**GameData**

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

**Class Specification**

**Project Reference 999/170**

**Document Reference GameData\_cs**

**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 | R.Miskin | Issued |
| 08/11/2010 | 2 | C Harrison | Correction after first year – attribute table for first constructor corrected, test .3 corrected |
| 02/12/2011 | 3 | Phil Lewis, C Harrison | Updated to reflect new Java standard, including diagrams. |
| 13/06/2013 | 4 | C Harrison | Updated processing logic to explain what’s going on. |
| 27/10/2014 | 5 | L Gilbraith | Removed throws IllegalArgumentException from constructor description. IllegalArgumentException is an unchecked exception i.e. throws clause for this type of exception is redundant. |

Contents

[1 Class Identity 1](#_Toc305145606)

[2 Description 1](#_Toc305145607)

[3 Interface 1](#_Toc305145608)

[4 Structure 1](#_Toc305145609)

[5 Element Descriptions and Interfaces 1](#_Toc305145610)

[5.1 Public Methods 1](#_Toc305145611)

[5.1.1 GameData(PlayerData, PlayerData, String) 1](#_Toc305145612)

[5.1.2 GameData(PlayerData, PlayerData, Reader) 2](#_Toc305145613)

[5.1.3 getBlackPlayer 3](#_Toc305145614)

[5.1.4 getFEN 4](#_Toc305145615)

[5.1.5 getHost 4](#_Toc305145616)

[5.1.6 getWhitePlayer 4](#_Toc305145617)

[5.1.7 isNetworkGame 5](#_Toc305145618)

[5.1.8 toString 5](#_Toc305145619)

[5.2 Package Access Methods 6](#_Toc305145620)

[5.3 Private Methods 6](#_Toc305145621)

[5.3.1 readPDN 6](#_Toc305145622)

[5.4 Public Fields 7](#_Toc305145623)

[5.5 Package Access Fields 7](#_Toc305145624)

[5.6 Private Fields 7](#_Toc305145625)

[6 Resource Requirements 7](#_Toc305145626)

[7 Test Plan 7](#_Toc305145627)

[8 Scenarios 8](#_Toc305145628)

[9 Supporting Documentation 8](#_Toc305145629)

# Class Identity

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

# Description

GameData contains PlayerData information for the Black and White players and a FEN tag describing a board layout. If the FEN tag is empty, then the game should be started with pieces placed in the default locations. There are two constructors for GameData, one that takes a String representing a FEN tag and another that take a java.io.Reader which represents a Portable Draughts Notation (PDN) file from which a FEN tag will be extracted.

# Interface

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

# Structure



# Element Descriptions and Interfaces

## Public Methods

### GameData(PlayerData, PlayerData, String)

Constructs a new GameData with players specified by the two PlayerData parameters and a String representing a FEN tag file. If the String is null it is ignored.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| black | PlayerData | The black player data |
| white | PlayerData | The white player data |
| fen | String | A string representing a FEN tag |

**Outputs**

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

**Processing Logic**

* If black or white are null throw an IllegalArgumentException with an appropriate detail message.
* Set **this**.black to black
* Set **this**.white to white
* If fen is not null assign it to **this**.fen, otherwise assign an empty String to **this**.fen

Now work out if we have a remote player and if we have, then it’s a network game and the host is the name of the remote player:

* If **this**.black has type PlayerType.REMOTE
  + If **this**.white has PlayerType.REMOTE
    - throw an IllegalArgumentException “Only one player may be remote”.
  + Otherwise
    - Set networkGame to true
    - Assign **this**.black.getName() to host
* Otherwise if **this**.white has type PlayerType.Remote
  + Set networkGame to true
  + Assign **this**.white.getName() to host
* Otherwise
  + Set networkGame to false
  + Assign an empty String to host.

### GameData(PlayerData, PlayerData, Reader)

Constructs a new GameData with players specified by the two PlayerData parameters and java.lang.Reader that represents PDN data to import. If the Reader is null it is ignored. If the Reader is non-null it will be used to read the PDN data and the FEN tag is stored.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| black | PlayerData | The black player data |
| white | PlayerData | The white player data |
| pdn | Reader | A Reader representing some PDN data to import. |

**Outputs**

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

**Processing logic**

* Use readPDN(pdn) to obtain a FEN tag
* Invoke the (PlayerData, PlayerData, String) constructor with black, white, and the FEN tag.

### getBlackPlayer

Returns the PlayerData describing the black player.

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| PlayerData | PlayerData describing the black player. |

**Processing Logic**

* Return black

### getFEN

Returns the FEN tag that this GameData was constructed with or an empty String if none was specified.

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| String | The FEN tag that the GameData was constructed with, or an empty String. |

**Processing Logic**

* Return fen

### getHost

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| String | A String representing the remote host to use for a network game. An empty String is returned if there is no remote host. |

**Processing Logic**

* Return host

### getWhitePlayer

Returns the PlayerData describing the white player.

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| PlayerData | PlayerData describing the white player. |

**Processing Logic**

* Return white

### isNetworkGame

Returns true if one of the players is remote, otherwise false.

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| boolean | true if one of the players is remote, otherwise false |

**Processing Logic**

* Return networkGame

### toString

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

**Inputs**

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

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| String | A representation of this GameData. |

**Processing Logic**

Return a string comprising of "Black: " + black.toString() + ". White: " + white.toString() + ". PDN: " + fen + "."

## Package Access Methods

None

## Private Methods

### readPDN

This method uses the supplied java.io.Reader and extracts a FEN tag. It will loop through the Reader looking for a line like [FEN “B:B1,2,K6:W12”] and if it finds it, it will return the contents of the quoted string e.g. B:B1,2,K6:W12.

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

**Inputs**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| pdn | Reader | A Reader representing a PDN file to import. |

**Outputs**

|  |  |
| --- | --- |
| Type | Description |
| String | The FEN tag from the supplied PDN file, or an empty String. |

**Processing Logic**

* If pdn is null return an empty String
* Create a BufferedReader, f, with pdn
* Create a local variable String readLine
* do
  + readLine = f.readLine();
  + if readLine != null readLine = readLine.toUpperCase();
* while(readLine != null && readLine doesn’t start with “[FEN”);
* Close the BufferedReader
* If readLine is null
  + return an empty String
* Otherwise
  + Extract the quoted String from the FEN tag and return that.
* If there is an IOException while using the Reader return an empty String.

## Public Fields

None

## Package Access Fields

None

## Private Fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Attributes | Value | Description |
| black | PlayerData | final | - | PlayerData for the black player |
| fen | String | final | - | - |
| host | String | final | - | The remote host to use if this is a network game. Will be an empty String if neither player is remote. |
| networkGame | boolean | final | - | True if one player is remote, otherwise false |
| white | PlayerData | final | - | PlayerData for the white player |

# Resource Requirements

None

# Test Plan

You should refer to the System Requirements Specification document for details of the format of the PDN file.

* 1. Create several GameData objects with different PlayerData and non-null FEN tags. Check that getWhitePlayer, getBlackPlayer, getFEN and isNetworkGame return the expected values
  2. Check that an IlegalArgumentException is thrown in each of the following cases
     1. black is null,
     2. white is null,
     3. Both black and white are constructed with PlayerType.Remote.
  3. Construct GameData objects with one player of type PlayerType.Remote and check getHost() is as expected. Construct a GameData object with no remote players and check getHost() is as expected.
  4. Construct a GameData with a null String and check that getFEN() returns an empty String
  5. Construct several GameData objects using a java.io.StringReader and test the following
     1. getFEN() returns an empty String if there is no FEN tag
     2. getFEN() returns the expected String when there is a FEN tag
  6. Construct a GameData object with a Reader that throws an IOException and check that getFEN() returns an empty String. The simplest way to do this is to create a new class that extends StringReader and override the read(char[], int, int) method such that it throws an IOException. You can then construct the GameData object with an instance of this new class.

**Note:** additional tests will need to be added to get full coverage, try to add these tests yourself.

# Scenarios

None

# Supporting Documentation

The Portable Draughts Notation (PDN) format is detailed in the System Requirements Specification.

**- End of Document -**