Mancala Games

Game
Property
Files

game_params.csv

game_param_descs.txt

mancala_games.pyw — defines → MancalaGames

UI to load/save game prop
files, allow changing all
parameters, and start
MancalaUI to play the
games.

game_dict

man_config.py

read_game

make_game

play_mancala.pyw — defines → GameChooser

UI to load game prop files,
select a game and start
MancalaUI to play the game.
No param change supported.

MancalaUI

play.py

Command to load a specified
game prop file and start the
MancalaUI to play the game.

MancalaUI

Notation Conventions

Class Diagram Conventions

Abstract Base Class

Primarily Data

◇———— Has-a

◇ * ———— Has-a > 1

- - - ◄ Implements

———◄ Extends

◆———◆ Composes

———► Uses
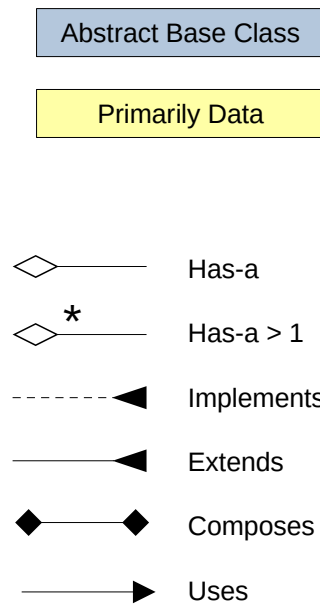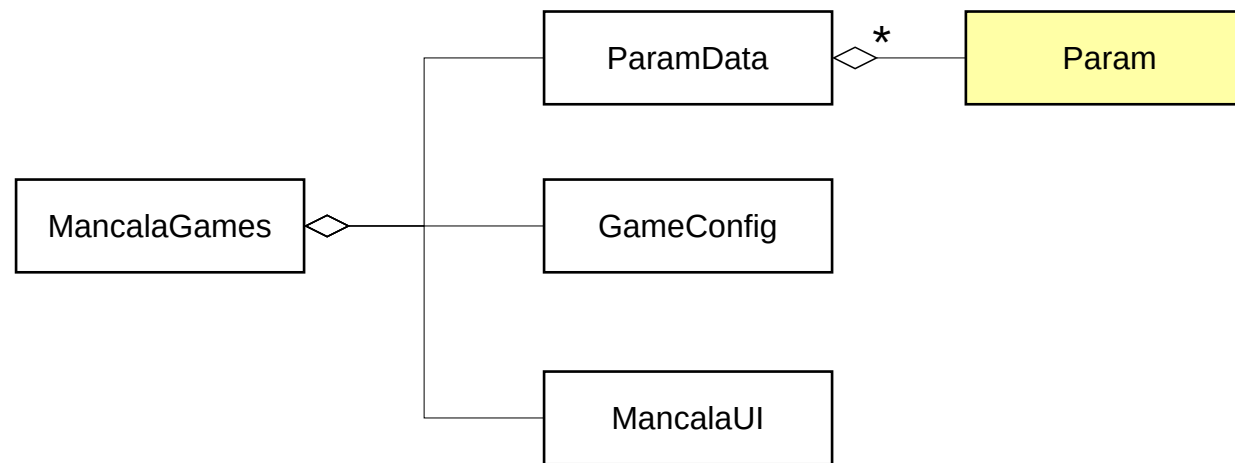
Deco Chain Conventions

- 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.
- Some deco's do not call down the deco chain even if there is a follow-on deco.
- All paths shown might not be possible (see ginfo_rules).

Optional deco

Deco Chain in Seperate Diagram
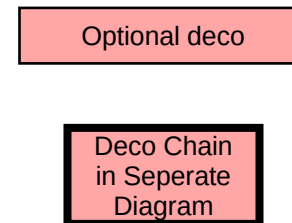
MancalaGames
(the Mancala Games UI class)

```
┌──────────────┐      ┌──────────────┐   *   ┌──────────────┐
│              │      │              │◇──────│              │
│              │      │  ParamData   │       │    Param     │
│              │   ┌──│              │       │              │
│              │   │  └──────────────┘       └──────────────┘
│              │   │
│ MancalaGames │◇──┤  ┌──────────────┐
│              │   │  │              │
│              │   ├──│  GameConfig  │
│              │   │  │              │
└──────────────┘   │  └──────────────┘
                   │
                   │  ┌──────────────┐
                   │  │              │
                   └──│  MancalaUI   │
                      │              │
                      └──────────────┘
```
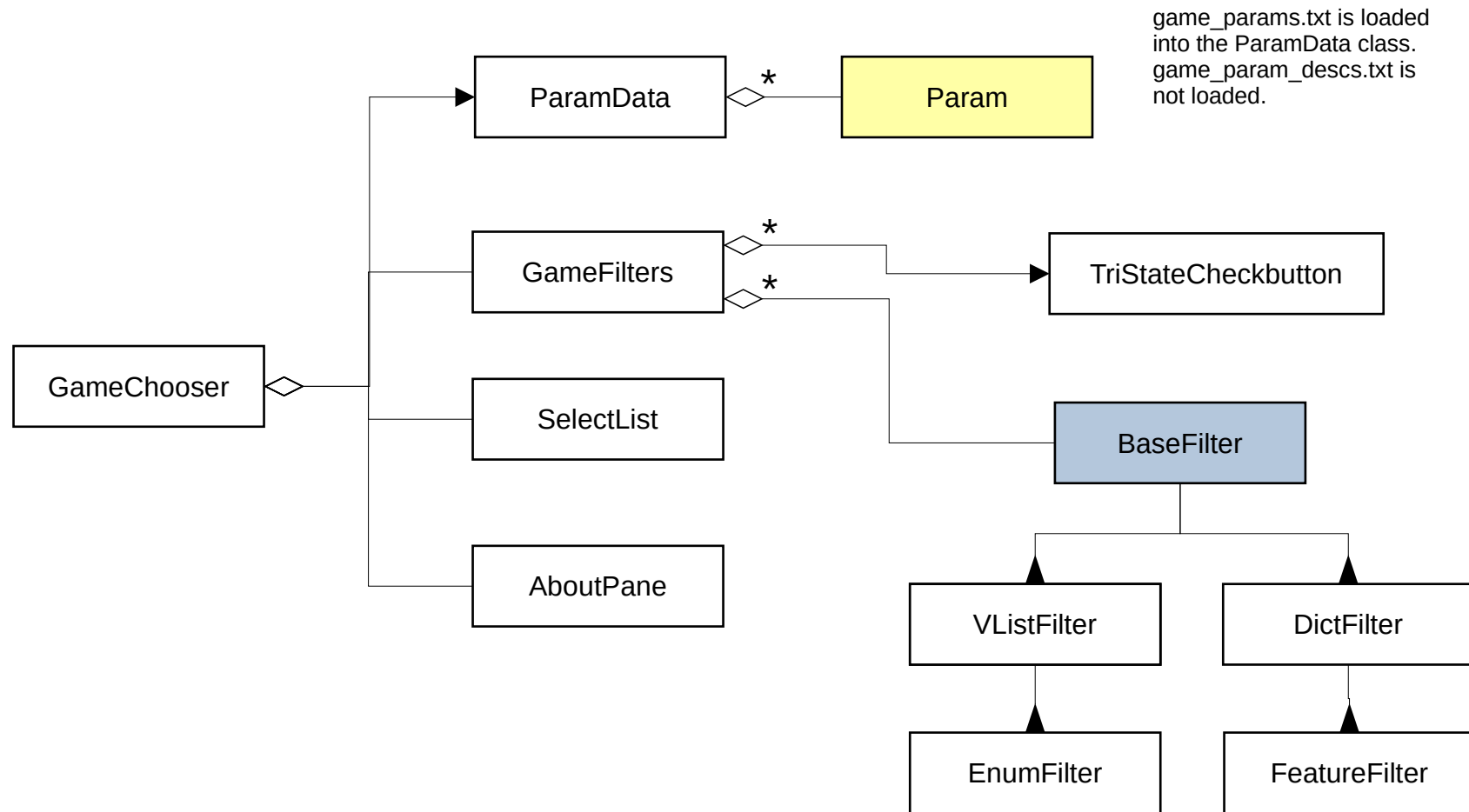
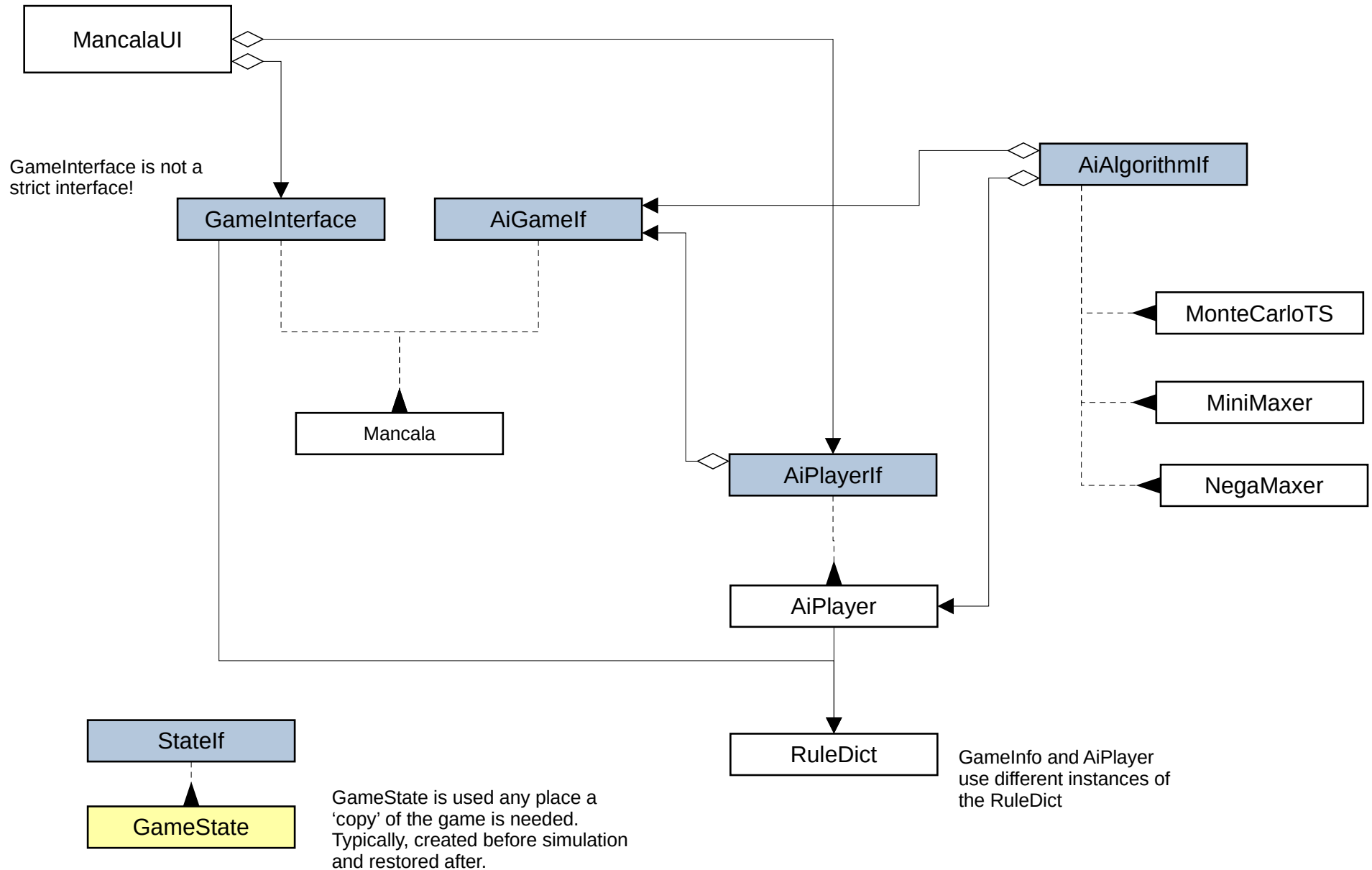game_params.txt and
game_param_descs.txt are
loaded into the ParamData
class.

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

GameChooser
(the Play Mancala UI class)

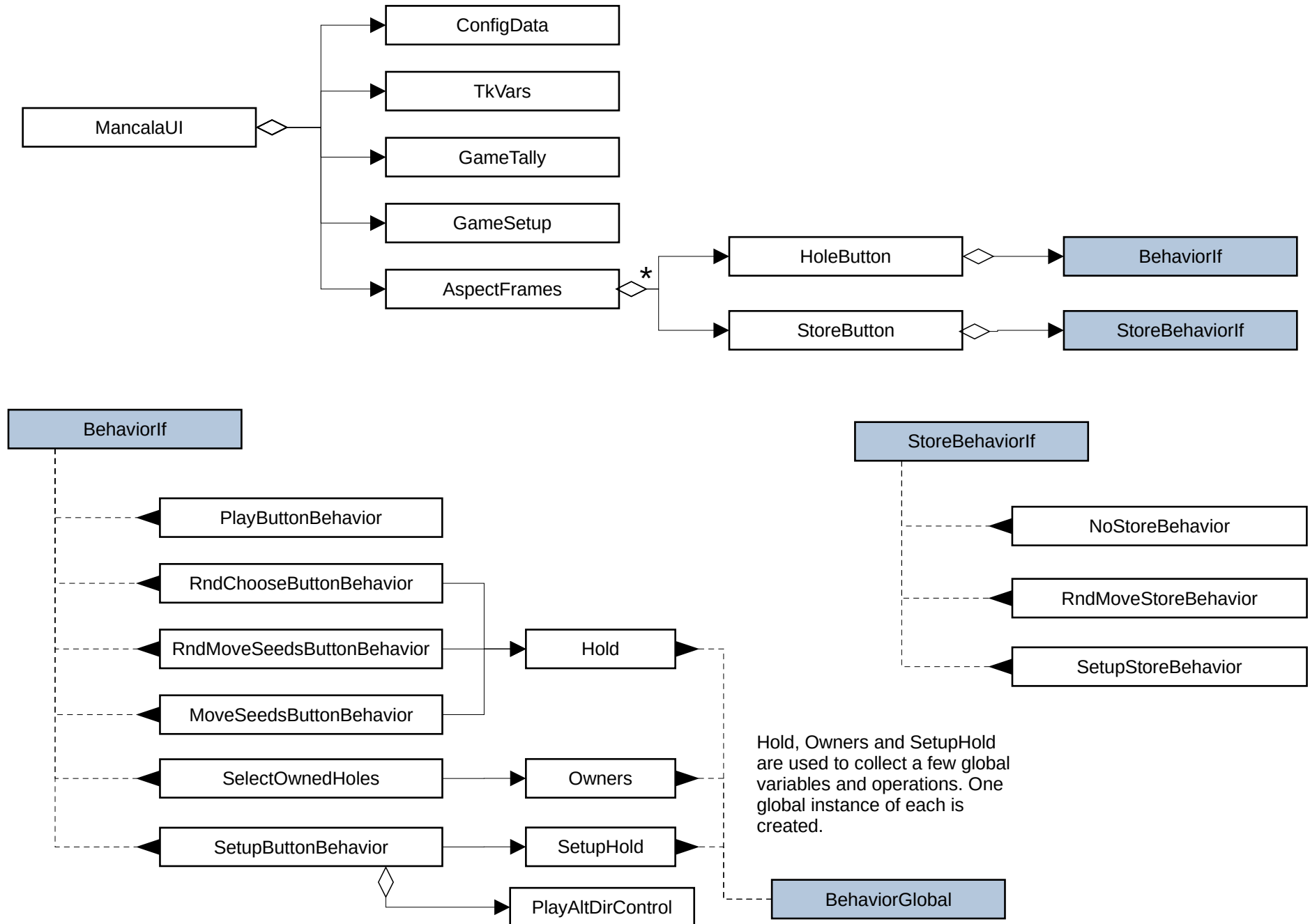game_params.txt is loaded
into the ParamData class.
game_param_descs.txt is
not loaded.

ParamData

Param

*

GameFilters

*

*

TriStateCheckbutton

GameChooser

SelectList

BaseFilter

AboutPane

VListFilter

DictFilter

EnumFilter

FeatureFilter

Mancala, GameState, AIPlayer and AIAlgorithm

MancalaUI

GameInterface is not a
strict interface!

GameInterface

AiGameIf

AiAlgorithmIf

MonteCarloTS

MiniMaxer

NegaMaxer

Mancala

AiPlayerIf

AiPlayer

StateIf

RuleDict

GameState

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

MancalaUI

ConfigData

TkVars

GameTally

GameSetup

AspectFrames

HoleButton — BehaviorIf

StoreButton — StoreBehaviorIf

BehaviorIf

PlayButtonBehavior

RndChooseButtonBehavior

RndMoveSeedsButtonBehavior

MoveSeedsButtonBehavior

Hold

SelectOwnedHoles

Owners

SetupButtonBehavior

SetupHold

PlayAltDirControl

BehaviorGlobal

StoreBehaviorIf

NoStoreBehavior

RndMoveStoreBehavior

SetupStoreBehavior

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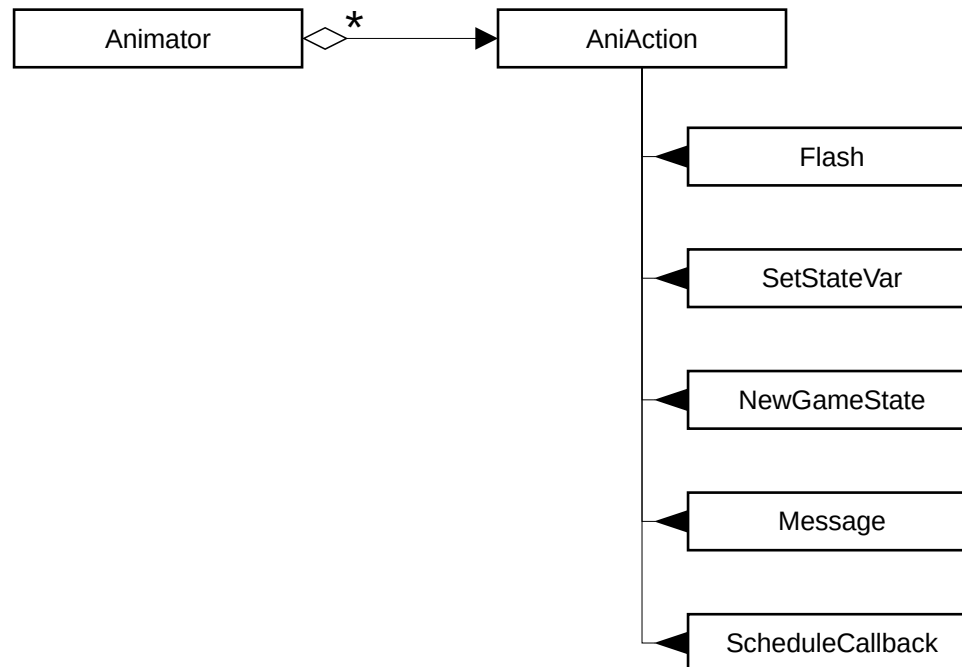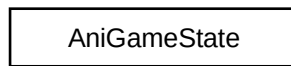global instance of each is
created.

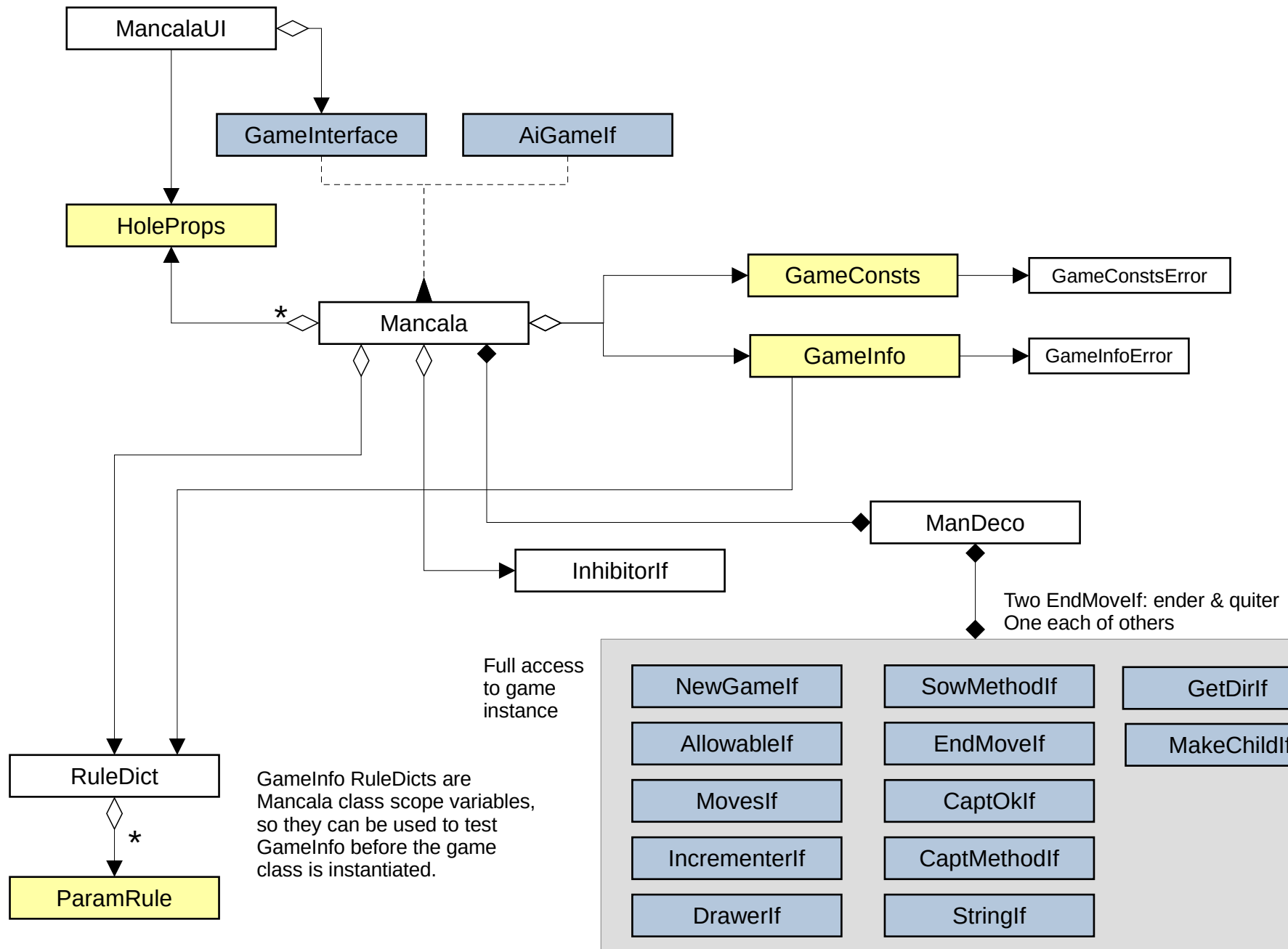# Animator Classes

Mancala ◇——*——▶ AniList

Assignments to an AniList generate SetStateVar animations.

These animator hooks are used for 5 state variable and only when they are configured for use in a game.

These hooks are not included if animator.ENABLED is set False.

AniGameState

Animator ◇——*——▶ AniAction

Flash

SetStateVar

NewGameState

Message

ScheduleCallback

Mancala Classes

MancalaUI

GameInterface

AiGameIf

HoleProps

Mancala

GameConsts

GameConstsError

GameInfo

GameInfoError

ManDeco

InhibitorIf

Two EndMoveIf: ender & quiter
One each of others

Full access
to game
instance

RuleDict

GameInfo RuleDicts are
Mancala class scope variables,
so they can be used to test
GameInfo before the game
class is instantiated.

ParamRule

*

NewGameIf

SowMethodIf

GetDirIf

AllowableIf

EndMoveIf

MakeChildIf

MovesIf

CaptOkIf

IncrementerIf

CaptMethodIf

DrawerIf

StringIf

Additional Game Classes
and Supporting Decorators

Mancala

- DiffusionV2
  - Diffusion
- SameSides
  - Ohojichi
- NorthSouthCycle
- EastWestCycle
- BearOff

GameState

- SSGameState
- BearOffState

IncrementerIf

- EastWestIncr
- NorthSouthIncr

SowMethodIf

- DiffusionSower
- SowSeedsNStore
  - NorthSouthSowSeedsNStore
- BearOffSow

animation msg
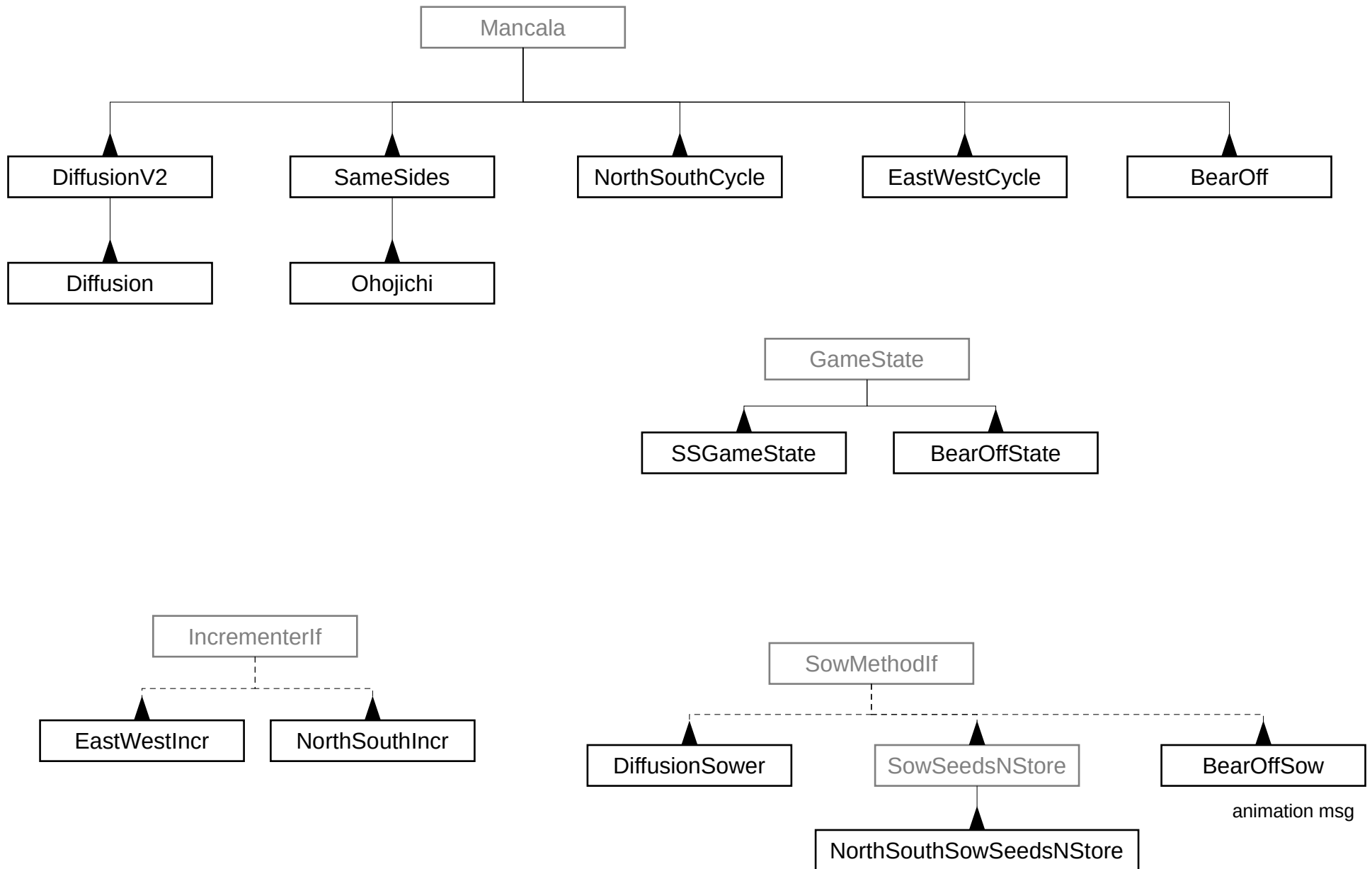
Import Classes for Moves

MoveTpl

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.

InhibitorIf

new_game()
get_state()
set_state()
clear_if()
set_on()
set_off()
set_child(condition)
stop_me_capt(turn)
stop_me_child(turn)

InhibitorNone

InhibitorCaptN

InhibitorChildrenOnly

InhibitorBoth

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

MoveData

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.

The current move's mdata is stored in Mancala, but anything stored directly into that could mess up the Monte Carlo Tree Search (it's node dictionary uses a limited version of game state, which does not include Mancala.mdata).

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

```
                              ●
                              │
                              ▼
                    ┌───────────────────┐
                    │   SeedCountCheck  │
                    └───────────────────┘
```
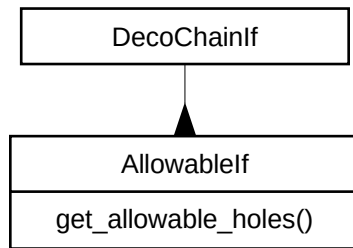
NewRound	NewRoundEven	TerritoryNewRound	NewRoundTally

```
                    ┌───────────────────┐
                    │   NewGamePattern  │◇──────────┐
                    └───────────────────┘           │
                              │                      ▼
                              ▼           ┌─────────────────────────┐
                    ┌───────────────────┐ │      StartPatternIf     │
                    │  NewFixedChildren │ ├─────────────────────────┤
                    └───────────────────┘ │     size_ok(holes)      │
                              │           │ nbr_seeds(holes, nbr_start) │
                              ▼           │    fill_seeds(game)     │
                    ┌───────────────────┐ └─────────────────────────┘
                    │      NewGame      │
                    └───────────────────┘
```
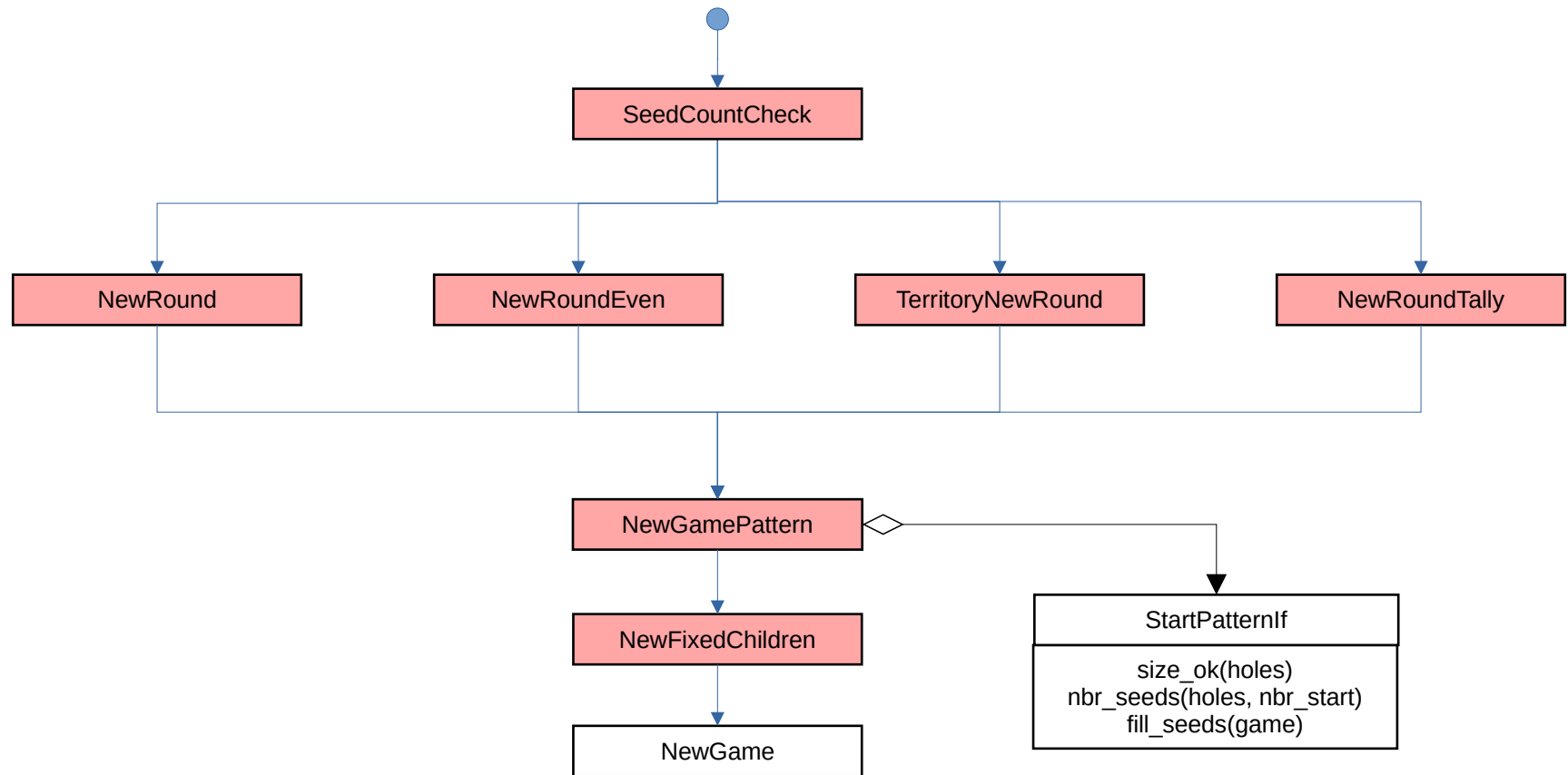
State variables changed: Parameters:
    blocked   blocks
    board   goal
    owner   min_move
    starter   round_starter
    store   round_fill
    turn   rounds
                                      start_pattern

```
                    ┌───────────────────┐
                    │     DecoChainIf   │
                    └───────────────────┘
                              △
                              │
         ┌─────────────────────────────────────────┐
         │               NewGameIf                 │
         ├─────────────────────────────────────────┤
         │    new_game(wind_cond, new_round_ok)    │
         └─────────────────────────────────────────┘
```

Allowables Decorators and Chain

DecoChainIf

AllowableIf

get_allowable_holes()

DontAnimateAllowable

MemoizeAllowable**

NoGrandSlam *

MustShare *   MustShareUdir *

Allow Rule
Decos

Allowable   AllowableTriples

State variables read: Parameters:
    turn   min_move
    board   allow_rule
    store   mlength
    blocked   mustshare
    owner   grandslam
    child   udir_holes
    mcount

Notes:
* Simulates some portion of moves to determine allowables
** MemoizeAllowable is used for deco's that simulate moves

Allow Rule Decos

Allow Rule Decos

OppOrEmptyEnd *

Occupied *

SingleToZero

OnlyIfAllN

SingleToZero

OnlyRightTwo

AllTwoRightmost

OnlyRightTwo

MoveAllFirst

DontUndoMoveOne

NotXfromOnes
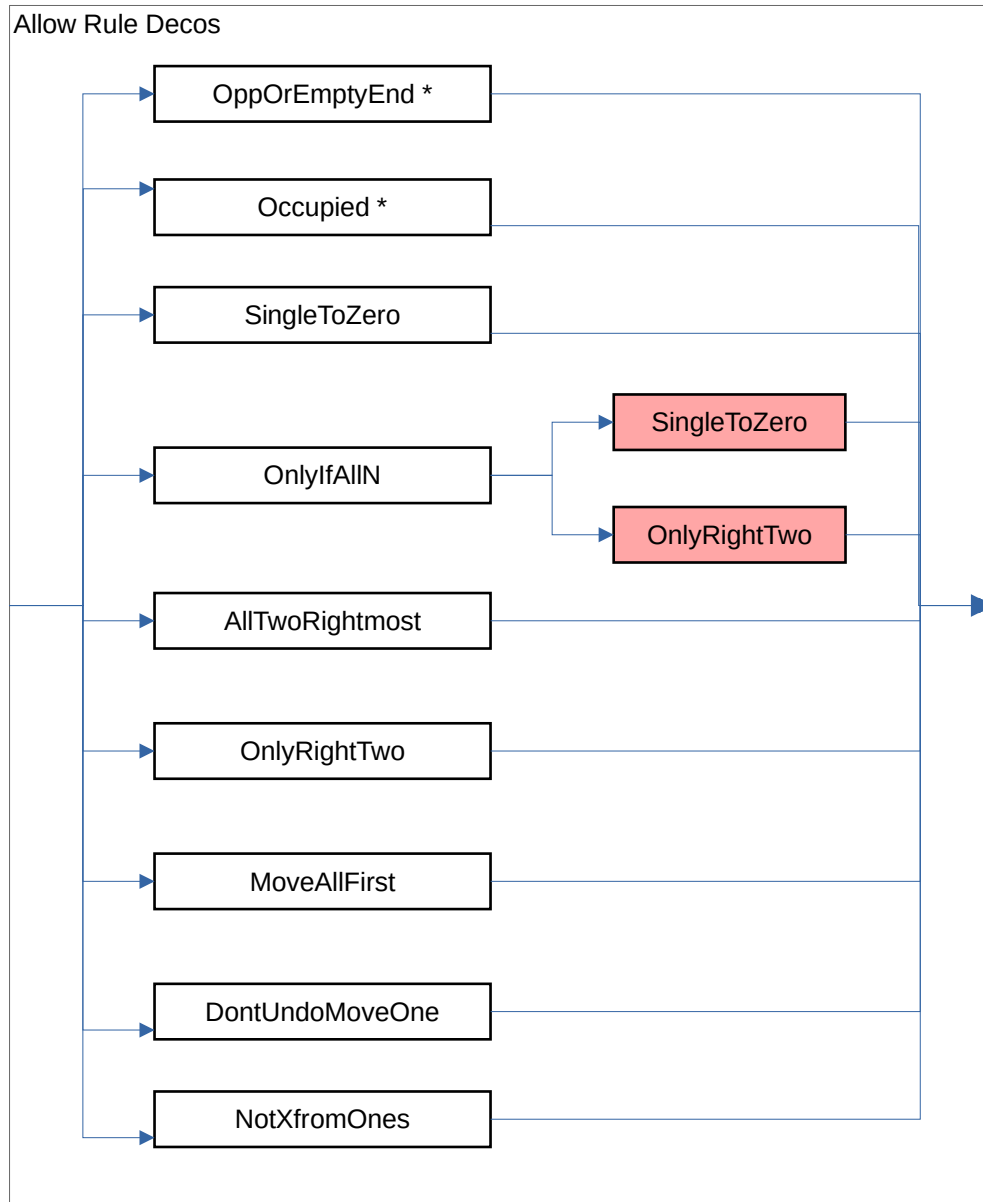
Notes:
Some allow rule decos are shown more than once for clarity.
* Simulates some portion of moves to determine allowables
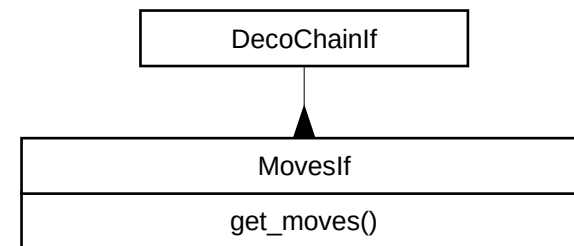
Get Moves Decorators and Chain

AltDirMoves

PassMoves
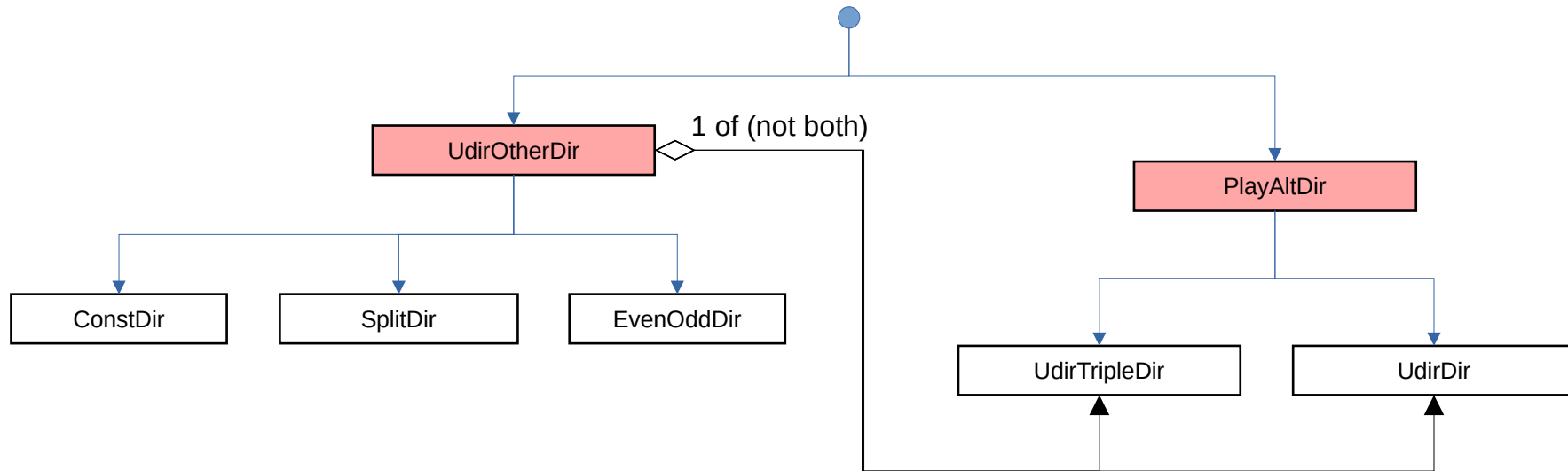
UdirMoves | MovesUdirTriples | MovesTriples | Moves
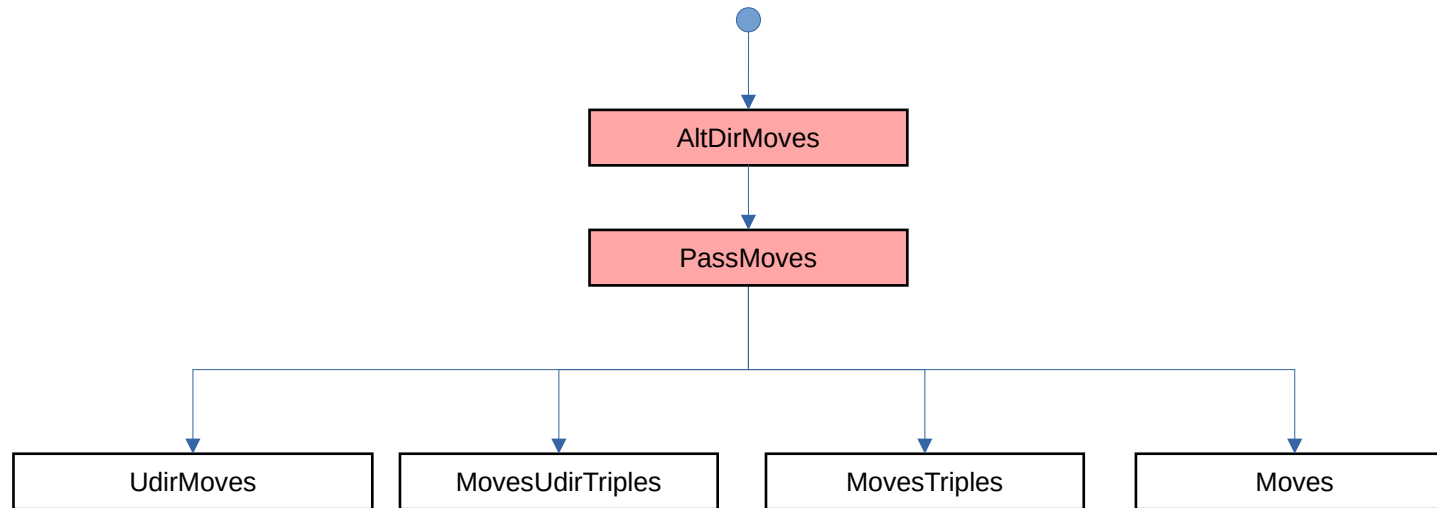
State variables read:
    blocked
    board
    owner
    starter
    store
    turn

Parameters:
    mlength
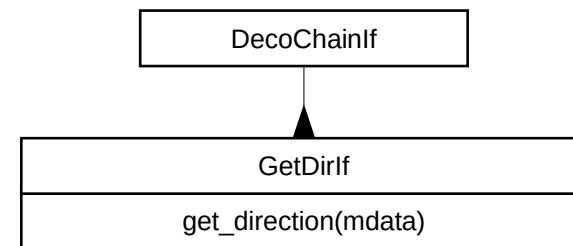   mustpass
    sow_direct
    udir_holes
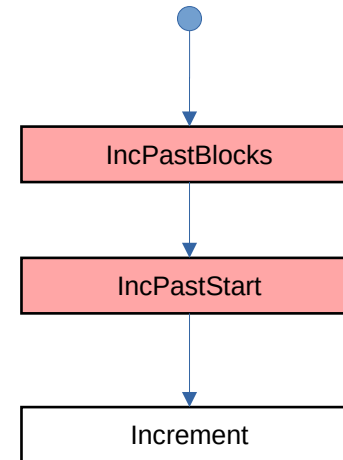   udirect

DecoChainIf

MovesIf

get_moves()

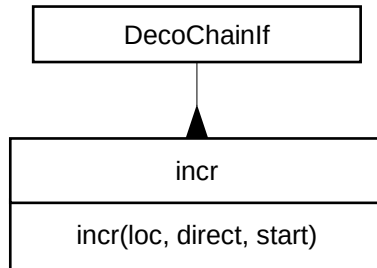# Get Direction Decorators and Chain

```
                                    ●
        ┌───────────────────────────┴──────────────────────────────┐
        ▼                                                            ▼
┌──────────────────┐  ◇ 1 of (not both)                    ┌──────────────────┐
│   UdirOtherDir   │─┘                                     │    PlayAltDir    │
└──────────────────┘                                       └──────────────────┘
    ┌──────┼──────────┐                                        ┌──────┴───────┐
    ▼      ▼          ▼                                        ▼              ▼
┌────────┐┌────────┐┌────────────┐                    ┌──────────────┐┌──────────────┐
│ConstDir││SplitDir││ EvenOddDir │                    │ UdirTripleDir││   UdirDir    │
└────────┘└────────┘└────────────┘                    └──────────────┘└──────────────┘
```

State variables read: Parameters:
    mcount   no_sides
    turn   sow_direct
                                     udir_holes
                                     udirect

```
                              ┌──────────────────────┐
                              │     DecoChainIf      │
                              └──────────────────────┘
                                         ▲
                              ┌──────────────────────┐
                              │       GetDirIf       │
                              ├──────────────────────┤
                              │ get_direction(mdata) │
                              └──────────────────────┘
```

Incrementer Decorators and Chains

DecoChainIf

incr

incr(loc, direct, start)

IncPastBlocks

IncPastStart

Increment

State variables read: Parameters:
    blocked   blocks
                                    skip_start

MakeChild Decorator and Chain

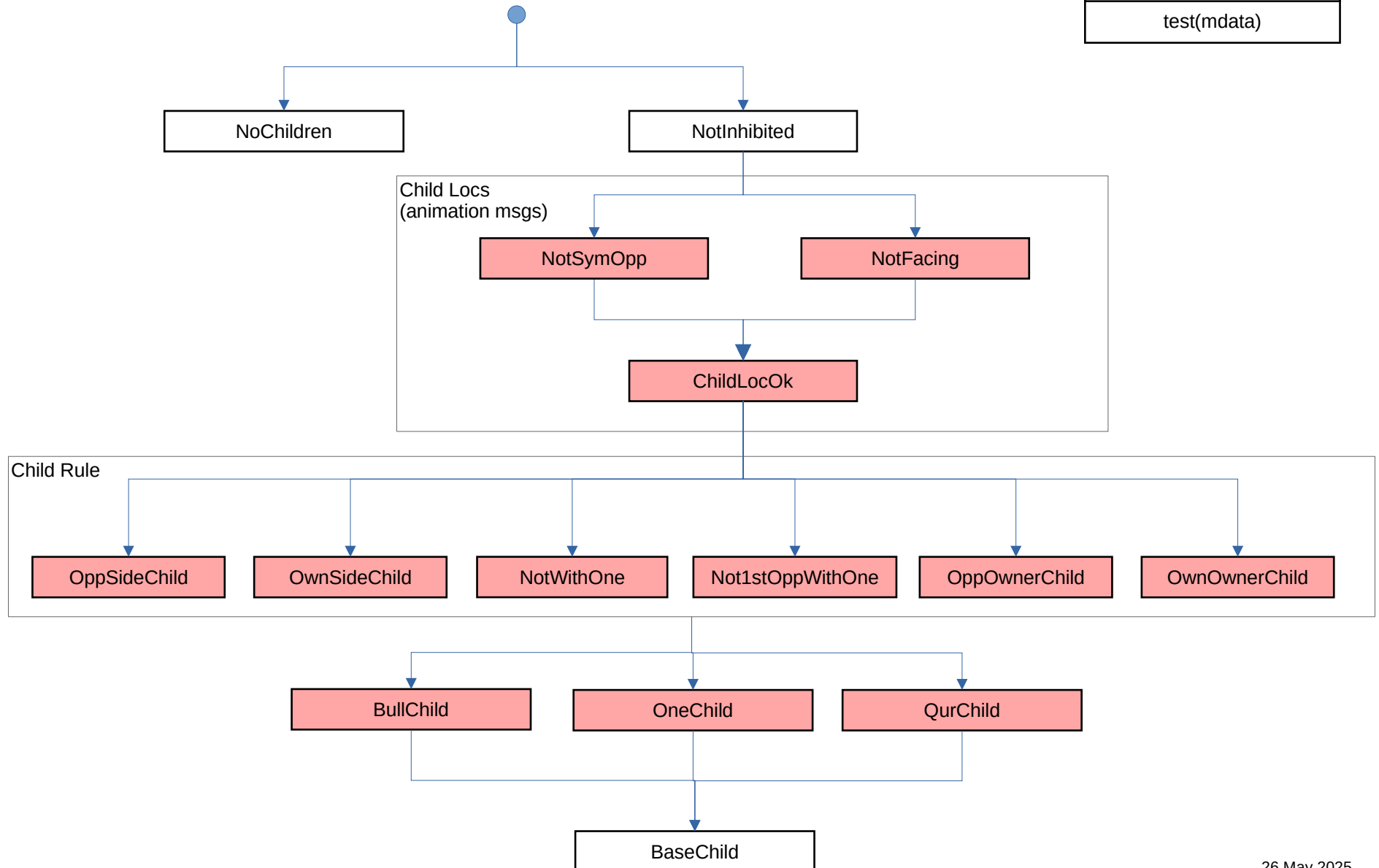State variables read:
- board
- child
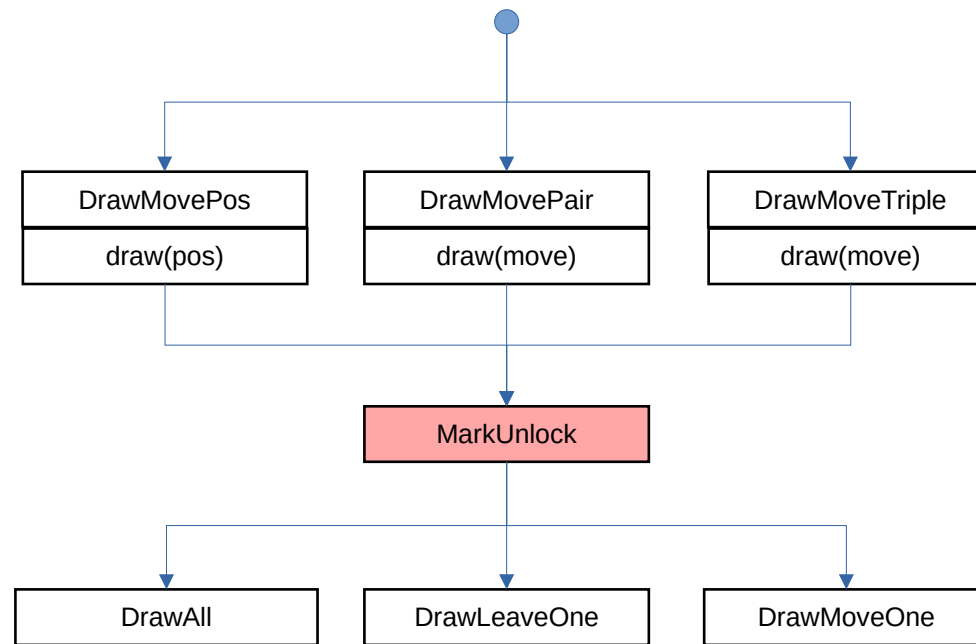- inhibitor
- owner
- turn

Parameters:
- child_cvt
- child_locs
- child_rule
- child_type

```
┌─────────────────┐
│   DecoChainIf   │
└────────┬────────┘
         ▲
┌────────┴────────┐
│   MakeChildIf   │
├─────────────────┤
│    test(mdata)  │
└─────────────────┘
```

```
NoChildren          NotInhibited
```

Child Locs (animation msgs)

NotSymOpp   NotFacing

ChildLocOk

Child Rule

OppSideChild   OwnSideChild   NotWithOne   Not1stOppWithOne   OppOwnerChild   OwnOwnerChild

BullChild   OneChild   QurChild

BaseChild

Draw Decorators and Chain

```
        ●
```

DrawMovePos	DrawMovePair	DrawMoveTriple
draw(pos)	draw(move)	draw(move)

The first drawer converts the move into board location.

MarkUnlock

DrawAll	DrawLeaveOne	DrawMoveOne

State variables:
    Read:
        turn
    Changed:
        board
        unlocked

Parameters:
    allow_rule
    mlength
    move_one
    moveunlock
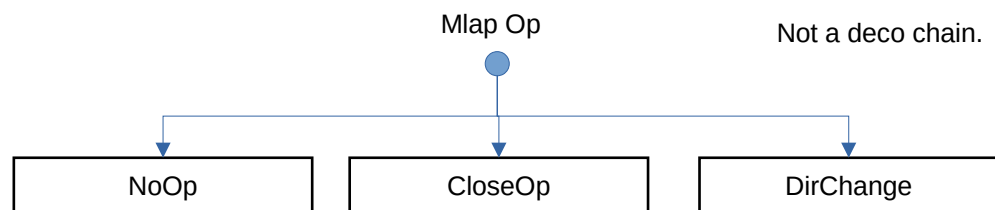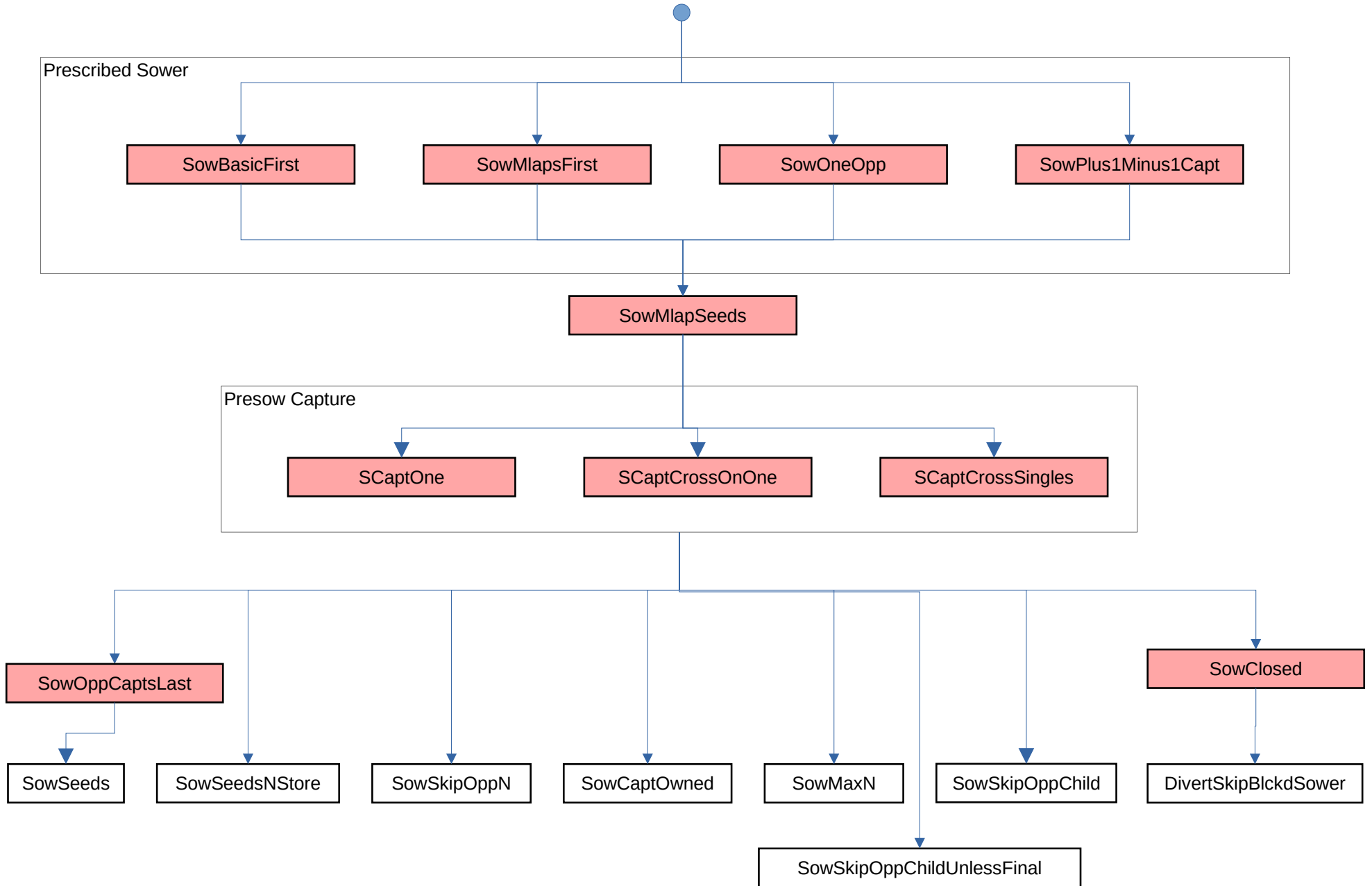   sow_start

DecoChainIf

DrawerIf
draw(loc)

Sower Decorators

```
                    ┌─────────────────────┐
                    │     DecoChainIf      │
                    └─────────────────────┘
                         ▲            │
                         │            │
         ┌───────────────────────┐    │
         │      SowMethodIf       │    │
         ├───────────────────────┤    │
         │  get_single_sower()    │    │
         │  sow_seeds(mdata)      │    │
         └───────────────────────┘    │
            ▲              ▲           │
            │              │           │
  ┌──────────────────┐  ┌──────────────────┐     ▲
  │  SowPrescribedIf │  │   MlapSowerIf    │◇─┐   │
  ├──────────────────┤  └──────────────────┘◇─┤   │
  │ do_prescribed(mdata)│                    │   │
  └──────────────────┘                       │   │
                                             │  ┌──────────────────────┐
                                             └─▶│    LapContinuerIf     │
                                                ├──────────────────────┤
                                                │ do_another_lap(mdata) │
                                                └──────────────────────┘

                                                ┌──────────────────────┐
                                          ─────▶│     MlapEndOpIf       │
                                                ├──────────────────────┤
                                                │     do_op(mdata)      │
                                                └──────────────────────┘
```

State variables:	Parameters:
Reads	capt_max
inhibitor	capt_min
turn	capt_on
child	child_type
mcount	crosscapt
Changes	evens
board	goal
store	gparam_one
blocked	mlaps
	prescribed
	presowcapt
	sow_direct
	sow_own_store
	sow_param
	sow_rule
	visit_opp

Mlap Op

Not a deco chain.

```
              ●
    ┌─────────┼─────────┐
    ▼         ▼         ▼
┌────────┐┌────────┐┌──────────┐
│  NoOp  ││ CloseOp││ DirChange│
└────────┘└────────┘└──────────┘
```

Sower Deco Chain

Lap Continuer Deco Chain and Mlap Operation

Lapper ●

```
AnimateLapStart
      │
      ▼
StopRepeatTurn
```

```
StopNotN          StopNotN
```

```
MustVisitOpp
```

```
NextLapCont          StopOnChild
```

```
ContIfXCapt   ContIfBasicCapt   ContWithCaptSeeds   GapNextCapt   StopCaptureSimul   StopCaptureSeeds
```

```
StopSingleSeed
```

```
ChildLapCont   DivertBlckdLapper   LapContinue
```
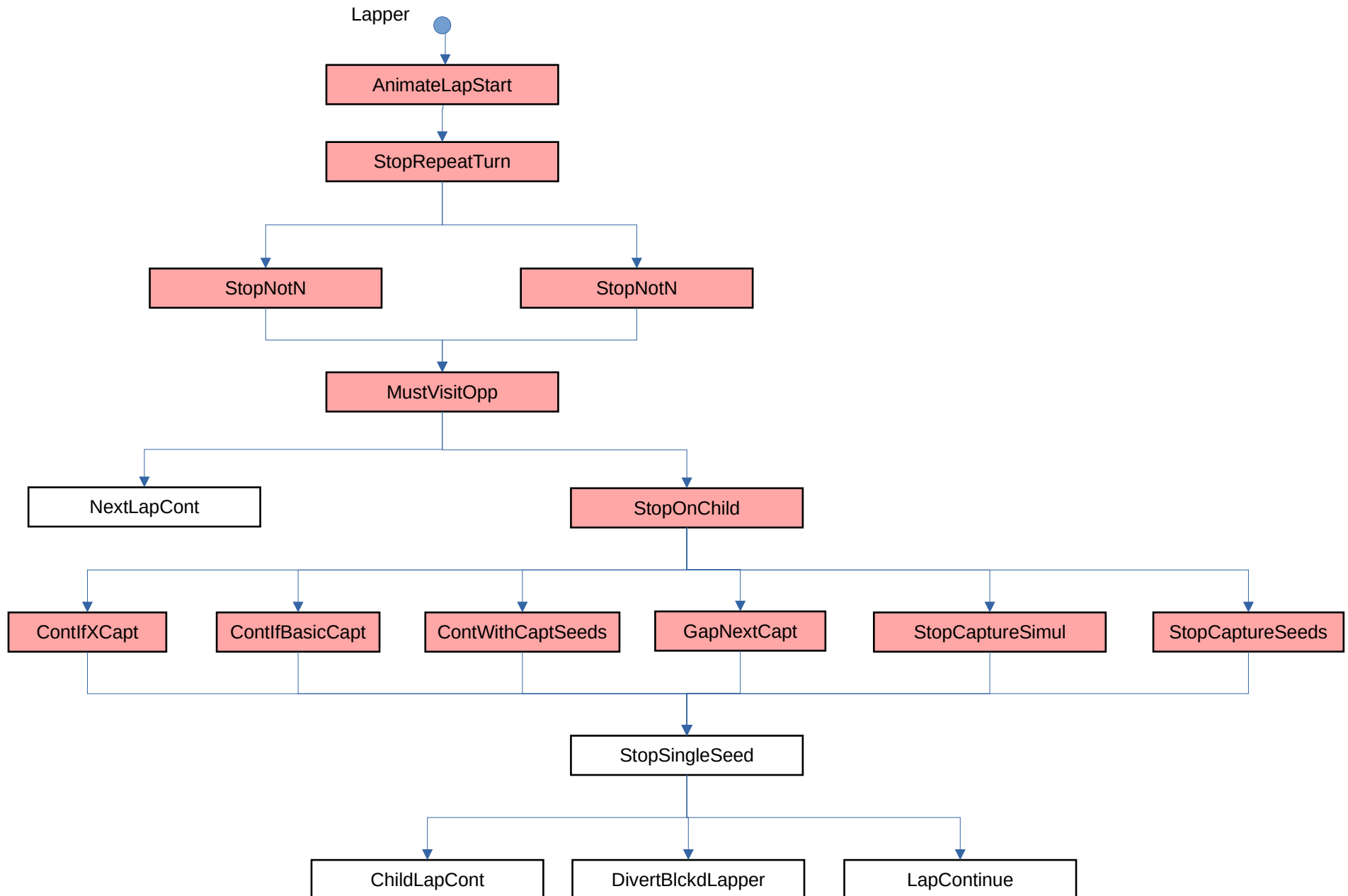
Capt Ok Decorators and Chains

DecoChainIf

CaptOkIf

capture_ok(mdata, loc)
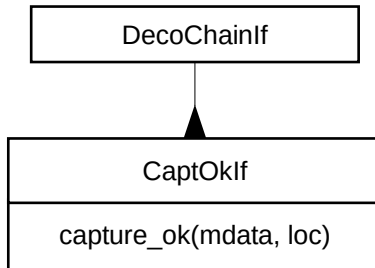
State variables read:
- board
- child
- turn
- unlocked

Parameters:
- capt_max
- capt_min
- capt_on
- capt_side
- moveunlock

CaptNeedSeedsNotChild

CaptUnlocked

CaptSideOk

CaptMax

CaptMin

CaptEvens

CaptOn

CaptTrue
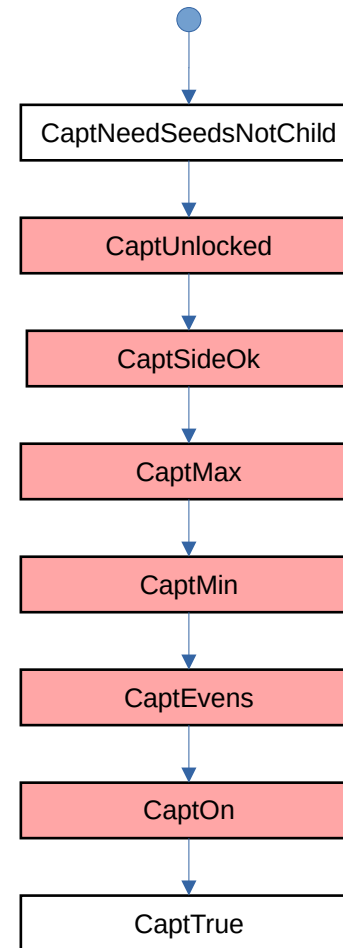
This is the Basic Capture Criteria.

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

Capturer Decorators and Chain

DecoChainIf

CaptMethodIf

do_captures(mdata)
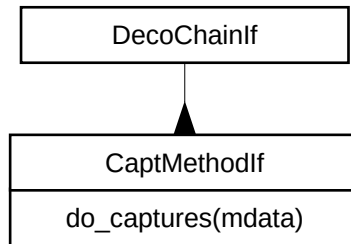
State variables
    Reads
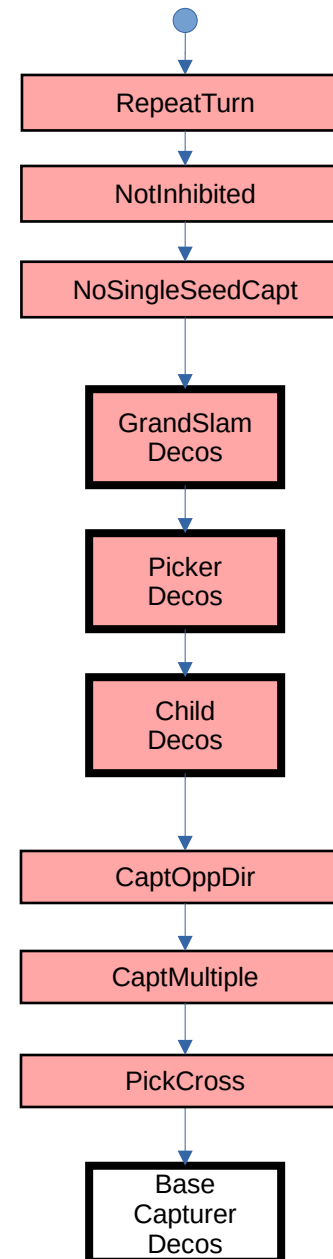   inhibitor
        starter
        turn
    Changes
        board
        child
        store

Parameters:
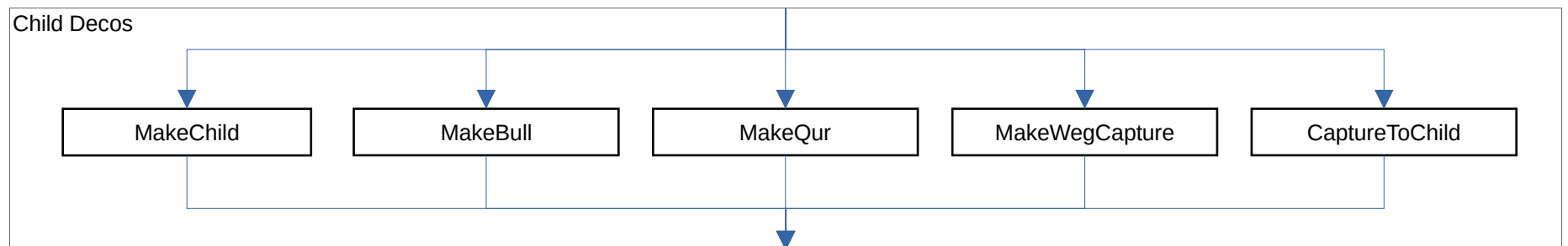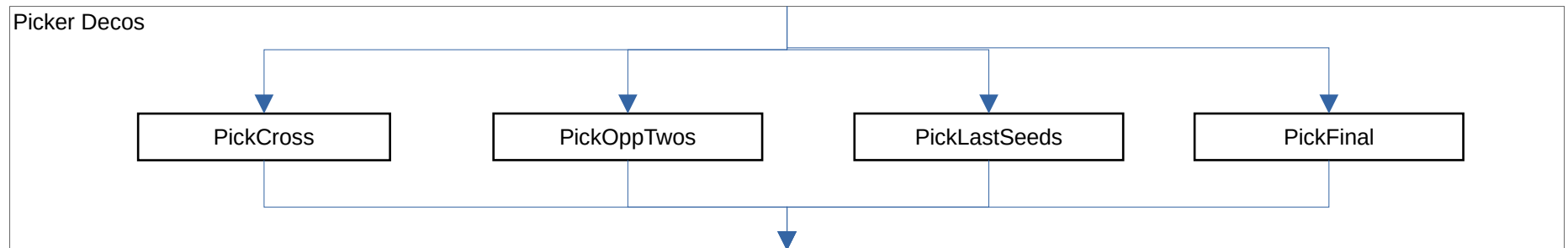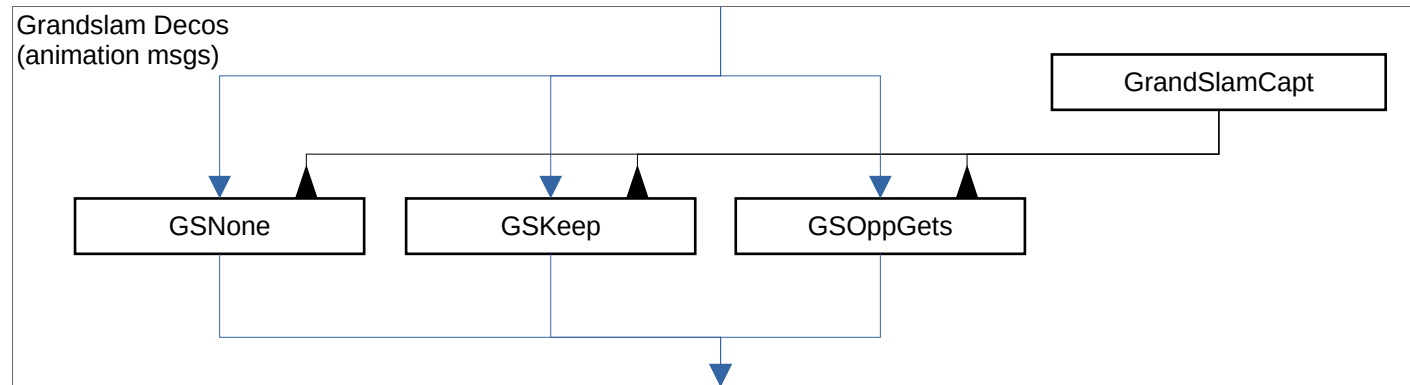   capsamedir
    capt_max
    capt_min
    capt_on
    capt_rturn
    capt_side
    capt_type
    child_cvt
    child_type
    crosscapt
    evens
    grandslam
    mlaps
    multicapt
    nocaptmoves
    nosinglecapt
    pickextra
    prescribed
    round_fill
    xc_sown
    xcpickown

Chain (top to bottom):

RepeatTurn

NotInhibited

NoSingleSeedCapt

GrandSlam Decos

Picker Decos

Child Decos

CaptOppDir

CaptMultiple
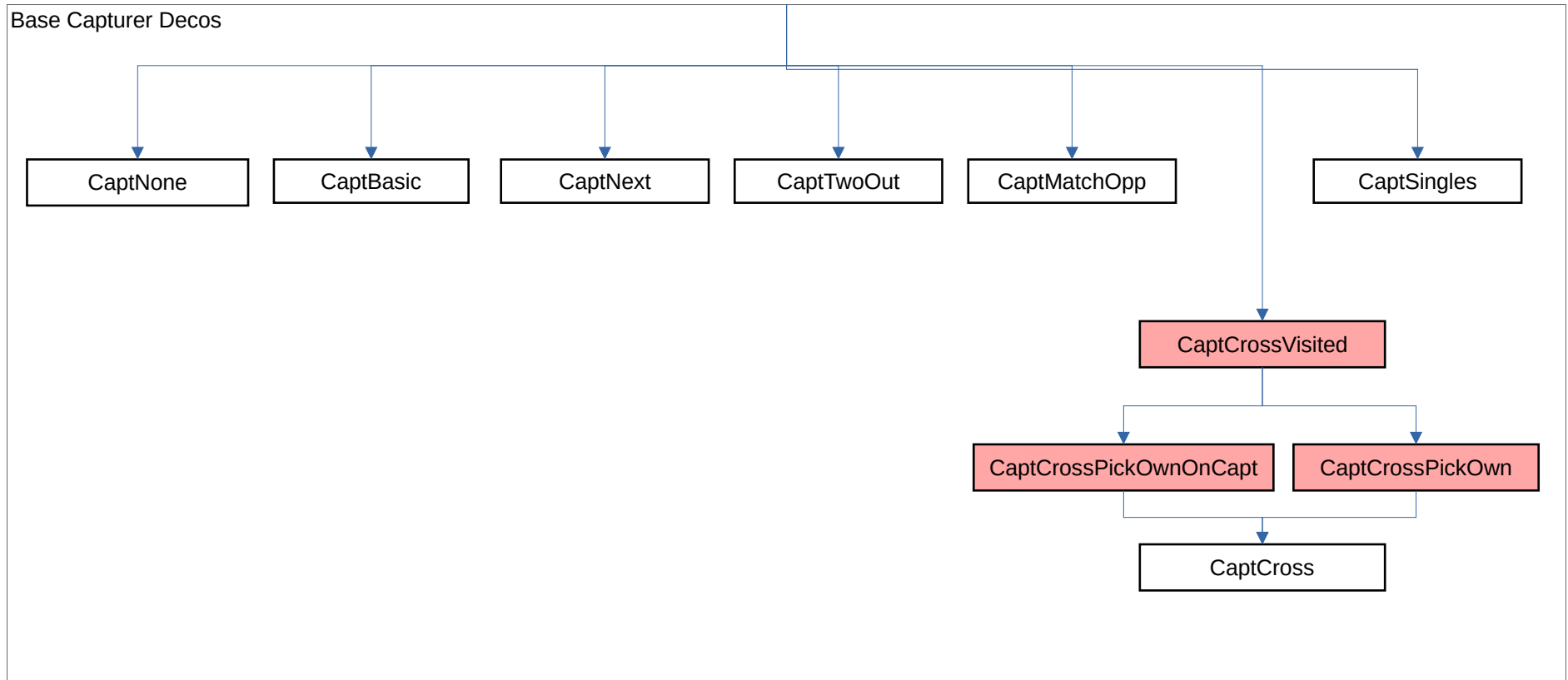
PickCross

Base Capturer Decos

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.
- PickCross is only put in the deco chain once, either in Picker Decos or after CaptMultiple.

Capturer Deco Chains
(1 of 2)

Grandslam Decos
(animation msgs)

GrandSlamCapt

GSNone   GSKeep   GSOppGets

Picker Decos

PickCross   PickOppTwos   PickLastSeeds   PickFinal

Child Decos

MakeChild   MakeBull   MakeQur   MakeWegCapture   CaptureToChild

Capturer Deco Chains
(2 of 2)

Base Capturer Decos

| CaptNone | CaptBasic | CaptNext | CaptTwoOut | CaptMatchOpp | CaptSingles |

CaptCrossVisited

CaptCrossPickOwnOnCapt   CaptCrossPickOwn

CaptCross

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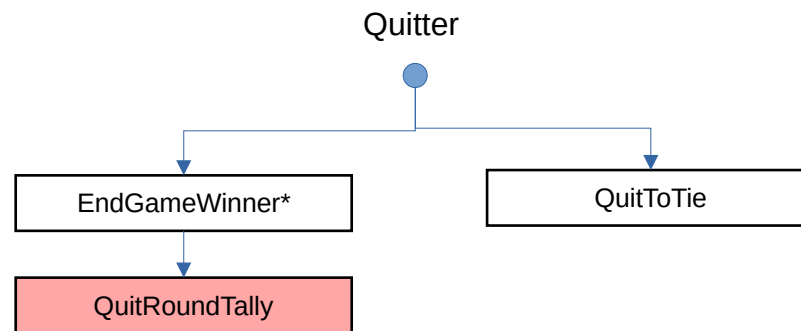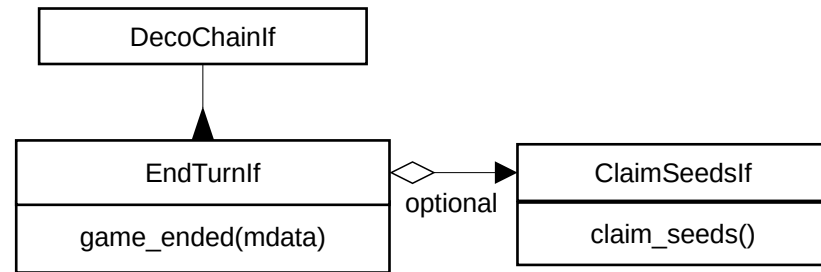## (1 of 2)

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

```
┌──────────────────┐
│   DecoChainIf    │
└──────────────────┘
         ▲
┌──────────────────┐        ┌──────────────────┐
│     EndTurnIf    │◇──────▶│   ClaimSeedsIf   │
├──────────────────┤ optional├──────────────────┤
│ game_ended(mdata)│        │   claim_seeds()  │
└──────────────────┘        └──────────────────┘
```

Quitter

```
              ●
      ┌───────┴────────┐
      ▼                ▼
┌──────────────┐  ┌──────────────┐
│EndGameWinner*│  │   QuitToTie  │
└──────────────┘  └──────────────┘
      ▼
┌──────────────┐
│ QuitRoundTally│
└──────────────┘
```

Note:
*A claimer, taker or divvier is
selected based on the
unclaimed, child_type and store
properties (see next page).

Ender & Quiter Decorators and Chains
(2 of 2)

Ender ●

AnimateEndMove

ChildNoStoresEnder

RoundTallyWinner CLAIMER

RoundEndLimit

DepriveEndGame

ImmobilizeEndGame *

ClearSeedsEndGame

RoundTallyWinner CLAIMER

RoundWinner CLAIMER

EndTurnNoMoves *

ClearWinner CLAIMER

NoOutcomeChange

EndTurnNotPlayable

EndTurnMustShare *

EndTurnNoMoves *

EndTurnPassPass *

EndGameWinner TAKER

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