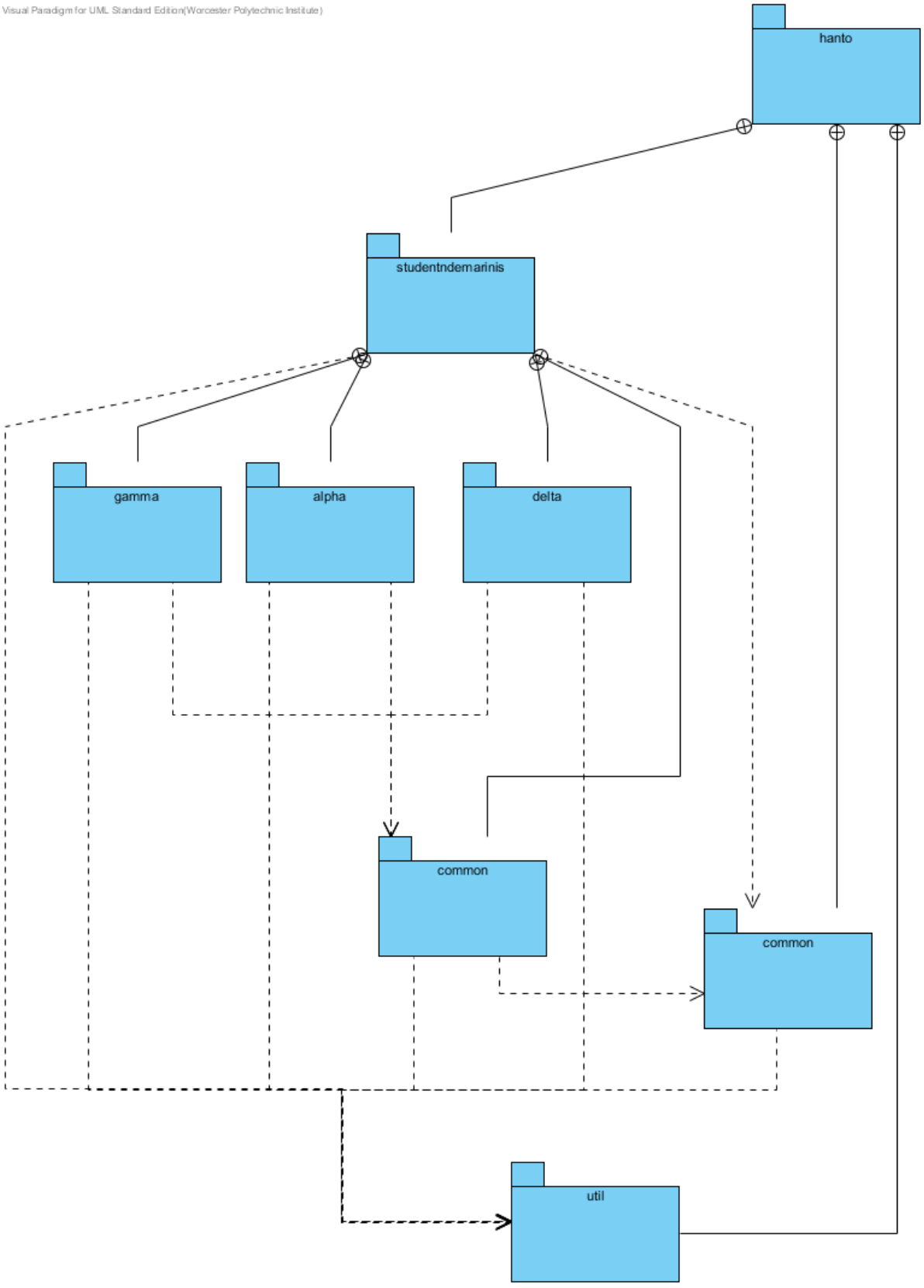
DeltaHanto Design Report

CS4233: Object-Oriented Analysis and Design

Nicholas DeMarinis

Table of Contents

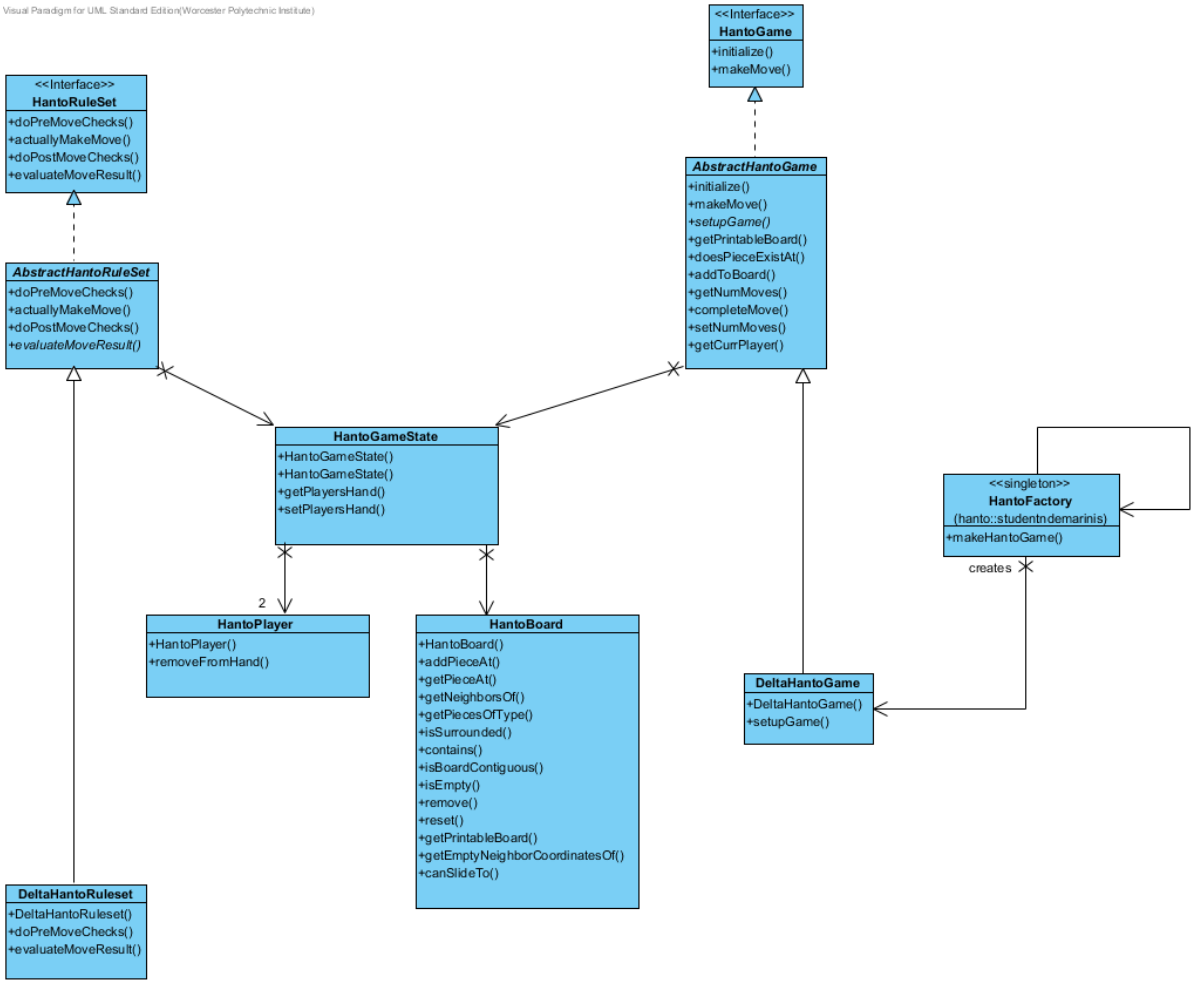
DeltaHanto Package Diagram



Summary

Name	Documentation
hanto	This is the top-level package for all of the Hanto source.
studentndemarinis	This package encapsulates my implementation of the HantoGame.
gamma	This package contains all classes necessary for the realization of GammaHanto, including its specific initialization methods and rules.
alpha	This package contains all classes necessary for the realization of AlphaHanto.
delta	This package contains all classes necessary for the realization of DeltaHanto, including its specific initialization methods and rules.
common	This package encapsulates all of the elements common to the HantoGames, such as the factory, abstract classes for the rules, and the board.
common	This package contains the high-level common elements, like the main game interface, as provided.
util	This package contains the ancillary enumerations for defining Hanto game elements, like the pieces, as given.

DeltaHanto Class Diagram



Summary

Name	Documentation
HantoGame	The HantoGame interface is the primary interface between the student's code and any external (non-student written) code. Every version of Hanto will have a realization of the HantoGame interface. @author gpollice @version Jan 12, 2013
HantoRuleSet	This interface represents methods required for a HantoRuleset. These methods will be called by AbstractHantoGame to provide a common implementation for making and verifying moves. @author ndemarinis @version Jan 31, 2013
AbstractHantoGame	This abstract class encapsulates the basic functionality for a HantoGame, including initialization. Lots of public methods exist here to expose extra implementation for testing purposes. @author ndemarinis @version Jan 31, 2013

AbstractHantoRuleSet	This abstract class encapsulates the common functionality for a HantoRuleSet. It contains default implementations for some of the major methods as well as protected rule methods that are common to the rulesets of Alpha, Gamma, and Delta Hanto. @author ndemarinis @version 4 February 2013
HantoGameState	Encapsulation of state information for a HantoGame. This includes all attributes necessary for the game and rule logic. @author ndemarinis @version Feb 7, 2013
HantoFactory	Factory for creating Hanto Games. Currently supports Alpha, Gamma, and Delta Hanto. @author ndemarinis @version Feb 9 2013
HantoPlayer	This class provides state information for each player, maintaining the types and counts of each piece in their hand. @author ndemarinis @version Jan 23, 2012
HantoBoard	This class represents the hexagonal Hanto Board. It maintains the pieces in a Map and provides methods for performing operations that require traversing the hex grid. @author ndemarinis @version Feb 7, 2013
DeltaHantoGame	This class is a concrete realization of the game for DeltaHanto. It provides the necessary initialization methods for DeltaHanto and relies on the ruleset and abstract classes to handle the rest of the implementation. @author ndemarinis @version Feb 9, 2013
DeltaHantoRuleset	This class is a concrete realization of the ruleset for Delta Hanto. It provides all of the rule methods specific to DeltaHanto. @author ndemarinis @version Feb 9, 2013

Details

HantoGame

Operations

public initialize (firstPlayer : HantoPlayerColor) : void

Parameters	**firstPlayer**	
	Documentation	the (color of) the player who moves first. If this is null, then the default player, as specified by the rule set, moves first.
	Multiplicity	Unspecified
	Type	 HantoPlayerColor
	Direction	inout
	Java Detail	N/A
Documentation	Initialize the game for play. While the constructor may already initialize the game, this method can be called any time. It will (re)initialize the game and make it ready to play. If the game is already initialized, or in progress, then this method will reset the game to its initial state.	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if any errors occur during initialization (such as the specified player violates the rules specified in the rule set).	
Query	false	

public makeMove (pieceType : HantoPieceType, from : HantoCoordinate, to : HantoCoordinate) : MoveResult		
Parameters	**pieceType**	
	Documentation	the piece type that is being moved
	Multiplicity	Unspecified
	Type	 HantoPieceType
	Direction	inout
	Java Detail	N/A
	from	
	Documentation	the coordinate where the piece begins. If the coordinate is null, then the piece begins off the board (that is, it is placed on the board in

		this move).
	Multiplicity	Unspecified
	Type	HantoCoordinate
	Direction	inout
	Java Detail	N/A
	to	
	Documentation	the coordinated where the piece is after the move has been made.
	Multiplicity	Unspecified
	Type	HantoCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	This method executes a move in the game. It is called for every move that must be made.	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	the result of the move @throws HantoException if there are any problems in making the move (such as specifying a coordinate that does not have the appropriate piece, or the color of the piece is the color of the player who is moving.	
Query	false	

HantoRuleSet

Operations

public doPreMoveChecks (piece : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void		
Parameters	**piece**	
	Documentation	The piece to add at the new location
	Multiplicity	Unspecified
	Type	HantoPieceType
	Direction	inout

	Java Detail	N/A
	from	
	Documentation	Source location of said piece, null if piece is not on the board
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
	to	
	Documentation	Destination coordinate of piece after the move
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Perform any checks that can happen before a move is made, throws a HantoException if the move is invalid.	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if proposed move violates a rule.	
Query	false	

public actuallyMakeMove (type : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void		
Parameters	**type**	
	Documentation	Piece type to place at the destination
	Multiplicity	Unspecified
	Type	HantoPieceType
	Direction	inout
	Java Detail	N/A

	from	
	Documentation	Source coordinate of the piece, null if piece is not on the board
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
	to	
	Documentation	Destination coordinate of the piece
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Make a move, regardless of whether or not it is valid. Any piece currently at the source and destination locations are REMOVED when this method is called	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if an error occurs during the move	
Query	false	

public doPostMoveChecks (to : HexCoordinate) : void		
Parameters	**to**	
	Documentation	Destination coordinate of the piece after the move
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Perform any checks based on the location of a newly-moved piece, throws HantoException of the move is invalid.	
Static	false	

Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if the proposed move violates a rule
Query	false

public evaluateMoveResult () : MoveResult	
Documentation	Check conditions to determine if the game needs to end. This is used for returning the result of a recent move
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	MoveResult with based on the current board's conditions @throws HantoException on an invalid board configuration
Query	false

AbstractHantoGame

Attributes

protected : HantoGameState			
Type	 HantoGameState		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	N/A		

protected rules : HantoRuleSet			
Type	HantoRuleSet		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	rules		

protected state : HantoGameState			
Type	HantoGameState		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	state		

Operations

public initialize (firstPlayer : HantoPlayerColor) : void		
Parameters	firstPlayer	
	Multiplicity	Unspecified
	Type	HantoPlayerColor
	Direction	inout
	Java Detail	N/A

Documentation	Abstract HantoGame providing basic implementation
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public makeMove (pieceType : HantoPieceType, from : HantoCoordinate, to : HantoCoordinate) : MoveResult	
Parameters	**pieceType**
	Multiplicity: Unspecified
	Type: HantoPieceType
	Direction: inout
	Java Detail: N/A
	from
	Multiplicity: Unspecified
	Type: HantoCoordinate
	Direction: inout
	Java Detail: N/A
	to
	Multiplicity: Unspecified
	Type: HantoCoordinate
	Direction: inout
	Java Detail: N/A
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public setupGame () : void	
Documentation	Game-specific method for performing any necessary setup tasks,

	called by initialize().
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public getPrintableBoard () : String	
Documentation	Return a string representing the current state of the board, empty string if the board is empty.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public doesPieceExistAt (c : HantoCoordinate) : boolean		
Parameters	**c**	
	Documentation	coordinate to check for a piece TODO: Move this to the test harness? NOTE: this name makes sense to me. I don't understand how the suggestions in CodePro's audit rule could make more sense here.
	Multiplicity	Unspecified
	Type	 HantoCoordinate
	Direction	inout
	Java Detail	N/A
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	true if a piece exists on the board	

Query	false

public addToBoard (color : HantoPlayerColor, type : HantoPieceType, c : HantoCoordinate) : void	
Parameters	**color**
	Documentation: color of new piece
	Multiplicity: Unspecified
	Type: HantoPlayerColor
	Direction: inout
	Java Detail: N/A
	type
	Documentation: type of new piece
	Multiplicity: Unspecified
	Type: HantoPieceType
	Direction: inout
	Java Detail: N/A
	c
	Documentation: location of new piece
	Multiplicity: Unspecified
	Type: HantoCoordinate
	Direction: inout
	Java Detail: N/A
Documentation	Add a coordinate to the board TODO: Move this to the test harness
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public getNumMoves () : int	
Static	false

Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	the number of moves made in this game
Query	false

public completeMove () : void	
Documentation	Perform actions necessary to finish a move, committing it as valid Currently switches the current player and increments the total number of moves
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public setNumMoves (numMoves : int) : void		
Parameters	numMoves	
	Documentation	the number of moves to set
	Multiplicity	Unspecified
	Type	int
	Direction	inout
	Java Detail	N/A
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

public getCurrPlayer () : HantoPlayerColor	
Static	false
Leaf	false

Upper	1
Ordered	false
Unique	true
Return Type Documentation	the current player up for a move
Query	false

AbstractHantoRuleSet

Attributes

private NUM_MOVES_PRE_BUTTERFLY : int			
Initial Value	3		
Type	int		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

protected null : HantoGameState			
Type	HantoGameState		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	N/A		

protected state : HantoGameState			
Type	HantoGameState		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	state		

Operations

public doPreMoveChecks (piece : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void

Parameters

piece	
Multiplicity	Unspecified
Type	HantoPieceType
Direction	inout
Java Detail	N/A

from	
Multiplicity	Unspecified
Type	HexCoordinate
Direction	inout
Java Detail	N/A

to	
Multiplicity	Unspecified
Type	HexCoordinate
Direction	inout
Java Detail	N/A

Documentation	Perform checks that must take place before a move. See HantoRuleSet for details
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public actuallyMakeMove (type : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void	
Parameters	**type**
	Multiplicity: Unspecified Type: HantoPieceType Direction: inout Java Detail: N/A
	from
	Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A
	to
	Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A
Documentation	Perform a move for real See HantoRuleSet for details.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public doPostMoveChecks (to : HexCoordinate) : void		
Parameters	to	
	Documentation	Destination coordinate
	Multiplicity	Unspecified
	Type	 HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Perform any checks that must take place after a move	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if any of these rules have been violated	
Query	false	

public evaluateMoveResult () : MoveResult	
Documentation	Determine the result of a move based on the game's rules. Must be overridden by concrete realization.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

protected verifyGameIsNotOver () : void	
Documentation	Verify the game is not over
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if the game is over
Query	false

protected verifySourceAndDestinationCoords (from : HexCoordinate, to : HexCoordinate) : void	
Parameters	**from**
	Documentation: Source coordinate Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A
	to
	Documentation: Destination coordinate Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A
Documentation	Verify the source and destination coordinates exist. If a source is provided, it must exist on the board; a destination coordinate must exist for a valid move.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if either of these conditions have been violated
Query	false

protected verifyMoveIsLegal (from : HexCoordinate, to : HexCoordinate) : void	
Parameters	**from**
	Documentation: Source coordinate of move to verify Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A

	to	
	Documentation	Destination coordinate of move to verify
	Multiplicity	Unspecified
	Type	 HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Verify a move is legal, meaning that the first piece must be at the origin, players can only move pieces of their own color, and that the destination coordinate must be empty	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if any of these conditions have been violated	
Query	false	

protected verifyBoardIsContiguous () : void	
Documentation	Verify all of the pieces on the board are in a single contiguous grouping.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if any pieces are separated from the group
Query	false

protected determineIfGameHasEnded (res : MoveResult) : void		
Parameters	**res**	
	Documentation	Result to determine game's ending state
	Multiplicity	Unspecified

	Type	MoveResult
	Direction	inout
	Java Detail	N/A
Documentation	Set whether or not the game has ended based on the current move result	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

protected winIfButterflyIsSurrounded () : MoveResult	
Documentation	Check if a player has won by surrounding their opponent's butterfly. If both butterflies are surrounded, it's a DRAW.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	winning player if they have surrounded their opponent's butterfly, DRAW if both are surrounded, OK if none of these conditions have been met
Query	false

protected verifyButterflyHasBeenPlacedByFourthTurn (piece : HantoPieceType) : void		
Parameters	**piece**	
	Documentation	The piece involved in the move
	Multiplicity	Unspecified
	Type	HantoPieceType
	Direction	inout
	Java Detail	N/A
Documentation	Ensure that a butterfly must be placed by the fourth term, as the rules specify. Therefore, a player moving on/after the fourth turn with no butterfly on the board MUST place their butterfly.	
Static	false	

Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if trying to place a butterfly without one for that player on the board
Query	false

HantoGameState

Attributes

package numMoves : int			
Stereotypes	Property		
Type	int		
Allow Empty Name	false		
Getter	true	Setter	true
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

package gameOver : boolean			
Documentation	Player that making the current/next move		
Stereotypes	Property		
Type	boolean		
Allow Empty Name	false		
Getter	true	Setter	true
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		

Leaf	false

package currPlayer : HantoPlayerColor			
Documentation	Total number of moves elapsed in the game so far		
Stereotypes	Property		
Type	HantoPlayerColor		
Allow Empty Name	false		
Getter	true	Setter	true
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	currPlayer		

package null : HantoBoard			
Documentation	Collection of pieces representing the board for now		
Stereotypes	Property		
Type	HantoBoard		
Allow Empty Name	false		
Getter	true	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	N/A		

package null : HantoPlayer	
Documentation	Maintain the player's hands here (as separate objects for now)
Type	HantoPlayer

Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	N/A		

package resignee : HantoPlayerColor			
Documentation	Whether or not the game has ended Whether or not the current player resigned This is kind of gross, but it's simple. I like TDD. =)		
Stereotypes	Property		
Initial Value	null		
Type	 HantoPlayerColor		
Allow Empty Name	false		
Getter	true	Setter	true
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	resignee		

package board : HantoBoard	
Documentation	Collection of pieces representing the board for now
Stereotypes	Property

Type	HantoBoard		
Allow Empty Name	false		
Getter	true	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	board		

package bluePlayer : HantoPlayer			
Documentation	Maintain the player's hands here (as separate objects for now)		
Type	HantoPlayer		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	bluePlayer		

Operations

public HantoGameState (startingPlayer : HantoPlayerColor, startingHand : java.util.Map)

Parameters

startingPlayer	
Documentation	Color of player to start
Multiplicity	Unspecified
Type	HantoPlayerColor
Direction	inout

	Java Detail	N/A
	startingHand	
	Documentation	Map of Piece->Count indicating how many of each piece the player has available for play.
	Multiplicity	Unspecified
	Template Type Bind Info	N/A
	Type	java.util.Map
	Direction	inout
	Java Detail	N/A
Documentation	Construct a state object for a HantoGame	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

public HantoGameState (startingPlayer : HantoPlayerColor)		
Parameters	**startingPlayer**	
	Documentation	Color of player to start
	Multiplicity	Unspecified
	Type	HantoPlayerColor
	Direction	inout
	Java Detail	N/A
Documentation	Construct a state object for a HantoGame	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

public getPlayersHand (p : HantoPlayerColor) : HantoPlayer		
Parameters	**p**	
	Documentation	The desired player

	Multiplicity	Unspecified
	Type	 HantoPlayerColor
	Direction	inout
	Java Detail	N/A
Documentation	Get the hand information for a given player	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	Hand information for that player	
Query	false	

public setPlayersHand (p : HantoPlayerColor, hand : java.util.Map) : void		
Parameters	**p**	
	Documentation	The desired player
	Multiplicity	Unspecified
	Type	 HantoPlayerColor
	Direction	inout
	Java Detail	N/A
	hand	
	Documentation	The hand to give the player
	Multiplicity	Unspecified
	Template Type Bind Info	N/A
	Type	java.util.Map
	Direction	inout
	Java Detail	N/A
Documentation	Set the hand information for a given player	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

HantoFactory

Attributes

private null : HantoFactory			
Stereotypes	Property		
Initial Value	null		
Type	HantoFactory		
Allow Empty Name	false		
Getter	true	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	N/A		

private instance : HantoFactory			
Stereotypes	Property		
Initial Value	null		
Type	HantoFactory		
Allow Empty Name	false		
Getter	true	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		
Referencing Association End	instance		

Operations

private HantoFactory ()	
Documentation	Factory for Hanto Games This constructor is private so this is a singleton
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public makeHantoGame (gameID : HantoGameID) : HantoGame		
Parameters	gameID	
	Documentation	Type of HantoGame to create
	Multiplicity	Unspecified
	Type	HantoGameID
	Direction	inout
	Java Detail	N/A
Documentation	Create an instance of a HantoGame based on the give game type. Only Alpha, and Gamma Hanto are currently supported.	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	Instance of the specified Hanto Game, null if game could not be made	
Query	false	

HantoPlayer

Attributes

private hand : java.util.Map	
Documentation	Map representing number of pieces of each type available for play
Stereotypes	Property

Template Type Bind Info	N/A		
Type	java.util.Map		
Allow Empty Name	false		
Getter	false	Setter	true
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

Operations

public HantoPlayer (hand : java.util.Map)		
Parameters	**hand**	
	Documentation	Map of pieces to the number of which the player has available to use
	Multiplicity	Unspecified
	Template Type Bind Info	N/A
	Type	java.util.Map
	Direction	inout
	Java Detail	N/A
Documentation	This class provides an abstraction for each player in GammaHanto. It maintains the types and numbers of pieces available for play.	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

public removeFromHand (type : HantoPieceType) : void		
Parameters	**type**	
	Documentation	Type to remove from their hand
	Multiplicity	Unspecified
	Type	HantoPieceType

	Direction	inout
	Java Detail	N/A
Documentation	Remove a piece of some type from the player's hand	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if player doesn't have any pieces of that type	
Query	false	

HantoBoard

Attributes

private MAX_NEIGHBORS : int			
Documentation	Maximum number of possible neighbors on a hex grid		
Initial Value	6		
Type	int		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

private pieces : java.util.Map			
Template Type Bind Info	N/A		
Type	java.util.Map		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		

Java Detail	N/A
Aggregation	None
Derived Union	false
Read Only	false
Leaf	false

Operations

public HantoBoard ()	
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public addPieceAt (p : HantoPiece, c : HexCoordinate) : void		
Parameters	p	
	Documentation	The piece to add
	Multiplicity	Unspecified
	Type	HantoPiece
	Direction	inout
	Java Detail	N/A
	c	
	Documentation	TODO
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Add a piece to the board Note that this method DOES NOT perform any error checking to ensure the piece is in a valid position	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	

Unique	true
Query	false

public getPieceAt (c : HexCoordinate) : HantoPiece		
Parameters	c	
	Documentation	Coordinate to search on the board
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Find a piece matching a given coordinate on the board	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	the piece matching that coordinate, null if none exists	
Query	false	

public getNeighborsOf (c : HexCoordinate) : java.util.Collection		
Parameters	c	
	Documentation	Coordinate to find neighbors
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Find neighboring pieces of a specific coordinate on the board	
Static	false	
Leaf	false	
Upper	1	
Template Type Bind Info	N/A	
Ordered	false	
Unique	true	
Return Type Documentation	Collection of neighbors, empty if none	

Query	false

public getPiecesOfType (t : HantoPieceType) : java.util.Collection		
Parameters	t	
	Documentation	The type for which to search on the board
	Multiplicity	Unspecified
	Type	HantoPieceType
	Direction	inout
	Java Detail	N/A
Documentation	Get pieces with a specific PieceType	
Static	false	
Leaf	false	
Upper	1	
Template Type Bind Info	N/A	
Ordered	false	
Unique	true	
Return Type Documentation	Collection of matching pieces	
Query	false	

public isSurrounded (c : HantoPiece) : boolean		
Parameters	c	
	Documentation	Coordinate to check
	Multiplicity	Unspecified
	Type	HantoPiece
	Direction	inout
	Java Detail	N/A
Documentation	Check if a coordinate is surrounded	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	true if the specified coordinate is surrounded	
Query	false	

public contains (c : HantoPlayerColor, t : HantoPieceType) : boolean	
Parameters	**c**
	Documentation: Color of piece to find Multiplicity: Unspecified Type: HantoPlayerColor Direction: inout Java Detail: N/A
	t
	Documentation: Type of piece to find Multiplicity: Unspecified Type: HantoPieceType Direction: inout Java Detail: N/A
Documentation	Check if a particular piece is somewhere on the board
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	true if at least one piece matching the type and color are on the board NOTE: this name makes sense to me. I don't understand how the suggestions in CodePro's audit rule could make more sense here.
Query	false

public isBoardContiguous () : boolean	
Documentation	Test if the pieces on the board are in a contiguous grouping, using BFS.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	true if pieces are in a contiguous grouping, false otherwise.
Query	false

public isEmpty () : boolean	
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	true if the board is empty, false otherwise
Query	false

public remove (p : HexCoordinate) : void		
Parameters	p	
	Documentation	Piece at HantoCoordinate to remove from the board
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

public reset () : void	
Documentation	Remove all pieces from the board
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public getPrintableBoard () : String	
Documentation	Return a string representing the current state of the board, empty string if the board is empty.

Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	string representing the board
Query	false

public getEmptyNeighborCoordinatesOf (c : HexCoordinate) : java.util.Collection		
Parameters	**c**	
	Documentation	Coordinate to find empty neighbors
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Get the empty neighbor coordinates of a specific coordinate	
Static	false	
Leaf	false	
Upper	1	
Template Type Bind Info	N/A	
Ordered	false	
Unique	true	
Return Type Documentation	Collection of neighbors, empty if none	
Query	false	

public canSlideTo (from : HexCoordinate, to : HexCoordinate) : boolean		
Parameters	**from**	
	Documentation	Source coordinate
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
	to	

	Documentation	Destination coordinate
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Determine if there is enough room for a piece to slide to its destination Currently only supports sliding for distances of one	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	true if piece can slide from from to to, false otherwise @throws HantoException if run with coordinates with distance > 1	
Query	false	

DeltaHantoGame

Attributes

private startingHand : java.util.Map			
Template Type Bind Info	N/A		
Initial Value	null		
Type	java.util.Map		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

private MAX_BUTTERFLIES : int	
Documentation	Counts of pieces in a player's hand
Initial Value	1

Type	int		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

private MAX_SPARROWS : int			
Initial Value	4		
Type	int		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

private MAX_CRABS : int			
Initial Value	4		
Type	int		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

Operations

public DeltaHantoGame ()	
Documentation	Create an instance of DeltaHanto
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

public setupGame () : void	
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Query	false

protected makeStartingHand () : java.util.Map	
Static	false
Leaf	false
Upper	1
Template Type Bind Info	N/A
Ordered	false
Unique	true
Query	false

DeltaHantoRuleset

Operations

protected verifyPieceCanMoveToDest (piece : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void		
Parameters	piece	
	Documentation	Piece being moved
	Multiplicity	Unspecified

	Type	HantoPieceType
	Direction	inout
	Java Detail	N/A
	from	
	Documentation	Source coordinate
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
	to	
	Documentation	Destination coordinate
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Documentation	Verify that a move that requires moving a piece is legal. This ensures that only butterflies and crabs can move one hex.	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	@throws HantoException if this condition has been violated	
Query	false	

protected verifyPlayerCanMovePieces (from : HexCoordinate, to : HexCoordinate) : void		
Parameters	**from**	
	Documentation	source coordinate
	Multiplicity	Unspecified
	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A

	to
	Multiplicity: Unspecified
	Type: HexCoordinate
	Direction: inout
	Java Detail: N/A
Documentation	Verify that the player is allowed to move pieces. In this case, they are allowed to do so if they have placed their butterfly.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if this condition has been violated
Query	false

protected playerHasResigned (type : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : boolean

Parameters	**type**
	Documentation: piece type for the move
	Multiplicity: Unspecified
	Type: HantoPieceType
	Direction: inout
	Java Detail: N/A
	from
	Documentation: source coordinate
	Multiplicity: Unspecified
	Type: HexCoordinate
	Direction: inout
	Java Detail: N/A
	to
	Documentation: destination coordinate
	Multiplicity: Unspecified

	Type	HexCoordinate
	Direction	inout
	Java Detail	N/A
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Return Type Documentation	true if the player has resigned, false otherwise	
Query	false	

protected otherPlayerWinsIfThisPlayerResigned () : MoveResult	
Documentation	Check whether or not this player has resigned and set the win condition appropriately.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	win for the opponent if the current player has resigned, OK otherwise
Query	false

public DeltaHantoRuleset (state : HantoGameState)		
Parameters	state	
	Documentation	the game's state object
	Multiplicity	Unspecified
	Type	HantoGameState
	Direction	inout
	Java Detail	N/A
Documentation	Create a ruleset for Delta Hanto	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	

Query	false

public doPreMoveChecks (piece : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void

Parameters

piece

Documentation	Piece to move
Multiplicity	Unspecified
Type	HantoPieceType
Direction	inout
Java Detail	N/A

from

Documentation	Source coordinate
Multiplicity	Unspecified
Type	HexCoordinate
Direction	inout
Java Detail	N/A

to

Documentation	Destination coordinate
Multiplicity	Unspecified
Type	HexCoordinate
Direction	inout
Java Detail	N/A

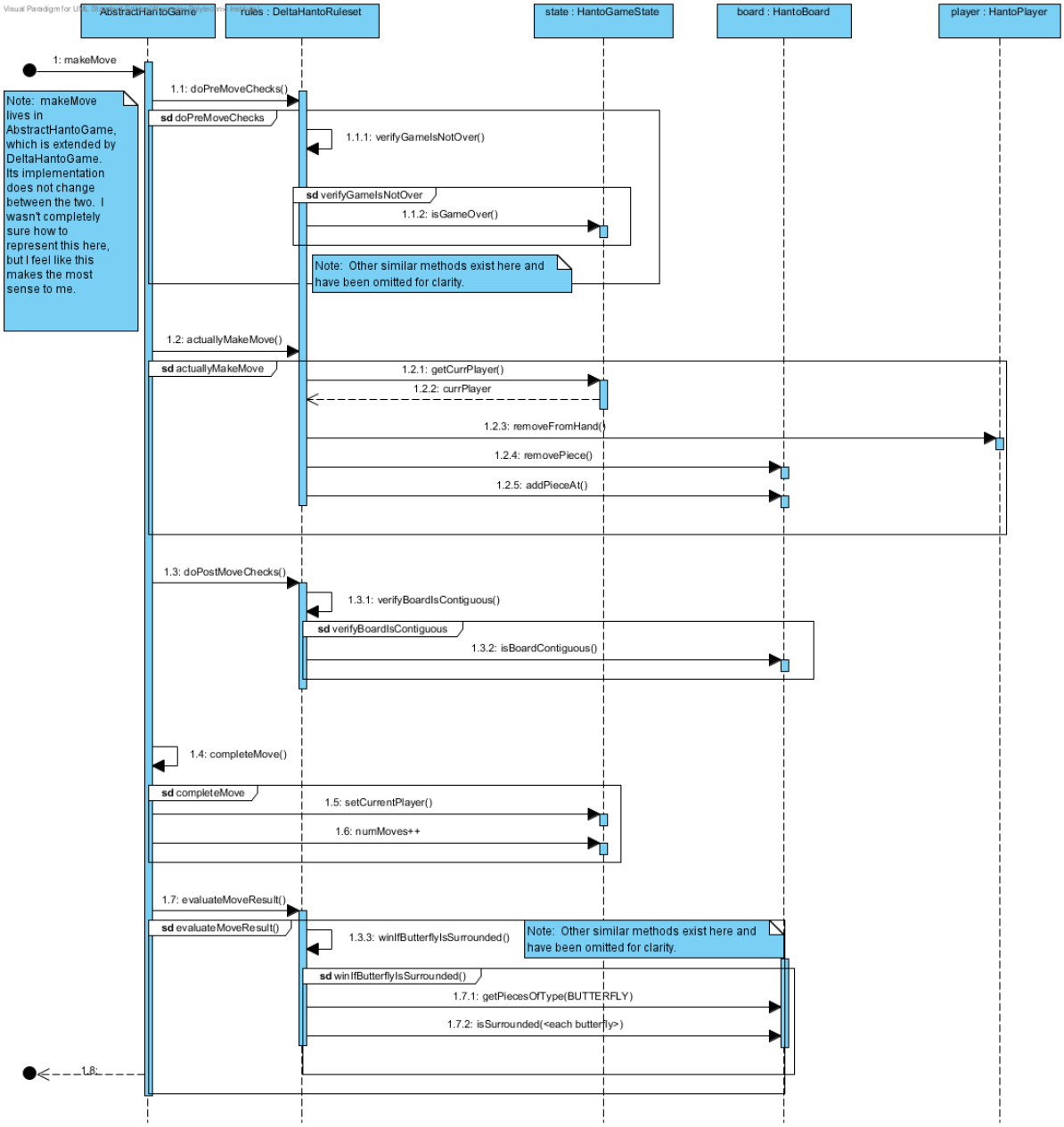
Documentation	Perform checks to be made before a move
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if any conditions have been violated
Query	false

public evaluateMoveResult () : MoveResult

Documentation	Determine result of a move based on specification;

	sets gameOver state if game has ended.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if board state is invalid
Query	false

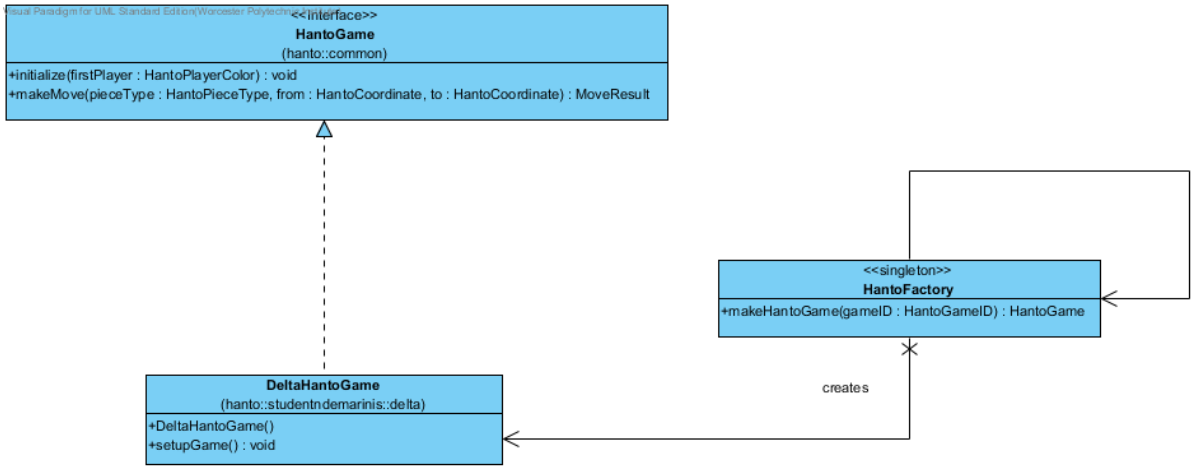
makeMove() Sequence Diagram



Summary

Documentation
Note: makeMove lives in AbstractHantoGame, which is extended by DeltaHantoGame. Its implementation does not change between the two. I wasn't completely sure how to represent this here, but I feel like this makes the most sense to me.
Note: Other similar methods exist here and have been omitted for clarity.
Note: Other similar methods exist here and have been omitted for clarity.

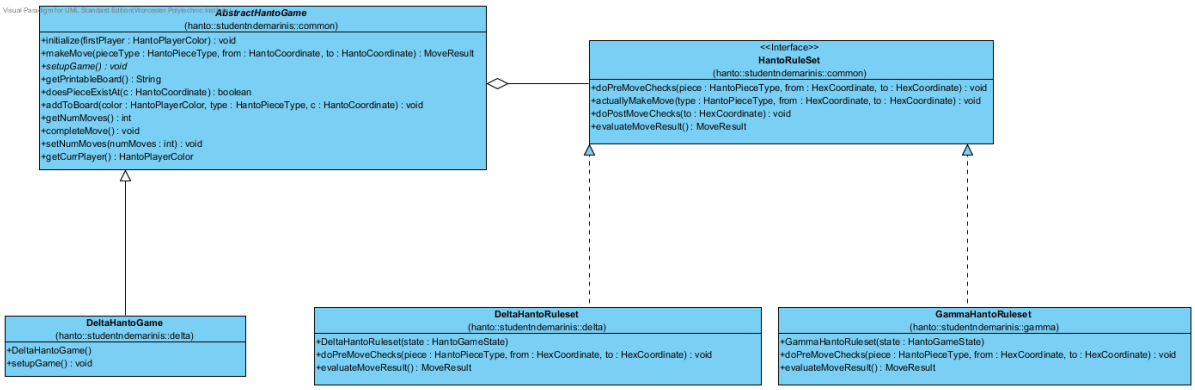
HantoFactory



Summary

Name	Documentation
HantoGame	The HantoGame interface is the primary interface between the student's code and any external (non-student written) code. Every version of Hanto will have a realization of the HantoGame interface. @author gpollice @version Jan 12, 2013
HantoFactory	Factory for creating Hanto Games. Currently supports Alpha, Gamma, and Delta Hanto. @author ndemarinis @version Feb 9 2013
DeltaHantoGame	This class is a concrete realization of the game for DeltaHanto. It provides the necessary initialization methods for DeltaHanto and relies on the ruleset and abstract classes to handle the rest of the implementation. @author ndemarinis @version Feb 9, 2013

Strategy Pattern: Hanto Rules



Summary

Name	Documentation
AbstractHantoGame	This abstract class encapsulates the basic functionality for a HantoGame, including initialization. Lots of public methods exist here to expose extra implementation for testing purposes. @author ndemarinis @version Jan 31, 2013
HantoRuleSet	This interface represents methods required for a HantoRuleset. These methods will be called by AbstractHantoGame to provide a common implementation for making and verifying moves. @author ndemarinis @version Jan 31, 2013
DeltaHantoRuleset	This class is a concrete realization of the ruleset for Delta Hanto. It provides all of the rule methods specific to DeltaHanto. @author ndemarinis @version Feb 9, 2013
GammaHantoRuleset	Abstraction for GammaHanto's move rules @author ndemarinis @version Jan 31, 2013
DeltaHantoGame	This class is a concrete realization of the game for DeltaHanto. It provides the necessary initialization methods for DeltaHanto and relies on the ruleset and abstract classes to handle the rest of the implementation. @author ndemarinis @version Feb 9, 2013

Details

GammaHantoRuleset

Attributes

private MAX_MOVES : int			
Documentation	Max number of moves before ending in a draw		
Initial Value	10 * 2		
Type	int		
Allow Empty Name	false		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Java Detail	N/A		
Aggregation	None		
Derived Union	false		
Read Only	false		
Leaf	false		

Operations

public GammaHantoRuleset (state : HantoGameState)		
Parameters	state	
	Documentation	The HantoGame we'll be checking
	Multiplicity	Unspecified
	Type	HantoGameState
	Direction	inout
	Java Detail	N/A
Documentation	Make a new set of GammaHanto's rules, given the game itself	
Static	false	
Leaf	false	
Upper	1	
Ordered	false	
Unique	true	
Query	false	

public doPreMoveChecks (piece : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void	
Parameters	**piece**
	Documentation: Piece to be placed at the given location Multiplicity: Unspecified Type: HantoPieceType Direction: inout Java Detail: N/A
	from
	Documentation: source coordinate of piece on the board, null if not on the board Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A
	to
	Documentation: destination coordinate for piece to move Multiplicity: Unspecified Type: HexCoordinate Direction: inout Java Detail: N/A
Documentation	Checks to be performed before a move is made
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if a rule has been violated, leaving the board in an illegal state
Query	false

public evaluateMoveResult () : MoveResult	
Documentation	Evaluate whether the game needs to end based on the board

	configuration. Intended to be called after each move to determine if a win has occurred.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if the board is in an illegal state
Query	false

protected verifyPieceCanMove (piece : HantoPieceType, from : HexCoordinate, to : HexCoordinate) : void

Parameters

piece

Documentation	Piece being moved
Multiplicity	Unspecified
Type	HantoPieceType
Direction	inout
Java Detail	N/A

from

Documentation	Source coordinate
Multiplicity	Unspecified
Type	HexCoordinate
Direction	inout
Java Detail	N/A

to

Documentation	Destination coordinate
Multiplicity	Unspecified
Type	HexCoordinate
Direction	inout
Java Detail	N/A

Documentation	Verify that a move that requires moving a piece is legal. This ensures that only butterflies can move one hex.
Static	false

Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	@throws HantoException if this condition has been violated
Query	false

protected endInDrawAfter10Moves () : MoveResult	
Documentation	Give result ending the game in a draw after 10 moves.
Static	false
Leaf	false
Upper	1
Ordered	false
Unique	true
Return Type Documentation	OK if game has been running for less than 10 moves, DRAW otherwise.
Query	false