**DraughtsController**

**com.ipl.training.induction.draughts.controller**

**Class Specification**

**Project Reference 999/170**

**Document Reference DraughtsController\_cs**

**25 July 2013**

**Issue 6**

**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 | B. Bassil | Issued |
| 21/09/2010 | 2 | C Harrison | Corrections after first run of the course |
| 21/10/10 | 3 | C Harrison | More corrections after second run of the course |
| 24/11/2011 | 4 | Phil Lewis, C Harrison | Updated to reflect new Java standard, including diagrams. |
| 13/06/2013 | 5 | C Harrison | Updated processing logic to explain what’s going on. |
| 25/07/2013 | 6 | C Harrison | Corrected the test plan |

Contents

[1 Class Identity 1](#_Toc358906142)

[2 Description 1](#_Toc358906143)

[3 Interface 1](#_Toc358906144)

[4 Structure 2](#_Toc358906145)

[5 Element Descriptions and Interfaces 3](#_Toc358906146)

[5.1 Public Methods 3](#_Toc358906147)

[5.1.1 getInstance 3](#_Toc358906148)

[5.1.2 addView 3](#_Toc358906149)

[5.1.3 setModel 4](#_Toc358906150)

[5.1.4 squareClicked 4](#_Toc358906151)

[5.1.5 propertyChange 5](#_Toc358906152)

[5.1.6 newGame 6](#_Toc358906153)

[5.1.7 resetBoard 6](#_Toc358906154)

[5.1.8 connectionFailed 7](#_Toc358906155)

[5.1.9 error 7](#_Toc358906156)

[5.1.10 export 8](#_Toc358906157)

[5.1.11 undo 8](#_Toc358906158)

[5.2 Package Access Methods 9](#_Toc358906159)

[5.3 Private Methods 9](#_Toc358906160)

[5.3.1 DraughtsController 9](#_Toc358906161)

[5.4 Public Fields 9](#_Toc358906162)

[5.5 Package Access Fields 9](#_Toc358906163)

[5.6 Private Fields 9](#_Toc358906164)

[6 Resource Requirements 10](#_Toc358906165)

[7 Test Plan 10](#_Toc358906166)

[7.1 Test Error message 10](#_Toc358906167)

[7.2 Connection Failed 10](#_Toc358906168)

[7.3 Export file 10](#_Toc358906169)

[7.4 Add View 10](#_Toc358906170)

[7.5 Property Change with 2 views 10](#_Toc358906171)

[7.6 Start a new game 10](#_Toc358906172)

[7.7 Local human clicks when it is their turn 10](#_Toc358906173)

[7.8 Local human clicks when not their turn 10](#_Toc358906174)

[7.9 Local human clicks when the human player is remote 11](#_Toc358906175)

[7.10 Click undo when there are no remote players 11](#_Toc358906176)

[7.11 Click undo when there are remote players 11](#_Toc358906177)

[8 Scenarios 11](#_Toc358906178)

[9 Supporting Documentation 11](#_Toc358906179)

# Class Identity

Class Name - DraughtsController  
Class Identity - com.ipl.training.induction.draughts.controller

# Description

DraughtsController takes the part of the controller in the Model-View-Controller system. It acts as an Observer with a single instance of IDraughtsModel acting as the Subject.

Multiple IDraughtsView objects can register with the DraughtsController to be notified when changes to the IDraughtsModel occur. The IDs of the events that are passed to the IDraughtsView objects are defined as public fields in DraughtsController.

Each IDraughtsView can interact with the DraughtsController via public methods. The DraughtsController determines if these interactions should be passed on to the DraughtsModel. The DraughtsController maintains an association between each IDraughtsView and zero, one or two players to determine if interactions should be passed to the IDraughtsModel. The DraughtsController does not provide public access to IDraughtsModel.

# Interface

|  |  |
| --- | --- |
| Attribute | Value |
| Visibility | Public |
| Modifiers | Final |
| Extends | - |
| Implements | java.beans.PropertyChangeListener  IDraughtsController |

# Structure



# Element Descriptions and Interfaces

## Public Methods

### getInstance

This is the factory static method that returns the single instance of a DraughtsController – only one instance is allowed. If it is called more than once it simply returns the previously created instance.

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| IDraughtsController | The single instance of a DraughtsController. |

**Processing Logic**

If instance is null assign a new DraughtsController to it.

Return “instance”.

### addView

This method is called by a view to register itself with the DraughtsController. Only registered views will receive updates from the controller. Multiple views can be registered with the DraughtsController and each of these views will be sent updates when required.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| view | IDraughtsView | View to be added to the controller |

**Outputs**

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

**Processing Logic**

Add view to views. Call registerController on the view passing in instance.

### setModel

This method is called by the model to set itself as the model that the DraughtsController is observing.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | IDraughtsModel | The model to connect to this controller |

**Outputs**

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

**Processing Logic**

Set **this**.model to be model. Call addPropertyChangeListener on model passing in **this**.

### squareClicked

Indicates to the controller that a click has occurred on a square. For use by IDraughtsView implementations. This method will filter out clicks not made by a valid player. For example, if the computer is controlling the black player and black is to move, then no clicks from the other player will be considered valid.

The controller shall instruct the model to propagate the click if it was received from a local view.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| view | IDraughtsView | The view the click came from. |
| squareID | int | The id of the square (1 to 32) |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | True if setClick succeeded. |

**Processing Logic**

If the view is null return false.

The rest of the processing is synchronized on model. If the click has come from a valid human player then tell the model, otherwise do nothing – note that this method has to determine whether or not the click should be propagated to a remote game:

If the view is local – use getType() – and the current player in the model is human – use getCurrentPlayer() – then call setClick with the square ID and true for propagate. Return the value returned from setClick.

If the view is remote and the current player is a remote player (this means the click has come from a remote game) then setClick with the square ID and false for propagate. Return the value returned from setClick.

Return false if none of the above conditions are met.

### propertyChange

This method is used to implement the PropertyChangeListener interface. Any model changes will be sent to this controller through the use of this method. This method will then fire a property change to all registered views.

|  |  |
| --- | --- |
| Attribute | Value |
| Modifiers | - |
| Overrides | PropertyChangeListener.propertyChange |
| Throws | - |

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| event | PropertyChangeEvent | Contains a model change |

**Outputs**

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

**Processing Logic**

Pass event onto each view registered in views using modelPropertyChange.

### newGame

Fires a NEW\_GAME event to all registered views when called.

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

**Inputs**

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

**Outputs**

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

**Processing Logic**

Call propertyChange passing in a new PropertyChangeEvent with **this** as the source, NEW\_GAME as the property, null for the old value and 1 as the new value.

### resetBoard

Resets the board to its default layout, clearing all stored data. This will force a new game dialog to be displayed.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| gameData | GameData | - |

**Outputs**

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

**Processing Logic**

Clear humanPlayers and remotePlayers.

For each player in gameData

* if they are Human add them to humanPlayers
* If they are Remote add them to remotePlayers

Call newGame on model passing in gamedata.

### connectionFailed

Called when a connection to the remote machine has failed. This condition will start a new game.

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

**Inputs**

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

**Outputs**

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

**Processing Logic**

Call error passing in the string “connection failed”.

### error

Called when an unrecoverable error has occurred. This condition will start a new game.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| errorMsg | String | Error string |

**Outputs**

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

**Processing Logic**

Call errorMsg on model passing in errorMsg.

### export

Export the game to a PDN file.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| writer | Writer | The writer to use when exporting PDN data |

**Outputs**

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

**Processing Logic**

Call export on the model passing in writer

### undo

This is called when a move has been undone. This is passed down to the model.

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

**Inputs**

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

**Outputs**

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

**Processing Logic**

If there are no remote players then call undo on model.

## Package Access Methods

None.

## Private Methods

### DraughtsController

Constructs a DraughtsController.

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

**Inputs**

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

**Outputs**

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

**Processing Logic**

Initialise views to be an empty ArrayList.

## Public Fields

None.

## Package Access Fields

None.

## Private Fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Modifiers | Contents | Description |
| humanPlayers | Set<PlayerColor> | - | EnumSet.noneOf( PlayerColor.class ) | Set of players that are not connected via a remote view |
| remotePlayers | Set<PlayerColor> | - | EnumSet.noneOf( PlayerColor.class ) | Set of players that are connected via a remote view |
| model | IDraughtsModel | - | - | The model connected to this controller |
| views | List<IDraughtsView> | - | - | List of all the views registered with this controller |
| instance | IDraughtsController | Static | - | The instance of the DraughtsController. |

# Resource Requirements

None.

# Test Plan

## Test Error message

* 1. Setup the controller with a mock model.
  2. Check that an error message is passed to the model when error is called.

## Connection Failed

* 1. Setup the controller with a mock model.
  2. Check that a connection failed error message is passed to the model when connectionFailed is called.

## Export file

* 1. Setup the controller with a mock model.
  2. Check the model is passed the correct file when export is called.

## Add View

* 1. Setup the controller with a mock view.
  2. Check that the controller is registered with the view.

## Property Change with 2 views

* 1. Setup the controller with 2 mock views.
  2. Check that when a propertyChange is fired both views are informed.

## Start a new game

* 1. Setup the controller with a mock view
  2. Check that the correct property is fired when a new game is initiated.

## Local human clicks when it is their turn

* 1. Setup the controller with a mock view and mock model.
  2. Use resetBoard to start a game with a white computer player and black human player.
  3. Click on square 2, with the mocks indicating a local view and black to play.
  4. Check that setClick is called on the model and the return from squareClicked.

## Local human clicks when not their turn

* 1. Setup the controller with a mock view and mock model.
  2. Use resetBoard to start a game with a white computer player and black human player.
  3. Click on square 2, with the mocks indicating a local view and white to play.
  4. Check that setClick is not called on the model and the return from squareClicked..

## Local human clicks when the human player is remote

* 1. Setup the controller with a mock view and mock model.
  2. Use resetBoard to start a game with a white computer player and black remote player.
  3. Click on square 2, with the mocks indicating a local view and black to play.
  4. Check that setClick is not called on the model and the return from squareClicked.

## Click undo when there are no remote players

* 1. Setup the controller with a mock view and mock model.
  2. Use resetBoard to start a game with two human players.
  3. Call undo on the controller and check that undo is called on the model.

## Click undo when there are remote players

* 1. Setup the controller with a mock view and mock model.
  2. Use resetBoard to start a game with a remote player and a human player.
  3. Call undo on the controller and check that undo is not called on the model.

# Scenarios

None.

# Supporting Documentation

None.

**- End of Document -**