Mancala Games

Game Configuration Files

game_params.csv

game_param_descs.txt

man_games_editor.pyw — defines → MancalaGames

UI to edit game configugration files,
allow changing all parameters, and
starting the MancalaUI to play games.

game_dict

man_config.py

read_game

make_game

play_mancala.pyw — defines → GameChooser

UI to load game config files, filter games
based on parameters, select a game and
start MancalaUI to play the game.

MancalaUI

mancala.ini

play.pyw

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

MancalaUI

mancala.ini

Notation Conventions

Class Diagram Conventions

Abstract Base Class

Primarily Data

◇——— Has-a

◇——*——— Has-a > 1

┄┄┄◀ Implements

——◀ Extends

◆——◆ Composes

——▶ Uses
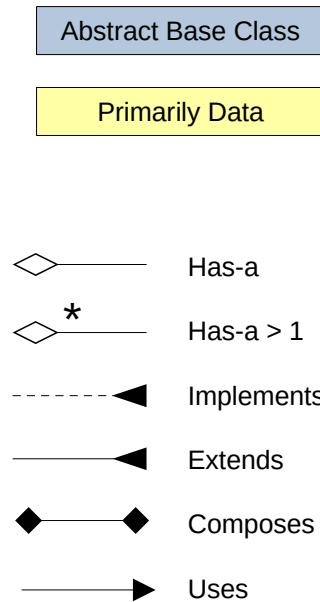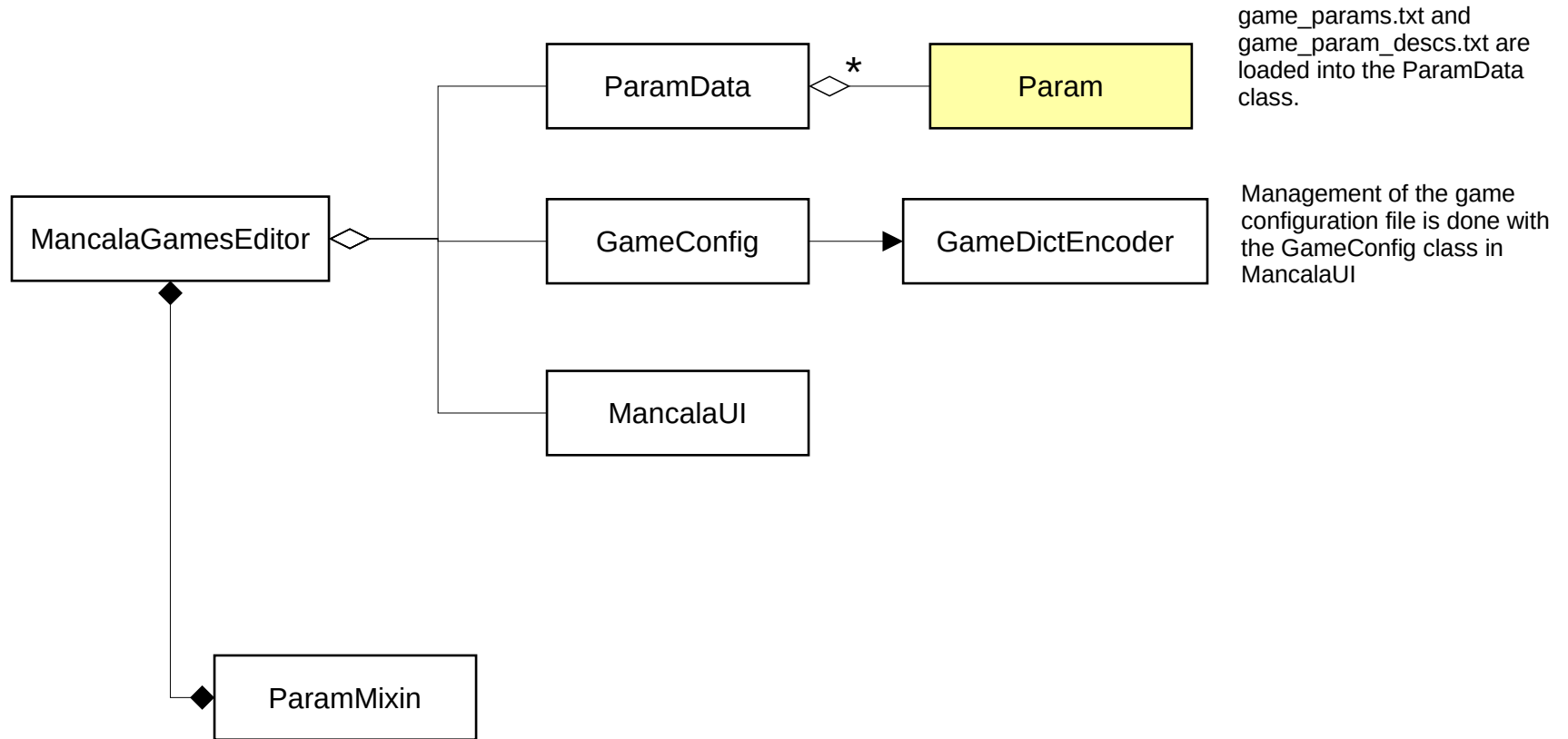
Deco Chain Conventions

- 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

MancalaGamesEditor
man_games_editor.pyw

ParamData

Param

*

game_params.txt and
game_param_descs.txt are
loaded into the ParamData
class.

MancalaGamesEditor

GameConfig

GameDictEncoder

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

MancalaUI

ParamMixin
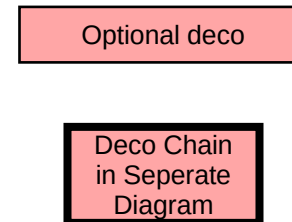
GameChooser
play_mancala.pyw

ParamData

*

Param

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

GameFilters

*

BaseFilter

GameChooser

SelectList

VListFilter

DictFilter

AboutPane

EnumFilter

FeatureFilter

*

TriStateCheckbutton
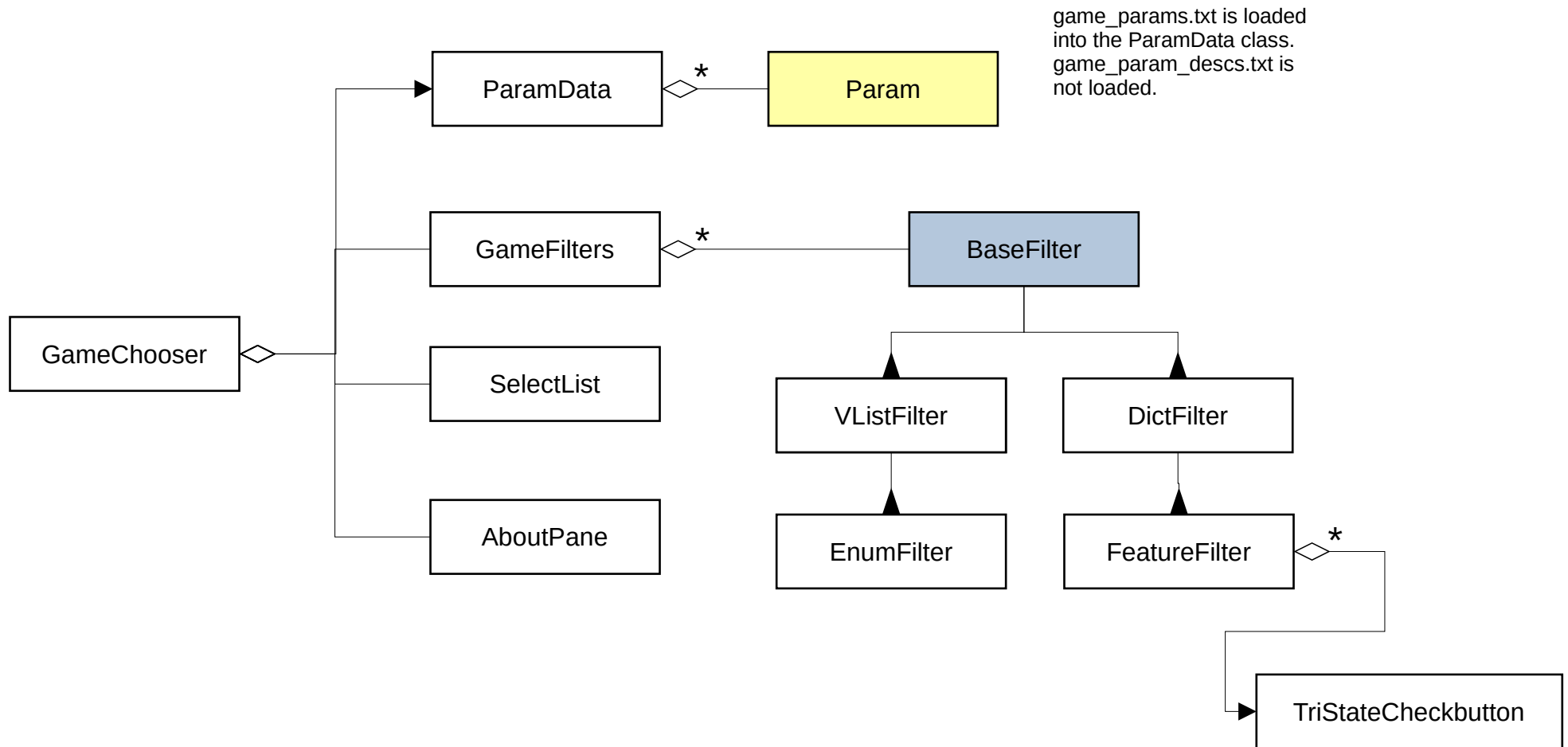
MancalaUI Classes
mancala_ui.py
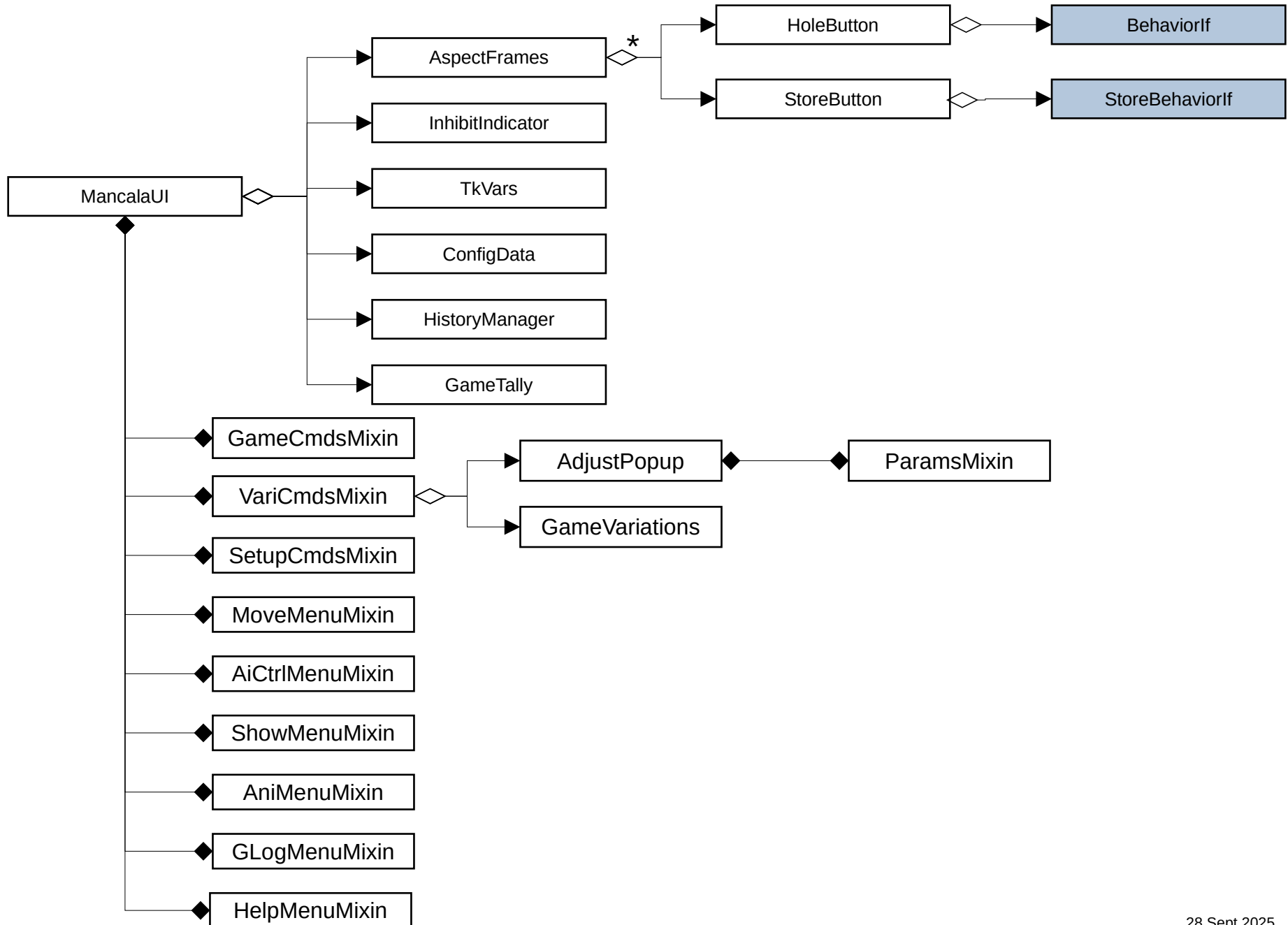
Behavior Classes for MancalaUI

BehaviorIf

PlayButtonBehavior

RndChooseButtonBehavior

RndMoveSeedsButtonBehavior

Hold

MoveSeedsButtonBehavior

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

SelectOwnedHoles

Owners

SetupButtonBehavior

SetupHold

PlayAltDirControl

BehaviorGlobal

StoreBehaviorIf

NoStoreBehavior

RndMoveStoreBehavior

SetupStoreBehavior

AIPlayer and AIAlgorithm Integration

MancalaUI

AiPlayerIf

AiPlayer

AiGameIf

Mancala

AiAlgorithmIf

The Mancala class does not know who is playing it or if there is a UI being used to play it.

RuleTester

MonteCarloTS

MiniMaxer

NegaMaxer

Get via the board_state property

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.

StateIf

GameState

Mancala Classes

MancalaUI

HoleProps

Mancala

ManMsgsMixin

GameConsts → GameConstsError

GameInfo → GameInfoError

GameInfo uses the 'rules' class-scope method of the Mancala class to test GameInfo before the game class is instantiated.

InhibitorIf

ManDeco

*

Full access to game instance attributes

Two EndMoveIf: ender & quiter
Two CaptOkIf: capt_basic and capt_check
One each of others

NewGameIf

AllowableIf

MovesIf

IncrementerIf

DrawerIf

SowMethodIf

EndMoveIf

CaptOkIf

CaptMethodIf

StringIf

GetDirIf

MakeChildIf

Animator Classes

Mancala ◇——*——▶ AniList
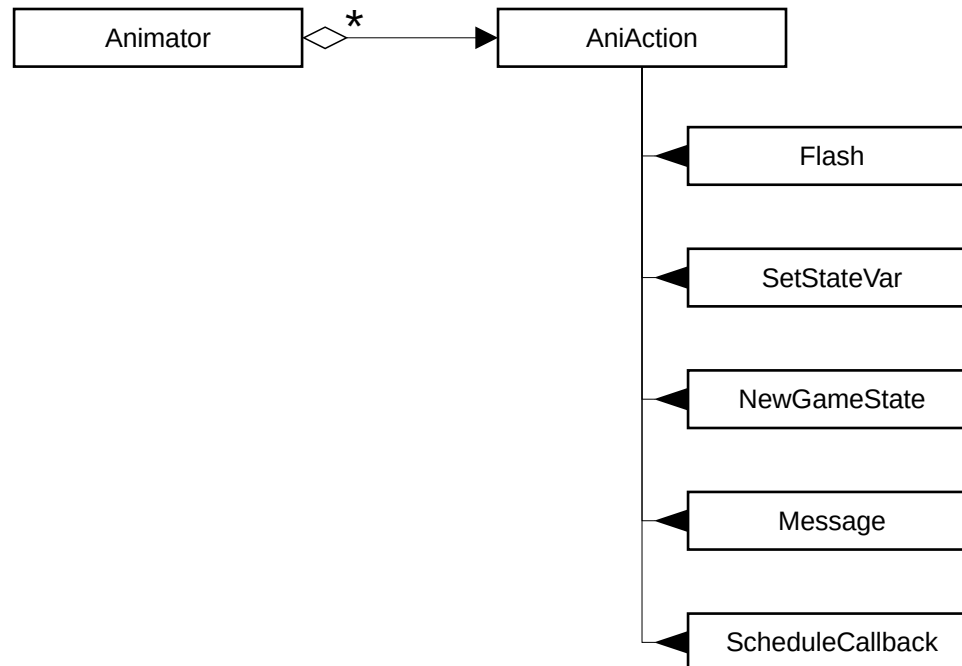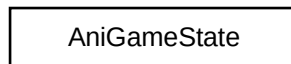
Assignments to an AniList generate SetStateVar animations.

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

Two Cycle Game Classes

- Attributes and methods shown are over-ridden, reassigned or reconfigured.

- Consult code for how deco chains are modified.

Mancala

NorthSouthCycle
rules(cls, ginfo, holes, skip)

ManDeco
incr
sower

EastWestCycle
rules(cls, ginfo, holes, skip)
disallow_endless(disallow)

GameConsts
false_range
true_range
board_side(loc)

GameInfo
mlength

ManDeco
allow
incr

IncrementerIf

NorthSouthIncr

EastWestIncr

NSIncOwnStores

EWIncOwnStores

NSIncBothStores

EWIncBothStores

NSIncFromStores

EWIncFromStores

AllowableIf

EastWestAllowable

SameSide and Ohojichi Game Classes

Mancala

SameSide

rules(cls, ginfo, holes, skip)
state
board_state
new_game(new_round)
end_game(quittter, user, game)
get_allowable_holes()
capture_seeds(mdata)
move(move)
empty_store --new attrib

GameInfo

no_sides
mlength

ManDeco

incr

Ohojichi

rules(cls, ginfo, holes, skip)
get_allowable_holes()

GameConsts

false_range
true_range
board_side(loc)

ManDeco

incr

GameState

SSGameState
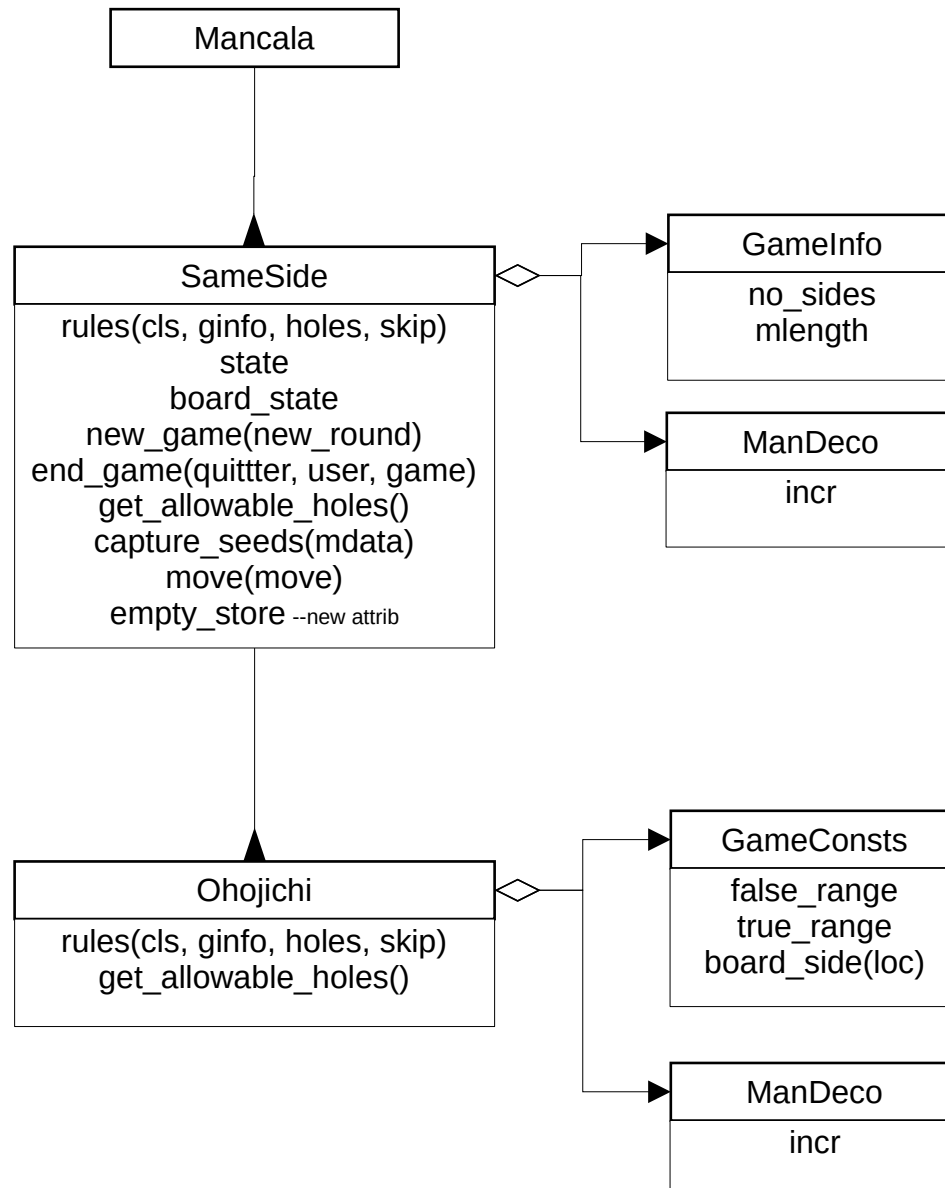
- Attributes and methods shown are over-ridden, reassigned or reconfigured.

- Each game class uses the appropriate two_cycle incrementer as the base incrementer.

- Ohojichi only calls the allow deco chain when on the first part of turns, not on the place seeds opposite part.
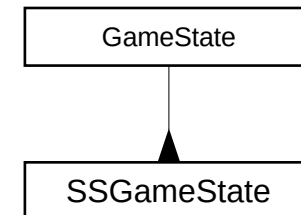
Diffusion and DiffusionV2 Game Classes

```
         ┌─────────────────────┐
         │       Mancala       │
         └─────────────────────┘
                    △
                    │
         ┌─────────────────────┐      ┌─────────────────────┐
         │      DiffusionV2     │◇────▶│       ManDeco       │
         ├─────────────────────┤      ├─────────────────────┤
         │ rules(cls, ginfo,   │      │        incr         │
         │ holes, skip)        │      │        sower        │
         └─────────────────────┘      └─────────────────────┘
                    △
                    │
         ┌─────────────────────┐      ┌─────────────────────┐
         │      Diffusion      │◇────▶│     GameConsts      │
         ├─────────────────────┤      ├─────────────────────┤
         │ rules(cls, ginfo,   │      │    false_range      │
         │ holes, skip)        │      │     true_range      │
         │ win_message(win_cond)│     │   board_side(loc)   │
         └─────────────────────┘      └─────────────────────┘
```
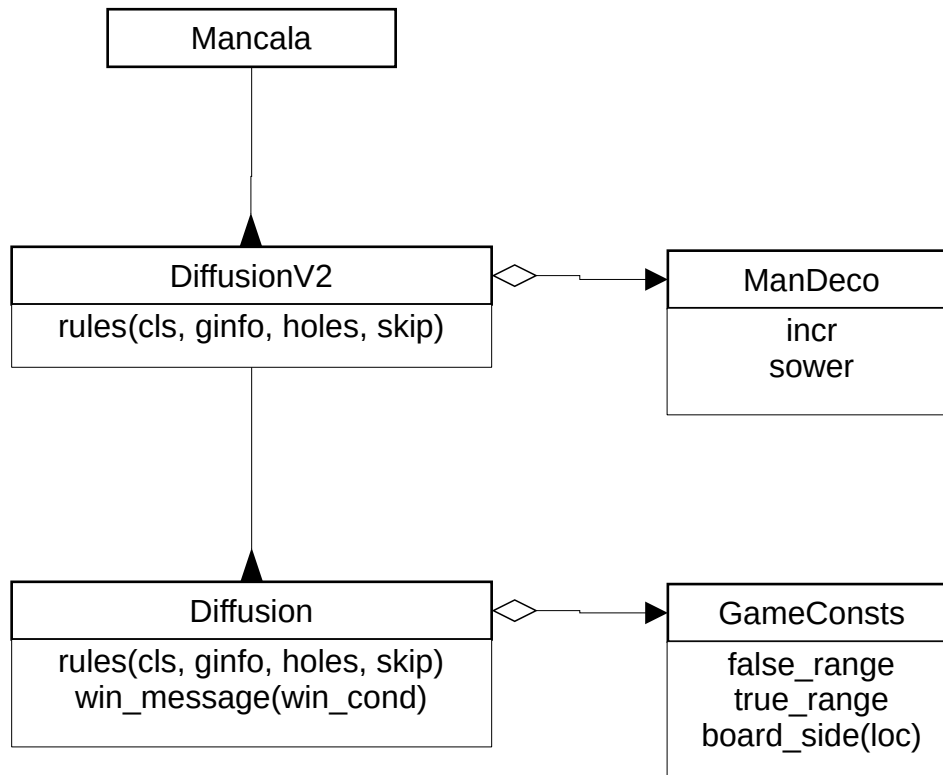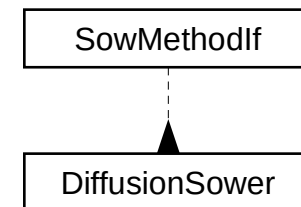
- Attributes and methods shown are over-ridden, reassigned or reconfigured.

- The incr deco chain is cleared because it should not be used: the sower is completely replaced and the capturer deco is CaptNone.

- Both game classes use the DiffusionSower.

```
         ┌─────────────────────┐
         │     SowMethodIf     │
         └─────────────────────┘
                    △
                    ┆
         ┌─────────────────────┐
         │    DiffusionSower   │
         └─────────────────────┘
```

Bear Off Game Class

Mancala

BearOff

rules(cls, ginfo, holes, skip)
state
board_state
new_game(new_round)
move(move)
normal_sow --new attrib

ManDeco
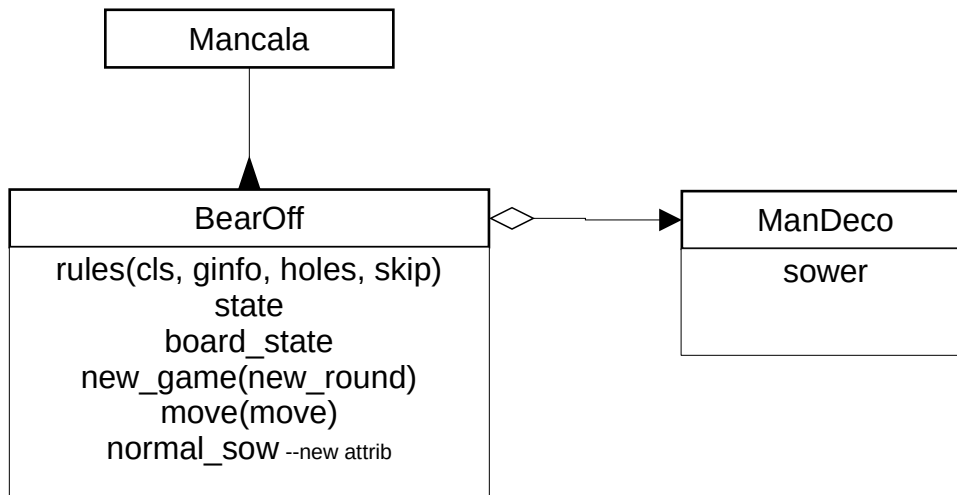
sower

- Attributes and methods shown are over-ridden, reassigned or reconfigured.

- The BearOff sower is inserted in the deco chain before the single sower. The BearOffSower either does the bear off style sowing or calls down the deco chain to the original single sower.

GameState

BearOffState

SowMethodIf

BearOffSow
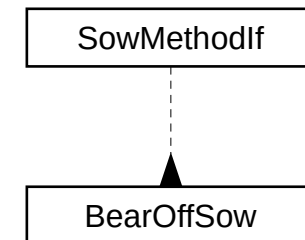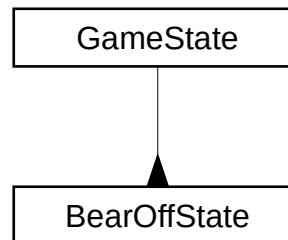
animation msg

ShareOne Game Class

```
      ┌─────────────────────┐
      │       Mancala       │
      └─────────────────────┘
                 △
                 │
 ┌──────────────────────────────┐        ┌──────────────────────┐
 │          ShareOne            │◇──────▶│       ManDeco        │
 ├──────────────────────────────┤        ├──────────────────────┤
 │  rules(cls, ginfo, holes, skip) │     │        sower         │
 │     get_allowable_holes()    │        └──────────────────────┘
 │         move(move)           │
 └──────────────────────────────┘
```

```
 ┌──────────────────────┐
 │     SowMethodIf      │
 └──────────────────────┘
            △
            ┊
 ┌──────────────────────┐
 │     ShareOneSow      │
 └──────────────────────┘
```
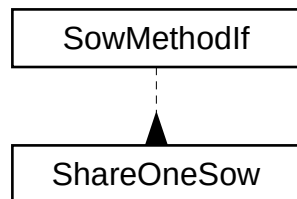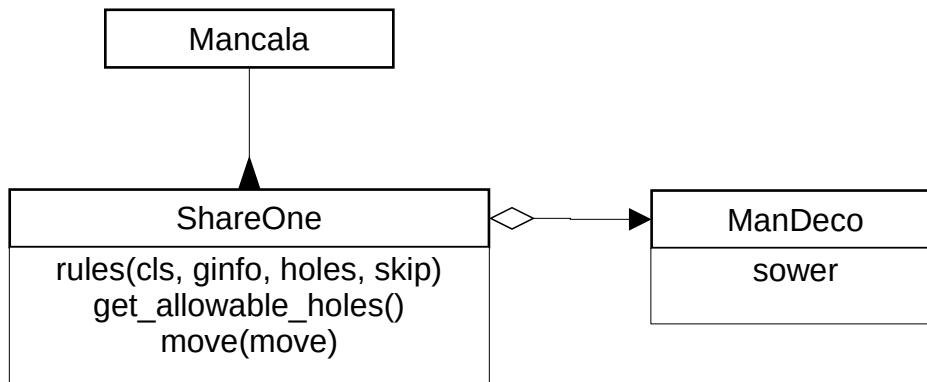
- Attributes and methods shown are over-ridden, reassigned or reconfigured.

- If the move will be a share one move, only holes that are not children with 2 or more seeds are allowable (the allow deco is not used); otherwise, the deco chain is used.

- If the next move is to share one seed and the animator is active, a message is popped up via the move method override.

- ShareOneSow wraps the existing deco chain and performs an alternate sow to share the one seed.

ZigZag Game Class

```
┌─────────────────┐
│     Mancala     │
└─────────────────┘
          △
          │
┌─────────────────┐      ┌─────────────────┐
│     ZigZag      │◇────▶│     ManDeco     │
├─────────────────┤      ├─────────────────┤
│ rules(cls, ginfo, hole,│      allow      │
│  s, skip)       │      │     get_dir     │
│ disallow_endless(disall│       incr      │
│  ow)            │      │                 │
└─────────────────┘      └─────────────────┘
```
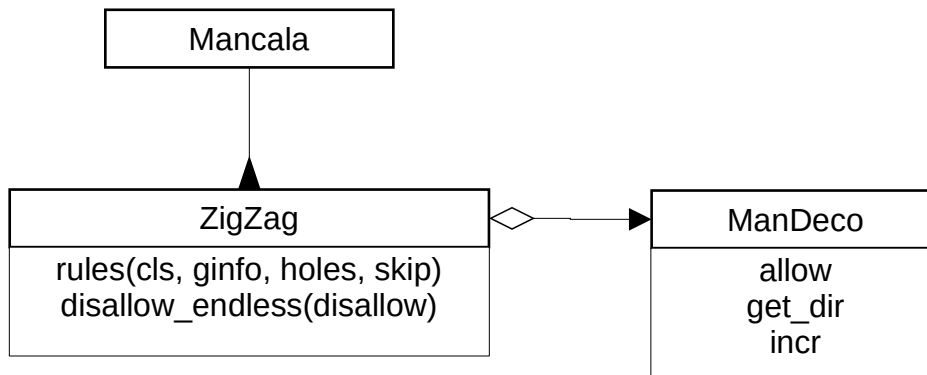
- Attributes and methods shown are over-ridden, reassigned or reconfigured.

- ZigZag Cycle:

    - The ZigZag cycle is similar to the normal cycle in that each hole is visited once before any hole is visited a second time.

    - The cycle is generated as though sowing from South's Leftmost hole (loc 0) throught the board to North's Rigtmost hole.

    - The sow direction describes which way through this cycle and the incrementer should move.

- Consult code for how deco chains are modified.

```
┌─────────────────┐      ┌─────────────────┐      ┌─────────────────┐
│   AllowableIf   │      │     GetDirIf    │      │  IncrementerIf  │
└─────────────────┘      └─────────────────┘      └─────────────────┘
         △                        △                        △
         ┆                        ┆                        ┆
┌─────────────────┐      ┌─────────────────┐      ┌─────────────────┐
│ DontUndoMoveOne │      │   ZigZagGetDir  │      │  MapIncrementer │
└─────────────────┘      └─────────────────┘      └─────────────────┘
         △                                                 △
         │                                                 │
┌──────────────────────┐                         ┌─────────────────┐
│ZigZagDontUndoMoveOne │                         │   ZigZagIncr    │
└──────────────────────┘                         └─────────────────┘
```
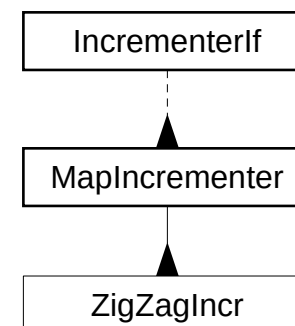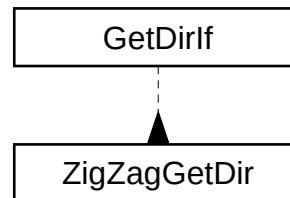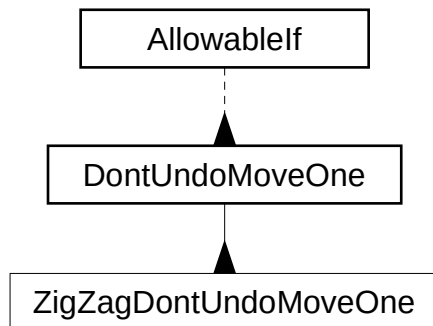
Important Classes for Moves

```
┌─────────────────────────┐
│        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
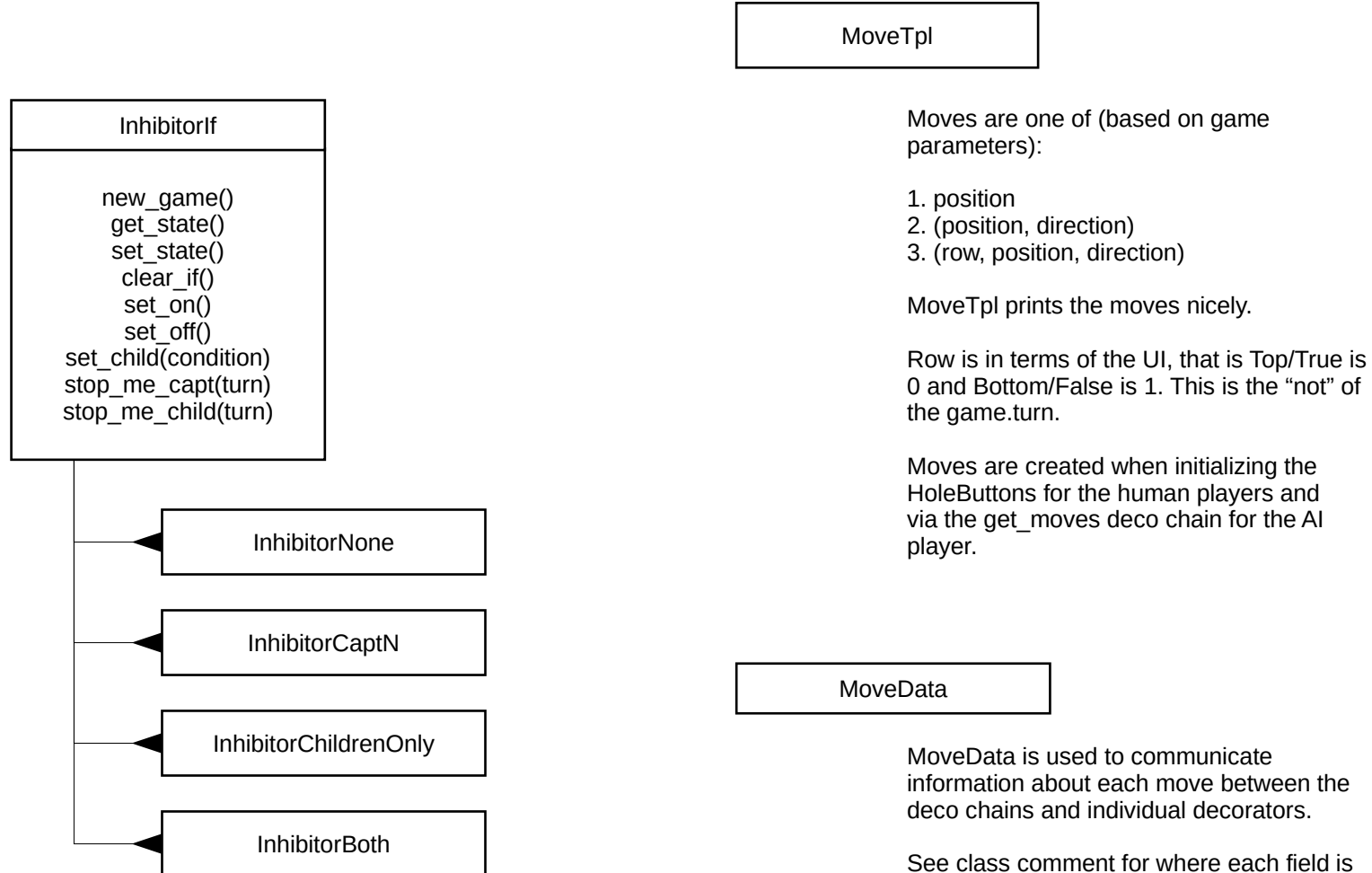
InhibitorBoth

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

animation msgs

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.

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	MoveData	The game needs to end either because of endless sow or user selection. If not configured to do something else, unclaimed seeds are divvied between the players.

New Game Decorators and Chain

```
                              ●
                              │
                              ▼
                     ┌─────────────────┐
                     │  SeedCountCheck │
                     └─────────────────┘
```

| NewRound | NewRoundEven | TerritoryNewRound | NewRoundTally | NewRoundLoserOnly |

```
                     ┌─────────────────┐      ◇──────┐
                     │  NewGamePattern │             │
                     └─────────────────┘             ▼
                              │              ┌──────────────────────────┐
                              ▼              │       StartPatternIf      │
                     ┌─────────────────┐     ├──────────────────────────┤
                     │ NewFixedChildren│     │      size_ok(holes)      │
                     └─────────────────┘     │         err_msg          │
                              │              │ 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(new_round)│
                     └─────────────────┘
```

Allowables Decorators and Chain

DecoChainIf

AllowableIf

get_allowable_holes()

DontAnimateAllowable

MemoizeAllowable**

DontUndoMoveOne

Included automatically
when it applies.

NoGrandSlam *

MustShare *   MustShareUdir *

Allow Rule
Decos

NoEndlessSows*

Allowable   AllowableTriples   AllowableNoSidesTriples
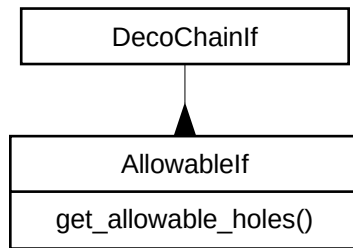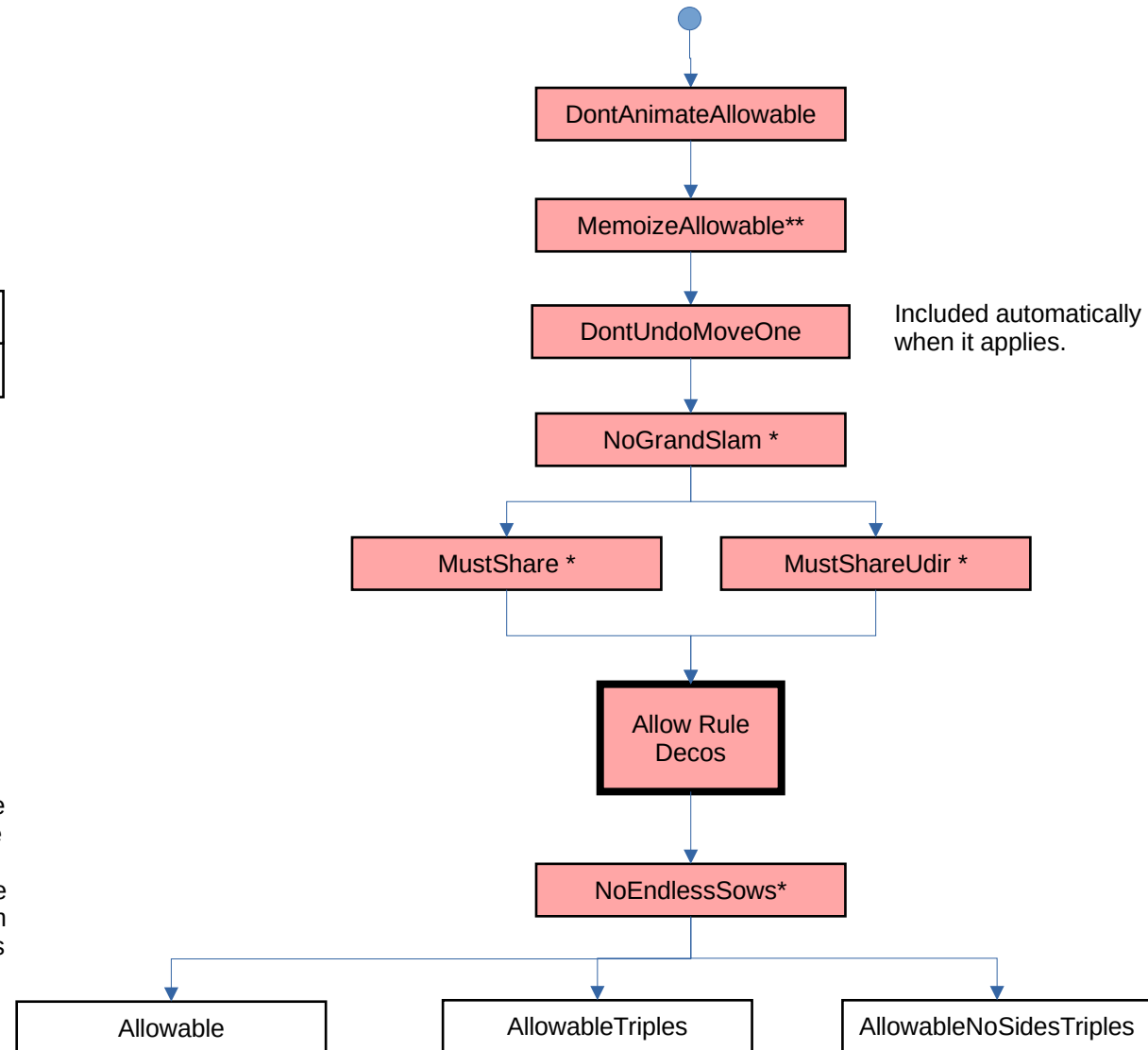
State variables read:
   turn
   board
   store
   blocked
   owner
   child
   mcount

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

Allow Rule Decos

OppOrEmptyEnd *

Occupied *

SingleToZero

OnlyIfAllN → SingleToZero

OnlyIfAllN → OnlyRightTwo

AllTwoRightmost

OnlyRightTwo

MoveAllFirst

NotXfromOnes

RightHalfFirsts

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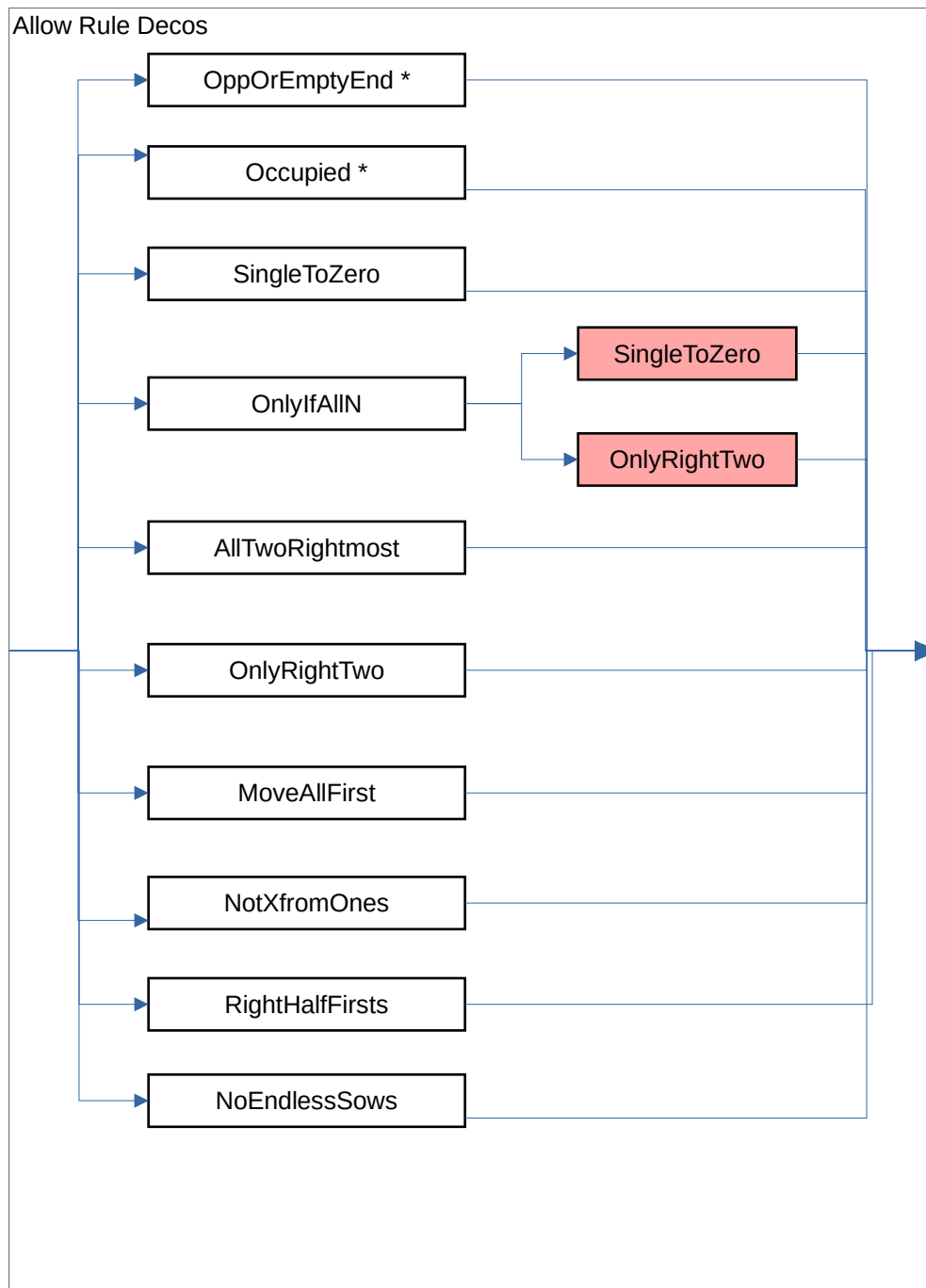
Notes:
Some allow rule decos are shown more than once for clarity.
* Simulates some portion of moves to determine allowables

Get Moves Decorators and Chain
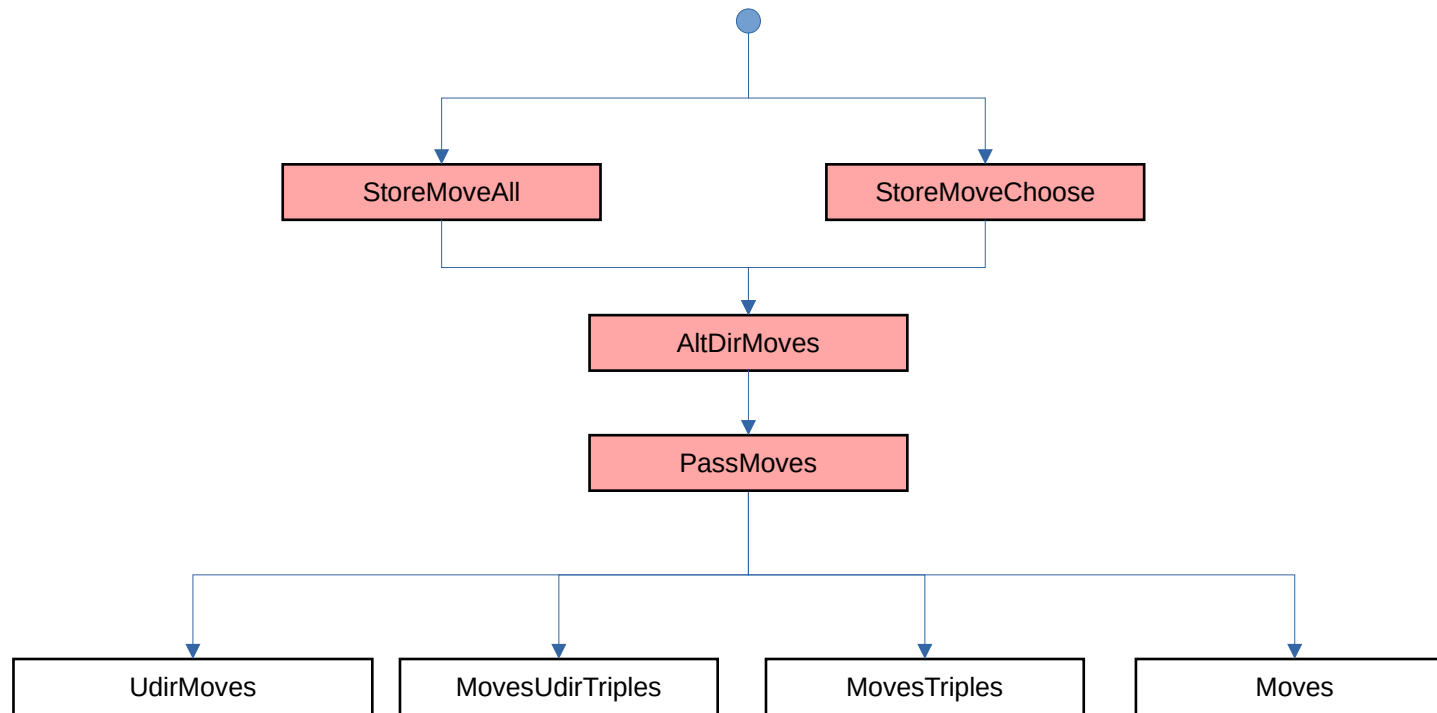
```
                              ●
             ┌────────────────┴────────────────┐
             ▼                                  ▼
    ┌─────────────────┐              ┌─────────────────────┐
    │  StoreMoveAll   │              │   StoreMoveChoose   │
    └─────────────────┘              └─────────────────────┘
             └────────────────┬────────────────┘
                              ▼
                    ┌─────────────────┐
                    │   AltDirMoves   │
                    └─────────────────┘
                              ▼
                    ┌─────────────────┐
                    │   PassMoves     │
                    └─────────────────┘
```
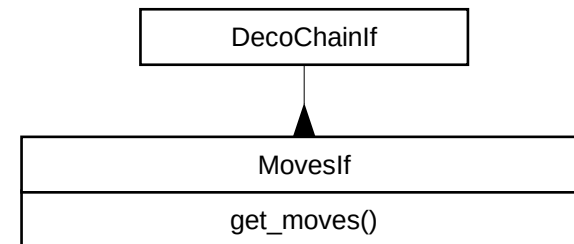
UdirMoves	MovesUdirTriples	MovesTriples	Moves

State variables read: Parameters:
    blocked   mlength
    board   mustpass
    owner   sow_direct
    starter   udir_holes
    store   udirect
    turn

┌─────────────────────┐
│   DecoChainIf   │
└─────────────────────┘
          ▲
┌─────────────────────┐
│   MovesIf   │
├─────────────────────┤
│   get_moves()   │
└─────────────────────┘

Get Direction Decorators and Chain

```
                                    ●
                                    │
        ┌───────────────────────────┴───────────────────────────┐
        ▼                                                        │
┌──────────────────┐  1 of (not both)                           │
│   UdirOtherDir    │◇─────────────────────┐                    ▼
└──────────────────┘                       │         ┌──────────────────┐
        │                                  │         │    PlayAltDir     │
 ┌──────┼──────┬──────────┬──────────┐     │         └──────────────────┘
 ▼      ▼      ▼          ▼           │              ┌────────┴────────┐
┌────────┐ ┌────────┐ ┌────────────┐ ┌──────────┐   ▼                 ▼
│ConstDir│ │SplitDir│ │CenterLineDir│ │EvenOddDir│ ┌──────────────┐ ┌──────────┐
└────────┘ └────────┘ └────────────┘ └──────────┘ │UdirTripleDir │ │ UdirDir  │
                                            │      └──────────────┘ └──────────┘
                                            │            ▲              ▲
                                            └────────────┴──────────────┘
```
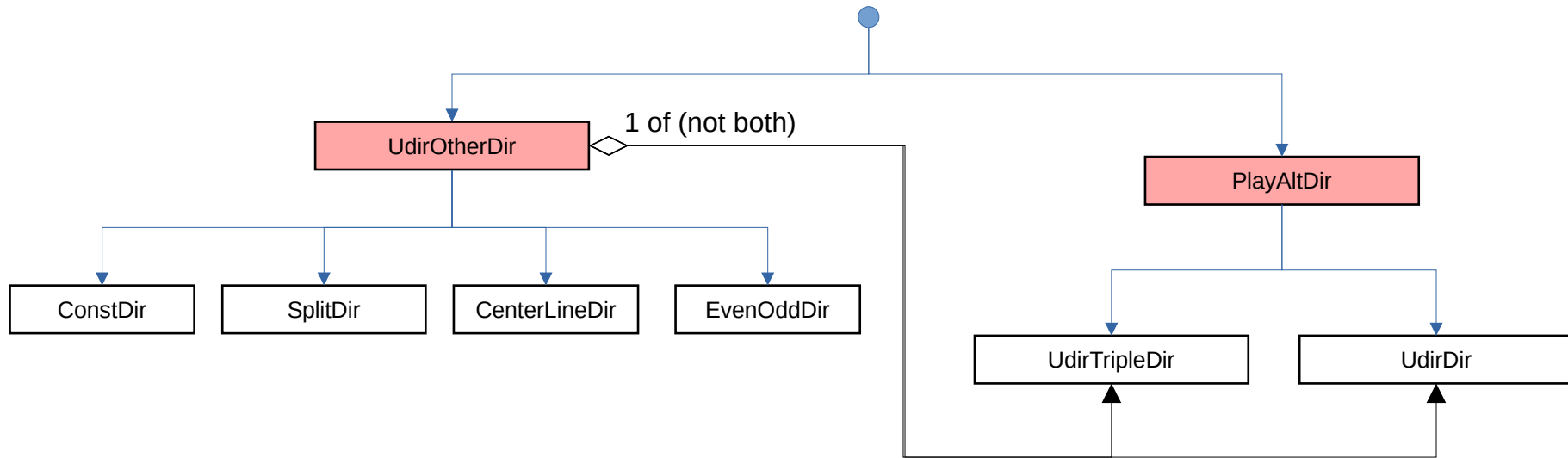
State variables read: Parameters:
     mcount   no_sides
     turn   sow_direct
                                     udir_holes
                                     udirect

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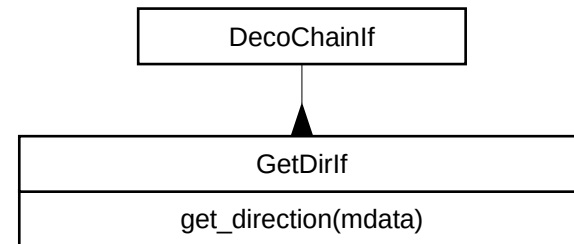
Incrementer Decorators and Chains

```
┌─────────────────────┐
│     DecoChainIf      │
└─────────────────────┘
           ▲
┌─────────────────────┐
│     Incrementer      │
├─────────────────────┤
│ incr(loc, direct, turn, start) │
└─────────────────────┘
      ▲            ▲
┌──────────────┐  ┌──────────────────┐
│ MapIncrement │  │ MapStoresIncrement│
├──────────────┤  ├──────────────────┤
│ incr(loc, direct, turn, start) │  │ incr(loc, direct, turn, start) │
└──────────────┘  └──────────────────┘
```

State variables read: Parameters:
        blocked   blocks
                                     skip_start
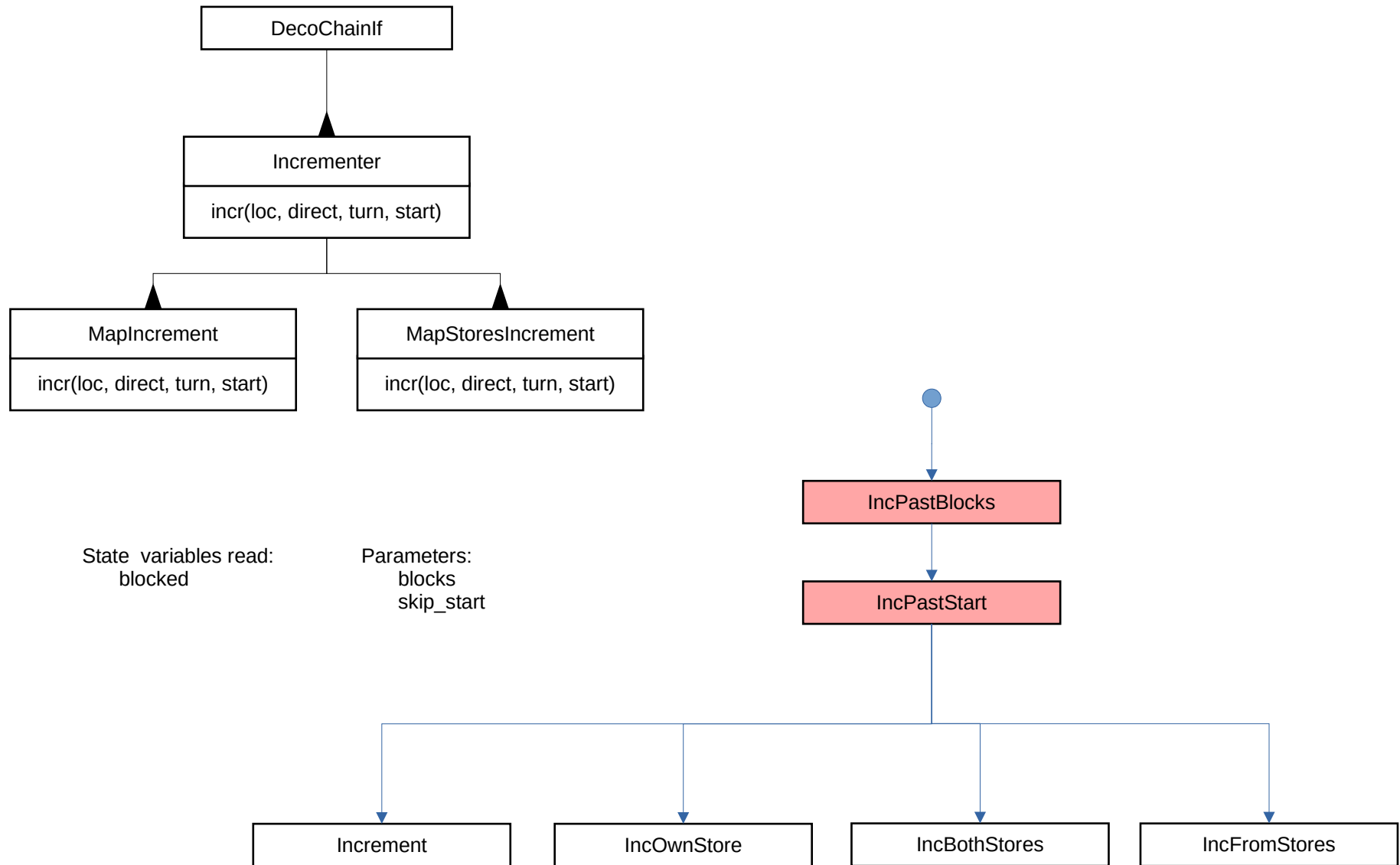
MakeChild Decorator and Chain

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

Parameters:
- child_cvt
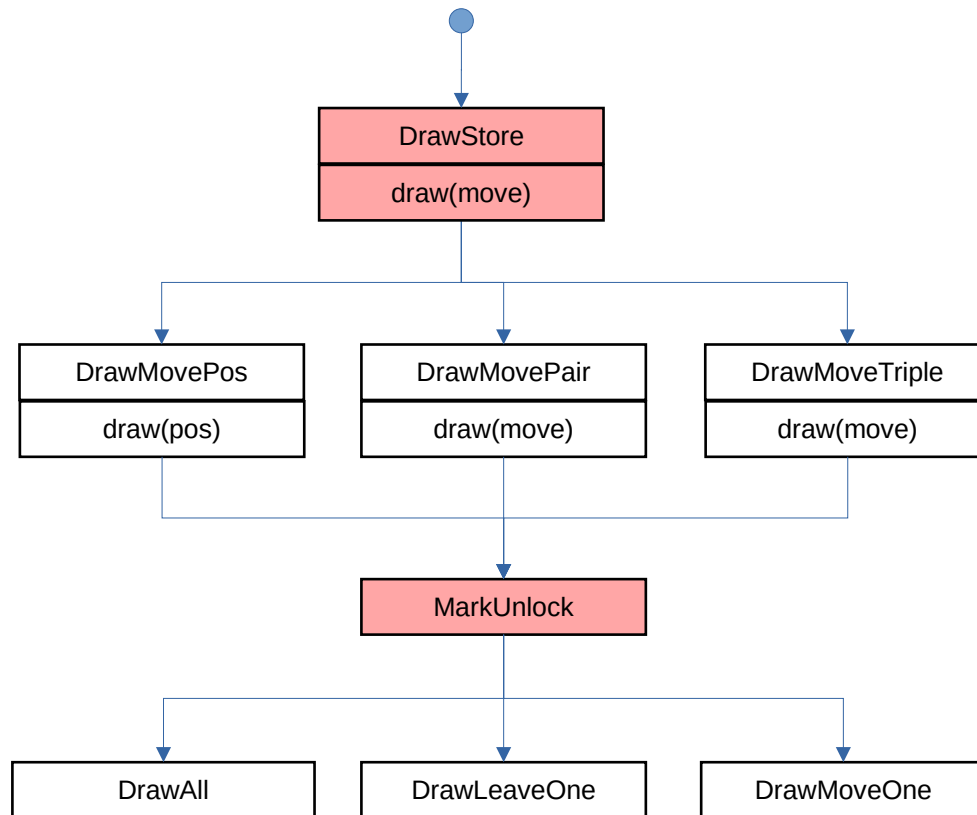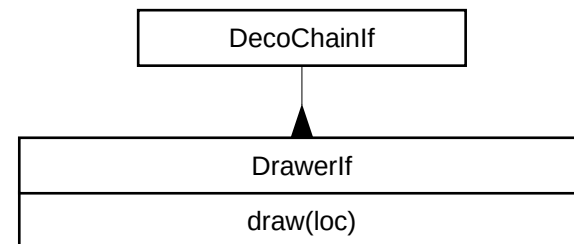- child_locs
- child_rule
- child_type

DecoChainIf

MakeChildIf

test(mdata)

NoChildren

NotInhibitedNoStore

Child Locs
(animation msgs)

NotSymOpp

NotFacing

ChildLocOk

Child Rule

OppSideChild

OwnSideChild

NotWithOne

Not1stOppWithOne

OppOwnerChild

OwnOwnerChild

BullChild

OneChild

QurChild

RamChild

BaseChild

Draw Decorators and Chain

```
DrawStore
────────────
draw(move)
```

```
DrawMovePos          DrawMovePair          DrawMoveTriple
────────────         ────────────          ────────────
draw(pos)            draw(move)            draw(move)
```
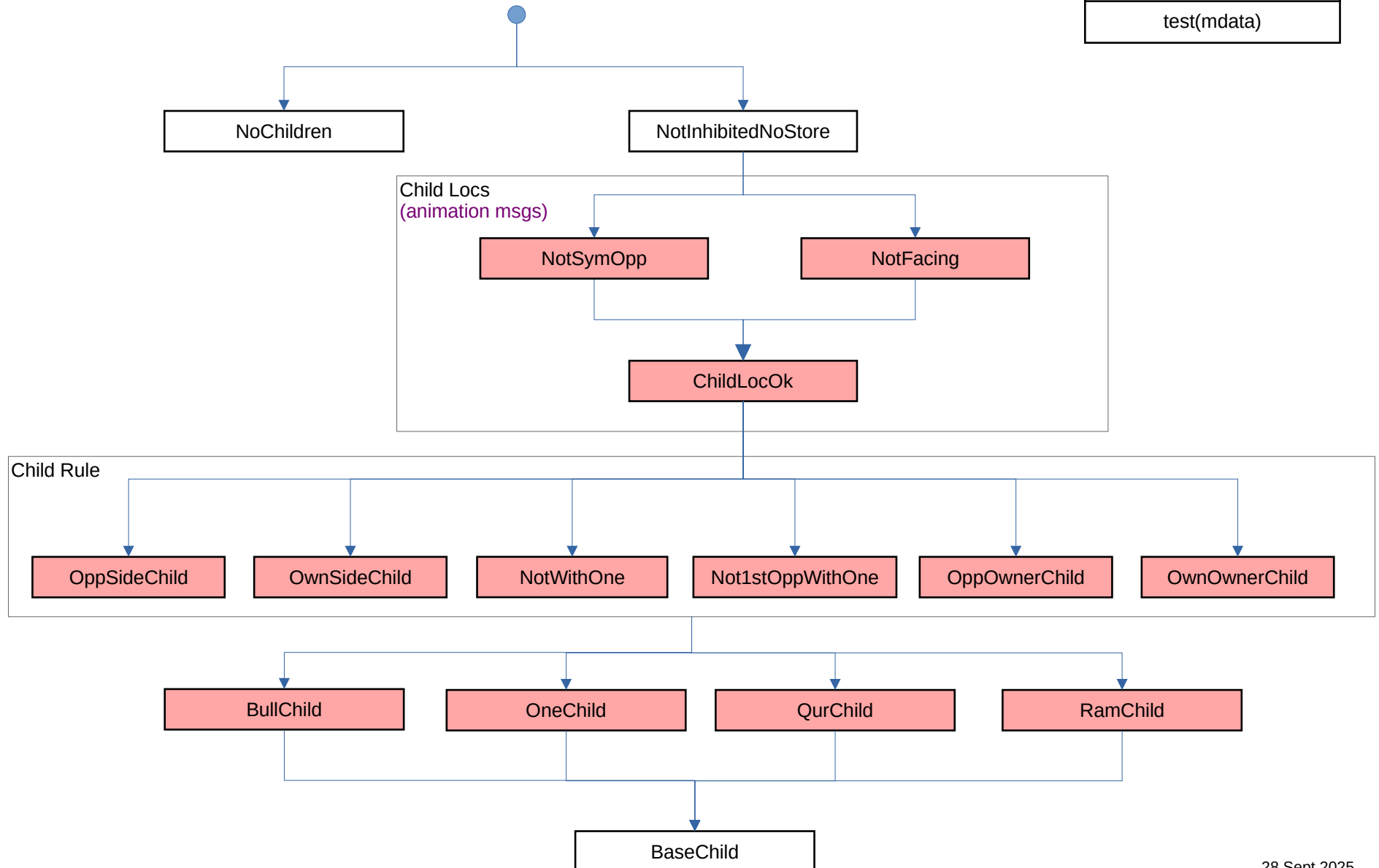
The drawer converts
the move into board
location.

```
MarkUnlock
```

```
DrawAll          DrawLeaveOne          DrawMoveOne
```

State variables: Parameters:
    Read: allow_rule
       turn   mlength
    Changed: move_one
       board   moveunlock
       unlocked   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) │
                                            │     └───────────────────────┘
          ┌───────────────────────┐        │
          │   SowPresSowerMixin   │        │     ┌───────────────────────┐
          ├───────────────────────┤        └────▶│     MlapEndOpIf       │
          │        sower          │              ├───────────────────────┤
          └───────────────────────┘              │     do_op(mdata)      │
                                                 └───────────────────────┘
```

Used for prescribed sowers that use another deco for the sowing.

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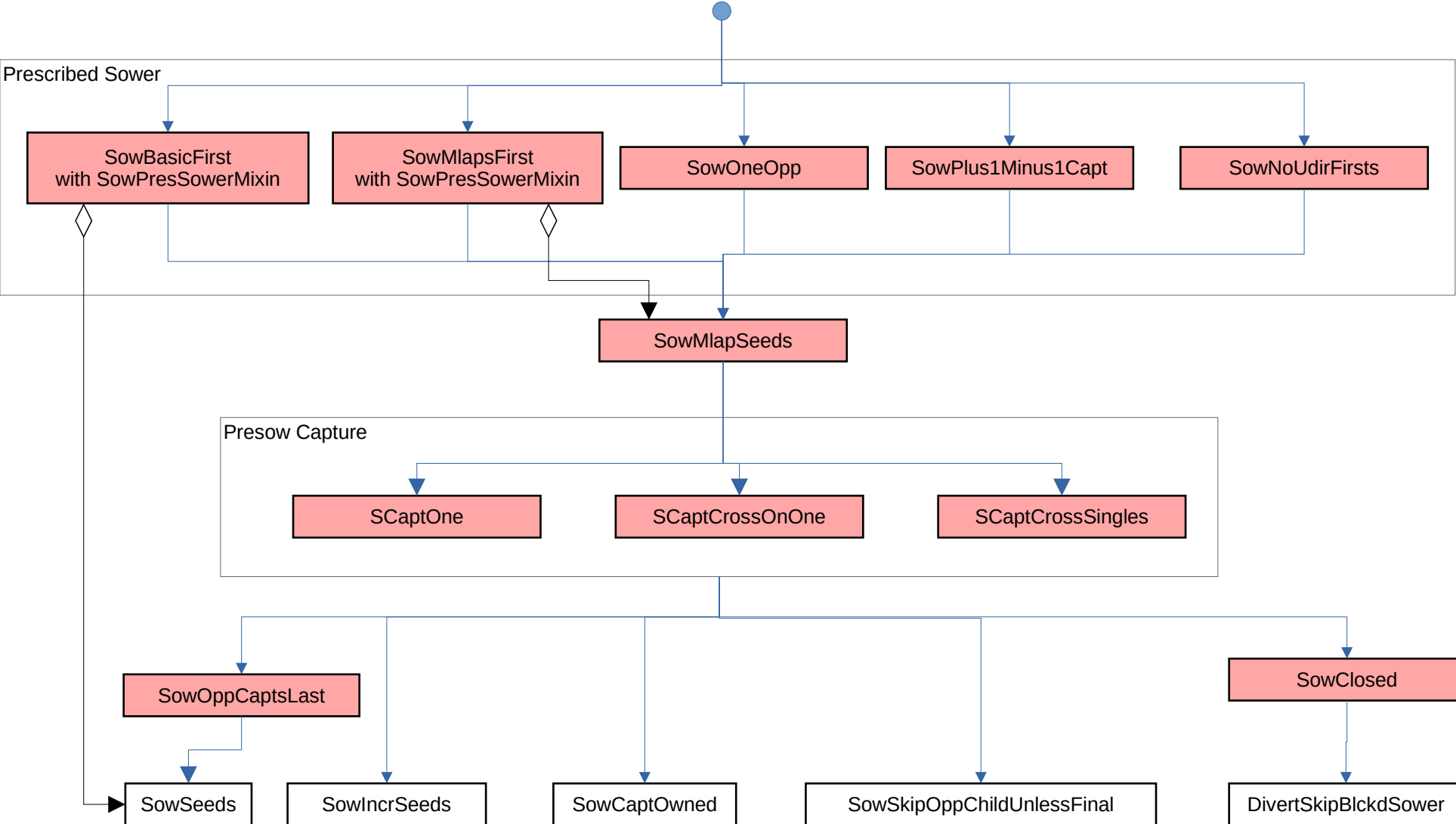
Parameters:
  capt_max
  capt_min
  capt_on
  child_type
  crosscapt
  evens
  goal
  gparam_one
  mlaps
  prescribed
  presowcapt
  sow_direct
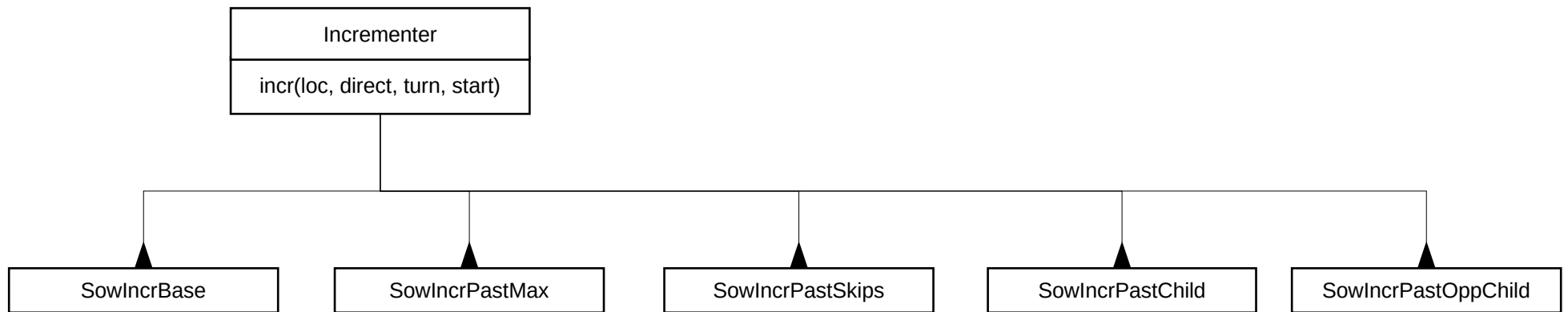 sow_own_store
  sow_param
  sow_rule
  visit_opp

Mlap Op

Not a deco chain.

```
          ┌─────────────┐   ┌─────────────┐   ┌─────────────┐
          │    NoOp     │   │   CloseOp   │   │  DirChange  │
          └─────────────┘   └─────────────┘   └─────────────┘
```

Sower Deco Chain

Prescribed Sower

SowBasicFirst
with SowPresSowerMixin

SowMlapsFirst
with SowPresSowerMixin

SowOneOpp

SowPlus1Minus1Capt

SowNoUdirFirsts

SowMlapSeeds

Presow Capture

SCaptOne

SCaptCrossOnOne

SCaptCrossSingles

SowOppCaptsLast

SowClosed

SowSeeds

SowIncrSeeds

SowCaptOwned

SowSkipOppChildUnlessFinal

DivertSkipBlckdSower

Incrementer Decos used with SowIncrSeeds

```
┌─────────────────────────────────┐
│         Incrementer             │
├─────────────────────────────────┤
│  incr(loc, direct, turn, start) │
└─────────────────────────────────┘
```

SowIncrBase	SowIncrPastMax	SowIncrPastSkips	SowIncrPastChild	SowIncrPastOppChild

Lap Continuer Deco Chain and Mlap Operation

Lapper ●

StopRepeatTurn

StopNotN StopLessN StopNotSide StopStore MustVisitOpp

StopOnChild

ContIfXCapt ContIfBasicCapt ContWithCaptSeeds GapNextCapt StopCaptureSimul StopCaptureSeeds

StopNoOppSeeds

StopSingleSeed StopXCapt

StopMakeChild

DivertBlckdLapper NextLapCont LapContinue
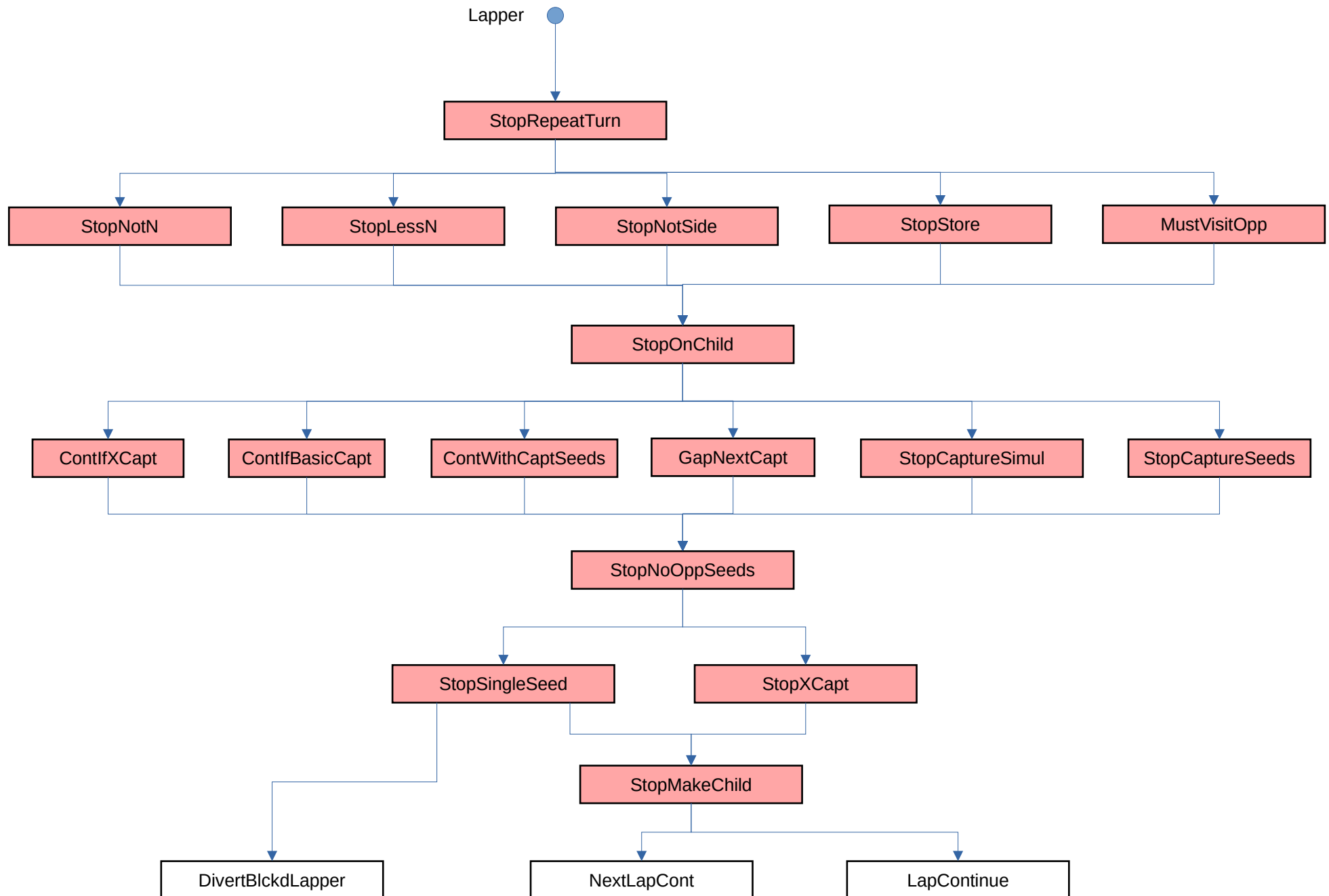
Capt Ok Decorators and Chains

capt_basic

capt_check

DecoChainIf

CaptOkIf

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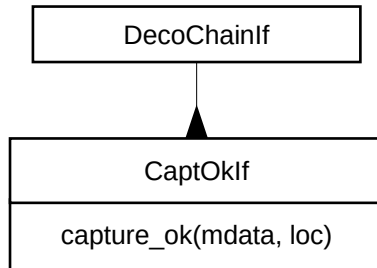
State variables read:
  board
  child
  turn
  unlocked

Parameters:
  capt_max
  capt_min
  capt_on
  capt_side
  moveunlock

capt_basic

CaptNotStoreNeedSeeds

CaptUnlocked

CaptUnlocked

CaptSideOk

CaptMax

CaptMin

CaptEvens

CaptOn

CaptTrue

capt_check

CaptNotStoreNeedSeeds

CaptUnlocked

CaptUnlocked

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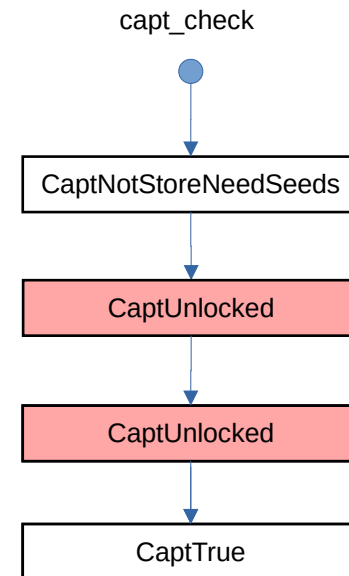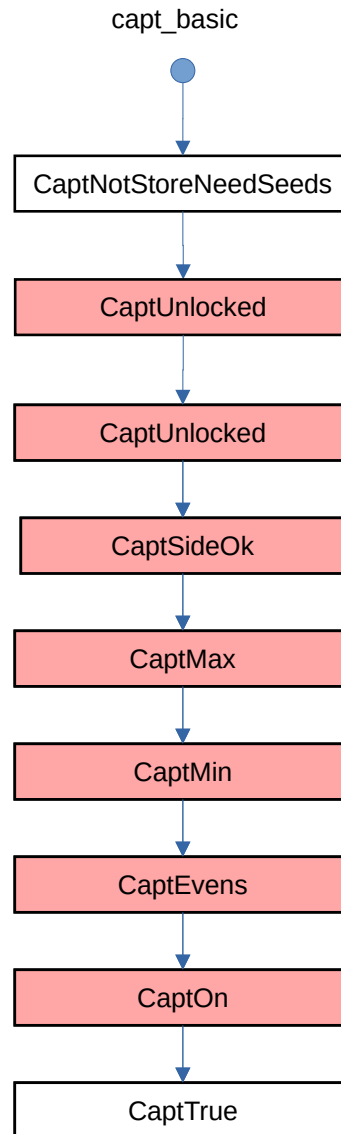
These are effectively ANDed.
If any deco condition is false, it
returns false, otherwise it calls
down the deco chain.

Capturer Decorators and Chain

DecoChainIf

CaptMethodIf

do_captures(mdata)

NoStoreCapt

RepeatTurn

NotInhibited

NoSingleSeedCapt

CaptureToChild

GrandSlam Decos

Picker Decos

Child Decos

CaptOppDir

CaptBothDir

CaptMultiple

PickCross

Base Capturer Decos
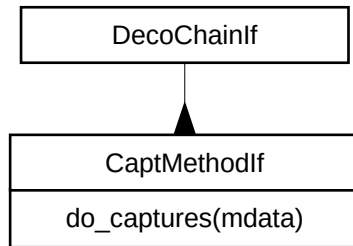
State variables
   Reads

      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
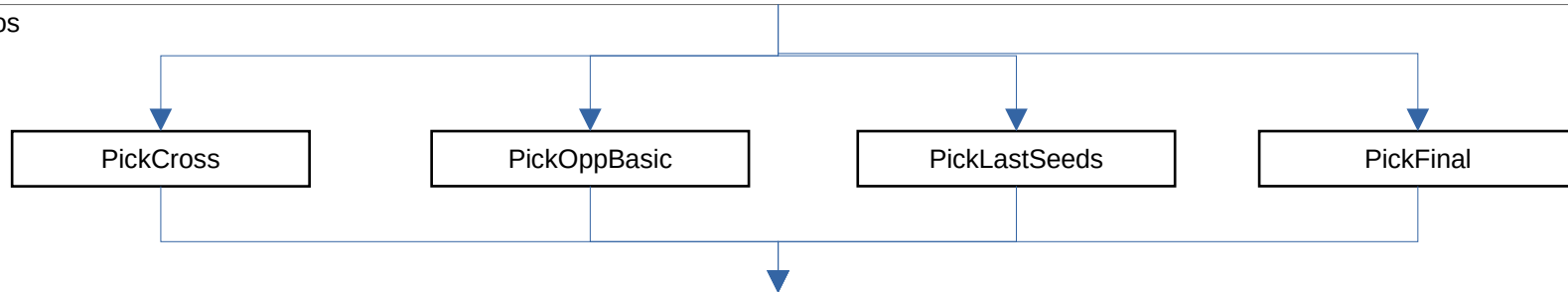   round_fill
      xc_sown
      xcpickown
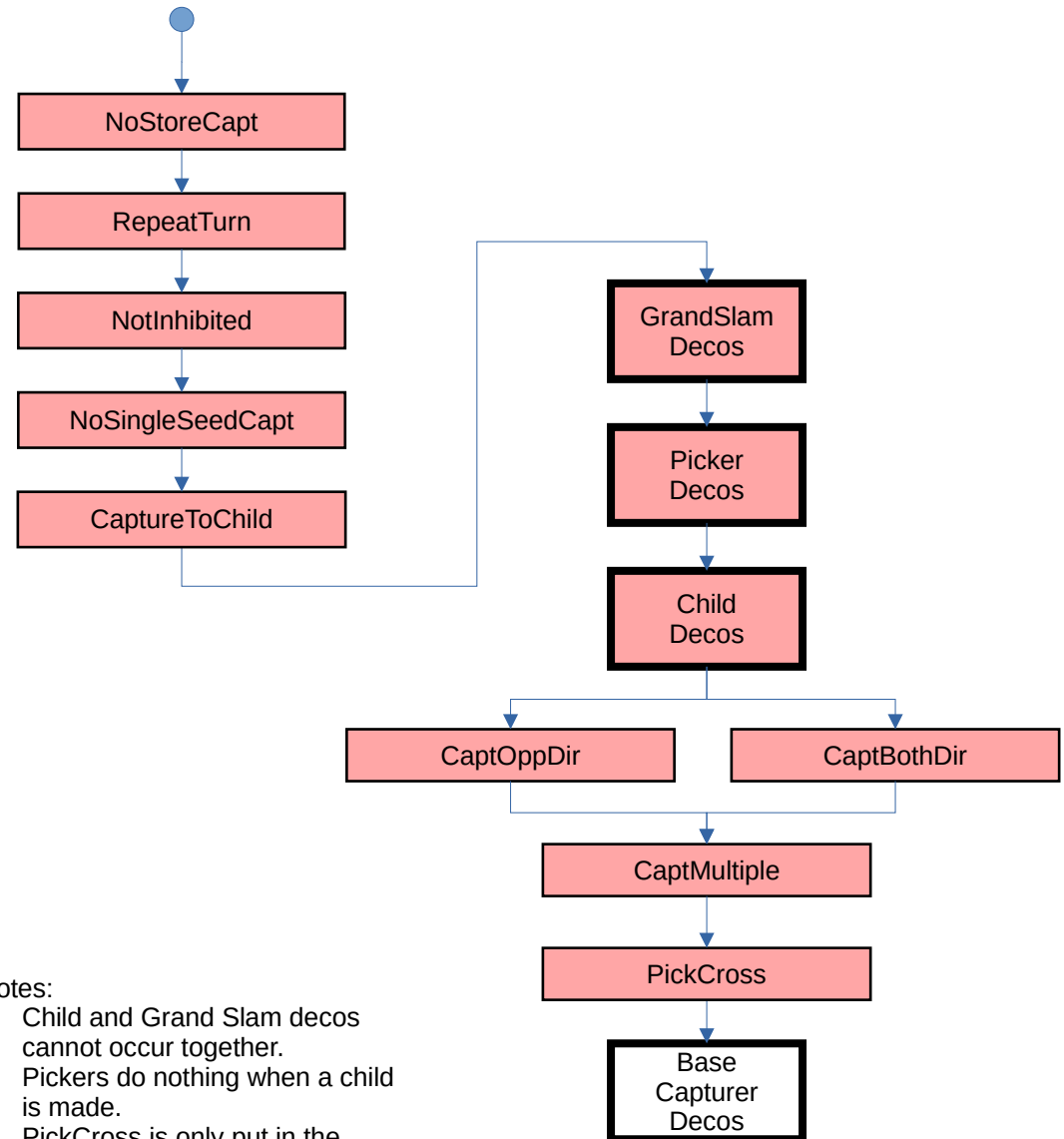
Notes:
- Child and Grand Slam decos cannot occur together.
- Pickers do nothing when a child is made.
- PickCross is only put in the deco chain once, either in Picker Decos or after CaptMultiple.
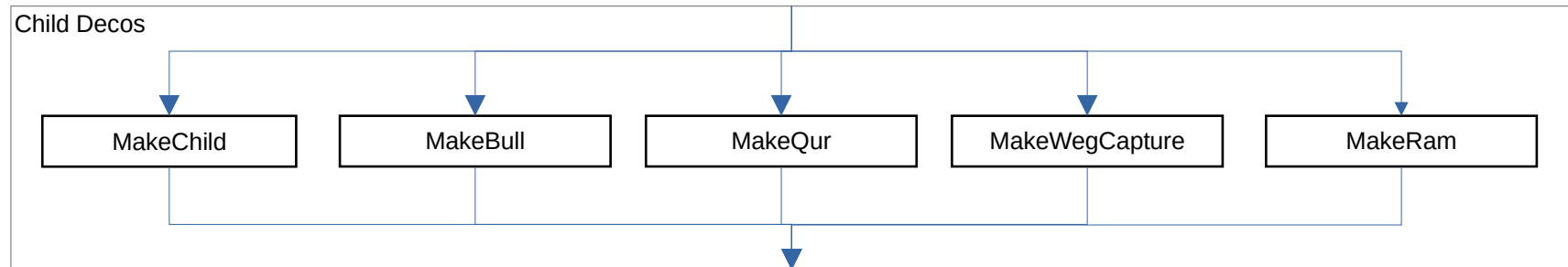
Capturer Deco Chains
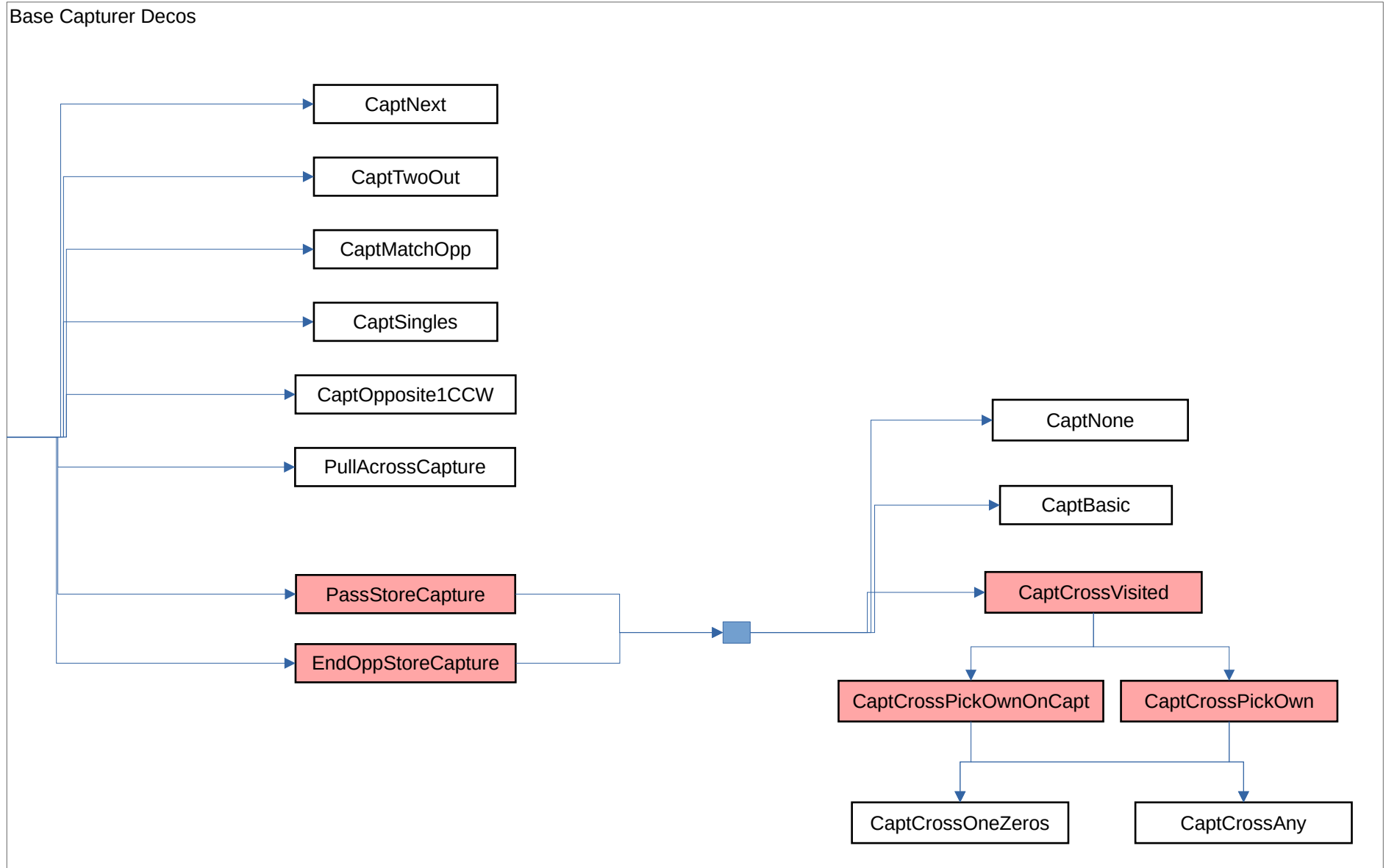# (1 of 2)

Grandslam Decos
(animation msgs)

GrandSlamCapt

GSNone	GSKeep	GSOppGets	GSLegalShare

Picker Decos

PickCross	PickOppBasic	PickLastSeeds	PickFinal

Child Decos

MakeChild	MakeBull	MakeQur	MakeWegCapture	MakeRam

Base Capturer Decos

CaptNext

CaptTwoOut

CaptMatchOpp

CaptSingles

CaptOpposite1CCW

CaptNone

PullAcrossCapture

CaptBasic

PassStoreCapture

CaptCrossVisited

EndOppStoreCapture

CaptCrossPickOwnOnCapt

CaptCrossPickOwn

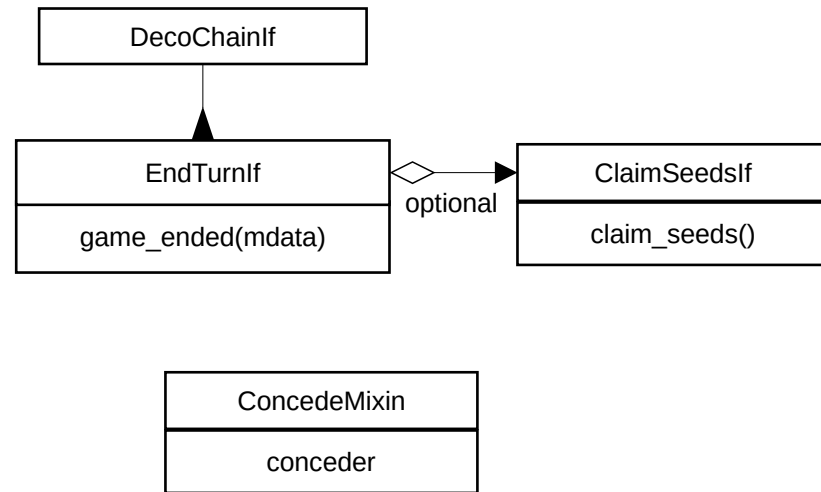CaptCrossOneZeros

CaptCrossAny

Ender & Quiter Decorators and Chains
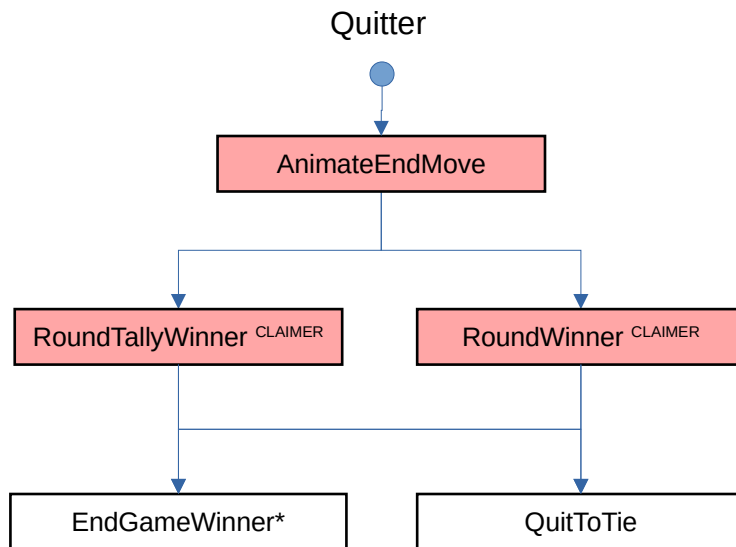
State variables:
Reads:
child
owner
turn
Changes:
board
store

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()     │
└──────────────────────┘        └──────────────────────┘
```

```
┌──────────────────────┐
│    ConcedeMixin      │
├──────────────────────┤
│      conceder        │
└──────────────────────┘
```

Used for enders that use a
different criteria for ending when
the user concedes a game.

Quitter

```
                    ●
                    │
                    ▼
        ┌──────────────────────┐
        │   AnimateEndMove     │
        └──────────────────────┘
            │              │
            ▼              ▼
┌────────────────────┐  ┌────────────────────┐
│RoundTallyWinner CLAIMER│  │ RoundWinner CLAIMER │
└────────────────────┘  └────────────────────┘
            │              │
            ▼              ▼
┌────────────────────┐  ┌────────────────────┐
│   EndGameWinner*   │  │     QuitToTie      │
└────────────────────┘  └────────────────────┘
```

Note:
* For EndGameWinner in the quitter: a claimer,
taker or divvier is selected based on the quitter,
child_type and store properties (see next page).

Ender Deco Chain

Ender

○

AnimateEndMove

| Non-Elim Ender Decos | Eliminate Ender Decos |
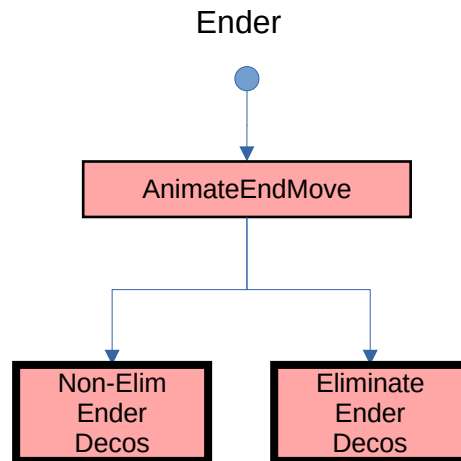
CLAIMER: counts owned seeds but does not move them:
ClaimSeeds
ChildClaimSeeds
ClaimOwnSeeds
ClaimBoardSeeds

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

DIVVY: unclaimed seeds are split between players:
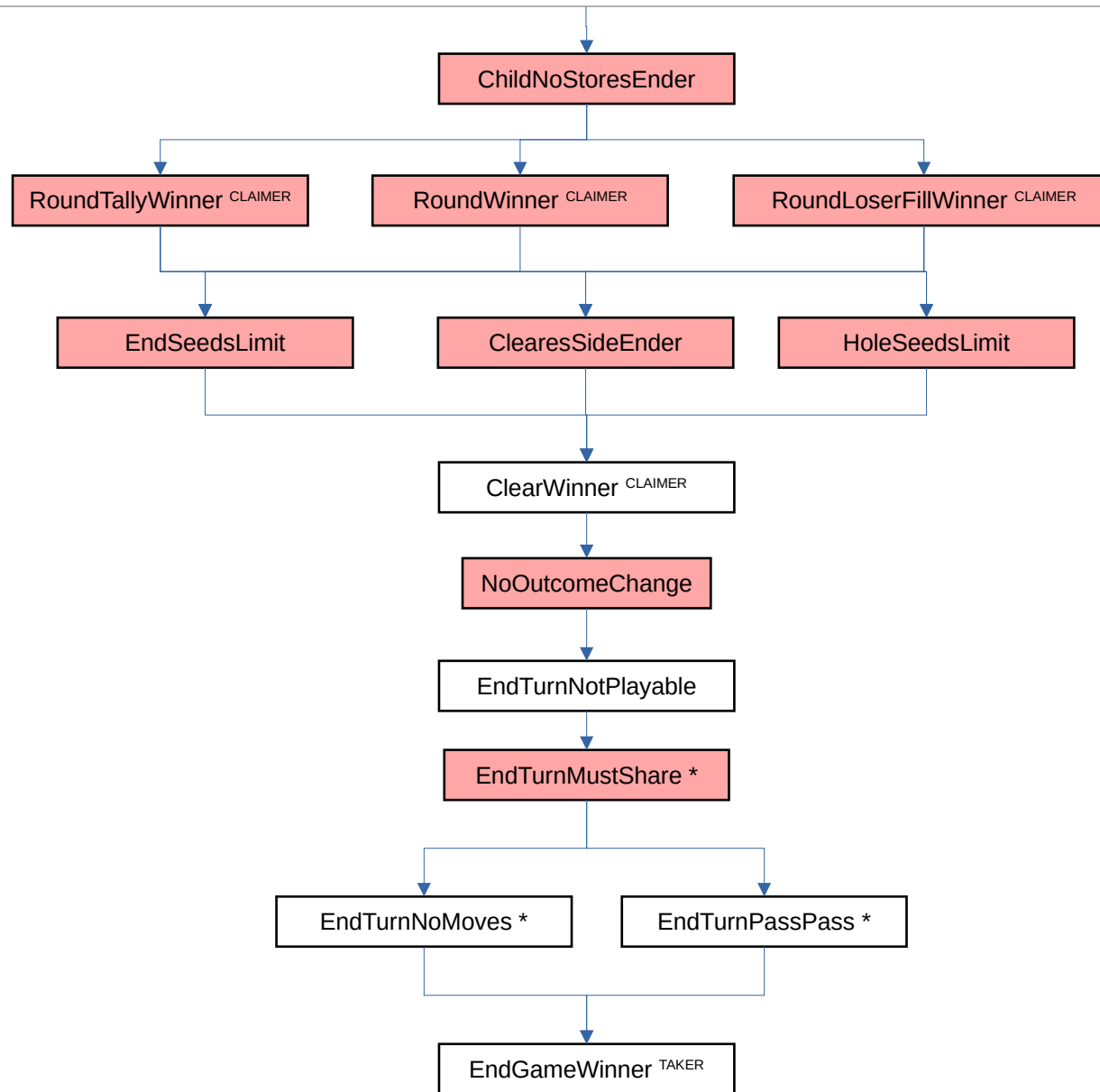DivvySeedsStores
DivvySeedsChildOnly

Note:
* Uses get_allowable_holes

Non-Eliminate Ender Deco Chain

Non-Elim Ender Deco

ChildNoStoresEnder

RoundTallyWinner CLAIMER

RoundWinner CLAIMER

RoundLoserFillWinner CLAIMER

EndSeedsLimit

ClearesSideEnder

HoleSeedsLimit

ClearWinner CLAIMER

NoOutcomeChange

EndTurnNotPlayable

EndTurnMustShare *

EndTurnNoMoves *

EndTurnPassPass *

EndGameWinner TAKER

Eliminate Ender Deco Chains

Eliminate Ended Deco

RoundTallyWinner ^{CLAIMER}

DepriveEndGame
with ConcedeMixin

ImmobilizeEndGame *
with ConcedeMixin

ClearSeedsEndGame
with ConcedeMixin

EndTurnNoMoves *

ConcedeDepImm

ConcedeClear