**Model**

**com.ipl.training.induction.draughts.model**

**Package Specification**

**Project Reference 999/170**

**Document Reference draughts\_model\_ps**

**13 June 2013**

**Issue 4**

**Copyright Notice**

This document has been prepared by IPL Information Processing Limited (“IPL”) and, subject to any existing rights of other parties, IPL is the owner of the copyright of this document. No part of this document may be copied, reproduced, stored in a retrieval system, disclosed to a third party or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of IPL.

**© IPL Information Processing Limited 2013**

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Issue | Author | Comments |
| 10/06/2010 | 1 | Brian Bassil | Issued |
| 28/09/2010 | 2 | C Harrison | Corrections after first run of the course |
| 22/11/2011 | 3 | J Poole, C Harrison | Updated to reflect new Java standard, including diagrams |
| 13/06/2013 | 4 | C Harrison | Added Direction to the Structure section |

Contents

[1 Package Identity 1](#_Toc358901641)

[2 Description 1](#_Toc358901642)

[3 Interface 1](#_Toc358901643)

[4 Structure 2](#_Toc358901644)

[5 Element Descriptions 3](#_Toc358901645)

[5.1 BoardLayout 3](#_Toc358901646)

[5.1.1 Public Methods 3](#_Toc358901647)

[5.1.2 Package Access Methods 4](#_Toc358901648)

[5.1.3 Protected Methods 9](#_Toc358901649)

[5.2 DraughtsModel 9](#_Toc358901650)

[5.2.1 Public Methods 9](#_Toc358901651)

[5.2.2 Package Access Methods 14](#_Toc358901652)

[5.2.3 Protected Methods 15](#_Toc358901653)

[5.2.4 Private Methods 15](#_Toc358901654)

[5.3 AbstractComputerPlayer 15](#_Toc358901655)

[5.3.1 Public Methods 16](#_Toc358901656)

[5.3.2 Package Access Methods 17](#_Toc358901657)

[5.3.3 Protected Methods 18](#_Toc358901658)

[5.3.4 Public Fields 20](#_Toc358901659)

[5.3.5 Package Access Fields 20](#_Toc358901660)

[5.3.6 Protected Fields 20](#_Toc358901661)

[5.4 FirstMovePlayer 20](#_Toc358901662)

[5.4.1 Public Methods 20](#_Toc358901663)

[5.4.2 Package Access Methods 20](#_Toc358901664)

[5.4.3 Package Access Methods 21](#_Toc358901665)

[5.4.4 Protected Methods 21](#_Toc358901666)

[5.5 MoveData 21](#_Toc358901667)

[5.5.1 Public Methods 21](#_Toc358901668)

[5.5.2 Package Access Methods 21](#_Toc358901669)

[5.5.3 Protected Methods 24](#_Toc358901670)

[5.6 NextSquare 24](#_Toc358901671)

[5.6.1 Public Methods 24](#_Toc358901672)

[5.6.2 Package Access Methods 25](#_Toc358901673)

[5.7 Enumeration NextSquare.Direction 26](#_Toc358901674)

[5.7.1 Public Methods 26](#_Toc358901675)

[5.7.2 Package Access Methods 26](#_Toc358901676)

[5.7.3 Public Fields 26](#_Toc358901677)

[5.7.4 Package Access Fields 26](#_Toc358901678)

[5.8 TurnData 26](#_Toc358901679)

[5.8.1 Public Methods 26](#_Toc358901680)

[5.8.2 Package Access Methods 27](#_Toc358901681)

[5.8.3 Public Fields 30](#_Toc358901682)

[5.8.4 Package Access Fields 30](#_Toc358901683)

[6 Resource Requirements 30](#_Toc358901684)

[7 Test Plan 30](#_Toc358901685)

[8 Scenarios 31](#_Toc358901686)

[8.1 New Game 31](#_Toc358901687)

[8.2 Set Click 32](#_Toc358901688)

[9 Supporting Documentation 33](#_Toc358901689)

# Package Identity

Package Name - Model  
Package Identity - com.ipl.training.induction.draughts.model

# Description

The model package contains an implementation of IDraughtsModel and contains all of the game logic. The model responds to messages from the controller and produces events.

The model package also contains the abstract base class AbstractComputerPlayer which is used to define a computer player.

# Interface

A new instance of IDraughtsModel can be accessed via the public static method DraughtsModel.createModel(). All other public methods are defined in IDraughtsModel.



# Structure



Note that the **NextSquare** class has a nested static enumeration:

**static** **enum** Direction {

*NW*,

*NE*,

*SW*,

*SE*

}

# Element Descriptions

## BoardLayout

BoardLayout represents the position of pieces on the board. All possible moves are determined by the constructor and the BoardLayout is immutable. New layouts can either be constructed from a FEN tag, or an existing BoardLayout and two int values that represent a move.

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Public |
| Modifiers | Final |
| Extends | - |
| Implements | Iterable<SquareData> |

### Public Methods

#### equals

An overridden version of equals() may be helpful to implementers of AbstractComputerPlayer. If equals is overridden hashCode() must also be overridden with a consistent implementation.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | Object.equals() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| obj | Object | The reference object with which to compare |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | True if obj is equal to this BoardLayout, otherwise false. |

#### hashCode

An overridden version of hashCode() may be helpful to implementers of AbstractComputerPlayer. If equals is overridden equals() must also be overridden with a consistent implementation

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | Object.hashCode() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| int | A hash code value for this object. |

#### toString

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | Object.toString() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| String | A FEN tag representing the BoardLayout |

### Package Access Methods

#### BoardLayout(String)

Constructs a new BoardLayout based on a FEN tag.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| fen | String | A String containing a FEN tag that describes the initial board layout. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### BoardLayout(BoardLayout, int, int)

Constructs a new BoardLayout based on an existing BoardLayout and a move.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| board | BoardLayout | The initial board layout. |
| start | int | The starting point of the move. |
| end | Int | The ending point of the move. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### canSelect

Returns a boolean indicating if the current player can select squareID.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | Integer | The square ID to query |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | True if the current player can select squareID, otherwise false |

#### diff

Compares two layouts and returns a Set containing the differing squares.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Static |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| oldLayout | BoardLayout | The intial layout |
| newLayout | BoardLayout | The new layout |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| Set<SquareData> | A set of SquareData that describes the changes needed for oldLayout to match newLayout. This will be an empty set if oldLayout matches newLayout. |

#### getCaptured

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| SquareData | A SquareData describing the piece that was captured creating this layout, or null |

#### getCurrentPlayer

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| PlayerColor | The current player |

#### getMoves

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| player | PlayerColor | The player to get moves for |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| SortedMap<Integer, MoveData> | A map of SquareID to possible moves. |

#### getRank

Given a square ID and a Player getRank returns the rank. For a black player rank 1 contains squares 1-4 and rank 8 contains squares 29-32. For a white player rank 1 contains squares 29-32 and rank 8 contains squares 1-4. This is done via a lookup table for speed.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | static |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | int | A square ID |
| player | PlayerColor | The player to get the rank of squareID for. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| int | A number between 1 and 8 inclusive. |

#### getSquare

Gets a square from squares based on a square ID. (That is, this method converts from 1-based ‘Square ID’ values used elsewhere to 0-based array indexes used internally in BoardLayout).

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | int | The square ID to query |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| Square | The square at squareID |

#### isEdge

Method to determine if a specific square is an edge square. This is done via a lookup so as to be fast.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | static |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | int | The square ID to query |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | True if squareID is an edge square, otherwise false. |

### Protected Methods

None

## DraughtsModel

DraughtsModel is a concrete implementation of IDraughtsModel. It will fire events when the board layout or current player changes. DraughtsModel handles calculation of legal moves and clicks to selection and/or moves.

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Public |
| Modifiers | Final |
| Extends | - |
| Implements | IDraughtsModel |

### Public Methods

#### createModel

Creates a new instance of DraughtsModel via a call to the private constructor but only returns the interface. This hides the implementation of details from the caller.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Static |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| IDraughtsModel | A newly constructed instance of DraughtsModel. |

#### addPropertyChangeListener

Registers a property change listener with the DraughtsModel. All registered listeners will be notified about property change events.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | IDraughtsModel.addPropertyChangeListener() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| listener | PropertyChangeListener | The property change listener to register |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### errorMsg

Used to inform the model that an error has occurred. This causes property change events to be fired, one to display an error message and another to trigger a new game.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | IDraughtsModel.errorMsg() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| errorMsg | String | The error message to display |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### export

Writes a PDN file containing the current board layout and the game history using the writer specified by writer.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | IDraughtsModel.export() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| writer | java.io.Writer | The writer to export the PDN to. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### getCurrentPlayer

Returns either IDraughtsModel.PlayerColor.Black or IDraughtsModel.PlayerColor.White indicating which player is currently taking a move.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | IDraughtsModel.getCurrentPlayer() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| PlayerColor | The value returned indicates whose turn it is. |

#### newGame

Sets up the DraughtsModel according to the supplied GameData object. If the GameData object specifies a PDN file then this is used to layout the board, otherwise the default layout is used.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | IDraughtsModel.newGame() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| newGameData | GameData | Initial data to use when setting up the game |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### setClick

This method is used to indicate to the model that a square has been ‘clicked’ by the user. Behaviour is described via a state diagram below.



If the call to setClick has no effect a IDraughtsController.STATUS event is fired indicating why this is the case.

If propagate is true then a IDraughtsController.SEND\_CLICK event is fired to indicate to all views that a click has occurred.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | synchronized |
| Overrides | IDraughtsModel.setClick() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | Integer | The square that has been clicked |
| propagate | boolean | Whether or not a SEND\_CLICK event should be fired. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | Returns true if a valid click is made. |

#### undo

Instructs the model to undo the last human move.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | IDraughtsModel.undo() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

### Package Access Methods

#### getCurrentLayout

Returns a BoardLayout representing the board state.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| BoardLayout | The current board layout |

### Protected Methods

None

### Private Methods

#### DraughtsModel

The constructor is private to prevent access outside of this class. The createModel() method must be used to construct a new instance.

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

## AbstractComputerPlayer

This is an abstract base class handling the processing common to all computer players.

The functionality includes:

* Waiting until it is the computer player’s turn.
* Timing out after 10 seconds.

Subclasses of AbstractComputerPlayer must include a public constructor which takes a PlayerColor and a DraughtsModel. This is so they can be loaded dynamically be DraughtsModel.

|  |  |
| --- | --- |
| Attribute | Value |
| **Visibility** | **Package** |
| **Modifiers** | **Abstract** |
| **Extends** | **Thread** |
| **Implements** | **-** |

The following diagram shows the states that AbstractComputerPlayer can be in.



### Public Methods

#### run

This method handles waiting until it is the computer player’s turn and then calls the abstract method takeMove() to take the move.

This method also handles the timeout – forcing a legal move to be taken after 10 seconds.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | Thread.run() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

### Package Access Methods

#### AbstractComputerPlayer

The only constructor, stores the DraughtsModel and a Player which is the player this AbstractComputerPlayer represents.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | DraughtsModel | The DraughtsModel that represents the game. |
| color | PlayerColor | The player that this AbstractComputerPlayer represents |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### determineMoves

An abstract method that must be overridden by sub-classes to determine which moves to take. Moves are stored by calling storeMove().

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Abstract |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### getPlayerColor

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| PlayerColor | The Player that this AbstractComputerPlayer represents |

### Protected Methods

#### getTimeRemaining

A method to query the time left before timeout. This can be used by subclasses to determine if they have sufficient time to do more processing.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| long | The time left for this move, in milliseconds. |

#### stopPlaying

Used to stop the looping in the run() method.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### storeClick

Should be called from determineMoves() to store the square that should be clicked to perform this move.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Final |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | Integer | The square that should be clicked on as part of this move. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

### Public Fields

None

### Package Access Fields

None

### Protected Fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Attributes | Value | Description |
| playerColor | IDraughtsModel.PlayerColor | Final | - | The Player that is represented by this instance of AbstractComputerPlayer. |

## FirstMovePlayer

This is the simplest possible implementation of AbstractComputerPlayer, the determineMoves method simply returns. This causes AbstractComputerPlayer to take the first available move.

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Package |
| Modifiers | - |
| Extends | AbstractComputerPlayer |
| Implements | - |

### Public Methods

None

### Package Access Methods

#### FirstMovePlayer

The only constructor, just calls the super constructor

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | AbstractComputerPlayer |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | DraughtsModel | The DraughtsModel that represents the game. |
| color | PlayerColor | The player that this AbstractComputerPlayer represents |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

### Package Access Methods

#### determineMoves

Just returns.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | Abstract |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

### Protected Methods

None

## MoveData

MoveData stores information about all of the squares that a piece could move to.

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Package |
| Modifiers | - |
| Extends | - |
| Implements | - |

### Public Methods

None

### Package Access Methods

#### MoveData

Constructs a new MoveData object.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ordinaryMoves | SortedSet<Integer> | The squares that the piece could make a non-capturing move to |
| capturingMoves | SortedSet<Integer> | The squares that the piece could make a capturing move to |
| capturedPieces | SortedMap<Integer,Integer> | Maps from the square moved into to the square containing a captured piece. |

#### canMakeCapturingMove

Returns a boolean indicating if a capturing move is possible.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | True if a capturing move can be made, otherwise false. |

#### canMove

Determines if the piece that this MoveData represents can make a move.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | Returns true if the piece can move, otherwise false. |

#### getCapturedPiece

Return the square ID of the piece that will be captured by a move to squareID. Return 0 if no piece will be captured.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | Integer | The square ID of a possible move. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| Integer | The square ID of the piece that will be captured by a move to squareID. Returns 0 if no piece will be captured. |

#### getMoves

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| SortedSet<Integer> | The set of all legal moves. If there are no legal moves are possible an empty set will be returned. |

#### setInvalid

This method marks this MoveData as invalid. Once setInvalid() has been called canMove() will always return false and getMoves() will return an empty set.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

### Protected Methods

None

## NextSquare

This class is used to describe all of the transitions that are possible from one square. Objects of this type are immutable.

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Package |
| Modifiers | - |
| Extends | - |
| Implements | - |

### Public Methods

None

### Package Access Methods

#### NextSquare

Creates a new NextSquare object containing the IDs of adjacent squares. A value of 0 for a direction indicates that there is no square there.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| nw | Integer | The square ID to the north west – use 0 to indicate no square. |
| ne | Integer | The square ID to the north east – use 0 to indicate no square. |
| sw | Integer | The square ID to the south west – use 0 to indicate no square. |
| se | Integer | The square ID to the south east – use 0 to indicate no square. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### get

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| direction | NextSquare.Direction | The direction to move |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| Integer | The square ID of the next square in when moving in direction. A return of 0 indicates there is no square in that direction. |

## Enumeration NextSquare.Direction

Direction is a package access enumeration type nested within NextSquare. It is used to describe the possible move directions.

### Public Methods

None

### Package Access Methods

None

### Public Fields

None

### Package Access Fields

|  |  |
| --- | --- |
| Name | Description |
| NW | Indicates a move to the north west |
| NE | Indicates a move to the north east |
| SW | Indicates a move to the south west |
| SE | Indicates a move to the south east |

## TurnData

The TurnData class stores information about the moves that occurred within a single turn. The toString method returns a String in a suitable form for use in a PDN file.

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Package |
| Modifiers | Final |
| Extends | - |
| Implements | - |

### Public Methods

#### toString

Returns a String representing the squares that were moved through in this turn. If the move was a capturing move then the numbers are separated by ‘x‘, otherwise they are separated by ‘-‘.

For example:

* “11-15”
* “11x18x25”.

The format matches that used for game history in a PDN file – see the Systems Requirements Specification for full details.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | Object.toString() |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| String | A String describing this TurnData. |

### Package Access Methods

#### TurnData

The sole constructor takes an int representing the starting point for a move and the BoardLayout at the start of the turn.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| squareID | int | The starting point for a move |
| layout | BoardLayout | The BoardLayout at the start of this turn. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### addCapturingMove

Adds a capturing move to this TurnData.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| destination | int | The destination of the move |
| capturedSquare | SquareData | Information about the piece that was captured |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### addNormalMove

Adds a normal move to this TurnData.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| destination | int | The destination of the move |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| - | - |

#### getBoardLayout

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| BoardLayout | The BoardLayout at the start of this turn. |

#### getCaptured

Returns a list of SquareData representing the pieces captured during this turn. The list should be a copy of the one stored internally to prevent modification.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| List<SquareData> | A list of SquareData objects representing the captured pieces. |

#### getCurrentPosition

Where the piece is currently, if no moves have been added it will return the value used in the constructor.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| int | The current position. |

#### getStartingPosition

The starting position for this turn – the value used in the constructor.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | - |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| - | - | - |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| int | The starting position for this turn. |

### Public Fields

None

### Package Access Fields

None

# Resource Requirements

None

# Test Plan

There will be no formal package level testing. Each class will be unit tested and then the entire system will undergo system level testing.

# Scenarios

## New Game



## Set Click



# Supporting Documentation

None.

**- End of Document -**