Model Documentation

Model Detail

This document provides a complete overview of all element details. For simpler and more focused reports, simply copy this initial template and turn off the sections not required.

Class Model

Type: **Package**
Status: Proposed. Version . Phase 1.0.
Package: Model
Detail: *Created on 19/03/2012. Last modified on 19/03/2012*
GUID: {7A1EB8B8-A618-40d7-8615-FB193BE6095A}

Class Model - *(Class diagram)*
Created By: juliano *on* 19/11/2005
Last Modified: 25/06/2012
Version: 1.0. *Locked:* False
GUID: {7D75C802-4822-4a4f-8406-DF1DF44F5F8A}

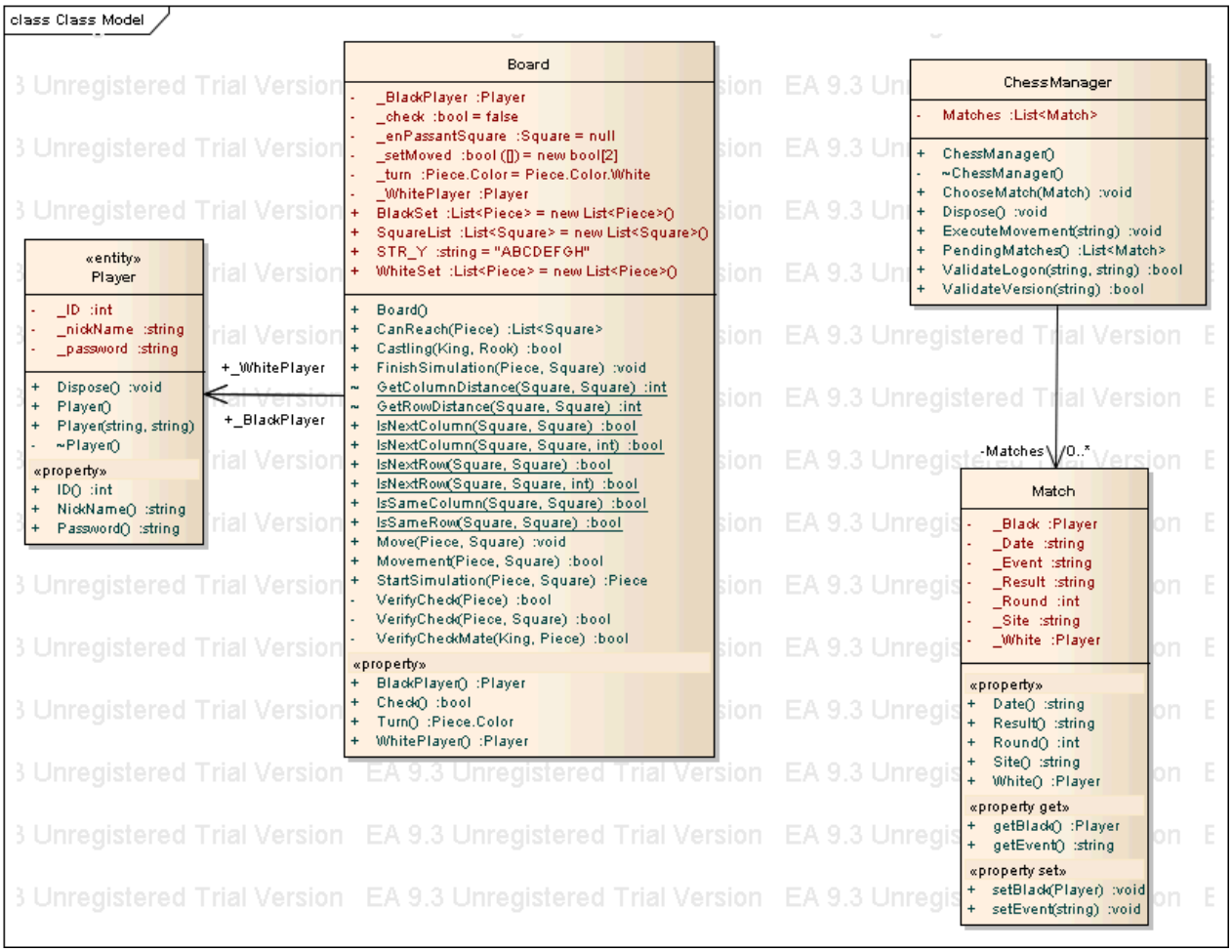

Figure: 1

Engine Class Model - *(Class diagram)*
Created By: juliano *on* 29/03/2012
Last Modified: 25/06/2012
Version: 1.0. *Locked:* False
GUID: {9CA92CA6-0428-4d02-A654-3855BDD4F75B}

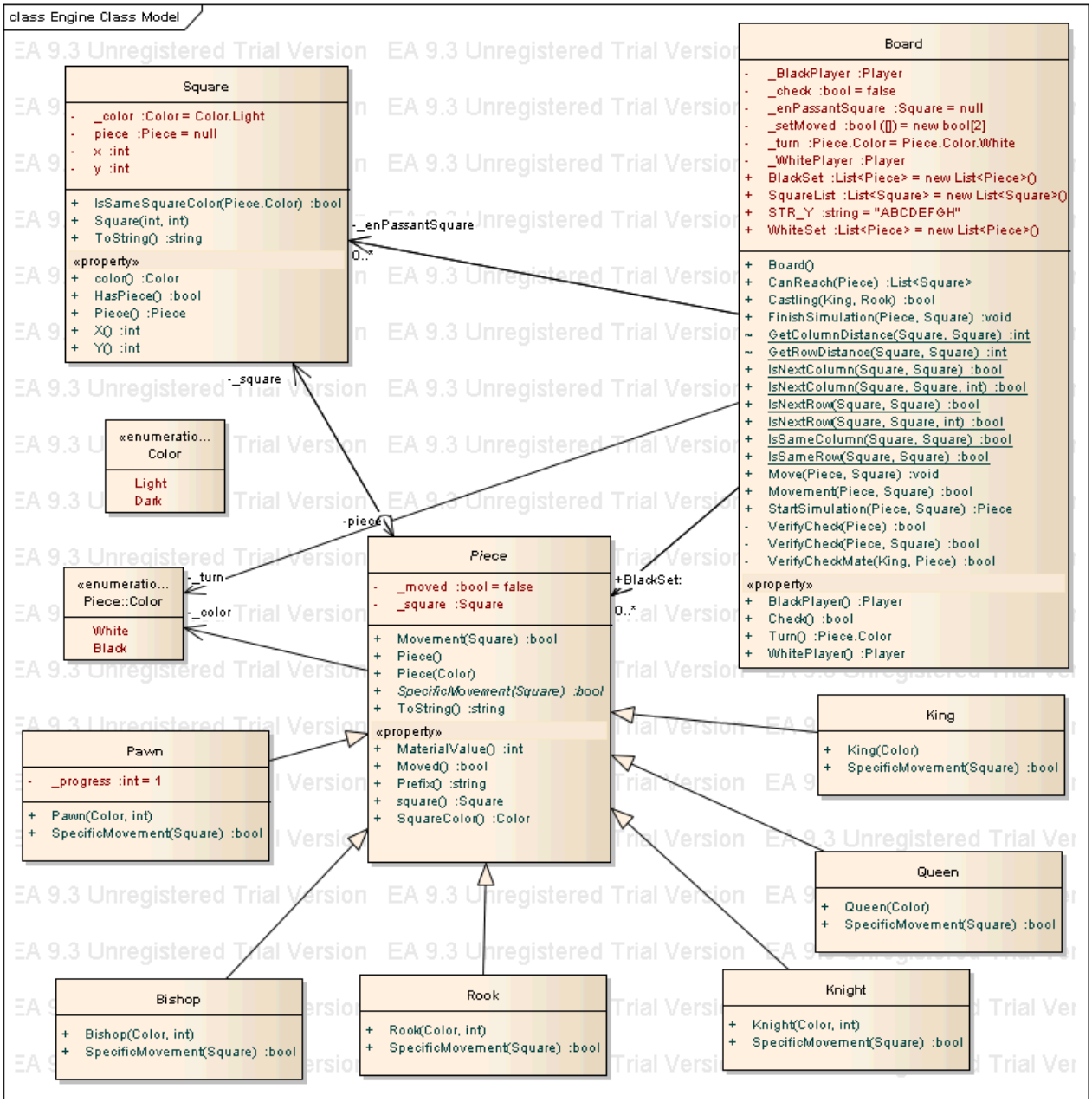

Figure: 2

Bishop

Type: **Class** **Piece**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {584A355B-40B0-4ee7-B84E-470733FF4EA2}

This is a kind of piece that can´t sit on a square of different color from its original one.

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public Bishop	Public Piece	

Operations

Method	Notes	Parameters
Bishop() Public		**Color** [in] color **int** [in] number
SpecificMovement() bool Public		**Square** [in] targetSquare

Board

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 22/06/2012.*
GUID: {67E5E92E-8CC9-410e-A098-602F1EA940EB}

This class is a composition of an eight-by-eight squares table. This class is responsible to create the instances of the squares and two sets of pieces. Besides, it is also responsible to place all those pieces created on its starting point position. When the game starts, it also manages the following items: The movement of the pieces, by calling the Movement method of the specific piece instance and then watching for interaction among the pieces.

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public Board	Private _enPassantSquare Square	
Association Source -> Destination	Public Board	Private _turn Color	

Connector	Source	Target	Notes
Association Source -> Destination	Public Board	Public BlackSet Piece	
Association Source -> Destination	Public Board	Public SquareList Square	
Association Source -> Destination	Public Board	Public WhiteSet Piece	
Association Source -> Destination	Public Board	Public _BlackPlayer Player	
Association Source -> Destination	Public Board	Public _WhitePlayer Player	
Sequence Movement(Piece, Square) Source -> Destination	Public Game Screen	Public Board	
Sequence SpecificMovement(Square) Source -> Destination	Public Board	Public Piece	
Sequence SpecificMovement(Square) Source -> Destination	Public Board	Public <anonymous>	
Sequence Move(Piece, Square) Source -> Destination	Public Game Screen	Public Board	
Sequence Move(Piece, Square) Source -> Destination	Public Game Screen	Public Board	

Attributes

Attribute	Notes	Constraints and tags
_BlackPlayer Player Private		*Default:*
_check bool Private		*Default:* false

Attribute	Notes	Constraints and tags
_enPassantSquare Square Private		*Default:* null
_setMoved bool Private Collection		*Default:* new bool[2]
_turn Piece.Color Private		*Default:* Piece.Color.White
_WhitePlayer Player Private		*Default:*
BlackSet List<Piece> Public		*Default:* new List<Piece>()
SquareList List<Square> Public	Initializes an empty list of squares	*Default:* new List<Square>()
STR_Y string Public		*Default:* "ABCDEFGH" [const = true]

Attribute	Notes	Constraints and tags
WhiteSet List<Piece> Public	Initilizes two sets of pieces	*Default:* new List<Piece>()

Operations

Method	Notes	Parameters
BlackPlayer() Player Public		
Board() Public	Default constructor that creates an eight-by-eight square table and places two sets of pieces.	
CanReach() List<Square> Public	Method that returns a list of squares that can be reach by a specific piece. @returns	**Piece** [in] piece The piece in focus
Castling() bool Public	Performs the castling movement, if possible @returns True if the castling was performed	**King** [in] king **Rook** [in] rook
Check() bool Public		
FinishSimulation() void Public		**Piece** [in] piece **Square** [in] square
Static **GetColumnDistance()** int Internal		**Square** [in] square1 **Square** [in] square2
Static **GetRowDistance()** int Internal		**Square** [in] square1 **Square** [in] square2
Static **IsNextColumn()** bool Public		**Square** [in] square1 **Square** [in] square2

Method	Notes	Parameters
Static **IsNextColumn()** bool Public		**Square** [in] square1 **Square** [in] square2 **int** [in] distance
Static **IsNextRow()** bool Public		**Square** [in] square1 **Square** [in] square2
Static **IsNextRow()** bool Public		**Square** [in] square1 **Square** [in] square2 **int** [in] distance
Static **IsSameColumn()** bool Public		**Square** [in] square1 **Square** [in] square2
Static **IsSameRow()** bool Public		**Square** [in] square1 **Square** [in] square2
Move() void Public	Performs the movement of a piece to the square.	**Piece** [in] piece **Square** [in] square
Movement() bool Public	Validates a movement of a piece to the square.	**Piece** [in] piece The instance of the piece being moved **Square** [in] square The instance of the destination square
StartSimulation() Piece Public	Method responsible for making an actual move for the specified piece, keeping a bookmark for the former position. Every call for this method must be followed by a corresponding call for FinishSimulation method. Besides, it must be guaranteed by a transaction context, assuring that the movement can be rolled back, in order to avoid damage to the game integrity. It is recommended that all treatment between the calls should be inside an exception context. @returns	**Piece** [in] piece **Square** [in] square
Turn() Piece.Color		

Method	Notes	Parameters
Public		
VerifyCheck() bool Private	This method can be called after the commit of the movement @returns	**Piece** [in] piece The piece just moved
VerifyCheck() bool Private	 @returns	**Piece** [in] piece The piece just moved or the piece candidate for the movement **Square** [in] square The square can be informed in case being not the real position of the piece
VerifyCheckMate() bool Private	This method can be called after any method of check verification has returned true @returns	**King** [in] king **Piece** [in] piece
WhitePlayer() Player Public		

ChessManager

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 20/06/2012. Last modified on 25/06/2012.*
GUID: {B2D3AC22-481A-487d-BF17-1F37C0C524BD}

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public ChessManager	Private Matches Match	

Attributes

Attribute	Notes	Constraints and tags
Matches List<Match> Private		*Default:*

Attribute	Notes	Constraints and tags

Operations

Method	Notes	Parameters
ChessManager() Public		
~ChessManager() Private		
ChooseMatch() void Public	@param ="Match"	**Match** [in] match
Dispose() void Public		
ExecuteMovement() void Public		**string** [in] movement
PendingMatches() List<Match> Public		
ValidateLogon() bool Public		**string** [in] user **string** [in] password
ValidateVersion() bool Public		**string** [in] version

Color

Type: **Enumeration**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 29/03/2012.*
GUID: {1A49E7D7-C52A-4a0f-BA47-22DD694F6926}

Custom Properties

- isActive = False

Attributes

Attribute	Notes	Constraints and tags
Light Public		*Default:*

Attribute	Notes	Constraints and tags
Dark Public		*Default:*

King

Type: **Class** **Piece**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {ED22266E-FBDB-4e35-874B-2E4B7A916EA8}

This is the most important piece in the game. This kind of piece participates in a special movement called castling

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public King	Public Piece	

Operations

Method	Notes	Parameters
King() Public		**Color** [in] color
SpecificMovement() bool Public		**Square** [in] targetSquare

Knight

Type: **Class** **Piece**
Status: Proposed. Version 1.0. Phase 1.0.

Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {156A6488-07B0-4c10-A767-E4EFF12B0298}

This is the only piece in the game that is able to jump over the others

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public Knight	Public Piece	

Operations

Method	Notes	Parameters
Knight() Public		**Color** [in] color **int** [in] number
SpecificMovement() bool Public		**Square** [in] targetSquare

Match

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 22/06/2012. Last modified on 26/06/2012.*
GUID: {5CBBD7D6-E403-4714-ADFD-2A06B11D239C}

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public ChessManager	Private Matches Match	

Attributes

Attribute	Notes	Constraints and tags
_Black Player Private		*Default:*
_Date string Private		*Default:*
_Event string Private		*Default:*
_Result string Private		*Default:*
_Round int Private		*Default:*
_Site string Private		*Default:*
_White Player Private		*Default:*

Attribute	Notes	Constraints and tags

Operations

Method	Notes	Parameters
Date() string Public		
getBlack() Player Public		
getEvent() string Public		
Result() string Public		
Round() int Public		
setBlack() void Public		**Player** [in] newVal
setEvent() void Public		**string** [in] newVal
Site() string Public		
White() Player Public		

Pawn

Type: **Class** **Piece**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {7D7D50F6-B018-4aea-B522-FD2BFF27C093}

This is the most ordinary piece in the game.

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public Pawn	Public Piece	

Attributes

Attribute	Notes	Constraints and tags
_progress int Private		*Default:* 1

Operations

Method	Notes	Parameters
Pawn() Public		**Color** [in] color **int** [in] number
SpecificMovement() bool Public		**Square** [in] targetSquare

Piece

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {1C3FAD3F-B2A6-48bd-94B4-2A32F5C2F16B}

This is an abstract class that represents a generic piece. It can be inherited to implement specific behaviors of that kind of piece. It cannot be created. It refers to a square where it sits. When it does not refers to a square, that means it´s out of the board.

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public Piece	Private _color Color	
Generalization Source -> Destination	Public Rook	Public Piece	
Generalization	Public	Public	

Connector	Source	Target	Notes
Source -> Destination	Knight	Piece	
Generalization Source -> Destination	Public Queen	Public Piece	
Generalization Source -> Destination	Public King	Public Piece	
Generalization Source -> Destination	Public Pawn	Public Piece	
Generalization Source -> Destination	Public Bishop	Public Piece	
Association Source -> Destination	Public Board	Public BlackSet Piece	
Association Source -> Destination	Public Board	Public WhiteSet Piece	
Association Source -> Destination	Public Square	Private piece Piece	
Association Source -> Destination	Public Piece	Private _square Square	
Sequence SpecificMovement(Square) Source -> Destination	Public Board	Public Piece	

Attributes

Attribute	Notes	Constraints and tags
_moved bool Private	private Color _color;	*Default:* false
_square Square Private		*Default:*

Operations

Method	Notes	Parameters
MaterialValue() int Public		
Moved() bool Public		
Movement() bool Public	Movement constraints are checked out in the pieces, since there are specific things to consider, for example, how many squares this piece is supposed to move. Every class related to a piece of the chess must implement this method and should call this one. @returns	**Square** [in] targetSquare
Piece() Public	int _value;	
Piece() Public	Constructor able to receive the color	**Color** [in] color
Prefix() string Public		
SpecificMovement() bool Public		**Square** [in] targetSquare
square() Square Public		
SquareColor() Color Public		
ToString() string Public		

Color

Type: **Enumeration**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {5D2562AE-8D21-42fe-8F92-75AD7006E29B}

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public Piece	Private _color Color	
Association Source -> Destination	Public Board	Private _turn Color	

Connector	Source	Target	Notes

Attributes

Attribute	Notes	Constraints and tags
White Public		*Default:*
Black Public		*Default:*

Player

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 22/06/2012.*
GUID: {83C68FA1-C2C9-4a30-85BD-7748A3F14FD2}

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public Board	Public _BlackPlayer Player	
Association Source -> Destination	Public Board	Public _WhitePlayer Player	

Attributes

Attribute	Notes	Constraints and tags

Attribute	Notes	Constraints and tags
_ID int Private		*Default:*
_nickName string Private		*Default:*
_password string Private		*Default:*

Operations

Method	Notes	Parameters
Dispose() void Public		
ID() int Public		
NickName() string Public		
Password() string Public		
Player() Public		
Player() Public		**string** [in] nickName **string** [in] password
~Player() Private		

Queen

Type: **Class** **Piece**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*

GUID: {908AABEB-2A12-4260-A59B-14EA09145420}

This is the second most important piece in the game.

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public Queen	Public Piece	

Operations

Method	Notes	Parameters
Queen() Public		**Color** [in] color
SpecificMovement() bool Public		**Square** [in] targetSquare

Rook

Type: **Class** **Piece**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 25/06/2012.*
GUID: {916B3729-FD01-4f55-B797-A94C66377CC0}

This is the third most important piece in the game. This kind of piece participates in a special movement called castling

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public Rook	Public Piece	

Operations

Method	Notes	Parameters
Rook() Public		**Color** [in] color **int** [in] number
SpecificMovement() bool Public		**Square** [in] targetSquare

Square

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {C95C5999-BDD9-40c7-8D65-E26A5A1F1126}

This is a class that compose a board. Each one has its fixed, immutable position, and refers or not to a single volatile piece in the game. When it does not refers to a piece, it means it´s empty.

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public Square	Private _color Color	
Association Source -> Destination	Public Board	Private _enPassantSquare Square	
Association Source -> Destination	Public Board	Public SquareList Square	
Association Source -> Destination	Public Square	Private piece Piece	
Association Source -> Destination	Public Piece	Private _square Square	

Attributes

Attribute	Notes	Constraints and tags
_color Color Private	This is not a relevant information, it can only be useful for the Bishop	*Default:* Color.Light

Attribute	Notes	Constraints and tags
piece Piece Private		*Default:* null
x int Private	represents the horizontal position of the square in the board	*Default:*
y int Private	represents the vertical position of the square in the board	*Default:*

Operations

Method	Notes	Parameters
color() Color Public	This is not a relevant information, it can only be useful for the Bishop	
HasPiece() bool Public		
IsSameSquareColor() bool Public		**Piece.Color** [in] squareColor
Piece() Piece Public	A piece placed in this instance of Square	
Square() Public		**int** [in] x **int** [in] y
ToString() string Public	Overrides the method in order to return a chess notation known internationally for the square @returns	
X() int Public	Represents a reference to the alphabetic sequence in the chess board	

Method	Notes	Parameters
Y() int Public	Represents a reference to the numeric sequence in the chess board	

Color

Type: **Enumeration**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Class Model *Keywords:*
Detail: *Created on 29/03/2012. Last modified on 24/05/2012.*
GUID: {7C39A9D9-93AC-48e5-9128-A5C44CB3DC88}

Custom Properties

- isActive = False

Connections

Connector	Source	Target	Notes
Association Source -> Destination	Public Square	Private _color Color	

Attributes

Attribute	Notes	Constraints and tags
Light Public		*Default:*
Dark Public		*Default:*