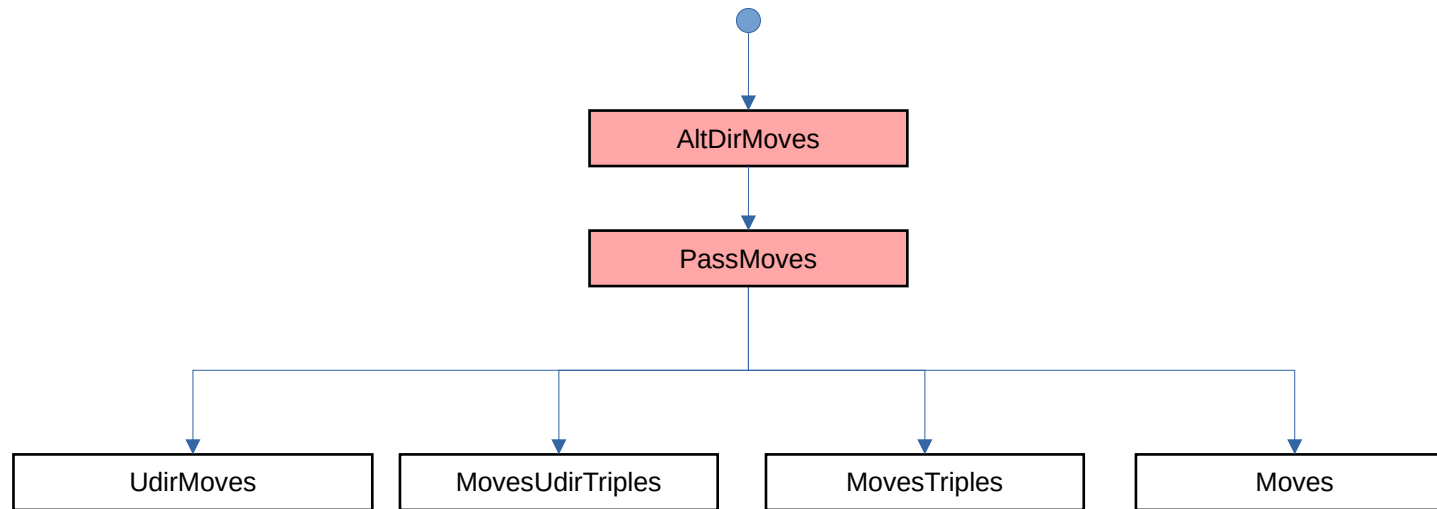
\*\* MemoizeAllowable is used for deco's that simulate moves

# Allow Rule Decos



Notes:  
Some allow rule decos are shown more than once  
for clarity.  
\* Simulates some portion of moves to determine  
allowables

# Get Moves Decorators and Chain

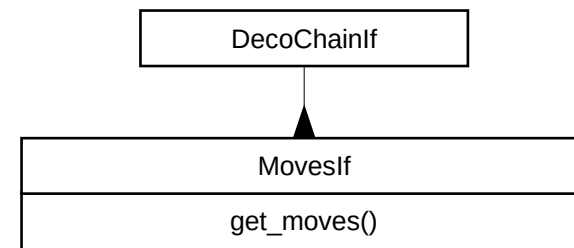


State variables read:

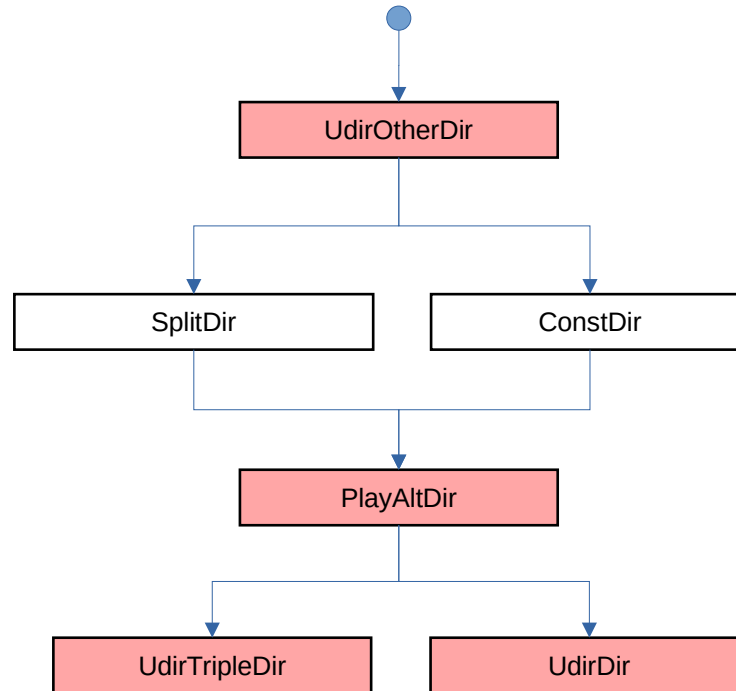
blocked  
board  
owner  
starter  
store  
turn

Parameters:

mlength  
mustpass  
sow\_direct  
udir\_holes  
udirect

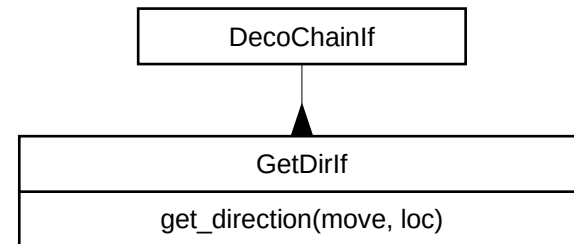


# Get Direction Decorators and Chain

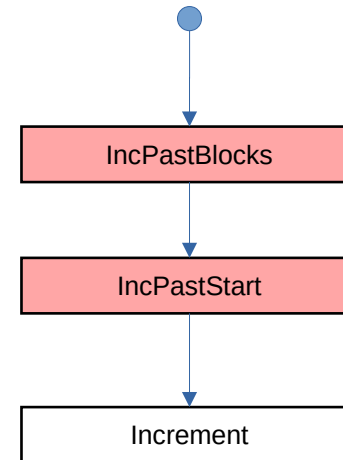
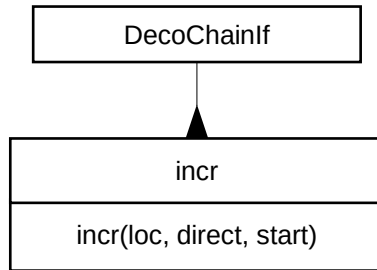


State variables read:  
mcount  
turn

Parameters:  
no\_sides  
sow\_direct  
udir\_holes  
udirect



# Incrementer Decorators and Chains



State variables read:  
blocked

Parameters:  
blocks  
skip\_start

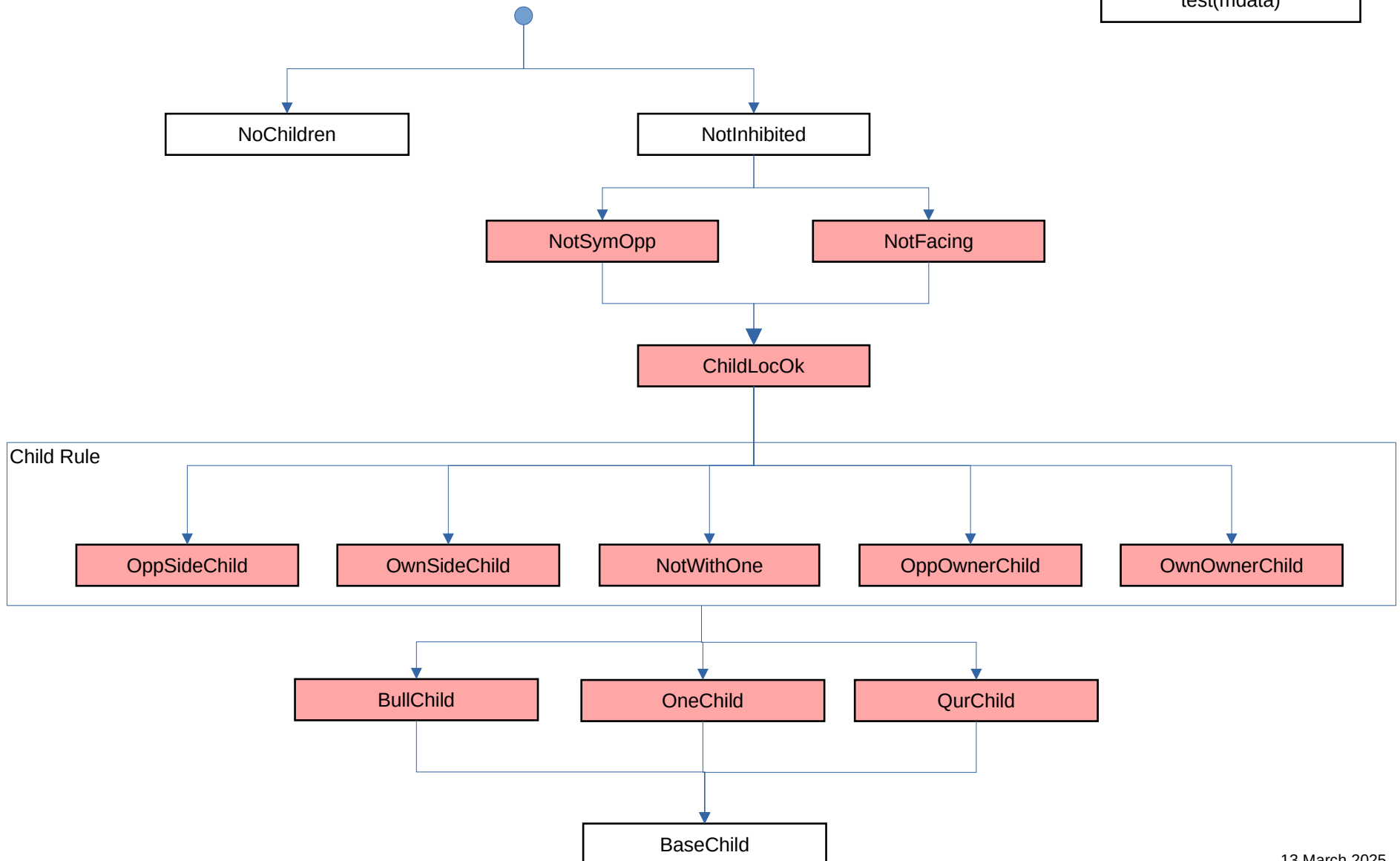
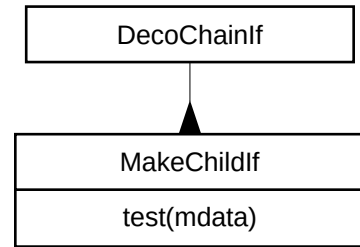
# MakeChild Decorator and Chain

State variables read:

board  
child  
inhibitor  
owner  
turn

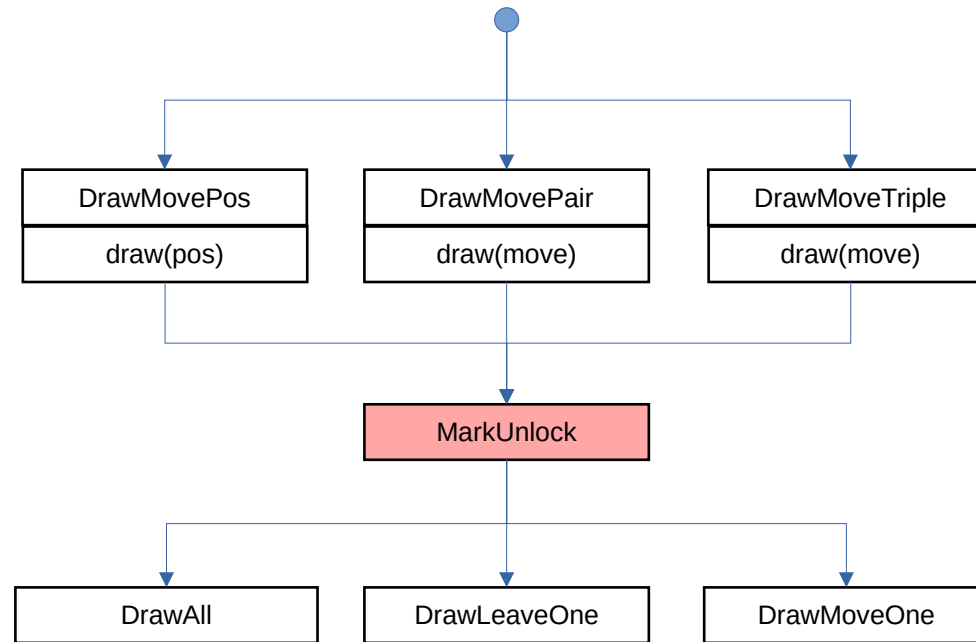
Parameters:

child\_cvt  
child\_locs  
child\_rule  
child\_type





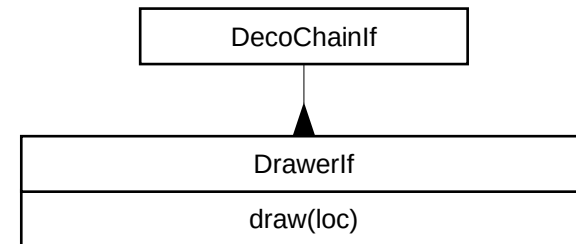
# Draw Decorators and Chain



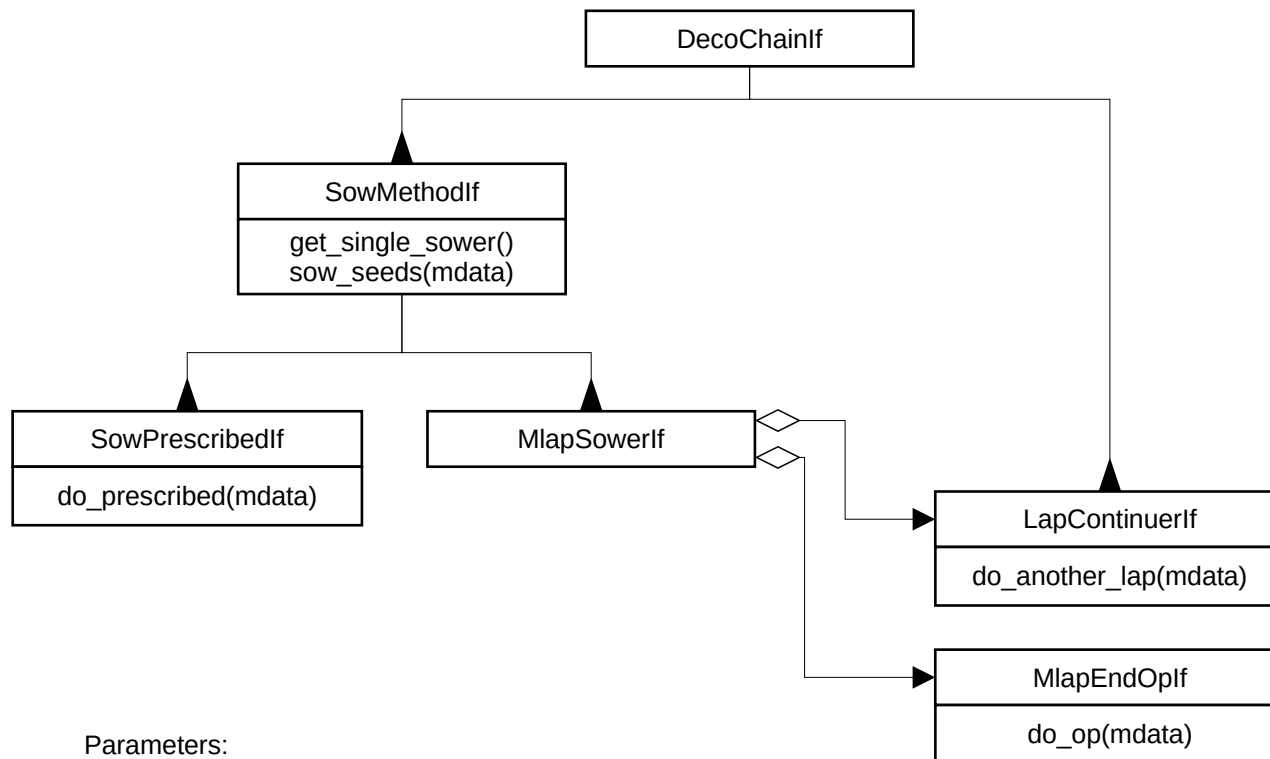
The first drawer converts the move into board location.

State variables:  
Read:  
    turn  
Changed:  
    board  
    unlocked

Parameters:  
allow\_rule  
mlength  
move\_one  
moveunlock  
sow\_start



# Sower Decorators



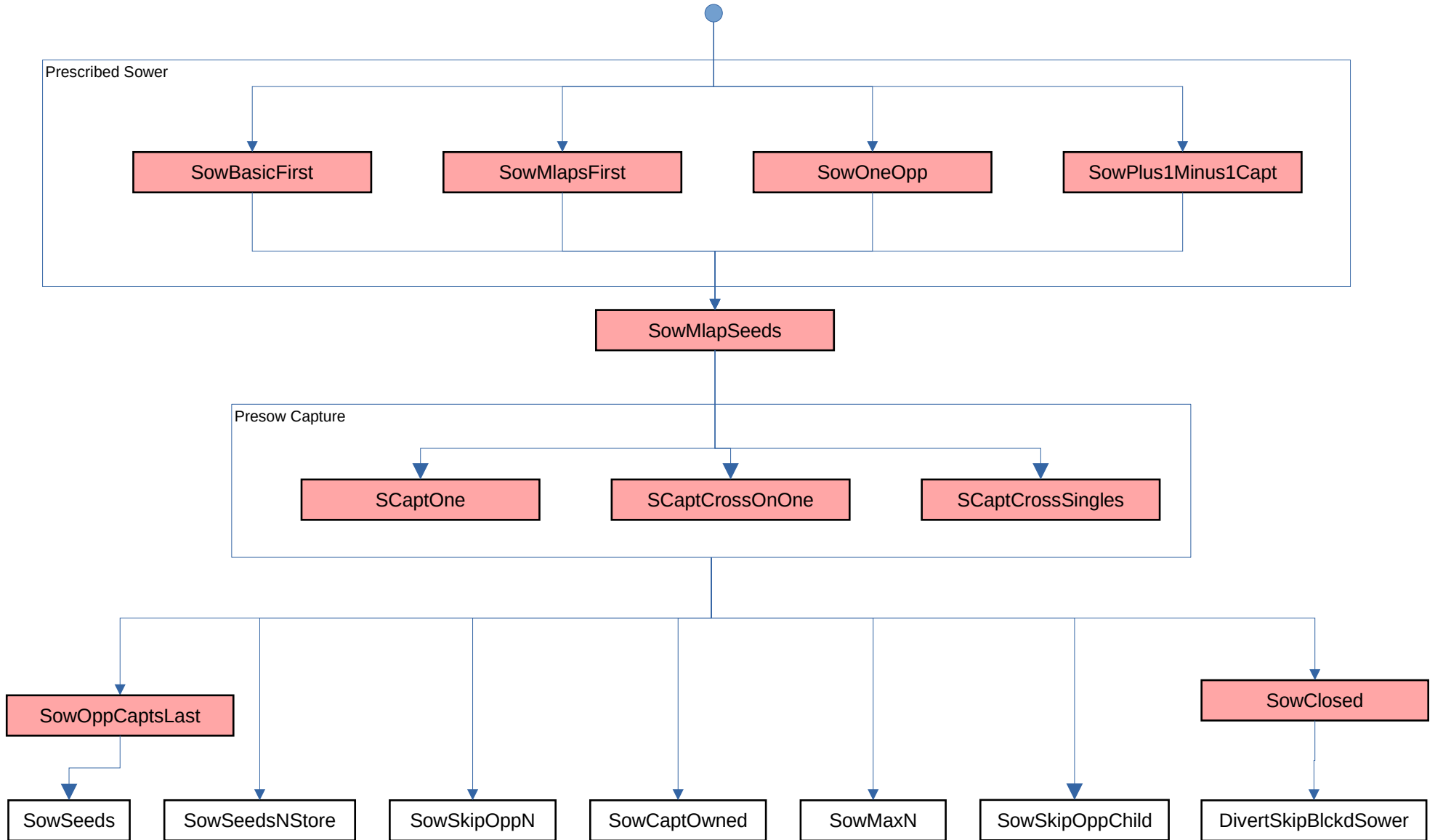
State variables:

Reads  
inhibitor  
turn  
child  
mcount  
Changes  
board  
store  
blocked

Parameters:

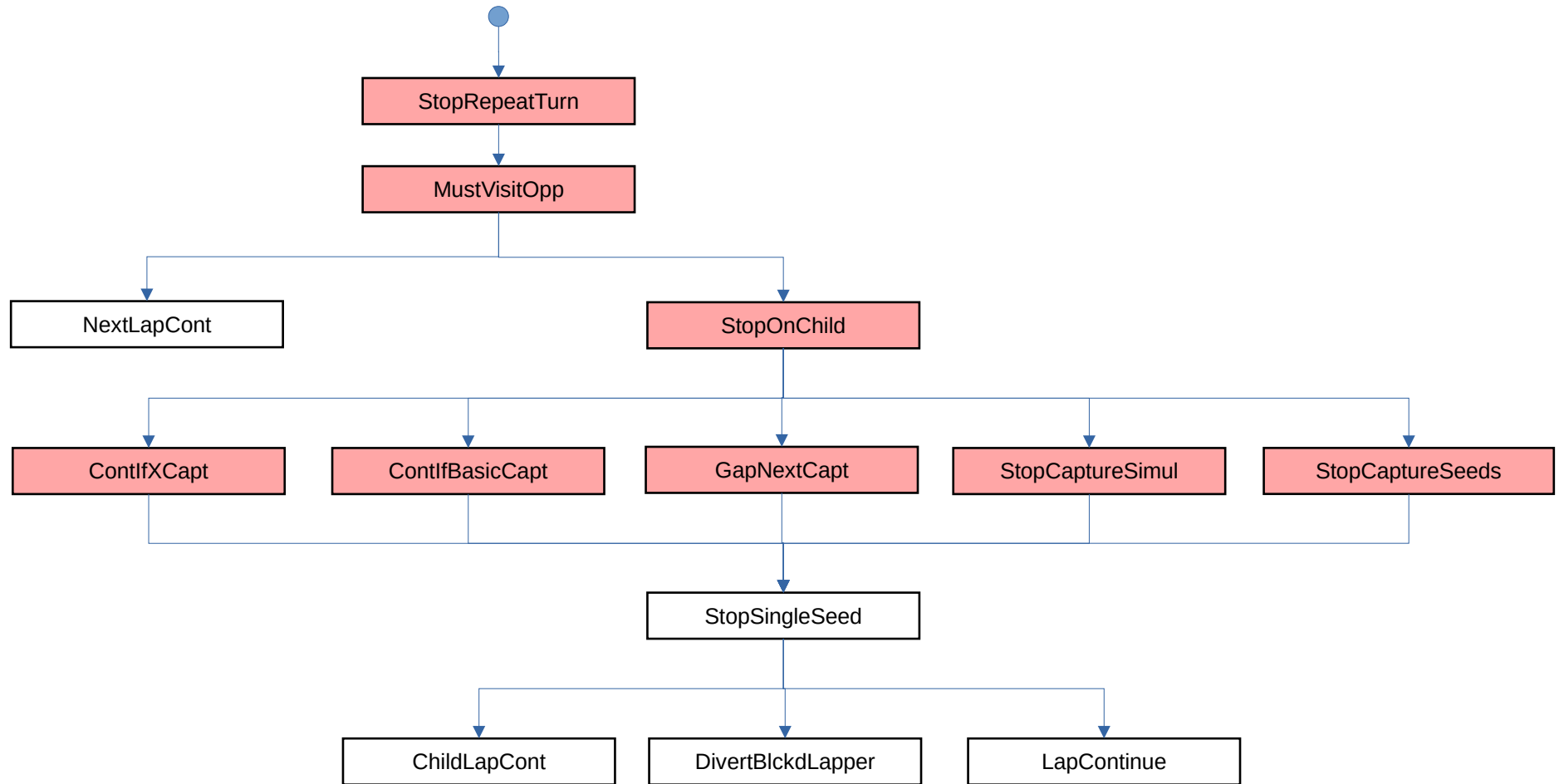
capt\_max  
capt\_min  
capt\_on  
child\_type  
crosscapt  
evens  
goal  
gparam\_one  
mlaps  
prescribed  
presowcapt  
sow\_direct  
sow\_own\_store  
sow\_param  
sow\_rule  
visit\_opp

# Sower Deco Chain



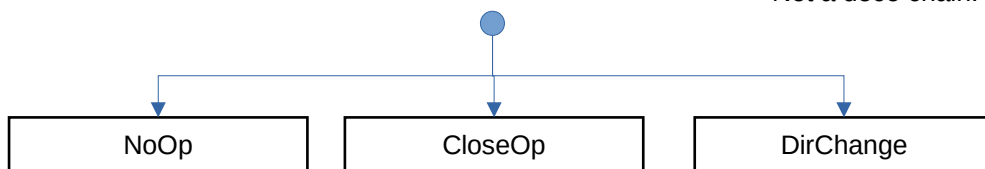
# Lap Continuer Deco Chain and Mlap Operation

Lapper

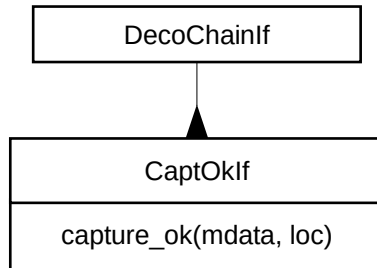


Mlap Op

Not a deco chain.



# Capt Ok Decorators and Chains

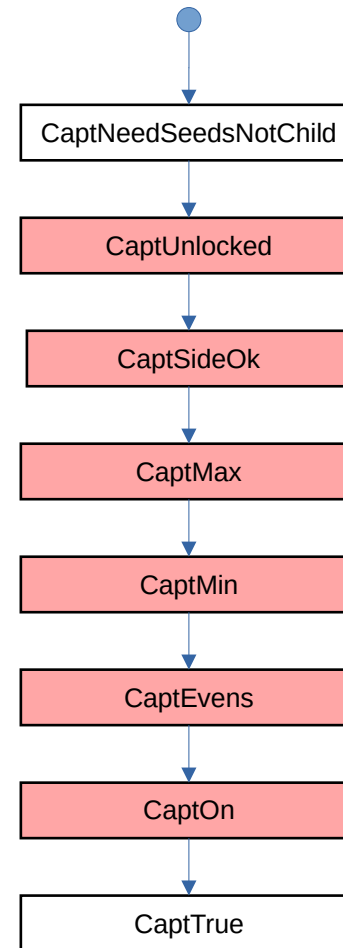


State variables read:

board  
child  
turn  
unlocked

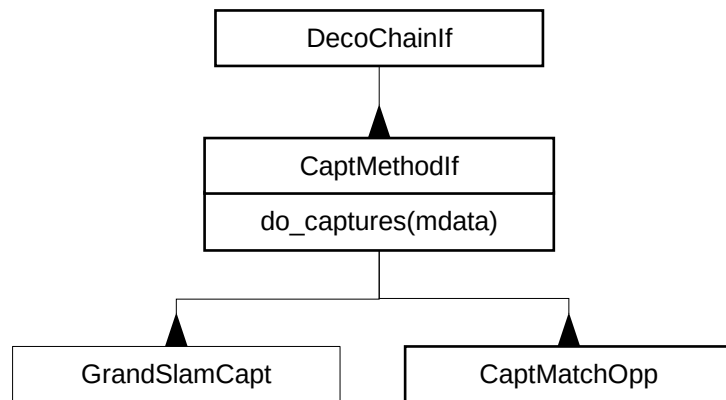
Parameters:

capt\_max  
capt\_min  
capt\_on  
capt\_side  
moveunlock



These are effectively ANDed.  
If any deco condition is false, it  
returns false, otherwise it calls  
down the deco chain.

# Capturer Decorators and Chain



State variables

Reads

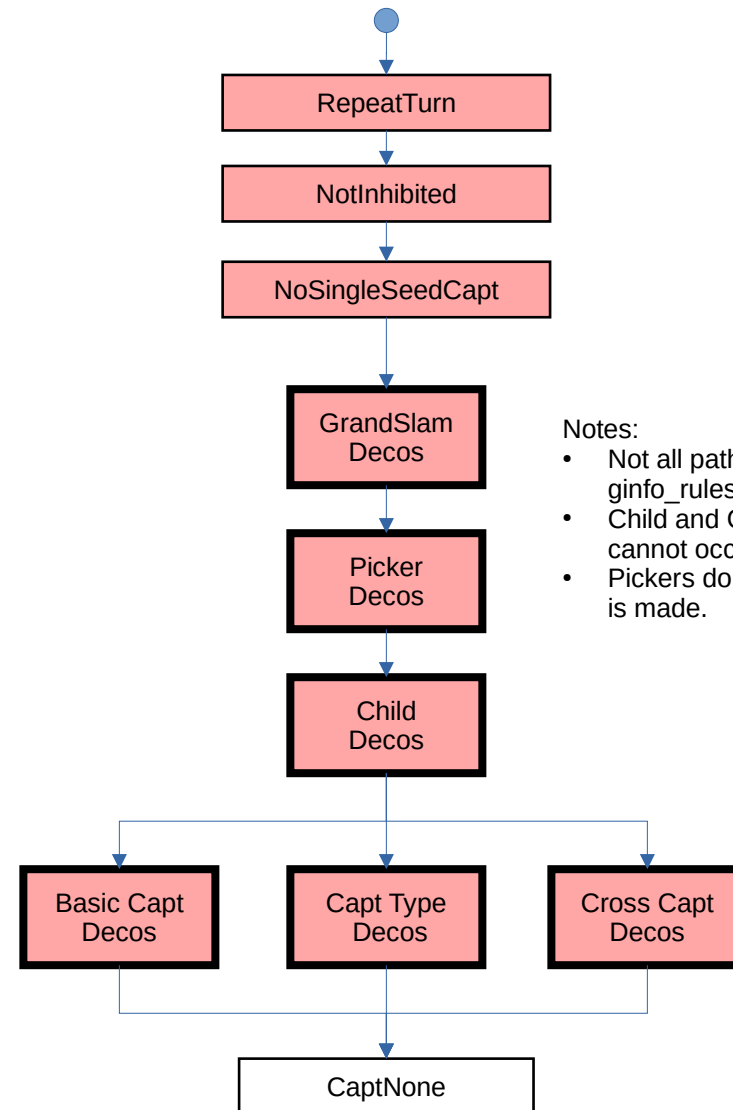
inhibitor  
starter  
turn

Changes

board  
child  
store

Parameters:

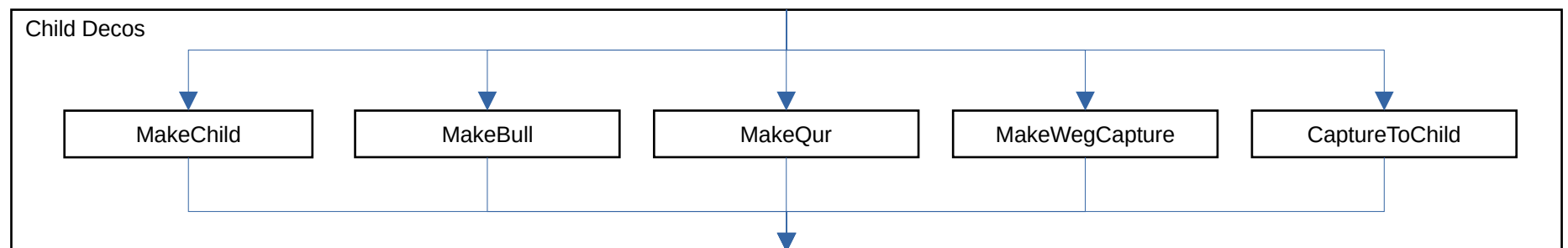
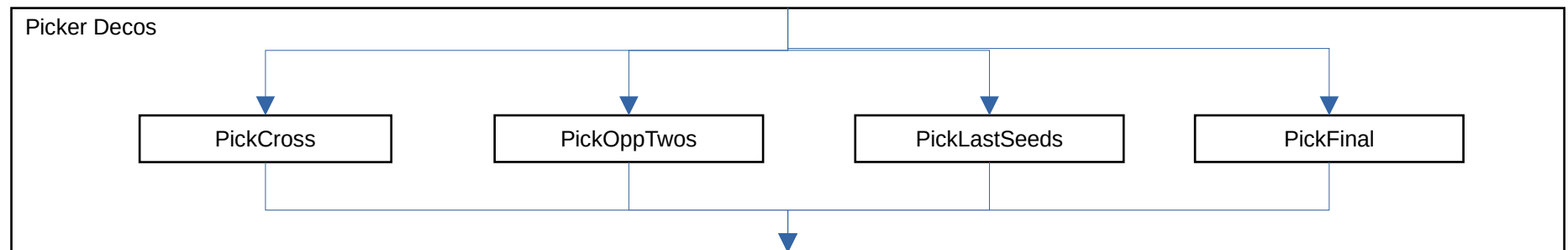
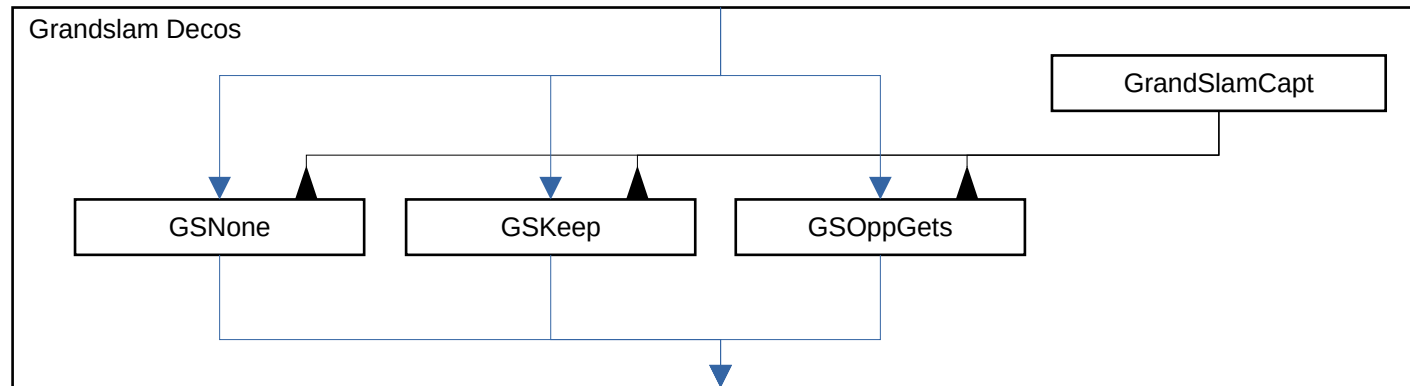
capsamedir  
capt\_max  
capt\_min  
capt\_on  
capt\_rturn  
capt\_side  
capt\_type  
child\_cvt  
child\_type  
crosscapt  
evens  
grandslam  
mlaps  
multicapt  
nocaptmoves  
nosingcapt  
pickextra  
prescribed  
round\_fill  
xc\_sown  
xcpickown



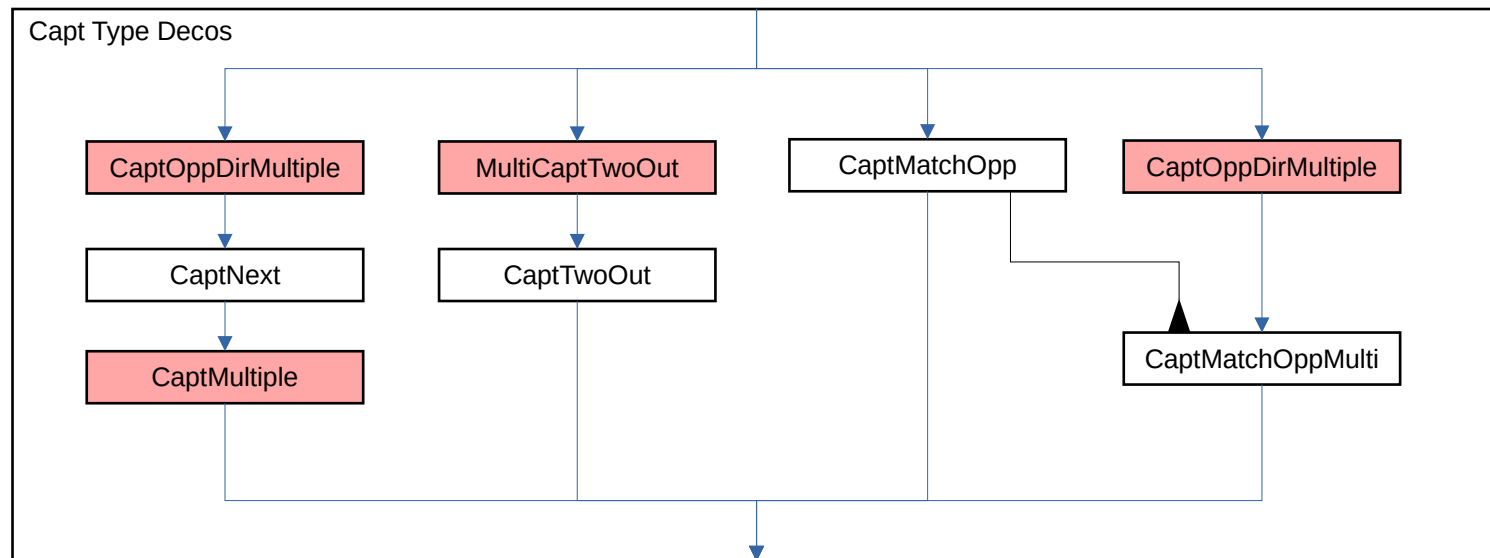
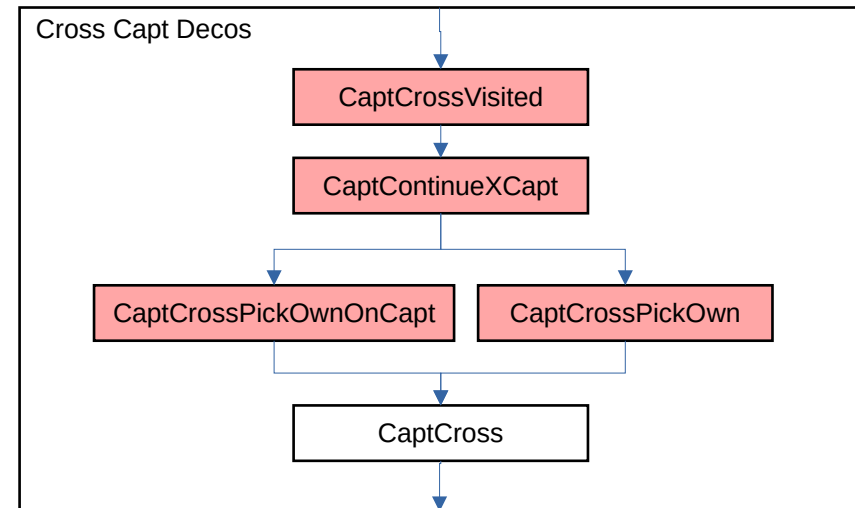
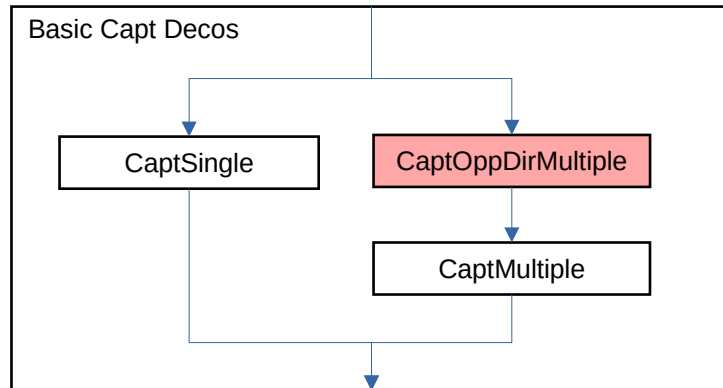
Notes:

- Not all paths are allowed: see ginfo\_rules.
- Child and Grand Slam decos cannot occur together.
- Pickers do nothing when a child is made.

# Capturer Deco Chains (1 of 2)



# Capturer Deco Chains (2 of 2)





# Ender & Quitter Decorators and Chains (1 of 2)

State variables:

Reads:

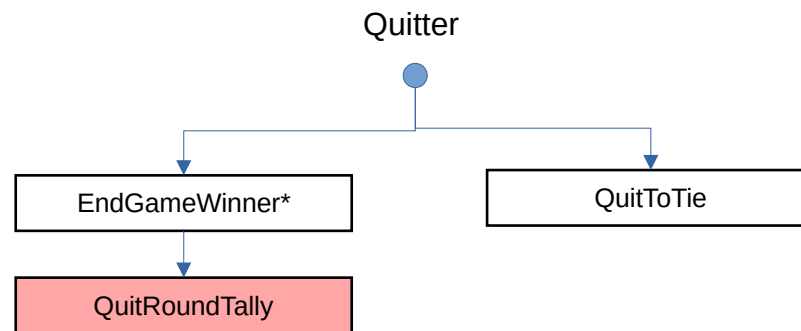
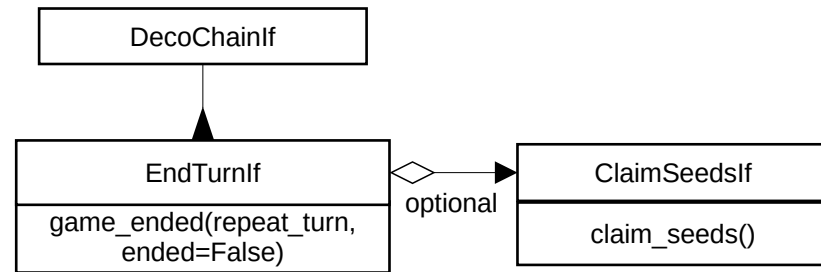
child  
owner  
turn

Changes:

board  
store

Parameters:

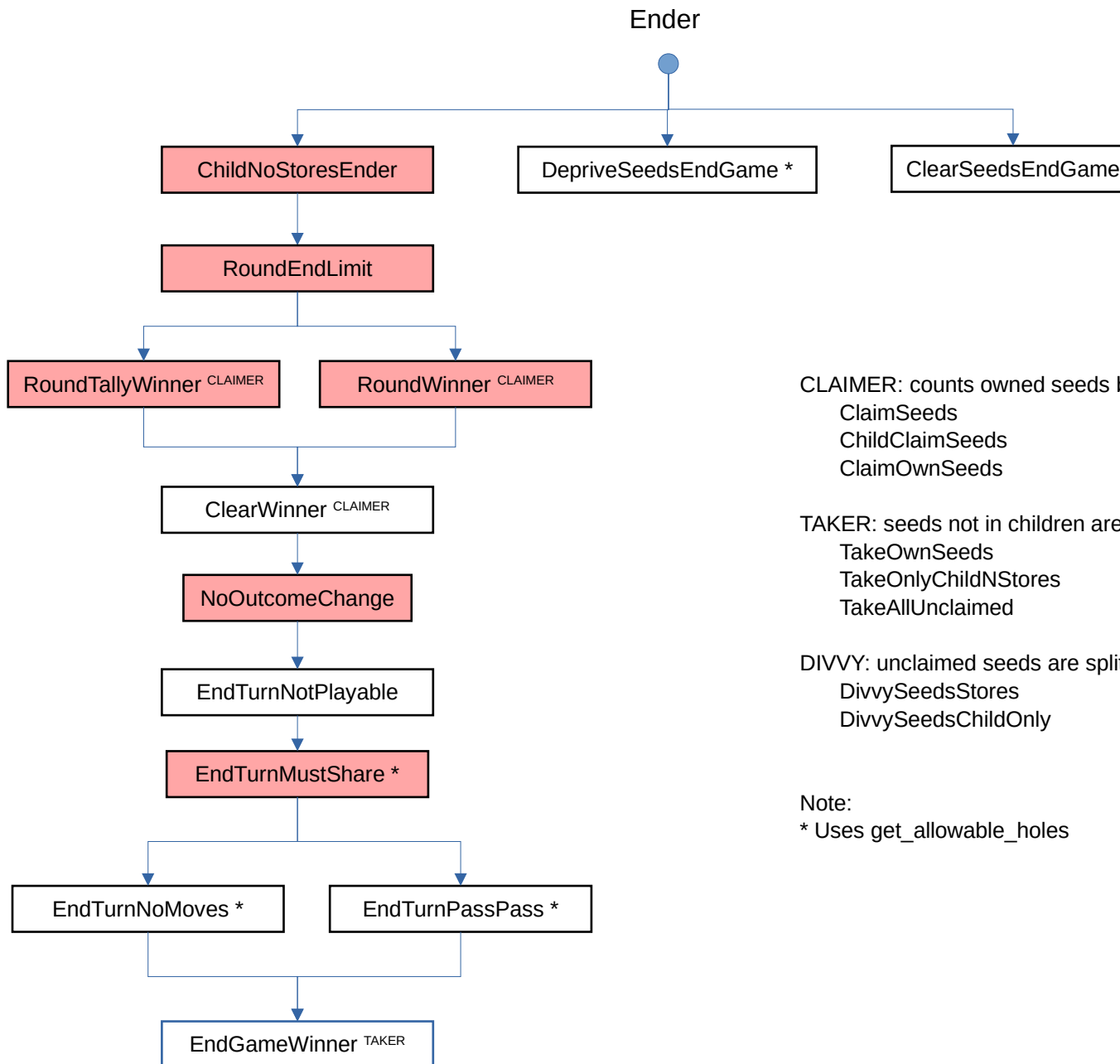
capt\_min  
capt\_next  
capt\_on  
capttwoout  
child\_cvt  
child\_type  
crosscapt  
evens  
goal  
gparam\_one  
min\_move  
mlaps  
mustpass  
mustshare  
no\_sides  
round\_fill  
rounds  
sow\_own\_store  
stores  
unclaimed



Note:

\*A claimer, taker or divvier is selected based on the unclaimed, child\_type and store properties (see next page).

# Ender & Quitter Decorators and Chains (2 of 2)



CLAIMER: counts owned seeds but does not move them:

ClaimSeeds  
ChildClaimSeeds  
ClaimOwnSeeds

TAKER: seeds not in children are claimed and moved to stores:

TakeOwnSeeds  
TakeOnlyChildNStores  
TakeAllUnclaimed

DIVVY: unclaimed seeds are split between players:

DivvySeedsStores  
DivvySeedsChildOnly

Note:

\* Uses get\_allowable\_holes