Othello, PROP 20-21 Q2

Third assignment. Group 13-3.2

Generated by Doxygen 1.8.18

Chapter 1

Othello, PROP 20-21 Q2.

This project is a Java implementation of the Othello game, also known as Reversi. The architecture used is a three layer design composed of the view, domain and repository classes. Classes of different layers are interconnected via their designated controllers. For testing, Drivers for all clases have been made, and also, unitary tests with the JUnit library for the Ranking and Entry classes. For persisting program data, we have opted for local JSON files using the org.JSON library.

Detailed Domain composition:

- Util Classes:
  - *Pair*
- Domain Classes:
  - *Player*
    - *User*
    - *Bot*
  - *Configuration*
  - *Game*
  - *Board*
  - *Difficulty*
    - *EasyDifficulty*
    - *MediumDifficulty*
    - *HardDifficulty*
  - *Entry*
  - *Ranking*
- Domain Controllers:
  - *PlayerCtrl*
  - *ConfigurationCtrl*
  - *GameCtrl*
  - *BoardCtrl*
  - *DifficultyCtrl*
  - *RankingCtrl*

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Package cmd

Packages

- package driver
- package unitary

Classes

- class othello

  Othello application entrypoint. By Alex Rodriguez.

5.2 Package cmd.driver

Classes

- class board

  Board driver entrypoint. By Alex Rodriguez.
- class bot

  Bot driver entrypoint. By Alex Rodriguez.
- class configuration

  Configuration driver entrypoint. By Alex Rodriguez.
- class easyDifficulty

  EasyDifficulty driver entrypoint. By Alex Rodriguez.
- class game

  Game driver entrypoint. By Alex Rodriguez.
- class hardDifficulty

  HardDifficulty driver entrypoint. By Alex Rodriguez.
- class mediumDifficulty

  MediumDifficulty driver entrypoint. By Alex Rodriguez.
- class pair

  Pair driver entrypoint. By Alex Rodriguez.
- class user

  User driver entrypoint. By Alex Rodriguez.

5.3 Package cmd.unitary

Classes

- class entry

  JUnit Entry tests entrypoint. By Alex Rodriguez.
- class ranking

  JUnit Ranking tests entrypoint. By Alex Rodriguez.

5.4 Package domain

Classes

- class Board
- class BoardCtrl

  This class represents the controller of the Board class, which is the classs that will be used to communicate with the other controllers.
- class Bot

  Represents a bot in our system.
- class Configuration

  Represents the rules of an Othello game including its name, whether the pieces can be eaten horizontally, vertically or diagonally, and its creator. By Alex Rodriguez.
- class ConfigurationCtrl

  Configuration domain sub-controller. It communicates with the main domain controller, the configuration repository controller and the game repository controller for certain integrity checks. It is also in charge of retrieving the initial boards associated with the configurations.
- class Difficulty

  Implements the abstract class and methods of all the difficulty implementations. By Arnau Pujantell.
- class DifficultyCtrl

  Difficulty domain sub-controller. Is in charge of EasyDifficulty, MediumDifficulty and HardDifficulty. It communicates with the main domain controller. It forwards the current bot's placePiece request to the correct algorithm depending on the current game's difficulty: 1 to 3: EasyDifficulty (Minimax). 4 to 6: MediumDifficulty (Minimax alpha beta pruning). 7 to 10: HardDifficulty (Montecarlo).
- class DomainCtrl

  Is the main domain controller. It keeps the current state of all the game and application. It serves as a forwarder for the specific domain class controllers, that's why most of the sub-controller methods receive as a parameter the current instance of the associated class. Moreover, it implements a few methods that do not have a specific domain class binded. It also gathers all the thrown exceptions in the sub-controllers and transforms them into string messages in order to pass them to the view layer without coupling any domain specific logic. By Manuel Navid.
- class EasyDifficulty

  Implements the Minimax algorithm to get the next best possible position for a given player. By Manuel Navid.
- class Entry

  Represents an entry in a Ranking table.
- class Exceptions

  Holds all the different custom Exceptions used in the whole project. By Alex Rodriguez.
- class Game

  Represents the state of an Othello game including its name, players, the current turn, the state, the configuration used, the winner if any, its creator and the creation timestamp. By Alex Rodriguez.
- class GameCtrl

  Game domain sub-controller. It communicates with the main domain controller, the game repository controller, the configuration repository controller and the player repository controller in order to source the necessary components to manage games.

- class HardDifficulty

  Implements the Monte Carlo Tree Search algorithm to get the next best possible position for a given player. By Roger Mollon.
- class MediumDifficulty

  Implements the Minimax algorithm with alpha-beta pruning to get the next best possible position for a given player. By Alex Rodriguez.
- class Player

  Represents a player in our system.
- class PlayerCtrl

  Player class controller.
- class Ranking

  Representation of a ranking table.
- class RankingCtrl

  Ranking domain sub-controller. It communicates with the main domain controller and the ranking repository controller. By Alex Rodriguez.
- class User

  Represents a human user in our system.

5.5 Package repository

Classes

- class ConfigurationRepository

  Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.
- class ConfigurationRepositoryCtrl

  Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.
- class FixtureRepository

  Implements various CRUD operations to work with the Fixture repository. By Alex Rodriguez.
- class GameRepository

  Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.
- class GameRepositoryCtrl

  Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.
- class PlayerRepository

  Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.
- class PlayerRepositoryCtrl

  Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.
- class RankingRepository

  Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.
- class RankingRepositoryCtrl

  Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.
- class Repository

  Implements various CRUD operations to work with the local file system JSON databases and TXT fixtures. By Alex Rodriguez.

5.6 Package test

Packages

- package driver
- package unitary

5.7 Package test.driver

Classes

- class BoardDriver
- class BotDriver
- class ConfigurationDriver

  Implements the different options for the Configuration driver application. By Alex Rodriguez.
- class Driver

  Implements various utilities to create a driver application. By Alex Rodriguez.
- class EasyDifficultyDriver

  Implements the different options for the EasyDifficulty driver application. By Manuel Navid.
- class GameDriver

  Implements the different options for the Game driver application. By Alex Rodriguez.
- class HardDifficultyDriver

  Implements the different options for the HardDifficulty driver application. By Roger Mollon.
- class MediumDifficultyDriver

  Implements the different options for the MediumDifficulty driver application. By Alex Rodriguez.
- class PairDriver

  Implements the different options for the Pair driver application. By Alex Rodriguez.
- class UserDriver

5.8 Package test.unitary

Classes

- class EntryJUnit

  Allows JUnit testing of class Entry.
- class RankingJUnit

  Allows JUnit testing of class Ranking.

5.9 Package util

Classes

- class Pair

  Implements a data structure containing two generic types. By Alex Rodriguez.

5.10 Package view

Classes

- class BotsConsultView
- class BotsCreateView
- class BotsModifyView
- class BotsView
- class ConfigConsultView
- class ConfigCreateView
- class ConfigModifyView
- class ConfigView
- class ConsultInitialBoardView
- class GameBoardView
- class GamesCreateView
- class GamesView
- class InitialBoardView
- class LogInView
- class ModifyInitialBoardView
- class PlayView
- class RankingConsultView
- class RankingView
- class RecordConsultView
- class SignUpView
- class UserDeleteView
- class UserModifyView
- class UserView
- class ViewCtrl

Chapter 6

Class Documentation

6.1 domain.Exceptions.BadConfirmationException Class Reference

The entered confirmation password doesn't match the user's password. By Alex Rodriguez.

Public Member Functions

- BadConfirmationException ()

6.1.1 Detailed Description

The entered confirmation password doesn't match the user's password. By Alex Rodriguez.

Definition at line 52 of file Exceptions.java.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 BadConfirmationException()

```
domain.Exceptions.BadConfirmationException.BadConfirmationException ( )
```

Definition at line 53 of file Exceptions.java.

```
53                                   {
54              super("ERR_BAD_CONFIRMATION");
55          }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.2 cmd.driver.board Class Reference

Board driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)
  Board driver main function. Creates an instance of the Board driver and starts it.

6.2.1 Detailed Description

Board driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file board.java.

6.2.2 Member Function Documentation

6.2.2.1 main()

```
static void cmd.driver.board.main (
            String[ ] args )  [static]
```

Board driver main function. Creates an instance of the Board driver and starts it.

Precondition

True.

Postcondition

The Board driver has started.

Definition at line 22 of file board.java.

```
                                              {
            new BoardDriver().start();
        }
```

The documentation for this class was generated from the following file:

- board.java

6.3 domain.Board Class Reference

Classes

- enum PieceType
  The status of a cell of the Board. An Othello Board is composed of 64 cells with their own unique position and three possible states:

Public Member Functions

- Board ()

  Creator method that instances a default Othello Board.
- Board (JSONObject jsonBoard)

  Creator method that instances a Board based off a JSON object jsonBoard.
- Board (PieceType[ ][ ] board)

  Creator method that instances a Board based off another board container (matrix of PieceTypes).
- JSONObject serialize ()

  Method that transforms the implicit parameter's board into a JSON format.
- PieceType[ ][ ] getBoard ()

  Get method that returns the implicit parameter's board attribute.
- Integer getPiecesPlayer1 ()

  Get method that returns the value of the implicit parameter's PiecesPlayer1 attribute.
- Integer getPiecesPlayer2 ()

  Get method that returns the value of the implicit parameter's PiecesPlayer2 attribute.
- void isValid (Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally) throws InvalidBoardException

  Method that warns us if an instance of a Board is invalid.
- ArrayList< Pair< Integer, Integer > > validPositions (PieceType myPieceType, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally)

  Method that returns an Array of the valid positions a player myPieceType taking into consideration the Configuration of the Game.
- void removePiece (Pair< Integer, Integer > position)

  Modifying method that removes a piece from the implicit parameter's board attribute.
- void placePiece (Pair< Integer, Integer > position, PieceType myPieceType, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally)

  Modifying method that adds a piece in the implicit parameter's board
- void placePieceConfig (Pair< Integer, Integer > position, PieceType myPieceType)

  Modifying method that adds a piece in the in the implicit parameter's board, which corresponds to an Initial Board of a Configuration.

Private Member Functions

- Boolean surroundingPieces (Pair< Integer, Integer > position, PieceType myPieceType, Boolean canEat↵ Horizontally, Boolean canEatVertically, Boolean canEatDiagonally)

  Private method that returns true if there is an opponent's PieceType surrounding a position in the board taking into account the capturing methods of the Game (Horizontal,Vertical or Diagonal).
- ArrayList< String > transcribeToCharacters ()

  Private method that returns an array of strings to transcribe the implicit parameter's board into a storing format.
- void transcribeToPieceType (String row, Integer numRow)

  Private method that adds a row of a board in the storing format into the board attribute of the implicit parameter.
- PieceType inversePlayer (PieceType myPieceType)

  Private method that inverts the Player's pieceType.
- ArrayList< Pair< Integer, Integer > > canPlaceHorizontal (Pair< Integer, Integer > position, PieceType myPieceType)

  Private method that returns an array of positions of the board in which you can conquer the pieces between them (horizontal search).
- ArrayList< Pair< Integer, Integer > > canPlaceVertical (Pair< Integer, Integer > position, PieceType my↵ PieceType)

  Private method that returns an array of positions of the board in which you can conquer the pieces between them (vertical search).

- ArrayList< Pair< Integer, Integer > > canPlaceDiagonal (Pair< Integer, Integer > position, PieceType my↩ PieceType)

  Private method that returns an array of positions of the board in which you can conquer the pieces between them (diagonal search).
- void changePieces (Pair< Integer, Integer > addPiece, Pair< Integer, Integer > lastPiece, PieceType my↩ PieceType)

  Private method that changes the pieces between two positions of the board.

Private Attributes

- PieceType[ ][ ] board

  A matrix of 64 cells that composes an Othello board. Its the data structure that stores the different cells of the Board.
- Integer piecesPlayer1

  PLAYER1's total number of pieces on the Board.
- Integer piecesPlayer2

  PLAYER2's total number of pieces on the Board.

6.3.1 Detailed Description

This class represents an Othello Board in our project.

Done by Manuel Navid

Definition at line 18 of file Board.java.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Board() [1/3]

`domain.Board.Board ( )`

Creator method that instances a default Othello Board.

Precondition

True

Postcondition

A new instance of Board is instanced with the default values inserted: 2 white pieces in the middle of the board crossed by 2 black pieces.

Therefore, *piecesPlayer1* = 2 and *piecesPlayer2* = 2.

Definition at line 51 of file Board.java.

```
    {
        this.board = new PieceType[8][8];

        //Initial Pieces
        this.board[3][3] = PieceType.PLAYER1;
        this.board[4][4] = PieceType.PLAYER1;
        this.board[4][3] = PieceType.PLAYER2;
        this.board[3][4] = PieceType.PLAYER2;

        this.piecesPlayer1 = 2;
        this.piecesPlayer2 = 2;
    }
```

6.3.2.2 Board() [2/3]

```
domain.Board.Board (
            JSONObject jsonBoard )
```

Creator method that instances a Board based off a JSON object *jsonBoard*.

Precondition

True

Postcondition

A new instance of Board is instanced with the *board* attribute equal to a board given to us by the JSON object *jsonBoard*.

In addition, the attributes *PiecesPlayer1* and *PiecesPlayer2* will have different values based off of the modified *board* attribute.

Parameters

jsonBoard	JSON object that stores a state of an Othello board (8 rows with 8 elements each with characters equal to: B,N or ?)

Definition at line 72 of file Board.java.

```
    {
        this.board = new PieceType[8][8];
        this.piecesPlayer1 = 0;
        this.piecesPlayer2 = 0;

        String row0 = jsonBoard.getString("row0");
        String row1 = jsonBoard.getString("row1");
        String row2 = jsonBoard.getString("row2");
        String row3 = jsonBoard.getString("row3");
        String row4 = jsonBoard.getString("row4");
        String row5 = jsonBoard.getString("row5");
        String row6 = jsonBoard.getString("row6");
        String row7 = jsonBoard.getString("row7");

        this.transcribeToPieceType(row0, 0);
        this.transcribeToPieceType(row1, 1);
        this.transcribeToPieceType(row2, 2);
        this.transcribeToPieceType(row3, 3);
        this.transcribeToPieceType(row4, 4);
        this.transcribeToPieceType(row5, 5);
        this.transcribeToPieceType(row6, 6);
        this.transcribeToPieceType(row7, 7);
    }
```

6.3.2.3 Board() [3/3]

```
domain.Board.Board (
            PieceType board[ ][ ] )
```

Creator method that instances a Board based off another board container (matrix of PieceTypes).

Precondition

The parameter *board* is of size 8x8.

Postcondition

An instance of Board is instanced with the *board* attribute equal to the *board* parameter.

In addition, the attributes *PiecesPlayer1* and *PiecesPlayer2* will have different values based off of the new *board* attribute.

Parameters

board	An 8x8 PieceType matrix that represents a state of an Othello board.

Definition at line 104 of file Board.java.

```
    {
        this.board = new PieceType[8][8];
        this.piecesPlayer1 = 0;
        this.piecesPlayer2 = 0;

        for(int i = 0; i < 8; ++i)
        {
            for(int j = 0; j < 8; j++)
            {
                this.board[i][j] = board[i][j];
                if(this.board[i][j] == PieceType.PLAYER1) this.piecesPlayer1 += 1;
                if(this.board[i][j] == PieceType.PLAYER2) this.piecesPlayer2 += 1;
            }
        }
    }
```

6.3.3 Member Function Documentation

6.3.3.1 serialize()

`JSONObject domain.Board.serialize ( )`

Method that transforms the implicit parameter's *board* into a JSON format.

Precondition

True

Postcondition

returns a JSON object that corresponds to the transformation of the implicit parameter's *board* attribute into the storing format we decided in class.

Definition at line 128 of file Board.java.

```
{
    ArrayList<String> boardCodified = this.transcribeToCharacters();
    JSONObject jsonBoard = new JSONObject();

    jsonBoard.put("row0", boardCodified.get(0));
    jsonBoard.put("row1", boardCodified.get(1));
    jsonBoard.put("row2", boardCodified.get(2));
    jsonBoard.put("row3", boardCodified.get(3));
    jsonBoard.put("row4", boardCodified.get(4));
    jsonBoard.put("row5", boardCodified.get(5));
    jsonBoard.put("row6", boardCodified.get(6));
    jsonBoard.put("row7", boardCodified.get(7));

    return jsonBoard;
}
```

6.3.3.2 getBoard()

`PieceType [][] domain.Board.getBoard ( )`

Get method that returns the implicit parameter's *board* attribute.

Precondition

True

Postcondition

The implicit parameter's *board* is returned.

Definition at line 152 of file Board.java.

```
{
    return this.board;
}
```

6.3.3.3 getPiecesPlayer1()

`Integer domain.Board.getPiecesPlayer1 ( )`

Get method that returns the value of the implicit parameter's *PiecesPlayer1* attribute.

Precondition

True

Postcondition

The implicit parameter's *piecesPlayer1* value is returned.

Definition at line 162 of file Board.java.

```
{
    return this.piecesPlayer1;
}
```

6.3.3.4 getPiecesPlayer2()

```
Integer domain.Board.getPiecesPlayer2 ( )
```

Get method that returns the value of the implicit parameter's *PiecesPlayer2* attribute.

Precondition

True

Postcondition

The implicit parameter's *piecesPlayer2* value is returned.

Definition at line 172 of file Board.java.

```
    {
        return this.piecesPlayer2;
    }
```

6.3.3.5 isValid()

```
void domain.Board.isValid (
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally ) throws InvalidBoardException
```

Method that warns us if an instance of a Board is invalid.

An invalid Board means that no player can add a piece in the current state of the implicit parameter's *board* attribute.

Precondition

All parameters aren't null.

Postcondition

If the Board instance is invalid, InvalidBoardException will be thrown, else nothing.

Definition at line 183 of file Board.java.

```
    {
        ArrayList<Pair<Integer, Integer>> player1 = validPositions(PieceType.PLAYER1, canEatHorizontally,
    canEatVertically, canEatDiagonally);
        ArrayList<Pair<Integer, Integer>> player2 = validPositions(PieceType.PLAYER2, canEatHorizontally,
    canEatVertically, canEatDiagonally);
        //If there is no possible movements == Board Invalid
        if(player1.isEmpty() && player2.isEmpty()) throw new InvalidBoardException();
    }
```

6.3.3.6 validPositions()

```
ArrayList<Pair<Integer,Integer> > domain.Board.validPositions (
              PieceType myPieceType,
              Boolean canEatHorizontally,
              Boolean canEatVertically,
              Boolean canEatDiagonally )
```

Method that returns an Array of the valid positions a player *myPieceType* taking into consideration the Configuration of the Game.

Precondition

All parameters aren't null.

Postcondition

An Array of valid positions(Pair<Integer,Integer>) is returned.

A valid position is one which it's cell state in the implicit parameter's *board* attribute is equal to null (meaning an empty cell) and there is at least one opponent PieceType surrounding that position (go to surroundingPieces to crystalize what the surrounding areas of a position are).

Parameters

myPieceType	PieceType variable that represents the player in a cell.
canEatHorizontally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Horizontal manner.
canEatVertically	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Vertical manner.
canEatDiagonally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Diagonal manner.

Definition at line 203 of file Board.java.

```
    {
        ArrayList<Pair<Integer,Integer» availablePos = new ArrayList<Pair<Integer,Integer»();
        boolean posValid = false;

        for(int i = 0; i < 8; ++i)
        {
            for(int j = 0; j < 8; ++j)
            {
                Pair<Integer, Integer> iterator = new Pair<Integer,Integer>(i,j);
                posValid = false;

                if(this.board[i][j] == null && surroundingPieces(iterator, myPieceType,
      canEatHorizontally, canEatVertically, canEatDiagonally))
                {
                    if(canEatHorizontally)
                    {
                        ArrayList<Pair<Integer,Integer» horizontal = canPlaceHorizontal(iterator,
      myPieceType);
                        //IF NOT EMPTY
                        if(!horizontal.isEmpty()) posValid = true;
                    }
                    if(canEatVertically)
                    {
                        ArrayList<Pair<Integer,Integer» vertical = canPlaceVertical(iterator,
      myPieceType);
                        //IF NOT EMPTY
                        if(!vertical.isEmpty()) posValid = true;
                    }
                    if(canEatDiagonally)
                    {
                        ArrayList<Pair<Integer,Integer» diagonal = canPlaceDiagonal(iterator,
    myPieceType);
                        //IF NOT EMPTY
                        if(!diagonal.isEmpty()) posValid = true;
                    }
                    //It's a valid position to add a Piece
                    if(posValid) availablePos.add(iterator);
                }
            }
        }

        return availablePos;
    }
```

6.3.3.7 removePiece()

```
void domain.Board.removePiece (
            Pair< Integer, Integer > position )
```

Modifying method that removes a piece from the implicit parameter's *board* attribute.

Precondition

The *position* parameter isn't null and has values between (0,0) and (7,7).

Postcondition

In the implicit parameter's *board*, the state of the cell in position *position* is converted to null, which means that now it's an empty cell on the board.

Parameters

position	Pair<Integer,Integer> that represents a position in a board.

Definition at line 252 of file Board.java.

```
    {
            Integer row = position.first;
            Integer column = position.second;

            if(this.board[row][column] == PieceType.PLAYER1) this.piecesPlayer1 -= 1;
            if(this.board[row][column] == PieceType.PLAYER2) this.piecesPlayer2 -= 1;

            this.board[row][column] = null;
    }
```

6.3.3.8 placePiece()

```
void domain.Board.placePiece (
            Pair< Integer, Integer > position,
            PieceType myPieceType,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally )
```

Modifying method that adds a piece in the implicit parameter's *board*

In addition, it applies the effect of adding that piece in the board by changing the pieces of the board taking into consideration the Configuration given.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

With the given Configuration, if the *position* parameter is correct then the implicit parameter's *board* will be modified with the addition of the piece *Piecetype* in the *position* parameter and its effect considering the Configuration given (pieces changing from the different taking piece methods). If the position isn't correct, the implicit parameter's *board* will not be changed.

A correct position is a position in the board where given the *PieceType* parameter, we will take at least one opponent piece with the Configuration given.

Parameters

myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.
canEatHorizontally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Horizontal manner..
canEatVertically	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Vertical manner.
canEatDiagonally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Diagonal manner.

Definition at line 277 of file Board.java.

```
    {
        ArrayList<Pair<Integer,Integer>> horizontal = new ArrayList<Pair<Integer,Integer>>();
        ArrayList<Pair<Integer,Integer>> vertical = new ArrayList<Pair<Integer,Integer>>();
        ArrayList<Pair<Integer,Integer>> diagonal = new ArrayList<Pair<Integer,Integer>>();

        //if the position given to us is not null, it means it's owned by PLAYER1 or PLAYER2. Therefore,
    we won't add a Piece there and we will return.
        //Although this will never happen when we use this method (because we will make sure it's a
    valid position),
        //  we added this so this method is more reusable for other future projects.
        if(this.board[position.first][position.second] != null) return;

        if(canEatHorizontally) //Includes eating HORIZONTALLY activated
        {
            horizontal = canPlaceHorizontal(position, myPieceType);
            for(int i = 0; i < horizontal.size(); i++)
                changePieces(position,horizontal.get(i),myPieceType);
        }

        if (canEatVertically) //Includes eating VERTICALLY activated
        {
            vertical = canPlaceVertical(position, myPieceType);
            for(int i = 0; i < vertical.size(); i++) {
                changePieces(position,vertical.get(i),myPieceType);
            }
        }

        if (canEatDiagonally) //Includes eating DIAGONALLY activated
        {
            diagonal = canPlaceDiagonal(position, myPieceType);
            for(int i = 0; i < diagonal.size(); i++)
                changePieces(position,diagonal.get(i),myPieceType);
        }
        //If we added a piece to the board, we must add this to the piecesPlayerx attribute
        if((canEatHorizontally && !horizontal.isEmpty()) || (canEatVertically && !vertical.isEmpty())
    || (canEatDiagonally && !diagonal.isEmpty()))
        {
            if(myPieceType == PieceType.PLAYER1) this.piecesPlayer1++;
            if(myPieceType == PieceType.PLAYER2) this.piecesPlayer2++;
        }
    }
```

6.3.3.9 placePieceConfig()

```
void domain.Board.placePieceConfig (
            Pair< Integer, Integer > position,
            PieceType myPieceType )
```

Modifying method that adds a piece in the in the implicit parameter's *board*, which corresponds to an Initial Board of a Configuration.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

The implicit parameter's *board* will be modified with the addition of the piece *PieceType* in position *position*.

Parameters

myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.

Definition at line 324 of file Board.java.

```
    {
        Integer row = position.first;
        Integer column = position.second;

        if(this.board[row][column] == PieceType.PLAYER1 && myPieceType == PieceType.PLAYER2)
        {
            this.piecesPlayer2 += 1;
            this.piecesPlayer1 -= 1;
        }

        if(this.board[row][column] == PieceType.PLAYER2 && myPieceType == PieceType.PLAYER1)
        {
            this.piecesPlayer1 += 1;
            this.piecesPlayer2 -= 1;
        }

        if(this.board[row][column] == null && myPieceType == PieceType.PLAYER1) this.piecesPlayer1 += 1;
        if(this.board[row][column] == null && myPieceType == PieceType.PLAYER2) this.piecesPlayer2 += 1;

        this.board[row][column] = myPieceType;
    }
```

6.3.3.10 surroundingPieces()

```
Boolean domain.Board.surroundingPieces (
            Pair< Integer, Integer > position,
            PieceType myPieceType,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally )  [private]
```

Private method that returns true if there is an opponent's PieceType surrounding a position in the board taking into account the capturing methods of the Game (Horizontal,Vertical or Diagonal).

This method is particularly useful to check if a position is valid, which means it's eligible to be chosen as a viable option to place a piece in.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

Returns *true* if there is an opponent's PieceType surrounding the *position* parameter in the board taking into account the capturing methods of the Game.

To crystalize what a piece surrounding a position is, its all the possible positions one can reach adding or substracting 1 to the y or x value (taking into consideration the board's limits obviously).

Parameters

myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.
canEatHorizontally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Horizontal manner.
canEatVertically	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Vertical manner.
canEatDiagonally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Diagonal manner.

Definition at line 362 of file Board.java.

```
    {
        PieceType opponentPiece = inversePlayer(myPieceType);

        if(canEatDiagonally)
        {
            //TOP LEFT
            if(position.first > 0 && position.second > 0 &&
this.board[position.first-1][position.second-1] == opponentPiece) return true;
            //TOP RIGHT
            if(position.first > 0 && position.second < 7 &&
this.board[position.first-1][position.second+1] == opponentPiece) return true;
            //BOTTOM RIGHT
            if(position.first < 7 && position.second < 7 &&
this.board[position.first+1][position.second+1] == opponentPiece) return true;
            //BOTTOM LEFT
            if(position.first < 7 && position.second > 0 &&
this.board[position.first+1][position.second-1] == opponentPiece) return true;
        }

        if(canEatVertically)
        {
            //BOTTOM
            if(position.first < 7 && this.board[position.first+1][position.second] == opponentPiece)
return true;
```

```
            //TOP
            if(position.first > 0 && this.board[position.first-1][position.second] == opponentPiece) return true;
        }

        if(canEatHorizontally)
        {
            //RIGHT
            if(position.second < 7 && this.board[position.first][position.second+1] == opponentPiece) return true;
            //LEFT
            if(position.second > 0 && this.board[position.first][position.second-1] == opponentPiece) return true;
        }
        //If none are true
        return false;
    }
```

6.3.3.11 transcribeToCharacters()

`ArrayList<String> domain.Board.transcribeToCharacters ( )  [private]`

Private method that returns an array of strings to transcribe the implicit parameter's *board* into a storing format.

Precondition

True

Postcondition

Returns an array of Strings size 8 that transcribes the implicit parameter's *board* into the storing format decided in class.

The storing format is: ? -> empty cell, B -> PLAYER1's piece, N -> PLAYER2's piece.

Definition at line 403 of file Board.java.

```
    {
        ArrayList<String> boardCodified = new ArrayList<String>(8);
        String operational = "";

            for(int i = 0; i < 8; ++i)
            {
                operational = "";
                for(int j = 0; j < 8; ++j)
                {
                    if(this.board[i][j] == PieceType.PLAYER1) operational = operational + "B";
                    if(this.board[i][j] == PieceType.PLAYER2) operational = operational + "N";
                    if(this.board[i][j] == null) operational = operational + "?";
                }
                boardCodified.add(operational);
            }

        return boardCodified;
    }
```

6.3.3.12 transcribeToPieceType()

```
void domain.Board.transcribeToPieceType (
            String row,
            Integer numRow )  [private]
```

Private method that adds a row of a board in the storing format into the *board* attribute of the implicit parameter.

This method is useful to load a Board from a file.

Precondition

Parameters aren't null and numRow has a value between 0 and 7.

Postcondition

The implicit parameter's *board* attribute is modified with a new row number *numRow* with the values specified. in the *row* parameter.

Parameters

row	String of a row of a board in storing format.
numRow	Number of the row in the board its transcribing.

Definition at line 432 of file Board.java.

```
    {
        for(int i = 0; i < 8; ++i)
        {
            if(row.charAt(i) == '?') this.board[numRow][i] = null;
            if(row.charAt(i) == 'B')
            {
                this.board[numRow][i] = PieceType.PLAYER1;
                this.piecesPlayer1++;
            }
            if(row.charAt(i) == 'N')
            {
                this.board[numRow][i] = PieceType.PLAYER2;
                this.piecesPlayer2++;
            }
        }
    }
```

6.3.3.13 inversePlayer()

```
PieceType domain.Board.inversePlayer (
            PieceType myPieceType )  [private]
```

Private method that inverts the Player's pieceType.

This method is particularly useful to get the opponent's PieceType in another method.

Precondition

myPieceType isn't null.

Postcondition

Returns a PieceType that is the opponent of *myPieceType*

Parameters

myPieceType	PieceType variable that represents the player in a cell.

Definition at line 457 of file Board.java.

```
    {
        if(myPieceType == PieceType.PLAYER1) return PieceType.PLAYER2;
        else return PieceType.PLAYER1;
    }
```

6.3.3.14 canPlaceHorizontal()

```
ArrayList< Pair<Integer,Integer> > domain.Board.canPlaceHorizontal (
            Pair< Integer, Integer > position,
            PieceType myPieceType ) [private]
```

Private method that returns an array of positions of the board in which you can conquer the pieces between them (horizontal search).

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

Returns an array of positions in which you can use with the method changePieces to conquer the pieces between them and the *position<e/m> parameter (which corresponds to the position we want to add a piece to).*

Parameters

position	*Pair<Integer,Integer> that represents a position in a board.*
myPieceType	*PieceType variable that represents the player in a cell.*

Definition at line 471 of file Board.java.

```
    {
        ArrayList< Pair<Integer,Integer> > result = new ArrayList<Pair<Integer, Integer»();

        Integer row = position.first;
        Integer column = position.second;
        PieceType opponentPiece = inversePlayer(myPieceType);

        if(column > 0) //To not go out of the boards boundaries
        {
            //to check if we can eat LEFT SIDE
            if(this.board[row][column-1] == opponentPiece)
            {
                Integer it1 = column-1;
                Boolean found1 = false;

                while(it1 >= 0 && this.board[row][it1] != null && found1 == false)
                //go through the line of the board to see if we can place the piece we want in
    "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[row][it1] == myPieceType) //found a piece that's mine = CAN PLACE
                    {
                        result.add(new Pair<Integer,Integer>(row,it1));
                        found1 = true;
                    }
                    else //found another piece of the opponent = CONTINUE THE HUNT
                        it1 -= 1;
                }
            }
        }

        if(column < 7) //To not go out of the boards boundaries
        {
            //to check if we can eat RIGHT SIDE
            if(this.board[row][column+1] == opponentPiece)
            {
                Integer it2 = column+1;
                Boolean found2 = false;

                while(it2 <= 7 && this.board[row][it2] != null && found2 == false)
                //go through the line of the board to see if we can place the piece we want in
       "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[row][it2] == myPieceType) //found a piece that's mine = CAN PLACE
                    {
                        result.add(new Pair<Integer,Integer>(row,it2));
                        found2 = true;
                    }
                    else //found another piece of the opponent = CONTINUE THE HUNT
                    {
                        it2 += 1;
                    }
                }
            }
        }

        return result;
    }
```

6.3.3.15 canPlaceVertical()

```
ArrayList< Pair<Integer,Integer> > domain.Board.canPlaceVertical (
            Pair< Integer, Integer > position,
            PieceType myPieceType ) [private]
```

Private method that returns an array of positions of the board in which you can conquer the pieces between them (vertical search).

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

Returns an array of positions in which you can use with the method changePieces to conquer the pieces between them and the *position<e/m> parameter (which corresponds to the position we want to add a piece to).*

Parameters

position	*Pair<Integer,Integer> that represents a position in a board.*
myPieceType	*PieceType variable that represents the player in a cell.*

Definition at line 536 of file Board.java.

```
    {
        ArrayList< Pair<Integer,Integer> > result = new ArrayList<Pair<Integer, Integer»();

        Integer row = position.first;
        Integer column = position.second;
        PieceType opponentPiece = inversePlayer(myPieceType);

        if(row > 0) //To not go out of the boards boundaries
        {
            if(this.board[row-1][column] == opponentPiece) //to check if left side can be eaten
            {
                Integer it1 = row-1;
                Boolean found1 = false;

                while(it1 >= 0 && this.board[it1][column] != null && found1 == false)
                //go through the line of the board to see if we can place the piece we want in
    "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[it1][column] == myPieceType) //found a piece that's mine = CAN PLACE
                    {
                        result.add(new Pair<Integer,Integer>(it1,column));
                        found1 = true;
                    }
                    else //found another piece of the opponent = CONTINUE THE HUNT
                        it1 -= 1;
                }
            }
        }

        if(row < 7)
        {
            if(this.board[row+1][column] == opponentPiece) //to check if right side can be eaten
            {
                Integer it2 = row+1;
                Boolean found2 = false;

                while(it2 <= 7 && this.board[it2][column] != null && found2 == false)
                //go through the line of the board to see if we can place the piece we want in
    "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[it2][column] == myPieceType) //found a piece that's mine = CAN PLACE
                    {
                        result.add(new Pair<Integer,Integer>(it2,column));
                        found2 = true;
                    }
                    else //found another piece of the opponent = CONTINUE THE HUNT
                    {
                        it2 += 1;
                    }
                }
            }
        }

        return result;
    }
```

6.3.3.16 canPlaceDiagonal()

```
ArrayList< Pair<Integer,Integer> > domain.Board.canPlaceDiagonal (
            Pair< Integer, Integer > position,
            PieceType myPieceType )  [private]
```

Private method that returns an array of positions of the board in which you can conquer the pieces between them (diagonal search).

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

Returns an array of positions in which you can use with the method changePieces to conquer the pieces between them and the *position<e/m> parameter (which corresponds to the position we want to add a piece to).*

Parameters

position	*Pair< Integer,Integer > that represents a position in a board.*
myPieceType	*PieceType variable that represents the player in a cell.*

Definition at line 599 of file Board.java.

```
    {
        ArrayList< Pair<Integer,Integer> > result = new ArrayList<Pair<Integer, Integer»();

        Integer row = position.first;
        Integer column = position.second;
        PieceType opponentPiece = inversePlayer(myPieceType);

        //DIAGONAL UP LEFT
        if(row > 0 && column > 0) //To not go out of the boards boundaries
        {
            if(this.board[row-1][column-1] == opponentPiece) //to check if we can eat some opponent
    pieces in the upper left diagonal
            {
                Integer itRow = row-1;
                Integer itCol = column-1;
                Boolean found = false;

                while(itRow >= 0 && itCol >= 0 && this.board[itRow][itCol] != null && found == false)
                //go through the line of the board to see if we can place the piece we want in
    "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[itRow][itCol] == myPieceType) //found a piece that's mine in the
    diagonal line = CAN PLACE
                    {
                        result.add(new Pair<Integer,Integer>(itRow,itCol));
                        found = true;
                    }
                    else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
                    {
                        itRow -= 1;
                        itCol -= 1;
                    }
                }

            }
        }

        //DIAGONAL UP RIGHT
        if(row > 0 && column < 7)//To not go out of the boards boundaries
        {
            if(this.board[row-1][column+1] == opponentPiece) //to check if we can eat some opponents
    pieces in the upper right diagonal
            {
                Integer itRow = row-1;
                Integer itCol = column+1;
                Boolean found = false;

                while(itRow >= 0 && itCol <= 7 && this.board[itRow][itCol] != null && found == false)
                //go through the line of the board to see if we can place the piece we want in
    "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[itRow][itCol] == myPieceType)
                    {
                        result.add(new Pair<Integer,Integer>(itRow,itCol));
                        found = true;
                    }
                    else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
                    {
                        itRow -= 1;
                        itCol += 1;
                    }
                }

            }
        }

        //DIAGONAL DOWN LEFT
        if(row < 7 && column > 0) //To not go out of the boards boundaries
        {
            if(this.board[row+1][column-1] == opponentPiece) //to check if we can eat some opponents
    pieces in the bottom left diagonal
            {
                Integer itRow = row+1;
                Integer itCol = column-1;
                Boolean found = false;

                while(itRow <= 7 && itCol >= 0 && this.board[itRow][itCol] != null && found == false)
                //go through the line of the board to see if we can place the piece we want in "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[itRow][itCol] == myPieceType)
                    {
                        result.add(new Pair<Integer,Integer>(itRow,itCol));
                        found = true;
                    }
                    else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
                    {
                        itRow += 1;
                        itCol -= 1;
                    }
                }

            }
        }

        //DIAGONAL DOWN RIGHT
        if(row < 7 && column < 7) //To not go out of the boards boundaries
        {
            if(this.board[row+1][column+1] == opponentPiece) //to check if we can eat some opponent pieces in the bottom right diagonal
            {
                Integer itRow = row+1;
                Integer itCol = column+1;
                Boolean found = false;

                while(itRow <= 7 && itCol <= 7 && this.board[itRow][itCol] != null && found == false)
                //go through the line of the board to see if we can place the piece we want in "position". If so, we add the position of the piece that closes in the result array.
                {
                    if(this.board[itRow][itCol] == myPieceType)
                    {
                        result.add(new Pair<Integer,Integer>(itRow,itCol));
                        found = true;
                    }
                    else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
                    {
                        itRow += 1;
                        itCol += 1;
                    }
                }

            }
        }

        return result;

    }
```

6.3.3.17 changePieces()

```
void domain.Board.changePieces (
            Pair< Integer, Integer > addPiece,
            Pair< Integer, Integer > lastPiece,
            PieceType myPieceType ) [private]
```

Private method that changes the pieces between two positions of the board.

Precondition

Parameters aren't null.

Postcondition

The pieces between the two positions in the board are changed to the *myPieceType* state.

Parameters

addPiece	Pair<Intenger,Integer> that represents a position in the board.
lastPiece	Pair<Intenger,Integer> that represents a position in the board.
myPieceType	PieceType variable that represents the player in a cell.

Definition at line 727 of file Board.java.

```
    {
        Pair<Integer,Integer> position = new Pair<Integer,Integer>(addPiece.first,addPiece.second);
        Integer diffRow = lastPiece.first - addPiece.first;
        Integer diffCol = lastPiece.second - addPiece.second;
        Integer dirRow = 0, dirCol = 0;
        PieceType opponent = inversePlayer(myPieceType);

        if(diffRow == 0) //HORIZONTAL
        {
            if(diffCol > 0) //RIGHT
                dirCol = 1;
            else //LEFT
                dirCol = -1;
        }

        if(diffCol == 0) //VERTICAL
        {
            if(diffRow > 0) //UP
                dirRow = 1;
            else //DOWN
                dirRow = -1;
        }

        if(diffCol != 0 && diffRow != 0) //DIAGONAL
        {
            if(diffRow > 0 && diffCol > 0) //DIAGONAL BOTTOM RIGHT
            {
                dirRow = 1;
                dirCol = 1;
            }
            if(diffRow > 0 && diffCol < 0) //DIAGONAL BOTTOM LEFT
            {
                dirRow = 1;
                dirCol = -1;
            }
            if(diffRow < 0 && diffCol > 0) //DIAGONAL TOP RIGHT
            {
                dirRow = -1;
                dirCol = 1;
            }
            if(diffRow < 0 && diffCol < 0) //DIAGONAL TOP LEFT
            {
                dirRow = -1;
                dirCol = -1;
            }
        }


        while((position.first != lastPiece.first) || (position.second != lastPiece.second)) {
            if(this.board[position.first][position.second] == opponent && opponent == PieceType.PLAYER1)
            {
                this.piecesPlayer1--;
                this.piecesPlayer2++;
            }
            if(this.board[position.first][position.second] == opponent && opponent == PieceType.PLAYER2)
            {
                this.piecesPlayer1++;
                this.piecesPlayer2--;
            }

            this.board[position.first][position.second] = myPieceType;
            position.first = position.first + dirRow;
            position.second = position.second + dirCol;
        }
    }
```

6.3.4 Member Data Documentation

6.3.4.1 board

`PieceType [][] domain.Board.board [private]`

A matrix of 64 cells that composes an Othello board. Its the data structure that stores the different cells of the Board.

Definition at line 32 of file Board.java.

6.3.4.2 piecesPlayer1

`Integer domain.Board.piecesPlayer1 [private]`

PLAYER1's total number of pieces on the Board.

Definition at line 36 of file Board.java.

6.3.4.3 piecesPlayer2

`Integer domain.Board.piecesPlayer2 [private]`

PLAYER2's total number of pieces on the Board.

Definition at line 40 of file Board.java.

The documentation for this class was generated from the following file:

- Board.java

6.4 domain.BoardCtrl Class Reference

This class represents the controller of the Board class, which is the classs that will be used to communicate with the other controllers.

Public Member Functions

- BoardCtrl ()

  Default creator method.
- Board placePiece (Board board, Configuration configuration, PieceType myPieceType, Pair< Integer, Integer > position)

  Modifying method that adds a piece in the board parameter.
- Board placePieceConfig (Board board, Pair< Integer, Integer > position, PieceType myPieceType)

  Modifying method that adds a piece in the board parameters board attribute, which corresponds to an Initial Board of a Configuration.
- Board removePiece (Board board, Pair< Integer, Integer > position)

  Modifying method that removes a piece from the board parameter.
- Pair< Integer, Integer > getNumPieces (Board board)

  Get method that returns the value of the board parameter's PiecesPlayer1 and PiecesPlayer2 attributes.
- ArrayList< Pair< Integer, Integer > > validPositions (Board board, Configuration configuration, PieceType myPieceType)

  Method that returns an Array of the valid positions in board of the player myPieceType taking into consideration the Configuration of the Game.
- void isValid (Board board, Configuration configuration) throws InvalidBoardException

  Method that warns us if an instance of the board parameters is invalid.

6.4.1 Detailed Description

This class represents the controller of the Board class, which is the classs that will be used to communicate with the other controllers.

Done by Manuel Navid

Definition at line 20 of file BoardCtrl.java.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 BoardCtrl()

```
domain.BoardCtrl.BoardCtrl ( )
```

Default creator method.

Precondition

True

Postcondition

Creates an instance of BoardCtrl

Definition at line 26 of file BoardCtrl.java.

```
                         {
     }
```

6.4.3 Member Function Documentation

6.4.3.1 placePiece()

```
Board domain.BoardCtrl.placePiece (
              Board board,
              Configuration configuration,
              PieceType myPieceType,
              Pair< Integer, Integer > position )
```

Modifying method that adds a piece in the *board* parameter.

In addition, it applies the effect of adding that piece in the board by changing the pieces of the board taking into consideration the Configuration given.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

With the given Configuration, if the *position* parameter is correct then the returned board's attribute *board* will contain the board situation with that piece added. *Piecetype* in the *position* parameter and its effect considering the Configuration given (pieces changing from the different taking piece methods). If the position isn't correct, the returned board *board* will not be changed.

A correct position is a position in the board where given the *PieceType* parameter, we will take at least one opponent piece with the Configuration given.

Parameters

board	Instance of a Board class which is the one we will modify and return.
myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.
canEatHorizontally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Horizontal manner..
canEatVertically	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Vertical manner.
canEatDiagonally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Diagonal manner.

Definition at line 44 of file BoardCtrl.java.

```
                                          {
        board.placePiece(position, myPieceType, configuration.getCanEatHorizontally(),
                configuration.getCanEatVertically(), configuration.getCanEatDiagonally());

        return board;
    }
```

6.4.3.2 placePieceConfig()

```
Board domain.BoardCtrl.placePieceConfig (
            Board board,
            Pair< Integer, Integer > position,
            PieceType myPieceType )
```

Modifying method that adds a piece in the *board* parameters *board* attribute, which corresponds to an Initial Board of a Configuration.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

Returns a Board with it's *board* attribute equal to the *board* parameter with the addition of the piece *PieceType* in position *position*.

Parameters

board	Instance of a Board class which is the one we will modify and return.
myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.

Definition at line 60 of file BoardCtrl.java.

```
                                                                                                {
        board.placePieceConfig(position, myPieceType);

        return board;
    }
```

6.4.3.3 removePiece()

```
Board domain.BoardCtrl.removePiece (
            Board board,
            Pair< Integer, Integer > position )
```

Modifying method that removes a piece from the *board* parameter.

Precondition

The *position* parameter isn't null and has values between (0,0) and (7,7).

Postcondition

In the *board* parameter, the state of the cell in position *position* is converted to null, which means that now it's an empty cell on the board.

Parameters

position	Pair<Integer,Integer> that represents a position in a board.
board	Instance of a Board class which is the one we will modify and return.

Definition at line 73 of file BoardCtrl.java.

```
                                                                     {
        board.removePiece(position);

        return board;
    }
```

6.4.3.4 getNumPieces()

```
Pair<Integer, Integer> domain.BoardCtrl.getNumPieces (
            Board board )
```

Get method that returns the value of the *board* parameter's *PiecesPlayer1* and *PiecesPlayer2* attributes.

Precondition

True

Postcondition

The attributes *piecesPlayer1* and *PiecesPlayer2* of the *board* parameter are returned in the first and second space of a Pair, respectively.

Parameters

board	Instance of a Board class which is the one we will modify and return.

Definition at line 85 of file BoardCtrl.java.

```
                                                   {
        Pair<Integer, Integer> totalPieces = new Pair<Integer, Integer>(board.getPiecesPlayer1(),
                board.getPiecesPlayer2());

        return totalPieces;
    }
```

6.4.3.5 validPositions()

```
ArrayList<Pair<Integer, Integer> > domain.BoardCtrl.validPositions (
              Board board,
              Configuration configuration,
              PieceType myPieceType )
```

Method that returns an Array of the valid positions in *board* of the player *myPieceType* taking into consideration the Configuration of the Game.

Precondition

All parameters aren't null.

Postcondition

An Array of valid positions(Pair<Integer,Integer>) is returned.

A valid position in a board is one which it's cell state is equal to null (meaning an empty cell) and there is at least one opponent PieceType surrounding that position (go to surroundingPieces to crystalize what the surrounding areas of a position are).

Parameters

board	Instance of a Board class which is the one we will modify and return.
myPieceType	PieceType variable that represents the player in a cell.
configuration	Instance of a Configuration class used to determine which piece capturing methods we apply in this method.

Definition at line 102 of file BoardCtrl.java.

```
                                                   {
        ArrayList<Pair<Integer, Integer» validPos = new ArrayList<Pair<Integer, Integer»();

        validPos = board.validPositions(myPieceType, configuration.getCanEatHorizontally(),
                configuration.getCanEatVertically(), configuration.getCanEatDiagonally());

        return validPos;
    }
```

6.4.3.6 isValid()

```
void domain.BoardCtrl.isValid (
              Board board,
              Configuration configuration ) throws InvalidBoardException
```

Method that warns us if an instance of the *board* parameters is invalid.

An invalid Board means that no player can add a piece in the current state of the implicit parameter's *board* attribute.

Precondition

All parameters aren't null.

Postcondition

If the *board* parameter is invalid, InvalidBoardException will be thrown, else nothing.

Definition at line 118 of file BoardCtrl.java.

```
                                                                                                  {
            board.isValid(configuration.getCanEatHorizontally(), configuration.getCanEatVertically(),
                    configuration.getCanEatDiagonally());
        }
```

The documentation for this class was generated from the following file:

- BoardCtrl.java

6.5 test.driver.BoardDriver Class Reference

Public Member Functions

- BoardDriver ()
- void start ()
- void defaultBoard ()
- void createBoard ()
- void loadBoard ()
- void serializeBoard ()
- void getPiecesPlayers ()
- void getCurrentBoard ()
- void placePieceBoard ()
- void removePieceBoard ()
- void placePieceInitialBoard ()
- void validBoard ()
- void addPositions ()
- void deserialize ()
- void printCurrentBoard ()

Public Attributes

- Board currentBoard
- String nameCurrentBoard

Private Member Functions

- ArrayList< String > transcribeToCharacters ()
- Integer correctNumber (String rowOrColumn)
- PieceType correctPieceType ()
- void playAgain (String method)
- Boolean correctBoolean (String eatingMethod)

6.5.1 Detailed Description

Definition at line 13 of file BoardDriver.java.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BoardDriver()

```
test.driver.BoardDriver.BoardDriver ( )
```

PLAYER1 = B de blancas PLAYER2 = N de negras

Definition at line 23 of file BoardDriver.java.

```
{}
```

6.5.3 Member Function Documentation

6.5.3.1 start()

```
void test.driver.BoardDriver.start ( )
```

Definition at line 24 of file BoardDriver.java.

```
    {
        String title = new String();
        while(true)
        {
            title = (this.currentBoard != null ? String.format("\tcurrent Board %s selected\n",
    this.nameCurrentBoard): "\tNo current Board is selected\n");
            switch (Driver.menu(title, "Board Driver Menu",
                new Pair<String, String>("0", "Create a Default Board"),
                new Pair<String, String>("1", "Create a Board based on the board of the currentBoard"),
                new Pair<String, String>("2", "Load a Board from a file"),
                new Pair<String, String>("3", "Serialize the current Board to JSON"),
                new Pair<String, String>("4", "Get the number of pieces of both players of the current
    Board"),
                new Pair<String, String>("5", "Get the whole current Board"),
                new Pair<String, String>("6", "Place a piece on the current Board"),
                new Pair<String, String>("7", "Remove a piece of the current Board"),
                new Pair<String, String>("8", "Place a piece on the current Board as if it was an initial
    board of a configuration"),
                new Pair<String, String>("9", "Check if the current Board is valid to play"),
                new Pair<String, String>("10", "Check which positions you can add a piece to in the
    current Board"),
                new Pair<String, String>("11", "Deserialize the current Board"),
                new Pair<String, String>("12", "Print the current Board")))
                {
                case "0":
                defaultBoard();
                break;
                case "1":
                createBoard();
                break;
                case "2":
                loadBoard();
                break;
                case "3":
                serializeBoard();
                break;
                case "4":
                getPiecesPlayers();
                break;
                case "5":
                getCurrentBoard();
                break;
                case "6":
                placePieceBoard();
                break;
                case "7":
                removePieceBoard();
                break;
                case "8":
                placePieceInitialBoard();
                break;
                case "9":
                validBoard();
                break;
                case "10":
                addPositions();
                break;
                case "11":
                deserialize();
                break;
                case "12":
                {
                    if (this.currentBoard != null)
                    {
                        System.out.println(String.format("==== Printing Board %s ====\n",
    this.nameCurrentBoard));
                        System.out.println(String.format("Board %s printed below!\n",
    this.nameCurrentBoard));
                    }
                    printCurrentBoard();
                }
                break;
            }
            Driver.pause();
        }
    }
```

6.5.3.2 defaultBoard()

```
void test.driver.BoardDriver.defaultBoard ( )
```

Definition at line 98 of file BoardDriver.java.

```
{
    Driver.clear();
    System.out.println("==== Creating a Default Board ====\n");
    this.currentBoard = new Board();
    this.nameCurrentBoard = "Default";

    System.out.println("Default Board was created succesfully, printed below:");
    System.out.println("\n");
    printCurrentBoard();
    System.out.println("\n");
    System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
    System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2());
    System.out.println("\n");
}
```

6.5.3.3 createBoard()

void test.driver.BoardDriver.createBoard ( )

Definition at line 114 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE SERIALIZING IT!\n\nGo back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            PieceType[][] board = this.currentBoard.getBoard();
            System.out.println("Getting the currentBoard board attribute and creating a new Board instance based off of it.\n");
            this.currentBoard = new Board(board);
            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
            System.out.println("Created the Board class instance successfully\n");
        }
    }
```

6.5.3.4 loadBoard()

void test.driver.BoardDriver.loadBoard ( )

Definition at line 134 of file BoardDriver.java.

```
    {
            Driver.clear();
            FixtureRepository fixtureRepo =  new FixtureRepository();
            System.out.println("==== What Board would you like to load? ====" + "\n");
            ArrayList<String> listBoards = fixtureRepo.listFiles();

            for(Integer i = 0; i < listBoards.size(); ++i)
            {
                System.out.println(String.format("[%d]   ", i) + listBoards.get(i));
            }

            System.out.println("\n");

            Boolean checkValid = false;
            Integer nameBoard = null;

            while(checkValid == false)
            {
                try {
                    nameBoard = Integer.parseInt(Driver.input("What Board would you like to load?"));
                    if(nameBoard < 0 || nameBoard >= listBoards.size() )
                        System.out.println("Incorrect Index! Try again :D");
                    else checkValid = true;
                } catch (Exception e) {
                    System.out.println("You didn't add an Integer! Try again :D");
                }
            }

            Driver.clear();
            String path = listBoards.get(nameBoard);
            JSONObject newBoard = fixtureRepo.boardFileToJSON(path);

            this.nameCurrentBoard = path;
            this.currentBoard = new Board(newBoard);

            System.out.println(String.format("%s was loaded succesfully. \n\nIt's printed below!\n", path));
            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
    }
```

6.5.3.5 serializeBoard()

```
void test.driver.BoardDriver.serializeBoard ( )
```

Definition at line 177 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE SERIALIZING IT!\n\nGo back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Serializing Board %s ====\n", this.nameCurrentBoard));
            printCurrentBoard();
            System.out.println("\n" + this.nameCurrentBoard + " Board was serialized correctly into JSON format.\n");
            System.out.println(this.currentBoard.serialize().toString(2) + "\n");
        }
    }
```

6.5.3.6 getPiecesPlayers()

```
void test.driver.BoardDriver.getPiecesPlayers ( )
```

Definition at line 193 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE GETTING ITS ATTRIBUTES!\n\nGo back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Getting the number of pieces of both players of the current Board %s ====\n", this.nameCurrentBoard));
            printCurrentBoard();
            System.out.println("\nPieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
        }
    }
```

6.5.3.7 getCurrentBoard()

```
void test.driver.BoardDriver.getCurrentBoard ( )
```

Definition at line 209 of file BoardDriver.java.

```
    {
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE PRINTING IT!\n\nGo back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Getting the current Board %s ====\n", this.nameCurrentBoard));
            PieceType[][] gotBoard = currentBoard.getBoard();
            System.out.println("Get Board was executed correctly and it’s saved in a PieceType[][] called gotBoard. gotBoard is printed below:\n");
            currentBoard = new Board(gotBoard);
            printCurrentBoard();
        }
    }
```

6.5.3.8 placePieceBoard()

```
void test.driver.BoardDriver.placePieceBoard ( )
```

Definition at line 225 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE PLACING A PIECE!\n\nGo back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Placing a piece on %s Board ====\n", this.nameCurrentBoard));

            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Write what position you would like to add a piece in the board\n");

            Integer row = correctNumber("row");
            Integer col = correctNumber("column");
            Pair<Integer, Integer> pos = new Pair<Integer, Integer>(row,col);
            PieceType myPieceType = correctPieceType();
            Boolean horizontal = correctBoolean("Horizontal"), vertical = correctBoolean("Vertical"), diagonal = correctBoolean("Diagonal");

            Integer sumPieces = this.currentBoard.getPiecesPlayer1() + this.currentBoard.getPiecesPlayer2();
            this.currentBoard.placePiece(pos, myPieceType, horizontal, vertical, diagonal);

            if(sumPieces == (this.currentBoard.getPiecesPlayer1() + this.currentBoard.getPiecesPlayer2()))
                System.out.println("\nNo pieces were added, as the parameters you inserted didn't give a valid position to place a piece.");
            else
                System.out.println("\nSuccesfully added a piece in position: " + pos);

            System.out.println("\n");
            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");


            playAgain("placePieceBoard");
        }
    }
```

6.5.3.9 removePieceBoard()

```
void test.driver.BoardDriver.removePieceBoard ( )
```

Definition at line 264 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE REMOVING A PIECE!\n\nGo back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Removing a piece on %s Board ====\n", this.nameCurrentBoard));

            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Write what position you would like to remove a piece from\n");
            Integer row = correctNumber("row");
            Integer col = correctNumber("column");
            Pair<Integer,Integer> pos = new Pair<Integer,Integer>(row,col);

            this.currentBoard.removePiece(pos);

            System.out.println("\n");
            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");

            playAgain("removePieceBoard");
        }
    }
```

6.5.3.10 placePieceInitialBoard()

```
void test.driver.BoardDriver.placePieceInitialBoard ( )
```

Definition at line 294 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE PLACING A PIECE!\n\nGo back to the
       main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Placing a piece on %s INITIAL Board ====\n",
       this.nameCurrentBoard));

            printCurrentBoard();
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
            System.out.println("Write what position you would like to add a piece to the board\n");

            Integer row = correctNumber("row");
            Integer col = correctNumber("column");
            Pair<Integer, Integer> pos = new Pair<Integer, Integer>(row,col);

            PieceType myPieceType = correctPieceType();

            this.currentBoard.placePieceConfig(pos, myPieceType);
            System.out.println("\n");
            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");

            playAgain("placePieceInitialBoard");
        }
    }
```

6.5.3.11 validBoard()

```
void test.driver.BoardDriver.validBoard ( )
```

Definition at line 327 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE CHECKING IF ITS VALID!\n\nGo back to
       the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Checking if %s Board is valid ====\n",
       this.nameCurrentBoard));
            printCurrentBoard();
            System.out.println("\n");

            try {
                       Boolean horizontal = correctBoolean("Horizontal"), vertical =
       correctBoolean("Vertical"), diagonal = correctBoolean("Diagonal");

                       this.currentBoard.isValid(horizontal, vertical, diagonal);

                       System.out.println("\nThe board is valid with the configuration you've inserted!\n");
                       ArrayList<Pair<Integer,Integer» validPosPlayer1 =
       this.currentBoard.validPositions(PieceType.PLAYER1, horizontal, vertical, diagonal);
                       ArrayList<Pair<Integer,Integer» validPosPlayer2 =
       this.currentBoard.validPositions(PieceType.PLAYER1, horizontal, vertical, diagonal);

                       if(!validPosPlayer1.isEmpty()) System.out.println("For example, PLAYER1(B) can add a
       piece in " + validPosPlayer1.get(0) + "\n");
                       else if(!validPosPlayer2.isEmpty()) System.out.println("For example, PLAYER2(B) can add
       a piece in " + validPosPlayer2.get(0) + "\n");

            } catch (Exception e) {
                       System.out.println("\nThe board is invalid with the configuration you've inserted,
       meaning neither of both players can add a piece.\n");
            }
        }
    }
```

6.5.3.12 addPositions()

`void test.driver.BoardDriver.addPositions ( )`

Definition at line 358 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE SEEING THE VALID POSITIONS!\n\nGo
back to the main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Getting all available positions of a PLAYER in %s
Board ====\n", this.nameCurrentBoard));
            printCurrentBoard();
            System.out.println("\n");

            Boolean checkValues = false;
            Boolean both = false;
            PieceType myPieceType = null;

            while(!checkValues)
            {
                try {
                    String typePiece = new String(Driver.input("Write which color piece you want to see
its valid positions (B or N or BN)"));
                    if(typePiece.equals("B"))
                    {
                        myPieceType = PieceType.PLAYER1;
                        checkValues = true;
                    }
                    else if (typePiece.equals("N"))
                    {
                        myPieceType = PieceType.PLAYER2;
                        checkValues = true;
                    }
                    else if (typePiece.equals("BN"))
                    {
                        myPieceType = PieceType.PLAYER2;
                        checkValues = true;
                        both = true;
                    }
                    else {
                        System.out.println("\nERROR! Piece is neither B nor N nor BN\n");
                    }
                } catch (Exception e) {
                    System.out.println("\nERROR! Piece is neither B nor N nor BN\n");
                }
            }

            Boolean horizontal = correctBoolean("Horizontal"), vertical = correctBoolean("Vertical"),
diagonal = correctBoolean("Diagonal");

```

```
            ArrayList<Pair<Integer,Integer» validPosPlayer1 =
this.currentBoard.validPositions(PieceType.PLAYER1, horizontal, vertical, diagonal);
            ArrayList<Pair<Integer,Integer» validPosPlayer2 =
this.currentBoard.validPositions(PieceType.PLAYER2, horizontal, vertical, diagonal);

            if(both || myPieceType == PieceType.PLAYER1)
            {
                for(int i = 0; i < validPosPlayer1.size(); ++i)
                {
                    if(i == 0) System.out.println("\nValid positions for PLAYER1(B):");
                    System.out.println("Position: " + validPosPlayer1.get(i));
                }
            }

            System.out.println("\n");

            if(both || myPieceType == PieceType.PLAYER2)
            {
                for(int i = 0; i < validPosPlayer2.size(); ++i)
                {
                    if(i == 0) System.out.println("Valid positions for PLAYER2(N):");
                    System.out.println("Position: " + validPosPlayer2.get(i));
                }
            }
            System.out.println("\n");
        }
    }
```

6.5.3.13 deserialize()

`void test.driver.BoardDriver.deserialize ( )`

Definition at line 431 of file BoardDriver.java.

```
    {
        Driver.clear();
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE DESERIALIZING IT!\n\nGo back to the
main menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {
            System.out.println(String.format("==== Deserializing Board %s ====\n",
this.nameCurrentBoard));

            System.out.println(this.currentBoard.serialize().toString(2) + "\n");
            this.currentBoard = new Board(this.currentBoard.serialize());

            System.out.println("The currentBoard has been deserialized from the JSON format
successfully, as we can see below:\n");
            printCurrentBoard();
            System.out.println("\n");
            System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
            System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
      }
    }
```

6.5.3.14 printCurrentBoard()

`void test.driver.BoardDriver.printCurrentBoard ( )`

Definition at line 452 of file BoardDriver.java.

```
    {
        if(this.currentBoard == null)
        {
            System.out.println("YOU MUST INITIALIZE A BOARD BEFORE PRINTING IT!\n\nGo back to the main
menu and create a Default Board or Load a preexisting one\n");
        }
        else
        {

            ArrayList<String> boardCodified = transcribeToCharacters();
            System.out.println("       0  1  2  3  4  5  6  7");
            System.out.println("    -------------------------");

        for(Integer i = 0; i < 8; ++i)
        {
            String row = boardCodified.get(i);
            System.out.println("  " + i + " |  " + row.charAt(0) + "  " + row.charAt(1) + "  " + row.charAt(2) + "  " + row.charAt(3) +
                               "  " + row.charAt(4) + "  " + row.charAt(5) + "  " + row.charAt(6) + "  " + row.charAt(7) + "  ");
        }

            System.out.println("\n");
        }
    }
```

6.5.3.15 transcribeToCharacters()

`ArrayList<String> test.driver.BoardDriver.transcribeToCharacters ( ) [private]`

Definition at line 478 of file BoardDriver.java.

```
    {
        ArrayList<String> boardCodified = new ArrayList<String>(8);
        String operational = "";
        PieceType[][] current = this.currentBoard.getBoard();

        if (current != null)
        {
            for(int i = 0; i < 8; ++i)
            {
                operational = "";
                for(int j = 0; j < 8; ++j)
                {
                    if(current[i][j] == PieceType.PLAYER1) operational = operational + "B";
                    if(current[i][j] == PieceType.PLAYER2) operational = operational + "N";
                    if(current[i][j] == null) operational = operational + "?";

                }
                boardCodified.add(operational);
            }
        }

        return boardCodified;
    }
```

6.5.3.16 correctNumber()

```
Integer test.driver.BoardDriver.correctNumber (
               String rowOrColumn ) [private]
```

Definition at line 504 of file BoardDriver.java.

```
    {
        Boolean checkValues = false;
        Integer returnNumber = -1;

        while(!checkValues)
        {
                try {
                    Integer input = Integer.parseInt(Driver.input(String.format("Write the %s number", rowOrColumn)));
                    if(input < 0 || input > 7) System.out.println(String.format("\nERROR! %s is not between 0 and 7\n", rowOrColumn));
                    else
                    {
                        checkValues = true;
                        returnNumber = input;
                    }
                } catch (Exception e) {
                    System.out.println(String.format("\nERROR! %s is not between 0 and 7\n", rowOrColumn));
                }
        }
        return returnNumber;
    }
```

6.5.3.17 correctPieceType()

PieceType test.driver.BoardDriver.correctPieceType ( ) [private]

Definition at line 526 of file BoardDriver.java.

```
    {
        Boolean checkValues = false;
        PieceType myPieceType = null;

        while(!checkValues)
        {
            try {
                char typePiece = new String(Driver.input("Write which color plays (B or N)")).charAt(0);
                if(typePiece == 'B')
                {
                    myPieceType = PieceType.PLAYER1;
                    checkValues = true;
                }
                else if (typePiece == 'N')
                {
                    myPieceType = PieceType.PLAYER2;
                    checkValues = true;
                }
                else {
                    System.out.println("\nERROR! Piece is not B or N\n");
                }
            } catch (Exception e) {
                System.out.println("\nERROR! Piece is not B or N\n");
            }
        }
        return myPieceType;
    }
```

6.5.3.18 playAgain()

void test.driver.BoardDriver.playAgain (
  String *method*) [private]

Definition at line 555 of file BoardDriver.java.

```
    {
        Boolean checkValues = false;
        String methodDescription = null;

        if(method.equals("placePieceInitialBoard") || method.equals("placePieceBoard"))
            methodDescription = "add";
        else if(method.equals("removePieceBoard"))
            methodDescription = "remove";

        while(!checkValues)
        {
            try {
                char cont = new String(Driver.input(String.format("Do you want to %s another piece? Write y or n", methodDescription))).charAt(0);
                if(cont == 'y')
                {
                    checkValues = true;
                    if(method.equals("placePieceInitialBoard"))
                        placePieceInitialBoard();
                    else if(method.equals("placePieceBoard"))
                        placePieceBoard();
                    else if(method.equals("removePieceBoard"))
                        removePieceBoard();
                }
                else if(cont == 'n')
                {
                    checkValues = true;
                    return;
                }
                else System.out.println("\nERROR! You didn't write y or n. Try again! :D\n");
            } catch (Exception e) {
                System.out.println("You didn't write y or n! Try again :D");
            }
        }
    }
```

6.5.3.19 correctBoolean()

```
Boolean test.driver.BoardDriver.correctBoolean (
              String eatingMethod )  [private]
```

Definition at line 591 of file BoardDriver.java.

```
    {
        Boolean checkValues = false;
        Boolean eating = null;

        while(!checkValues)
        {
            try {
                char input = new String(Driver.input(String.format("Can we eat in %s? Write y or n",
       eatingMethod))).charAt(0);
                if(input == 'y')
                {
                    eating = true;
                    checkValues = true;
                }
                else if(input == 'n')
                {
                    eating = false;
                    checkValues = true;
                }
                else System.out.println(String.format("\nERROR! %s eating was neither affirmative nor
       negative\n", eatingMethod));
            } catch (Exception e) {
                System.out.println(String.format("\nERROR! %s eating was neither affirmative nor
       negative\n", eatingMethod));
            }
        }
        return eating;
    }
```

6.5.4 Member Data Documentation

6.5.4.1 currentBoard

```
Board test.driver.BoardDriver.currentBoard
```

Definition at line 14 of file BoardDriver.java.

6.5.4.2 nameCurrentBoard

```
String test.driver.BoardDriver.nameCurrentBoard
```

Definition at line 15 of file BoardDriver.java.

The documentation for this class was generated from the following file:

- BoardDriver.java

6.6 domain.Bot Class Reference

Represents a bot in our system.

Public Member Functions

- Bot (String name, int difficulty, UUID id, UUID creatorID)
  Creator that, given a name 'name', a difficulty 'difficulty', an ID id and an ID creatorID, returns a Bot.
- Bot (JSONObject bot)
  Creator that, given a JSONObject bot, deserializes it.
- JSONObject serialize ()
  Creator that serializes the current object to a JSON Object.
- int getDifficulty ()
  Consultant that returns the implicit parameter's difficulty.
- UUID getCreatorID ()
  Consultant that returns the implicit parameter's creatorID.
- void setDifficulty (int difficulty) throws InvalidDifficultyException
  Modifier that, given a difficulty, changes the implicit parameter's difficulty for a new difficulty 'difficulty'.

Private Attributes

- int difficulty
  bot's difficulty
- UUID creatorID
  bot's creator ID

Additional Inherited Members

6.6.1 Detailed Description

Represents a bot in our system.

Done by Arnau Pujantell

Subclass that represents a bot. It contains a difficulty and a creatorID.

Definition at line 20 of file Bot.java.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Bot() [1/2]

```
domain.Bot.Bot (
            String name,
            int difficulty,
            UUID id,
            UUID creatorID )
```

Creator that, given a name 'name', a difficulty 'difficulty', an ID id and an ID creatorID, returns a Bot.

CREATORS

Precondition

None of the elements is null

Postcondition

It creates a new bot with name 'name', difficulty 'difficulty', id 'id', type 'BOT', isDeleted as 'false' and creatorID creatorID.

Definition at line 34 of file Bot.java.

```
    {
        this.name = name;
        this.difficulty = difficulty;
        this.id = id;
        this.isDeleted = false;
        this.creatorID = creatorID;
    }
```

6.6.2.2 Bot() [2/2]

```
domain.Bot.Bot (
            JSONObject bot )
```

Creator that, given a JSONObject bot, deserializes it.

Precondition

bot is not null

Postcondition

bot is not a JSONObject anymore

Definition at line 47 of file Bot.java.

```
                            {
        this.name = bot.getString("name");
        this.id = UUID.fromString(bot.getString("id"));
        this.difficulty = bot.getInt("difficulty");
        this.isDeleted = bot.getBoolean("is_deleted");
        this.creatorID = UUID.fromString(bot.getString("creator_id"));
    }
```

6.6.3 Member Function Documentation

6.6.3.1 serialize()

`JSONObject domain.Bot.serialize ( )`

Creator that serializes the current object to a JSON Object.

Precondition

True

Postcondition

The current bot becomes a JSON Object

Definition at line 59 of file Bot.java.

```
                               {
        JSONObject bot = new JSONObject();
        bot.put("name", this.name);
        bot.put("id", this.id.toString());
        bot.put("difficulty", this.difficulty);
        bot.put("type", "BOT");
        bot.put("is_deleted", this.isDeleted);
        bot.put("creator_id", this.creatorID.toString());

        return bot;
    }
```

6.6.3.2 getDifficulty()

`int domain.Bot.getDifficulty ( )`

Consultant that returns the implicit parameter's difficulty.

CONSULTANTS

Precondition

True

Postcondition

The implicit parameter's difficulty is returned.

Returns

Definition at line 79 of file Bot.java.

```
                            {
        return this.difficulty;
    }
```

6.6.3.3 getCreatorID()

```
UUID domain.Bot.getCreatorID ( )
```

Consultant that returns the implicit parameter's creatorID.

Precondition

True

Postcondition

The implicit parameter's creatorID is returned.

Returns

Definition at line 88 of file Bot.java.

```
                                {
        return this.creatorID;
    }
```

6.6.3.4 setDifficulty()

```
void domain.Bot.setDifficulty (
             int difficulty ) throws InvalidDifficultyException
```

Modifier that, given a difficulty, changes the implicit parameter's difficulty for a new difficulty 'difficulty'.

MODIFIERS

Precondition

difficulty is not null

Postcondition

The implicit parameter's difficulty has changed.

Definition at line 98 of file Bot.java.

```
                                                                            {
        if(difficulty < 1 || difficulty > 10) {
            throw new InvalidDifficultyException();
        }
        this.difficulty = difficulty;
    }
```

6.6.4 Member Data Documentation

6.6.4.1 difficulty

```
int domain.Bot.difficulty [private]
```

bot's difficulty

Definition at line 22 of file Bot.java.

6.6.4.2 creatorID

```
UUID domain.Bot.creatorID [private]
```

bot's creator ID

Definition at line 24 of file Bot.java.

The documentation for this class was generated from the following file:

- Bot.java

6.7 cmd.driver.bot Class Reference

Bot driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)
  Bot driver main function. Creates an instance of the Bot driver and starts it.

6.7.1 Detailed Description

Bot driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file bot.java.

6.7.2 Member Function Documentation

6.7.2.1 main()

```
static void cmd.driver.bot.main (
             String[] args )  [static]
```

Bot driver main function. Creates an instance of the Bot driver and starts it.

Precondition

True.

Postcondition

The Bot driver has started.

Definition at line 22 of file bot.java.

```
                                            {
           new BotDriver().start();
       }
```

The documentation for this class was generated from the following file:

- bot.java

6.8 test.driver.BotDriver Class Reference

Public Member Functions

- BotDriver ()
- void start ()

Public Attributes

- Bot currentBot

Private Member Functions

- void mainMenu ()
- void createBot ()
- void serialize ()
- void deserialize ()
- void getName ()
- void getDifficulty ()
- void getIsDeleted ()
- void getType ()
- void getID ()
- void getCreatorID ()
- void setName ()
- void setDifficulty ()
- void setIsDeleted ()

Additional Inherited Members

6.8.1 Detailed Description

Definition at line 13 of file BotDriver.java.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 BotDriver()

```
test.driver.BotDriver.BotDriver ( )
```

Definition at line 17 of file BotDriver.java.

```
                           {
        this.currentBot = null;
    }
```

6.8.3 Member Function Documentation

6.8.3.1 start()

```
void test.driver.BotDriver.start ( )
```

Definition at line 21 of file BotDriver.java.

```
                            {
        while(true) {
            this.mainMenu();
        }
    }
```

6.8.3.2 mainMenu()

`void test.driver.BotDriver.mainMenu ( ) [private]`

Definition at line 27 of file BotDriver.java.

```java
                            {
        String title = (this.currentBot != null ? String.format("Current: %s\n", this.currentBot.getName()) : null);
        switch (Driver.menu(title, "Bot Driver",
                new Pair<String, String>("1", "Create Bot"),
                new Pair<String, String>("2", "Get name"),
                new Pair<String, String>("3", "Set name"),
                new Pair<String, String>("4", "Get difficulty"),
                new Pair<String, String>("5", "Set difficulty"),
                new Pair<String, String>("6", "Get state"),
                new Pair<String, String>("7", "Set state"),
                new Pair<String, String>("8", "Get type"),
                new Pair<String, String>("9", "Get ID"),
                new Pair<String, String>("10", "Get creatorID"),
                new Pair<String, String>("11", "Serialize Bot to JSON"),
                new Pair<String, String>("12", "Deserialize Bot from JSON"))) {
            case "1":
                this.createBot();
                break;
            case "2":
                this.getName();
                break;
            case "3":
                this.setName();
                break;
            case "4":
                this.getDifficulty();
                break;
            case "5":
                this.setDifficulty();
                break;
            case "6":
                this.getIsDeleted();
                break;
            case "7":
                this.setIsDeleted();
                break;
            case "8":
                this.getType();
                break;
            case "9":
                this.getID();
                break;
            case "10":
                this.getCreatorID();
                break;
            case "11":
                this.serialize();
                break;
            case "12":
                this.deserialize();
                break;

        }
        Driver.pause();
    }
```

6.8.3.3 createBot()

`void test.driver.BotDriver.createBot ( ) [private]`

Definition at line 83 of file BotDriver.java.

```java
                             {
        System.out.println("Take into account that UUIDs will be randomly generated because typing them in will be a hassle.\n");
        String name = Driver.input("Name?");
        Integer difficulty = Driver.inputInt("Difficulty? (From 1 to 10)");
        try {
            Bot bot = new Bot("Default name", 0, UUID.randomUUID(), UUID.randomUUID());
            bot.setName(name);
            bot.setDifficulty(difficulty);
            this.currentBot = bot;
            System.out.println(String.format("%s created successfully!", this.currentBot.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
      e.getMessage()));
        }
    }
```

6.8.3.4 serialize()

`void test.driver.BotDriver.serialize ( )  [private]`

Definition at line 98 of file BotDriver.java.

```
                                 {
        if(this.currentBot == null) {
            System.out.println("No current Bot");
            return;
        }
        System.out.println(String.format("%s's serialized to JSON is: %s", this.currentBot.getName(),
                this.currentBot.serialize().toString(2)));
    }
```

6.8.3.5 deserialize()

`void test.driver.BotDriver.deserialize ( )  [private]`

Definition at line 108 of file BotDriver.java.

```
                                   {
        if(this.currentBot == null) {
            System.out.println("No current Bot");
            return;
        }
        System.out.println(this.currentBot.serialize().toString(2));
        this.currentBot = new Bot(this.currentBot.serialize());
        System.out.println(String.format("\n%s's deserialized from the above JSON successfully!\n",
                this.currentBot.getName()));
        System.out.println(String.format("name:\t\t\t%s", this.currentBot.getName()));
        System.out.println(String.format("id:\t\t\t%s", this.currentBot.getID()));
        System.out.println(String.format("difficulty:\t\t%s", this.currentBot.getDifficulty()));
        System.out.println(String.format("is_deleted:\t\t%s", this.currentBot.getIsDeleted()));
        System.out.println(String.format("creator_id:\t\t%s", this.currentBot.getCreatorID()));

    }
```

6.8.3.6 getName()

`void test.driver.BotDriver.getName ( )  [private]`

Definition at line 125 of file BotDriver.java.

```
                               {
        if(this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        System.out.println(String.format("%s's bot name is: %s", this.currentBot.getName(),
      this.currentBot.getName()));
    }
```

6.8.3.7 getDifficulty()

```
void test.driver.BotDriver.getDifficulty ( )  [private]
```

Definition at line 133 of file BotDriver.java.

```
                                {
        if(this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        System.out.println(String.format("%s's difficulty is: %s", this.currentBot.getName(),
    this.currentBot.getDifficulty()));
    }
```

6.8.3.8 getIsDeleted()

```
void test.driver.BotDriver.getIsDeleted ( )  [private]
```

Definition at line 141 of file BotDriver.java.

```
                               {
        if(this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        System.out.print(String.format("%s's state is: ", this.currentBot.getName()));
        if(this.currentBot.getIsDeleted())
            System.out.println("DELETED");
        else
            System.out.println("ACTIVE");
    }
```

6.8.3.9 getType()

```
void test.driver.BotDriver.getType ( )  [private]
```

Definition at line 153 of file BotDriver.java.

```
                          {
        System.out.println("Bot's type attribute was removed because of professor's feedback. However,
    this option is still here to have backwards compatibility with old delivered documents.");
    }
```

6.8.3.10 getID()

```
void test.driver.BotDriver.getID ( )  [private]
```

Definition at line 157 of file BotDriver.java.

```
                        {
        if(this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        System.out.println(String.format("%s's ID is: %s", this.currentBot.getName(),
    this.currentBot.getID()));
    }
```

6.8.3.11 getCreatorID()

```
void test.driver.BotDriver.getCreatorID ( )  [private]
```

Definition at line 165 of file BotDriver.java.

```
                                     {
        if(this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        System.out.println(String.format("%s's CreatorID is: %s", this.currentBot.getName(),
this.currentBot.getCreatorID()));

    }
```

6.8.3.12 setName()

```
void test.driver.BotDriver.setName ( )  [private]
```

Definition at line 174 of file BotDriver.java.

```
                                {
        if (this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        try {
            this.currentBot.setName(Driver.input("New name?"));
            System.out.println(String.format("%s name changed successfully!",
this.currentBot.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
        }
    }
```

6.8.3.13 setDifficulty()

```
void test.driver.BotDriver.setDifficulty ( )  [private]
```

Definition at line 187 of file BotDriver.java.

```
                                      {
        if (this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        try {
            this.currentBot.setDifficulty(Driver.inputInt("Choose a difficulty level from 1 to 10"));
            System.out.println(String.format("%s's difficulty has been changed successfully!",
this.currentBot.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
            setDifficulty();
        }
    }
```

6.8.3.14 setIsDeleted()

```
void test.driver.BotDriver.setIsDeleted ( )  [private]
```

Definition at line 201 of file BotDriver.java.

```
                                {
        if(this.currentBot == null) {
            System.out.println("No current bot!");
            return;
        }
        if(Driver.inputBool("Do you want to delete the current bot?")) {
            this.currentBot.setIsDeleted(true);
            System.out.println(String.format("%s's state has changed to DELETED!",
      this.currentBot.getName()));
        }
        else {
            System.out.println(String.format("%s's state has not changed!", this.currentBot.getName()));
        }
    }
```

6.8.4 Member Data Documentation

6.8.4.1 currentBot

```
Bot test.driver.BotDriver.currentBot
```

Definition at line 15 of file BotDriver.java.

The documentation for this class was generated from the following file:

- BotDriver.java

6.9 view.BotsConsultView Class Reference

Public Member Functions

- BotsConsultView ()

    Class creator.
- void initialize ()

    Initialize method which is executed when the scene is shown.
- void user () throws IOException

    Event method which is executed when the User tab is clicked.
- void onChangeBotChooser () throws IOException

    Event method which is executed when the Bot Chooser is clicked.
- void config () throws IOException

    Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

    Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

    Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

Event method which is executed when the Play tab is clicked.
- void createBot () throws IOException

  Event method which is executed when the createBot button is clicked.
- void modifyBot () throws IOException

  Event method which is executed when the modifyBot button is clicked.
- void consultBot () throws IOException

  Event method which is executed when the consultBot button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createBot

  Bot create button text.
- Rectangle createBotButton

  Bot create button.
- Text modifyBot

  Bot modify button text.
- Rectangle modifyBotButton

  Bot modify button.
- Text consultBot

  Bot consult button text.
- Rectangle consultBotButton

  Bot consult button.
- ChoiceBox botChooser

  Bot choiceBox.
- Label name

  Bot name label.
- Label difficultyNumber

  Bot difficulty label.
- Label consultConfigResult

  Creator name label.
- Label consultBotResult

  Exception output message label.
- Label currentUserName

  Current user name.
- Label creator

  Bot creator.
- Text logOut

  LogOut button.
- Map$<$ String, UUID $>$ botMap

  Map of bots.

6.9.1 Detailed Description

This class represents the scene controller of the consult function of a bot.

Done by Arnau Pujantell

Definition at line 30 of file BotsConsultView.java.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 BotsConsultView()

```
view.BotsConsultView.BotsConsultView ( )
```

Class creator.

Definition at line 36 of file BotsConsultView.java.

```
                              {
    }
```

6.9.3 Member Function Documentation

6.9.3.1 initialize()

```
void view.BotsConsultView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. All bot names are inserted in the Bot choiceBox and the bot map is setted.

Definition at line 153 of file BotsConsultView.java.

```
                               {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        botMap = new HashMap<String, UUID>();
        ArrayList<Pair<String, UUID» botList = ViewCtrl.domainCtrl.listBots();
        for(Pair<String, UUID> bot : botList) {
            botChooser.getItems().add(bot.first);
            botMap.put(bot.first, bot.second);
        }
    }
```

6.9.3.2 user()

```
void view.BotsConsultView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 168 of file BotsConsultView.java.

```
                                    {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.9.3.3 onChangeBotChooser()

```
void view.BotsConsultView.onChangeBotChooser ( ) throws IOException
```

Event method which is executed when the Bot Chooser is clicked.

Precondition

True

Postcondition

The Bot information is shown.

Definition at line 177 of file BotsConsultView.java.

```
                                                  {
        String chosenBot = (String) botChooser.getValue();
        if (chosenBot != null) {
            Pair<JSONObject, String> result = ViewCtrl.domainCtrl.getBot(botMap.get(chosenBot));
            if (result.second != null) {
                switch (result.second) {
                    case "ERR_INEXISTING_PLAYER":
                        consultBotResult.setText("The bot doesn't exist!");
                        break;
                    default:
                        consultBotResult.setText("Something went wrong, try again!");
                        break;
                }
            }
            else {
                consultBotResult.setText("");
                name.setText(result.first.getString("name"));
                difficultyNumber.setText(Integer.toString(result.first.getInt("difficulty")));
                Pair<JSONObject, String> user =
    ViewCtrl.domainCtrl.getUser(UUID.fromString(result.first.getString("creator_id")));
                creator.setText((user.first != null ? user.first.getString("name") : "Unknown"));
            }
        }
    }
```

6.9.3.4 config()

`void view.BotsConsultView.config ( ) throws IOException`

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 206 of file BotsConsultView.java.

```
                                    {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.9.3.5 games()

`void view.BotsConsultView.games ( ) throws IOException`

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 215 of file BotsConsultView.java.

```
                                   {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.9.3.6 ranking()

`void view.BotsConsultView.ranking ( ) throws IOException`

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 224 of file BotsConsultView.java.

```
                                     {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.9.3.7 play()

```
void view.BotsConsultView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 233 of file BotsConsultView.java.

```
                                    {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.9.3.8 createBot()

```
void view.BotsConsultView.createBot ( ) throws IOException
```

Event method which is executed when the createBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsCreateView.

Definition at line 242 of file BotsConsultView.java.

```
                                         {
        ViewCtrl.changeScene("template/BotsCreateView.fxml");
    }
```

6.9.3.9 modifyBot()

```
void view.BotsConsultView.modifyBot ( ) throws IOException
```

Event method which is executed when the modifyBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsModifyView.

Definition at line 251 of file BotsConsultView.java.

```
                                         {
        ViewCtrl.changeScene("template/BotsModifyView.fxml");
    }
```

6.9.3.10 consultBot()

```
void view.BotsConsultView.consultBot ( ) throws IOException
```

Event method which is executed when the consultBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsConsultView.

Definition at line 260 of file BotsConsultView.java.

```
                                        {
            ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.9.3.11 logOut()

```
void view.BotsConsultView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 269 of file BotsConsultView.java.

```
                                    {
            Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
        ButtonType.YES, ButtonType.NO);
            confirm.showAndWait();

            if (confirm.getResult() == ButtonType.YES) {
                ViewCtrl.domainCtrl.logout();
                ViewCtrl.changeScene("template/LogInView.fxml");
            }
    }
```

6.9.4 Member Data Documentation

6.9.4.1 user

```
Text view.BotsConsultView.user  [private]
```

Menu User tab.

Definition at line 45 of file BotsConsultView.java.

6.9.4.2 bots

```
Text view.BotsConsultView.bots  [private]
```

Menu Bots tab.

Definition at line 50 of file BotsConsultView.java.

6.9.4.3 config

```
Text view.BotsConsultView.config  [private]
```

Menu Configuration tab.

Definition at line 55 of file BotsConsultView.java.

6.9.4.4 games

```
Text view.BotsConsultView.games  [private]
```

Menu Games tab.

Definition at line 60 of file BotsConsultView.java.

6.9.4.5 ranking

```
Text view.BotsConsultView.ranking  [private]
```

Menu Ranking tab.

Definition at line 65 of file BotsConsultView.java.

6.9.4.6 play

```
Text view.BotsConsultView.play [private]
```

Menu Play tab.

Definition at line 70 of file BotsConsultView.java.

6.9.4.7 createBot

```
Text view.BotsConsultView.createBot [private]
```

Bot create button text.

Definition at line 75 of file BotsConsultView.java.

6.9.4.8 createBotButton

```
Rectangle view.BotsConsultView.createBotButton [private]
```

Bot create button.

Definition at line 80 of file BotsConsultView.java.

6.9.4.9 modifyBot

```
Text view.BotsConsultView.modifyBot [private]
```

Bot modify button text.

Definition at line 85 of file BotsConsultView.java.

6.9.4.10 modifyBotButton

```
Rectangle view.BotsConsultView.modifyBotButton [private]
```

Bot modify button.

Definition at line 90 of file BotsConsultView.java.

6.9.4.11 consultBot

```
Text view.BotsConsultView.consultBot  [private]
```

Bot consult button text.

Definition at line 95 of file BotsConsultView.java.

6.9.4.12 consultBotButton

```
Rectangle view.BotsConsultView.consultBotButton  [private]
```

Bot consult button.

Definition at line 100 of file BotsConsultView.java.

6.9.4.13 botChooser

```
ChoiceBox view.BotsConsultView.botChooser  [private]
```

Bot choiceBox.

Definition at line 105 of file BotsConsultView.java.

6.9.4.14 name

```
Label view.BotsConsultView.name  [private]
```

Bot name label.

Definition at line 110 of file BotsConsultView.java.

6.9.4.15 difficultyNumber

```
Label view.BotsConsultView.difficultyNumber  [private]
```

Bot difficulty label.

Definition at line 115 of file BotsConsultView.java.

6.9.4.16 consultConfigResult

`Label view.BotsConsultView.consultConfigResult [private]`

Creator name label.

Definition at line 120 of file BotsConsultView.java.

6.9.4.17 consultBotResult

`Label view.BotsConsultView.consultBotResult [private]`

Exception output message label.

Definition at line 125 of file BotsConsultView.java.

6.9.4.18 currentUserName

`Label view.BotsConsultView.currentUserName [private]`

Current user name.

Definition at line 130 of file BotsConsultView.java.

6.9.4.19 creator

`Label view.BotsConsultView.creator [private]`

Bot creator.

Definition at line 135 of file BotsConsultView.java.

6.9.4.20 logOut

`Text view.BotsConsultView.logOut [private]`

LogOut button.

Definition at line 140 of file BotsConsultView.java.

6.9.4.21 botMap

```
Map<String, UUID> view.BotsConsultView.botMap  [private]
```

Map of bots.

Definition at line 144 of file BotsConsultView.java.

The documentation for this class was generated from the following file:

- BotsConsultView.java

6.10 view.BotsCreateView Class Reference

Public Member Functions

- BotsCreateView ()

  Class creator.
- void initialize () throws Exception

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void createBot () throws IOException

  Event method which is executed when the createBot button is clicked.
- void modifyBot () throws IOException

  Event method which is executed when the modifyBot button is clicked.
- void consultBot () throws IOException

  Event method which is executed when the consultBot button is clicked.
- void showLevel ()

  Event method which is executed when the value of the difficulty slider is changed.
- void createBotConfirm () throws IOException

  Event method which is executed when the create button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createBot

  Bot create button text.
- Rectangle createBotButton

  Bot create button.
- Text modifyBot

  Bot modify button text.
- Rectangle modifyBotButton

  Bot modify button.
- Text consultBot

  Bot consult button text.
- Rectangle consultBotButton

  Bot consult button.
- TextField nbotname

  New Bot name text field.
- Slider difficultyLevel

  Slider that controles the difficulty level.
- Label difficultyNumber

  Bot difficulty label.
- Label createBotResult

  Exception output message label.
- Text createBotConfirm

  Bot create confirm text button.
- Rectangle createBotConfirmButton

  Bot create confirm button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.10.1 Detailed Description

This class represents the scene controller of the create function of a bot.

Done by Arnau Pujantell

Definition at line 27 of file BotsCreateView.java.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 BotsCreateView()

`view.BotsCreateView.BotsCreateView ( )`

Class creator.

Definition at line 34 of file BotsCreateView.java.

```
                            {
    }
```

6.10.3 Member Function Documentation

6.10.3.1 initialize()

`void view.BotsCreateView.initialize ( ) throws Exception`

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 147 of file BotsCreateView.java.

```
                                          {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
    }
```

6.10.3.2 user()

`void view.BotsCreateView.user ( ) throws IOException`

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 156 of file BotsCreateView.java.

```
                                      {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.10.3.3 config()

```
void view.BotsCreateView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 165 of file BotsCreateView.java.

```
                                            {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.10.3.4 games()

```
void view.BotsCreateView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 174 of file BotsCreateView.java.

```
                                           {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.10.3.5 ranking()

```
void view.BotsCreateView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 183 of file BotsCreateView.java.

```
                                             {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.10.3.6 play()

`void view.BotsCreateView.play ( ) throws IOException`

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 192 of file BotsCreateView.java.

```
                                          {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.10.3.7 createBot()

`void view.BotsCreateView.createBot ( ) throws IOException`

Event method which is executed when the createBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsCreateView.

Definition at line 201 of file BotsCreateView.java.

```
                                               {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.10.3.8 modifyBot()

`void view.BotsCreateView.modifyBot ( ) throws IOException`

Event method which is executed when the modifyBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsModifyView.

Definition at line 210 of file BotsCreateView.java.

```
                                               {
        ViewCtrl.changeScene("template/BotsModifyView.fxml");
    }
```

6.10.3.9 consultBot()

```
void view.BotsCreateView.consultBot ( ) throws IOException
```

Event method which is executed when the consultBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsConsultView.

Definition at line 219 of file BotsCreateView.java.

```
                                    {
        ViewCtrl.changeScene("template/BotsConsultView.fxml");
    }
```

6.10.3.10 showLevel()

```
void view.BotsCreateView.showLevel ( )
```

Event method which is executed when the value of the difficulty slider is changed.

Precondition

True

Postcondition

The label shows the difficulty level as an Integer.

Definition at line 228 of file BotsCreateView.java.

```
                        {
        difficultyNumber.setText(String.valueOf((int) difficultyLevel.getValue()));
    }
```

6.10.3.11 createBotConfirm()

```
void view.BotsCreateView.createBotConfirm ( ) throws IOException
```

Event method which is executed when the create button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the credentials introduced create a new Bot. Finally, scene changes to BotsView.

Definition at line 237 of file BotsCreateView.java.

```
                                                     {
        Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createBot(nbotname.getText(), (int)
    difficultyLevel.getValue());
        if (result.second != null) {
            switch (result.second) {
                case "ERR_INVALID_NAME":
                    createBotResult.setText("Bot name can't be empty!");
                    break;
                case "ERR_INVALID_DIFFICULTY":
                    createBotResult.setText("This is an invalid difficulty!");
                    break;
                case "ERR_EXISTING_PLAYER":
                    createBotResult.setText("The name is already taken!");
                    break;
                default:
                    createBotResult.setText("Something went wrong, try again!");
                    break;
            }
        }
        else {
            nbotname.clear();
            difficultyLevel.setValue(1);
            showLevel();
            createBotResult.setText("Success!");
        }
    }
```

6.10.3.12 logOut()

```
void view.BotsCreateView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 268 of file BotsCreateView.java.

```
                                           {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.10.4 Member Data Documentation

6.10.4.1 user

```
Text view.BotsCreateView.user  [private]
```

Menu User tab.

Definition at line 43 of file BotsCreateView.java.

6.10.4.2 bots

```
Text view.BotsCreateView.bots  [private]
```

Menu Bots tab.

Definition at line 48 of file BotsCreateView.java.

6.10.4.3 config

```
Text view.BotsCreateView.config  [private]
```

Menu Configuration tab.

Definition at line 53 of file BotsCreateView.java.

6.10.4.4 games

```
Text view.BotsCreateView.games  [private]
```

Menu Games tab.

Definition at line 58 of file BotsCreateView.java.

6.10.4.5 ranking

```
Text view.BotsCreateView.ranking  [private]
```

Menu Ranking tab.

Definition at line 63 of file BotsCreateView.java.

6.10.4.6 play

```
Text view.BotsCreateView.play  [private]
```

Menu Play tab.

Definition at line 68 of file BotsCreateView.java.

6.10.4.7 createBot

```
Text view.BotsCreateView.createBot  [private]
```

Bot create button text.

Definition at line 73 of file BotsCreateView.java.

6.10.4.8 createBotButton

```
Rectangle view.BotsCreateView.createBotButton  [private]
```

Bot create button.

Definition at line 78 of file BotsCreateView.java.

6.10.4.9 modifyBot

```
Text view.BotsCreateView.modifyBot  [private]
```

Bot modify button text.

Definition at line 83 of file BotsCreateView.java.

6.10.4.10 modifyBotButton

```
Rectangle view.BotsCreateView.modifyBotButton  [private]
```

Bot modify button.

Definition at line 88 of file BotsCreateView.java.

6.10.4.11 consultBot

```
Text view.BotsCreateView.consultBot  [private]
```

Bot consult button text.

Definition at line 93 of file BotsCreateView.java.

6.10.4.12 consultBotButton

```
Rectangle view.BotsCreateView.consultBotButton  [private]
```

Bot consult button.

Definition at line 98 of file BotsCreateView.java.

6.10.4.13 nbotname

```
TextField view.BotsCreateView.nbotname  [private]
```

New Bot name text field.

Definition at line 103 of file BotsCreateView.java.

6.10.4.14 difficultyLevel

```
Slider view.BotsCreateView.difficultyLevel  [private]
```

Slider that controles the difficulty level.

Definition at line 108 of file BotsCreateView.java.

6.10.4.15 difficultyNumber

```
Label view.BotsCreateView.difficultyNumber  [private]
```

Bot difficulty label.

Definition at line 113 of file BotsCreateView.java.

6.10.4.16 createBotResult

```
Label view.BotsCreateView.createBotResult  [private]
```

Exception output message label.

Definition at line 118 of file BotsCreateView.java.

6.10.4.17 createBotConfirm

```
Text view.BotsCreateView.createBotConfirm  [private]
```

Bot create confirm text button.

Definition at line 123 of file BotsCreateView.java.

6.10.4.18 createBotConfirmButton

```
Rectangle view.BotsCreateView.createBotConfirmButton  [private]
```

Bot create confirm button.

Definition at line 128 of file BotsCreateView.java.

6.10.4.19 currentUserName

```
Label view.BotsCreateView.currentUserName  [private]
```

Current user name.

Definition at line 133 of file BotsCreateView.java.

6.10.4.20 logOut

```
Text view.BotsCreateView.logOut  [private]
```

LogOut button.

Definition at line 138 of file BotsCreateView.java.

The documentation for this class was generated from the following file:

- BotsCreateView.java

6.11 view.BotsModifyView Class Reference

Public Member Functions

- BotsModifyView ()

  Class creator.
- void initialize ()

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void onChangeBotChooser () throws IOException

  Event method which is executed when the Bot chooser is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void createBot () throws IOException

  Event method which is executed when the createBot button is clicked.
- void modifyBot () throws IOException

  Event method which is executed when the modifyBot button is clicked.
- void consultBot () throws IOException

  Event method which is executed when the consultBot button is clicked.
- void showLevel ()

  Event method which is executed when the value of the difficulty slider is changed.
- void modifyBotConfirm () throws IOException

  Event method which is executed when the modify button is clicked.
- void deleteBot () throws IOException

  Event method which is executed when the delete button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createBot

Bot create button text.
- Rectangle createBotButton

  Bot create button.
- Text modifyBot

  Bot modify button text.
- Rectangle modifyBotButton

  Bot modify button.
- Text consultBot

  Bot consult button text.
- Rectangle consultBotButton

  Bot consult button.
- ChoiceBox botChooser

  Bot choiceBox.
- TextField nbotname

  New Bot name text field.
- Slider difficultyLevel

  Slider that controles the difficulty level.
- Label difficultyNumber

  Bot difficulty label.
- Label modifyBotResult

  Exception output message label.
- Text modifyBotConfirm

  Bot modify confirm text button.
- Rectangle modifyBotConfirmButton

  Bot modify confirm button.
- ImageView deleteBot

  Bot delete image.
- Circle deleteBotButton

  Bot delete button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.
- Map$<$ String, UUID $>$ botMap

  Map of bots.

6.11.1 Detailed Description

This class represents the scene controller of modify function of a bot.

Done by Arnau Pujantell

Definition at line 30 of file BotsModifyView.java.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 BotsModifyView()

```
view.BotsModifyView.BotsModifyView ( )
```

Class creator.

Definition at line 37 of file BotsModifyView.java.

```
                                {
    }
```

6.11.3 Member Function Documentation

6.11.3.1 initialize()

```
void view.BotsModifyView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. All bot names are inserted in the Bot choiceBox and Bot Map is setted.

Definition at line 169 of file BotsModifyView.java.

```
                              {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        botMap = new HashMap<String, UUID>();
        ArrayList<Pair<String, UUID» botList = ViewCtrl.domainCtrl.listBots();
        for(Pair<String, UUID> bot : botList) {
            botChooser.getItems().add(bot.first);
            botMap.put(bot.first, bot.second);
        }
    }
```

6.11.3.2 user()

```
void view.BotsModifyView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 184 of file BotsModifyView.java.

```
                                         {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.11.3.3 onChangeBotChooser()

```
void view.BotsModifyView.onChangeBotChooser ( ) throws IOException
```

Event method which is executed when the Bot chooser is clicked.

Precondition

True

Postcondition

Bot information is shown.

Definition at line 193 of file BotsModifyView.java.

```
                                                          {
        String chosenBot = (String) botChooser.getValue();
        if (chosenBot != null) {
            Pair<JSONObject, String> bot = ViewCtrl.domainCtrl.getBot(botMap.get(chosenBot));
            if (bot.second == null) {
                nbotname.setText(bot.first.getString("name"));
                difficultyLevel.setValue((double) bot.first.getInt("difficulty"));
                showLevel();
            }
        }
    }
```

6.11.3.4 config()

```
void view.BotsModifyView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 210 of file BotsModifyView.java.

```
                                            {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.11.3.5 games()

```
void view.BotsModifyView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 219 of file BotsModifyView.java.

```
                                   {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.11.3.6 ranking()

```
void view.BotsModifyView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 228 of file BotsModifyView.java.

```
                                     {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.11.3.7 play()

```
void view.BotsModifyView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 237 of file BotsModifyView.java.

```
                                  {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.11.3.8 createBot()

`void view.BotsModifyView.createBot ( ) throws IOException`

Event method which is executed when the createBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsCreateView.

Definition at line 246 of file BotsModifyView.java.

```
246                                              {
247         ViewCtrl.changeScene("template/BotsCreateView.fxml");
248     }
```

6.11.3.9 modifyBot()

`void view.BotsModifyView.modifyBot ( ) throws IOException`

Event method which is executed when the modifyBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsModifyView.

Definition at line 255 of file BotsModifyView.java.

```
255                                              {
256         ViewCtrl.changeScene("template/BotsView.fxml");
257     }
```

6.11.3.10 consultBot()

`void view.BotsModifyView.consultBot ( ) throws IOException`

Event method which is executed when the consultBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsConsultView.

Definition at line 264 of file BotsModifyView.java.

```
264                                               {
265         ViewCtrl.changeScene("template/BotsConsultView.fxml");
266     }
```

6.11.3.11 showLevel()

```
void view.BotsModifyView.showLevel ( )
```

Event method which is executed when the value of the difficulty slider is changed.

Precondition

True

Postcondition

The label shows the difficulty level as an Integer.

Definition at line 273 of file BotsModifyView.java.

```
                                {
        difficultyNumber.setText(String.valueOf((int) difficultyLevel.getValue()));
    }
```

6.11.3.12 modifyBotConfirm()

```
void view.BotsModifyView.modifyBotConfirm ( ) throws IOException
```

Event method which is executed when the modify button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the credentials introduced modifies the selected Bot. Finally, scene changes to BotsView.

Definition at line 282 of file BotsModifyView.java.

```
                                                        {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "This bot will be modified. Are you sure?",
   ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            String chosenBot = (String) botChooser.getValue();
            if (chosenBot != null) {
                Pair<JSONObject, String> result = ViewCtrl.domainCtrl.modifyBot(botMap.get(chosenBot),
   nbotname.getText(), (int) difficultyLevel.getValue());
                if (result.second != null) {
                    switch (result.second) {
                        case "ERR_INVALID_NAME":
                            modifyBotResult.setText("Bot name can’t be empty!");
                            break;
                        case "ERR_INVALID_DIFFICULTY":
                            modifyBotResult.setText("This is an invalid difficulty!");
                            break;
                        case "ERR_EXISTING_PLAYER":
                            modifyBotResult.setText("The name is already taken!");
                            break;
                        case "ERR_INEXISTING_PLAYER":
                            modifyBotResult.setText("The player doesn’t exist!");
                            break;
                        case "ERR_BOT_USED":
                            modifyBotResult.setText("This bot is already part of a game!");
                            break;
                        case "ERR_NOT_CREATOR":
                            modifyBotResult.setText("You are not the creator of this bot!");
                            break;
                        default:
                            modifyBotResult.setText("Something went wrong, try again!");
                            break;
                    }
                }
                else {
                    botChooser.getItems().clear();
                    initialize();
                    botChooser.getSelectionModel().select(nbotname.getText());
                    modifyBotResult.setText("Success!");
                }
            }
        }
    }
```

6.11.3.13 deleteBot()

`void view.BotsModifyView.deleteBot ( ) throws IOException`

Event method which is executed when the delete button is clicked.

Precondition

True

Postcondition

The current bot is deleted and the scene is changed to BotsView.

Definition at line 330 of file BotsModifyView.java.

```
                                        {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "This bot will be deleted. Are you sure?",
      ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            String chosenBot = (String) botChooser.getValue();
            if (chosenBot != null) {
                String result = ViewCtrl.domainCtrl.deleteBot(botMap.get(chosenBot));
                if (result != null) {
                    switch (result) {
                    case "ERR_INEXISTING_PLAYER":
                        modifyBotResult.setText("The player doesn't exist!");
                        break;
                    case "ERR_BOT_USED":
                        modifyBotResult.setText("This bot is already part of a game!");
                        break;
                    case "ERR_NOT_CREATOR":
                        modifyBotResult.setText("You are not the creator of this bot!");
                        break;
                    default:
                        modifyBotResult.setText("Something went wrong, try again!");
                        break;
                }
            }
            else {
                nbotname.clear();
                botChooser.getItems().clear();
                difficultyLevel.setValue(1);
                initialize();
                showLevel();
                modifyBotResult.setText("Success!");
            }
        }
    }
}
```

6.11.3.14 logOut()

```
void view.BotsModifyView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 371 of file BotsModifyView.java.

```
                                     {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
     ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.11.4 Member Data Documentation

6.11.4.1 user

```
Text view.BotsModifyView.user  [private]
```

Menu User tab.

Definition at line 46 of file BotsModifyView.java.

6.11.4.2 bots

```
Text view.BotsModifyView.bots  [private]
```

Menu Bots tab.

Definition at line 51 of file BotsModifyView.java.

6.11.4.3 config

```
Text view.BotsModifyView.config  [private]
```

Menu Configuration tab.

Definition at line 56 of file BotsModifyView.java.

6.11.4.4 games

```
Text view.BotsModifyView.games  [private]
```

Menu Games tab.

Definition at line 61 of file BotsModifyView.java.

6.11.4.5 ranking

```
Text view.BotsModifyView.ranking  [private]
```

Menu Ranking tab.

Definition at line 66 of file BotsModifyView.java.

6.11.4.6 play

```
Text view.BotsModifyView.play  [private]
```

Menu Play tab.

Definition at line 71 of file BotsModifyView.java.

6.11.4.7 createBot

```
Text view.BotsModifyView.createBot  [private]
```

Bot create button text.

Definition at line 76 of file BotsModifyView.java.

6.11.4.8 createBotButton

```
Rectangle view.BotsModifyView.createBotButton [private]
```

Bot create button.

Definition at line 81 of file BotsModifyView.java.

6.11.4.9 modifyBot

```
Text view.BotsModifyView.modifyBot [private]
```

Bot modify button text.

Definition at line 86 of file BotsModifyView.java.

6.11.4.10 modifyBotButton

```
Rectangle view.BotsModifyView.modifyBotButton [private]
```

Bot modify button.

Definition at line 91 of file BotsModifyView.java.

6.11.4.11 consultBot

```
Text view.BotsModifyView.consultBot [private]
```

Bot consult button text.

Definition at line 96 of file BotsModifyView.java.

6.11.4.12 consultBotButton

```
Rectangle view.BotsModifyView.consultBotButton [private]
```

Bot consult button.

Definition at line 101 of file BotsModifyView.java.

6.11.4.13 botChooser

```
ChoiceBox view.BotsModifyView.botChooser [private]
```

Bot choiceBox.

Definition at line 106 of file BotsModifyView.java.

6.11.4.14 nbotname

```
TextField view.BotsModifyView.nbotname [private]
```

New Bot name text field.

Definition at line 111 of file BotsModifyView.java.

6.11.4.15 difficultyLevel

```
Slider view.BotsModifyView.difficultyLevel [private]
```

Slider that controles the difficulty level.

Definition at line 116 of file BotsModifyView.java.

6.11.4.16 difficultyNumber

```
Label view.BotsModifyView.difficultyNumber [private]
```

Bot difficulty label.

Definition at line 121 of file BotsModifyView.java.

6.11.4.17 modifyBotResult

```
Label view.BotsModifyView.modifyBotResult [private]
```

Exception output message label.

Definition at line 126 of file BotsModifyView.java.

6.11.4.18 modifyBotConfirm

```
Text view.BotsModifyView.modifyBotConfirm  [private]
```

Bot modify confirm text button.

Definition at line 131 of file BotsModifyView.java.

6.11.4.19 modifyBotConfirmButton

```
Rectangle view.BotsModifyView.modifyBotConfirmButton  [private]
```

Bot modify confirm button.

Definition at line 136 of file BotsModifyView.java.

6.11.4.20 deleteBot

```
ImageView view.BotsModifyView.deleteBot  [private]
```

Bot delete image.

Definition at line 141 of file BotsModifyView.java.

6.11.4.21 deleteBotButton

```
Circle view.BotsModifyView.deleteBotButton  [private]
```

Bot delete button.

Definition at line 146 of file BotsModifyView.java.

6.11.4.22 currentUserName

```
Label view.BotsModifyView.currentUserName  [private]
```

Current user name.

Definition at line 151 of file BotsModifyView.java.

6.11.4.23 logOut

```
Text view.BotsModifyView.logOut  [private]
```

LogOut button.

Definition at line 156 of file BotsModifyView.java.

6.11.4.24 botMap

```
Map<String, UUID> view.BotsModifyView.botMap  [private]
```

Map of bots.

Definition at line 160 of file BotsModifyView.java.

The documentation for this class was generated from the following file:

- BotsModifyView.java

6.12 view.BotsView Class Reference

Public Member Functions

- BotsView ()

  Class creator.
- void initialize () throws Exception

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void createBot () throws IOException

  Event method which is executed when the createBot button is clicked.
- void modifyBot () throws IOException

  Event method which is executed when the modifyBot button is clicked.
- void consultBot () throws IOException

  Event method which is executed when the consultBot button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createBot

  Bot create button text.
- Rectangle createBotButton

  Bot create button.
- Text modifyBot

  Bot modify button text.
- Rectangle modifyBotButton

  Bot modify button.
- Text consultBot

  Bot consult button text.
- Rectangle consultBotButton

  Bot consult button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.12.1 Detailed Description

This class represents the scene controller of the Bot Menu .

Done by Arnau Pujantell

Definition at line 22 of file BotsView.java.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 BotsView()

```
view.BotsView.BotsView ( )
```

Class creator.

Definition at line 29 of file BotsView.java.

```
                          {
    }
```

6.12.3 Member Function Documentation

6.12.3.1 initialize()

```
void view.BotsView.initialize ( ) throws Exception
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 111 of file BotsView.java.

```
                                            {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
    }
```

6.12.3.2 user()

```
void view.BotsView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 120 of file BotsView.java.

```
                                        {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.12.3.3 config()

```
void view.BotsView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 129 of file BotsView.java.

```
                                                 {
            ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.12.3.4 games()

```
void view.BotsView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 138 of file BotsView.java.

```
                                                {
            ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.12.3.5 ranking()

```
void view.BotsView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 147 of file BotsView.java.

```
                                                  {
            ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.12.3.6 play()

```
void view.BotsView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 156 of file BotsView.java.

```
                                                {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.12.3.7 createBot()

```
void view.BotsView.createBot ( ) throws IOException
```

Event method which is executed when the createBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsCreateView.

Definition at line 165 of file BotsView.java.

```
                                                        {
        ViewCtrl.changeScene("template/BotsCreateView.fxml");
    }
```

6.12.3.8 modifyBot()

```
void view.BotsView.modifyBot ( ) throws IOException
```

Event method which is executed when the modifyBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsModifyView.

Definition at line 174 of file BotsView.java.

```
                                                        {
        ViewCtrl.changeScene("template/BotsModifyView.fxml");
    }
```

6.12.3.9 consultBot()

`void view.BotsView.consultBot ( ) throws IOException`

Event method which is executed when the consultBot button is clicked.

Precondition

True

Postcondition

Scene changes to BotsConsultView.

Definition at line 183 of file BotsView.java.

```
                                          {
        ViewCtrl.changeScene("template/BotsConsultView.fxml");
    }
```

6.12.3.10 logOut()

`void view.BotsView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 192 of file BotsView.java.

```
                                      {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
   ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.12.4 Member Data Documentation

6.12.4.1 user

```
Text view.BotsView.user [private]
```

Menu User tab.

Definition at line 38 of file BotsView.java.

6.12.4.2 bots

```
Text view.BotsView.bots [private]
```

Menu Bots tab.

Definition at line 43 of file BotsView.java.

6.12.4.3 config

```
Text view.BotsView.config [private]
```

Menu Configuration tab.

Definition at line 48 of file BotsView.java.

6.12.4.4 games

```
Text view.BotsView.games [private]
```

Menu Games tab.

Definition at line 53 of file BotsView.java.

6.12.4.5 ranking

```
Text view.BotsView.ranking [private]
```

Menu Ranking tab.

Definition at line 58 of file BotsView.java.

6.12.4.6 play

```
Text view.BotsView.play  [private]
```

Menu Play tab.

Definition at line 63 of file BotsView.java.

6.12.4.7 createBot

```
Text view.BotsView.createBot  [private]
```

Bot create button text.

Definition at line 68 of file BotsView.java.

6.12.4.8 createBotButton

```
Rectangle view.BotsView.createBotButton  [private]
```

Bot create button.

Definition at line 73 of file BotsView.java.

6.12.4.9 modifyBot

```
Text view.BotsView.modifyBot  [private]
```

Bot modify button text.

Definition at line 78 of file BotsView.java.

6.12.4.10 modifyBotButton

```
Rectangle view.BotsView.modifyBotButton  [private]
```

Bot modify button.

Definition at line 83 of file BotsView.java.

6.12.4.11 consultBot

```
Text view.BotsView.consultBot  [private]
```

Bot consult button text.

Definition at line 88 of file BotsView.java.

6.12.4.12 consultBotButton

```
Rectangle view.BotsView.consultBotButton  [private]
```

Bot consult button.

Definition at line 93 of file BotsView.java.

6.12.4.13 currentUserName

```
Label view.BotsView.currentUserName  [private]
```

Current user name.

Definition at line 98 of file BotsView.java.

6.12.4.14 logOut

```
Text view.BotsView.logOut  [private]
```

LogOut button.

Definition at line 103 of file BotsView.java.

The documentation for this class was generated from the following file:

- BotsView.java

6.13 domain.Exceptions.BotUsedException Class Reference

A bot cannot be modified or deleted if it is already part of a game. By Alex Rodriguez.

Public Member Functions

- BotUsedException ()

6.13.1 Detailed Description

A bot cannot be modified or deleted if it is already part of a game. By Alex Rodriguez.

Definition at line 107 of file Exceptions.java.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 BotUsedException()

```
domain.Exceptions.BotUsedException.BotUsedException ( )
```

Definition at line 108 of file Exceptions.java.

```
                                    {
                super("ERR_BOT_USED");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.14 view.ConfigConsultView Class Reference

Public Member Functions

- ConfigConsultView ()

  Class creator.
- void initialize ()

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void createConfig () throws IOException

  Event method which is executed when the createConfig button is clicked.
- void modifyConfig () throws IOException

  Event method which is executed when the modifyConfig button is clicked.
- void consultConfig () throws IOException

  Event method which is executed when the consultConfig button is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void onChangeConfigChooser () throws IOException

  Event method which is executed when the Configuration chooser is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void checkInitialBoard () throws IOException

  Event method which is executed when the checkInitialBoard button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createConfig

  Configuration create button text.
- Rectangle createConfigButton

  Configuration create button.
- Text modifyConfig

  Configuration modify button text.
- Rectangle modifyConfigButton

  Configuration modify button.
- Text consultConfig

  Configuration consult button text.
- Rectangle consultConfigButton

  Configuration consult button.
- Text consultConfigConfirm

  Configuration consult confirm text button.
- Rectangle consultConfigConfirmButton

  Configuration consult confirm button.
- ChoiceBox configChooser

  Configuration choiceBox.
- Label name

  Configuration name label.
- Label ceh

  Configuration CanEatHorizontally label.
- Label cev

  Configuration CanEatVertically label.
- Label ced

  Configuration CanEatDiagonally label.
- Text checkInitialBoard

  Initial board check text button.
- Rectangle checkInitialBoardButton

  Initial board check button.
- Label creator

  Creator name label.
- Label consultConfigResult

  Exception output message label.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.14.1 Detailed Description

This class represents the scene controller of consult function of a configuration.

Done by Arnau Pujantell

Definition at line 27 of file ConfigConsultView.java.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 ConfigConsultView()

```
view.ConfigConsultView.ConfigConsultView ( )
```

Class creator.

Definition at line 34 of file ConfigConsultView.java.

```
                                  {
    }
```

6.14.3 Member Function Documentation

6.14.3.1 initialize()

```
void view.ConfigConsultView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. All configuration names are inserted in the Configuration choiceBox.

Definition at line 171 of file ConfigConsultView.java.

```
                                  {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        ArrayList<String> configList = ViewCtrl.domainCtrl.listConfigurations().first;
        for(String configName : configList) configChooser.getItems().add(configName);
    }
```

6.14.3.2 user()

```
void view.ConfigConsultView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 182 of file ConfigConsultView.java.

```
                                         {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.14.3.3 createConfig()

```
void view.ConfigConsultView.createConfig ( ) throws IOException
```

Event method which is executed when the createConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigCreateView.

Definition at line 191 of file ConfigConsultView.java.

```
                                                 {
        ViewCtrl.changeScene("template/ConfigCreateView.fxml");
    }
```

6.14.3.4 modifyConfig()

```
void view.ConfigConsultView.modifyConfig ( ) throws IOException
```

Event method which is executed when the modifyConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigModifyView.

Definition at line 200 of file ConfigConsultView.java.

```
                                                 {
        ViewCtrl.changeScene("template/ConfigModifyView.fxml");
    }
```

6.14.3.5 consultConfig()

```
void view.ConfigConsultView.consultConfig ( ) throws IOException
```

Event method which is executed when the consultConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigConsultView.

Definition at line 209 of file ConfigConsultView.java.

```
                                        {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.14.3.6 bots()

```
void view.ConfigConsultView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 218 of file ConfigConsultView.java.

```
                               {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.14.3.7 onChangeConfigChooser()

`void view.ConfigConsultView.onChangeConfigChooser ( ) throws IOException`

Event method which is executed when the Configuration chooser is clicked.

Precondition

True

Postcondition

Configuration information is shown.

Definition at line 227 of file ConfigConsultView.java.

```
227                                                             {
228         String chosenConfig = (String) configChooser.getValue();
229         if (chosenConfig != null) {
230             Pair<JSONObject, String> result = ViewCtrl.domainCtrl.getConfiguration(chosenConfig);
231             if (result.second != null) {
232                 switch (result.second) {
233                     case "ERR_INEXISTING_CONFIGURATION":
234                         consultConfigResult.setText("The configuration doesn't exist!");
235                         break;
236                     default:
237                         consultConfigResult.setText("Something went wrong, try again!");
238                         break;
239                 }
240             }
241             else {
242                 consultConfigResult.setText("");
243                 ViewCtrl.domainCtrl.modifyInitialBoard(result.first.getString("name")); // Load onto
       memory the chosen config Board
244                 name.setText(result.first.getString("name"));
245                 ceh.setText((result.first.getBoolean("can_eat_horizontally") ? "Can Eat Horizontally" :
       ""));
246                 cev.setText((result.first.getBoolean("can_eat_vertically") ? "Can Eat Vertically" :
       ""));
247                 ced.setText((result.first.getBoolean("can_eat_diagonally") ? "Can Eat Diagonally" :
       ""));
248                 Pair<JSONObject, String> user =
       ViewCtrl.domainCtrl.getUser(UUID.fromString(result.first.getString("creator_id")));
249                 creator.setText((user.first != null ? user.first.getString("name") : "Unknown"));
250             }
251         }
252     }
```

6.14.3.8 games()

`void view.ConfigConsultView.games ( ) throws IOException`

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 259 of file ConfigConsultView.java.

```
259                                             {
260         ViewCtrl.changeScene("template/GamesView.fxml");
261     }
```

6.14.3.9 ranking()

```
void view.ConfigConsultView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 268 of file ConfigConsultView.java.

```
                                                {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.14.3.10 play()

```
void view.ConfigConsultView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 277 of file ConfigConsultView.java.

```
                                             {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.14.3.11 checkInitialBoard()

```
void view.ConfigConsultView.checkInitialBoard ( ) throws IOException
```

Event method which is executed when the checkInitialBoard button is clicked.

Precondition

True

Postcondition

Scene changes to ConsultInitialBoardView.

Definition at line 286 of file ConfigConsultView.java.

```
                                                          {
        String chosenConfig = (String) configChooser.getValue();
        if (chosenConfig != null) {
            ViewCtrl.newWindow("template/ConsultInitialBoardView.fxml");
        }
    }
```

6.14.3.12 logOut()

```
void view.ConfigConsultView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 298 of file ConfigConsultView.java.

```
                                              {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.14.4 Member Data Documentation

6.14.4.1 user

```
Text view.ConfigConsultView.user [private]
```

Menu User tab.

Definition at line 42 of file ConfigConsultView.java.

6.14.4.2 bots

```
Text view.ConfigConsultView.bots [private]
```

Menu Bots tab.

Definition at line 47 of file ConfigConsultView.java.

6.14.4.3 config

```
Text view.ConfigConsultView.config  [private]
```

Menu Configuration tab.

Definition at line 52 of file ConfigConsultView.java.

6.14.4.4 games

```
Text view.ConfigConsultView.games  [private]
```

Menu Games tab.

Definition at line 57 of file ConfigConsultView.java.

6.14.4.5 ranking

```
Text view.ConfigConsultView.ranking  [private]
```

Menu Ranking tab.

Definition at line 62 of file ConfigConsultView.java.

6.14.4.6 play

```
Text view.ConfigConsultView.play  [private]
```

Menu Play tab.

Definition at line 67 of file ConfigConsultView.java.

6.14.4.7 createConfig

```
Text view.ConfigConsultView.createConfig  [private]
```

Configuration create button text.

Definition at line 72 of file ConfigConsultView.java.

6.14.4.8 createConfigButton

```
Rectangle view.ConfigConsultView.createConfigButton  [private]
```

Configuration create button.

Definition at line 77 of file ConfigConsultView.java.

6.14.4.9 modifyConfig

```
Text view.ConfigConsultView.modifyConfig  [private]
```

Configuration modify button text.

Definition at line 82 of file ConfigConsultView.java.

6.14.4.10 modifyConfigButton

```
Rectangle view.ConfigConsultView.modifyConfigButton  [private]
```

Configuration modify button.

Definition at line 87 of file ConfigConsultView.java.

6.14.4.11 consultConfig

```
Text view.ConfigConsultView.consultConfig  [private]
```

Configuration consult button text.

Definition at line 92 of file ConfigConsultView.java.

6.14.4.12 consultConfigButton

```
Rectangle view.ConfigConsultView.consultConfigButton  [private]
```

Configuration consult button.

Definition at line 97 of file ConfigConsultView.java.

6.14.4.13 consultConfigConfirm

```
Text view.ConfigConsultView.consultConfigConfirm  [private]
```

Configuration consult confirm text button.

Definition at line 102 of file ConfigConsultView.java.

6.14.4.14 consultConfigConfirmButton

```
Rectangle view.ConfigConsultView.consultConfigConfirmButton  [private]
```

Configuration consult confirm button.

Definition at line 107 of file ConfigConsultView.java.

6.14.4.15 configChooser

```
ChoiceBox view.ConfigConsultView.configChooser  [private]
```

Configuration choiceBox.

Definition at line 112 of file ConfigConsultView.java.

6.14.4.16 name

```
Label view.ConfigConsultView.name  [private]
```

Configuration name label.

Definition at line 117 of file ConfigConsultView.java.

6.14.4.17 ceh

```
Label view.ConfigConsultView.ceh  [private]
```

Configuration CanEatHorizontally label.

Definition at line 122 of file ConfigConsultView.java.

6.14.4.18 cev

```
Label view.ConfigConsultView.cev  [private]
```

Configuration CanEatVertically label.

Definition at line 127 of file ConfigConsultView.java.

6.14.4.19 ced

```
Label view.ConfigConsultView.ced  [private]
```

Configuration CanEatDiagonally label.

Definition at line 132 of file ConfigConsultView.java.

6.14.4.20 checkInitialBoard

```
Text view.ConfigConsultView.checkInitialBoard  [private]
```

Initial board check text button.

Definition at line 137 of file ConfigConsultView.java.

6.14.4.21 checkInitialBoardButton

```
Rectangle view.ConfigConsultView.checkInitialBoardButton  [private]
```

Initial board check button.

Definition at line 142 of file ConfigConsultView.java.

6.14.4.22 creator

```
Label view.ConfigConsultView.creator  [private]
```

Creator name label.

Definition at line 147 of file ConfigConsultView.java.

6.14.4.23 consultConfigResult

```
Label view.ConfigConsultView.consultConfigResult  [private]
```

Exception output message label.

Definition at line 152 of file ConfigConsultView.java.

6.14.4.24 currentUserName

```
Label view.ConfigConsultView.currentUserName  [private]
```

Current user name.

Definition at line 157 of file ConfigConsultView.java.

6.14.4.25 logOut

```
Text view.ConfigConsultView.logOut  [private]
```

LogOut button.

Definition at line 162 of file ConfigConsultView.java.

The documentation for this class was generated from the following file:

- ConfigConsultView.java

6.15 view.ConfigCreateView Class Reference

Public Member Functions

- ConfigCreateView ()

  Class creator.
- void initialize ()

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

Event method which is executed when the Play tab is clicked.

- void createConfig () throws IOException

    Event method which is executed when the createConfig button is clicked.
- void modifyConfig () throws IOException

    Event method which is executed when the modifyConfig button is clicked.
- void consultConfig () throws IOException

    Event method which is executed when the consultConfig button is clicked.
- void createInitialBoard () throws IOException

    Event method which is executed when the createInitialBoard button is clicked.
- void createConfigConfirm () throws IOException

    Event method which is executed when the create button is clicked.
- void logOut () throws IOException

    Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

    Menu User tab.
- Text bots

    Menu Bots tab.
- Text config

    Menu Configuration tab.
- Text games

    Menu Games tab.
- Text ranking

    Menu Ranking tab.
- Text play

    Menu Play tab.
- Text createConfig

    Configuration create button text.
- Rectangle createConfigButton

    Configuration create button.
- Text modifyConfig

    Configuration modify button text.
- Rectangle modifyConfigButton

    Configuration modify button.
- Text consultConfig

    Configuration consult button text.
- Rectangle consultConfigButton

    Configuration consult button.
- Text createInitialBoard

    Initial board creation button text.
- Rectangle createInitialBoardButton

    Initial board creation button.
- TextField nconfname

    New Configuration name text field.
- RadioButton canEatHorizontally

    CanEatHorizontally selector.
- RadioButton canEatVertically

    CanEatVertically selector.

- RadioButton canEatDiagonally

  CanEatDiagonally selector.
- Label createConfigResult

  Exception output message label.
- Text createConfigConfirm

  Configuration create confirm button text.
- Rectangle createConfigConfirmButton

  Configuration create confirm button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.15.1 Detailed Description

This class represents the scene controller of creation function of a configuration.

Done by Arnau Pujantell

Definition at line 26 of file ConfigCreateView.java.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 ConfigCreateView()

```
view.ConfigCreateView.ConfigCreateView ( )
```

Class creator.

Definition at line 33 of file ConfigCreateView.java.

```
                                    {
    }
```

6.15.3 Member Function Documentation

6.15.3.1 initialize()

```
void view.ConfigCreateView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. All configuration names are inserted in the Configuration choiceBox.

Definition at line 160 of file ConfigCreateView.java.

```
                                    {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        ViewCtrl.domainCtrl.createInitialBoard();
    }
```

6.15.3.2 user()

```
void view.ConfigCreateView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 170 of file ConfigCreateView.java.

```
                                        {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.15.3.3 bots()

```
void view.ConfigCreateView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 179 of file ConfigCreateView.java.

```
                                        {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.15.3.4 games()

```
void view.ConfigCreateView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 188 of file ConfigCreateView.java.

```
                                         {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.15.3.5 ranking()

```
void view.ConfigCreateView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 197 of file ConfigCreateView.java.

```
                                       {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.15.3.6 play()

```
void view.ConfigCreateView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 206 of file ConfigCreateView.java.

```
                                    {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.15.3.7 createConfig()

```
void view.ConfigCreateView.createConfig ( ) throws IOException
```

Event method which is executed when the createConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigCreateView.

Definition at line 215 of file ConfigCreateView.java.

```
                                            {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.15.3.8 modifyConfig()

`void view.ConfigCreateView.modifyConfig ( ) throws IOException`

Event method which is executed when the modifyConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigModifyView.

Definition at line 224 of file ConfigCreateView.java.

```
                                            {
        ViewCtrl.changeScene("template/ConfigModifyView.fxml");
    }
```

6.15.3.9 consultConfig()

`void view.ConfigCreateView.consultConfig ( ) throws IOException`

Event method which is executed when the consultConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigConsultView.

Definition at line 233 of file ConfigCreateView.java.

```
                                             {
        ViewCtrl.changeScene("template/ConfigConsultView.fxml");
    }
```

6.15.3.10 createInitialBoard()

`void view.ConfigCreateView.createInitialBoard ( ) throws IOException`

Event method which is executed when the createInitialBoard button is clicked.

Precondition

True

Postcondition

Scene changes to InitialBoardView.

Definition at line 242 of file ConfigCreateView.java.

```
                                                  {
        ViewCtrl.newWindow("template/InitialBoardView.fxml");
    }
```

6.15.3.11 createConfigConfirm()

```
void view.ConfigCreateView.createConfigConfirm ( ) throws IOException
```

Event method which is executed when the create button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the new Configuration is created.

Definition at line 251 of file ConfigCreateView.java.

```
                                                          {
        Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createConfiguration(nconfname.getText(),
    canEatHorizontally.isSelected(), canEatVertically.isSelected(), canEatDiagonally.isSelected());
        if (result.second != null) {
            switch (result.second) {
                case "ERR_INVALID_NAME":
                    createConfigResult.setText("Configuration name can't be empty!");
                    break;
                case "ERR_EXISTING_CONFIGURATION":
                    createConfigResult.setText("The configuration name is already taken!");
                    break;
                case "ERR_INVALID_BOARD":
                    createConfigResult.setText("The initial board is invalid!");
                    break;
                case "ERR_INVALID_RULES":
                    createConfigResult.setText("You must select at least one rule!");
                    break;
                default:
                    createConfigResult.setText("Something went wrong, try again!");
                    break;
            }
        }
        else {
            nconfname.clear();
            canEatHorizontally.setSelected(false);
            canEatVertically.setSelected(false);
            canEatDiagonally.setSelected(false);
            initialize();
            createConfigResult.setText("Success!");
        }
    }
```

6.15.3.12 logOut()

```
void view.ConfigCreateView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 287 of file ConfigCreateView.java.

```
                                             {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.15.4 Member Data Documentation

6.15.4.1 user

`Text view.ConfigCreateView.user [private]`

Menu User tab.

Definition at line 41 of file ConfigCreateView.java.

6.15.4.2 bots

`Text view.ConfigCreateView.bots [private]`

Menu Bots tab.

Definition at line 46 of file ConfigCreateView.java.

6.15.4.3 config

`Text view.ConfigCreateView.config [private]`

Menu Configuration tab.

Definition at line 51 of file ConfigCreateView.java.

6.15.4.4 games

`Text view.ConfigCreateView.games [private]`

Menu Games tab.

Definition at line 56 of file ConfigCreateView.java.

6.15.4.5 ranking

`Text view.ConfigCreateView.ranking [private]`

Menu Ranking tab.

Definition at line 61 of file ConfigCreateView.java.

6.15.4.6 play

```
Text view.ConfigCreateView.play [private]
```

Menu Play tab.

Definition at line 66 of file ConfigCreateView.java.

6.15.4.7 createConfig

```
Text view.ConfigCreateView.createConfig [private]
```

Configuration create button text.

Definition at line 71 of file ConfigCreateView.java.

6.15.4.8 createConfigButton

```
Rectangle view.ConfigCreateView.createConfigButton [private]
```

Configuration create button.

Definition at line 76 of file ConfigCreateView.java.

6.15.4.9 modifyConfig

```
Text view.ConfigCreateView.modifyConfig [private]
```

Configuration modify button text.

Definition at line 81 of file ConfigCreateView.java.

6.15.4.10 modifyConfigButton

```
Rectangle view.ConfigCreateView.modifyConfigButton [private]
```

Configuration modify button.

Definition at line 86 of file ConfigCreateView.java.

6.15.4.11 consultConfig

```
Text view.ConfigCreateView.consultConfig  [private]
```

Configuration consult button text.

Definition at line 91 of file ConfigCreateView.java.

6.15.4.12 consultConfigButton

```
Rectangle view.ConfigCreateView.consultConfigButton  [private]
```

Configuration consult button.

Definition at line 96 of file ConfigCreateView.java.

6.15.4.13 createInitialBoard

```
Text view.ConfigCreateView.createInitialBoard  [private]
```

Initial board creation button text.

Definition at line 101 of file ConfigCreateView.java.

6.15.4.14 createInitialBoardButton

```
Rectangle view.ConfigCreateView.createInitialBoardButton  [private]
```

Initial board creation button.

Definition at line 106 of file ConfigCreateView.java.

6.15.4.15 nconfname

```
TextField view.ConfigCreateView.nconfname  [private]
```

New Configuration name text field.

Definition at line 111 of file ConfigCreateView.java.

6.15.4.16 canEatHorizontally

```
RadioButton view.ConfigCreateView.canEatHorizontally  [private]
```

CanEatHorizontally selector.

Definition at line 116 of file ConfigCreateView.java.

6.15.4.17 canEatVertically

```
RadioButton view.ConfigCreateView.canEatVertically  [private]
```

CanEatVertically selector.

Definition at line 121 of file ConfigCreateView.java.

6.15.4.18 canEatDiagonally

```
RadioButton view.ConfigCreateView.canEatDiagonally  [private]
```

CanEatDiagonally selector.

Definition at line 126 of file ConfigCreateView.java.

6.15.4.19 createConfigResult

```
Label view.ConfigCreateView.createConfigResult  [private]
```

Exception output message label.

Definition at line 131 of file ConfigCreateView.java.

6.15.4.20 createConfigConfirm

```
Text view.ConfigCreateView.createConfigConfirm  [private]
```

Configuration create confirm button text.

Definition at line 136 of file ConfigCreateView.java.

6.15.4.21 createConfigConfirmButton

```
Rectangle view.ConfigCreateView.createConfigConfirmButton  [private]
```

Configuration create confirm button.

Definition at line 141 of file ConfigCreateView.java.

6.15.4.22 currentUserName

```
Label view.ConfigCreateView.currentUserName  [private]
```

Current user name.

Definition at line 146 of file ConfigCreateView.java.

6.15.4.23 logOut

```
Text view.ConfigCreateView.logOut  [private]
```

LogOut button.

Definition at line 151 of file ConfigCreateView.java.

The documentation for this class was generated from the following file:

- ConfigCreateView.java

6.16 view.ConfigModifyView Class Reference

Public Member Functions

- ConfigModifyView ()
  Class creator.
- void initialize ()
  Initialize method which is executed when the scene is shown.
- void user () throws IOException
  Event method which is executed when the User tab is clicked.
- void bots () throws IOException
  Event method which is executed when the Bots tab is clicked.
- void onChangeConfigChooser () throws IOException
  Event method which is executed when the Configuration chooser is clicked.
- void games () throws IOException
  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void createConfig () throws IOException

  Event method which is executed when the createConfig button is clicked.
- void modifyConfig () throws IOException

  Event method which is executed when the modifyConfig button is clicked.
- void consultConfig () throws IOException

  Event method which is executed when the consultConfig button is clicked.
- void modifyInitialBoard () throws IOException

  Event method which is executed when the modifyInitialBoard button is clicked.
- void modifyConfigConfirm () throws IOException

  Event method which is executed when the modify button is clicked.
- void deleteConfig () throws IOException

  Event method which is executed when the delete button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createConfig

  Configuration create button text.
- Rectangle createConfigButton

  Configuration create button.
- Text modifyConfig

  Configuration modify button text.
- Rectangle modifyConfigButton

  Configuration modify button.
- Text consultConfig

  Configuration consult button text.
- Rectangle consultConfigButton

  Configuration consult button.
- RadioButton canEatHorizontally

  CanEatHorizontally selector.
- RadioButton canEatVertically

  CanEatVertically selector.
- RadioButton canEatDiagonally

  CanEatDiagonally selector.

- Text modifyInitialBoard

  Modify initial board button text.
- Rectangle modifyInitialBoardButton

  Modify initial board button.
- Text modifyConfigConfirm

  Configuration modify confirm button text.
- Rectangle modifyConfigConfirmButton

  Configuration modify confirm button.
- ImageView deleteConfig

  Configuration delete button image.
- Circle deleteConfigButton

  Configuration delete button.
- ChoiceBox configChooser

  Configuration choiceBox.
- Label modifyConfigResult

  Exception output message label.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.16.1 Detailed Description

This class represents the scene controller of modify function of a configuration.

Done by Arnau Pujantell

Definition at line 29 of file ConfigModifyView.java.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 ConfigModifyView()

```
view.ConfigModifyView.ConfigModifyView ( )
```

Class creator.

Definition at line 36 of file ConfigModifyView.java.

```
                                    {
    }
```

6.16.3 Member Function Documentation

6.16.3.1 initialize()

```
void view.ConfigModifyView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

All The current username is shown. configuration names are inserted in the Configuration choiceBox.

Definition at line 174 of file ConfigModifyView.java.

```
                             {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        ArrayList<String> configList = ViewCtrl.domainCtrl.listConfigurations().first;
        for(String configName : configList) configChooser.getItems().add(configName);
    }
```

6.16.3.2 user()

```
void view.ConfigModifyView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 185 of file ConfigModifyView.java.

```
                                          {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.16.3.3 bots()

```
void view.ConfigModifyView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 194 of file ConfigModifyView.java.

```
                                          {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.16.3.4 onChangeConfigChooser()

```
void view.ConfigModifyView.onChangeConfigChooser ( ) throws IOException
```

Event method which is executed when the Configuration chooser is clicked.

Precondition

True

Postcondition

Configuration information is shown.

Definition at line 203 of file ConfigModifyView.java.

```
                                                            {
        String chosenConfig = (String) configChooser.getValue();
        if (chosenConfig != null) {
            Pair<JSONObject, String> config = ViewCtrl.domainCtrl.getConfiguration(chosenConfig);
            if (config.second == null) {
                ViewCtrl.domainCtrl.modifyInitialBoard(config.first.getString("name")); // Load onto
       memory the chosen config Board
                canEatHorizontally.setSelected(config.first.getBoolean("can_eat_horizontally"));
                canEatVertically.setSelected(config.first.getBoolean("can_eat_vertically"));
                canEatDiagonally.setSelected(config.first.getBoolean("can_eat_diagonally"));
            }
        }
    }
```

6.16.3.5 games()

```
void view.ConfigModifyView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 221 of file ConfigModifyView.java.

```
                                           {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.16.3.6 ranking()

```
void view.ConfigModifyView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 230 of file ConfigModifyView.java.

```
                                         {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.16.3.7 play()

```
void view.ConfigModifyView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 239 of file ConfigModifyView.java.

```
                                      {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.16.3.8 createConfig()

```
void view.ConfigModifyView.createConfig ( ) throws IOException
```

Event method which is executed when the createConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigCreateView.

Definition at line 248 of file ConfigModifyView.java.

```
                                              {
        ViewCtrl.changeScene("template/ConfigCreateView.fxml");
    }
```

6.16.3.9 modifyConfig()

`void view.ConfigModifyView.modifyConfig ( ) throws IOException`

Event method which is executed when the modifyConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigModifyView.

Definition at line 257 of file ConfigModifyView.java.

```
                                                     {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.16.3.10 consultConfig()

`void view.ConfigModifyView.consultConfig ( ) throws IOException`

Event method which is executed when the consultConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigConsultView.

Definition at line 266 of file ConfigModifyView.java.

```
                                                      {
        ViewCtrl.changeScene("template/ConfigConsultView.fxml");
    }
```

6.16.3.11 modifyInitialBoard()

`void view.ConfigModifyView.modifyInitialBoard ( ) throws IOException`

Event method which is executed when the modifyInitialBoard button is clicked.

Precondition

True

Postcondition

Scene changes to ModifyInitialBoardView.

Definition at line 275 of file ConfigModifyView.java.

```
                                                           {
        String chosenConfig = (String) configChooser.getValue();
        if (chosenConfig != null) {
            ViewCtrl.newWindow("template/ModifyInitialBoardView.fxml");
        }
    }
```

6.16.3.12 modifyConfigConfirm()

`void view.ConfigModifyView.modifyConfigConfirm ( ) throws IOException`

Event method which is executed when the modify button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the new Configuration is modified.

Definition at line 287 of file ConfigModifyView.java.

```
287                                                          {
288         Alert confirm = new Alert(AlertType.CONFIRMATION, "This configuration will be modified. Are you sure?", ButtonType.YES, ButtonType.NO);
289         confirm.showAndWait();
290 
291         if (confirm.getResult() == ButtonType.YES) {
292             String chosenConfig = (String) configChooser.getValue();
293             if (chosenConfig != null) {
294                 Pair<JSONObject, String> result = ViewCtrl.domainCtrl.modifyConfiguration(chosenConfig, canEatHorizontally.isSelected(), canEatVertically.isSelected(), canEatDiagonally.isSelected());
295                 if (result.second != null) {
296                     switch (result.second) {
297                         case "ERR_CONFIGURATION_USED":
298                             modifyConfigResult.setText("This configuration has been already used in a game!");
299                             break;
300                         case "ERR_NOT_CREATOR":
301                             modifyConfigResult.setText("You are not the creator of this configuration!");
302                             break;
303                         case "ERR_INEXISTING_CONFIGURATION":
304                             modifyConfigResult.setText("This configuration doesn't exist!");
305                             break;
306                         case "ERR_INVALID_BOARD":
307                             modifyConfigResult.setText("The initial board is invalid!");
308                             break;
309                         case "ERR_INVALID_RULES":
310                             modifyConfigResult.setText("You must select at least one rule!");
311                             break;
312                         default:
313                             modifyConfigResult.setText("Something went wrong, try again!");
314                             break;
315                     }
316                 }
317                 else {
318                     configChooser.getItems().clear();
319                     initialize();
320                     configChooser.getSelectionModel().select(chosenConfig);
321                     modifyConfigResult.setText("Success!");
322                 }
323             }
324         }
325     }
```

6.16.3.13 deleteConfig()

`void view.ConfigModifyView.deleteConfig ( ) throws IOException`

Event method which is executed when the delete button is clicked.

Precondition

True

Postcondition

The current configuration is deleted.

Definition at line 332 of file ConfigModifyView.java.

```
                                             {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "This configuration will be deleted. Are you
    sure?", ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            String chosenConfig = (String) configChooser.getValue();
            if (chosenConfig != null) {
                String result = ViewCtrl.domainCtrl.deleteConfiguration(chosenConfig);
                if (result != null) {
                    switch (result) {
                        case "ERR_INEXISTING_CONFIGURATION":
                            modifyConfigResult.setText("This configuration doesn't exist!");
                            break;
                        case "ERR_NOT_CREATOR":
                            modifyConfigResult.setText("You are not the creator of this
    configuration!");
                            break;
                        case "ERR_CONFIGURATION_USED":
                            modifyConfigResult.setText("This configuration has been already used in a
    game!");
                            break;
                        default:
                            modifyConfigResult.setText("Something went wrong, try again!");
                            break;
                    }
                }
                else {
                    configChooser.getItems().clear();
                    canEatHorizontally.setSelected(false);
                    canEatVertically.setSelected(false);
                    canEatDiagonally.setSelected(false);
                    initialize();
                    modifyConfigResult.setText("Success!");
                }
            }
        }
    }
```

6.16.3.14 logOut()

```
void view.ConfigModifyView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 373 of file ConfigModifyView.java.

```
                                        {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.16.4 Member Data Documentation

6.16.4.1 user

```
Text view.ConfigModifyView.user  [private]
```

Menu User tab.

Definition at line 45 of file ConfigModifyView.java.

6.16.4.2 bots

```
Text view.ConfigModifyView.bots  [private]
```

Menu Bots tab.

Definition at line 50 of file ConfigModifyView.java.

6.16.4.3 config

```
Text view.ConfigModifyView.config  [private]
```

Menu Configuration tab.

Definition at line 55 of file ConfigModifyView.java.

6.16.4.4 games

```
Text view.ConfigModifyView.games  [private]
```

Menu Games tab.

Definition at line 60 of file ConfigModifyView.java.

6.16.4.5 ranking

```
Text view.ConfigModifyView.ranking  [private]
```

Menu Ranking tab.

Definition at line 65 of file ConfigModifyView.java.

6.16.4.6 play

```
Text view.ConfigModifyView.play  [private]
```

Menu Play tab.

Definition at line 70 of file ConfigModifyView.java.

6.16.4.7 createConfig

```
Text view.ConfigModifyView.createConfig  [private]
```

Configuration create button text.

Definition at line 75 of file ConfigModifyView.java.

6.16.4.8 createConfigButton

```
Rectangle view.ConfigModifyView.createConfigButton  [private]
```

Configuration create button.

Definition at line 80 of file ConfigModifyView.java.

6.16.4.9 modifyConfig

```
Text view.ConfigModifyView.modifyConfig  [private]
```

Configuration modify button text.

Definition at line 85 of file ConfigModifyView.java.

6.16.4.10 modifyConfigButton

```
Rectangle view.ConfigModifyView.modifyConfigButton  [private]
```

Configuration modify button.

Definition at line 90 of file ConfigModifyView.java.

6.16.4.11 consultConfig

```
Text view.ConfigModifyView.consultConfig [private]
```

Configuration consult button text.

Definition at line 95 of file ConfigModifyView.java.

6.16.4.12 consultConfigButton

```
Rectangle view.ConfigModifyView.consultConfigButton [private]
```

Configuration consult button.

Definition at line 100 of file ConfigModifyView.java.

6.16.4.13 canEatHorizontally

```
RadioButton view.ConfigModifyView.canEatHorizontally [private]
```

CanEatHorizontally selector.

Definition at line 105 of file ConfigModifyView.java.

6.16.4.14 canEatVertically

```
RadioButton view.ConfigModifyView.canEatVertically [private]
```

CanEatVertically selector.

Definition at line 110 of file ConfigModifyView.java.

6.16.4.15 canEatDiagonally

```
RadioButton view.ConfigModifyView.canEatDiagonally [private]
```

CanEatDiagonally selector.

Definition at line 115 of file ConfigModifyView.java.

6.16.4.16 modifyInitialBoard

```
Text view.ConfigModifyView.modifyInitialBoard  [private]
```

Modify initial board button text.

Definition at line 120 of file ConfigModifyView.java.

6.16.4.17 modifyInitialBoardButton

```
Rectangle view.ConfigModifyView.modifyInitialBoardButton  [private]
```

Modify initial board button.

Definition at line 125 of file ConfigModifyView.java.

6.16.4.18 modifyConfigConfirm

```
Text view.ConfigModifyView.modifyConfigConfirm  [private]
```

Configuration modify confirm button text.

Definition at line 130 of file ConfigModifyView.java.

6.16.4.19 modifyConfigConfirmButton

```
Rectangle view.ConfigModifyView.modifyConfigConfirmButton  [private]
```

Configuration modify confirm button.

Definition at line 135 of file ConfigModifyView.java.

6.16.4.20 deleteConfig

```
ImageView view.ConfigModifyView.deleteConfig  [private]
```

Configuration delete button image.

Definition at line 140 of file ConfigModifyView.java.

6.16.4.21 deleteConfigButton

```
Circle view.ConfigModifyView.deleteConfigButton  [private]
```

Configuration delete button.

Definition at line 145 of file ConfigModifyView.java.

6.16.4.22 configChooser

```
ChoiceBox view.ConfigModifyView.configChooser  [private]
```

Configuration choiceBox.

Definition at line 150 of file ConfigModifyView.java.

6.16.4.23 modifyConfigResult

```
Label view.ConfigModifyView.modifyConfigResult  [private]
```

Exception output message label.

Definition at line 155 of file ConfigModifyView.java.

6.16.4.24 currentUserName

```
Label view.ConfigModifyView.currentUserName  [private]
```

Current user name.

Definition at line 160 of file ConfigModifyView.java.

6.16.4.25 logOut

```
Text view.ConfigModifyView.logOut  [private]
```

LogOut button.

Definition at line 165 of file ConfigModifyView.java.

The documentation for this class was generated from the following file:

- ConfigModifyView.java

6.17 domain.Configuration Class Reference

Represents the rules of an Othello game including its name, whether the pieces can be eaten horizontally, vertically or diagonally, and its creator. By Alex Rodriguez.

Public Member Functions

- Configuration (String name, UUID creatorID, boolean canEatHorizontally, boolean canEatVertically, boolean canEatDiagonally)

  Create a Configuration instance.
- Configuration (JSONObject configuration)

  Create a Configuration instance from a JSONObject representation of a Configuration.
- JSONObject serialize ()

  Create a JSONObject representation of a Configuration from the implicit Configuration.
- String getName ()

  Get the name of the implicit Configuration.
- void setName (String name) throws InvalidNameException

  Set the name of the implicit Configuration.
- UUID getCreatorID ()

  Get the creatorID of the implicit Configuration.
- boolean getCanEatHorizontally ()

  Get the canEatHorizontally of the implicit Configuration.
- void setCanEatHorizontally (boolean canEatHorizontally) throws InvalidRulesException

  Set the canEatHorizontally of the implicit Configuration.
- boolean getCanEatVertically ()

  Get the canEatVertically of the implicit Configuration.
- void setCanEatVertically (boolean canEatVertically) throws InvalidRulesException

  Set the canEatVertically of the implicit Configuration.
- boolean getCanEatDiagonally ()

  Get the canEatDiagonally of the implicit Configuration.
- void setCanEatDiagonally (boolean canEatDiagonally) throws InvalidRulesException

  Set the canEatDiagonally of the implicit Configuration.

Private Attributes

- String name

  Name of the Configuration.
- UUID creatorID

  Player ID of the Configuration's creator.
- boolean canEatHorizontally

  Whether the pieces of a Game can be eaten horizontally.
- boolean canEatVertically

  Whether the pieces of a Game can be eaten vertically.
- boolean canEatDiagonally

  Whether the pieces of a Game can be eaten diagonally.

6.17.1 Detailed Description

Represents the rules of an Othello game including its name, whether the pieces can be eaten horizontally, vertically or diagonally, and its creator. By Alex Rodriguez.

Definition at line 21 of file Configuration.java.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Configuration() [1/2]

```
domain.Configuration.Configuration (
            String name,
            UUID creatorID,
            boolean canEatHorizontally,
            boolean canEatVertically,
            boolean canEatDiagonally )
```

Create a Configuration instance.

Precondition

True

Postcondition

A Configuration instance is created and its implicits name, creatorID, canEatHorizontally, canEatVertically and canEatDiagonally attributes are setted.

Parameters

name	Name of the Configuration.
creatorID	Player ID of the Configuration's creator.
canEatHorizontally	Whether the pieces of a Game can be eaten horizontally.
canEatVertically	Whether the pieces of a Game can be eaten vertically.
canEatDiagonally	Whether the pieces of a Game can be eaten diagonally.

Definition at line 58 of file Configuration.java.

```
                                  {
        this.name = name;
        this.creatorID = creatorID;
        this.canEatHorizontally = canEatHorizontally;
        this.canEatVertically = canEatVertically;
        this.canEatDiagonally = canEatDiagonally;
    }
```

6.17.2.2 Configuration() [2/2]

```
domain.Configuration.Configuration (
            JSONObject configuration )
```

Create a Configuration instance from a JSONObject representation of a Configuration.

Precondition

True

Postcondition

A Configuration instance is created and its implicits name, creatorID, canEatHorizontally, canEatVertically and canEatDiagonally are setted.

Parameters

configuration	JSONObject representation of a Configuration.

Definition at line 74 of file Configuration.java.

```
                                             {
        this.name = configuration.getString("name");
        this.creatorID = UUID.fromString(configuration.getString("creator_id"));
        this.canEatHorizontally = configuration.getBoolean("can_eat_horizontally");
        this.canEatVertically = configuration.getBoolean("can_eat_vertically");
        this.canEatDiagonally = configuration.getBoolean("can_eat_diagonally");
    }
```

6.17.3 Member Function Documentation

6.17.3.1 serialize()

```
JSONObject domain.Configuration.serialize ( )
```

Create a JSONObject representation of a Configuration from the implicit Configuration.

Precondition

True

Postcondition

A JSONObject representing the implicit Configuration is returned.

Returns

JSONObject representation of a Configuration.

Definition at line 90 of file Configuration.java.

```
                            {
        JSONObject configuration = new JSONObject();

        configuration.put("name", this.name);
        configuration.put("creator_id", this.creatorID.toString());
        configuration.put("can_eat_horizontally", this.canEatHorizontally);
        configuration.put("can_eat_vertically", this.canEatVertically);
        configuration.put("can_eat_diagonally", this.canEatDiagonally);

        return configuration;
    }
```

6.17.3.2 getName()

```
String domain.Configuration.getName ( )
```

Get the name of the implicit Configuration.

Precondition

True

Postcondition

The name attribute of the implicit Configuration is returned.

Returns

Name of the implicit Configuration.

Definition at line 108 of file Configuration.java.

```
                        {
        return this.name;
    }
```

6.17.3.3 setName()

```
void domain.Configuration.setName (
              String name ) throws InvalidNameException
```

Set the name of the implicit Configuration.

Precondition

True

Postcondition

The name attribute of the implicit Configuration is setted if it is not blank, otherwise an InvalidNameException is thrown.

Parameters

name	Name of the Configuration.

Definition at line 119 of file Configuration.java.

```
                                                                   {
        if (name.isBlank())
            throw new InvalidNameException();

        this.name = name;
    }
```

6.17.3.4 getCreatorID()

```
UUID domain.Configuration.getCreatorID ( )
```

Get the creatorID of the implicit Configuration.

Precondition

True

Postcondition

The creatorID attribute of the implicit Configuration is returned.

Returns

CreatorID of the implicit Configuration.

Definition at line 132 of file Configuration.java.

```
                                  {
        return this.creatorID;
    }
```

6.17.3.5 getCanEatHorizontally()

```
boolean domain.Configuration.getCanEatHorizontally ( )
```

Get the canEatHorizontally of the implicit Configuration.

Precondition

True

Postcondition

The canEatHorizontally attribute of the implicit Configuration is returned.

Returns

CanEatHorizontally of the implicit Configuration.

Definition at line 142 of file Configuration.java.

```
                                              {
        return this.canEatHorizontally;
    }
```

6.17.3.6 setCanEatHorizontally()

```
void domain.Configuration.setCanEatHorizontally (
            boolean canEatHorizontally ) throws InvalidRulesException
```

Set the canEatHorizontally of the implicit Configuration.

Precondition

True

Postcondition

The canEatHorizontally attribute of the implicit Configuration is setted if all the rules aren't false, otherwise an InvalidRulesException is thrown.

Parameters

canEatHorizontally	Whether the pieces of a Game can be eaten horizontally.

Definition at line 153 of file Configuration.java.

```
                                                                                    {
        if (canEatHorizontally == false && this.canEatVertically == false && this.canEatDiagonally ==
    false)
            throw new InvalidRulesException();

        this.canEatHorizontally = canEatHorizontally;
    }
```

6.17.3.7 getCanEatVertically()

```
boolean domain.Configuration.getCanEatVertically ( )
```

Get the canEatVertically of the implicit Configuration.

Precondition

True

Postcondition

The canEatVertically attribute of the implicit Configuration is returned.

Returns

CanEatVertically of the implicit Configuration.

Definition at line 166 of file Configuration.java.

```
                                            {
        return this.canEatVertically;
    }
```

6.17.3.8 setCanEatVertically()

```
void domain.Configuration.setCanEatVertically (
            boolean canEatVertically ) throws InvalidRulesException
```

Set the canEatVertically of the implicit Configuration.

Precondition

True

Postcondition

The canEatVertically attribute of the implicit Configuration is setted if all the rules aren't false, otherwise an InvalidRulesException is thrown.

Parameters

canEatVertically	Whether the pieces of a Game can be eaten vertically.

Definition at line 177 of file Configuration.java.

```
                                                                    {
        if (this.canEatHorizontally == false && canEatVertically == false && this.canEatDiagonally ==
    false)
            throw new InvalidRulesException();

        this.canEatVertically = canEatVertically;
    }
```

6.17.3.9 getCanEatDiagonally()

```
boolean domain.Configuration.getCanEatDiagonally ( )
```

Get the canEatDiagonally of the implicit Configuration.

Precondition

True

Postcondition

The canEatDiagonally attribute of the implicit Configuration is returned.

Returns

CanEatDiagonally of the implicit Configuration.

Definition at line 190 of file Configuration.java.

```
                                    {
        return this.canEatDiagonally;
    }
```

6.17.3.10 setCanEatDiagonally()

```
void domain.Configuration.setCanEatDiagonally (
            boolean canEatDiagonally ) throws InvalidRulesException
```

Set the canEatDiagonally of the implicit Configuration.

Precondition

True

Postcondition

The canEatDiagonally attribute of the implicit Configuration is setted if all the rules aren't false, otherwise an InvalidRulesException is thrown.

Parameters

canEatDiagonally	Whether the pieces of a Game can be eaten diagonally.

Definition at line 201 of file Configuration.java.

```
                                                                              {
        if (this.canEatHorizontally == false && this.canEatVertically == false && canEatDiagonally ==
    false)
            throw new InvalidRulesException();

        this.canEatDiagonally = canEatDiagonally;
    }
```

6.17.4 Member Data Documentation

6.17.4.1 name

```
String domain.Configuration.name  [private]
```

Name of the Configuration.

Definition at line 27 of file Configuration.java.

6.17.4.2 creatorID

```
UUID domain.Configuration.creatorID  [private]
```

Player ID of the Configuration's creator.

Definition at line 31 of file Configuration.java.

6.17.4.3 canEatHorizontally

```
boolean domain.Configuration.canEatHorizontally  [private]
```

Whether the pieces of a Game can be eaten horizontally.

Definition at line 35 of file Configuration.java.

6.17.4.4 canEatVertically

```
boolean domain.Configuration.canEatVertically  [private]
```

Whether the pieces of a Game can be eaten vertically.

Definition at line 39 of file Configuration.java.

6.17.4.5 canEatDiagonally

```
boolean domain.Configuration.canEatDiagonally  [private]
```

Whether the pieces of a Game can be eaten diagonally.

Definition at line 43 of file Configuration.java.

The documentation for this class was generated from the following file:

- Configuration.java

6.18 cmd.driver.configuration Class Reference

Configuration driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)
  Configuration driver main function. Creates an instance of the Configuration driver and starts it.

6.18.1 Detailed Description

Configuration driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file configuration.java.

6.18.2 Member Function Documentation

6.18.2.1 main()

```
static void cmd.driver.configuration.main (
             String[] args )  [static]
```

Configuration driver main function. Creates an instance of the Configuration driver and starts it.

Precondition

True.

Postcondition

The Configuration driver has started.

Definition at line 22 of file configuration.java.

```
                                              {
        new ConfigurationDriver().start();
    }
```

The documentation for this class was generated from the following file:

- configuration.java

6.19 domain.ConfigurationCtrl Class Reference

Configuration domain sub-controller. It communicates with the main domain controller, the configuration repository controller and the game repository controller for certain integrity checks. It is also in charge of retrieving the initial boards associated with the configurations.

Public Member Functions

- ConfigurationCtrl ()

    Creator method that creates an instance of Configuration Controller.
- Configuration create (String name, Boolean canEatHorizontally, Boolean canEatVertically, Boolean can↩EatDiagonally, Board initialBoard, UUID creatorID) throws InvalidNameException, ExistingConfiguration↩Exception, InvalidBoardException, InvalidRulesException

    Lets the current user create a new configuration with a name, rules and the initial board.
- Configuration modify (String name, Boolean canEatHorizontally, Boolean canEatVertically, Boolean can↩EatDiagonally, Board initialBoard, UUID modificatorID) throws NotCreatorException, ConfigurationUsed↩Exception, InvalidBoardException, InvalidRulesException, InexistingConfigurationException

    Lets the current user modify a configuration he/she created. The configuration's rules and the initial board can be changed.
- void delete (String name, UUID deleterID) throws NotCreatorException, ConfigurationUsedException, InexistingConfigurationException

    Lets the current user delete a configuration he/she created.
- Configuration getConfiguration (String name) throws InexistingConfigurationException

    Returns the configuration identified by the name.
- Board getInitialBoard (String name) throws InexistingConfigurationException

    Returns the initial board associated with the given configuration name.
- ArrayList< String > list ()

    Returns a list of all configurations names in the system.

Private Member Functions

- Configuration save (String name, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEat↩ Diagonally, Board initialBoard, UUID creatorID) throws InvalidBoardException, InvalidRulesException

    Method that, given a name, a set of rules and an initial board, allows us to save a configuration in the repository.

Private Attributes

- ConfigurationRepositoryCtrl repositoryCtrl

    Configuration repository controller.
- GameRepositoryCtrl gameRepositoryCtrl

    Game repository controller.

6.19.1 Detailed Description

Configuration domain sub-controller. It communicates with the main domain controller, the configuration repository controller and the game repository controller for certain integrity checks. It is also in charge of retrieving the initial boards associated with the configurations.

By Alex Rodriguez.

See also

domain.Configuration

Definition at line 31 of file ConfigurationCtrl.java.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 ConfigurationCtrl()

```
domain.ConfigurationCtrl.ConfigurationCtrl ( )
```

Creator method that creates an instance of Configuration Controller.

Precondition

True

Postcondition

An instance of ConfigirationCtrl is created.

Definition at line 50 of file ConfigurationCtrl.java.

```
                                 {
         this.repositoryCtrl = new ConfigurationRepositoryCtrl();
         this.gameRepositoryCtrl = new GameRepositoryCtrl();
     }
```

6.19.3 Member Function Documentation

6.19.3.1 create()

```
Configuration domain.ConfigurationCtrl.create (
            String name,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally,
            Board initialBoard,
            UUID creatorID ) throws InvalidNameException, ExistingConfigurationException,
InvalidBoardException, InvalidRulesException
```

Lets the current user create a new configuration with a name, rules and the initial board.

Precondition

canEatHorizontally, vertically and diagonally aren't null

Postcondition

The created Configuration is returned if no exception is thrown. Else, an exception will be thrown

Parameters

name	name of a Configuration
canEatHorizontally	Boolean that represents if you can capture pieces in a horizontal manner.
canEatVertically	Boolean that represents if you can capture pieces in a vertical manner.
canEatDiagonally	Boolean that represents if you can capture pieces in a diagonal manner.
initialBoard	Instance of a Board
creatorID	UUID of the creator.

Returns

Configuration.

Definition at line 69 of file ConfigurationCtrl.java.

```
                {
        if (name.isBlank())
            throw new InvalidNameException();

        if (this.repositoryCtrl.getConfiguration(name) != null)
            throw new ExistingConfigurationException();

        return this.save(name, canEatHorizontally, canEatVertically, canEatDiagonally, initialBoard,
    creatorID);
    }
```

6.19.3.2 modify()

```
Configuration domain.ConfigurationCtrl.modify (
              String name,
              Boolean canEatHorizontally,
              Boolean canEatVertically,
              Boolean canEatDiagonally,
              Board initialBoard,
              UUID modificatorID ) throws NotCreatorException, ConfigurationUsedException,
InvalidBoardException, InvalidRulesException, InexistingConfigurationException
```

Lets the current user modify a configuration he/she created. The configuration's rules and the initial board can be changed.

Precondition

True

Postcondition

Modified Configuration is returned if no exception is thrown. Else, an exception will be thrown

Parameters

name	name of a Configuration
canEatHorizontally	Boolean that represents if you can capture pieces in a horizontal manner.
canEatVertically	Boolean that represents if you can capture pieces in a vertical manner.
canEatDiagonally	Boolean that represents if you can capture pieces in a diagonal manner.
initialBoard	Instance of a Board
modificatorID	Modifier Player UUID

Returns

Configuration

Definition at line 93 of file ConfigurationCtrl.java.

```
                  {
          Configuration original = this.getConfiguration(name);
          Board originalInitialBoard = this.getInitialBoard(name);

          if (!original.getCreatorID().equals(modificatorID))
               throw new NotCreatorException();

           if (this.gameRepositoryCtrl.existsGameByConfigurationName(name))
               throw new ConfigurationUsedException();

           if (canEatHorizontally != null)
               original.setCanEatHorizontally(canEatHorizontally);

           if (canEatVertically != null)
               original.setCanEatVertically(canEatVertically);

           if (canEatDiagonally != null)
               original.setCanEatDiagonally(canEatDiagonally);

           if (initialBoard != null)
               originalInitialBoard = initialBoard;

           return this.save(original.getName(), original.getCanEatHorizontally(),
       original.getCanEatVertically(),
                   original.getCanEatDiagonally(), originalInitialBoard, original.getCreatorID());
       }
```

6.19.3.3 save()

```
Configuration domain.ConfigurationCtrl.save (
              String name,
              Boolean canEatHorizontally,
              Boolean canEatVertically,
              Boolean canEatDiagonally,
              Board initialBoard,
              UUID creatorID ) throws InvalidBoardException, InvalidRulesException  [private]
```

Method that, given a name, a set of rules and an initial board, allows us to save a configuration in the repository.

Precondition

name and creatorID aren't null.

Postcondition

Saved Configuration is returned if no exception is thrown. Else, an exception will be thrown.

Parameters

name	name of a Configuration
canEatHorizontally	Boolean that represents if you can capture pieces in a horizontal manner.
canEatVertically	Boolean that represents if you can capture pieces in a vertical manner.
canEatDiagonally	Boolean that represents if you can capture pieces in a diagonal manner.
initialBoard	Instance of a Board
creatorID	UUID of the creator.

Returns

Configuration

Definition at line 133 of file ConfigurationCtrl.java.

```
                                                                {
        if (initialBoard == null)
            throw new InvalidBoardException();

        if (canEatHorizontally == false && canEatVertically == false && canEatDiagonally == false)
            throw new InvalidRulesException();

        Configuration configuration = new Configuration(name, creatorID, canEatHorizontally,
    canEatVertically,
                canEatDiagonally);

        this.repositoryCtrl.save(configuration.serialize(), initialBoard.serialize());
        return configuration;
    }
```

6.19.3.4 delete()

```
void domain.ConfigurationCtrl.delete (
            String name,
            UUID deleterID ) throws NotCreatorException, ConfigurationUsedException, InexistingConfigurationE
```

Lets the current user delete a configuration he/she created.

Precondition

True

Postcondition

The Configuration is deleted if no exception is thrown. Else, an exception will be thrown.

Parameters

name	Name of a Configuration
deleterID	UUID of a Player.

Definition at line 156 of file ConfigurationCtrl.java.

```
                                                                                    {
        Configuration original = this.getConfiguration(name);

        if (!original.getCreatorID().equals(deleterID))
            throw new NotCreatorException();

        if (this.gameRepositoryCtrl.existsGameByConfigurationName(name))
            throw new ConfigurationUsedException();

        this.repositoryCtrl.delete(name);
    }
```

6.19.3.5 getConfiguration()

```
Configuration domain.ConfigurationCtrl.getConfiguration (
            String name ) throws InexistingConfigurationException
```

Returns the configuration identified by the name.

Precondition

True

Postcondition

The Configuration identified by name is returned if no exception is thrown. Else, an exception will be thrown.

Parameters

name	Name of a Configuration

Returns

Configuration

Definition at line 176 of file ConfigurationCtrl.java.
```
                                                                                    {
        JSONObject rawConfiguration = this.repositoryCtrl.getConfiguration(name);
        if (rawConfiguration == null)
            throw new InexistingConfigurationException();

        return new Configuration(rawConfiguration);
    }
```

6.19.3.6 getInitialBoard()

```
Board domain.ConfigurationCtrl.getInitialBoard (
            String name ) throws InexistingConfigurationException
```

Returns the initial board associated with the given configuration name.

Precondition

True

Postcondition

The initial board associated with the given configuration name is returned if no exception is returned. Else, an exception will be returned.

Parameters

name	Name of a Configuration.

Returns

Board

Definition at line 191 of file ConfigurationCtrl.java.
```
                                                                              {
        JSONObject rawInitialBoard = this.repositoryCtrl.getBoard(name);
        if (rawInitialBoard == null)
            throw new InexistingConfigurationException();

        return new Board(rawInitialBoard);
    }
```

6.19.3.7 list()

```
ArrayList<String> domain.ConfigurationCtrl.list ( )
```

Returns a list of all configurations names in the system.

Precondition

True/em>

Postcondition

ArrayList of Strings with the names of all the Configurations in the system

Returns

ArrayList<String>

Definition at line 205 of file ConfigurationCtrl.java.

```
                                {
        return this.repositoryCtrl.listConfigurations();
    }
```

6.19.4 Member Data Documentation

6.19.4.1 repositoryCtrl

`ConfigurationRepositoryCtrl domain.ConfigurationCtrl.repositoryCtrl [private]`

Configuration repository controller.

Definition at line 37 of file ConfigurationCtrl.java.

6.19.4.2 gameRepositoryCtrl

`GameRepositoryCtrl domain.ConfigurationCtrl.gameRepositoryCtrl [private]`

Game repository controller.

Definition at line 41 of file ConfigurationCtrl.java.

The documentation for this class was generated from the following file:

- ConfigurationCtrl.java

6.20 test.driver.ConfigurationDriver Class Reference

Implements the different options for the Configuration driver application. By Alex Rodriguez.

Public Member Functions

- ConfigurationDriver ()
- void start ()

Public Attributes

- Configuration currentConfiguration

Private Member Functions

- void mainMenu ()
- void create ()
- void getName ()
- void setName ()
- void getCreatorID ()
- void getCanEatHorizontally ()
- void setCanEatHorizontally ()
- void getCanEatVertically ()
- void setCanEatVertically ()
- void getCanEatDiagonally ()
- void setCanEatDiagonally ()
- void serialize ()
- void deserialize ()

Additional Inherited Members

6.20.1 Detailed Description

Implements the different options for the Configuration driver application. By Alex Rodriguez.

Definition at line 18 of file ConfigurationDriver.java.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 ConfigurationDriver()

```
test.driver.ConfigurationDriver.ConfigurationDriver ( )
```

Definition at line 25 of file ConfigurationDriver.java.

```
                                {
        this.currentConfiguration = null;
    }
```

6.20.3 Member Function Documentation

6.20.3.1 start()

`void test.driver.ConfigurationDriver.start ( )`

Definition at line 31 of file ConfigurationDriver.java.

```
                        {
        while (true) {
            this.mainMenu();
        }
    }
```

6.20.3.2 mainMenu()

`void test.driver.ConfigurationDriver.mainMenu ( ) [private]`

Definition at line 37 of file ConfigurationDriver.java.

```
                           {
        String title = (this.currentConfiguration != null
                ? String.format("Current: %s\n", this.currentConfiguration.getName())
                : null);
        switch (Driver.menu(title, "Configuration Driver",
                new Pair<String, String>("1", "Create Configuration"),
                new Pair<String, String>("2", "Get name"),
                new Pair<String, String>("3", "Set name"),
                new Pair<String, String>("4", "Get creatorID"),
                new Pair<String, String>("5", "Get canEatHorizontally"),
                new Pair<String, String>("6", "Set canEatHorizontally"),
                new Pair<String, String>("7", "Get canEatVertically"),
                new Pair<String, String>("8", "Set canEatVertically"),
                new Pair<String, String>("9", "Get canEatDiagonally"),
                new Pair<String, String>("10", "Set canEatDiagonally"),
                new Pair<String, String>("11", "Serialize to JSON"),
                new Pair<String, String>("12", "Deserialize from JSON"))) {
        case "1":
            this.create();
            break;
        case "2":
            this.getName();
            break;
        case "3":
            this.setName();
            break;
        case "4":
            this.getCreatorID();
            break;
        case "5":
            this.getCanEatHorizontally();
            break;
        case "6":
            this.setCanEatHorizontally();
            break;
        case "7":
            this.getCanEatVertically();
            break;
        case "8":
            this.setCanEatVertically();
            break;
        case "9":
            this.getCanEatDiagonally();
            break;
        case "10":
            this.setCanEatDiagonally();
            break;
        case "11":
            this.serialize();
            break;
        case "12":
            this.deserialize();
            break;
        }
        Driver.pause();
    }
```

6.20.3.3 create()

void test.driver.ConfigurationDriver.create ( ) [private]

Definition at line 94 of file ConfigurationDriver.java.

```
                         {
        System.out.println(
                "Take into account that UUIDs will be randomly generated because typing them in will be a hassle.\n");
        String name = Driver.input("Name?");
        boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
        boolean canEatVertically = Driver.inputBool("Can eat vertically?");
        boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
        try {
            Configuration configuration = new Configuration("Default name", UUID.randomUUID(), true, true, true);
            configuration.setName(name);
            configuration.setCanEatHorizontally(canEatHorizontally);
            configuration.setCanEatVertically(canEatVertically);
            configuration.setCanEatDiagonally(canEatDiagonally);
            this.currentConfiguration = configuration;
            System.out.println(String.format("%s created successfully!", this.currentConfiguration.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s", e.getMessage()));
        }
    }
```

6.20.3.4 getName()

void test.driver.ConfigurationDriver.getName ( ) [private]

Definition at line 114 of file ConfigurationDriver.java.

```
                          {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        System.out.println(String.format("%s's name is: %s", this.currentConfiguration.getName(),
                this.currentConfiguration.getName()));
    }
```

6.20.3.5 setName()

void test.driver.ConfigurationDriver.setName ( ) [private]

Definition at line 124 of file ConfigurationDriver.java.

```
                          {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        try {
            this.currentConfiguration.setName(Driver.input("Name?"));
            System.out.println(String.format("%s's name changed successfully!", this.currentConfiguration.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s", e.getMessage()));
        }
    }
```

6.20.3.6 getCreatorID()

void test.driver.ConfigurationDriver.getCreatorID ( ) [private]

Definition at line 138 of file ConfigurationDriver.java.

```
                                 {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        System.out.println(String.format("%s's creatorID is: %s", this.currentConfiguration.getName(),
                this.currentConfiguration.getCreatorID()));
    }
```

6.20.3.7 getCanEatHorizontally()

void test.driver.ConfigurationDriver.getCanEatHorizontally ( ) [private]

Definition at line 148 of file ConfigurationDriver.java.

```
                                          {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        System.out.println(String.format("%s's canEatHorizontally is: %s",
      this.currentConfiguration.getName(),
                this.currentConfiguration.getCanEatHorizontally()));
    }
```

6.20.3.8 setCanEatHorizontally()

void test.driver.ConfigurationDriver.setCanEatHorizontally ( ) [private]

Definition at line 158 of file ConfigurationDriver.java.

```
                                          {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        try {
            this.currentConfiguration.setCanEatHorizontally(Driver.inputBool("Can eat horizontally?"));
            System.out.println(String.format("%s's canEatHorizontally changed successfully!",
                    this.currentConfiguration.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
      e.getMessage()));
        }
    }
```

6.20.3.9 getCanEatVertically()

void test.driver.ConfigurationDriver.getCanEatVertically ( ) [private]

Definition at line 173 of file ConfigurationDriver.java.

```
                                        {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        System.out.println(String.format("%s's canEatVertically is: %s",
      this.currentConfiguration.getName(),
                this.currentConfiguration.getCanEatVertically()));
    }
```

6.20.3.10 setCanEatVertically()

```
void test.driver.ConfigurationDriver.setCanEatVertically ( ) [private]
```

Definition at line 183 of file ConfigurationDriver.java.

```
                                     {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        try {
            this.currentConfiguration.setCanEatVertically(Driver.inputBool("Can eat vertically?"));
            System.out.println(
                    String.format("%s’s canEatVertically changed successfully!",
    this.currentConfiguration.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
    e.getMessage()));
        }
    }
```

6.20.3.11 getCanEatDiagonally()

```
void test.driver.ConfigurationDriver.getCanEatDiagonally ( ) [private]
```

Definition at line 198 of file ConfigurationDriver.java.

```
                                     {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        System.out.println(String.format("%s’s canEatDiagonally is: %s",
    this.currentConfiguration.getName(),
                this.currentConfiguration.getCanEatDiagonally()));
    }
```

6.20.3.12 setCanEatDiagonally()

```
void test.driver.ConfigurationDriver.setCanEatDiagonally ( ) [private]
```

Definition at line 208 of file ConfigurationDriver.java.

```
                                     {
        if (this.currentConfiguration == null) {
            System.out.println("No current Configuration!");
            return;
        }

        try {
            this.currentConfiguration.setCanEatDiagonally(Driver.inputBool("Can eat diagonally?"));
            System.out.println(
                    String.format("%s’s canEatDiagonally changed successfully!",
    this.currentConfiguration.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
    e.getMessage()));
        }
    }
```

6.20.3.13 serialize()

```
void test.driver.ConfigurationDriver.serialize ( )  [private]
```

Definition at line 223 of file ConfigurationDriver.java.

```
                                  {
            if (this.currentConfiguration == null) {
                System.out.println("No current Configuration!");
                return;
            }

            System.out.println(String.format("%s's serialized to JSON is: %s", this.currentConfiguration.getName(),
                    this.currentConfiguration.serialize().toString(2)));
        }
```

6.20.3.14 deserialize()

```
void test.driver.ConfigurationDriver.deserialize ( )  [private]
```

Definition at line 233 of file ConfigurationDriver.java.

```
                                    {
            if (this.currentConfiguration == null) {
                System.out.println("No current Configuration!");
                return;
            }

            System.out.println(this.currentConfiguration.serialize().toString(2));
            this.currentConfiguration = new Configuration(this.currentConfiguration.serialize());
            System.out.println(String.format("\n%s's deserialized from the above JSON successfully!\n",
                    this.currentConfiguration.getName()));
            System.out.println(String.format("name:\t\t\t%s", this.currentConfiguration.getName()));
            System.out.println(String.format("creatorID:\t\t%s", this.currentConfiguration.getCreatorID()));
            System.out.println(String.format("canEatHorizontally:\t%s", this.currentConfiguration.getCanEatHorizontally()));
            System.out.println(String.format("canEatVertically:\t%s", this.currentConfiguration.getCanEatVertically()));
            System.out.println(String.format("canEatDiagonally:\t%s", this.currentConfiguration.getCanEatDiagonally()));
        }
```

6.20.4 Member Data Documentation

6.20.4.1 currentConfiguration

```
Configuration test.driver.ConfigurationDriver.currentConfiguration
```

Definition at line 21 of file ConfigurationDriver.java.

The documentation for this class was generated from the following file:

- ConfigurationDriver.java

6.21 repository.ConfigurationRepository Class Reference

Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.

Public Member Functions

- ConfigurationRepository ()

    Create a ConfigurationRepository instance.
- void save (JSONObject configuration, JSONObject board)

    Save a Configuration into the configuration database.
- void delete (String name)

    Delete a Configuration by name from the configuration database.
- JSONObject getConfiguration (String name)

    Get the Configuration by name from the configuration database or null if it does not exist.
- JSONObject getBoard (String name)

    Get the initial Board of a Configuration by name from the configuration database or null if it does not exist.
- ArrayList< String > listConfigurations ()

    List all Configurations of the configuration database.

Additional Inherited Members

6.21.1 Detailed Description

Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.

See also

repository.Repository

Definition at line 18 of file ConfigurationRepository.java.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 ConfigurationRepository()

```
repository.ConfigurationRepository.ConfigurationRepository ( )
```

Create a ConfigurationRepository instance.

Precondition

The Configuration repository JSON files exists.

Postcondition

A ConfigurationRepository instance is created.

Definition at line 28 of file ConfigurationRepository.java.

```
                                   {
        super(RepositoryType.CONFIGURATION);
    }
```

6.21.3 Member Function Documentation

6.21.3.1 save()

```
void repository.ConfigurationRepository.save (
            JSONObject configuration,
            JSONObject board )
```

Save a Configuration into the configuration database.

Precondition

The Configuration repository JSON files exists.

Postcondition

The Configuration and its initial Board are saved into the configuration database.

Parameters

configuration	Configuration to be saved.
board	Initial Board of the Configuration to be saved.

Definition at line 41 of file ConfigurationRepository.java.

```
                                                          {
        String name = configuration.getString("name");
        configuration.put("board", board);
        this.createOrUpdate(name, configuration);
    }
```

6.21.3.2 delete()

```
void repository.ConfigurationRepository.delete (
            String name )
```

Delete a Configuration by name from the configuration database.

Precondition

The Configuration repository JSON files exists.

Postcondition

The Configuration and its initial Board are deleted from the configuration database by name.

Parameters

name	Name of the Configuration to be deleted.

Definition at line 53 of file ConfigurationRepository.java.

```
                                {
        this.remove(name);
    }
```

6.21.3.3 getConfiguration()

```
JSONObject repository.ConfigurationRepository.getConfiguration (
            String name )
```

Get the Configuration by name from the configuration database or null if it does not exist.

Precondition

The Configuration repository JSON files exists.

Postcondition

A JSONObject representing the Configuration by name from the configuration database is returned or null if it does not exist.

Parameters

name	Name of the Configuration to be getted.

Returns

JSONObject that represents the Configuration by name from the configuration database or null if it does not exist.

Definition at line 64 of file ConfigurationRepository.java.

```
                                             {
        JSONObject configuration = this.get(name);
        if (configuration == null)
            return null;

        configuration.remove("board");
        return configuration;
    }
```

6.21.3.4 getBoard()

```
JSONObject repository.ConfigurationRepository.getBoard (
            String name )
```

Get the initial Board of a Configuration by name from the configuration database or null if it does not exist.

Precondition

The Configuration repository JSON files exists.

Postcondition

A JSONObject representing the initial Board of a Configuration by name from the configuration database is returned or null if it does not exist.

Parameters

name	Name of the initial Board's Configuration to be getted.

Returns

JSONObject that represents the initial Board of a Configuration by name from the configuration database or null if it does not exist.

Definition at line 80 of file ConfigurationRepository.java.

```
                                                {
        JSONObject configuration = this.get(name);
        if (configuration == null)
            return null;

        return configuration.getJSONObject("board");
    }
```

6.21.3.5 listConfigurations()

`ArrayList<String> repository.ConfigurationRepository.listConfigurations ( )`

List all Configurations of the configuration database.

Precondition

The Configuration repository JSON files exists.

Postcondition

An ArrayList containing the Configuration names of the configuration database is returned.

Returns

ArrayList of the Configuration names of the configuration database.

Definition at line 94 of file ConfigurationRepository.java.

```
                                                      {
        return new ArrayList<String>(this.list().keySet());
    }
```

The documentation for this class was generated from the following file:

- ConfigurationRepository.java

6.22 repository.ConfigurationRepositoryCtrl Class Reference

Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.

Public Member Functions

- ConfigurationRepositoryCtrl ()
  Create a ConfigurationRepositoryCtrl instance.
- void save (JSONObject configuration, JSONObject board)
  Save a Configuration into the configuration database.
- void delete (String name)
  Delete a Configuration by name from the configuration database.
- JSONObject getConfiguration (String name)
  Get the Configuration by name from the configuration database or null if it does not exist.
- JSONObject getBoard (String name)
  Get the initial Board of a Configuration by name from the configuration database or null if it does not exist.
- ArrayList< String > listConfigurations ()
  List all Configurations of the configuration database.

Private Attributes

- ConfigurationRepository repository
  ConfigurationRepository instance.

6.22.1 Detailed Description

Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.

See also

repository.ConfigurationRepository

Definition at line 18 of file ConfigurationRepositoryCtrl.java.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 ConfigurationRepositoryCtrl()

```
repository.ConfigurationRepositoryCtrl.ConfigurationRepositoryCtrl ( )
```

Create a ConfigurationRepositoryCtrl instance.

Precondition

The Configuration repository JSON files exists.

Postcondition

A ConfigurationRepositoryCtrl instance is created.

Definition at line 33 of file ConfigurationRepositoryCtrl.java.

```
                                        {
        this.repository = new ConfigurationRepository();
    }
```

6.22.3 Member Function Documentation

6.22.3.1 save()

```
void repository.ConfigurationRepositoryCtrl.save (
            JSONObject configuration,
            JSONObject board )
```

Save a Configuration into the configuration database.

Precondition

The Configuration repository JSON files exists.

Postcondition

The Configuration and its initial Board are saved into the configuration database.

Parameters

configuration	Configuration to be saved.
board	Initial Board of the Configuration to be saved.

Definition at line 46 of file ConfigurationRepositoryCtrl.java.

```
46                                                            {
47          this.repository.save(configuration, board);
48      }
```

6.22.3.2 delete()

```
void repository.ConfigurationRepositoryCtrl.delete (
            String name )
```

Delete a Configuration by name from the configuration database.

Precondition

The Configuration repository JSON files exists.

Postcondition

The Configuration and its initial Board are deleted from the configuration database by name.

Parameters

name	Name of the Configuration to be deleted.

Definition at line 56 of file ConfigurationRepositoryCtrl.java.

```
                                {
        this.repository.delete(name);
    }
```

6.22.3.3 getConfiguration()

```
JSONObject repository.ConfigurationRepositoryCtrl.getConfiguration (
            String name )
```

Get the Configuration by name from the configuration database or null if it does not exist.

Precondition

The Configuration repository JSON files exists.

Postcondition

A JSONObject representing the Configuration by name from the configuration database is returned or null if it does not exist.

Parameters

name	Name of the Configuration to be getted.

Returns

JSONObject that represents the Configuration by name from the configuration database or null if it does not exist.

Definition at line 67 of file ConfigurationRepositoryCtrl.java.

```
                                               {
        return this.repository.getConfiguration(name);
    }
```

6.22.3.4 getBoard()

```
JSONObject repository.ConfigurationRepositoryCtrl.getBoard (
            String name )
```

Get the initial Board of a Configuration by name from the configuration database or null if it does not exist.

Precondition

The Configuration repository JSON files exists.

Postcondition

A JSONObject representing the initial Board of a Configuration by name from the configuration database is returned or null if it does not exist.

Parameters

name	Name of the initial Board's Configuration to be getted.

Returns

JSONObject that represents the initial Board of a Configuration by name from the configuration database or null if it does not exist.

Definition at line 78 of file ConfigurationRepositoryCtrl.java.

```
                                           {
        return this.repository.getBoard(name);
    }
```

6.22.3.5 listConfigurations()

`ArrayList<String> repository.ConfigurationRepositoryCtrl.listConfigurations ( )`

List all Configurations of the configuration database.

Precondition

The Configuration repository JSON files exists.

Postcondition

An ArrayList containing the Configuration names of the configuration database is returned.

Returns

ArrayList of the Configuration names of the configuration database.

Definition at line 88 of file ConfigurationRepositoryCtrl.java.

```
                                                  {
        return this.repository.listConfigurations();
    }
```

6.22.4 Member Data Documentation

6.22.4.1 repository

```
ConfigurationRepository repository.ConfigurationRepositoryCtrl.repository  [private]
```

ConfigurationRepository instance.

Definition at line 24 of file ConfigurationRepositoryCtrl.java.

The documentation for this class was generated from the following file:

- ConfigurationRepositoryCtrl.java

6.23 domain.Exceptions.ConfigurationUsedException Class Reference

A configuration cannot be modified or deleted if it is already used in a game. By Alex Rodriguez.

Public Member Functions

- ConfigurationUsedException ()

6.23.1 Detailed Description

A configuration cannot be modified or deleted if it is already used in a game. By Alex Rodriguez.

Definition at line 140 of file Exceptions.java.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 ConfigurationUsedException()

```
domain.Exceptions.ConfigurationUsedException.ConfigurationUsedException ( )
```

Definition at line 141 of file Exceptions.java.

```
                                    {
            super("ERR_CONFIGURATION_USED");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.24 view.ConfigView Class Reference

Public Member Functions

- ConfigView ()

  Class creator.
- void initialize () throws Exception

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void createConfig () throws IOException

  Event method which is executed when the createConfig button is clicked.
- void modifyConfig () throws IOException

  Event method which is executed when the modifyConfig button is clicked.
- void consultConfig () throws IOException

  Event method which is executed when the consultConfig button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createConfig

  Configuration create button text.
- Rectangle createConfigButton

  Configuration create button.
- Text modifyConfig

  Configuration modify button text.
- Rectangle modifyConfigButton

  Configuration modify button.
- Text consultConfig

BConfiguration consult button text.

- Rectangle consultConfigButton

  Configuration consult button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.24.1 Detailed Description

This class represents the scene controller of the Configuration Menu .

Done by Arnau Pujantell

Definition at line 22 of file ConfigView.java.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 ConfigView()

```
view.ConfigView.ConfigView ( )
```

Class creator.

Definition at line 29 of file ConfigView.java.

```
29                              {
30     }
```

6.24.3 Member Function Documentation

6.24.3.1 initialize()

```
void view.ConfigView.initialize ( ) throws Exception
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 112 of file ConfigView.java.

```
112                                          {
113         currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
114     }
```

6.24.3.2 user()

```
void view.ConfigView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 121 of file ConfigView.java.

```
                                    {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.24.3.3 bots()

```
void view.ConfigView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 130 of file ConfigView.java.

```
                                    {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.24.3.4 games()

```
void view.ConfigView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 139 of file ConfigView.java.

```
                                     {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.24.3.5 ranking()

```
void view.ConfigView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 149 of file ConfigView.java.

```
                                          {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.24.3.6 play()

```
void view.ConfigView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 158 of file ConfigView.java.

```
                                       {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.24.3.7 createConfig()

```
void view.ConfigView.createConfig ( ) throws IOException
```

Event method which is executed when the createConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigCreateView.

Definition at line 167 of file ConfigView.java.

```
                                               {
        ViewCtrl.changeScene("template/ConfigCreateView.fxml");
    }
```

6.24.3.8 modifyConfig()

`void view.ConfigView.modifyConfig ( ) throws IOException`

Event method which is executed when the modifyConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigModifyView.

Definition at line 176 of file ConfigView.java.

```
                                                {
        ViewCtrl.changeScene("template/ConfigModifyView.fxml");
    }
```

6.24.3.9 consultConfig()

`void view.ConfigView.consultConfig ( ) throws IOException`

Event method which is executed when the consultConfig button is clicked.

Precondition

True

Postcondition

Scene changes to ConfigConsultView.

Definition at line 185 of file ConfigView.java.

```
                                                 {
        ViewCtrl.changeScene("template/ConfigConsultView.fxml");
    }
```

6.24.3.10 logOut()

`void view.ConfigView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 194 of file ConfigView.java.

```
                                          {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.24.4 Member Data Documentation

6.24.4.1 user

```
Text view.ConfigView.user  [private]
```

Menu User tab.

Definition at line 38 of file ConfigView.java.

6.24.4.2 bots

```
Text view.ConfigView.bots  [private]
```

Menu Bots tab.

Definition at line 43 of file ConfigView.java.

6.24.4.3 config

```
Text view.ConfigView.config  [private]
```

Menu Configuration tab.

Definition at line 48 of file ConfigView.java.

6.24.4.4 games

```
Text view.ConfigView.games  [private]
```

Menu Games tab.

Definition at line 53 of file ConfigView.java.

6.24.4.5 ranking

```
Text view.ConfigView.ranking  [private]
```

Menu Ranking tab.

Definition at line 58 of file ConfigView.java.

6.24.4.6 play

```
Text view.ConfigView.play  [private]
```

Menu Play tab.

Definition at line 63 of file ConfigView.java.

6.24.4.7 createConfig

```
Text view.ConfigView.createConfig  [private]
```

Configuration create button text.

Definition at line 68 of file ConfigView.java.

6.24.4.8 createConfigButton

```
Rectangle view.ConfigView.createConfigButton  [private]
```

Configuration create button.

Definition at line 73 of file ConfigView.java.

6.24.4.9 modifyConfig

```
Text view.ConfigView.modifyConfig  [private]
```

Configuration modify button text.

Definition at line 78 of file ConfigView.java.

6.24.4.10 modifyConfigButton

```
Rectangle view.ConfigView.modifyConfigButton  [private]
```

Configuration modify button.

Definition at line 83 of file ConfigView.java.

6.24.4.11 consultConfig

```
Text view.ConfigView.consultConfig  [private]
```

BConfiguration consult button text.

Definition at line 88 of file ConfigView.java.

6.24.4.12 consultConfigButton

```
Rectangle view.ConfigView.consultConfigButton  [private]
```

Configuration consult button.

Definition at line 93 of file ConfigView.java.

6.24.4.13 currentUserName

```
Label view.ConfigView.currentUserName  [private]
```

Current user name.

Definition at line 98 of file ConfigView.java.

6.24.4.14 logOut

```
Text view.ConfigView.logOut  [private]
```

LogOut button.

Definition at line 103 of file ConfigView.java.

The documentation for this class was generated from the following file:

- ConfigView.java

6.25 view.ConsultInitialBoardView Class Reference

Public Member Functions

- ConsultInitialBoardView ()
  Class creator.
- void initialize ()
  Initialize method which is executed when the scene is shown.
- void goToMenu () throws IOException
  Event method which is executed when the save button is clicked.

Private Member Functions

- void render ()

  Method executed everytime there is a change in the board.
- void drawPiece (Pair< Integer, Integer > pos, char pieceType)

  Painting method executed everytime there is a change in the board.
- Circle getCircle (Pair< Integer, Integer > pos)

  Method executed everytime there is a change in the board.

Private Attributes

- Text goToMenu

  goToMenu button.
- Circle f1c1

  Piece located in (1, 1).
- Circle f1c2

  Piece located in (1, 2).
- Circle f1c3

  Piece located in (1, 3).
- Circle f1c4

  Piece located in (1, 4).
- Circle f1c5

  Piece located in (1, 5).
- Circle f1c6

  Piece located in (1, 6).
- Circle f1c7

  Piece located in (1, 7).
- Circle f1c8

  Piece located in (1, 8).
- Circle f2c1

  Piece located in (2, 1).
- Circle f2c2

  Piece located in (2, 2).
- Circle f2c3

  Piece located in (2, 3).
- Circle f2c4

  Piece located in (2, 4).
- Circle f2c5

  Piece located in (2, 5).
- Circle f2c6

  Piece located in (2, 6).
- Circle f2c7

  Piece located in (2, 7).
- Circle f2c8

  Piece located in (2, 8).
- Circle f3c1

  Piece located in (3, 1).
- Circle f3c2

  Piece located in (3, 2).
- Circle f3c3

Piece located in (3, 3).
- Circle f3c4

  Piece located in (3, 4).
- Circle f3c5

  Piece located in (3, 5).
- Circle f3c6

  Piece located in (3, 6).
- Circle f3c7

  Piece located in (3, 7).
- Circle f3c8

  Piece located in (3, 8).
- Circle f4c1

  Piece located in (4, 1).
- Circle f4c2

  Piece located in (4, 2).
- Circle f4c3

  Piece located in (4, 3).
- Circle f4c4

  Piece located in (4, 4).
- Circle f4c5

  Piece located in (4, 5).
- Circle f4c6

  Piece located in (4, 6).
- Circle f4c7

  Piece located in (4, 7).
- Circle f4c8

  Piece located in (4, 8).
- Circle f5c1

  Piece located in (5, 1).
- Circle f5c2

  Piece located in (5, 2).
- Circle f5c3

  Piece located in (5, 3).
- Circle f5c4

  Piece located in (5, 4).
- Circle f5c5

  Piece located in (5, 5).
- Circle f5c6

  Piece located in (5, 6).
- Circle f5c7

  Piece located in (5, 7).
- Circle f5c8

  Piece located in (5, 8).
- Circle f6c1

  Piece located in (6, 1).
- Circle f6c2

  Piece located in (6, 2).
- Circle f6c3

  Piece located in (6, 3).
- Circle f6c4

  Piece located in (6, 4).

- Circle f6c5

  Piece located in (6, 5).
- Circle f6c6

  Piece located in (6, 6).
- Circle f6c7

  Piece located in (6, 7).
- Circle f6c8

  Piece located in (6, 8).
- Circle f7c1

  Piece located in (7, 1).
- Circle f7c2

  Piece located in (7, 2).
- Circle f7c3

  Piece located in (7, 3).
- Circle f7c4

  Piece located in (7, 4).
- Circle f7c5

  Piece located in (7, 5).
- Circle f7c6

  Piece located in (7, 6).
- Circle f7c7

  Piece located in (7, 7).
- Circle f7c8

  Piece located in (7, 8).
- Circle f8c1

  Piece located in (8, 1).
- Circle f8c2

  Piece located in (8, 2).
- Circle f8c3

  Piece located in (8, 3).
- Circle f8c4

  Piece located in (8, 4).
- Circle f8c5

  Piece located in (8, 5).
- Circle f8c6

  Piece located in (8, 6).
- Circle f8c7

  Piece located in (8, 7).
- Circle f8c8

  Piece located in (8, 8).
- JSONObject board

  Current board.

6.25.1 Detailed Description

This class represents the scene controller of the consult initial board view.

By Alex Rodriguez

Definition at line 23 of file ConsultInitialBoardView.java.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 ConsultInitialBoardView()

`view.ConsultInitialBoardView.ConsultInitialBoardView ( )`

Class creator.

Definition at line 29 of file ConsultInitialBoardView.java.

```
                                    {
    }
```

6.25.3 Member Function Documentation

6.25.3.1 initialize()

`void view.ConsultInitialBoardView.initialize ( )`

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 371 of file ConsultInitialBoardView.java.

```
                             {
        board = ViewCtrl.domainCtrl.viewBoard();
        render();
    }
```

6.25.3.2 goToMenu()

`void view.ConsultInitialBoardView.goToMenu ( ) throws IOException`

Event method which is executed when the save button is clicked.

Precondition

True

Postcondition

The game is saved and user can close the game.

Definition at line 381 of file ConsultInitialBoardView.java.

```
                                             {
        Stage currentWindow = (Stage) goToMenu.getScene().getWindow();
        currentWindow.close();
    }
```

6.25.3.3 render()

```
void view.ConsultInitialBoardView.render ( )  [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

The change is setted in the board.

Definition at line 391 of file ConsultInitialBoardView.java.

```
                                {
        for (int i = 0; i < 8; i++) {
            char[] row = board.getString(String.format("row%d", i)).toCharArray();
            for (int j = 0; j < 8; j++) drawPiece(new Pair<Integer, Integer>(i, j), row[j]);
        }
    }
```

6.25.3.4 drawPiece()

```
void view.ConsultInitialBoardView.drawPiece (
             Pair< Integer, Integer > pos,
             char pieceType )  [private]
```

Painting method executed everytime there is a change in the board.

Precondition

True

Postcondition

Pieces change to the correct color.

Definition at line 403 of file ConsultInitialBoardView.java.

```
                                                                          {
        Circle circle = getCircle(pos);
        switch (pieceType) {
            case 'B':
                circle.setFill(Color.web("0xFFFFFF", 1.0));
                break;
            case 'N':
                circle.setFill(Color.web("0x000000", 1.0));
                break;
            case '?':
                circle.setFill(Color.web("0x34d399", 1.0));
                break;
            default:
                break;
        }
    }
```

6.25.3.5 getCircle()

```
Circle view.ConsultInitialBoardView.getCircle (
            Pair< Integer, Integer > pos )  [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

Return the circle which belongs to the position.

Definition at line 425 of file ConsultInitialBoardView.java.

```
                                                  {
        try {
            Field field = this.getClass().getDeclaredField(String.format("f%sc%s", pos.first + 1,
      pos.second + 1));
            field.setAccessible(true);
            return (Circle) field.get(this);
        } catch (Exception e) {
            return new Circle();
        }
    }
```

6.25.4 Member Data Documentation

6.25.4.1 goToMenu

```
Text view.ConsultInitialBoardView.goToMenu  [private]
```

goToMenu button.

Definition at line 38 of file ConsultInitialBoardView.java.

6.25.4.2 f1c1

```
Circle view.ConsultInitialBoardView.f1c1  [private]
```

Piece located in (1, 1).

Definition at line 43 of file ConsultInitialBoardView.java.

6.25.4.3 f1c2

```
Circle view.ConsultInitialBoardView.f1c2 [private]
```

Piece located in (1, 2).

Definition at line 48 of file ConsultInitialBoardView.java.

6.25.4.4 f1c3

```
Circle view.ConsultInitialBoardView.f1c3 [private]
```

Piece located in (1, 3).

Definition at line 53 of file ConsultInitialBoardView.java.

6.25.4.5 f1c4

```
Circle view.ConsultInitialBoardView.f1c4 [private]
```

Piece located in (1, 4).

Definition at line 58 of file ConsultInitialBoardView.java.

6.25.4.6 f1c5

```
Circle view.ConsultInitialBoardView.f1c5 [private]
```

Piece located in (1, 5).

Definition at line 63 of file ConsultInitialBoardView.java.

6.25.4.7 f1c6

```
Circle view.ConsultInitialBoardView.f1c6 [private]
```

Piece located in (1, 6).

Definition at line 68 of file ConsultInitialBoardView.java.

6.25.4.8 f1c7

```
Circle view.ConsultInitialBoardView.f1c7  [private]
```

Piece located in (1, 7).

Definition at line 73 of file ConsultInitialBoardView.java.

6.25.4.9 f1c8

```
Circle view.ConsultInitialBoardView.f1c8  [private]
```

Piece located in (1, 8).

Definition at line 78 of file ConsultInitialBoardView.java.

6.25.4.10 f2c1

```
Circle view.ConsultInitialBoardView.f2c1  [private]
```

Piece located in (2, 1).

Definition at line 83 of file ConsultInitialBoardView.java.

6.25.4.11 f2c2

```
Circle view.ConsultInitialBoardView.f2c2  [private]
```

Piece located in (2, 2).

Definition at line 88 of file ConsultInitialBoardView.java.

6.25.4.12 f2c3

```
Circle view.ConsultInitialBoardView.f2c3  [private]
```

Piece located in (2, 3).

Definition at line 93 of file ConsultInitialBoardView.java.

6.25.4.13 f2c4

```
Circle view.ConsultInitialBoardView.f2c4  [private]
```

Piece located in (2, 4).

Definition at line 98 of file ConsultInitialBoardView.java.

6.25.4.14 f2c5

```
Circle view.ConsultInitialBoardView.f2c5  [private]
```

Piece located in (2, 5).

Definition at line 103 of file ConsultInitialBoardView.java.

6.25.4.15 f2c6

```
Circle view.ConsultInitialBoardView.f2c6  [private]
```

Piece located in (2, 6).

Definition at line 108 of file ConsultInitialBoardView.java.

6.25.4.16 f2c7

```
Circle view.ConsultInitialBoardView.f2c7  [private]
```

Piece located in (2, 7).

Definition at line 113 of file ConsultInitialBoardView.java.

6.25.4.17 f2c8

```
Circle view.ConsultInitialBoardView.f2c8  [private]
```

Piece located in (2, 8).

Definition at line 118 of file ConsultInitialBoardView.java.

6.25.4.18 f3c1

```
Circle view.ConsultInitialBoardView.f3c1  [private]
```

Piece located in (3, 1).

Definition at line 123 of file ConsultInitialBoardView.java.

6.25.4.19 f3c2

```
Circle view.ConsultInitialBoardView.f3c2  [private]
```

Piece located in (3, 2).

Definition at line 128 of file ConsultInitialBoardView.java.

6.25.4.20 f3c3

```
Circle view.ConsultInitialBoardView.f3c3  [private]
```

Piece located in (3, 3).

Definition at line 133 of file ConsultInitialBoardView.java.

6.25.4.21 f3c4

```
Circle view.ConsultInitialBoardView.f3c4  [private]
```

Piece located in (3, 4).

Definition at line 138 of file ConsultInitialBoardView.java.

6.25.4.22 f3c5

```
Circle view.ConsultInitialBoardView.f3c5  [private]
```

Piece located in (3, 5).

Definition at line 143 of file ConsultInitialBoardView.java.

6.25.4.23 f3c6

```
Circle view.ConsultInitialBoardView.f3c6  [private]
```

Piece located in (3, 6).

Definition at line 148 of file ConsultInitialBoardView.java.

6.25.4.24 f3c7

```
Circle view.ConsultInitialBoardView.f3c7  [private]
```

Piece located in (3, 7).

Definition at line 153 of file ConsultInitialBoardView.java.

6.25.4.25 f3c8

```
Circle view.ConsultInitialBoardView.f3c8  [private]
```

Piece located in (3, 8).

Definition at line 158 of file ConsultInitialBoardView.java.

6.25.4.26 f4c1

```
Circle view.ConsultInitialBoardView.f4c1  [private]
```

Piece located in (4, 1).

Definition at line 163 of file ConsultInitialBoardView.java.

6.25.4.27 f4c2

```
Circle view.ConsultInitialBoardView.f4c2  [private]
```

Piece located in (4, 2).

Definition at line 168 of file ConsultInitialBoardView.java.

6.25.4.28 f4c3

```
Circle view.ConsultInitialBoardView.f4c3  [private]
```

Piece located in (4, 3).

Definition at line 173 of file ConsultInitialBoardView.java.

6.25.4.29 f4c4

```
Circle view.ConsultInitialBoardView.f4c4  [private]
```

Piece located in (4, 4).

Definition at line 178 of file ConsultInitialBoardView.java.

6.25.4.30 f4c5

```
Circle view.ConsultInitialBoardView.f4c5  [private]
```

Piece located in (4, 5).

Definition at line 183 of file ConsultInitialBoardView.java.

6.25.4.31 f4c6

```
Circle view.ConsultInitialBoardView.f4c6  [private]
```

Piece located in (4, 6).

Definition at line 188 of file ConsultInitialBoardView.java.

6.25.4.32 f4c7

```
Circle view.ConsultInitialBoardView.f4c7  [private]
```

Piece located in (4, 7).

Definition at line 193 of file ConsultInitialBoardView.java.

6.25.4.33 f4c8

```
Circle view.ConsultInitialBoardView.f4c8 [private]
```

Piece located in (4, 8).

Definition at line 198 of file ConsultInitialBoardView.java.

6.25.4.34 f5c1

```
Circle view.ConsultInitialBoardView.f5c1 [private]
```

Piece located in (5, 1).

Definition at line 203 of file ConsultInitialBoardView.java.

6.25.4.35 f5c2

```
Circle view.ConsultInitialBoardView.f5c2 [private]
```

Piece located in (5, 2).

Definition at line 208 of file ConsultInitialBoardView.java.

6.25.4.36 f5c3

```
Circle view.ConsultInitialBoardView.f5c3 [private]
```

Piece located in (5, 3).

Definition at line 213 of file ConsultInitialBoardView.java.

6.25.4.37 f5c4

```
Circle view.ConsultInitialBoardView.f5c4 [private]
```

Piece located in (5, 4).

Definition at line 218 of file ConsultInitialBoardView.java.

6.25.4.38 f5c5

```
Circle view.ConsultInitialBoardView.f5c5 [private]
```

Piece located in (5, 5).

Definition at line 223 of file ConsultInitialBoardView.java.

6.25.4.39 f5c6

```
Circle view.ConsultInitialBoardView.f5c6 [private]
```

Piece located in (5, 6).

Definition at line 228 of file ConsultInitialBoardView.java.

6.25.4.40 f5c7

```
Circle view.ConsultInitialBoardView.f5c7 [private]
```

Piece located in (5, 7).

Definition at line 233 of file ConsultInitialBoardView.java.

6.25.4.41 f5c8

```
Circle view.ConsultInitialBoardView.f5c8 [private]
```

Piece located in (5, 8).

Definition at line 238 of file ConsultInitialBoardView.java.

6.25.4.42 f6c1

```
Circle view.ConsultInitialBoardView.f6c1 [private]
```

Piece located in (6, 1).

Definition at line 243 of file ConsultInitialBoardView.java.

6.25.4.43 f6c2

```
Circle view.ConsultInitialBoardView.f6c2 [private]
```

Piece located in (6, 2).

Definition at line 248 of file ConsultInitialBoardView.java.

6.25.4.44 f6c3

```
Circle view.ConsultInitialBoardView.f6c3 [private]
```

Piece located in (6, 3).

Definition at line 253 of file ConsultInitialBoardView.java.

6.25.4.45 f6c4

```
Circle view.ConsultInitialBoardView.f6c4 [private]
```

Piece located in (6, 4).

Definition at line 258 of file ConsultInitialBoardView.java.

6.25.4.46 f6c5

```
Circle view.ConsultInitialBoardView.f6c5 [private]
```

Piece located in (6, 5).

Definition at line 263 of file ConsultInitialBoardView.java.

6.25.4.47 f6c6

```
Circle view.ConsultInitialBoardView.f6c6 [private]
```

Piece located in (6, 6).

Definition at line 268 of file ConsultInitialBoardView.java.

6.25.4.48 f6c7

```
Circle view.ConsultInitialBoardView.f6c7 [private]
```

Piece located in (6, 7).

Definition at line 273 of file ConsultInitialBoardView.java.

6.25.4.49 f6c8

```
Circle view.ConsultInitialBoardView.f6c8 [private]
```

Piece located in (6, 8).

Definition at line 278 of file ConsultInitialBoardView.java.

6.25.4.50 f7c1

```
Circle view.ConsultInitialBoardView.f7c1 [private]
```

Piece located in (7, 1).

Definition at line 283 of file ConsultInitialBoardView.java.

6.25.4.51 f7c2

```
Circle view.ConsultInitialBoardView.f7c2 [private]
```

Piece located in (7, 2).

Definition at line 288 of file ConsultInitialBoardView.java.

6.25.4.52 f7c3

```
Circle view.ConsultInitialBoardView.f7c3 [private]
```

Piece located in (7, 3).

Definition at line 293 of file ConsultInitialBoardView.java.

6.25.4.53 f7c4

```
Circle view.ConsultInitialBoardView.f7c4  [private]
```

Piece located in (7, 4).

Definition at line 298 of file ConsultInitialBoardView.java.

6.25.4.54 f7c5

```
Circle view.ConsultInitialBoardView.f7c5  [private]
```

Piece located in (7, 5).

Definition at line 303 of file ConsultInitialBoardView.java.

6.25.4.55 f7c6

```
Circle view.ConsultInitialBoardView.f7c6  [private]
```

Piece located in (7, 6).

Definition at line 308 of file ConsultInitialBoardView.java.

6.25.4.56 f7c7

```
Circle view.ConsultInitialBoardView.f7c7  [private]
```

Piece located in (7, 7).

Definition at line 313 of file ConsultInitialBoardView.java.

6.25.4.57 f7c8

```
Circle view.ConsultInitialBoardView.f7c8  [private]
```

Piece located in (7, 8).

Definition at line 318 of file ConsultInitialBoardView.java.

6.25.4.58 f8c1

```
Circle view.ConsultInitialBoardView.f8c1  [private]
```

Piece located in (8, 1).

Definition at line 323 of file ConsultInitialBoardView.java.

6.25.4.59 f8c2

```
Circle view.ConsultInitialBoardView.f8c2  [private]
```

Piece located in (8, 2).

Definition at line 328 of file ConsultInitialBoardView.java.

6.25.4.60 f8c3

```
Circle view.ConsultInitialBoardView.f8c3  [private]
```

Piece located in (8, 3).

Definition at line 333 of file ConsultInitialBoardView.java.

6.25.4.61 f8c4

```
Circle view.ConsultInitialBoardView.f8c4  [private]
```

Piece located in (8, 4).

Definition at line 338 of file ConsultInitialBoardView.java.

6.25.4.62 f8c5

```
Circle view.ConsultInitialBoardView.f8c5  [private]
```

Piece located in (8, 5).

Definition at line 343 of file ConsultInitialBoardView.java.

6.25.4.63 f8c6

`Circle view.ConsultInitialBoardView.f8c6 [private]`

Piece located in (8, 6).

Definition at line 348 of file ConsultInitialBoardView.java.

6.25.4.64 f8c7

`Circle view.ConsultInitialBoardView.f8c7 [private]`

Piece located in (8, 7).

Definition at line 353 of file ConsultInitialBoardView.java.

6.25.4.65 f8c8

`Circle view.ConsultInitialBoardView.f8c8 [private]`

Piece located in (8, 8).

Definition at line 358 of file ConsultInitialBoardView.java.

6.25.4.66 board

`JSONObject view.ConsultInitialBoardView.board [private]`

Current board.

Definition at line 362 of file ConsultInitialBoardView.java.

The documentation for this class was generated from the following file:

- ConsultInitialBoardView.java

6.26 domain.Difficulty Class Reference

Implements the abstract class and methods of all the difficulty implementations. By Arnau Pujantell.

Public Member Functions

- Difficulty (Integer difficulty, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally, PieceType pieceType)

  Create a Difficulty instance.
- int getDifficulty ()

  Get the difficulty of the implicit chosen Difficulty.
- boolean getCanEatHorizontally ()

  Get the canEatHorizontally of the implicit chosen Difficulty.
- boolean getCanEatVertically ()

  Get the canEatVertically of the implicit chosen Difficulty.
- boolean getCanEatDiagonally ()

  Get the canEatDiagonally of the implicit chosen Difficulty.
- PieceType getPieceType ()

  Get the pieceType of the implicit chosen Difficulty.
- int getMaxDepth ()

  Get the maxDepth of the implicit chosen Difficulty.
- void setMaxDepth (int maxDepth)

  Set the maxDepth of the implicit chosen Difficulty.
- abstract Pair< Integer, Integer > place (PieceType[ ][ ] playingBoard)

  Get the next best possible position for the implicit player.

Static Protected Member Functions

- static PieceType inversePieceType (PieceType pieceType)

  Get the inverse of the given player.

Protected Attributes

- Integer maxDepth

  Max depth for the heuristics of the chosen algorithm. It is calculated from the implicit difficulty.
- Integer difficulty

  Difficulty for the chosen algorithm. It is mainly used to calculate the implicit max depth.
- Boolean canEatHorizontally

  Whether the pieces of the current Game can be eaten horizontally.
- Boolean canEatVertically

  Whether the pieces of the current Game can be eaten vertically.
- Boolean canEatDiagonally

  Whether the pieces of the current Game can be eaten diagonally.
- PieceType pieceType

  Player that wants to be maximized.

6.26.1 Detailed Description

Implements the abstract class and methods of all the difficulty implementations. By Arnau Pujantell.

Definition at line 16 of file Difficulty.java.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Difficulty()

```
domain.Difficulty.Difficulty (
            Integer difficulty,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally,
            PieceType pieceType )
```

Create a Difficulty instance.

Precondition

The given difficulty is a positive number. The given rules are not all false.

Postcondition

An Difficulty instance is created and its implicits difficulty, canEatHorizontally, canEatVertically, canEat↩ Diagonally and pieceType attributes are setted. The implicit maxDepth attribute is setted to the double of the entered difficulty.

Parameters

difficulty	Difficulty for the chosen algorithm.
canEatHorizontally	Whether the pieces of the current Game can be eaten horizontally.
canEatVertically	Whether the pieces of the current Game can be eaten vertically.
canEatDiagonally	Whether the pieces of the current Game can be eaten diagonally.
pieceType	Player that wants to be maximized.

Definition at line 57 of file Difficulty.java.

```
                                                               {
        this.difficulty = difficulty;
        this.canEatHorizontally = canEatHorizontally;
        this.canEatVertically = canEatVertically;
        this.canEatDiagonally = canEatDiagonally;
        this.pieceType = pieceType;
        this.maxDepth = difficulty * 2;
    }
```

6.26.3 Member Function Documentation

6.26.3.1 getDifficulty()

```
int domain.Difficulty.getDifficulty ( )
```

Get the difficulty of the implicit chosen Difficulty.

Precondition

True

Postcondition

The difficulty attribute of the implicit chosen Difficulty is returned.

Returns

difficulty of the implicit chosen Difficulty.

Definition at line 75 of file Difficulty.java.

```
                                {
        return this.difficulty;
    }
```

6.26.3.2 getCanEatHorizontally()

`boolean domain.Difficulty.getCanEatHorizontally ( )`

Get the canEatHorizontally of the implicit chosen Difficulty.

Precondition

True

Postcondition

The canEatHorizontally attribute of the implicit chosen Difficulty is returned.

Returns

canEatHorizontally of the implicit chosen Difficulty.

Definition at line 85 of file Difficulty.java.

```
                                         {
        return this.canEatHorizontally;
    }
```

6.26.3.3 getCanEatVertically()

`boolean domain.Difficulty.getCanEatVertically ( )`

Get the canEatVertically of the implicit chosen Difficulty.

Precondition

True

Postcondition

The canEatVertically attribute of the implicit chosen Difficulty is returned.

Returns

canEatVertically of the implicit chosen Difficulty.

Definition at line 95 of file Difficulty.java.

```
                                       {
        return this.canEatVertically;
    }
```

6.26.3.4 getCanEatDiagonally()

```
boolean domain.Difficulty.getCanEatDiagonally ( )
```

Get the canEatDiagonally of the implicit chosen Difficulty.

Precondition

True

Postcondition

The canEatDiagonally attribute of the implicit chosen Difficulty is returned.

Returns

canEatDiagonally of the implicit chosen Difficulty.

Definition at line 105 of file Difficulty.java.

```
                                     {
        return this.canEatDiagonally;
    }
```

6.26.3.5 getPieceType()

```
PieceType domain.Difficulty.getPieceType ( )
```

Get the pieceType of the implicit chosen Difficulty.

Precondition

True

Postcondition

The pieceType attribute of the implicit chosen Difficulty is returned.

Returns

pieceType of the implicit chosen Difficulty.

Definition at line 115 of file Difficulty.java.

```
                                {
        return this.pieceType;
    }
```

6.26.3.6 getMaxDepth()

```
int domain.Difficulty.getMaxDepth ( )
```

Get the maxDepth of the implicit chosen Difficulty.

Precondition

True

Postcondition

The maxDepth attribute of the implicit chosen Difficulty is returned.

Returns

maxDepth of the implicit chosen Difficulty.

Definition at line 125 of file Difficulty.java.

```
                            {
        return this.maxDepth;
    }
```

6.26.3.7 setMaxDepth()

```
void domain.Difficulty.setMaxDepth (
             int maxDepth )
```

Set the maxDepth of the implicit chosen Difficulty.

Precondition

The given maxDepth is a positive number.

Postcondition

The maxDepth attribute of the implicit chosen Difficulty is setted.

Parameters

maxDepth	Max depth for the heuristics of the chosen algorithm.

Definition at line 135 of file Difficulty.java.

```
                                        {
        this.maxDepth = maxDepth;
    }
```

6.26.3.8 inversePieceType()

```
static PieceType domain.Difficulty.inversePieceType (
            PieceType pieceType ) [static], [protected]
```

Get the inverse of the given player.

Precondition

True

Postcondition

It is returned the inverse, that is the opponent, of the given player.

Parameters

pieceType	Player to be inversed.

Returns

The opponent player.

Definition at line 146 of file Difficulty.java.

```
                                                                   {
        return (pieceType == PieceType.PLAYER2 ? PieceType.PLAYER1 : PieceType.PLAYER2);
    }
```

6.26.3.9 place()

```
abstract Pair<Integer, Integer> domain.Difficulty.place (
            PieceType playingBoard[ ][ ] ) [abstract]
```

Get the next best possible position for the implicit player.

Precondition

True

Postcondition

It is returned the next best possible position for the implicit player, using the chosen algorithm with the implicit maximum depth, or null if there isn't any.

Parameters

playingBoard	Current playing Board.

Returns

The next best possible position for the implicit player or null if there isn't any.

Reimplemented in domain.MediumDifficulty, domain.EasyDifficulty, and domain.HardDifficulty.

6.26.4 Member Data Documentation

6.26.4.1 maxDepth

```
Integer domain.Difficulty.maxDepth [protected]
```

Max depth for the heuristics of the chosen algorithm. It is calculated from the implicit difficulty.

Definition at line 22 of file Difficulty.java.

6.26.4.2 difficulty

```
Integer domain.Difficulty.difficulty [protected]
```

Difficulty for the chosen algorithm. It is mainly used to calculate the implicit max depth.

Definition at line 26 of file Difficulty.java.

6.26.4.3 canEatHorizontally

```
Boolean domain.Difficulty.canEatHorizontally [protected]
```

Whether the pieces of the current Game can be eaten horizontally.

Definition at line 30 of file Difficulty.java.

6.26.4.4 canEatVertically

```
Boolean domain.Difficulty.canEatVertically [protected]
```

Whether the pieces of the current Game can be eaten vertically.

Definition at line 34 of file Difficulty.java.

6.26.4.5 canEatDiagonally

```
Boolean domain.Difficulty.canEatDiagonally  [protected]
```

Whether the pieces of the current Game can be eaten diagonally.

Definition at line 38 of file Difficulty.java.

6.26.4.6 pieceType

```
PieceType domain.Difficulty.pieceType  [protected]
```

Player that wants to be maximized.

Definition at line 42 of file Difficulty.java.

The documentation for this class was generated from the following file:

- Difficulty.java

6.27 domain.DifficultyCtrl Class Reference

Difficulty domain sub-controller. Is in charge of EasyDifficulty, MediumDifficulty and HardDifficulty. It communicates with the main domain controller. It forwards the current bot's placePiece request to the correct algorithm depending on the current game's difficulty: 1 to 3: EasyDifficulty (Minimax). 4 to 6: MediumDifficulty (Minimax alpha beta pruning). 7 to 10: HardDifficulty (Montecarlo).

Public Member Functions

- DifficultyCtrl ()

  Creator method that creates an instance of Difficulty Controller.
- Pair< Integer, Integer > getBestPositionByBot (Bot bot, Configuration configuration, Board board, PieceType myPieceType)

  Returns the next best possible position, or null if none, to place a piece on the current game for the current bot. It forwards the placePiece request to the correct algorithm depending on the current bot's difficulty.
- Pair< Integer, Integer > getBestPosition (Integer difficulty, Configuration configuration, Board board, PieceType myPieceType)

  Returns the next best possible position, or null if none, to place a piece on the current game for the given player. It forwards the placePiece request to the correct algorithm depending on the difficulty. This method can be used to implement the assisted mode.

6.27.1 Detailed Description

Difficulty domain sub-controller. Is in charge of EasyDifficulty, MediumDifficulty and HardDifficulty. It communicates with the main domain controller. It forwards the current bot's placePiece request to the correct algorithm depending on the current game's difficulty: 1 to 3: EasyDifficulty (Minimax). 4 to 6: MediumDifficulty (Minimax alpha beta pruning). 7 to 10: HardDifficulty (Montecarlo).

By Alex Rodriguez.

See also

domain.Difficulty

Definition at line 22 of file DifficultyCtrl.java.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 DifficultyCtrl()

```
domain.DifficultyCtrl.DifficultyCtrl ( )
```

Creator method that creates an instance of Difficulty Controller.

Precondition

True

Postcondition

An instance of DifficultyCtrl is instanced.

Definition at line 32 of file DifficultyCtrl.java.

```
32                                {
33     }
```

6.27.3 Member Function Documentation

6.27.3.1 getBestPositionByBot()

```
Pair<Integer, Integer> domain.DifficultyCtrl.getBestPositionByBot (
            Bot bot,
            Configuration configuration,
            Board board,
            PieceType myPieceType )
```

Returns the next best possible position, or null if none, to place a piece on the current game for the current bot. It forwards the placePiece request to the correct algorithm depending on the current bot's difficulty.

Precondition

All parameters aren't null.

Postcondition

The best position for the bot is returned.

Parameters

bot	An instance of the Bot Class
configuration	An instance of the Configuration Class
board	An instance of the Board Class
myPieceType	PieceType variable that represents a Player in a Board

Returns

The best position for the bot is returned.

Definition at line 48 of file DifficultyCtrl.java.

```
                                  {
        return this.getBestPosition(bot.getDifficulty(), configuration, board, myPieceType);
    }
```

6.27.3.2 getBestPosition()

```
Pair<Integer, Integer> domain.DifficultyCtrl.getBestPosition (
            Integer difficulty,
            Configuration configuration,
            Board board,
            PieceType myPieceType )
```

Returns the next best possible position, or null if none, to place a piece on the current game for the given player. It forwards the placePiece request to the correct algorithm depending on the difficulty. This method can be used to implement the assisted mode.

Precondition

All parameters aren't null.

Postcondition

The best position is returned.

Parameters

difficulty	Integer that represents the level of the assisted mode.
configuration	An instance of the Configuration Class
board	An instance of the Board Class
myPieceType	PieceType variable that represents a Player in a Board

Returns

The best position is returned.

Definition at line 64 of file DifficultyCtrl.java.

```
                                {
        Pair<Integer, Integer> bestPosition = null;
        boolean cH = configuration.getCanEatHorizontally();
        boolean cV = configuration.getCanEatVertically();
        boolean cD = configuration.getCanEatDiagonally();
        PieceType[][] b = board.getBoard();

        switch (difficulty) {
            case 1:
                bestPosition = new HardDifficulty(7, cH, cV, cD, myPieceType).place(b);
                break;
            case 2:
                bestPosition = new HardDifficulty(8, cH, cV, cD, myPieceType).place(b);
                break;
            case 3:
                bestPosition = new HardDifficulty(9, cH, cV, cD, myPieceType).place(b);
                break;
            case 4:
                bestPosition = new HardDifficulty(10, cH, cV, cD, myPieceType).place(b);
                break;
            case 5:
                EasyDifficulty ed5 = new EasyDifficulty(1, cH, cV, cD, myPieceType); ed5.setMaxDepth(1);
                bestPosition = ed5.place(b);
                break;
            case 6:
                bestPosition = new EasyDifficulty(1, cH, cV, cD, myPieceType).place(b);
                break;
            case 7:
                MediumDifficulty md7 = new MediumDifficulty(1, cH, cV, cD, myPieceType);
    md7.setMaxDepth(3);
                bestPosition = md7.place(b);
                break;
            case 8:
                bestPosition = new MediumDifficulty(2, cH, cV, cD, myPieceType).place(b);
                break;
            case 9:
                MediumDifficulty md9 = new MediumDifficulty(2, cH, cV, cD, myPieceType);
    md9.setMaxDepth(5);
                bestPosition = md9.place(b);
                break;
            case 10:
                bestPosition = new MediumDifficulty(3, cH, cV, cD, myPieceType).place(b);
                break;
        }

        return bestPosition;
    }
```

The documentation for this class was generated from the following file:

- DifficultyCtrl.java

6.28 domain.DomainCtrl Class Reference

Is the main domain controller. It keeps the current state of all the game and application. It serves as a forwarder for the specific domain class controllers, that's why most of the sub-controller methods receive as a parameter the current instance of the associated class. Moreover, it implements a few methods that do not have a specific domain class binded. It also gathers all the thrown exceptions in the sub-controllers and transforms them into string messages in order to pass them to the view layer without coupling any domain specific logic. By Manuel Navid.

Public Member Functions

- DomainCtrl ()

  Default creator method.
- void exitApplication ()

  Method to exit the application.
- void logout ()

  Method to logout from the current User.

- void exitGame ()

  Method to exit a Game.
- Pair< JSONObject, String > createUser (String name, String password, String confirmation)

  Creator that, given a name and a password, creates a new user in the repository.
- Pair< JSONObject, String > createBot (String name, Integer difficulty)

  Method that, given a name, a difficulty and an ID, creates a new bot in the repository.
- Pair< JSONObject, String > login (String name, String password)

  Method that, given a name and a password, allows us to log in the Othello game.
- Pair< JSONObject, String > getUser (UUID userID)

  Method that, given an ID, returns a user.
- Pair< JSONObject, String > getBot (UUID botID)

  Method that, given an ID, returns a bot.
- Pair< JSONObject, String > getPlayer (UUID playerID)

  Method that, given an ID, returns a player.
- ArrayList< Pair< String, UUID > > listUsers ()

  Method that lists all the users from the repository.
- ArrayList< Pair< String, UUID > > listBots ()

  Method that lists all the bots from the repository.
- Pair< JSONObject, String > modifyUser (String name, String password, String confirmation)

  Modifier that, given an ID, a name and a password, allows us to modify the user's credentials changing the name, the password or both.
- Pair< JSONObject, String > modifyBot (UUID botID, String name, Integer difficulty)

  Method that, given an ID, a name, a difficulty and an ID, allows us to modify the bot's credentials changing the name, the difficulty or both.
- String deleteUser (String password)

  Method that, given an ID, a name and a password, allows us to delete a user.
- String deleteBot (UUID botID)

  Method that, given a name, a botID and a deleterID, allows us to delete a bot.
- JSONObject viewUser ()

  Method to get the current User data.
- Pair< JSONObject, JSONObject > viewPlayers ()

  Method to get the current Players(Player1 and 2) data.
- Pair< JSONObject, String > createConfiguration (String name, Boolean canEatHorizontally, Boolean can←EatVertically, Boolean canEatDiagonally)

  Lets the current user to create a new configuration with a name, rules and the initial board.
- String createInitialBoard ()

  Lets the current user create a default initial board to start modifying it in the configuration's creation.
- Pair< JSONObject, String > modifyConfiguration (String name, Boolean canEatHorizontally, Boolean can←EatVertically, Boolean canEatDiagonally)

  Lets the current user modify a configuration he/she created. The configuration's rules and the initial board can be changed.
- String modifyInitialBoard (String name)

  Lets the current user to modify the initial board of a configuration he/she created.
- String deleteConfiguration (String name)

  Lets the current user delete a configuration he/she created if it has not been used.
- Pair< JSONObject, String > getConfiguration (String name)

  Returns the configuration identified by the name.
- Pair< ArrayList< String >, String > listConfigurations ()

  Returns a list of all configurations names in the system.
- Pair< JSONObject, String > getInitialBoard (String name)

  Get an Initial Board of a Configuration.

- JSONObject viewConfiguration ()

    Method to get the current Configuration data.
- Pair< JSONObject, String > createGame (UUID player1ID, UUID player2ID, String configurationName)

    Lets the current user create a new game, selecting both players and a configuration of rules and initial board.
- Pair< JSONObject, String > saveGame ()

    Lets the current user manually save the current game and playing board state.
- Pair< JSONObject, String > getGame (String name)

    Returns the game identified by its name and any of the participant player IDs.
- Pair< JSONObject, String > getPlayingBoard (String name)

    Returns the playing board associated with the given game name and any of the participant player IDs.
- Pair< ArrayList< String >, String > listGames ()

    Returns a list of all games names identified by any of the participant player IDs.
- Pair< JSONObject, String > selectGame (String name)

    Lets the current user load a selected game by name in order to play it afterwards.
- Pair< JSONObject, String > play ()

    Lets the current user start playing on the current loaded game.
- Pair< JSONObject, String > surrender (UUID surrendeeID)

    Lets a player of the current game surrender, setting the winner as the opponent.
- JSONObject viewGame ()

    Method to get the current Game data.
- Pair< Integer, Integer > getBestPosition (Integer difficulty, String myPieceType)

    Returns the next best possible position, or null if none, to place a piece on the current for a given player. It forwards the placePiece request to the correct algorithm depending on the difficulty. This method can be used to implement the assisted mode.
- JSONObject placePieceConfig (Pair< Integer, Integer > position, String myPieceType)

    Modifying method returns the board modified with the added position in JSON format.
- JSONObject removePiece (Pair< Integer, Integer > position)

    Modifying method that removes a piece from currentBoard and returns the Board in JSON format.
- Pair< Integer, Integer > getNumPieces ()

    Get method that returns the value of the board parameter's PiecesPlayer1 and PiecesPlayer2 attributes.
- ArrayList< Pair< Integer, Integer > > validPositions (String myPieceType)

    Method that returns an Array of the valid positions in board of the player myPieceType taking into consideration the Configuration of the currentGame.
- String isValidBoard ()

    Method that warns us if an instance of the board parameters is invalid.
- Pair< Pair< JSONObject, String >, String > placePiece (Pair< Integer, Integer > position, UUID playerID, String pieceType)

    Modifying method that adds a piece in the board parameter.
- JSONObject viewBoard ()

    Method to get the current Board data.
- JSONObject getRanking (String name)

    Returns the ranking identified by name.
- ArrayList< String > listRankings ()

    Returns a list of all ranking names in the system.
- ArrayList< Pair< String, JSONObject > > listRecords ()

    Returns the entries with the highest score of the current user for each ranking in the system.

Private Member Functions

- PieceType stringToPieceType (String pieceType)

  Method to convert an String from the presentation level to a PieceType in order to decouple domain specific knowledgement.
- PieceType inversePieceType (PieceType pieceType)

  Private method that inverts the Player's pieceType in order to get its opponent.
- Pair< Pair< JSONObject, String >, String > finishGame ()

  Lets the system to automatically finish the game when any players win or when there aren't any valid positions left to place a piece on the board, setting the winner of the game or setting that the game has ended in a draw.
- Pair< Pair< JSONObject, String >, String > nextTurn ()

  Lets the system to automatically pass the turn of the current game.
- Pair< Pair< JSONObject, String >, String > currentTurn ()

  Lets the system to automatically decide the current turn of the current game.
- void createEntries ()

  Lets the system to automatically create the entries of the associated ranking when the current user finishes a game.

Private Attributes

- PlayerCtrl playerCtrl

  Player Controller.
- ConfigurationCtrl configurationCtrl

  Configuration Controller.
- BoardCtrl boardCtrl

  Board Controller.
- GameCtrl gameCtrl

  Game Controller.
- RankingCtrl rankingCtrl

  Ranking Controller.
- DifficultyCtrl difficultyCtrl

  Difficulty Controller.
- User currentUser

  Current logged User.
- Player currentPlayer1

  Player 1 of the current game. Can be either a User or a Bot.
- Player currentPlayer2

  Player 2 of the current game. Can be either a User or a Bot.
- Board currentBoard

  Current loaded board from the current configuration or game.
- Configuration currentConfiguration

  Current loaded configuration.
- Game currentGame

  Current loaded game.

6.28.1 Detailed Description

Is the main domain controller. It keeps the current state of all the game and application. It serves as a forwarder for the specific domain class controllers, that's why most of the sub-controller methods receive as a parameter the current instance of the associated class. Moreover, it implements a few methods that do not have a specific domain class binded. It also gathers all the thrown exceptions in the sub-controllers and transforms them into string messages in order to pass them to the view layer without coupling any domain specific logic. By Manuel Navid.

Definition at line 29 of file DomainCtrl.java.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 DomainCtrl()

`domain.DomainCtrl.DomainCtrl ( )`

Default creator method.

Precondition

True

Postcondition

An instance of domainCtrl is created.

Definition at line 88 of file DomainCtrl.java.

```
                        {
        this.playerCtrl = new PlayerCtrl();
        this.configurationCtrl = new ConfigurationCtrl();
        this.boardCtrl = new BoardCtrl();
        this.gameCtrl = new GameCtrl();
        this.rankingCtrl = new RankingCtrl();
        this.difficultyCtrl = new DifficultyCtrl();
        this.currentUser = null;
        this.currentPlayer1 = null;
        this.currentPlayer2 = null;
        this.currentBoard = null;
        this.currentConfiguration = null;
         this.currentGame = null;
    }
```

6.28.3 Member Function Documentation

6.28.3.1 exitApplication()

`void domain.DomainCtrl.exitApplication ( )`

Method to exit the application.

Precondition

True

Postcondition

We exited the application, therefore it closes.

Definition at line 110 of file DomainCtrl.java.

```
                                  {
        this.logout();
        System.exit(0);
    }
```

6.28.3.2 logout()

```
void domain.DomainCtrl.logout ( )
```

Method to logout from the current User.

Precondition

True

Postcondition

The current User is null, meaning there is no user logged in.

Definition at line 120 of file DomainCtrl.java.

```
                        {
        this.exitGame();
        this.currentUser = null;
    }
```

6.28.3.3 exitGame()

```
void domain.DomainCtrl.exitGame ( )
```

Method to exit a Game.

Precondition

True

Postcondition

All the current attributes of DomainCtrl used to play a Game are changed to null.

Definition at line 130 of file DomainCtrl.java.

```
                          {
        this.currentPlayer1 = null;
        this.currentPlayer2 = null;
        this.currentBoard = null;
        this.currentConfiguration = null;
        this.currentGame = null;
    }
```

6.28.3.4 createUser()

```
Pair<JSONObject, String> domain.DomainCtrl.createUser (
            String name,
            String password,
            String confirmation )
```

Creator that, given a name and a password, creates a new user in the repository.

Precondition

True

Parameters

name	Name of a User
password	Password of a User
confirmation	Confirmation of the entered password

Postcondition

The user is saved and is returned in JSON format if no exceptions where triggered, else the excepction will be returned in a string.

Definition at line 149 of file DomainCtrl.java.

```
                                                                                                {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            User user = this.playerCtrl.createUser(name, password, confirmation);
            result.first = user.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.5 createBot()

```
Pair<JSONObject, String> domain.DomainCtrl.createBot (
              String name,
              Integer difficulty )
```

Method that, given a name, a difficulty and an ID, creates a new bot in the repository.

Precondition

currentUser is not null.

Parameters

name	Name of a Bot
difficulty	Difficulty of a Bot

Postcondition

The created bot is saved and is returned in JSON format if no exceptions were triggered, else the excepction will be returned in a string.

Definition at line 170 of file DomainCtrl.java.

```
                                                                              {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Bot bot = this.playerCtrl.createBot(name, difficulty, this.currentUser.getID());
            result.first = bot.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.6 login()

```
Pair<JSONObject, String> domain.DomainCtrl.login (
            String name,
            String password )
```

Method that, given a name and a password, allows us to log in the Othello game.

Precondition

True

Parameters

name	Name of a User
password	Password of a User

Postcondition

The user found in the repository is returned in JSON format if there is no exception triggered, else the exception will be returned in a string.

Definition at line 191 of file DomainCtrl.java.

```
                                                                       {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            User user = this.playerCtrl.login(name, password);
            this.currentUser = user;
            result.first = user.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.7 getUser()

```
Pair<JSONObject, String> domain.DomainCtrl.getUser (
            UUID userID )
```

Method that, given an ID, returns a user.

Precondition

userID is not null

Parameters

userID	UUID of a User

Postcondition

User is found in repository and returned in JSON format if there was no exceptions triggered, else the exception will be returned in a string.

Definition at line 211 of file DomainCtrl.java.

```
                                                        {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            User user = this.playerCtrl.getUser(userID);
            result.first = user.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.8 getBot()

```
Pair<JSONObject, String> domain.DomainCtrl.getBot (
              UUID botID )
```

Method that, given an ID, returns a bot.

Precondition

botID is not null

Parameters

botID	UUID of a Bot

Postcondition

Bot is found in repository and returned in JSON format if there was no exceptions triggered, else the exception will be returned in a string.

Definition at line 230 of file DomainCtrl.java.

```
                                                      {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Bot bot = this.playerCtrl.getBot(botID);
            result.first = bot.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.9 getPlayer()

```
Pair<JSONObject, String> domain.DomainCtrl.getPlayer (
            UUID playerID )
```

Method that, given an ID, returns a player.

Precondition

playerID is not null

Parameters

playerID	UUID of a Bot

Postcondition

Player is found in repository and returned in JSON format if there was no exceptions triggered, else the exception will be returned in a string.

Definition at line 249 of file DomainCtrl.java.

```
                                                              {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            try {
                User user = this.playerCtrl.getUser(playerID);
                result.first = user.serialize();
            } catch (Exception e) {
                Bot bot = this.playerCtrl.getBot(playerID);
                result.first = bot.serialize();
            }
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.10 listUsers()

```
ArrayList<Pair<String, UUID> > domain.DomainCtrl.listUsers ( )
```

Method that lists all the users from the repository.

Precondition

True

Postcondition

All bots are returned in an ArrayList with their names and IDs.

Definition at line 272 of file DomainCtrl.java.

```
                                                      {
        return this.playerCtrl.listUsers();
    }
```

6.28.3.11 listBots()

```
ArrayList<Pair<String, UUID> > domain.DomainCtrl.listBots ( )
```

Method that lists all the bots from the repository.

Precondition

True

Postcondition

All bots are returned in an ArrayList with their names and IDs.

Definition at line 281 of file DomainCtrl.java.

```
                                                      {
        return this.playerCtrl.listBots();
    }
```

6.28.3.12 modifyUser()

```
Pair<JSONObject, String> domain.DomainCtrl.modifyUser (
              String name,
              String password,
              String confirmation )
```

Modifier that, given an ID, a name and a password, allows us to modify the user's credentials changing the name, the password or both.

Precondition

currentUser is not null.

Parameters

name	Name of a User
password	Password of a User
confirmation	Confirmation of the entered password

Postcondition

Name, password or both are changed, saved in currentUser and it's returned in JSON format if no exceptions were triggered, else the exception will be returned in a string.

Definition at line 294 of file DomainCtrl.java.

```
                                                                                                     {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            User user = this.playerCtrl.modifyUser(this.currentUser.getID(), name, password,
    confirmation);
            this.currentUser = user;
            result.first = user.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.13 modifyBot()

```
Pair<JSONObject, String> domain.DomainCtrl.modifyBot (
              UUID botID,
              String name,
              Integer difficulty )
```

Method that, given an ID, a name, a difficulty and an ID, allows us to modify the bot's credentials changing the name, the difficulty or both.

Precondition

currentUser is not null.

Parameters

botID	ID of a Bot
name	Name
difficulty	The difficulty of a Bot

Postcondition

Bot's name, difficulty or both are modified and the modified bot is returned in JSON format if no exception was triggered. Else, it will return the exception in a string.

Definition at line 319 of file DomainCtrl.java.

```
                                                                                      {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Bot bot = this.playerCtrl.modifyBot(botID, name, difficulty, this.currentUser.getID());
            result.first = bot.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.14 deleteUser()

```
String domain.DomainCtrl.deleteUser (
              String password )
```

Method that, given an ID, a name and a password, allows us to delete a user.

Precondition

currentUser is not null.

Parameters

password	Password of the user

Postcondition

The user is deleted from the repository and we logout from the User.

Definition at line 339 of file DomainCtrl.java.

```
                                        {
        String result = null;

        try {
            this.playerCtrl.deleteUser(this.currentUser.getID(), password);
            this.logout();
        } catch (Exception e) {
            return e.getMessage();
        }

        return result;
    }
```

6.28.3.15 deleteBot()

```
String domain.DomainCtrl.deleteBot (
              UUID botID )
```

Method that, given a name, a botID and a deleterID, allows us to delete a bot.

Precondition

currentUser is not null.

Postcondition

The bot is deleted from the repository.

Parameters

botID	ID of a bot

Definition at line 359 of file DomainCtrl.java.

```
                                   {
        String result = null;

        try {
            this.playerCtrl.deleteBot(botID, this.currentUser.getID());
        } catch (Exception e) {
            return e.getMessage();
        }

        return result;
    }
```

6.28.3.16 viewUser()

`JSONObject domain.DomainCtrl.viewUser ( )`

Method to get the current User data.

Precondition

True

Postcondition

The current User is returned in JSON format.

Definition at line 376 of file DomainCtrl.java.

```
                              {
        if (this.currentUser == null)
            return null;

        return this.currentUser.serialize();
    }
```

6.28.3.17 viewPlayers()

`Pair<JSONObject, JSONObject> domain.DomainCtrl.viewPlayers ( )`

Method to get the current Players(Player1 and 2) data.

Precondition

True

Postcondition

The current Player1 and Player2 are returned in JSON format.

Definition at line 388 of file DomainCtrl.java.

```
                                                  {
        Pair<JSONObject, JSONObject> result = new Pair<JSONObject, JSONObject>(null, null);

        if (this.currentPlayer1 instanceof User)
            result.first = ((User) this.currentPlayer1).serialize();
        else if (this.currentPlayer1 instanceof Bot)
            result.first = ((Bot) this.currentPlayer1).serialize();

        if (this.currentPlayer2 instanceof User)
            result.second = ((User) this.currentPlayer2).serialize();
        else if (this.currentPlayer2 instanceof Bot)
            result.second = ((Bot) this.currentPlayer2).serialize();

        return result;
    }
```

6.28.3.18 createConfiguration()

```
Pair<JSONObject, String> domain.DomainCtrl.createConfiguration (
            String name,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally )
```

Lets the current user to create a new configuration with a name, rules and the initial board.

Precondition

CurrentUser and currentBoard is not null

Parameters

name	Name of the Configuration
canEatHorizontally	Boolean that determines if you can capture pieces in a horizontal manner.
canEatVertically	Boolean that determines if you can capture pieces in a vertical manner.
canEatDiagonally	Boolean that determines if you can capture pieces in a diagonal manner.

Postcondition

The created Configuration is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 415 of file DomainCtrl.java.

```
                                                                   {
            Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

            try {
                Configuration configuration = new Configuration(name, this.currentUser.getID(),
        canEatHorizontally,
                        canEatVertically, canEatDiagonally);
                configuration.setCanEatHorizontally(canEatHorizontally); // To ensure raising the rules
        exception
                configuration.setCanEatVertically(canEatVertically);
                configuration.setCanEatDiagonally(canEatDiagonally);
                this.boardCtrl.isValid(this.currentBoard, configuration);
                configuration = this.configurationCtrl.create(name, canEatHorizontally, canEatVertically,
        canEatDiagonally,
                        this.currentBoard, this.currentUser.getID());
                result.first = configuration.serialize();
            } catch (Exception e) {
                return new Pair<JSONObject, String>(null, e.getMessage());
            }

            return result;
        }
```

6.28.3.19 createInitialBoard()

`String domain.DomainCtrl.createInitialBoard ( )`

Lets the current user create a default initial board to start modifying it in the configuration's creation.

Precondition

True

Postcondition

The exception is returned in a String if any are triggered.

Definition at line 441 of file DomainCtrl.java.

```
                                         {
            String result = null;

            try {
                this.currentBoard = new Board();
            } catch (Exception e) {
                return e.getMessage();
            }

            return result;
        }
```

6.28.3.20 modifyConfiguration()

```
Pair<JSONObject, String> domain.DomainCtrl.modifyConfiguration (
            String name,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally )
```

Lets the current user modify a configuration he/she created. The configuration's rules and the initial board can be changed.

Precondition

CurrentUser is not null.

Parameters

canEatHorizontally	Boolean that determines if you can capture pieces in a horizontal manner.
canEatVertically	Boolean that determines if you can capture pieces in a vertical manner.
canEatDiagonally	Boolean that determines if you can capture pieces in a diagonal manner.

Postcondition

The modified Configuration is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 461 of file DomainCtrl.java.

```
                                                          {
       Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

       try {
           Configuration configuration = new Configuration(name, this.currentUser.getID(),
  canEatHorizontally,
                   canEatVertically, canEatDiagonally);
           configuration.setCanEatHorizontally(canEatHorizontally); // To ensure raising the rules
  exception
           configuration.setCanEatVertically(canEatVertically);
           configuration.setCanEatDiagonally(canEatDiagonally);
           this.boardCtrl.isValid(this.currentBoard, configuration);
           configuration = this.configurationCtrl.modify(name, canEatHorizontally, canEatVertically,
  canEatDiagonally,
                   this.currentBoard, this.currentUser.getID());
           result.first = configuration.serialize();
       } catch (Exception e) {
           return new Pair<JSONObject, String>(null, e.getMessage());
       }

       return result;
   }
```

6.28.3.21 modifyInitialBoard()

```
String domain.DomainCtrl.modifyInitialBoard (
            String name )
```

Lets the current user to modify the initial board of a configuration he/she created.

Precondition

True

Parameters

name	Name of the Configuration

Postcondition

The exception is returned in a String if any are triggered.

Definition at line 488 of file DomainCtrl.java.

```
                                                  {
        String result = null;

        try {
            this.currentBoard = this.configurationCtrl.getInitialBoard(name);
        } catch (Exception e) {
            return e.getMessage();
        }

        return result;
    }
```

6.28.3.22 deleteConfiguration()

```
String domain.DomainCtrl.deleteConfiguration (
              String name )
```

Lets the current user delete a configuration he/she created if it has not been used.

Precondition

currentUser is not null.

Parameters

name	Name of the Configuration

Postcondition

The exception is returned in a String if any is triggered.

Definition at line 506 of file DomainCtrl.java.

```
                                                   {
        String result = null;

        try {
            this.configurationCtrl.delete(name, this.currentUser.getID());
        } catch (Exception e) {
            return e.getMessage();
        }

        return result;
    }
```

6.28.3.23 getConfiguration()

```
Pair<JSONObject, String> domain.DomainCtrl.getConfiguration (
            String name )
```

Returns the configuration identified by the name.

Precondition

True

Parameters

name	Name of the Configuration

Postcondition

The Configuration defined by the name is returned in JSON Format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 524 of file DomainCtrl.java.

```
                                                                {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Configuration configuration = this.configurationCtrl.getConfiguration(name);
            result.first = configuration.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.24 listConfigurations()

```
Pair<ArrayList<String>, String> domain.DomainCtrl.listConfigurations ( )
```

Returns a list of all configurations names in the system.

Precondition

True

Postcondition

The list of Configuration names is returned in an Array of Strings if no exception is triggered. Else, the exception is returned in a String.

Definition at line 542 of file DomainCtrl.java.

```
                                                              {
        Pair<ArrayList<String>, String> result = new Pair<ArrayList<String>, String>(null, null);

        try {
            result.first = this.configurationCtrl.list();
        } catch (Exception e) {
            return new Pair<ArrayList<String>, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.25 getInitialBoard()

```
Pair<JSONObject, String> domain.DomainCtrl.getInitialBoard (
            String name )
```

Get an Initial Board of a Configuration.

Precondition

True

Parameters

name	Name of the Configuration

Postcondition

The initial board of the Configuration identified by name is returned in JSON Format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 560 of file DomainCtrl.java.

```
                                                      {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Board initialBoard = this.configurationCtrl.getInitialBoard(name);
            result.first = initialBoard.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.26 viewConfiguration()

```
JSONObject domain.DomainCtrl.viewConfiguration ( )
```

Method to get the current Configuration data.

Precondition

True

Postcondition

The current Configuration is returned in JSON format.

Definition at line 578 of file DomainCtrl.java.

```
                                        {
        if (this.currentConfiguration == null)
            return null;

        return this.currentConfiguration.serialize();
    }
```

6.28.3.27 createGame()

```
Pair<JSONObject, String> domain.DomainCtrl.createGame (
            UUID player1ID,
            UUID player2ID,
            String configurationName )
```

Lets the current user create a new game, selecting both players and a configuration of rules and initial board.

Precondition

CurrentUser is not null

Parameters

player1ID	UUID of Player1
player2ID	UUID of Player1
configurationName	Name of a Configuration

Postcondition

The created Game is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 595 of file DomainCtrl.java.

```
    {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Game game = this.gameCtrl.create(player1ID, player2ID, configurationName,
    this.currentUser.getID());
            result.first = game.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
     }
```

6.28.3.28 saveGame()

```
Pair<JSONObject, String> domain.DomainCtrl.saveGame ( )
```

Lets the current user manually save the current game and playing board state.

Precondition

CurrentUser, currentGame and currentBoard are not null

Postcondition

The current Game is saved and returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 613 of file DomainCtrl.java.

```
                                   {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Game game = this.gameCtrl.save(this.currentGame, this.currentBoard,
    this.currentUser.getID());
            result.first = game.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.29 getGame()

```
Pair<JSONObject, String> domain.DomainCtrl.getGame (
              String name )
```

Returns the game identified by its name and any of the participant player IDs.

Precondition

CurrentUser is not null

Parameters

name	Name of a Game

Postcondition

Returns the Game identified by its name in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 632 of file DomainCtrl.java.

```
                                             {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Game game = this.gameCtrl.getGame(name, this.currentUser.getID());
            result.first = game.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.30 getPlayingBoard()

```
Pair<JSONObject, String> domain.DomainCtrl.getPlayingBoard (
              String name )
```

Returns the playing board associated with the given game name and any of the participant player IDs.

Precondition

currentUser is not null

Parameters

name	Name of a Game

Postcondition

The playing board associated with the given game name is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 651 of file DomainCtrl.java.

```
                                                            {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Board playingBoard = this.gameCtrl.getPlayingBoard(name, this.currentUser.getID());
            result.first = playingBoard.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.31 listGames()

`Pair<ArrayList<String>, String> domain.DomainCtrl.listGames ( )`

Returns a list of all games names identified by any of the participant player IDs.

Precondition

currentUser is not null

Postcondition

The list of all game names is returned in an ArrayList of Strings if no exception is triggered. Else, the exception is returned in a String.

Definition at line 669 of file DomainCtrl.java.

```
                                                  {
        Pair<ArrayList<String>, String> result = new Pair<ArrayList<String>, String>(null, null);

        try {
            result.first = this.gameCtrl.list(this.currentUser.getID());
        } catch (Exception e) {
            return new Pair<ArrayList<String>, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.32 selectGame()

```
Pair<JSONObject, String> domain.DomainCtrl.selectGame (
                String name )
```

Lets the current user load a selected game by name in order to play it afterwards.

Precondition

currentUser is not null

Parameters

name	Name of a Game

Postcondition

The game selected by its name is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 687 of file DomainCtrl.java.

```
                                                                            {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            Game game = this.gameCtrl.getGame(name, this.currentUser.getID());
            Board board = this.gameCtrl.getPlayingBoard(name, this.currentUser.getID());
            Configuration configuration =
      this.configurationCtrl.getConfiguration(game.getConfigurationName());
            Player player1 = null;
            Player player2 = null;
            try { player1 = this.playerCtrl.getUser(game.getPlayer1ID()); } catch (Exception e) {
      player1 = this.playerCtrl.getBot(game.getPlayer1ID()); }
            try { player2 = this.playerCtrl.getUser(game.getPlayer2ID()); } catch (Exception e) {
      player2 = this.playerCtrl.getBot(game.getPlayer2ID()); }
            this.currentGame = game;
            this.currentBoard = board;
            this.currentConfiguration = configuration;
            this.currentPlayer1 = player1;
            this.currentPlayer2 = player2;
            result.first = game.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.33 play()

`Pair<JSONObject, String> domain.DomainCtrl.play ( )`

Lets the current user start playing on the current loaded game.

Precondition

currentGame is not null

Postcondition

The user starts to play the game and the Game is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 716 of file DomainCtrl.java.

```
                                                 {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            this.currentGame = this.gameCtrl.play(this.currentGame);
            result.first = this.currentGame.serialize();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.34 surrender()

```
Pair<JSONObject, String> domain.DomainCtrl.surrender (
            UUID surrendeeID )
```

Lets a player of the current game surrender, setting the winner as the opponent.

Precondition

currentGame is not null

Parameters

surrendeeID	UUID of the User

Postcondition

The game ends with a winner and the Game is returned in JSON format if no exception is triggered. Else, the exception is returned in a String.

Definition at line 735 of file DomainCtrl.java.

```
                                                                {
        Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);

        try {
            this.currentGame = this.gameCtrl.surrender(this.currentGame, surrendeeID);
            result.first = this.currentGame.serialize();
            this.createEntries();
        } catch (Exception e) {
            return new Pair<JSONObject, String>(null, e.getMessage());
        }

        return result;
    }
```

6.28.3.35 viewGame()

```
JSONObject domain.DomainCtrl.viewGame ( )
```

Method to get the current Game data.

Precondition

True

Postcondition

The current Game is returned in JSON format.

Definition at line 754 of file DomainCtrl.java.

```
                                {
        if (this.currentGame == null)
            return null;

        return this.currentGame.serialize();
    }
```

6.28.3.36 stringToPieceType()

```
PieceType domain.DomainCtrl.stringToPieceType (
              String pieceType )  [private]
```

Method to convert an String from the presentation level to a PieceType in order to decouple domain specific knowledgement.

Precondition

True

Parameters

pieceType	String that represents a pieceType

Postcondition

The corresponding PieceType will be returned.

Definition at line 769 of file DomainCtrl.java.

```
                                                          {
        if (pieceType == null)
            return null;

        if (pieceType.equals(PieceType.PLAYER1.toString()))
            return PieceType.PLAYER1;

        if (pieceType.equals(PieceType.PLAYER2.toString()))
            return PieceType.PLAYER2;

        return null;
    }
```

6.28.3.37 getBestPosition()

```
Pair<Integer, Integer> domain.DomainCtrl.getBestPosition (
              Integer difficulty,
              String myPieceType )
```

Returns the next best possible position, or null if none, to place a piece on the current for a given player. It forwards the placePiece request to the correct algorithm depending on the difficulty. This method can be used to implement the assisted mode.

Precondition

currentBoard and currentConfiguration is not null

Parameters

difficulty	Difficulty of the Bot
myPieceType	String that represents a pieceType

Postcondition

The best position will be returned.

Definition at line 790 of file DomainCtrl.java.

```
                                                                                    {
        return this.difficultyCtrl.getBestPosition(difficulty, this.currentConfiguration,
    this.currentBoard,
              this.stringToPieceType(myPieceType));
    }
```

6.28.3.38 placePieceConfig()

```
JSONObject domain.DomainCtrl.placePieceConfig (
            Pair< Integer, Integer > position,
            String myPieceType )
```

Modifying method returns the board modified with the added position in JSON format.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7).

Postcondition

currentBoard is modified and is returned in JSONObject format.

Parameters

myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.

Definition at line 805 of file DomainCtrl.java.

```
                                                                                      {
        this.currentBoard = this.boardCtrl.placePieceConfig(this.currentBoard, position,
              this.stringToPieceType(myPieceType));
        return this.currentBoard.serialize();
    }
```

6.28.3.39 removePiece()

```
JSONObject domain.DomainCtrl.removePiece (
            Pair< Integer, Integer > position )
```

Modifying method that removes a piece from currentBoard and returns the Board in JSON format.

Precondition

The *position* parameter isn't null and has values between (0,0) and (7,7).

Postcondition

currentBoard is modified and is returned in JSONObject format.

Parameters

position	Pair<Integer,Integer> that represents a position in a board.

Definition at line 819 of file DomainCtrl.java.

```
                                                                {
        this.currentBoard = this.boardCtrl.removePiece(this.currentBoard, position);
        return this.currentBoard.serialize();
    }
```

6.28.3.40 getNumPieces()

`Pair<Integer, Integer> domain.DomainCtrl.getNumPieces ( )`

Get method that returns the value of the *board* parameter's *PiecesPlayer1* and *PiecesPlayer2* attributes.

Precondition

True

Postcondition

The attributes *piecesPlayer1* and *PiecesPlayer2* of the currentBoard are returned in the first and second space of a Pair, respectively.

Definition at line 832 of file DomainCtrl.java.

```
                                              {
        return this.boardCtrl.getNumPieces(this.currentBoard);
    }
```

6.28.3.41 validPositions()

```
ArrayList<Pair<Integer, Integer> > domain.DomainCtrl.validPositions (
              String myPieceType )
```

Method that returns an Array of the valid positions in *board* of the player *myPieceType* taking into consideration the Configuration of the currentGame.

Precondition

All parameters aren't null.

Postcondition

An Array of valid positions(Pair<Integer,Integer>) is returned.

A valid position in a board is one which it's cell state is equal to null (meaning an empty cell) and there is at least one opponent PieceType surrounding that position (go to surroundingPieces to crystalize what the surrounding areas of a position are).

Parameters

myPieceType	PieceType variable that represents the player in a cell.

Definition at line 851 of file DomainCtrl.java.

```
                                                                                      {
        return this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
                this.stringToPieceType(myPieceType));
    }
```

6.28.3.42 isValidBoard()

`String domain.DomainCtrl.isValidBoard ( )`

Method that warns us if an instance of the *board* parameters is invalid.

An invalid Board means that no player can add a piece in the current state of the implicit parameter's *board* attribute.

Precondition

All parameters aren't null.

Postcondition

If the currentBoard is invalid, InvalidBoardException will be thrown, else a null String will be returned.

Definition at line 867 of file DomainCtrl.java.

```
                                  {
        String result = null;

        try {
            this.boardCtrl.isValid(this.currentBoard, this.currentConfiguration);
        } catch (Exception e) {
            return e.getMessage();
        }

        return result;
    }
```

6.28.3.43 placePiece()

```
Pair<Pair<JSONObject, String>, String> domain.DomainCtrl.placePiece (
            Pair< Integer, Integer > position,
            UUID playerID,
            String pieceType )
```

Modifying method that adds a piece in the *board* parameter.

In addition, it applies the effect of adding that piece in the board by changing the pieces of the board taking into consideration the Configuration given.

Precondition

Parameters aren't null and *position* is between values (0,0) and (7,7). There is an exception if the playerID corresponds to a bot, then the *position* parameter CAN BE null. currentGame, currentConfiguration, current←Board, currentPlayer1, currentPlayer2

Postcondition

If the playerID is a bot, automatically the best position is calculated for that board state. If the playerID is a user, it will add a piece with the position given. The modified current board is then returned in a JSON format ready to be played by the opponent if no exception is triggered. Else, the exception will be returned in a string.

Parameters

board	Instance of a Board class which is the one we will modify and return.
myPieceType	PieceType variable that represents the player in a cell.
position	Pair<Integer,Integer> that represents a position in a board.
canEatHorizontally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Horizontal manner..
canEatVertically	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Vertical manner.
canEatDiagonally	Boolean value from Configuration that determines if we can capture pieces of the *myPieceType* opponent in a Diagonal manner.

Definition at line 908 of file DomainCtrl.java.

```
                              {
        ArrayList<Pair<Integer, Integer» validPos1 = new ArrayList<Pair<Integer, Integer»();
        ArrayList<Pair<Integer, Integer» validPos2 = new ArrayList<Pair<Integer, Integer»();

        PieceType myPieceType = this.stringToPieceType(pieceType);

        try {
            this.gameCtrl.checkPlaceRights(this.currentGame, playerID, myPieceType);

            validPos1 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration, myPieceType);
            if (validPos1.isEmpty()) {
                validPos2 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
                        this.inversePieceType(myPieceType));
                if (validPos2.isEmpty())
                    return this.finishGame();
                return this.nextTurn();
            }

            Player player = (this.currentPlayer1.getID().equals(playerID) ? this.currentPlayer1 : this.currentPlayer2);
            if (player instanceof Bot) {
                Bot bot = ((Bot) player);
                position = this.difficultyCtrl.getBestPositionByBot(bot, this.currentConfiguration, this.currentBoard,
                        myPieceType);
            }

            if (!validPos1.contains(position))
                throw new InvalidPositionException();

            this.currentBoard = this.boardCtrl.placePiece(this.currentBoard, this.currentConfiguration, myPieceType,
                    position);

            validPos2 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
                    this.inversePieceType(myPieceType));
            if (!validPos2.isEmpty())
                return this.nextTurn();

            validPos1 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration, myPieceType);
            if (validPos1.isEmpty())
                return this.finishGame();

            return this.currentTurn();
        } catch (Exception e) {
            return new Pair<Pair<JSONObject, String>, String>(null, e.getMessage());
        }
    }
```

6.28.3.44 inversePieceType()

```
PieceType domain.DomainCtrl.inversePieceType (
              PieceType pieceType )  [private]
```

Private method that inverts the Player's pieceType in order to get its opponent.

Precondition

pieceType is not null

Parameters

PieceType	PieceType variable that represents the player in a cell.

Postcondition

The opponent's PieceType is returned

Definition at line 961 of file DomainCtrl.java.

```
                                                          {
        return (pieceType == PieceType.PLAYER2 ? PieceType.PLAYER1 : PieceType.PLAYER2);
    }
```

6.28.3.45 finishGame()

`Pair<Pair<JSONObject, String>, String> domain.DomainCtrl.finishGame ( ) [private]`

Lets the system to automatically finish the game when any players win or when there aren't any valid positions left to place a piece on the board, setting the winner of the game or setting that the game has ended in a draw.

Precondition

currentGame and currentBoard aren't null

Postcondition

The game is finised and returned in JSON format if there is no exception triggered. Else, the exception is returned in a String.

Definition at line 970 of file DomainCtrl.java.

```
                                                              {
        Pair<Integer, Integer> numPieces = this.boardCtrl.getNumPieces(this.currentBoard);

        try {
            if (numPieces.first > numPieces.second) {
                this.currentGame = this.gameCtrl.finish(this.currentGame,
     this.currentGame.getPlayer1ID());
                this.createEntries();
                return new Pair<Pair<JSONObject, String>, String>(
                        new Pair<JSONObject, String>(this.currentBoard.serialize(), null),
                        new FinishedGameException().getMessage());
            } else if (numPieces.second > numPieces.first) {
                this.currentGame = this.gameCtrl.finish(this.currentGame,
     this.currentGame.getPlayer2ID());
                this.createEntries();
                return new Pair<Pair<JSONObject, String>, String>(
                        new Pair<JSONObject, String>(this.currentBoard.serialize(), null),
                        new FinishedGameException().getMessage());
            } else {
                this.currentGame = this.gameCtrl.finish(this.currentGame, null);
                this.createEntries();
                return new Pair<Pair<JSONObject, String>, String>(
                        new Pair<JSONObject, String>(this.currentBoard.serialize(), null),
                        new FinishedGameException().getMessage());
            }
        } catch (Exception e) {
            return new Pair<Pair<JSONObject, String>, String>(null, e.getMessage());
        }
    }
```

6.28.3.46 nextTurn()

```
Pair<Pair<JSONObject, String>, String> domain.DomainCtrl.nextTurn ( ) [private]
```

Lets the system to automatically pass the turn of the current game.

Precondition

currentGame is not null.

Postcondition

he Game is returned with the next turn in JSON format if there is no exception triggered. Else, the exception is returned in a String.

Definition at line 1003 of file DomainCtrl.java.

```
                                                                {
        try {
            this.currentGame = this.gameCtrl.nextTurn(this.currentGame);
        } catch (Exception e) {
            return new Pair<Pair<JSONObject, String>, String>(null, e.getMessage());
        }
        return this.currentTurn();
    }
```

6.28.3.47 currentTurn()

```
Pair<Pair<JSONObject, String>, String> domain.DomainCtrl.currentTurn ( ) [private]
```

Lets the system to automatically decide the current turn of the current game.

Precondition

currentGame and currentBoard aren't not null.

Postcondition

The Game is returned with the current turn in JSON format if there is no exception triggered. Else, the exception is returned in a String.

Definition at line 1017 of file DomainCtrl.java.

```
                                                                   {
        return new Pair<Pair<JSONObject, String>, String>(
                new Pair<JSONObject, String>(this.currentBoard.serialize(),
   this.currentGame.getTurn().toString()),
                null);
    }
```

6.28.3.48 viewBoard()

```
JSONObject domain.DomainCtrl.viewBoard ( )
```

Method to get the current Board data.

Precondition

True

Postcondition

The current Board is returned in JSON format.

Definition at line 1028 of file DomainCtrl.java.

```
                                   {
        if (this.currentBoard == null)
            return null;

        return this.currentBoard.serialize();
    }
```

6.28.3.49 getRanking()

```
JSONObject domain.DomainCtrl.getRanking (
              String name )
```

Returns the ranking identified by name.

Precondition

True

Parameters

name	of a Ranking

Postcondition

The ranking identified by name is returned in JSON format.

Definition at line 1043 of file DomainCtrl.java.

```
                                               {
        return this.rankingCtrl.getRanking(name).serialize();
    }
```

6.28.3.50 listRankings()

```
ArrayList<String> domain.DomainCtrl.listRankings ( )
```

Returns a list of all ranking names in the system.

Precondition

True

Postcondition

The list of ranking names is returned in an ArrayList of strings.

Definition at line 1052 of file DomainCtrl.java.

```
                                                    {
        return this.rankingCtrl.listRankings();
    }
```

6.28.3.51 listRecords()

```
ArrayList<Pair<String, JSONObject> > domain.DomainCtrl.listRecords ( )
```

Returns the entries with the highest score of the current user for each ranking in the system.

Precondition

currentUser isn't null

Postcondition

The list of records and its ranking names is returned in an ArrayList of JSONObjects and Strings.

Definition at line 1061 of file DomainCtrl.java.

```
                                                                        {
        ArrayList<Pair<String, JSONObject»> result = new ArrayList<Pair<String, JSONObject»();
        ArrayList<Pair<String, Entry» records = this.rankingCtrl.listRecords(this.currentUser.getID());

        for (Pair<String, Entry> record : records)
            result.add(new Pair<String, JSONObject>(record.first, record.second.serialize()));

        return result;
    }
```

6.28.3.52 createEntries()

```
void domain.DomainCtrl.createEntries ( ) [private]
```

Lets the system to automatically create the entries of the associated ranking when the current user finishes a game.

Precondition

currentGame, currentConfiguration, currentPlayer1, currentPlayer2 and currentBoard aren't null

Postcondition

The entries are created in the system.

Definition at line 1076 of file DomainCtrl.java.

```
                                  {
        ArrayList<String> rules = new ArrayList<String>();

        if (this.currentConfiguration.getCanEatHorizontally())
            rules.add("horizontally");

        if (this.currentConfiguration.getCanEatVertically())
            rules.add("vertically");

        if (this.currentConfiguration.getCanEatDiagonally())
            rules.add("diagonally");

        String rankingName;

        if (this.currentGame.getWinnerID() != null) {
            Player winner = null;
            Player loser = null;

            if (this.currentGame.getWinnerID().equals(this.currentPlayer1.getID())) {
                winner = this.currentPlayer1;
                loser = this.currentPlayer2;
            } else {
                winner = this.currentPlayer2;
                loser = this.currentPlayer1;
            }

            // Games won vs <Player>
            rankingName = String.format("Games won vs %s", loser.getName());
            this.rankingCtrl.createEntry(rankingName, winner.getID(), 1, RankingType.INCREMENTAL);

            // Games won with <Rules>
            rankingName = String.format("Games won with %s rules", String.join(", ", rules));
            this.rankingCtrl.createEntry(rankingName, winner.getID(), 1, RankingType.INCREMENTAL);

            // Games lost with <Rules>
            rankingName = String.format("Games lost with %s rules", String.join(", ", rules));
            this.rankingCtrl.createEntry(rankingName, loser.getID(), 1, RankingType.INCREMENTAL);
        } else {
            // Games tied with <Rules>
            rankingName = String.format("Games tied with %s rules", String.join(", ", rules));
            this.rankingCtrl.createEntry(rankingName, this.currentPlayer1.getID(), 1,
      RankingType.INCREMENTAL);
            this.rankingCtrl.createEntry(rankingName, this.currentPlayer2.getID(), 1,
      RankingType.INCREMENTAL);
        }

        // Maximum pieces obtained in a game with <Rules>
        rankingName = String.format("Maximum pieces obtained in a game with %s rules", String.join(",
      ", rules));
        this.rankingCtrl.createEntry(rankingName, currentPlayer1.getID(),
      this.currentBoard.getPiecesPlayer1(),
                RankingType.UNIQUE);
        this.rankingCtrl.createEntry(rankingName, currentPlayer2.getID(),
      this.currentBoard.getPiecesPlayer2(),
                RankingType.UNIQUE);
    }
```

6.28.4 Member Data Documentation

6.28.4.1 playerCtrl

`PlayerCtrl domain.DomainCtrl.playerCtrl [private]`

Player Controller.

Definition at line 35 of file DomainCtrl.java.

6.28.4.2 configurationCtrl

```
ConfigurationCtrl domain.DomainCtrl.configurationCtrl  [private]
```

Configuration Controller.

Definition at line 39 of file DomainCtrl.java.

6.28.4.3 boardCtrl

```
BoardCtrl domain.DomainCtrl.boardCtrl  [private]
```

Board Controller.

Definition at line 43 of file DomainCtrl.java.

6.28.4.4 gameCtrl

```
GameCtrl domain.DomainCtrl.gameCtrl  [private]
```

Game Controller.

Definition at line 47 of file DomainCtrl.java.

6.28.4.5 rankingCtrl

```
RankingCtrl domain.DomainCtrl.rankingCtrl  [private]
```

Ranking Controller.

Definition at line 51 of file DomainCtrl.java.

6.28.4.6 difficultyCtrl

```
DifficultyCtrl domain.DomainCtrl.difficultyCtrl  [private]
```

Difficulty Controller.

Definition at line 55 of file DomainCtrl.java.

6.28.4.7 currentUser

```
User domain.DomainCtrl.currentUser  [private]
```

Current logged User.

Definition at line 59 of file DomainCtrl.java.

6.28.4.8 currentPlayer1

```
Player domain.DomainCtrl.currentPlayer1  [private]
```

Player 1 of the current game. Can be either a User or a Bot.

Definition at line 63 of file DomainCtrl.java.

6.28.4.9 currentPlayer2

```
Player domain.DomainCtrl.currentPlayer2  [private]
```

Player 2 of the current game. Can be either a User or a Bot.

Definition at line 67 of file DomainCtrl.java.

6.28.4.10 currentBoard

```
Board domain.DomainCtrl.currentBoard  [private]
```

Current loaded board from the current configuration or game.

Definition at line 71 of file DomainCtrl.java.

6.28.4.11 currentConfiguration

```
Configuration domain.DomainCtrl.currentConfiguration  [private]
```

Current loaded configuration.

Definition at line 75 of file DomainCtrl.java.

6.28.4.12 currentGame

```
Game domain.DomainCtrl.currentGame [private]
```

Current loaded game.

Definition at line 79 of file DomainCtrl.java.

The documentation for this class was generated from the following file:

- DomainCtrl.java

6.29 test.driver.Driver Class Reference

Implements various utilities to create a driver application. By Alex Rodriguez.

Static Public Member Functions

- static String menu (String title, String name, Pair< String, String >... options)

  Print to standard output a menu with the list of options given and show a prompt asking to select one.
- static void pause ()

  Pause the driver application until enter is pressed.
- static void clear ()

  Clear the console.
- static String input (String prompt)

  Prompt the user and return the entered value as String.
- static Integer inputInt (String prompt)

  Prompt the user and return the entered value as Integer.
- static boolean inputBool (String prompt)

  Prompt the user and return the entered value as boolean.

6.29.1 Detailed Description

Implements various utilities to create a driver application. By Alex Rodriguez.

Definition at line 17 of file Driver.java.

6.29.2 Member Function Documentation

6.29.2.1 menu()

```
static String test.driver.Driver.menu (
            String title,
            String name,
            Pair< String, String >... options ) [static]
```

Print to standard output a menu with the list of options given and show a prompt asking to select one.

Precondition

True

Postcondition

A menu with the options specified is printed to standard output and a prompt asking to select an option is shown. It returns the identifier of the selected option or terminates the driver application if the option was "e".

Parameters

title	A text inserted before the printed menu.
name	The name of the shown menu.
options	List of options to show.

Returns

The identifier of the selected option.

Definition at line 29 of file Driver.java.

```
                                                                                  {
        String selected = new String();

        if (name == null)
            name = "Options";

        do {
            Driver.clear();
            if (title != null)
                System.out.println(title);
            System.out.println(String.format("==== %s ====", name));
            for (Pair<String, String> option : options)
                System.out.println(String.format("[%s]\t%s", option.first, option.second));
            System.out.println("[e]\tExit driver\n");
            selected = Driver.input("What do you want to do?");
            Driver.clear();

            for (Pair<String, String> option : options)
                if (selected.equals(option.first))
                    return selected;

        } while (!selected.equals("e") && !selected.equals("E"));

        System.exit(0);

        return null;
    }
```

6.29.2.2 pause()

`static void test.driver.Driver.pause ( )  [static]`

Pause the driver application until enter is pressed.

Precondition

True

Postcondition

The driver application is paused until enter is pressed.

Definition at line 62 of file Driver.java.

```
                                {
        Driver.input("Press enter to continue");
    }
```

6.29.2.3 clear()

```
static void test.driver.Driver.clear ( )  [static]
```

Clear the console.

Precondition

True

Postcondition

The console is cleared.

Definition at line 71 of file Driver.java.

```
                               {
        System.out.print("\033\143");
    }
```

6.29.2.4 input()

```
static String test.driver.Driver.input (
             String prompt )  [static]
```

Prompt the user and return the entered value as String.

Precondition

True

Postcondition

A prompt is shown waiting for user input from stdin.

Parameters

prompt	The text of the shown prompt.

Returns

The entered value as String.

Definition at line 82 of file Driver.java.

```
                                             {
        String in = new String();
        System.out.print(String.format("» %s: ", prompt));
        Scanner stdin = new Scanner(System.in);

        try {
            in = stdin.nextLine();
        } catch (Exception e) {
            stdin.close();
        }

        return in;
    }
```

6.29.2.5 inputInt()

```
static Integer test.driver.Driver.inputInt (
              String prompt )  [static]
```

Prompt the user and return the entered value as Integer.

Precondition

True

Postcondition

A prompt is shown waiting for user input from stdin.

Parameters

prompt	The text of the shown prompt.

Returns

The entered value as Integer.

Definition at line 103 of file Driver.java.

```
                                          {
        boolean trick = true; // Necessary or Java won't compile...
        do {
            try {
                return Integer.parseInt(Driver.input(prompt));
            } catch (NumberFormatException e) {
                System.out.println("That is not an integer!");
            }
        } while (trick);

        return 0;
    }
```

6.29.2.6 inputBool()

```
static boolean test.driver.Driver.inputBool (
              String prompt )  [static]
```

Prompt the user and return the entered value as boolean.

Precondition

True

Postcondition

A prompt is shown waiting for user input from stdin.

Parameters

prompt	The text of the shown prompt.

Returns

The entered value as boolean.

Definition at line 123 of file Driver.java.

```
                                                    {
        boolean trick = true; // Necessary or Java won't compile...
        String in = new String();
        do {
            in = Driver.input(String.format("%s [y/n]", prompt));
            if (in.toLowerCase().equals("yes") || in.toLowerCase().equals("y"))
                return true;
            else if (in.toLowerCase().equals("no") || in.toLowerCase().equals("n"))
                return false;
            System.out.println("That is not a yes or no!");
        } while (trick);

        return false;
    }
```

The documentation for this class was generated from the following file:

- Driver.java

6.30 cmd.driver.easyDifficulty Class Reference

EasyDifficulty driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)

  EasyDifficulty driver main function. Creates an instance of the EasyDifficulty driver and starts it.

6.30.1 Detailed Description

EasyDifficulty driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file easyDifficulty.java.

6.30.2 Member Function Documentation

6.30.2.1 main()

```
static void cmd.driver.easyDifficulty.main (
              String[] args )  [static]
```

EasyDifficulty driver main function. Creates an instance of the EasyDifficulty driver and starts it.

Precondition

True.

Postcondition

The EasyDifficulty driver has started.

Definition at line 22 of file easyDifficulty.java.

```
                                            {
        new EasyDifficultyDriver().start();
    }
```

The documentation for this class was generated from the following file:

- easyDifficulty.java

6.31 domain.EasyDifficulty Class Reference

Implements the Minimax algorithm to get the next best possible position for a given player. By Manuel Navid.

Public Member Functions

- EasyDifficulty (Integer difficulty, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally, PieceType pieceType)

  Create a EasyDifficulty instance.
- Pair< Integer, Integer > place (PieceType[ ][ ] playingBoard)

  Get the next best possible position for the implicit player.

Private Member Functions

- int evaluation (Board currentBoard)

  Get the heuristic evaluation for the given Board state.
- int minimax (Board currentBoard, PieceType currentPieceType, int depth)

  Recursive implementation of the Minimax algorithm.

Additional Inherited Members

6.31.1 Detailed Description

Implements the Minimax algorithm to get the next best possible position for a given player. By Manuel Navid.

Definition at line 18 of file EasyDifficulty.java.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 EasyDifficulty()

```
domain.EasyDifficulty.EasyDifficulty (
            Integer difficulty,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally,
            PieceType pieceType )
```

Create a EasyDifficulty instance.

Precondition

The given difficulty is a positive number. The given rules are not all false.

Postcondition

An EasyDifficulty instance is created and its implicits difficulty, canEatHorizontally, canEatVertically, canEat↩ Diagonally and pieceType attributes are setted. The implicit maxDepth attribute is setted to the double of the entered difficulty.

Parameters

difficulty	Difficulty for the Minimax algorithm.
canEatHorizontally	Whether the pieces of the current Game can be eaten horizontally.
canEatVertically	Whether the pieces of the current Game can be eaten vertically.
canEatDiagonally	Whether the pieces of the current Game can be eaten diagonally.
pieceType	Player that wants to be maximized.

Definition at line 34 of file EasyDifficulty.java.

```
                                                {
        super(difficulty, canEatHorizontally, canEatVertically, canEatDiagonally, pieceType);
    }
```

6.31.3 Member Function Documentation

6.31.3.1 evaluation()

```
int domain.EasyDifficulty.evaluation (
            Board currentBoard )  [private]
```

Get the heuristic evaluation for the given Board state.

Precondition

True

Postcondition

It is returned the heuristic evaluation for the given Board state. The evaluation is the subtraction of the maximized player's control of the board minus the control of the board for the opponent. A player's control of the board is obtained with the number of pieces in his control and adding or subtracting to that based on important positions in the board. Those important positions are corners, positions adjacent to corners, borders of the board which aren't adjacent to corners and positions adjacent to the centre square of the board.

Parameters

currentBoard	Current playing Board to get the heuristic evaluation from.

Returns

The heuristic evaluation for the given Board state.

Definition at line 51 of file EasyDifficulty.java.

```
                                        {
        int player1 = currentBoard.getPiecesPlayer1();
        int player2 = currentBoard.getPiecesPlayer2();

        PieceType[][] board = currentBoard.getBoard();

        // Check corners of the Board
        if (board[0][0] == PieceType.PLAYER1) player1 += 50;
        else if (board[0][0] == PieceType.PLAYER2) player2 += 50;

        if (board[0][7] == PieceType.PLAYER1) player1 += 50;
        else if (board[0][7] == PieceType.PLAYER2) player2 += 50;

        if (board[7][0] == PieceType.PLAYER1) player1 += 50;
        else if (board[7][0] == PieceType.PLAYER2) player2 += 50;

        if (board[7][7] == PieceType.PLAYER1) player1 += 50;
        else if (board[7][7] == PieceType.PLAYER2) player2 += 50;

        // Check borders not next to corner
        for (int k = 2; k < 6; ++k) {
            if (board[k][0] == PieceType.PLAYER1) player1 += 17;
            else if (board[k][0] == PieceType.PLAYER2) player2 += 17;

            if (board[k][7] == PieceType.PLAYER1) player1 += 17;
            else if (board[k][7] == PieceType.PLAYER2) player2 += 17;

            if (board[0][k] == PieceType.PLAYER1) player1 += 17;
            else if (board[0][k] == PieceType.PLAYER2) player2 += 17;

            if (board[7][k] == PieceType.PLAYER1) player1 += 17;
            else if (board[7][k] == PieceType.PLAYER2) player2 += 17;
        }

        // Check next to center of the Board
        for (int i = 2; i < 6; ++i) {
            if (board[i][2] == PieceType.PLAYER1) player1 += 10;
            else if (board[i][2] == PieceType.PLAYER2) player2 += 10;

            if (board[i][5] == PieceType.PLAYER1) player1 += 10;
            else if (board[i][5] == PieceType.PLAYER2) player2 += 10;

            if (board[2][i] == PieceType.PLAYER1) player1 += 10;
            else if (board[2][i] == PieceType.PLAYER2) player2 += 10;

            if (board[5][i] == PieceType.PLAYER1) player1 += 10;
            else if (board[5][i] == PieceType.PLAYER2) player2 += 10;
        }

        // Check next to corners
        for (int j = 0; j < 2; ++j) {
            if (board[1][j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[1][j] == PieceType.PLAYER2) player2 -= 25;

            if (board[1][7 - j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[1][7 - j] == PieceType.PLAYER2) player2 -= 25;

            if (board[6][j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[6][j] == PieceType.PLAYER2) player2 -= 25;

            if (board[6][7 - j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[6][7 - j] == PieceType.PLAYER2) player2 -= 25;
        }

        if (board[0][1] == PieceType.PLAYER1) player1 -= 25;
        else if (board[0][1] == PieceType.PLAYER2) player2 -= 25;

        if (board[7][1] == PieceType.PLAYER1) player1 -= 25;
        else if (board[7][1] == PieceType.PLAYER2) player2 -= 25;

        if (board[0][6] == PieceType.PLAYER1) player1 -= 25;
        else if (board[0][6] == PieceType.PLAYER2) player2 -= 25;

        if (board[7][6] == PieceType.PLAYER1) player1 -= 25;
        else if (board[7][6] == PieceType.PLAYER2) player2 -= 25;

        if (this.pieceType == PieceType.PLAYER1) return player1 - player2;
        else return player2 - player1;
    }
```

6.31.3.2 minimax()

```
int domain.EasyDifficulty.minimax (
            Board currentBoard,
            PieceType currentPieceType,
            int depth )  [private]
```

Recursive implementation of the Minimax algorithm.

Precondition

True

Postcondition

It is returned the heuristic evaluation for the current possible position on the tree of possibilities. If there aren't any possible valid positions left or the maximum depth is reached it stops. The implicit player is maximized and the opponent is minimized.

Parameters

currentBoard	current Board in the tree of possibilities.
currentPieceType	current turn in the tree of possibilities.
depth	current depth in the tree of possibilities.

Returns

The heuristic evaluation for the current possible position on the tree of possibilities.

Definition at line 142 of file EasyDifficulty.java.

```
                                                                {
```

```
     ArrayList<Pair<Integer, Integer» validPositions = currentBoard.validPositions(currentPieceType,
             this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);

     if (validPositions.isEmpty() || depth == 0)
         return this.evaluation(currentBoard);

     // Maximizer
     if (currentPieceType == this.pieceType) {
         int max = Integer.MIN_VALUE, currentMax = 0;

         for (Pair<Integer, Integer> position : validPositions) {
             // Make a duplicate in order not to work with the same Board pointer!
             Board board = new Board(currentBoard.getBoard());
             board.placePiece(position, currentPieceType, this.canEatHorizontally,
this.canEatVertically,
                     this.canEatDiagonally);

             currentMax = this.minimax(board, EasyDifficulty.inversePieceType(currentPieceType),
depth - 1);
             if (currentMax > max)
                 max = currentMax;
         }

         return max;
     }

     // Minimizer
     else {
         Integer min = Integer.MAX_VALUE, currentMin = 0;

         for (Pair<Integer, Integer> position : validPositions) {
             // Make a duplicate in order not to work with the same Board pointer!
             Board board = new Board(currentBoard.getBoard());
             board.placePiece(position, currentPieceType, this.canEatHorizontally,
this.canEatVertically,
                     this.canEatDiagonally);

             currentMin = this.minimax(board, EasyDifficulty.inversePieceType(currentPieceType),
depth - 1);
             if (currentMin < min)
                 min = currentMin;
         }

         return min;
     }
 }
```

6.31.3.3 place()

```
Pair<Integer, Integer> domain.EasyDifficulty.place (
        PieceType playingBoard[ ][ ] )
```

Get the next best possible position for the implicit player.

Precondition

True

Postcondition

It is returned the next best possible position for the implicit player, using the Minimax algorithm with the implicit maximum depth, or null if there isn't any.

Parameters

playingBoard	Current playing Board.

Returns

The next best possible position for the implicit player or null if there isn't any.

Reimplemented from domain.Difficulty.

Definition at line 195 of file EasyDifficulty.java.

```
                                                                    {
        Pair<Integer, Integer> bestPosition = null;

        Board initialBoard = new Board(playingBoard);

        ArrayList<Pair<Integer, Integer» validPositions = initialBoard.validPositions(this.pieceType,
                this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);

        int max = Integer.MIN_VALUE, currentMax = 0;

        for (Pair<Integer, Integer> position : validPositions) {
            // Make a duplicate in order not to work with the same Board pointer!
            Board board = new Board(initialBoard.getBoard());
            board.placePiece(position, this.pieceType, this.canEatHorizontally, this.canEatVertically,
                    this.canEatDiagonally);

            currentMax = this.minimax(board, EasyDifficulty.inversePieceType(this.pieceType),
       this.maxDepth - 1);
            if (currentMax > max) {
                max = currentMax;
                bestPosition = position;
            }
        }

        return bestPosition;
    }
```

The documentation for this class was generated from the following file:

- EasyDifficulty.java

6.32 test.driver.EasyDifficultyDriver Class Reference

Implements the different options for the EasyDifficulty driver application. By Manuel Navid.

Public Member Functions

- EasyDifficultyDriver ()
- void start ()

Public Attributes

- EasyDifficulty currentEasyDifficulty
- Board currentBoard
- String nameCurrentBoard
- FixtureRepository fixtureRepository

Private Member Functions

- void mainMenu ()
- void create ()
- void getDifficulty ()
- void getCanEatHorizontally ()
- void getCanEatVertically ()
- void getCanEatDiagonally ()
- void getPieceType ()
- void getMaxDepth ()
- void setMaxDepth ()
- void loadBoard ()
- void printCurrentBoard ()
- void getNextBestPosition ()
- Pair< String, Board > listBoardFixtures ()
- void printBoard (Board board)
- ArrayList< String > transcribeToCharacters (Board board)

Additional Inherited Members

6.32.1 Detailed Description

Implements the different options for the EasyDifficulty driver application. By Manuel Navid.

Definition at line 21 of file EasyDifficultyDriver.java.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 EasyDifficultyDriver()

```
test.driver.EasyDifficultyDriver.EasyDifficultyDriver ( )
```

Definition at line 33 of file EasyDifficultyDriver.java.

```
                                    {
        this.currentEasyDifficulty = null;
        this.fixtureRepository = new FixtureRepository();
    }
```

6.32.3 Member Function Documentation

6.32.3.1 start()

```
void test.driver.EasyDifficultyDriver.start ( )
```

Definition at line 40 of file EasyDifficultyDriver.java.

```
                            {
        while (true) {
            this.mainMenu();
        }
    }
```

6.32.3.2 mainMenu()

```
void test.driver.EasyDifficultyDriver.mainMenu ( )  [private]
```

Definition at line 46 of file EasyDifficultyDriver.java.

```
                               {
        String title = null;
        if (this.currentEasyDifficulty != null)
            title = String.format("Current maximum depth: %s\n",
      this.currentEasyDifficulty.getMaxDepth());
        if (this.currentBoard != null)
            title += String.format("Current Board: %s\n", this.nameCurrentBoard);

        switch (Driver.menu(title, "EasyDifficulty (Minimax) Driver",
                new Pair<String, String>("1", "Create EasyDifficulty"),
                new Pair<String, String>("2", "Get difficulty"),
                new Pair<String, String>("3", "Get canEatHorizontally"),
                new Pair<String, String>("4", "Get canEatVertically"),
                new Pair<String, String>("5", "Get canEatDiagonally"),
                new Pair<String, String>("6", "Get pieceType"),
                new Pair<String, String>("7", "Get maxDepth"),
                new Pair<String, String>("8", "Set maxDepth"),
                new Pair<String, String>("9", "Load Board"),
                new Pair<String, String>("10", "Print Current Board"),
                new Pair<String, String>("11", "Get next best position"))) {
        case "1":
            this.create();
            break;
        case "2":
            this.getDifficulty();
            break;
        case "3":
            this.getCanEatHorizontally();
            break;
        case "4":
            this.getCanEatVertically();
            break;
        case "5":
            this.getCanEatDiagonally();
            break;
        case "6":
            this.getPieceType();
            break;
        case "7":
            this.getMaxDepth();
            break;
        case "8":
            this.setMaxDepth();
            break;
        case "9":
            this.loadBoard();
            break;
        case "10":
            this.printCurrentBoard();
            break;
        case "11":
            this.getNextBestPosition();
            break;
        }
        Driver.pause();
    }
```

6.32.3.3 create()

```
void test.driver.EasyDifficultyDriver.create ( )  [private]
```

Definition at line 102 of file EasyDifficultyDriver.java.

```
                    {
        System.out.println(
                "Take into account that the default maximum depth is the double of the entered
difficulty.\nMinimax with higher depths requires more time to execute. A value of 2 for the
difficulty is reasonable.\n");

        Integer difficulty = Driver.inputInt("Difficulty (positive)?");
        Boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
        Boolean canEatVertically = Driver.inputBool("Can eat vertically?");
        Boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
        PieceType pieceType = null;

        switch (Driver.menu(null, "Select Bot pieces",
                new Pair<String, String>("1", "PLAYER 1 pieces"),
                new Pair<String, String>("2", "PLAYER 2 pieces"))) {
        case "1":
            pieceType = PieceType.PLAYER1;
            break;
        case "2":
            pieceType = PieceType.PLAYER2;
            break;
        }

        this.currentEasyDifficulty = new EasyDifficulty(difficulty, canEatHorizontally,
canEatVertically,
                canEatDiagonally, pieceType);

        System.out.println(String.format("EasyDifficulty with a maximum depth of %s created
successfully!",
                this.currentEasyDifficulty.getMaxDepth()));
    }
```

6.32.3.4 getDifficulty()

```
void test.driver.EasyDifficultyDriver.getDifficulty ( )  [private]
```

Definition at line 130 of file EasyDifficultyDriver.java.

```
                           {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out.println(
                String.format("EasyDifficulty's difficulty is: %s",
this.currentEasyDifficulty.getDifficulty()));
    }
```

6.32.3.5 getCanEatHorizontally()

```
void test.driver.EasyDifficultyDriver.getCanEatHorizontally ( )  [private]
```

Definition at line 140 of file EasyDifficultyDriver.java.

```
                                   {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out.println(String.format("EasyDifficulty's canEatHorizontally is: %s",
                this.currentEasyDifficulty.getCanEatHorizontally()));
    }
```

6.32.3.6 getCanEatVertically()

`void test.driver.EasyDifficultyDriver.getCanEatVertically ( ) [private]`

Definition at line 150 of file EasyDifficultyDriver.java.

```
                                        {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out.println(String.format("EasyDifficulty's canEatVertically is: %s",
                this.currentEasyDifficulty.getCanEatVertically()));
    }
```

6.32.3.7 getCanEatDiagonally()

`void test.driver.EasyDifficultyDriver.getCanEatDiagonally ( ) [private]`

Definition at line 160 of file EasyDifficultyDriver.java.

```
                                        {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out.println(String.format("EasyDifficulty's canEatDiagonally is: %s",
                this.currentEasyDifficulty.getCanEatDiagonally()));
    }
```

6.32.3.8 getPieceType()

`void test.driver.EasyDifficultyDriver.getPieceType ( ) [private]`

Definition at line 170 of file EasyDifficultyDriver.java.

```
                                 {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out
                .println(String.format("EasyDifficulty's pieceType is: %s",
    this.currentEasyDifficulty.getPieceType()));
    }
```

6.32.3.9 getMaxDepth()

`void test.driver.EasyDifficultyDriver.getMaxDepth ( ) [private]`

Definition at line 180 of file EasyDifficultyDriver.java.

```
                                {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out.println(String.format("EasyDifficulty's maxDepth is: %s",
    this.currentEasyDifficulty.getMaxDepth()));
    }
```

6.32.3.10 setMaxDepth()

```
void test.driver.EasyDifficultyDriver.setMaxDepth ( )  [private]
```

Definition at line 189 of file EasyDifficultyDriver.java.

```
                              {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        System.out.println(
                "Take into account that minimax with higher depths requires more time to execute. A
       value of 5 is reasonable.\n");

        this.currentEasyDifficulty.setMaxDepth(Driver.inputInt("Maximum depth (positive)?"));
        System.out.println("EasyDifficulty's maxDepth changed successfully!");
    }
```

6.32.3.11 loadBoard()

```
void test.driver.EasyDifficultyDriver.loadBoard ( )  [private]
```

Definition at line 202 of file EasyDifficultyDriver.java.

```
                            {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        Pair<String, Board> selectedBoard = this.listBoardFixtures();

        this.nameCurrentBoard = selectedBoard.first;
        this.currentBoard = selectedBoard.second;

        System.out.println(String.format("Board %s loaded successfully!\n", this.nameCurrentBoard));
        this.printBoard(this.currentBoard);
    }
```

6.32.3.12 printCurrentBoard()

```
void test.driver.EasyDifficultyDriver.printCurrentBoard ( )  [private]
```

Definition at line 217 of file EasyDifficultyDriver.java.

```
                                    {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        if (this.currentBoard == null) {
            System.out.println("No current Board!");
            return;
        }

        System.out.println(String.format("Board %s printed successfully!\n", this.nameCurrentBoard));
        this.printBoard(this.currentBoard);
    }
```

6.32.3.13 getNextBestPosition()

```
void test.driver.EasyDifficultyDriver.getNextBestPosition ( )  [private]
```

Definition at line 232 of file EasyDifficultyDriver.java.

```
                                          {
        if (this.currentEasyDifficulty == null) {
            System.out.println("No current EasyDifficulty!");
            return;
        }

        if (this.currentBoard == null) {
            System.out.println("No current Board!");
            return;
        }

        System.out.println("Take into account that the state of the current Board won't be globally
modified.\n");

        this.printBoard(this.currentBoard);

        long startTime = System.currentTimeMillis();
        Pair<Integer, Integer> nextBestPosition =
this.currentEasyDifficulty.place(this.currentBoard.getBoard());
        long durationTime = System.currentTimeMillis() - startTime;

        Board tempBoard = new Board(this.currentBoard.getBoard());

        if (nextBestPosition != null) {
            tempBoard.placePiece(nextBestPosition, this.currentEasyDifficulty.getPieceType(),
                    this.currentEasyDifficulty.getCanEatHorizontally(),
                    this.currentEasyDifficulty.getCanEatVertically(),
this.currentEasyDifficulty.getCanEatDiagonally());
            System.out.println(
                    String.format("The best position calculated in %s ms is %s\n", durationTime,
nextBestPosition));
            System.out.println("The addition of the piece would look like this:\n");
            this.printBoard(tempBoard);
        } else {
            System.out.println("There isn't any possible position left to place a piece on.");
        }
    }
```

6.32.3.14 listBoardFixtures()

```
Pair<String, Board> test.driver.EasyDifficultyDriver.listBoardFixtures ( )  [private]
```

Definition at line 266 of file EasyDifficultyDriver.java.

```
                                                     {
        Integer selectedBoard = -1;
        ArrayList<String> listBoards = this.fixtureRepository.listFiles();

        while (selectedBoard < 0 || selectedBoard >= listBoards.size()) {
            Driver.clear();
            System.out.println("==== Available Boards ====\n");

            for (Integer i = 0; i < listBoards.size(); ++i)
                System.out.println(String.format("[%d]\t%s", i, listBoards.get(i)));
            System.out.println("");

            selectedBoard = Driver.inputInt("What Board would you like to load?");
        }

        Driver.clear();

        return new Pair<String, Board>(listBoards.get(selectedBoard),
                new Board(this.fixtureRepository.boardFileToJSON(listBoards.get(selectedBoard))));
    }
```

6.32.3.15 printBoard()

```
void test.driver.EasyDifficultyDriver.printBoard (
            Board board )  [private]
```

Definition at line 287 of file EasyDifficultyDriver.java.

```
287                                           {
288         ArrayList<String> boardCodified = this.transcribeToCharacters(board);
289         System.out.println("       0  1  2  3  4  5  6  7");
290         System.out.println("    -------------------------");
291 
292         for (Integer i = 0; i < 8; ++i) {
293             String row = boardCodified.get(i);
294             System.out.println("  " + i + " |  " + row.charAt(0) + "  " + row.charAt(1) + "  " +
       row.charAt(2) + "  "
295                     + row.charAt(3) + "  " + row.charAt(4) + "  " + row.charAt(5) + "  " + row.charAt(6)
       + "  "
296                     + row.charAt(7) + "  ");
297         }
298         System.out.println("\n");
299     }
```

6.32.3.16 transcribeToCharacters()

```
ArrayList<String> test.driver.EasyDifficultyDriver.transcribeToCharacters (
            Board board )  [private]
```

Definition at line 301 of file EasyDifficultyDriver.java.

```
301                                                                       {
302         ArrayList<String> boardCodified = new ArrayList<String>(8);
303         String operational = "";
304         PieceType[][] current = board.getBoard();
305 
306         for (int i = 0; i < 8; ++i) {
307             operational = "";
308             for (int j = 0; j < 8; ++j) {
309                 if (current[i][j] == PieceType.PLAYER1)
310                     operational = operational + "B";
311                 if (current[i][j] == PieceType.PLAYER2)
312                     operational = operational + "N";
313                 if (current[i][j] == null)
314                     operational = operational + "?";
315 
316             }
317             boardCodified.add(operational);
318         }
319 
320         return boardCodified;
321     }
```

6.32.4 Member Data Documentation

6.32.4.1 currentEasyDifficulty

```
EasyDifficulty test.driver.EasyDifficultyDriver.currentEasyDifficulty
```

Definition at line 24 of file EasyDifficultyDriver.java.

6.32.4.2 currentBoard

```
Board test.driver.EasyDifficultyDriver.currentBoard
```

Definition at line 26 of file EasyDifficultyDriver.java.

6.32.4.3 nameCurrentBoard

```
String test.driver.EasyDifficultyDriver.nameCurrentBoard
```

Definition at line 27 of file EasyDifficultyDriver.java.

6.32.4.4 fixtureRepository

```
FixtureRepository test.driver.EasyDifficultyDriver.fixtureRepository
```

Definition at line 29 of file EasyDifficultyDriver.java.

The documentation for this class was generated from the following file:

- EasyDifficultyDriver.java

6.33 cmd.unitary.entry Class Reference

JUnit Entry tests entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)
  JUnit Entry tests main function. Calls the JUnitCore main entrypoint and runs the Entry unitary tests.

6.33.1 Detailed Description

JUnit Entry tests entrypoint. By Alex Rodriguez.

Definition at line 17 of file entry.java.

6.33.2 Member Function Documentation

6.33.2.1 main()

```
static void cmd.unitary.entry.main (
             String[] args )  [static]
```

JUnit Entry tests main function. Calls the JUnitCore main entrypoint and runs the Entry unitary tests.

Precondition

True.

Postcondition

The JUnit Entry tests have started.

Definition at line 24 of file entry.java.

```
                                     {
           JUnitCore.main(new EntryJUnit().getClass().getName());
       }
```

The documentation for this class was generated from the following file:

- entry.java

6.34 domain.Entry Class Reference

Represents an entry in a Ranking table.

Public Member Functions

- Entry (UUID playerID, int value)

  Builder operation that has parameters playerID and playerValue and creates a new Entry with them.
- Entry (JSONObject entry)

  Builder operation that creates a new Entry using the information from a parameter entry.
- JSONObject serialize ()

  Operation that translates an Entry into a JSONObject.
- UUID getPlayerID ()

  Consulting operation that returns the id of the player.
- int getValue ()

  Consulting operation that returns the value of the player.
- void setPlayerID (UUID newPlayerID)

  Modifying operation that swaps the playerID in Entry for the parameter newPlayerID.
- void setValue (int newValue)

  Modifying operation that swaps the value in Entry for the parameter newValue.

Private Attributes

- UUID playerID

  ID of the player.
- int value

  Value of the player.

6.34.1 Detailed Description

Represents an entry in a Ranking table.

Created by Roger Mollon

Class that represents an entry. It contains a player ID and a player value

Definition at line 19 of file Entry.java.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 Entry() [1/2]

```
domain.Entry.Entry (
            UUID playerID,
            int value )
```

Builder operation that has parameters playerID and playerValue and creates a new Entry with them.

Precondition

value > 0

Postcondition

An Entry with playerID and value has been created

Parameters

playerID	ID of the player about to be created
value	value of the player about to be created

Definition at line 31 of file Entry.java.

```
31                                              {
32         this.playerID = playerID;
33         this.value = value;
34     }
```

6.34.2.2 Entry() [2/2]

```
domain.Entry.Entry (
            JSONObject entry )
```

Builder operation that creates a new Entry using the information from a parameter entry.

Precondition

entry.getInt("value") > 0

Postcondition

An Entry with its attributes based on entry has been created

Parameters

entry	JSONObject which contains information to create an Entry

Definition at line 41 of file Entry.java.

```
                                {
        this.playerID = UUID.fromString(entry.getString("player_id"));
        this.value = entry.getInt("value");
    }
```

6.34.3 Member Function Documentation

6.34.3.1 serialize()

`JSONObject domain.Entry.serialize ( )`

Operation that translates an Entry into a JSONObject.

Precondition

True

Postcondition

A new JSONObject with the information from the implicit Entry has been returned

Returns

JSONObject with the attributes from implicit Entry

Definition at line 51 of file Entry.java.

```
                               {
        JSONObject entry = new JSONObject();

        entry.put("player_id", this.playerID.toString());
        entry.put("value", this.value);

        return entry;
    }
```

6.34.3.2 getPlayerID()

```
UUID domain.Entry.getPlayerID ( )
```

Consulting operation that returns the id of the player.

Precondition

True

Postcondition

The ID of the player in the Entry has been returned

Returns

UUID of the player in the Entry

Definition at line 65 of file Entry.java.

```
                                  {
        return this.playerID;
    }
```

6.34.3.3 getValue()

```
int domain.Entry.getValue ( )
```

Consulting operation that returns the value of the player.

Precondition

True

Postcondition

The value of the player in the Entry has been returned

Returns

Value of the Entry

Definition at line 74 of file Entry.java.

```
                              {
        return this.value;
    }
```

6.34.3.4 setPlayerID()

```
void domain.Entry.setPlayerID (
             UUID newPlayerID )
```

Modifying operation that swaps the playerID in Entry for the parameter newPlayerID.

Precondition

True

Postcondition

playerID has been changed to newPlayerID

Parameters

newPlayerID	New ID of the player

Definition at line 83 of file Entry.java.

```
                                        {
        this.playerID = newPlayerID;
    }
```

6.34.3.5 setValue()

```
void domain.Entry.setValue (
            int newValue )
```

Modifying operation that swaps the value in Entry for the parameter newValue.

Precondition

newValue > 0

Postcondition

value has been changed to newValue

Parameters

newValue	New value of the player

Definition at line 92 of file Entry.java.

```
                                  {
        this.value = newValue;
    }
```

6.34.4 Member Data Documentation

6.34.4.1 playerID

```
UUID domain.Entry.playerID [private]
```

ID of the player.

Definition at line 21 of file Entry.java.

6.34.4.2 value

```
int domain.Entry.value [private]
```

Value of the player.

Definition at line 23 of file Entry.java.

The documentation for this class was generated from the following file:

- Entry.java

6.35 test.unitary.EntryJUnit Class Reference

Allows JUnit testing of class Entry.

Public Member Functions

- void Entry ()
- void deserialize ()
- void serialize ()
- void getPlayerID ()
- void getValue ()
- void setPlayerID ()
- void setValue ()

6.35.1 Detailed Description

Allows JUnit testing of class Entry.

Created by Roger Mollon

Class that represents a testing of class Entry. It contains tester methods for all public Entry methods

Definition at line 22 of file EntryJUnit.java.

6.35.2 Member Function Documentation

6.35.2.1 Entry()

```
void test.unitary.EntryJUnit.Entry ( )
```

Definition at line 25 of file EntryJUnit.java.

```
                            {
        UUID playerID = UUID.randomUUID();
        Entry e = new Entry(playerID, 25);
        assertEquals("Entry failed because", playerID, e.getPlayerID());
        assertEquals("Entry failed because", 25, e.getValue());
    }
```

6.35.2.2 deserialize()

```
void test.unitary.EntryJUnit.deserialize ( )
```

Definition at line 33 of file EntryJUnit.java.

```
33                           {
34         Entry e = new Entry(UUID.randomUUID(), 22);
35         JSONObject jobj = e.serialize();
36         Entry e1 = new Entry(jobj);
37         assertEquals("deserialize failed because", e.getPlayerID(), e1.getPlayerID());
38         assertEquals("deserialize failed because", e.getValue(), e1.getValue());
39     }
```

6.35.2.3 serialize()

```
void test.unitary.EntryJUnit.serialize ( )
```

Definition at line 42 of file EntryJUnit.java.

```
42                         {
43         Entry e = new Entry(UUID.randomUUID(), 100);
44         JSONObject jobj = e.serialize();
45         assertEquals("serialize failed because", e.getPlayerID().toString(),
       jobj.getString("player_id"));
46         assertEquals("serialize failed because", 100, jobj.getInt("value"));
47     }
```

6.35.2.4 getPlayerID()

```
void test.unitary.EntryJUnit.getPlayerID ( )
```

Definition at line 50 of file EntryJUnit.java.

```
50                           {
51         UUID playerID = UUID.randomUUID();
52         Entry e = new Entry(playerID, 50);
53         assertEquals("getPlayerId failed because", playerID, e.getPlayerID());
54     }
```

6.35.2.5 getValue()

```
void test.unitary.EntryJUnit.getValue ( )
```

Definition at line 57 of file EntryJUnit.java.

```
57                        {
58         Entry e = new Entry(UUID.randomUUID(), 75);
59         assertEquals("getValue failed because", 75, e.getValue());
60     }
```

6.35.2.6 setPlayerID()

```
void test.unitary.EntryJUnit.setPlayerID ( )
```

Definition at line 63 of file EntryJUnit.java.

```
                              {
        UUID playerID = UUID.randomUUID();
        Entry e = new Entry(UUID.randomUUID(), 150);
        e.setPlayerID(playerID);
        assertEquals("setPlayerID failed because", playerID, e.getPlayerID());
    }
```

6.35.2.7 setValue()

```
void test.unitary.EntryJUnit.setValue ( )
```

Definition at line 71 of file EntryJUnit.java.

```
                           {
        Entry e = new Entry(UUID.randomUUID(), 180);
        e.setValue(150);
        assertEquals("setValue failed because", 150, e.getValue());
    }
```

The documentation for this class was generated from the following file:

- EntryJUnit.java

6.36 domain.Exceptions Class Reference

Holds all the different custom Exceptions used in the whole project. By Alex Rodriguez.

Classes

- class BadConfirmationException

  The entered confirmation password doesn't match the user's password. By Alex Rodriguez.
- class BotUsedException

  A bot cannot be modified or deleted if it is already part of a game. By Alex Rodriguez.
- class ConfigurationUsedException

  A configuration cannot be modified or deleted if it is already used in a game. By Alex Rodriguez.
- class ExistingConfigurationException

  There is already a configuration with the same name and creator ID in the system. By Alex Rodriguez.
- class ExistingPlayerException

  There is already a player with the same name in the system. By Alex Rodriguez.
- class FinishedGameException

  The game is already finished. By Alex Rodriguez.
- class IncorrectCredentialsException

  Wrong password or name. By Alex Rodriguez.
- class InexistingConfigurationException

  There isn't any configuration with the entered name. By Alex Rodriguez.
- class InexistingPlayerException

  There isn't any player with the entered name. By Alex Rodriguez.

- class InvalidBoardException

  The current board is in an illegal state. By Alex Rodriguez.
- class InvalidConfigurationException

  The entered configuration is null, empty or blank. By Alex Rodriguez.
- class InvalidDifficultyException

  The entered difficulty is null, empty, blank, less than 0 or greater than 10. By Alex Rodriguez.
- class InvalidNameException

  The entered name is null, empty or blank. By Alex Rodriguez.
- class InvalidPasswordException

  The entered password is null, empty or blank. By Alex Rodriguez.
- class InvalidPlayersException

  The entered players are the same, null, empty, blank or both bots have the same difficulty. By Alex Rodriguez.
- class InvalidPositionException

  The entered position is null, empty, blank or ends up with an invalid board. By Alex Rodriguez.
- class InvalidRulesException

  The entered configuration rules are all deactivated. By Alex Rodriguez.
- class NotCreatorException

  The user that tries to perform an action on a object is not the creator of it. By Alex Rodriguez.
- class NotPlayerException

  The player that wants to perform an action is not part of the game. By Alex Rodriguez.
- class NotPlayerPieceException

  The player that wants to perform an action tries to use an opponent piece. By Alex Rodriguez.
- class NotPlayerTurnException

  It is not the turn of the player that wants to perform an action. By Alex Rodriguez.
- class NotStartedGameException

  The game has not yet started. By Alex Rodriguez.

6.36.1 Detailed Description

Holds all the different custom Exceptions used in the whole project. By Alex Rodriguez.

Definition at line 13 of file Exceptions.java.

The documentation for this class was generated from the following file:

- Exceptions.java

6.37 domain.Exceptions.ExistingConfigurationException Class Reference

There is already a configuration with the same name and creator ID in the system. By Alex Rodriguez.

Public Member Functions

- ExistingConfigurationException ()

6.37.1 Detailed Description

There is already a configuration with the same name and creator ID in the system. By Alex Rodriguez.

Definition at line 129 of file Exceptions.java.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 ExistingConfigurationException()

```
domain.Exceptions.ExistingConfigurationException.ExistingConfigurationException ( )
```

Definition at line 130 of file Exceptions.java.

```
                                    {
            super("ERR_EXISTING_CONFIGURATION");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.38 domain.Exceptions.ExistingPlayerException Class Reference

There is already a player with the same name in the system. By Alex Rodriguez.

Public Member Functions

- ExistingPlayerException ()

6.38.1 Detailed Description

There is already a player with the same name in the system. By Alex Rodriguez.

Definition at line 19 of file Exceptions.java.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 ExistingPlayerException()

```
domain.Exceptions.ExistingPlayerException.ExistingPlayerException ( )
```

Definition at line 20 of file Exceptions.java.

```
                                    {
            super("ERR_EXISTING_PLAYER");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.39 domain.Exceptions.FinishedGameException Class Reference

The game is already finished. By Alex Rodriguez.

Public Member Functions

- FinishedGameException ()

6.39.1 Detailed Description

The game is already finished. By Alex Rodriguez.

Definition at line 239 of file Exceptions.java.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 FinishedGameException()

```
domain.Exceptions.FinishedGameException.FinishedGameException ( )
```

Definition at line 240 of file Exceptions.java.

```
                                  {
            super("ERR_FINISHED_GAME");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.40 repository.FixtureRepository Class Reference

Implements various CRUD operations to work with the Fixture repository. By Alex Rodriguez.

Public Member Functions

- FixtureRepository ()

  Create a FixtureRepository instance.
- ArrayList< String > listFiles ()

  List all the files of the local TXT fixtures directory.
- JSONObject boardFileToJSON (String path)

  Read a Board from a TXT file identified by the path and convert it to its JSON representation.

Private Member Functions

- List< String > getLines (String path)

  Read all lines of a file identified by a path.

Additional Inherited Members

6.40.1 Detailed Description

Implements various CRUD operations to work with the Fixture repository. By Alex Rodriguez.

Definition at line 22 of file FixtureRepository.java.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 FixtureRepository()

```
repository.FixtureRepository.FixtureRepository ( )
```

Create a FixtureRepository instance.

Precondition

The Fixture repository TXT files exists.

Postcondition

A FixtureRepository instance is created.

Definition at line 30 of file FixtureRepository.java.

```
                                    {
            super(RepositoryType.FIXTURE);
        }
```

6.40.3 Member Function Documentation

6.40.3.1 listFiles()

```
ArrayList<String> repository.FixtureRepository.listFiles ( )
```

List all the files of the local TXT fixtures directory.

Precondition

The Fixture repository TXT files exists.

Postcondition

An ArrayList containing the names of the local TXT fixtures directory is returned.

Returns

ArrayList of the names of the local TXT fixtures directory.

Definition at line 42 of file FixtureRepository.java.

```
                                         {
        try {
            ArrayList<String> list = new
    ArrayList<String>(Files.walk(Paths.get(this.path)).filter(Files::isRegularFile)
                    .map(file -> file.toString()).collect(Collectors.toList()));
            return list;
        } catch (Exception e) {
            e.printStackTrace();
        }

        return new ArrayList<String>();
    }
```

6.40.3.2 getLines()

```
List<String> repository.FixtureRepository.getLines (
              String path )  [private]
```

Read all lines of a file identified by a path.

Precondition

The Fixture repository TXT files exists.

Postcondition

A List containing the lines of the file identified by the path is returned.

Parameters

path	Path of the file to be read.

Returns

List of the lines of the file identified by the path.

Definition at line 61 of file FixtureRepository.java.

```
61                                         {
62         List<String> lines = new ArrayList<String>();
63 
64         try {
65             lines = Files.readAllLines(Paths.get(path), StandardCharsets.UTF_8);
66         } catch (Exception e) {
67             e.printStackTrace();
68         }
69 
70         return lines;
71     }
```

6.40.3.3 boardFileToJSON()

```
JSONObject repository.FixtureRepository.boardFileToJSON (
              String path )
```

Read a Board from a TXT file identified by the path and convert it to its JSON representation.

Precondition

The Fixture repository TXT files exists.

Postcondition

A JSONObject representing the Board contained in the file identified by the path is returned.

Parameters

path	Path of the file containing the Board to be read.

Returns

JSONObject that represents the Board contained in the file identified by the path.

Definition at line 80 of file FixtureRepository.java.

```
80                                                {
81         JSONObject board = new JSONObject();
82 
83         List<String> lines = this.getLines(path);
84 
85         for (int i = 0; i < 8; i++) {
86             String row = "";
87             if (lines.size() > i)
88                 row = lines.get(i);
89             board.put("row" + i, (row.replaceAll("[^BN\\?,.]+", "") + "????????").substring(0, 8));
90         }
91 
92         return board;
93     }
```

The documentation for this class was generated from the following file:

- FixtureRepository.java

6.41 domain.Game Class Reference

Represents the state of an Othello game including its name, players, the current turn, the state, the configuration used, the winner if any, its creator and the creation timestamp. By Alex Rodriguez.

Classes

- enum GameState

  State of a Game. Whether it has not started, it is currently being played or it has already finished.

Public Member Functions

- Game (String name, UUID player1ID, UUID player2ID, String configurationName, UUID creatorID)

  Create a Game instance.
- Game (JSONObject game)

  Create a Game instance from a JSONObject representation of a Game.
- JSONObject serialize ()

  Create a JSONObject representation of a Game from the implicit Game.
- String getName ()

  Get the name of the implicit Game.
- void setName (String name) throws InvalidNameException

  Set the name of the implicit Game.
- UUID getPlayer1ID ()

  Get the player1ID of the implicit Game.
- UUID getPlayer2ID ()

  Get the player2ID of the implicit Game.
- String getConfigurationName ()

  Get the configurationName of the implicit Game.
- void setConfigurationName (String configurationName) throws InvalidConfigurationException

  Set the configurationName of the implicit Game.
- PieceType getTurn ()

  Get the turn of the implicit Game.
- void setTurn (PieceType turn)

  Set the turn of the implicit Game.
- GameState getState ()

  Get the state of the implicit Game.
- void setState (GameState state)

  Set the state of the implicit Game.
- UUID getWinnerID ()

  Get the winnerID of the implicit Game.
- UUID getCreatorID ()

  Get the creatorID of the implicit Game.
- LocalDateTime getCreatedAt ()

  Get the createdAt of the implicit Game.
- void play () throws FinishedGameException

  Start playing in the implicit Game.
- void surrender (UUID surrendeeID) throws NotPlayerException, FinishedGameException, NotStarted↩ GameException

  Surrender the implicit Game.

- void finish (UUID winnerID) throws NotPlayerException, FinishedGameException, NotStartedGameException

  Finish the implicit Game and set a winner if the Game did not end in a draw.
- void checkPlaceRights (UUID playerID, PieceType pieceType) throws NotPlayerException, NotPlayerPiece↩ Exception, NotPlayerTurnException, FinishedGameException, NotStartedGameException

  Check whether a Player is able to place a piece in the implicit Game.
- void nextTurn () throws FinishedGameException, NotStartedGameException

  Pass the turn of the implicit Game.

Private Attributes

- String name

  Name of the Game.
- UUID player1ID

  First player ID of the Game.
- UUID player2ID

  Second player ID of the Game.
- String configurationName

  Name of the Configuration used to create the Game.
- PieceType turn

  Current turn of the Game.
- GameState state

  Current state of the Game.
- UUID winnerID

  Winner, if any, of the Game. If the state is FINISHED and it is null, it means the Game ended in a draw.
- UUID creatorID

  Player ID of the Game's creator.
- LocalDateTime createdAt

  Game creation timestamp.

6.41.1 Detailed Description

Represents the state of an Othello game including its name, players, the current turn, the state, the configuration used, the winner if any, its creator and the creation timestamp. By Alex Rodriguez.

Definition at line 28 of file Game.java.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Game() [1/2]

```
domain.Game.Game (
            String name,
            UUID player1ID,
            UUID player2ID,
            String configurationName,
            UUID creatorID )
```

Create a Game instance.

Precondition

True

Postcondition

A Game instance is created and its implicits name, player1ID, player2ID, configurationName and creatorID attributes are setted. The current turn is setted to PLAYER1, the state to NOT_STARTED, the winnerID to null and the createdAt to the current timestamp.

Parameters

name	Name of the Game.
player1ID	First player ID of the Game.
player2ID	Second player ID of the Game.
configurationName	Name of the Configuration used to create the Game.
creatorID	Player ID of the Game's creator.

Definition at line 88 of file Game.java.

```
                                                                                          {
        this.name = name;
        this.player1ID = player1ID;
        this.player2ID = player2ID;
        this.configurationName = configurationName;
        this.turn = PieceType.PLAYER1;
        this.state = GameState.NOT_STARTED;
        this.winnerID = null;
        this.creatorID = creatorID;
        this.createdAt = LocalDateTime.now();
    }
```

6.41.2.2 Game() [2/2]

```
domain.Game.Game (
            JSONObject game )
```

Create a Game instance from a JSONObject representation of a Game.

Precondition

True

Postcondition

A Game instance is created and its implicits name, player1ID, player2ID, configurationName and creatorID attributes are setted. The current turn is setted to PLAYER1, the state to NOT_STARTED, the winnerID to null and the createdAt to the current timestamp.

Parameters

game	JSONObject representation of a Game.

Definition at line 107 of file Game.java.

```
                                  {
        this.name = game.getString("name");
        this.player1ID = UUID.fromString(game.getString("player1_id"));
        this.player2ID = UUID.fromString(game.getString("player2_id"));
        this.configurationName = game.getString("configuration_name");
        this.turn = game.getEnum(PieceType.class, "turn");
        this.state = game.getEnum(GameState.class, "state");

        this.winnerID = null;
        String winnerID = game.optString("winner_id", null);
        if (winnerID != null)
            this.winnerID = UUID.fromString(winnerID);

        this.creatorID = UUID.fromString(game.getString("creator_id"));
        this.createdAt = LocalDateTime.parse(game.getString("created_at"));
    }
```

6.41.3 Member Function Documentation

6.41.3.1 serialize()

`JSONObject domain.Game.serialize ( )`

Create a JSONObject representation of a Game from the implicit Game.

Precondition

True

Postcondition

A JSONObject representing the implicit Game is returned.

Returns

JSONObject representation of a Game.

Definition at line 132 of file Game.java.

```
                                   {
        JSONObject game = new JSONObject();

        game.put("name", this.name);
        game.put("player1_id", this.player1ID.toString());
        game.put("player2_id", this.player2ID.toString());
        game.put("configuration_name", this.configurationName);
        game.put("turn", this.turn);
        game.put("state", this.state);

        if (this.winnerID != null)
            game.put("winner_id", this.winnerID.toString());
        else
            game.put("winner_id", JSONObject.NULL);

        game.put("creator_id", this.creatorID.toString());
        game.put("created_at", this.createdAt.toString());

        return game;
    }
```

6.41.3.2 getName()

```
String domain.Game.getName ( )
```

Get the name of the implicit Game.

Precondition

True

Postcondition

The name attribute of the implicit Game is returned.

Returns

Name of the implicit Game.

Definition at line 159 of file Game.java.

```
                                {
        return this.name;
    }
```

6.41.3.3 setName()

```
void domain.Game.setName (
             String name ) throws InvalidNameException
```

Set the name of the implicit Game.

Precondition

True

Postcondition

The name attribute of the implicit Game is setted if it is not blank, otherwise an InvalidNameException is thrown.

Parameters

name	Name of the Game.

Definition at line 170 of file Game.java.

```
                                                            {
        if (name.isBlank())
            throw new InvalidNameException();

        this.name = name;
    }
```

6.41.3.4 getPlayer1ID()

```
UUID domain.Game.getPlayer1ID ( )
```

Get the player1ID of the implicit Game.

Precondition

True

Postcondition

The player1ID attribute of the implicit Game is returned.

Returns

Player1ID of the implicit Game.

Definition at line 183 of file Game.java.

```
                               {
        return this.player1ID;
    }
```

6.41.3.5 getPlayer2ID()

```
UUID domain.Game.getPlayer2ID ( )
```

Get the player2ID of the implicit Game.

Precondition

True

Postcondition

The player2ID attribute of the implicit Game is returned.

Returns

Player2ID of the implicit Game.

Definition at line 193 of file Game.java.

```
                               {
        return this.player2ID;
    }
```

6.41.3.6 getConfigurationName()

```
String domain.Game.getConfigurationName ( )
```

Get the configurationName of the implicit Game.

Precondition

True

Postcondition

The configurationName attribute of the implicit Game is returned.

Returns

ConfigurationName of the implicit Game.

Definition at line 203 of file Game.java.

```
                                          {
        return this.configurationName;
    }
```

6.41.3.7 setConfigurationName()

```
void domain.Game.setConfigurationName (
              String configurationName ) throws InvalidConfigurationException
```

Set the configurationName of the implicit Game.

Precondition

True

Postcondition

The configurationName attribute of the implicit Game is setted if it is not blank, otherwise an InvalidName←Exception is thrown.

Parameters

configurationName	Name of the Configuration used to create the Game.

Definition at line 214 of file Game.java.

```
                                                                                                      {
        if (configurationName.isBlank())
            throw new InvalidConfigurationException();

        this.configurationName = configurationName;
    }
```

6.41.3.8 getTurn()

```
PieceType domain.Game.getTurn ( )
```

Get the turn of the implicit Game.

Precondition

True

Postcondition

The turn attribute of the implicit Game is returned.

Returns

Turn of the implicit Game.

Definition at line 227 of file Game.java.

```
                                {
        return this.turn;
    }
```

6.41.3.9 setTurn()

```
void domain.Game.setTurn (
            PieceType turn )
```

Set the turn of the implicit Game.

Precondition

True

Postcondition

The turn attribute of the implicit Game is setted.

Parameters

turn	Current turn of the Game.

Definition at line 237 of file Game.java.

```
                                        {
        this.turn = turn;
    }
```

6.41.3.10 getState()

```
GameState domain.Game.getState ( )
```

Get the state of the implicit Game.

Precondition

True

Postcondition

The state attribute of the implicit Game is returned.

Returns

State of the implicit Game.

Definition at line 247 of file Game.java.

```
                                {
        return this.state;
    }
```

6.41.3.11 setState()

```
void domain.Game.setState (
             GameState state )
```

Set the state of the implicit Game.

Precondition

True

Postcondition

The state attribute of the implicit Game is setted.

Parameters

state	Current state of the Game.

Definition at line 257 of file Game.java.

```
                                         {
        this.state = state;
    }
```

6.41.3.12 getWinnerID()

```
UUID domain.Game.getWinnerID ( )
```

Get the winnerID of the implicit Game.

Precondition

True

Postcondition

The winnerID attribute of the implicit Game is returned.

Returns

WinnerID of the implicit Game.

Definition at line 267 of file Game.java.

```
                             {
        return this.winnerID;
    }
```

6.41.3.13 getCreatorID()

```
UUID domain.Game.getCreatorID ( )
```

Get the creatorID of the implicit Game.

Precondition

True

Postcondition

The creatorID attribute of the implicit Game is returned.

Returns

CreatorID of the implicit Game.

Definition at line 277 of file Game.java.

```
                              {
        return this.creatorID;
    }
```

6.41.3.14 getCreatedAt()

```
LocalDateTime domain.Game.getCreatedAt ( )
```

Get the createdAt of the implicit Game.

Precondition

True

Postcondition

The createdAt attribute of the implicit Game is returned.

Returns

CreatedAt of the implicit Game.

Definition at line 287 of file Game.java.

```
                                        {
        return this.createdAt;
    }
```

6.41.3.15 play()

```
void domain.Game.play ( ) throws FinishedGameException
```

Start playing in the implicit Game.

Precondition

The state attribute of the implicit Game is NOT_STARTED.

Postcondition

The state attribute of the implicit Game is setted to IN_PROGRESS if any of the following exceptions are not thrown:

- FinishedGameException if the implicit Game has already finished.

Definition at line 297 of file Game.java.

```
                                                    {
        if (this.state == GameState.FINISHED)
            throw new FinishedGameException();

        this.state = GameState.IN_PROGRESS;
    }
```

6.41.3.16 surrender()

```
void domain.Game.surrender (
            UUID surrendeeID ) throws NotPlayerException, FinishedGameException, NotStartedGameException
```

Surrender the implicit Game.

Precondition

True

Postcondition

The state attribute is setted to FINISHED and the winnerID of the implicit Game is setted to the oponent Player if any of the following exceptions are not thrown:

- NotPlayerException if the player that wants to surrender is not part of the implicit Game.
- FinishedGameException if the implicit Game has already finished.
- NotStartedGameException if the implicit Game has not yet started.

Parameters

surrendeeID	ID of the Player that surrends.

Definition at line 314 of file Game.java.

```
                {
        if (surrendeeID.equals(this.player1ID))
            this.finish(this.player2ID);
        else if (surrendeeID.equals(this.player2ID))
            this.finish(this.player1ID);
        else
            throw new NotPlayerException();
    }
```

6.41.3.17 finish()

```
void domain.Game.finish (
            UUID winnerID ) throws NotPlayerException, FinishedGameException, NotStartedGameException
```

Finish the implicit Game and set a winner if the Game did not end in a draw.

Precondition

True

Postcondition

The state attribute is setted to FINISHED and the winnerID of the implicit Game is setted to the winner Player or null if the Game ended in a draw, if any of the following exceptions are not thrown:

- NotPlayerException if the player that wants to finish is not part of the implicit Game.
- FinishedGameException if the implicit Game has already finished.
- NotStartedGameException if the implicit Game has not yet started.

Parameters

winnerID	ID of the Player that wins or null if the implicit Game ended in a draw.

Definition at line 333 of file Game.java.

```
                {
        if (this.state == GameState.NOT_STARTED)
            throw new NotStartedGameException();

        if (this.state == GameState.FINISHED)
            throw new FinishedGameException();

        if (winnerID != null && !winnerID.equals(this.player1ID) && !winnerID.equals(this.player2ID))
            throw new NotPlayerException();

        this.state = GameState.FINISHED;
        this.winnerID = winnerID;
    }
```

6.41.3.18 checkPlaceRights()

```
void domain.Game.checkPlaceRights (
              UUID playerID,
              PieceType pieceType ) throws NotPlayerException, NotPlayerPieceException, NotPlayerTurnException,
FinishedGameException, NotStartedGameException
```

Check whether a Player is able to place a piece in the implicit Game.

Precondition

True

Postcondition

It executes successfully if any of the following exceptions are not thrown:

- NotPlayerException if the player that wants to place a piece is not part of the implicit Game.
- NotPlayerPieceException if the player wants to place an opponent piece.
- NotPlayerTurnException if it is not the turn of the player that wants to place a piece.
- FinishedGameException if the implicit Game has already finished.
- NotStartedGameException if the implicit Game has not yet started.

Parameters

playerID	ID of the Player that wants to place a piece in the implicit Game.
pieceType	Type of the piece that the Player wants to place in the implicit Game.

Definition at line 359 of file Game.java.

```
                                                                          {
        if (this.state == GameState.NOT_STARTED)
            throw new NotStartedGameException();

        if (this.state == GameState.FINISHED)
            throw new FinishedGameException();

        if (playerID.equals(this.player1ID)) {
            if (pieceType != PieceType.PLAYER1)
                throw new NotPlayerPieceException();
        } else if (playerID.equals(this.player2ID)) {
            if (pieceType != PieceType.PLAYER2)
                throw new NotPlayerPieceException();
        } else {
            throw new NotPlayerException();
        }

        if (pieceType != this.turn)
            throw new NotPlayerTurnException();
    }
```

6.41.3.19 nextTurn()

`void domain.Game.nextTurn ( ) throws FinishedGameException, NotStartedGameException`

Pass the turn of the implicit Game.

Precondition

True

Postcondition

The turn attribute of the implicit Game is setted to the opponent Player if any of the following exceptions are not thrown:

- FinishedGameException if the implicit Game has already finished.
- NotStartedGameException if the implicit Game has not yet started.

Definition at line 388 of file Game.java.

```
                                                                                     {
        if (this.state == GameState.NOT_STARTED)
            throw new NotStartedGameException();

        if (this.state == GameState.FINISHED)
            throw new FinishedGameException();

        this.turn = (this.turn == PieceType.PLAYER1 ? PieceType.PLAYER2 : PieceType.PLAYER1);
    }
```

6.41.4 Member Data Documentation

6.41.4.1 name

`String domain.Game.name [private]`

Name of the Game.

Definition at line 41 of file Game.java.

6.41.4.2 player1ID

```
UUID domain.Game.player1ID  [private]
```

First player ID of the Game.

Definition at line 45 of file Game.java.

6.41.4.3 player2ID

```
UUID domain.Game.player2ID  [private]
```

Second player ID of the Game.

Definition at line 49 of file Game.java.

6.41.4.4 configurationName

```
String domain.Game.configurationName  [private]
```

Name of the Configuration used to create the Game.

Definition at line 53 of file Game.java.

6.41.4.5 turn

```
PieceType domain.Game.turn  [private]
```

Current turn of the Game.

Definition at line 57 of file Game.java.

6.41.4.6 state

```
GameState domain.Game.state  [private]
```

Current state of the Game.

Definition at line 61 of file Game.java.

6.41.4.7 winnerID

```
UUID domain.Game.winnerID  [private]
```

Winner, if any, of the Game. If the state is FINISHED and it is null, it means the Game ended in a draw.

Definition at line 65 of file Game.java.

6.41.4.8 creatorID

```
UUID domain.Game.creatorID  [private]
```

Player ID of the Game's creator.

Definition at line 69 of file Game.java.

6.41.4.9 createdAt

```
LocalDateTime domain.Game.createdAt  [private]
```

Game creation timestamp.

Definition at line 73 of file Game.java.

The documentation for this class was generated from the following file:

- Game.java

6.42 cmd.driver.game Class Reference

Game driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)
  Game driver main function. Creates an instance of the Game driver and starts it.

6.42.1 Detailed Description

Game driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file game.java.

6.42.2 Member Function Documentation

6.42.2.1 main()

```
static void cmd.driver.game.main (
              String[] args )  [static]
```

Game driver main function. Creates an instance of the Game driver and starts it.

Precondition

True.

Postcondition

The Game driver has started.

Definition at line 22 of file game.java.

```
                                          {
          new GameDriver().start();
      }
```

The documentation for this class was generated from the following file:

- game.java

6.43 view.GameBoardView Class Reference

Public Member Functions

- GameBoardView ()
  Class creator.
- void initialize ()
  Initialize method which is executed when the scene is shown.
- void goToMenu () throws IOException
  Event method which is executed when the goToMenu button is clicked.
- void save () throws IOException
  Event method which is executed when the save button is clicked.
- void surrender () throws IOException
  Event method which is executed when the surrender button is clicked.
- void transform (MouseEvent mouseEvent)
  Event method which is executed when a piece is clicked.
- void onChangeAssistedMode ()
  Method executed everytime there is a change in the Assisted mode radio button.

Private Member Functions

- void renderState ()

    Render the current game state.
- void renderResult (UUID winnerID)

    Render the result of a game.
- void render ()

    Method executed everytime there is a change in the board.
- void drawPiece (Pair< Integer, Integer > pos, char pieceType, boolean stroke)

    Painting method executed everytime there is a change in the board.
- Pair< Integer, Integer > getClickedPos (MouseEvent mouseEvent)

    Painting method executed everytime a player clicks on the board.
- Circle getCircle (Pair< Integer, Integer > pos)

    Method executed everytime there is a change in the board.

Private Attributes

- Text goToMenu

    goToMenu button.
- Circle f1c1

    Piece located in (1, 1).
- Circle f1c2

    Piece located in (1, 2).
- Circle f1c3

    Piece located in (1, 3).
- Circle f1c4

    Piece located in (1, 4).
- Circle f1c5

    Piece located in (1, 5).
- Circle f1c6

    Piece located in (1, 6).
- Circle f1c7

    Piece located in (1, 7).
- Circle f1c8

    Piece located in (1, 8).
- Circle f2c1

    Piece located in (2, 1).
- Circle f2c2

    Piece located in (2, 2).
- Circle f2c3

    Piece located in (2, 3).
- Circle f2c4

    Piece located in (2, 4).
- Circle f2c5

    Piece located in (2, 5).
- Circle f2c6

    Piece located in (2, 6).
- Circle f2c7

    Piece located in (2, 7).
- Circle f2c8

Piece located in (2, 8).
- Circle f3c1

  Piece located in (3, 1).
- Circle f3c2

  Piece located in (3, 2).
- Circle f3c3

  Piece located in (3, 3).
- Circle f3c4

  Piece located in (3, 4).
- Circle f3c5

  Piece located in (3, 5).
- Circle f3c6

  Piece located in (3, 6).
- Circle f3c7

  Piece located in (3, 7).
- Circle f3c8

  Piece located in (3, 8).
- Circle f4c1

  Piece located in (4, 1).
- Circle f4c2

  Piece located in (4, 2).
- Circle f4c3

  Piece located in (4, 3).
- Circle f4c4

  Piece located in (4, 4).
- Circle f4c5

  Piece located in (4, 5).
- Circle f4c6

  Piece located in (4, 6).
- Circle f4c7

  Piece located in (4, 7).
- Circle f4c8

  Piece located in (4, 8).
- Circle f5c1

  Piece located in (5, 1).
- Circle f5c2

  Piece located in (5, 2).
- Circle f5c3

  Piece located in (5, 3).
- Circle f5c4

  Piece located in (5, 4).
- Circle f5c5

  Piece located in (5, 5).
- Circle f5c6

  Piece located in (5, 6).
- Circle f5c7

  Piece located in (5, 7).
- Circle f5c8

  Piece located in (5, 8).
- Circle f6c1

  Piece located in (6, 1).

- Circle f6c2

  Piece located in (6, 2).
- Circle f6c3

  Piece located in (6, 3).
- Circle f6c4

  Piece located in (6, 4).
- Circle f6c5

  Piece located in (6, 5).
- Circle f6c6

  Piece located in (6, 6).
- Circle f6c7

  Piece located in (6, 7).
- Circle f6c8

  Piece located in (6, 8).
- Circle f7c1

  Piece located in (7, 1).
- Circle f7c2

  Piece located in (7, 2).
- Circle f7c3

  Piece located in (7, 3).
- Circle f7c4

  Piece located in (7, 4).
- Circle f7c5

  Piece located in (7, 5).
- Circle f7c6

  Piece located in (7, 6).
- Circle f7c7

  Piece located in (7, 7).
- Circle f7c8

  Piece located in (7, 8).
- Circle f8c1

  Piece located in (8, 1).
- Circle f8c2

  Piece located in (8, 2).
- Circle f8c3

  Piece located in (8, 3).
- Circle f8c4

  Piece located in (8, 4).
- Circle f8c5

  Piece located in (8, 5).
- Circle f8c6

  Piece located in (8, 6).
- Circle f8c7

  Piece located in (8, 7).
- Circle f8c8

  Piece located in (8, 8).
- Text save

  Save board button text.
- Rectangle saveButton

  Save board button.
- Text surrender

Surrender board button text.

- Rectangle surrenderButton

  Surrender board button text.

- ImageView tieIcon

  Tie icon image.

- ImageView winIcon
- Label gameResult
- Label player2
- Label player2Turn
- Label player2Pieces
- Label player2Type
- Label player1
- Label player1Turn
- Label player1Pieces
- Label player1Type
- RadioButton assistedMode
- JSONObject board

  Current board.

- JSONObject game

  Current game.

- Pair< JSONObject, JSONObject > players

  Current players.

- JSONObject user

  Current user.

- UUID turnPlayerID

  Current ID of the turn's player.

- Boolean turnPlayerIsBot

  Whether the current turn's player is a bot.

- String turnPieceType

  Current turn's piece type.

- Boolean isSpectating

  Whether the current user is spectating a game.

- Boolean isVsBot

  Whether the current user is vs bot.

- Timer timer

  Timer to automatically perform bot placing trough runtimes threads asynchronously.

6.43.1 Detailed Description

This class represents the scene controller of the game board view .

By Alex Rodriguez

Definition at line 38 of file GameBoardView.java.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 GameBoardView()

```
view.GameBoardView.GameBoardView ( )
```

Class creator.

Definition at line 45 of file GameBoardView.java.

```
45                            {
46     }
```

6.43.3 Member Function Documentation

6.43.3.1 initialize()

```
void view.GameBoardView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The board is setted.

Definition at line 503 of file GameBoardView.java.

```
503                              {
504         board = ViewCtrl.domainCtrl.viewBoard();
505         game = ViewCtrl.domainCtrl.viewGame();
506         players = ViewCtrl.domainCtrl.viewPlayers();
507         user = ViewCtrl.domainCtrl.viewUser();
508         isSpectating = (!user.getString("id").equals(players.first.getString("id"))
509                 && !user.getString("id").equals(players.second.getString("id")));
510         if (isSpectating) {
511             surrender.setVisible(false);
512             surrenderButton.setVisible(false);
513             save.setVisible(false);
514             saveButton.setVisible(false);
515             assistedMode.setVisible(false);
516         }
517         isVsBot = (!isSpectating
518                 && (players.first.getString("type").equals("BOT") ||
       players.second.getString("type").equals("BOT")));
519         player1.setText(players.first.getString("name"));
520         player1Type.setText(players.first.getString("type"));
521         player2.setText(players.second.getString("name"));
522         player2Type.setText(players.second.getString("type"));
523         turnPieceType = game.get("turn").toString();
524         tieIcon.setVisible(false);
525         winIcon.setVisible(false);
526         gameResult.setText("");
527         renderState();
528         if (game.get("state").toString().equals("FINISHED")) {
529             isSpectating = true;
530             String winner = game.optString("winner_id", null);
531             UUID winnerID = (winner != null ? UUID.fromString(winner) : null);
532             renderResult(winnerID);
533         } else {
534             if (turnPlayerIsBot) {
535                 timer = new Timer();
536                 timer.schedule(new BotTask(), 500);
537             }
538         }
539     }
```

6.43.3.2 goToMenu()

`void view.GameBoardView.goToMenu ( ) throws IOException`

Event method which is executed when the goToMenu button is clicked.

Precondition

True

Postcondition

The scene is changed to PlayView.

Definition at line 546 of file GameBoardView.java.

```
                                  {
        if (!game.get("state").toString().equals("FINISHED")) {
            Alert confirm = new Alert(AlertType.CONFIRMATION, "You will exit without saving. Are you
    sure?",
                    ButtonType.YES, ButtonType.NO);
            confirm.showAndWait();
            if (confirm.getResult() == ButtonType.YES) {
                ViewCtrl.domainCtrl.exitGame();
                ViewCtrl.changeScene("template/PlayView.fxml");
            }
        } else {
            ViewCtrl.domainCtrl.exitGame();
            ViewCtrl.changeScene("template/PlayView.fxml");
        }
    }
```

6.43.3.3 save()

`void view.GameBoardView.save ( ) throws IOException`

Event method which is executed when the save button is clicked.

Precondition

True

Postcondition

The game is saved and user can close the game.

Definition at line 566 of file GameBoardView.java.

```
                               {
        Pair<JSONObject, String> result = ViewCtrl.domainCtrl.saveGame();
        if (result.second != null) {
            switch (result.second) {
                case "ERR_NOT_PLAYER":
                    gameResult.setText("You are not part of this game!");
                    break;
                default:
                    gameResult.setText("Something went wrong, try again!");
                    break;
            }
        } else {
            gameResult.setText("");
            Alert confirm = new Alert(AlertType.CONFIRMATION, "Do you also want to exit the current
    game?",
                    ButtonType.YES, ButtonType.NO);
            confirm.showAndWait();
            if (confirm.getResult() == ButtonType.YES) {
                ViewCtrl.changeScene("template/PlayView.fxml");
            }
        }
    }
```

6.43.3.4 surrender()

```
void view.GameBoardView.surrender ( ) throws IOException
```

Event method which is executed when the surrender button is clicked.

Precondition

True

Postcondition

The game is finished and user automatically loses the game.

Definition at line 593 of file GameBoardView.java.

```
                                                {
        if (!isSpectating) {
            if (!isVsBot || turnPlayerID.equals(UUID.fromString(user.getString("id")))) {
                Alert confirm = new Alert(AlertType.CONFIRMATION,
                        "You will surrender and the game will be saved. Are you sure?", ButtonType.YES, ButtonType.NO);
                confirm.showAndWait();
                if (confirm.getResult() == ButtonType.YES) {
                    Pair<JSONObject, String> result = ViewCtrl.domainCtrl.surrender(turnPlayerID);
                    if (result.second != null) {
                        switch (result.second) {
                            case "ERR_NOT_PLAYER":
                                gameResult.setText("You are not part of this game!");
                                break;
                            case "ERR_FINISHED_GAME":
                                gameResult.setText("This game is already finished!");
                                break;
                            case "ERR_NOT_STARTED_GAME":
                                gameResult.setText("This game has not yet started!");
                                break;
                            default:
                                gameResult.setText("Something went wrong, try again!");
                                break;
                        }
                    } else {
                        gameResult.setText("");
                        game = result.first;
                        renderState();
                        renderResult(UUID.fromString(game.getString("winner_id")));
                        ViewCtrl.domainCtrl.saveGame();
                        confirm = new Alert(AlertType.CONFIRMATION, "Do you also want to exit the current game?",
                                ButtonType.YES, ButtonType.NO);
                        confirm.showAndWait();
                        if (confirm.getResult() == ButtonType.YES) {
                            ViewCtrl.changeScene("template/PlayView.fxml");
                        }
                    }
                }
            }
        }
    }
```

6.43.3.5 transform()

```
void view.GameBoardView.transform (
               MouseEvent mouseEvent )
```

Event method which is executed when a piece is clicked.

Precondition

True

Postcondition

The piece changes into white or black.

Definition at line 639 of file GameBoardView.java.

```
                                                 {
        if (mouseEvent == null || !isSpectating) {
            if (mouseEvent == null || !isVsBot ||
       turnPlayerID.equals(UUID.fromString(user.getString("id")))) {
                Pair<Integer, Integer> pos = (!turnPlayerIsBot ? getClickedPos(mouseEvent) : null);
                Pair<Pair<JSONObject, String>, String> result = ViewCtrl.domainCtrl.placePiece(pos,
       turnPlayerID,
                        turnPieceType);
                if (result.second != null) {
                    switch (result.second) {
                        case "ERR_NOT_PLAYER":
                            gameResult.setText("You are not part of this game!");
                            break;
                        case "ERR_NOT_PLAYER_PIECE":
                            gameResult.setText("Is not your turn!");
                            break;
                        case "ERR_NOT_PLAYER_TURN":
                            gameResult.setText("Is not your turn!");
                            break;
                        case "ERR_NOT_STARTED_GAME":
                            gameResult.setText("This game has not yet started!");
                            break;
                        case "ERR_INVALID_POSITION":
                            gameResult.setText("Can't place there!");
                            break;
                        case "ERR_FINISHED_GAME":
                            gameResult.setText("");
                            board = result.first.first;
                            turnPieceType = result.first.second;
                            game = ViewCtrl.domainCtrl.viewGame();
                            renderState();
                            String winner = game.optString("winner_id", null);
                            UUID winnerID = (winner != null ? UUID.fromString(winner) : null);
                            ViewCtrl.domainCtrl.saveGame();
                            renderResult(winnerID);
                            break;
                        default:
                            gameResult.setText("Something went wrong, try again!");
                            break;
                    }
                } else {
                    try {
                        board = result.first.first;
                        turnPieceType = result.first.second;
                        gameResult.setText("");
                        renderState();
                        if (turnPlayerIsBot) {
                            timer = new Timer();
                            timer.schedule(new BotTask(), 500);
                        }
                    } catch (Exception e) {
                    }
                }
            }
        }
    }
```

6.43.3.6 renderState()

```
void view.GameBoardView.renderState ( )  [private]
```

Render the current game state.

Precondition

True

Postcondition

The current game state is rendered onto the view.

Definition at line 699 of file GameBoardView.java.

```
699                             {
700         if (turnPieceType == "PLAYER1") {
701             turnPlayerID = UUID.fromString(players.first.getString("id"));
702             turnPlayerIsBot = (players.first.getString("type").equals("BOT"));
703             player1Turn.setVisible(true);
704             player2Turn.setVisible(false);
705         } else if (turnPieceType == "PLAYER2") {
706             turnPlayerID = UUID.fromString(players.second.getString("id"));
707             turnPlayerIsBot = (players.second.getString("type").equals("BOT"));
708             player1Turn.setVisible(false);
709             player2Turn.setVisible(true);
710         } else {
711             player1Turn.setVisible(false);
712             player2Turn.setVisible(false);
713         }
714 
715         Pair<Integer, Integer> numPieces = ViewCtrl.domainCtrl.getNumPieces();
716         player1Pieces.setText(String.format("x%d", numPieces.first));
717         player2Pieces.setText(String.format("x%d", numPieces.second));
718 
719         render();
720         if (!(turnPlayerIsBot || game.get("state").toString().equals("FINISHED"))) {
721             ArrayList<Pair<Integer, Integer» validPositions =
       ViewCtrl.domainCtrl.validPositions(turnPieceType);
722             for (Pair<Integer, Integer> pos : validPositions)
723                 drawPiece(pos, (turnPieceType == "PLAYER1" ? 'B' : 'N'), true);
724             if (assistedMode.isSelected()) {
725                 Pair<Integer, Integer> bestPos = ViewCtrl.domainCtrl.getBestPosition(10, turnPieceType);
726                 if (bestPos != null)
727                     drawPiece(bestPos, 'X', true);
728             }
729         }
730     }
```

6.43.3.7 renderResult()

```
void view.GameBoardView.renderResult (
              UUID winnerID )  [private]
```

Render the result of a game.

Precondition

True

Postcondition

The current game's result is rendered onto the view.

Definition at line 737 of file GameBoardView.java.

```
                                                    {
        surrender.setVisible(false);
        surrenderButton.setVisible(false);
        save.setVisible(false);
        saveButton.setVisible(false);
        assistedMode.setVisible(false);
        player1Turn.setVisible(false);
        player2Turn.setVisible(false);
        isSpectating = true;

        if (winnerID == null) {
            gameResult.setText("The game has ended in a draw.");
            tieIcon.setVisible(true);
        } else if (winnerID.equals(UUID.fromString(players.first.getString("id")))) {
            gameResult.setTextFill(Color.web("0xFFFFFF", 1.0));
            gameResult.setText(String.format("%s has won the game.", players.first.getString("name")));
            winIcon.setVisible(true);
        } else {
            gameResult.setTextFill(Color.web("0x000000", 1.0));
            gameResult.setText(String.format("%s has won the game.", players.second.getString("name")));
            winIcon.setVisible(true);
        }
    }
```

6.43.3.8 render()

`void view.GameBoardView.render ( ) [private]`

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

The change is setted in the board.

Definition at line 766 of file GameBoardView.java.

```
                              {
        for (int i = 0; i < 8; i++) {
            char[] row = board.getString(String.format("row%d", i)).toCharArray();
            for (int j = 0; j < 8; j++)
                drawPiece(new Pair<Integer, Integer>(i, j), row[j], false);
        }
    }
```

6.43.3.9 drawPiece()

```
void view.GameBoardView.drawPiece (
            Pair< Integer, Integer > pos,
            char pieceType,
            boolean stroke )  [private]
```

Painting method executed everytime there is a change in the board.

Precondition

True

Postcondition

Pieces change to the correct color.

Definition at line 779 of file GameBoardView.java.

```
                                                                                                  {
            Circle circle = getCircle(pos);
            switch (pieceType) {
                case 'B':
                    if (!stroke) {
                        circle.setFill(Color.web("0xFFFFFF", 1.0));
                        circle.setStrokeWidth(0);
                    } else {
                        circle.setStrokeWidth(3);
                        circle.setStrokeType(StrokeType.INSIDE);
                        circle.setStroke(Color.web("0xFFFFFF", 1.0));
                    }
                    break;
                case 'N':
                    if (!stroke) {
                        circle.setFill(Color.web("0x000000", 1.0));
                        circle.setStrokeWidth(0);
                    } else {
                        circle.setStrokeWidth(3);
                        circle.setStrokeType(StrokeType.INSIDE);
                        circle.setStroke(Color.web("0x000000", 1.0));
                    }
                    break;
                case 'X':
                    circle.setStrokeWidth(3);
                    circle.setStrokeType(StrokeType.INSIDE);
                    circle.setStroke(Color.web("0x0059ff", 1.0));
                    break;
                case '?':
                    circle.setFill(Color.web("0x34d399", 1.0));
                    circle.setStrokeWidth(0);
                    break;
                default:
                    break;
            }
        }
```

6.43.3.10 getClickedPos()

```
Pair<Integer, Integer> view.GameBoardView.getClickedPos (
            MouseEvent mouseEvent )  [private]
```

Painting method executed everytime a player clicks on the board.

Precondition

True

Postcondition

The piece clicked is transformed into a pair.

Definition at line 821 of file GameBoardView.java.

```
                                                                                    {
        Pair<Integer, Integer> pos = new Pair<Integer, Integer>(-1, -1);
        String piece = ((Circle) mouseEvent.getPickResult().getIntersectedNode()).getId();
        pos.first = Character.getNumericValue(piece.charAt(1)) - 1;
        pos.second = Character.getNumericValue(piece.charAt(3)) - 1;
        return pos;
    }
```

6.43.3.11 getCircle()

```
Circle view.GameBoardView.getCircle (
            Pair< Integer, Integer > pos ) [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

Return the circle which belongs to the position.

Definition at line 834 of file GameBoardView.java.

```
                                                                  {
        try {
            Field field = this.getClass().getDeclaredField(String.format("f%sc%s", pos.first + 1,
    pos.second + 1));
            field.setAccessible(true);
            return (Circle) field.get(this);
        } catch (Exception e) {
            return new Circle();
        }
    }
```

6.43.3.12 onChangeAssistedMode()

```
void view.GameBoardView.onChangeAssistedMode ( )
```

Method executed everytime there is a change in the Assisted mode radio button.

Precondition

True

Postcondition

Whether the assisted mode visual help is rendered onto the current board.

Definition at line 859 of file GameBoardView.java.

```
                                          {
        renderState();
    }
```

6.43.4 Member Data Documentation

6.43.4.1 goToMenu

`Text view.GameBoardView.goToMenu [private]`

goToMenu button.

Definition at line 54 of file GameBoardView.java.

6.43.4.2 f1c1

`Circle view.GameBoardView.f1c1 [private]`

Piece located in (1, 1).

Definition at line 59 of file GameBoardView.java.

6.43.4.3 f1c2

`Circle view.GameBoardView.f1c2 [private]`

Piece located in (1, 2).

Definition at line 64 of file GameBoardView.java.

6.43.4.4 f1c3

`Circle view.GameBoardView.f1c3 [private]`

Piece located in (1, 3).

Definition at line 69 of file GameBoardView.java.

6.43.4.5 f1c4

`Circle view.GameBoardView.f1c4 [private]`

Piece located in (1, 4).

Definition at line 74 of file GameBoardView.java.

6.43.4.6 f1c5

```
Circle view.GameBoardView.f1c5  [private]
```

Piece located in (1, 5).

Definition at line 79 of file GameBoardView.java.

6.43.4.7 f1c6

```
Circle view.GameBoardView.f1c6  [private]
```

Piece located in (1, 6).

Definition at line 84 of file GameBoardView.java.

6.43.4.8 f1c7

```
Circle view.GameBoardView.f1c7  [private]
```

Piece located in (1, 7).

Definition at line 89 of file GameBoardView.java.

6.43.4.9 f1c8

```
Circle view.GameBoardView.f1c8  [private]
```

Piece located in (1, 8).

Definition at line 94 of file GameBoardView.java.

6.43.4.10 f2c1

```
Circle view.GameBoardView.f2c1  [private]
```

Piece located in (2, 1).

Definition at line 99 of file GameBoardView.java.

6.43.4.11 f2c2

```
Circle view.GameBoardView.f2c2 [private]
```

Piece located in (2, 2).

Definition at line 104 of file GameBoardView.java.

6.43.4.12 f2c3

```
Circle view.GameBoardView.f2c3 [private]
```

Piece located in (2, 3).

Definition at line 109 of file GameBoardView.java.

6.43.4.13 f2c4

```
Circle view.GameBoardView.f2c4 [private]
```

Piece located in (2, 4).

Definition at line 114 of file GameBoardView.java.

6.43.4.14 f2c5

```
Circle view.GameBoardView.f2c5 [private]
```

Piece located in (2, 5).

Definition at line 119 of file GameBoardView.java.

6.43.4.15 f2c6

```
Circle view.GameBoardView.f2c6 [private]
```

Piece located in (2, 6).

Definition at line 124 of file GameBoardView.java.

6.43.4.16 f2c7

```
Circle view.GameBoardView.f2c7 [private]
```

Piece located in (2, 7).

Definition at line 129 of file GameBoardView.java.

6.43.4.17 f2c8

```
Circle view.GameBoardView.f2c8 [private]
```

Piece located in (2, 8).

Definition at line 134 of file GameBoardView.java.

6.43.4.18 f3c1

```
Circle view.GameBoardView.f3c1 [private]
```

Piece located in (3, 1).

Definition at line 139 of file GameBoardView.java.

6.43.4.19 f3c2

```
Circle view.GameBoardView.f3c2 [private]
```

Piece located in (3, 2).

Definition at line 144 of file GameBoardView.java.

6.43.4.20 f3c3

```
Circle view.GameBoardView.f3c3 [private]
```

Piece located in (3, 3).

Definition at line 149 of file GameBoardView.java.

6.43.4.21 f3c4

```
Circle view.GameBoardView.f3c4  [private]
```

Piece located in (3, 4).

Definition at line 154 of file GameBoardView.java.

6.43.4.22 f3c5

```
Circle view.GameBoardView.f3c5  [private]
```

Piece located in (3, 5).

Definition at line 159 of file GameBoardView.java.

6.43.4.23 f3c6

```
Circle view.GameBoardView.f3c6  [private]
```

Piece located in (3, 6).

Definition at line 164 of file GameBoardView.java.

6.43.4.24 f3c7

```
Circle view.GameBoardView.f3c7  [private]
```

Piece located in (3, 7).

Definition at line 169 of file GameBoardView.java.

6.43.4.25 f3c8

```
Circle view.GameBoardView.f3c8  [private]
```

Piece located in (3, 8).

Definition at line 174 of file GameBoardView.java.

6.43.4.26 f4c1

```
Circle view.GameBoardView.f4c1  [private]
```

Piece located in (4, 1).

Definition at line 179 of file GameBoardView.java.

6.43.4.27 f4c2

```
Circle view.GameBoardView.f4c2  [private]
```

Piece located in (4, 2).

Definition at line 184 of file GameBoardView.java.

6.43.4.28 f4c3

```
Circle view.GameBoardView.f4c3  [private]
```

Piece located in (4, 3).

Definition at line 189 of file GameBoardView.java.

6.43.4.29 f4c4

```
Circle view.GameBoardView.f4c4  [private]
```

Piece located in (4, 4).

Definition at line 194 of file GameBoardView.java.

6.43.4.30 f4c5

```
Circle view.GameBoardView.f4c5  [private]
```

Piece located in (4, 5).

Definition at line 199 of file GameBoardView.java.

6.43.4.31 f4c6

```
Circle view.GameBoardView.f4c6  [private]
```

Piece located in (4, 6).

Definition at line 204 of file GameBoardView.java.

6.43.4.32 f4c7

```
Circle view.GameBoardView.f4c7  [private]
```

Piece located in (4, 7).

Definition at line 209 of file GameBoardView.java.

6.43.4.33 f4c8

```
Circle view.GameBoardView.f4c8  [private]
```

Piece located in (4, 8).

Definition at line 214 of file GameBoardView.java.

6.43.4.34 f5c1

```
Circle view.GameBoardView.f5c1  [private]
```

Piece located in (5, 1).

Definition at line 219 of file GameBoardView.java.

6.43.4.35 f5c2

```
Circle view.GameBoardView.f5c2  [private]
```

Piece located in (5, 2).

Definition at line 224 of file GameBoardView.java.

6.43.4.36 f5c3

```
Circle view.GameBoardView.f5c3  [private]
```

Piece located in (5, 3).

Definition at line 229 of file GameBoardView.java.

6.43.4.37 f5c4

```
Circle view.GameBoardView.f5c4  [private]
```

Piece located in (5, 4).

Definition at line 234 of file GameBoardView.java.

6.43.4.38 f5c5

```
Circle view.GameBoardView.f5c5  [private]
```

Piece located in (5, 5).

Definition at line 239 of file GameBoardView.java.

6.43.4.39 f5c6

```
Circle view.GameBoardView.f5c6  [private]
```

Piece located in (5, 6).

Definition at line 244 of file GameBoardView.java.

6.43.4.40 f5c7

```
Circle view.GameBoardView.f5c7  [private]
```

Piece located in (5, 7).

Definition at line 249 of file GameBoardView.java.

6.43.4.41 f5c8

```
Circle view.GameBoardView.f5c8  [private]
```

Piece located in (5, 8).

Definition at line 254 of file GameBoardView.java.

6.43.4.42 f6c1

```
Circle view.GameBoardView.f6c1  [private]
```

Piece located in (6, 1).

Definition at line 259 of file GameBoardView.java.

6.43.4.43 f6c2

```
Circle view.GameBoardView.f6c2  [private]
```

Piece located in (6, 2).

Definition at line 264 of file GameBoardView.java.

6.43.4.44 f6c3

```
Circle view.GameBoardView.f6c3  [private]
```

Piece located in (6, 3).

Definition at line 269 of file GameBoardView.java.

6.43.4.45 f6c4

```
Circle view.GameBoardView.f6c4  [private]
```

Piece located in (6, 4).

Definition at line 274 of file GameBoardView.java.

6.43.4.46 f6c5

```
Circle view.GameBoardView.f6c5 [private]
```

Piece located in (6, 5).

Definition at line 279 of file GameBoardView.java.

6.43.4.47 f6c6

```
Circle view.GameBoardView.f6c6 [private]
```

Piece located in (6, 6).

Definition at line 284 of file GameBoardView.java.

6.43.4.48 f6c7

```
Circle view.GameBoardView.f6c7 [private]
```

Piece located in (6, 7).

Definition at line 289 of file GameBoardView.java.

6.43.4.49 f6c8

```
Circle view.GameBoardView.f6c8 [private]
```

Piece located in (6, 8).

Definition at line 294 of file GameBoardView.java.

6.43.4.50 f7c1

```
Circle view.GameBoardView.f7c1 [private]
```

Piece located in (7, 1).

Definition at line 299 of file GameBoardView.java.

6.43.4.51 f7c2

```
Circle view.GameBoardView.f7c2  [private]
```

Piece located in (7, 2).

Definition at line 304 of file GameBoardView.java.

6.43.4.52 f7c3

```
Circle view.GameBoardView.f7c3  [private]
```

Piece located in (7, 3).

Definition at line 309 of file GameBoardView.java.

6.43.4.53 f7c4

```
Circle view.GameBoardView.f7c4  [private]
```

Piece located in (7, 4).

Definition at line 314 of file GameBoardView.java.

6.43.4.54 f7c5

```
Circle view.GameBoardView.f7c5  [private]
```

Piece located in (7, 5).

Definition at line 319 of file GameBoardView.java.

6.43.4.55 f7c6

```
Circle view.GameBoardView.f7c6  [private]
```

Piece located in (7, 6).

Definition at line 324 of file GameBoardView.java.

6.43.4.56 f7c7

```
Circle view.GameBoardView.f7c7  [private]
```

Piece located in (7, 7).

Definition at line 329 of file GameBoardView.java.

6.43.4.57 f7c8

```
Circle view.GameBoardView.f7c8  [private]
```

Piece located in (7, 8).

Definition at line 334 of file GameBoardView.java.

6.43.4.58 f8c1

```
Circle view.GameBoardView.f8c1  [private]
```

Piece located in (8, 1).

Definition at line 339 of file GameBoardView.java.

6.43.4.59 f8c2

```
Circle view.GameBoardView.f8c2  [private]
```

Piece located in (8, 2).

Definition at line 344 of file GameBoardView.java.

6.43.4.60 f8c3

```
Circle view.GameBoardView.f8c3  [private]
```

Piece located in (8, 3).

Definition at line 349 of file GameBoardView.java.

6.43.4.61 f8c4

```
Circle view.GameBoardView.f8c4 [private]
```

Piece located in (8, 4).

Definition at line 354 of file GameBoardView.java.

6.43.4.62 f8c5

```
Circle view.GameBoardView.f8c5 [private]
```

Piece located in (8, 5).

Definition at line 359 of file GameBoardView.java.

6.43.4.63 f8c6

```
Circle view.GameBoardView.f8c6 [private]
```

Piece located in (8, 6).

Definition at line 364 of file GameBoardView.java.

6.43.4.64 f8c7

```
Circle view.GameBoardView.f8c7 [private]
```

Piece located in (8, 7).

Definition at line 369 of file GameBoardView.java.

6.43.4.65 f8c8

```
Circle view.GameBoardView.f8c8 [private]
```

Piece located in (8, 8).

Definition at line 374 of file GameBoardView.java.

6.43.4.66 save

```
Text view.GameBoardView.save  [private]
```

Save board button text.

Definition at line 379 of file GameBoardView.java.

6.43.4.67 saveButton

```
Rectangle view.GameBoardView.saveButton  [private]
```

Save board button.

Definition at line 384 of file GameBoardView.java.

6.43.4.68 surrender

```
Text view.GameBoardView.surrender  [private]
```

Surrender board button text.

Definition at line 389 of file GameBoardView.java.

6.43.4.69 surrenderButton

```
Rectangle view.GameBoardView.surrenderButton  [private]
```

Surrender board button text.

Definition at line 394 of file GameBoardView.java.

6.43.4.70 tieIcon

```
ImageView view.GameBoardView.tieIcon  [private]
```

Tie icon image.

Definition at line 399 of file GameBoardView.java.

6.43.4.71 winIcon

```
ImageView view.GameBoardView.winIcon  [private]
```

Win cup icon image.

Definition at line 404 of file GameBoardView.java.

6.43.4.72 gameResult

```
Label view.GameBoardView.gameResult  [private]
```

Exception output message label.

Definition at line 409 of file GameBoardView.java.

6.43.4.73 player2

```
Label view.GameBoardView.player2  [private]
```

Second player name label.

Definition at line 414 of file GameBoardView.java.

6.43.4.74 player2Turn

```
Label view.GameBoardView.player2Turn  [private]
```

Second player turn label.

Definition at line 419 of file GameBoardView.java.

6.43.4.75 player2Pieces

```
Label view.GameBoardView.player2Pieces  [private]
```

Second player number of pieces label.

Definition at line 424 of file GameBoardView.java.

6.43.4.76 player2Type

```
Label view.GameBoardView.player2Type [private]
```

Second player type tag label.

Definition at line 429 of file GameBoardView.java.

6.43.4.77 player1

```
Label view.GameBoardView.player1 [private]
```

First player name label.

Definition at line 434 of file GameBoardView.java.

6.43.4.78 player1Turn

```
Label view.GameBoardView.player1Turn [private]
```

First player turn label.

Definition at line 439 of file GameBoardView.java.

6.43.4.79 player1Pieces

```
Label view.GameBoardView.player1Pieces [private]
```

First player number of pieces label.

Definition at line 444 of file GameBoardView.java.

6.43.4.80 player1Type

```
Label view.GameBoardView.player1Type [private]
```

First player type tag label.

Definition at line 449 of file GameBoardView.java.

6.43.4.81 assistedMode

```
RadioButton view.GameBoardView.assistedMode  [private]
```

Assisted mode option radio button.

Definition at line 454 of file GameBoardView.java.

6.43.4.82 board

```
JSONObject view.GameBoardView.board  [private]
```

Current board.

Definition at line 458 of file GameBoardView.java.

6.43.4.83 game

```
JSONObject view.GameBoardView.game  [private]
```

Current game.

Definition at line 462 of file GameBoardView.java.

6.43.4.84 players

```
Pair<JSONObject, JSONObject> view.GameBoardView.players  [private]
```

Current players.

Definition at line 466 of file GameBoardView.java.

6.43.4.85 user

```
JSONObject view.GameBoardView.user  [private]
```

Current user.

Definition at line 470 of file GameBoardView.java.

6.43.4.86 turnPlayerID

```
UUID view.GameBoardView.turnPlayerID  [private]
```

Current ID of the turn's player.

Definition at line 474 of file GameBoardView.java.

6.43.4.87 turnPlayerIsBot

```
Boolean view.GameBoardView.turnPlayerIsBot  [private]
```

Whether the current turn's player is a bot.

Definition at line 478 of file GameBoardView.java.

6.43.4.88 turnPieceType

```
String view.GameBoardView.turnPieceType  [private]
```

Current turn's piece type.

Definition at line 482 of file GameBoardView.java.

6.43.4.89 isSpectating

```
Boolean view.GameBoardView.isSpectating  [private]
```

Whether the current user is spectating a game.

Definition at line 486 of file GameBoardView.java.

6.43.4.90 isVsBot

```
Boolean view.GameBoardView.isVsBot  [private]
```

Whether the current user is vs bot.

Definition at line 490 of file GameBoardView.java.

6.43.4.91 timer

```
Timer view.GameBoardView.timer [private]
```

Timer to automatically perform bot placing trough runtimes threads asynchronously.

Definition at line 494 of file GameBoardView.java.

The documentation for this class was generated from the following file:

- GameBoardView.java

6.44 domain.GameCtrl Class Reference

Game domain sub-controller. It communicates with the main domain controller, the game repository controller, the configuration repository controller and the player repository controller in order to source the necessary components to manage games.

Public Member Functions

- GameCtrl ()

  Creator method that creates an instance of Game Controller.
- Game create (UUID player1ID, UUID player2ID, String configurationName, UUID creatorID) throws Invalid↩ PlayersException, InvalidConfigurationException, InvalidBoardException

  Lets the current user create a new game, selecting both players and a configuration of rules and initial board.
- Game getGame (String name, UUID playerID) throws NotPlayerException

  Returns the game identified by its name and any of the participant player IDs.
- Board getPlayingBoard (String name, UUID playerID) throws NotPlayerException

  Returns the playing board associated with the given game name and any of the participant player IDs.
- ArrayList< String > list (UUID playerID)

  Returns a list of all games names identified by any of the participant player IDs.
- Game save (Game game, Board playingBoard, UUID playerID) throws NotPlayerException

  Lets the current user manually save the current game and playing board state.
- Game play (Game game) throws FinishedGameException

  Lets the current user load a saved game or a newly created one, and start playing on it.
- Game surrender (Game game, UUID surrendeeID) throws NotPlayerException, FinishedGameException, NotStartedGameException

  Lets a player of the current game surrender, setting the winner as the opponent.
- Game finish (Game game, UUID winnerID) throws NotPlayerException, FinishedGameException, Not↩ StartedGameException

  Lets the system to automatically finish the game when any players win or when there aren't any valid positions left to place a piece on the board, setting the winner of the game or setting that the game has ended in a draw.
- void checkPlaceRights (Game game, UUID playerID, PieceType pieceType) throws NotPlayerException, NotPlayerPieceException, NotPlayerTurnException, FinishedGameException, NotStartedGameException

  Lets the system check whether the player that wants to place a piece on the board of the current game is able to do so, that is, its his/her turn and the piece type its his/hers.
- Game nextTurn (Game game) throws FinishedGameException, NotStartedGameException

  Lets the system to automatically pass the turn of the current game.

Private Attributes

- GameRepositoryCtrl repositoryCtrl

  Game repository controller.
- ConfigurationRepositoryCtrl configurationRepositoryCtrl

  Configuration repository controller.
- PlayerRepositoryCtrl playerRepositoryCtrl

  Player repository controller.

6.44.1 Detailed Description

Game domain sub-controller. It communicates with the main domain controller, the game repository controller, the configuration repository controller and the player repository controller in order to source the necessary components to manage games.

By Alex Rodriguez.

See also

domain.Game

Definition at line 36 of file GameCtrl.java.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 GameCtrl()

```
domain.GameCtrl.GameCtrl ( )
```

Creator method that creates an instance of Game Controller.

Precondition

True

Postcondition

Instance of GameCtrl is created with the default values

Definition at line 61 of file GameCtrl.java.

```
61                     {
62         this.repositoryCtrl = new GameRepositoryCtrl();
63         this.configurationRepositoryCtrl = new ConfigurationRepositoryCtrl();
64         this.playerRepositoryCtrl = new PlayerRepositoryCtrl();
65     }
```

6.44.3 Member Function Documentation

6.44.3.1 create()

```
Game domain.GameCtrl.create (
            UUID player1ID,
            UUID player2ID,
            String configurationName,
            UUID creatorID ) throws InvalidPlayersException, InvalidConfigurationException,
InvalidBoardException
```

Lets the current user create a new game, selecting both players and a configuration of rules and initial board.

Precondition

True

Postcondition

A Game is returned with its specified attributes if no exception is thrown. Else, an exception will be thrown

Parameters

player1ID	UUID of Player1
player2ID	UUID of Player2
configuration	Instance of a Configuration
creatorID	UUID of the creator User.

Returns

Game

Definition at line 79 of file GameCtrl.java.

```
                                                                                                {

        if (player1ID.equals(player2ID))
            throw new InvalidPlayersException();

        JSONObject rawPlayer1 = this.playerRepositoryCtrl.get(player1ID);
        if (rawPlayer1 == null)
            throw new InvalidPlayersException();

        JSONObject rawPlayer2 = this.playerRepositoryCtrl.get(player2ID);
        if (rawPlayer2 == null)
            throw new InvalidPlayersException();

        if (rawPlayer1.getBoolean("is_deleted") || rawPlayer2.getBoolean("is_deleted"))
            throw new InvalidPlayersException();

        if (rawPlayer1.getString("type").equals("BOT") && rawPlayer2.getString("type").equals("BOT"))
            if (rawPlayer1.getInt("difficulty") == rawPlayer2.getInt("difficulty"))
                throw new InvalidPlayersException();

        JSONObject rawConfiguration =
    this.configurationRepositoryCtrl.getConfiguration(configurationName);
        if (rawConfiguration == null)
            throw new InvalidConfigurationException();

        JSONObject rawBoard = this.configurationRepositoryCtrl.getBoard(configurationName);
        if (rawBoard == null)
            throw new InvalidBoardException();

        Board playingBoard = new Board(rawBoard);

        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        String name = String.format("%s VS %s | %s", rawPlayer1.getString("name"),
    rawPlayer2.getString("name"),
                now.format(dateFormat));

        Game game = new Game(name, player1ID, player2ID, configurationName, creatorID);

        this.repositoryCtrl.save(game.serialize(), playingBoard.serialize());
        return game;
    }
```

6.44.3.2 getGame()

```
Game domain.GameCtrl.getGame (
             String name,
             UUID playerID ) throws NotPlayerException
```

Returns the game identified by its name and any of the participant player IDs.

Precondition

True

Postcondition

Game is returned specified by its name and a Players UUID if no excepti

Parameters

name	Name of a Game
playerID	UUID of a Player

Returns

Game

Definition at line 129 of file GameCtrl.java.

```
                                                                                {
        if (name.isBlank())
            throw new NotPlayerException();

        JSONObject rawGame = this.repositoryCtrl.getGame(name);

        if (rawGame == null)
            throw new NotPlayerException();

        Game game = new Game(rawGame);

        if (!game.getPlayer1ID().equals(playerID) && !game.getPlayer2ID().equals(playerID)
                && !game.getCreatorID().equals(playerID))
            throw new NotPlayerException();

        return game;
    }
```

6.44.3.3 getPlayingBoard()

```
Board domain.GameCtrl.getPlayingBoard (
             String name,
             UUID playerID ) throws NotPlayerException
```

Returns the playing board associated with the given game name and any of the participant player IDs.

Precondition

True

Postcondition

Returns the playing board of a game if no exception is thrown. Else, an exception will be thrown.

Parameters

name	Name of a Game
playerID	UUID of a Player

Returns

Board

Definition at line 155 of file GameCtrl.java.

```
                                                                                                       {
        if (name.isBlank())
            throw new NotPlayerException();

        JSONObject rawPlayingBoard = this.repositoryCtrl.getBoard(name);

        if (rawPlayingBoard == null)
            throw new NotPlayerException();

        return new Board(rawPlayingBoard);
    }
```

6.44.3.4 list()

```
ArrayList<String> domain.GameCtrl.list (
             UUID playerID )
```

Returns a list of all games names identified by any of the participant player IDs.

Precondition

True

Postcondition

An ArrayList of all the names of the Games will be returned.

Parameters

playerID	UUID of a Player.

Returns

ArrayList<String>

Definition at line 174 of file GameCtrl.java.

```
                                         {
        return this.repositoryCtrl.listGames(playerID);
    }
```

6.44.3.5 save()

```
Game domain.GameCtrl.save (
            Game game,
            Board playingBoard,
            UUID playerID ) throws NotPlayerException
```

Lets the current user manually save the current game and playing board state.

Precondition

game and playingBoard aren't null

Postcondition

The saved Game is returned if no exception is thrown. Else, an exception will be thrown

Parameters

game	Game instance
playingBoard	Board instance
playerID	UUID instance.

Returns

Game.

Definition at line 187 of file GameCtrl.java.

```
                                                                                                  {
        if (!game.getPlayer1ID().equals(playerID) && !game.getPlayer2ID().equals(playerID)
                && !game.getCreatorID().equals(playerID))
            throw new NotPlayerException();

        this.repositoryCtrl.save(game.serialize(), playingBoard.serialize());
        return game;
    }
```

6.44.3.6 play()

```
Game domain.GameCtrl.play (
            Game game ) throws FinishedGameException
```

Lets the current user load a saved game or a newly created one, and start playing on it.

Precondition

game is not null

Postcondition

The playing Game is returned if no exception was thrown. Else, an exception will be thrown.

Parameters

game	Game instance

Returns

Playing Game

Definition at line 203 of file GameCtrl.java.

```
                                                            {
            game.play();
            return game;
        }
```

6.44.3.7 surrender()

```
Game domain.GameCtrl.surrender (
            Game game,
            UUID surrendeeID ) throws NotPlayerException, FinishedGameException, NotStartedGameException
```

Lets a player of the current game surrender, setting the winner as the opponent.

Precondition

game is not null

Postcondition

The surrendered Game is returned if no exception was thrown. Else, an exception will be thrown.

Parameters

game	Game instance
surrendeeID	UUID of Player

Returns

Game

Definition at line 216 of file GameCtrl.java.

```
                                                                                                  {
            game.surrender(surrendeeID);
            return game;
        }
```

6.44.3.8 finish()

```
Game domain.GameCtrl.finish (
            Game game,
            UUID winnerID ) throws NotPlayerException, FinishedGameException, NotStartedGameException
```

Lets the system to automatically finish the game when any players win or when there aren't any valid positions left to place a piece on the board, setting the winner of the game or setting that the game has ended in a draw.

Precondition

game is not null

Postcondition

The finished Game is returned if no exception was thrown. Else, an exception will be thrown.

Parameters

game	Game instance
winnerID	UUID of Player

Returns

Game

Definition at line 231 of file GameCtrl.java.

```
                                                                                                  {
            game.finish(winnerID);
            return game;
        }
```

6.44.3.9 checkPlaceRights()

```
void domain.GameCtrl.checkPlaceRights (
            Game game,
            UUID playerID,
            PieceType pieceType ) throws NotPlayerException, NotPlayerPieceException, NotPlayerTurnException,
FinishedGameException, NotStartedGameException
```

Lets the system check whether the player that wants to place a piece on the board of the current game is able to do so, that is, its his/her turn and the piece type its his/hers.

Precondition

game is not null

Postcondition

If the Player is able to place a piece, nothing happens. Else, an exception will be thrown.

Parameters

game	Game instance
winnerID	UUID of Player
pieceType	PieceType

Definition at line 246 of file GameCtrl.java.

```
        {
        game.checkPlaceRights(playerID, pieceType);
    }
```

6.44.3.10 nextTurn()

```
Game domain.GameCtrl.nextTurn (
            Game game ) throws FinishedGameException, NotStartedGameException
```

Lets the system to automatically pass the turn of the current game.

Precondition

game is not null

Postcondition

Returns the Game with the next turn if no exception was thrown. Else, an exception will be thrown.

Parameters

game	Instance of a Game

Returns

Game

Definition at line 258 of file GameCtrl.java.

```
                                                                                    {
        game.nextTurn();
        return game;
    }
```

6.44.4 Member Data Documentation

6.44.4.1 repositoryCtrl

`GameRepositoryCtrl domain.GameCtrl.repositoryCtrl [private]`

Game repository controller.

Definition at line 42 of file GameCtrl.java.

6.44.4.2 configurationRepositoryCtrl

`ConfigurationRepositoryCtrl domain.GameCtrl.configurationRepositoryCtrl [private]`

Configuration repository controller.

Definition at line 47 of file GameCtrl.java.

6.44.4.3 playerRepositoryCtrl

`PlayerRepositoryCtrl domain.GameCtrl.playerRepositoryCtrl [private]`

Player repository controller.

Definition at line 52 of file GameCtrl.java.

The documentation for this class was generated from the following file:

- GameCtrl.java

6.45 test.driver.GameDriver Class Reference

Implements the different options for the Game driver application. By Alex Rodriguez.

Public Member Functions

- GameDriver ()
- void start ()

Public Attributes

- Game currentGame

Private Member Functions

- void mainMenu ()
- void create ()
- void getName ()
- void setName ()
- void getPlayer1ID ()
- void getPlayer2ID ()
- void getConfigurationName ()
- void setConfigurationName ()
- void getTurn ()
- void setTurn ()
- void getState ()
- void setState ()
- void getWinnerID ()
- void getCreatorID ()
- void getCreatedAt ()
- void serialize ()
- void deserialize ()
- void play ()
- void surrender ()
- void finish ()
- void checkPlaceRights ()
- void nextTurn ()

Additional Inherited Members

6.45.1 Detailed Description

Implements the different options for the Game driver application. By Alex Rodriguez.

Definition at line 21 of file GameDriver.java.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 GameDriver()

```
test.driver.GameDriver.GameDriver ( )
```

Definition at line 28 of file GameDriver.java.

```
                        {
        this.currentGame = null;
    }
```

6.45.3 Member Function Documentation

6.45.3.1 start()

```
void test.driver.GameDriver.start ( )
```

Definition at line 34 of file GameDriver.java.

```
                         {
        while (true) {
            this.mainMenu();
        }
    }
```

6.45.3.2 mainMenu()

```
void test.driver.GameDriver.mainMenu ( )  [private]
```

Definition at line 40 of file GameDriver.java.

```
                             {
        String title = (this.currentGame != null ? String.format("Current: %s\n",
    this.currentGame.getName()) : null);
        switch (Driver.menu(title, "Game Driver",
                new Pair<String, String>("1", "Create Game"),
                new Pair<String, String>("2", "Get name"),
                new Pair<String, String>("3", "Set name"),
                new Pair<String, String>("4", "Get player1ID"),
                new Pair<String, String>("5", "Get player2ID"),
                new Pair<String, String>("6", "Get configurationName"),
                new Pair<String, String>("7", "Set configurationName"),
                new Pair<String, String>("8", "Get turn"),
                new Pair<String, String>("9", "Set turn"),
                new Pair<String, String>("10", "Get state"),
                new Pair<String, String>("11", "Set state"),
                new Pair<String, String>("12", "Get winnerID"),
                new Pair<String, String>("13", "Get creatorID"),
                new Pair<String, String>("14", "Get createdAt"),
                new Pair<String, String>("15", "Serialize to JSON"),
                new Pair<String, String>("16", "Deserialize from JSON"),
                new Pair<String, String>("17", "Execute play"),
                new Pair<String, String>("18", "Execute surrender"),
                new Pair<String, String>("19", "Execute finish"),
                new Pair<String, String>("20", "Execute checkPlaceRights"),
                new Pair<String, String>("21", "Execute nextTurn"))) {
        case "1":
            this.create();
            break;
        case "2":
            this.getName();
            break;
        case "3":
            this.setName();
            break;
        case "4":
            this.getPlayer1ID();
            break;
        case "5":
            this.getPlayer2ID();
            break;
        case "6":
            this.getConfigurationName();
            break;
        case "7":
            this.setConfigurationName();
            break;
        case "8":
            this.getTurn();
            break;
        case "9":
            this.setTurn();
```

```
90              break;
91          case "10":
92              this.getState();
93              break;
94          case "11":
95              this.setState();
96              break;
97          case "12":
98              this.getWinnerID();
99              break;
100         case "13":
101             this.getCreatorID();
102             break;
103         case "14":
104             this.getCreatedAt();
105             break;
106         case "15":
107             this.serialize();
108             break;
109         case "16":
110             this.deserialize();
111             break;
112         case "17":
113             this.play();
114             break;
115         case "18":
116             this.surrender();
117             break;
118         case "19":
119             this.finish();
120             break;
121         case "20":
122             this.checkPlaceRights();
123             break;
124         case "21":
125             this.nextTurn();
126             break;
127         }
128         Driver.pause();
129     }
```

6.45.3.3 create()

`void test.driver.GameDriver.create ( ) [private]`

Definition at line 131 of file GameDriver.java.

```
131                      {
132         System.out.println(
133                 "Take into account that UUIDs will be randomly generated because typing them in will be
       a hassle.\n");
134         String name = Driver.input("Name?");
135         String configurationName = Driver.input("Configuration name?");
136         try {
137             Game game = new Game("Default name", UUID.randomUUID(), UUID.randomUUID(), "Default
       configurationName",
138                     UUID.randomUUID());
139             game.setName(name);
140             game.setConfigurationName(configurationName);
141             this.currentGame = game;
142             System.out.println(String.format("%s created successfully!", this.currentGame.getName()));
143         } catch (Exception e) {
144             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
       e.getMessage()));
145         }
146     }
```

6.45.3.4 getName()

`void test.driver.GameDriver.getName ( ) [private]`

Definition at line 148 of file GameDriver.java.

```
                              {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(String.format("%s's name is: %s", this.currentGame.getName(),
      this.currentGame.getName()));
    }
```

6.45.3.5 setName()

`void test.driver.GameDriver.setName ( ) [private]`

Definition at line 157 of file GameDriver.java.

```
                              {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        try {
            this.currentGame.setName(Driver.input("Name?"));
            System.out.println(String.format("%s's name changed successfully!",
      this.currentGame.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
      e.getMessage()));
        }
    }
```

6.45.3.6 getPlayer1ID()

`void test.driver.GameDriver.getPlayer1ID ( ) [private]`

Definition at line 171 of file GameDriver.java.

```
                                   {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(
                String.format("%s's player1ID is: %s", this.currentGame.getName(),
      this.currentGame.getPlayer1ID()));
    }
```

6.45.3.7 getPlayer2ID()

`void test.driver.GameDriver.getPlayer2ID ( ) [private]`

Definition at line 181 of file GameDriver.java.

```
                                   {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(
                String.format("%s's player2ID is: %s", this.currentGame.getName(),
      this.currentGame.getPlayer2ID()));
    }
```

6.45.3.8 getConfigurationName()

```
void test.driver.GameDriver.getConfigurationName ( )  [private]
```

Definition at line 191 of file GameDriver.java.

```
                                            {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(String.format("%s's configurationName is: %s", this.currentGame.getName(),
                this.currentGame.getConfigurationName()));
    }
```

6.45.3.9 setConfigurationName()

```
void test.driver.GameDriver.setConfigurationName ( )  [private]
```

Definition at line 201 of file GameDriver.java.

```
                                            {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        try {
            this.currentGame.setConfigurationName(Driver.input("Configuration name?"));
            System.out
                    .println(String.format("%s's configurationName changed successfully!",
     this.currentGame.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
     e.getMessage()));
        }
    }
```

6.45.3.10 getTurn()

```
void test.driver.GameDriver.getTurn ( )  [private]
```

Definition at line 216 of file GameDriver.java.

```
                              {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(String.format("%s's turn is: %s", this.currentGame.getName(),
     this.currentGame.getTurn()));
    }
```

6.45.3.11 setTurn()

```
void test.driver.GameDriver.setTurn ( )  [private]
```

Definition at line 225 of file GameDriver.java.

```
                            {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        switch (Driver.menu(null, "Select Turn",
                new Pair<String, String>("1", "PLAYER 1"),
                new Pair<String, String>("2", "PLAYER 2"))) {
        case "1":
            this.currentGame.setTurn(PieceType.PLAYER1);
            break;
        case "2":
            this.currentGame.setTurn(PieceType.PLAYER2);
            break;
        }
        System.out.println(String.format("%s's turn changed successfully!",
    this.currentGame.getName()));
    }
```

6.45.3.12 getState()

```
void test.driver.GameDriver.getState ( )  [private]
```

Definition at line 244 of file GameDriver.java.

```
                             {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(String.format("%s's state is: %s", this.currentGame.getName(),
    this.currentGame.getState()));
    }
```

6.45.3.13 setState()

```
void test.driver.GameDriver.setState ( )  [private]
```

Definition at line 253 of file GameDriver.java.

```
                             {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        switch (Driver.menu(null, "Select State",
                new Pair<String, String>("1", "NOT_STARTED"),
                new Pair<String, String>("2", "IN_PROGRESS"),
                new Pair<String, String>("3", "FINISHED"))) {
        case "1":
            this.currentGame.setState(GameState.NOT_STARTED);
            break;
        case "2":
            this.currentGame.setState(GameState.IN_PROGRESS);
            break;
        case "3":
            this.currentGame.setState(GameState.FINISHED);
            break;
        }
        System.out.println(String.format("%s's state changed successfully!",
    this.currentGame.getName()));
    }
```

6.45.3.14 getWinnerID()

```
void test.driver.GameDriver.getWinnerID ( )  [private]
```

Definition at line 276 of file GameDriver.java.

```
                              {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(
                String.format("%s's winnerID is: %s", this.currentGame.getName(),
       this.currentGame.getWinnerID()));
    }
```

6.45.3.15 getCreatorID()

```
void test.driver.GameDriver.getCreatorID ( )  [private]
```

Definition at line 286 of file GameDriver.java.

```
                               {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(
                String.format("%s's creatorID is: %s", this.currentGame.getName(),
       this.currentGame.getCreatorID()));
    }
```

6.45.3.16 getCreatedAt()

```
void test.driver.GameDriver.getCreatedAt ( )  [private]
```

Definition at line 296 of file GameDriver.java.

```
                               {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        System.out.println(String.format("%s's createdAt is: %s", this.currentGame.getName(),
                this.currentGame.getCreatedAt().format(dateFormat)));
    }
```

6.45.3.17 serialize()

```
void test.driver.GameDriver.serialize ( )  [private]
```

Definition at line 307 of file GameDriver.java.

```
                            {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(String.format("%s's serialized to JSON is: %s", this.currentGame.getName(),
                this.currentGame.serialize().toString(2)));
    }
```

6.45.3.18 deserialize()

```
void test.driver.GameDriver.deserialize ( )  [private]
```

Definition at line 317 of file GameDriver.java.

```
                             {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        System.out.println(this.currentGame.serialize().toString(2));
        this.currentGame = new Game(this.currentGame.serialize());
        System.out.println(
                String.format("\n%s's deserialized from the above JSON successfully!\n",
     this.currentGame.getName()));
        System.out.println(String.format("name:\t\t\t%s", this.currentGame.getName()));
        System.out.println(String.format("player1ID:\t\t%s", this.currentGame.getPlayer1ID()));
        System.out.println(String.format("player2ID:\t\t%s", this.currentGame.getPlayer2ID()));
        System.out.println(String.format("configurationName:\t%s",
     this.currentGame.getConfigurationName()));
        System.out.println(String.format("turn:\t\t\t%s", this.currentGame.getTurn()));
        System.out.println(String.format("state:\t\t\t%s", this.currentGame.getState()));
        System.out.println(String.format("winnerID:\t\t%s", this.currentGame.getWinnerID()));
        System.out.println(String.format("creatorID:\t\t%s", this.currentGame.getCreatorID()));
        DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        System.out.println(String.format("createdAt:\t\t%s",
     this.currentGame.getCreatedAt().format(dateFormat)));
    }
```

6.45.3.19 play()

```
void test.driver.GameDriver.play ( )  [private]
```

Definition at line 339 of file GameDriver.java.

```
                      {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        try {
            this.currentGame.play();
            System.out.println(String.format("The Game state has changed to %s",
     this.currentGame.getState()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
     e.getMessage()));
        }
    }
```

6.45.3.20 surrender()

```
void test.driver.GameDriver.surrender ( )  [private]
```

Definition at line 353 of file GameDriver.java.

```
                           {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        try {
            switch (Driver.menu(null, "Select who surrenders",
                    new Pair<String, String>("1", "PLAYER 1"),
                    new Pair<String, String>("2", "PLAYER 2"))) {
            case "1":
                this.currentGame.surrender(this.currentGame.getPlayer1ID());
                System.out.println("PLAYER 2 has won the Game");
                break;
            case "2":
                this.currentGame.surrender(this.currentGame.getPlayer2ID());
                System.out.println("PLAYER 1 has won the Game");
                break;
            }
            System.out.println(String.format("The Game winnerID has changed to %s",
this.currentGame.getWinnerID()));
            System.out.println(String.format("The Game state has changed to %s",
this.currentGame.getState()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
        }
    }
```

6.45.3.21 finish()

`void test.driver.GameDriver.finish ( ) [private]`

Definition at line 379 of file GameDriver.java.

```
                          {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        try {
            switch (Driver.menu(null, "Select who wins",
                    new Pair<String, String>("1", "PLAYER 1"),
                    new Pair<String, String>("2", "PLAYER 2"),
                    new Pair<String, String>("3", "DRAW"))) {
            case "1":
                this.currentGame.finish(this.currentGame.getPlayer1ID());
                System.out.println("PLAYER 1 has won the Game");
                break;
            case "2":
                this.currentGame.finish(this.currentGame.getPlayer2ID());
                System.out.println("PLAYER 2 has won the Game");
                break;
            case "3":
                this.currentGame.finish(null);
                System.out.println("The Game has resulted in a draw");
                break;
            }
            System.out.println(String.format("The Game winnerID has changed to %s",
this.currentGame.getWinnerID()));
            System.out.println(String.format("The Game state has changed to %s",
this.currentGame.getState()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
        }
    }
```

6.45.3.22 checkPlaceRights()

`void test.driver.GameDriver.checkPlaceRights ( ) [private]`

Definition at line 410 of file GameDriver.java.

```
                                    {
        if (this.currentGame == null) {
            System.out.println("No current Game!");
            return;
        }

        UUID playerID = null;
        switch (Driver.menu(null, "Select who places",
                new Pair<String, String>("1", "PLAYER 1"),
                new Pair<String, String>("2", "PLAYER 2"))) {
        case "1":
            playerID = this.currentGame.getPlayer1ID();
            break;
        case "2":
            playerID = this.currentGame.getPlayer2ID();
            break;
        }
        try {
            switch (Driver.menu(null, "Select piece type",
                    new Pair<String, String>("1", "PLAYER 1 pieces"),
                    new Pair<String, String>("2", "PLAYER 2 pieces"))) {
            case "1":
                this.currentGame.checkPlaceRights(playerID, PieceType.PLAYER1);
                break;
            case "2":
                this.currentGame.checkPlaceRights(playerID, PieceType.PLAYER2);
                break;
            }
            System.out.println("The player did place the piece successfully");
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s", e.getMessage()));
        }
    }
```

6.45.3.23 nextTurn()

void test.driver.GameDriver.nextTurn ( ) [private]

Definition at line 444 of file GameDriver.java.

```
                         {
     if (this.currentGame == null) {
         System.out.println("No current Game!");
         return;
     }

     try {
         this.currentGame.nextTurn();
         System.out.println(String.format("The Game turn has changed to %s", this.currentGame.getTurn()));
     } catch (Exception e) {
         System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s", e.getMessage()));
     }
 }
```

6.45.4 Member Data Documentation

6.45.4.1 currentGame

Game test.driver.GameDriver.currentGame

Definition at line 24 of file GameDriver.java.

The documentation for this class was generated from the following file:

- GameDriver.java

6.46 repository.GameRepository Class Reference

Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.

Public Member Functions

- GameRepository ()

    Create a GameRepository instance.
- void save (JSONObject game, JSONObject board)

    Save a Game into the game database.
- JSONObject getGame (String name)

    Get the Game by name from the game database or null if it does not exist.
- JSONObject getBoard (String name)

    Get the playing Board of a Game by name from the game database or null if it does not exist.
- Boolean existsGameByConfigurationName (String configurationName)

    Check whether there exists a Game with the given Configuration name in the game database.
- Boolean existsGameByPlayerID (String playerID)

    Check whether there exists a Game with the given Player ID in the game database.
- ArrayList< String > listGames (String playerID)

    List all Games by Player ID of the game database.

Additional Inherited Members

6.46.1 Detailed Description

Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.

See also

repository.Repository

Definition at line 18 of file GameRepository.java.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 GameRepository()

```
repository.GameRepository.GameRepository ( )
```

Create a GameRepository instance.

Precondition

The Game repository JSON files exists.

Postcondition

A GameRepository instance is created.

Definition at line 28 of file GameRepository.java.

```
                       {
        super(RepositoryType.GAME);
    }
```

6.46.3 Member Function Documentation

6.46.3.1 save()

```
void repository.GameRepository.save (
            JSONObject game,
            JSONObject board )
```

Save a Game into the game database.

Precondition

The Game repository JSON files exists.

Postcondition

The Game and its playing Board are saved into the game database.

Parameters

game	Game to be saved.
board	Playing Board of the Game to be saved.

Definition at line 41 of file GameRepository.java.

```
                                                      {
        String name = game.getString("name");
        game.put("board", board);
        this.createOrUpdate(name, game);
    }
```

6.46.3.2 getGame()

```
JSONObject repository.GameRepository.getGame (
            String name )
```

Get the Game by name from the game database or null if it does not exist.

Precondition

The Game repository JSON files exists.

Postcondition

A JSONObject representing the Game by name from the game database is returned or null if it does not exist.

Parameters

name	Name of the Game to be getted.

Returns

JSONObject that represents the Game by name from the game database or null if it does not exist.

Definition at line 54 of file GameRepository.java.

```
                                                        {
        JSONObject game = this.get(name);
        if (game == null)
            return null;

        game.remove("board");
        return game;
    }
```

6.46.3.3 getBoard()

```
JSONObject repository.GameRepository.getBoard (
            String name )
```

Get the playing Board of a Game by name from the game database or null if it does not exist.

Precondition

The Game repository JSON files exists.

Postcondition

A JSONObject representing the playing Board of a Game by name from the game database is returned or null if it does not exist.

Parameters

name	Name of the playing Board's Game to be getted.

Returns

JSONObject that represents the playing Board of a Game by name from the game database or null if it does not exist.

Definition at line 70 of file GameRepository.java.

```
                                                         {
        JSONObject game = this.get(name);
        if (game == null)
            return null;

        return game.getJSONObject("board");
    }
```

6.46.3.4 existsGameByConfigurationName()

```
Boolean repository.GameRepository.existsGameByConfigurationName (
            String configurationName )
```

Check whether there exists a Game with the given Configuration name in the game database.

Precondition

The Game repository JSON files exists.

Postcondition

If there exists a Game with the given Configuration name in the game database is returned true otherwise false.

Parameters

configurationName	Name of the Game's Configuration to be searched.

Returns

Whether there exists a Game with the given Configuration name in the game database.

Definition at line 85 of file GameRepository.java.

```
                                                                   {
        JSONObject all = this.list();

        JSONObject current;
        for (String key : all.keySet()) {
            current = all.getJSONObject(key);
            if (current.getString("configuration_name").equals(configurationName))
                return true;
        }

        return false;
    }
```

6.46.3.5 existsGameByPlayerID()

```
Boolean repository.GameRepository.existsGameByPlayerID (
            String playerID )
```

Check whether there exists a Game with the given Player ID in the game database.

Precondition

The Game repository JSON files exists.

Postcondition

If there exists a Game with the given Player ID in the game database is returned true otherwise false.

Parameters

playerID	Name of the Game's Player to be searched.

Returns

Whether there exists a Game with the given Player ID in the game database.

Definition at line 105 of file GameRepository.java.

```
                                                          {
        JSONObject all = this.list();

        JSONObject current;
        for (String key : all.keySet()) {
            current = all.getJSONObject(key);
            if (current.getString("player1_id").equals(playerID) ||
    current.getString("player2_id").equals(playerID)
                    || current.getString("creator_id").equals(playerID))
                return true;
        }

        return false;
    }
```

6.46.3.6 listGames()

```
ArrayList<String> repository.GameRepository.listGames (
              String playerID )
```

List all Games by Player ID of the game database.

Precondition

The Game repository JSON files exists.

Postcondition

An ArrayList containing the Game names by Player ID of the game database is returned.

Parameters

playerID	Player ID of a player in the Games to be getted.

Returns

ArrayList of the Game names by Player ID of the game database.

Definition at line 126 of file GameRepository.java.

```
                                                         {
        ArrayList<String> list = new ArrayList<String>();
        JSONObject all = this.list();

        JSONObject current;
        for (String key : all.keySet()) {
            current = all.getJSONObject(key);
            if (current.getString("player1_id").equals(playerID) ||
    current.getString("player2_id").equals(playerID)
                    || current.getString("creator_id").equals(playerID))
                list.add(key);
        }

        return list;
    }
```

The documentation for this class was generated from the following file:

- GameRepository.java

6.47 repository.GameRepositoryCtrl Class Reference

Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.

Public Member Functions

- GameRepositoryCtrl ()

    Create a GameRepositoryCtrl instance.
- void save (JSONObject game, JSONObject board)

    Save a Game into the game database.
- JSONObject getGame (String name)

    Get the Game by name from the game database or null if it does not exist.
- JSONObject getBoard (String name)

    Get the playing Board of a Game by name from the game database or null if it does not exist.
- Boolean existsGameByConfigurationName (String configurationName)

    Check whether there exists a Game with the given Configuration name in the game database.
- Boolean existsGameByPlayerID (UUID playerID)

    Check whether there exists a Game with the given Player ID in the game database.
- ArrayList< String > listGames (UUID playerID)

    List all Games by Player ID of the game database.

Private Attributes

- GameRepository repository

    GameRepository instance.

6.47.1 Detailed Description

Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.

See also

repository.GameRepository

Definition at line 19 of file GameRepositoryCtrl.java.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 GameRepositoryCtrl()

```
repository.GameRepositoryCtrl.GameRepositoryCtrl ( )
```

Create a GameRepositoryCtrl instance.

Precondition

The Game repository JSON files exists.

Postcondition

A GameRepositoryCtrl instance is created.

Definition at line 34 of file GameRepositoryCtrl.java.

```
                                  {
        this.repository = new GameRepository();
    }
```

6.47.3 Member Function Documentation

6.47.3.1 save()

```
void repository.GameRepositoryCtrl.save (
            JSONObject game,
            JSONObject board )
```

Save a Game into the game database.

Precondition

The Game repository JSON files exists.

Postcondition

The Game and its playing Board are saved into the game database.

Parameters

game	Game to be saved.
board	Playing Board of the Game to be saved.

Definition at line 47 of file GameRepositoryCtrl.java.

```
                                                    {
        this.repository.save(game, board);
    }
```

6.47.3.2 getGame()

```
JSONObject repository.GameRepositoryCtrl.getGame (
            String name )
```

Get the Game by name from the game database or null if it does not exist.

Precondition

The Game repository JSON files exists.

Postcondition

A JSONObject representing the Game by name from the game database is returned or null if it does not exist.

Parameters

name	Name of the Game to be getted.

Returns

JSONObject that represents the Game by name from the game database or null if it does not exist.

Definition at line 58 of file GameRepositoryCtrl.java.

```
                                         {
        return this.repository.getGame(name);
    }
```

6.47.3.3 getBoard()

```
JSONObject repository.GameRepositoryCtrl.getBoard (
            String name )
```

Get the playing Board of a Game by name from the game database or null if it does not exist.

Precondition

The Game repository JSON files exists.

Postcondition

A JSONObject representing the playing Board of a Game by name from the game database is returned or null if it does not exist.

Parameters

name	Name of the playing Board's Game to be getted.

Returns

JSONObject that represents the playing Board of a Game by name from the game database or null if it does not exist.

Definition at line 69 of file GameRepositoryCtrl.java.

```
                                         {
         return this.repository.getBoard(name);
     }
```

6.47.3.4 existsGameByConfigurationName()

```
Boolean repository.GameRepositoryCtrl.existsGameByConfigurationName (
            String configurationName )
```

Check whether there exists a Game with the given Configuration name in the game database.

Precondition

The Game repository JSON files exists.

Postcondition

If there exists a Game with the given Configuration name in the game database is returned true otherwise false.

Parameters

configurationName	Name of the Game's Configuration to be searched.

Returns

Whether there exists a Game with the given Configuration name in the game database.

Definition at line 80 of file GameRepositoryCtrl.java.

```
                                                                   {
         return this.repository.existsGameByConfigurationName(configurationName);
     }
```

6.47.3.5 existsGameByPlayerID()

```
Boolean repository.GameRepositoryCtrl.existsGameByPlayerID (
            UUID playerID )
```

Check whether there exists a Game with the given Player ID in the game database.

Precondition

The Game repository JSON files exists.

Postcondition

If there exists a Game with the given Player ID in the game database is returned true otherwise false.

Parameters

playerID	Name of the Game's Player to be searched.

Returns

Whether there exists a Game with the given Player ID in the game database.

Definition at line 91 of file GameRepositoryCtrl.java.

```
                                                    {
        return this.repository.existsGameByPlayerID(playerID.toString());
    }
```

6.47.3.6 listGames()

```
ArrayList<String> repository.GameRepositoryCtrl.listGames (
              UUID playerID )
```

List all Games by Player ID of the game database.

Precondition

The Game repository JSON files exists.

Postcondition

An ArrayList containing the Game names by Player ID of the game database is returned.

Parameters

playerID	Player ID of a player in the Games to be getted.

Returns

ArrayList of the Game names by Player ID of the game database.

Definition at line 102 of file GameRepositoryCtrl.java.

```
                                                   {
        return this.repository.listGames(playerID.toString());
    }
```

6.47.4 Member Data Documentation

6.47.4.1 repository

```
GameRepository repository.GameRepositoryCtrl.repository  [private]
```

GameRepository instance.

Definition at line 25 of file GameRepositoryCtrl.java.

The documentation for this class was generated from the following file:

- GameRepositoryCtrl.java

6.48 view.GamesCreateView Class Reference

Public Member Functions

- GamesCreateView ()

  Class creator.
- void initialize ()

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void createGame () throws IOException

  Event method which is executed when the createGame button is clicked.
- void userChooser1 ()

  Method which is executed when Player1User RadioButton is selected.
- void userChooser2 ()

  Method which is executed when Player2User RadioButton is selected.
- void botChooser1 ()

  Method which is executed when Player1Bot RadioButton is selected.
- void botChooser2 ()

  Method which is executed when Player1Bot RadioButton is selected.
- void createGameConfirm () throws IOException

  Event method which is executed when the create button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text createGame

  Game create button text.
- Rectangle createGameButton

  Game create button.
- ChoiceBox configChooser

  Configuration choiceBox.
- RadioButton pl1User

  Player 1 user selector.
- RadioButton pl1Bot

  Player 1 bot selector.
- ChoiceBox userChooser1

  Configuration choiceBox.
- ChoiceBox botChooser1

  Configuration choiceBox.
- RadioButton pl2User

  Player 2 user selector.
- RadioButton pl2Bot

  Player 2 bot selector.
- ChoiceBox userChooser2

  Configuration choiceBox.
- ChoiceBox botChooser2

  Configuration choiceBox.
- Text createGameConfirm

  Game create confirm button text.
- Rectangle createGameConfirmButton

  Game create confirm button.
- Label createGameResult

  Exception output message label.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.
- Map< String, UUID > userMap
- Map< String, UUID > botMap

6.48.1 Detailed Description

This class represents the scene controller of creation function of a game.

Done by Arnau Pujantell

Definition at line 30 of file GamesCreateView.java.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 GamesCreateView()

```
view.GamesCreateView.GamesCreateView ( )
```

Class creator.

Definition at line 37 of file GamesCreateView.java.

```
                                {
    }
```

6.48.3 Member Function Documentation

6.48.3.1 initialize()

```
void view.GamesCreateView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. All configuration names are inserted in the Configuration choiceBox.

Definition at line 168 of file GamesCreateView.java.

```
                                {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        userMap = new HashMap<String, UUID>();
        botMap = new HashMap<String, UUID>();

        ArrayList<Pair<String, UUID» userList = ViewCtrl.domainCtrl.listUsers();
        for(Pair<String, UUID> user : userList) {
            userChooser1.getItems().add(user.first);
            userChooser2.getItems().add(user.first);
            userMap.put(user.first, user.second);
        }

        ArrayList<Pair<String, UUID» botList = ViewCtrl.domainCtrl.listBots();
        for(Pair<String, UUID> bot : botList) {
            botChooser1.getItems().add(bot.first);
            botChooser2.getItems().add(bot.first);
            botMap.put(bot.first, bot.second);
        }

        ArrayList<String> configList = ViewCtrl.domainCtrl.listConfigurations().first;
        for(String configName : configList) configChooser.getItems().add(configName);

        userChooser1();
        botChooser2();
    }
```

6.48.3.2 user()

```
void view.GamesCreateView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 199 of file GamesCreateView.java.

```
                                              {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.48.3.3 bots()

```
void view.GamesCreateView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 208 of file GamesCreateView.java.

```
                                              {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.48.3.4 config()

```
void view.GamesCreateView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 217 of file GamesCreateView.java.

```
                                                {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.48.3.5 ranking()

```
void view.GamesCreateView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 226 of file GamesCreateView.java.

```
                                            {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.48.3.6 play()

```
void view.GamesCreateView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 235 of file GamesCreateView.java.

```
                                         {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.48.3.7 createGame()

```
void view.GamesCreateView.createGame ( ) throws IOException
```

Event method which is executed when the createGame button is clicked.

Precondition

True

Postcondition

Scene changes to GameCreateView.

Definition at line 244 of file GamesCreateView.java.

```
                                               {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.48.3.8 userChooser1()

```
void view.GamesCreateView.userChooser1 ( )
```

Method which is executed when Player1User RadioButton is selected.

Precondition

True

Postcondition

All users are inserted in userChooser1.

Definition at line 253 of file GamesCreateView.java.

```
                                {
        userChooser1.setDisable(false);
        botChooser1.setDisable(true);
    }
```

6.48.3.9 userChooser2()

```
void view.GamesCreateView.userChooser2 ( )
```

Method which is executed when Player2User RadioButton is selected.

Precondition

True

Postcondition

All users are inserted in userChooser2.

Definition at line 263 of file GamesCreateView.java.

```
                                {
        userChooser2.setDisable(false);
        botChooser2.setDisable(true);
    }
```

6.48.3.10 botChooser1()

```
void view.GamesCreateView.botChooser1 ( )
```

Method which is executed when Player1Bot RadioButton is selected.

Precondition

True

Postcondition

All users are inserted in botChooser1.

Definition at line 273 of file GamesCreateView.java.

```
                               {
        userChooser1.setDisable(true);
        botChooser1.setDisable(false);
    }
```

6.48.3.11 botChooser2()

```
void view.GamesCreateView.botChooser2 ( )
```

Method which is executed when Player1Bot RadioButton is selected.

Precondition

True

Postcondition

All users are inserted in botChooser2.

Definition at line 283 of file GamesCreateView.java.

```
                                {
        userChooser2.setDisable(true);
        botChooser2.setDisable(false);
    }
```

6.48.3.12 createGameConfirm()

```
void view.GamesCreateView.createGameConfirm ( ) throws IOException
```

Event method which is executed when the create button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the new Game is created.

Definition at line 293 of file GamesCreateView.java.

```
                                                       {
        String chosenConfig = (String) configChooser.getValue();
        String chosenUser1 = (String) userChooser1.getValue();
        String chosenBot1 = (String) botChooser1.getValue();
        String chosenUser2 = (String) userChooser2.getValue();
        String chosenBot2 = (String) botChooser2.getValue();

        if (chosenConfig != null) {
            UUID player1ID = null;
            UUID player2ID = null;

            if(pl1User.isSelected() && chosenUser1 != null) player1ID = userMap.get(chosenUser1);
            if(pl1Bot.isSelected() && chosenBot1 != null) player1ID = botMap.get(chosenBot1);
            if(pl2User.isSelected() && chosenUser2 != null) player2ID = userMap.get(chosenUser2);
            if(pl2Bot.isSelected() && chosenBot2 != null) player2ID = botMap.get(chosenBot2);

            if (player1ID != null && player2ID != null) {
                Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createGame(player1ID, player2ID,
       chosenConfig);
                if (result.second != null) {
                    switch (result.second) {
                        case "ERR_INVALID_PLAYERS":
                            createGameResult.setText("The player selection is invalid!");
                            break;
                        case "ERR_INVALID_CONFIGURATION":
                            createGameResult.setText("The selected configuration is invalid!");
                            break;
                        case "ERR_INVALID_BOARD":
                            createGameResult.setText("The playing board is invalid!");
                            break;
                        default:
                            createGameResult.setText("Something went wrong, try again!");
                            break;
                    }
                }
                else {
                    userChooser1.getSelectionModel().select(null);
                    botChooser1.getSelectionModel().select(null);
                    userChooser2.getSelectionModel().select(null);
                    botChooser2.getSelectionModel().select(null);
                    createGameResult.setText("Success!");
                }
            }
        }
    }
```

6.48.3.13 logOut()

`void view.GamesCreateView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 343 of file GamesCreateView.java.

```
343                                            {
344         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
      ButtonType.YES, ButtonType.NO);
345         confirm.showAndWait();
346 
347         if (confirm.getResult() == ButtonType.YES) {
348             ViewCtrl.domainCtrl.logout();
349             ViewCtrl.changeScene("template/LogInView.fxml");
350         }
351     }
```

6.48.4 Member Data Documentation

6.48.4.1 user

`Text view.GamesCreateView.user [private]`

Menu User tab.

Definition at line 46 of file GamesCreateView.java.

6.48.4.2 bots

```
Text view.GamesCreateView.bots [private]
```

Menu Bots tab.

Definition at line 51 of file GamesCreateView.java.

6.48.4.3 config

```
Text view.GamesCreateView.config [private]
```

Menu Configuration tab.

Definition at line 56 of file GamesCreateView.java.

6.48.4.4 games

```
Text view.GamesCreateView.games [private]
```

Menu Games tab.

Definition at line 61 of file GamesCreateView.java.

6.48.4.5 ranking

```
Text view.GamesCreateView.ranking [private]
```

Menu Ranking tab.

Definition at line 66 of file GamesCreateView.java.

6.48.4.6 play

```
Text view.GamesCreateView.play [private]
```

Menu Play tab.

Definition at line 71 of file GamesCreateView.java.

6.48.4.7 createGame

```
Text view.GamesCreateView.createGame  [private]
```

Game create button text.

Definition at line 76 of file GamesCreateView.java.

6.48.4.8 createGameButton

```
Rectangle view.GamesCreateView.createGameButton  [private]
```

Game create button.

Definition at line 81 of file GamesCreateView.java.

6.48.4.9 configChooser

```
ChoiceBox view.GamesCreateView.configChooser  [private]
```

Configuration choiceBox.

Definition at line 86 of file GamesCreateView.java.

6.48.4.10 pl1User

```
RadioButton view.GamesCreateView.pl1User  [private]
```

Player 1 user selector.

Definition at line 91 of file GamesCreateView.java.

6.48.4.11 pl1Bot

```
RadioButton view.GamesCreateView.pl1Bot  [private]
```

Player 1 bot selector.

Definition at line 96 of file GamesCreateView.java.

6.48.4.12 userChooser1

```
ChoiceBox view.GamesCreateView.userChooser1  [private]
```

Configuration choiceBox.

Definition at line 101 of file GamesCreateView.java.

6.48.4.13 botChooser1

```
ChoiceBox view.GamesCreateView.botChooser1  [private]
```

Configuration choiceBox.

Definition at line 106 of file GamesCreateView.java.

6.48.4.14 pl2User

```
RadioButton view.GamesCreateView.pl2User  [private]
```

Player 2 user selector.

Definition at line 111 of file GamesCreateView.java.

6.48.4.15 pl2Bot

```
RadioButton view.GamesCreateView.pl2Bot  [private]
```

Player 2 bot selector.

Definition at line 116 of file GamesCreateView.java.

6.48.4.16 userChooser2

```
ChoiceBox view.GamesCreateView.userChooser2  [private]
```

Configuration choiceBox.

Definition at line 121 of file GamesCreateView.java.

6.48.4.17 botChooser2

```
ChoiceBox view.GamesCreateView.botChooser2  [private]
```

Configuration choiceBox.

Definition at line 126 of file GamesCreateView.java.

6.48.4.18 createGameConfirm

```
Text view.GamesCreateView.createGameConfirm  [private]
```

Game create confirm button text.

Definition at line 131 of file GamesCreateView.java.

6.48.4.19 createGameConfirmButton

```
Rectangle view.GamesCreateView.createGameConfirmButton  [private]
```

Game create confirm button.

Definition at line 136 of file GamesCreateView.java.

6.48.4.20 createGameResult

```
Label view.GamesCreateView.createGameResult  [private]
```

Exception output message label.

Definition at line 141 of file GamesCreateView.java.

6.48.4.21 currentUserName

```
Label view.GamesCreateView.currentUserName  [private]
```

Current user name.

Definition at line 146 of file GamesCreateView.java.

6.48.4.22 logOut

```
Text view.GamesCreateView.logOut  [private]
```

LogOut button.

Definition at line 151 of file GamesCreateView.java.

6.48.4.23 userMap

```
Map<String, UUID> view.GamesCreateView.userMap  [private]
```

Map of users.

Definition at line 155 of file GamesCreateView.java.

6.48.4.24 botMap

```
Map<String, UUID> view.GamesCreateView.botMap  [private]
```

Map of bots.

Definition at line 159 of file GamesCreateView.java.

The documentation for this class was generated from the following file:

- GamesCreateView.java

6.49 domain.Game.GameState Enum Reference

State of a Game. Whether it has not started, it is currently being played or it has already finished.

Public Attributes

- NOT_STARTED
- IN_PROGRESS
- FINISHED

6.49.1 Detailed Description

State of a Game. Whether it has not started, it is currently being played or it has already finished.

Definition at line 32 of file Game.java.

6.49.2 Member Data Documentation

6.49.2.1 NOT_STARTED

```
domain.Game.GameState.NOT_STARTED
```

Definition at line 33 of file Game.java.

6.49.2.2 IN_PROGRESS

```
domain.Game.GameState.IN_PROGRESS
```

Definition at line 33 of file Game.java.

6.49.2.3 FINISHED

```
domain.Game.GameState.FINISHED
```

Definition at line 34 of file Game.java.

The documentation for this enum was generated from the following file:

- Game.java

6.50 view.GamesView Class Reference

Public Member Functions

- GamesView ()
  Class creator.
- void initialize () throws Exception
  Initialize method which is executed when the scene is shown.
- void user () throws IOException
  Event method which is executed when the User tab is clicked.
- void bots () throws IOException
  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException
  Event method which is executed when the Configuration tab is clicked.
- void ranking () throws IOException
  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException
  Event method which is executed when the Play tab is clicked.
- void createGame () throws IOException
  Event method which is executed when the createGame button is clicked.
- void logOut () throws IOException
  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

    Menu User tab.
- Text bots

    Menu Bots tab.
- Text config

    Menu Configuration tab.
- Text games

    Menu Games tab.
- Text ranking

    Menu Ranking tab.
- Text play

    Menu Play tab.
- Text createGame

    Game create button text.
- Rectangle createGameButton

    Game create button.
- Label currentUserName

    Current user name.
- Text logOut

    LogOut button.

6.50.1 Detailed Description

This class represents the scene controller of the Game Menu .

Done by Arnau Pujantell

Definition at line 22 of file GamesView.java.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 GamesView()

```
view.GamesView.GamesView ( )
```

Class creator.

Definition at line 29 of file GamesView.java.

```
29                            {
30      }
```

6.50.3 Member Function Documentation

6.50.3.1 initialize()

`void view.GamesView.initialize ( ) throws Exception`

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 91 of file GamesView.java.

```
                                        {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
    }
```

6.50.3.2 user()

`void view.GamesView.user ( ) throws IOException`

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 100 of file GamesView.java.

```
                                        {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.50.3.3 bots()

`void view.GamesView.bots ( ) throws IOException`

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 109 of file GamesView.java.

```
                                        {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.50.3.4 config()

`void view.GamesView.config ( ) throws IOException`

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 118 of file GamesView.java.

```
                                {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.50.3.5 ranking()

`void view.GamesView.ranking ( ) throws IOException`

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 127 of file GamesView.java.

```
                                 {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.50.3.6 play()

`void view.GamesView.play ( ) throws IOException`

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 136 of file GamesView.java.

```
                              {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.50.3.7 createGame()

`void view.GamesView.createGame ( ) throws IOException`

Event method which is executed when the createGame button is clicked.

Precondition

True

Postcondition

Scene changes to GameCreateView.

Definition at line 145 of file GamesView.java.

```
                                            {
        ViewCtrl.changeScene("template/GamesCreateView.fxml");
    }
```

6.50.3.8 logOut()

`void view.GamesView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 154 of file GamesView.java.

```
                                        {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
   ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.50.4 Member Data Documentation

6.50.4.1 user

```
Text view.GamesView.user  [private]
```

Menu User tab.

Definition at line 38 of file GamesView.java.

6.50.4.2 bots

```
Text view.GamesView.bots  [private]
```

Menu Bots tab.

Definition at line 43 of file GamesView.java.

6.50.4.3 config

```
Text view.GamesView.config  [private]
```

Menu Configuration tab.

Definition at line 48 of file GamesView.java.

6.50.4.4 games

```
Text view.GamesView.games  [private]
```

Menu Games tab.

Definition at line 53 of file GamesView.java.

6.50.4.5 ranking

```
Text view.GamesView.ranking  [private]
```

Menu Ranking tab.

Definition at line 58 of file GamesView.java.

6.50.4.6 play

```
Text view.GamesView.play  [private]
```

Menu Play tab.

Definition at line 63 of file GamesView.java.

6.50.4.7 createGame

```
Text view.GamesView.createGame  [private]
```

Game create button text.

Definition at line 68 of file GamesView.java.

6.50.4.8 createGameButton

```
Rectangle view.GamesView.createGameButton  [private]
```

Game create button.

Definition at line 73 of file GamesView.java.

6.50.4.9 currentUserName

```
Label view.GamesView.currentUserName  [private]
```

Current user name.

Definition at line 78 of file GamesView.java.

6.50.4.10 logOut

```
Text view.GamesView.logOut  [private]
```

LogOut button.

Definition at line 83 of file GamesView.java.

The documentation for this class was generated from the following file:

- GamesView.java

6.51 domain.HardDifficulty Class Reference

Implements the Monte Carlo Tree Search algorithm to get the next best possible position for a given player. By Roger Mollon.

Classes

- class TreeNode

Public Member Functions

- HardDifficulty (Integer difficulty, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally, PieceType pieceType)

  Create a HardDifficulty instance.
- Pair< Integer, Integer > place (PieceType[ ][ ] playingBoard)

  Get the next best possible position for the implicit player.

Static Private Attributes

- static Random random = new Random()

  Random number used in the UCT (Upper Confidence bounds applied to Trees) formula to break ties when choosing a path.
- static double epsilon = 1e-6

  Small number used to prevent divisions by zero.

Additional Inherited Members

6.51.1 Detailed Description

Implements the Monte Carlo Tree Search algorithm to get the next best possible position for a given player. By Roger Mollon.

Definition at line 21 of file HardDifficulty.java.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 HardDifficulty()

```
domain.HardDifficulty.HardDifficulty (
            Integer difficulty,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally,
            PieceType pieceType )
```

Create a HardDifficulty instance.

Precondition

The given difficulty is a positive number. The given rules are not all false.

Postcondition

A HardDifficulty instance is created and its implicits difficulty, canEatHorizontally, canEatVertically, canEat↵ Diagonally and pieceType attributes are setted. The implicit maxDepth attribute is setted to 1000 times the entered difficulty.

Parameters

difficulty	Difficulty for the Monte Carlo Tree Search algorithm.
canEatHorizontally	Whether the pieces of the current Game can be eaten horizontally.
canEatVertically	Whether the pieces of the current Game can be eaten vertically.
canEatDiagonally	Whether the pieces of the current Game can be eaten diagonally.
pieceType	Player that wants to be maximized.

Definition at line 46 of file HardDifficulty.java.

```
                                                  {
        super(difficulty, canEatHorizontally, canEatVertically, canEatDiagonally, pieceType);
        this.maxDepth = difficulty * 1000;
    }
```

6.51.3 Member Function Documentation

6.51.3.1 place()

```
Pair<Integer, Integer> domain.HardDifficulty.place (
            PieceType playingBoard[ ][ ] )
```

Get the next best possible position for the implicit player.

Precondition

True

Postcondition

It is returned the next best possible position for the implicit player, using the Monte Carlo Tree Search algorithm with the implicit maximum depth, or null if there isn't any.

Parameters

playingBoard	Current playing Board.

Returns

The next best possible position for the implicit player or null if there isn't any.

Reimplemented from domain.Difficulty.

Definition at line 64 of file HardDifficulty.java.
```
                                                                    {
          Pair<Integer, Integer> bestPosition = null;

          Board initialBoard = new Board(playingBoard);

          TreeNode rootGame = new TreeNode(
              this.pieceType, this.pieceType, initialBoard, this.canEatHorizontally, this.canEatVertically,
      this.canEatDiagonally, null);

          for (int i = 0; i < this.maxDepth; ++i) rootGame.play();

          ArrayList<TreeNode> playedGames = rootGame.getChildren();

          double maxWinRatio = Double.NEGATIVE_INFINITY;
          for (TreeNode game: playedGames) {
              if (game.getWinRatio() > maxWinRatio) {
                  maxWinRatio = game.getWinRatio();
                  bestPosition = game.getSelectedPosition();
              }
          }

          return bestPosition;
      }
```

6.51.4 Member Data Documentation

6.51.4.1 random

`Random domain.HardDifficulty.random = new Random()  [static], [private]`

Random number used in the UCT (Upper Confidence bounds applied to Trees) formula to break ties when choosing a path.

Definition at line 27 of file HardDifficulty.java.

6.51.4.2 epsilon

`double domain.HardDifficulty.epsilon = 1e-6  [static], [private]`

Small number used to prevent divisions by zero.

Definition at line 31 of file HardDifficulty.java.

The documentation for this class was generated from the following file:

- HardDifficulty.java

6.52 cmd.driver.hardDifficulty Class Reference

HardDifficulty driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)

  HardDifficulty driver main function. Creates an instance of the HardDifficulty driver and starts it.

6.52.1 Detailed Description

HardDifficulty driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file hardDifficulty.java.

6.52.2 Member Function Documentation

6.52.2.1 main()

```
static void cmd.driver.hardDifficulty.main (
             String[ ] args )  [static]
```

HardDifficulty driver main function. Creates an instance of the HardDifficulty driver and starts it.

Precondition

True.

Postcondition

The HardDifficulty driver has started.

Definition at line 22 of file hardDifficulty.java.

```
                                              {
           new HardDifficultyDriver().start();
       }
```

The documentation for this class was generated from the following file:

- hardDifficulty.java

6.53 test.driver.HardDifficultyDriver Class Reference

Implements the different options for the HardDifficulty driver application. By Roger Mollon.

Public Member Functions

- HardDifficultyDriver ()
- void start ()

Public Attributes

- HardDifficulty currentHardDifficulty
- Board currentBoard
- String nameCurrentBoard
- FixtureRepository fixtureRepository

Private Member Functions

- void mainMenu ()
- void create ()
- void getDifficulty ()
- void getCanEatHorizontally ()
- void getCanEatVertically ()
- void getCanEatDiagonally ()
- void getPieceType ()
- void getMaxDepth ()
- void setMaxDepth ()
- void loadBoard ()
- void printCurrentBoard ()
- void getNextBestPosition ()
- Pair< String, Board > listBoardFixtures ()
- void printBoard (Board board)
- ArrayList< String > transcribeToCharacters (Board board)

Additional Inherited Members

6.53.1 Detailed Description

Implements the different options for the HardDifficulty driver application. By Roger Mollon.

Definition at line 21 of file HardDifficultyDriver.java.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 HardDifficultyDriver()

```
test.driver.HardDifficultyDriver.HardDifficultyDriver ( )
```

Definition at line 33 of file HardDifficultyDriver.java.

```
                                   {
        this.currentHardDifficulty = null;
        this.fixtureRepository = new FixtureRepository();
    }
```

6.53.3 Member Function Documentation

6.53.3.1 start()

```
void test.driver.HardDifficultyDriver.start ( )
```

Definition at line 40 of file HardDifficultyDriver.java.

```
                       {
        while (true) {
            this.mainMenu();
        }
    }
```

6.53.3.2 mainMenu()

```
void test.driver.HardDifficultyDriver.mainMenu ( )  [private]
```

Definition at line 46 of file HardDifficultyDriver.java.

```
                           {
        String title = null;
        if (this.currentHardDifficulty != null)
            title = String.format("Current maximum depth: %s\n",
    this.currentHardDifficulty.getMaxDepth());
        if (this.currentBoard != null)
            title += String.format("Current Board: %s\n", this.nameCurrentBoard);

        switch (Driver.menu(title, "HardDifficulty (Montecarlo) Driver",
                new Pair<String, String>("1", "Create HardDifficulty"),
                new Pair<String, String>("2", "Get difficulty"),
                new Pair<String, String>("3", "Get canEatHorizontally"),
                new Pair<String, String>("4", "Get canEatVertically"),
                new Pair<String, String>("5", "Get canEatDiagonally"),
                new Pair<String, String>("6", "Get pieceType"),
                new Pair<String, String>("7", "Get maxDepth"),
                new Pair<String, String>("8", "Set maxDepth"),
                new Pair<String, String>("9", "Load Board"),
                new Pair<String, String>("10", "Print Current Board"),
                new Pair<String, String>("11", "Get next best position"))) {
        case "1":
            this.create();
            break;
        case "2":
            this.getDifficulty();
            break;
        case "3":
            this.getCanEatHorizontally();
            break;
        case "4":
            this.getCanEatVertically();
            break;
        case "5":
            this.getCanEatDiagonally();
            break;
        case "6":
            this.getPieceType();
            break;
        case "7":
            this.getMaxDepth();
            break;
        case "8":
            this.setMaxDepth();
            break;
        case "9":
            this.loadBoard();
            break;
        case "10":
            this.printCurrentBoard();
            break;
        case "11":
            this.getNextBestPosition();
            break;
        }
        Driver.pause();
    }
```

6.53.3.3 create()

```
void test.driver.HardDifficultyDriver.create ( )  [private]
```

Definition at line 102 of file HardDifficultyDriver.java.

```
                        {
        System.out.println(
                "Take into account that the default maximum depth is 1000 times the entered
difficulty.\nMontecarlo with higher number of games to simulate requires more time to execute. A
value of 10 for the difficulty is reasonable.\n");

        Integer difficulty = Driver.inputInt("Difficulty (positive)?");
        Boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
        Boolean canEatVertically = Driver.inputBool("Can eat vertically?");
        Boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
        PieceType pieceType = null;

        switch (Driver.menu(null, "Select Bot pieces",
                new Pair<String, String>("1", "PLAYER 1 pieces"),
                new Pair<String, String>("2", "PLAYER 2 pieces"))) {
        case "1":
            pieceType = PieceType.PLAYER1;
            break;
        case "2":
            pieceType = PieceType.PLAYER2;
            break;
        }

        this.currentHardDifficulty = new HardDifficulty(difficulty, canEatHorizontally,
canEatVertically,
                canEatDiagonally, pieceType);

        System.out.println(String.format("HardDifficulty with a maximum depth of %s created
successfully!",
                this.currentHardDifficulty.getMaxDepth()));
    }
```

6.53.3.4 getDifficulty()

```
void test.driver.HardDifficultyDriver.getDifficulty ( )  [private]
```

Definition at line 130 of file HardDifficultyDriver.java.

```
                               {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(
                String.format("HardDifficulty's difficulty is: %s",
this.currentHardDifficulty.getDifficulty()));
    }
```

6.53.3.5 getCanEatHorizontally()

```
void test.driver.HardDifficultyDriver.getCanEatHorizontally ( )  [private]
```

Definition at line 140 of file HardDifficultyDriver.java.

```
                                       {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(String.format("HardDifficulty's canEatHorizontally is: %s",
                this.currentHardDifficulty.getCanEatHorizontally()));
    }
```

6.53.3.6 getCanEatVertically()

```
void test.driver.HardDifficultyDriver.getCanEatVertically ( )  [private]
```

Definition at line 150 of file HardDifficultyDriver.java.

```
                                    {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(String.format("HardDifficulty's canEatVertically is: %s",
                this.currentHardDifficulty.getCanEatVertically()));
    }
```

6.53.3.7 getCanEatDiagonally()

```
void test.driver.HardDifficultyDriver.getCanEatDiagonally ( )  [private]
```

Definition at line 160 of file HardDifficultyDriver.java.

```
                                    {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(String.format("HardDifficulty's canEatDiagonally is: %s",
                this.currentHardDifficulty.getCanEatDiagonally()));
    }
```

6.53.3.8 getPieceType()

```
void test.driver.HardDifficultyDriver.getPieceType ( )  [private]
```

Definition at line 170 of file HardDifficultyDriver.java.

```
                             {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(
                String.format("HardDifficulty's pieceType is: %s",
      this.currentHardDifficulty.getPieceType()));
    }
```

6.53.3.9 getMaxDepth()

```
void test.driver.HardDifficultyDriver.getMaxDepth ( )  [private]
```

Definition at line 180 of file HardDifficultyDriver.java.

```
                            {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(
                String.format("HardDifficulty's maxDepth is: %s",
      this.currentHardDifficulty.getMaxDepth()));
    }
```

6.53.3.10 setMaxDepth()

```
void test.driver.HardDifficultyDriver.setMaxDepth ( ) [private]
```

Definition at line 190 of file HardDifficultyDriver.java.

```
                             {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        System.out.println(
                "Take into account that Montecarlo algorithm with higher number of games to simulate
       requires more time to execute. A value of 10000 is reasonable.\n");

        this.currentHardDifficulty.setMaxDepth(Driver.inputInt("Maximum depth (positive)?"));
        System.out.println("HardDifficulty's maxDepth changed successfully!");
    }
```

6.53.3.11 loadBoard()

```
void test.driver.HardDifficultyDriver.loadBoard ( ) [private]
```

Definition at line 203 of file HardDifficultyDriver.java.

```
                           {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        Pair<String, Board> selectedBoard = this.listBoardFixtures();

        this.nameCurrentBoard = selectedBoard.first;
        this.currentBoard = selectedBoard.second;

        System.out.println(String.format("Board %s loaded successfully!\n", this.nameCurrentBoard));
        this.printBoard(this.currentBoard);
    }
```

6.53.3.12 printCurrentBoard()

```
void test.driver.HardDifficultyDriver.printCurrentBoard ( ) [private]
```

Definition at line 218 of file HardDifficultyDriver.java.

```
                                   {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        if (this.currentBoard == null) {
            System.out.println("No current Board!");
            return;
        }

        System.out.println(String.format("Board %s printed successfully!\n", this.nameCurrentBoard));
        this.printBoard(this.currentBoard);
    }
```

6.53.3.13 getNextBestPosition()

```
void test.driver.HardDifficultyDriver.getNextBestPosition ( ) [private]
```

Definition at line 233 of file HardDifficultyDriver.java.

```
                                          {
        if (this.currentHardDifficulty == null) {
            System.out.println("No current HardDifficulty!");
            return;
        }

        if (this.currentBoard == null) {
            System.out.println("No current Board!");
            return;
        }

        System.out.println("Take into account that the state of the current Board won't be globally
modified.\n");

        this.printBoard(this.currentBoard);

        long startTime = System.currentTimeMillis();
        Pair<Integer, Integer> nextBestPosition =
this.currentHardDifficulty.place(this.currentBoard.getBoard());
        long durationTime = System.currentTimeMillis() - startTime;

        Board tempBoard = new Board(this.currentBoard.getBoard());

        if (nextBestPosition != null) {
            tempBoard.placePiece(nextBestPosition, this.currentHardDifficulty.getPieceType(),
                    this.currentHardDifficulty.getCanEatHorizontally(),
                    this.currentHardDifficulty.getCanEatVertically(),
                    this.currentHardDifficulty.getCanEatDiagonally());
            System.out.println(
                    String.format("The best position calculated in %s ms is %s\n", durationTime,
nextBestPosition));
            System.out.println("The addition of the piece would look like this:\n");
            this.printBoard(tempBoard);
        } else {
            System.out.println("There isn't any possible position left to place a piece on.");
        }
    }
```

6.53.3.14 listBoardFixtures()

```
Pair<String, Board> test.driver.HardDifficultyDriver.listBoardFixtures ( ) [private]
```

Definition at line 268 of file HardDifficultyDriver.java.

```
                                                     {
        Integer selectedBoard = -1;
        ArrayList<String> listBoards = this.fixtureRepository.listFiles();

        while (selectedBoard < 0 || selectedBoard >= listBoards.size()) {
            Driver.clear();
            System.out.println("==== Available Boards ====\n");

            for (Integer i = 0; i < listBoards.size(); ++i)
                System.out.println(String.format("[%d]\t%s", i, listBoards.get(i)));
            System.out.println("");

            selectedBoard = Driver.inputInt("What Board would you like to load?");
        }

        Driver.clear();

        return new Pair<String, Board>(listBoards.get(selectedBoard),
                new Board(this.fixtureRepository.boardFileToJSON(listBoards.get(selectedBoard))));
    }
```

6.53.3.15 printBoard()

```
void test.driver.HardDifficultyDriver.printBoard (
            Board board )  [private]
```

Definition at line 289 of file HardDifficultyDriver.java.

```
                                      {
     ArrayList<String> boardCodified = this.transcribeToCharacters(board);
     System.out.println("       0  1  2  3  4  5  6  7");
     System.out.println("    -------------------------");

     for (Integer i = 0; i < 8; ++i) {
         String row = boardCodified.get(i);
         System.out.println("  " + i + " |  " + row.charAt(0) + "  " + row.charAt(1) + "  " +
row.charAt(2) + "  "
                 + row.charAt(3) + "  " + row.charAt(4) + "  " + row.charAt(5) + "  " + row.charAt(6)
+ "  "
                 + row.charAt(7) + "  ");
     }
     System.out.println("\n");
 }
```

6.53.3.16 transcribeToCharacters()

```
ArrayList<String> test.driver.HardDifficultyDriver.transcribeToCharacters (
            Board board )  [private]
```

Definition at line 303 of file HardDifficultyDriver.java.

```
                                                               {
     ArrayList<String> boardCodified = new ArrayList<String>(8);
     String operational = "";
     PieceType[][] current = board.getBoard();

     for (int i = 0; i < 8; ++i) {
         operational = "";
         for (int j = 0; j < 8; ++j) {
             if (current[i][j] == PieceType.PLAYER1)
                 operational = operational + "B";
             if (current[i][j] == PieceType.PLAYER2)
                 operational = operational + "N";
             if (current[i][j] == null)
                 operational = operational + "?";

         }
         boardCodified.add(operational);
     }

     return boardCodified;
 }
```

6.53.4 Member Data Documentation

6.53.4.1 currentHardDifficulty

```
HardDifficulty test.driver.HardDifficultyDriver.currentHardDifficulty
```

Definition at line 24 of file HardDifficultyDriver.java.

6.53.4.2 currentBoard

```
Board test.driver.HardDifficultyDriver.currentBoard
```

Definition at line 26 of file HardDifficultyDriver.java.

6.53.4.3 nameCurrentBoard

```
String test.driver.HardDifficultyDriver.nameCurrentBoard
```

Definition at line 27 of file HardDifficultyDriver.java.

6.53.4.4 fixtureRepository

```
FixtureRepository test.driver.HardDifficultyDriver.fixtureRepository
```

Definition at line 29 of file HardDifficultyDriver.java.

The documentation for this class was generated from the following file:

- HardDifficultyDriver.java

6.54 domain.Exceptions.IncorrectCredentialsException Class Reference

Wrong password or name. By Alex Rodriguez.

Public Member Functions

- IncorrectCredentialsException ()

6.54.1 Detailed Description

Wrong password or name. By Alex Rodriguez.

Definition at line 85 of file Exceptions.java.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 IncorrectCredentialsException()

`domain.Exceptions.IncorrectCredentialsException.IncorrectCredentialsException ( )`

Definition at line 86 of file Exceptions.java.

```
                                            {
            super("ERR_INCORRECT_CREDENTIALS");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.55 domain.Exceptions.InexistingConfigurationException Class Reference

There isn't any configuration with the entered name. By Alex Rodriguez.

Public Member Functions

- InexistingConfigurationException ()

6.55.1 Detailed Description

There isn't any configuration with the entered name. By Alex Rodriguez.

Definition at line 74 of file Exceptions.java.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 InexistingConfigurationException()

`domain.Exceptions.InexistingConfigurationException.InexistingConfigurationException ( )`

Definition at line 75 of file Exceptions.java.

```
                                               {
            super("ERR_INEXISTING_CONFIGURATION");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.56 domain.Exceptions.InexistingPlayerException Class Reference

There isn't any player with the entered name. By Alex Rodriguez.

Public Member Functions

- InexistingPlayerException ()

6.56.1 Detailed Description

There isn't any player with the entered name. By Alex Rodriguez.

Definition at line 63 of file Exceptions.java.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 InexistingPlayerException()

```
domain.Exceptions.InexistingPlayerException.InexistingPlayerException ( )
```

Definition at line 64 of file Exceptions.java.

```
                                        {
            super("ERR_INEXISTING_PLAYER");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.57 view.InitialBoardView Class Reference

Public Member Functions

- InitialBoardView ()

    Class creator.
- void initialize ()

    Initialize method which is executed when the scene is shown.
- void transform (MouseEvent mouseEvent)

    Event method which is executed when a piece is clicked.
- void save () throws IOException

    Event method which is executed when the save button is clicked.

Private Member Functions

- void render ()

    Method executed everytime there is a change in the board.
- void drawPiece (Pair< Integer, Integer > pos, char pieceType)

    Painting method executed everytime there is a change in the board.
- Pair< Integer, Integer > getClickedPos (MouseEvent mouseEvent)

    Painting method executed everytime a player clicks on the board.
- Circle getCircle (Pair< Integer, Integer > pos)

    Method executed everytime there is a change in the board.

Private Attributes

- Circle f1c1

    Piece located in (1, 1).
- Circle f1c2

    Piece located in (1, 2).
- Circle f1c3

    Piece located in (1, 3).
- Circle f1c4

    Piece located in (1, 4).
- Circle f1c5

    Piece located in (1, 5).
- Circle f1c6

    Piece located in (1, 6).
- Circle f1c7

    Piece located in (1, 7).
- Circle f1c8

    Piece located in (1, 8).
- Circle f2c1

    Piece located in (2, 1).
- Circle f2c2

    Piece located in (2, 2).
- Circle f2c3

    Piece located in (2, 3).
- Circle f2c4

    Piece located in (2, 4).
- Circle f2c5

    Piece located in (2, 5).
- Circle f2c6

    Piece located in (2, 6).
- Circle f2c7

    Piece located in (2, 7).
- Circle f2c8

    Piece located in (2, 8).
- Circle f3c1

    Piece located in (3, 1).
- Circle f3c2

    Piece located in (3, 2).
- Circle f3c3

Piece located in (3, 3).

- Circle f3c4

  Piece located in (3, 4).
- Circle f3c5

  Piece located in (3, 5).
- Circle f3c6

  Piece located in (3, 6).
- Circle f3c7

  Piece located in (3, 7).
- Circle f3c8

  Piece located in (3, 8).
- Circle f4c1

  Piece located in (4, 1).
- Circle f4c2

  Piece located in (4, 2).
- Circle f4c3

  Piece located in (4, 3).
- Circle f4c4

  Piece located in (4, 4).
- Circle f4c5

  Piece located in (4, 5).
- Circle f4c6

  Piece located in (4, 6).
- Circle f4c7

  Piece located in (4, 7).
- Circle f4c8

  Piece located in (4, 8).
- Circle f5c1

  Piece located in (5, 1).
- Circle f5c2

  Piece located in (5, 2).
- Circle f5c3

  Piece located in (5, 3).
- Circle f5c4

  Piece located in (5, 4).
- Circle f5c5

  Piece located in (5, 5).
- Circle f5c6

  Piece located in (5, 6).
- Circle f5c7

  Piece located in (5, 7).
- Circle f5c8

  Piece located in (5, 8).
- Circle f6c1

  Piece located in (6, 1).
- Circle f6c2

  Piece located in (6, 2).
- Circle f6c3

  Piece located in (6, 3).
- Circle f6c4

  Piece located in (6, 4).

- Circle f6c5

  Piece located in (6, 5).
- Circle f6c6

  Piece located in (6, 6).
- Circle f6c7

  Piece located in (6, 7).
- Circle f6c8

  Piece located in (6, 8).
- Circle f7c1

  Piece located in (7, 1).
- Circle f7c2

  Piece located in (7, 2).
- Circle f7c3

  Piece located in (7, 3).
- Circle f7c4

  Piece located in (7, 4).
- Circle f7c5

  Piece located in (7, 5).
- Circle f7c6

  Piece located in (7, 6).
- Circle f7c7

  Piece located in (7, 7).
- Circle f7c8

  Piece located in (7, 8).
- Circle f8c1

  Piece located in (8, 1).
- Circle f8c2

  Piece located in (8, 2).
- Circle f8c3

  Piece located in (8, 3).
- Circle f8c4

  Piece located in (8, 4).
- Circle f8c5

  Piece located in (8, 5).
- Circle f8c6

  Piece located in (8, 6).
- Circle f8c7

  Piece located in (8, 7).
- Circle f8c8

  Piece located in (8, 8).
- Text save

  Save board button text.
- Rectangle saveButton

  Save board button.
- RadioButton placeWhitePieces

  White colour pieces selector.
- RadioButton placeBlackPieces

  Black colour pieces selector.
- RadioButton quitPieces

  Remove pieces selector.
- Label saveInitialBoardResult

  Exception message output.
- JSONObject board

  Current board.

6.57.1 Detailed Description

This class represents the scene controller of the initial board view .

By Alex Rodriguez

Definition at line 29 of file InitialBoardView.java.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 InitialBoardView()

```
view.InitialBoardView.InitialBoardView ( )
```

Class creator.

Definition at line 36 of file InitialBoardView.java.

```
                             {
    }
```

6.57.3 Member Function Documentation

6.57.3.1 initialize()

```
void view.InitialBoardView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The board is setted.

Definition at line 403 of file InitialBoardView.java.

```
                             {
        board = ViewCtrl.domainCtrl.viewBoard();
        render();
    }
```

6.57.3.2 transform()

```
void view.InitialBoardView.transform (
            MouseEvent mouseEvent )
```

Event method which is executed when a piece is clicked.

Precondition

True

Postcondition

The piece changes into white, black or is removed.

Definition at line 413 of file InitialBoardView.java.

```
                                              {
        Pair<Integer, Integer> pos = getClickedPos(mouseEvent);
        if (placeWhitePieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos, "PLAYER1");
        else if (placeBlackPieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos,
   "PLAYER2");
        else if (quitPieces.isSelected()) board = ViewCtrl.domainCtrl.removePiece(pos);
        render();
    }
```

6.57.3.3 save()

```
void view.InitialBoardView.save ( ) throws IOException
```

Event method which is executed when the save button is clicked.

Precondition

True

Postcondition

The game is saved and user can close the game.

Definition at line 426 of file InitialBoardView.java.

```
                                       {
        Stage currentWindow = (Stage) save.getScene().getWindow();
        currentWindow.close();
    }
```

6.57.3.4 render()

```
void view.InitialBoardView.render ( )  [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

The change is setted in the board.

Definition at line 436 of file InitialBoardView.java.

```
                          {
        for (int i = 0; i < 8; i++) {
            char[] row = board.getString(String.format("row%d", i)).toCharArray();
            for (int j = 0; j < 8; j++) drawPiece(new Pair<Integer, Integer>(i, j), row[j]);
        }
    }
```

6.57.3.5 drawPiece()

```
void view.InitialBoardView.drawPiece (
             Pair< Integer, Integer > pos,
             char pieceType )  [private]
```

Painting method executed everytime there is a change in the board.

Precondition

True

Postcondition

Pieces change to the correct color.

Definition at line 448 of file InitialBoardView.java.

```
                                                                            {
        Circle circle = getCircle(pos);
        switch (pieceType) {
            case 'B':
                circle.setFill(Color.web("0xFFFFFF", 1.0));
                break;
            case 'N':
                circle.setFill(Color.web("0x000000", 1.0));
                break;
            case '?':
                circle.setFill(Color.web("0x34d399", 1.0));
                break;
            default:
                break;
        }
    }
```

6.57.3.6 getClickedPos()

```
Pair<Integer, Integer> view.InitialBoardView.getClickedPos (
            MouseEvent mouseEvent )  [private]
```

Painting method executed everytime a player clicks on the board.

Precondition

True

Postcondition

The piece clicked is transformed into a pair.

Definition at line 470 of file InitialBoardView.java.

```
                                                                  {
        Pair<Integer, Integer> pos = new Pair<Integer, Integer>(-1, -1);
        String piece = ((Circle) mouseEvent.getPickResult().getIntersectedNode()).getId();
        pos.first = Character.getNumericValue(piece.charAt(1)) - 1;
        pos.second = Character.getNumericValue(piece.charAt(3)) - 1;
        return pos;
    }
```

6.57.3.7 getCircle()

```
Circle view.InitialBoardView.getCircle (
            Pair< Integer, Integer > pos )  [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

Return the circle which belongs to the position.

Definition at line 483 of file InitialBoardView.java.

```
                                                    {
        try {
            Field field = this.getClass().getDeclaredField(String.format("f%sc%s", pos.first + 1,
      pos.second + 1));
            field.setAccessible(true);
            return (Circle) field.get(this);
        } catch (Exception e) {
            return new Circle();
        }
    }
```

6.57.4 Member Data Documentation

6.57.4.1 f1c1

`Circle view.InitialBoardView.f1c1 [private]`

Piece located in (1, 1).

Definition at line 45 of file InitialBoardView.java.

6.57.4.2 f1c2

`Circle view.InitialBoardView.f1c2 [private]`

Piece located in (1, 2).

Definition at line 50 of file InitialBoardView.java.

6.57.4.3 f1c3

`Circle view.InitialBoardView.f1c3 [private]`

Piece located in (1, 3).

Definition at line 55 of file InitialBoardView.java.

6.57.4.4 f1c4

`Circle view.InitialBoardView.f1c4 [private]`

Piece located in (1, 4).

Definition at line 60 of file InitialBoardView.java.

6.57.4.5 f1c5

`Circle view.InitialBoardView.f1c5 [private]`

Piece located in (1, 5).

Definition at line 65 of file InitialBoardView.java.

6.57.4.6 f1c6

```
Circle view.InitialBoardView.f1c6  [private]
```

Piece located in (1, 6).

Definition at line 70 of file InitialBoardView.java.

6.57.4.7 f1c7

```
Circle view.InitialBoardView.f1c7  [private]
```

Piece located in (1, 7).

Definition at line 75 of file InitialBoardView.java.

6.57.4.8 f1c8

```
Circle view.InitialBoardView.f1c8  [private]
```

Piece located in (1, 8).

Definition at line 80 of file InitialBoardView.java.

6.57.4.9 f2c1

```
Circle view.InitialBoardView.f2c1  [private]
```

Piece located in (2, 1).

Definition at line 85 of file InitialBoardView.java.

6.57.4.10 f2c2

```
Circle view.InitialBoardView.f2c2  [private]
```

Piece located in (2, 2).

Definition at line 90 of file InitialBoardView.java.

6.57.4.11 f2c3

```
Circle view.InitialBoardView.f2c3 [private]
```

Piece located in (2, 3).

Definition at line 95 of file InitialBoardView.java.

6.57.4.12 f2c4

```
Circle view.InitialBoardView.f2c4 [private]
```

Piece located in (2, 4).

Definition at line 100 of file InitialBoardView.java.

6.57.4.13 f2c5

```
Circle view.InitialBoardView.f2c5 [private]
```

Piece located in (2, 5).

Definition at line 105 of file InitialBoardView.java.

6.57.4.14 f2c6

```
Circle view.InitialBoardView.f2c6 [private]
```

Piece located in (2, 6).

Definition at line 110 of file InitialBoardView.java.

6.57.4.15 f2c7

```
Circle view.InitialBoardView.f2c7 [private]
```

Piece located in (2, 7).

Definition at line 115 of file InitialBoardView.java.

6.57.4.16 f2c8

```
Circle view.InitialBoardView.f2c8  [private]
```

Piece located in (2, 8).

Definition at line 120 of file InitialBoardView.java.

6.57.4.17 f3c1

```
Circle view.InitialBoardView.f3c1  [private]
```

Piece located in (3, 1).

Definition at line 125 of file InitialBoardView.java.

6.57.4.18 f3c2

```
Circle view.InitialBoardView.f3c2  [private]
```

Piece located in (3, 2).

Definition at line 130 of file InitialBoardView.java.

6.57.4.19 f3c3

```
Circle view.InitialBoardView.f3c3  [private]
```

Piece located in (3, 3).

Definition at line 135 of file InitialBoardView.java.

6.57.4.20 f3c4

```
Circle view.InitialBoardView.f3c4  [private]
```

Piece located in (3, 4).

Definition at line 140 of file InitialBoardView.java.

6.57.4.21 f3c5

```
Circle view.InitialBoardView.f3c5  [private]
```

Piece located in (3, 5).

Definition at line 145 of file InitialBoardView.java.

6.57.4.22 f3c6

```
Circle view.InitialBoardView.f3c6  [private]
```

Piece located in (3, 6).

Definition at line 150 of file InitialBoardView.java.

6.57.4.23 f3c7

```
Circle view.InitialBoardView.f3c7  [private]
```

Piece located in (3, 7).

Definition at line 155 of file InitialBoardView.java.

6.57.4.24 f3c8

```
Circle view.InitialBoardView.f3c8  [private]
```

Piece located in (3, 8).

Definition at line 160 of file InitialBoardView.java.

6.57.4.25 f4c1

```
Circle view.InitialBoardView.f4c1  [private]
```

Piece located in (4, 1).

Definition at line 165 of file InitialBoardView.java.

6.57.4.26 f4c2

```
Circle view.InitialBoardView.f4c2 [private]
```

Piece located in (4, 2).

Definition at line 170 of file InitialBoardView.java.

6.57.4.27 f4c3

```
Circle view.InitialBoardView.f4c3 [private]
```

Piece located in (4, 3).

Definition at line 175 of file InitialBoardView.java.

6.57.4.28 f4c4

```
Circle view.InitialBoardView.f4c4 [private]
```

Piece located in (4, 4).

Definition at line 180 of file InitialBoardView.java.

6.57.4.29 f4c5

```
Circle view.InitialBoardView.f4c5 [private]
```

Piece located in (4, 5).

Definition at line 185 of file InitialBoardView.java.

6.57.4.30 f4c6

```
Circle view.InitialBoardView.f4c6 [private]
```

Piece located in (4, 6).

Definition at line 190 of file InitialBoardView.java.

6.57.4.31 f4c7

```
Circle view.InitialBoardView.f4c7 [private]
```

Piece located in (4, 7).

Definition at line 195 of file InitialBoardView.java.

6.57.4.32 f4c8

```
Circle view.InitialBoardView.f4c8 [private]
```

Piece located in (4, 8).

Definition at line 200 of file InitialBoardView.java.

6.57.4.33 f5c1

```
Circle view.InitialBoardView.f5c1 [private]
```

Piece located in (5, 1).

Definition at line 205 of file InitialBoardView.java.

6.57.4.34 f5c2

```
Circle view.InitialBoardView.f5c2 [private]
```

Piece located in (5, 2).

Definition at line 210 of file InitialBoardView.java.

6.57.4.35 f5c3

```
Circle view.InitialBoardView.f5c3 [private]
```

Piece located in (5, 3).

Definition at line 215 of file InitialBoardView.java.

6.57.4.36 f5c4

```
Circle view.InitialBoardView.f5c4  [private]
```

Piece located in (5, 4).

Definition at line 220 of file InitialBoardView.java.

6.57.4.37 f5c5

```
Circle view.InitialBoardView.f5c5  [private]
```

Piece located in (5, 5).

Definition at line 225 of file InitialBoardView.java.

6.57.4.38 f5c6

```
Circle view.InitialBoardView.f5c6  [private]
```

Piece located in (5, 6).

Definition at line 230 of file InitialBoardView.java.

6.57.4.39 f5c7

```
Circle view.InitialBoardView.f5c7  [private]
```

Piece located in (5, 7).

Definition at line 235 of file InitialBoardView.java.

6.57.4.40 f5c8

```
Circle view.InitialBoardView.f5c8  [private]
```

Piece located in (5, 8).

Definition at line 240 of file InitialBoardView.java.

6.57.4.41 f6c1

```
Circle view.InitialBoardView.f6c1 [private]
```

Piece located in (6, 1).

Definition at line 245 of file InitialBoardView.java.

6.57.4.42 f6c2

```
Circle view.InitialBoardView.f6c2 [private]
```

Piece located in (6, 2).

Definition at line 250 of file InitialBoardView.java.

6.57.4.43 f6c3

```
Circle view.InitialBoardView.f6c3 [private]
```

Piece located in (6, 3).

Definition at line 255 of file InitialBoardView.java.

6.57.4.44 f6c4

```
Circle view.InitialBoardView.f6c4 [private]
```

Piece located in (6, 4).

Definition at line 260 of file InitialBoardView.java.

6.57.4.45 f6c5

```
Circle view.InitialBoardView.f6c5 [private]
```

Piece located in (6, 5).

Definition at line 265 of file InitialBoardView.java.

6.57.4.46 f6c6

```
Circle view.InitialBoardView.f6c6 [private]
```

Piece located in (6, 6).

Definition at line 270 of file InitialBoardView.java.

6.57.4.47 f6c7

```
Circle view.InitialBoardView.f6c7 [private]
```

Piece located in (6, 7).

Definition at line 275 of file InitialBoardView.java.

6.57.4.48 f6c8

```
Circle view.InitialBoardView.f6c8 [private]
```

Piece located in (6, 8).

Definition at line 280 of file InitialBoardView.java.

6.57.4.49 f7c1

```
Circle view.InitialBoardView.f7c1 [private]
```

Piece located in (7, 1).

Definition at line 285 of file InitialBoardView.java.

6.57.4.50 f7c2

```
Circle view.InitialBoardView.f7c2 [private]
```

Piece located in (7, 2).

Definition at line 290 of file InitialBoardView.java.

6.57.4.51 f7c3

`Circle view.InitialBoardView.f7c3 [private]`

Piece located in (7, 3).

Definition at line 295 of file InitialBoardView.java.

6.57.4.52 f7c4

`Circle view.InitialBoardView.f7c4 [private]`

Piece located in (7, 4).

Definition at line 300 of file InitialBoardView.java.

6.57.4.53 f7c5

`Circle view.InitialBoardView.f7c5 [private]`

Piece located in (7, 5).

Definition at line 305 of file InitialBoardView.java.

6.57.4.54 f7c6

`Circle view.InitialBoardView.f7c6 [private]`

Piece located in (7, 6).

Definition at line 310 of file InitialBoardView.java.

6.57.4.55 f7c7

`Circle view.InitialBoardView.f7c7 [private]`

Piece located in (7, 7).

Definition at line 315 of file InitialBoardView.java.

6.57.4.56 f7c8

```
Circle view.InitialBoardView.f7c8 [private]
```

Piece located in (7, 8).

Definition at line 320 of file InitialBoardView.java.

6.57.4.57 f8c1

```
Circle view.InitialBoardView.f8c1 [private]
```

Piece located in (8, 1).

Definition at line 325 of file InitialBoardView.java.

6.57.4.58 f8c2

```
Circle view.InitialBoardView.f8c2 [private]
```

Piece located in (8, 2).

Definition at line 330 of file InitialBoardView.java.

6.57.4.59 f8c3

```
Circle view.InitialBoardView.f8c3 [private]
```

Piece located in (8, 3).

Definition at line 335 of file InitialBoardView.java.

6.57.4.60 f8c4

```
Circle view.InitialBoardView.f8c4 [private]
```

Piece located in (8, 4).

Definition at line 340 of file InitialBoardView.java.

6.57.4.61 f8c5

```
Circle view.InitialBoardView.f8c5 [private]
```

Piece located in (8, 5).

Definition at line 345 of file InitialBoardView.java.

6.57.4.62 f8c6

```
Circle view.InitialBoardView.f8c6 [private]
```

Piece located in (8, 6).

Definition at line 350 of file InitialBoardView.java.

6.57.4.63 f8c7

```
Circle view.InitialBoardView.f8c7 [private]
```

Piece located in (8, 7).

Definition at line 355 of file InitialBoardView.java.

6.57.4.64 f8c8

```
Circle view.InitialBoardView.f8c8 [private]
```

Piece located in (8, 8).

Definition at line 360 of file InitialBoardView.java.

6.57.4.65 save

```
Text view.InitialBoardView.save [private]
```

Save board button text.

Definition at line 365 of file InitialBoardView.java.

6.57.4.66 saveButton

`Rectangle view.InitialBoardView.saveButton [private]`

Save board button.

Definition at line 370 of file InitialBoardView.java.

6.57.4.67 placeWhitePieces

`RadioButton view.InitialBoardView.placeWhitePieces [private]`

White colour pieces selector.

Definition at line 375 of file InitialBoardView.java.

6.57.4.68 placeBlackPieces

`RadioButton view.InitialBoardView.placeBlackPieces [private]`

Black colour pieces selector.

Definition at line 380 of file InitialBoardView.java.

6.57.4.69 quitPieces

`RadioButton view.InitialBoardView.quitPieces [private]`

Remove pieces selector.

Definition at line 385 of file InitialBoardView.java.

6.57.4.70 saveInitialBoardResult

`Label view.InitialBoardView.saveInitialBoardResult [private]`

Exception message output.

Definition at line 390 of file InitialBoardView.java.

6.57.4.71 board

```
JSONObject view.InitialBoardView.board [private]
```

Current board.

Definition at line 394 of file InitialBoardView.java.

The documentation for this class was generated from the following file:

- InitialBoardView.java

6.58 domain.Exceptions.InvalidBoardException Class Reference

The current board is in an illegal state. By Alex Rodriguez.

Public Member Functions

- InvalidBoardException ()

6.58.1 Detailed Description

The current board is in an illegal state. By Alex Rodriguez.

Definition at line 151 of file Exceptions.java.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 InvalidBoardException()

```
domain.Exceptions.InvalidBoardException.InvalidBoardException ( )
```

Definition at line 152 of file Exceptions.java.

```
                                            {
                super("ERR_INVALID_BOARD");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.59 domain.Exceptions.InvalidConfigurationException Class Reference

The entered configuration is null, empty or blank. By Alex Rodriguez.

Public Member Functions

- InvalidConfigurationException ()

6.59.1 Detailed Description

The entered configuration is null, empty or blank. By Alex Rodriguez.

Definition at line 195 of file Exceptions.java.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 InvalidConfigurationException()

```
domain.Exceptions.InvalidConfigurationException.InvalidConfigurationException ( )
```

Definition at line 196 of file Exceptions.java.

```
                                               {
            super("ERR_INVALID_CONFIGURATION");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.60 domain.Exceptions.InvalidDifficultyException Class Reference

The entered difficulty is null, empty, blank, less than 0 or greater than 10. By Alex Rodriguez.

Public Member Functions

- InvalidDifficultyException ()

6.60.1 Detailed Description

The entered difficulty is null, empty, blank, less than 0 or greater than 10. By Alex Rodriguez.

Definition at line 118 of file Exceptions.java.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 InvalidDifficultyException()

```
domain.Exceptions.InvalidDifficultyException.InvalidDifficultyException ( )
```

Definition at line 119 of file Exceptions.java.

```
                                             {
            super("ERR_INVALID_DIFFICULTY");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.61 domain.Exceptions.InvalidNameException Class Reference

The entered name is null, empty or blank. By Alex Rodriguez.

Public Member Functions

- InvalidNameException ()

6.61.1 Detailed Description

The entered name is null, empty or blank. By Alex Rodriguez.

Definition at line 30 of file Exceptions.java.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 InvalidNameException()

```
domain.Exceptions.InvalidNameException.InvalidNameException ( )
```

Definition at line 31 of file Exceptions.java.

```
                                         {
            super("ERR_INVALID_NAME");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.62 domain.Exceptions.InvalidPasswordException Class Reference

The entered password is null, empty or blank. By Alex Rodriguez.

Public Member Functions

- InvalidPasswordException ()

6.62.1 Detailed Description

The entered password is null, empty or blank. By Alex Rodriguez.

Definition at line 41 of file Exceptions.java.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 InvalidPasswordException()

```
domain.Exceptions.InvalidPasswordException.InvalidPasswordException ( )
```

Definition at line 42 of file Exceptions.java.

```
                                       {
            super("ERR_INVALID_PASSWORD");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.63 domain.Exceptions.InvalidPlayersException Class Reference

The entered players are the same, null, empty, blank or both bots have the same difficulty. By Alex Rodriguez.

Public Member Functions

- InvalidPlayersException ()

6.63.1 Detailed Description

The entered players are the same, null, empty, blank or both bots have the same difficulty. By Alex Rodriguez.

Definition at line 184 of file Exceptions.java.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 InvalidPlayersException()

```
domain.Exceptions.InvalidPlayersException.InvalidPlayersException ( )
```

Definition at line 185 of file Exceptions.java.

```
                                         {
            super("ERR_INVALID_PLAYERS");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.64 domain.Exceptions.InvalidPositionException Class Reference

The entered position is null, empty, blank or ends up with an invalid board. By Alex Rodriguez.

Public Member Functions

- InvalidPositionException ()

6.64.1 Detailed Description

The entered position is null, empty, blank or ends up with an invalid board. By Alex Rodriguez.

Definition at line 173 of file Exceptions.java.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 InvalidPositionException()

```
domain.Exceptions.InvalidPositionException.InvalidPositionException ( )
```

Definition at line 174 of file Exceptions.java.

```
                                          {
            super("ERR_INVALID_POSITION");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.65 domain.Exceptions.InvalidRulesException Class Reference

The entered configuration rules are all deactivated. By Alex Rodriguez.

Public Member Functions

- InvalidRulesException ()

6.65.1 Detailed Description

The entered configuration rules are all deactivated. By Alex Rodriguez.

Definition at line 162 of file Exceptions.java.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 InvalidRulesException()

```
domain.Exceptions.InvalidRulesException.InvalidRulesException ( )
```

Definition at line 163 of file Exceptions.java.

```
                                        {
              super("ERR_INVALID_RULES");
          }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.66 view.LogInView Class Reference

Public Member Functions

- LogInView ()

  Class creator.
- void signIn () throws IOException

  Event method which is executed when the logIn button is clicked.
- void signUp () throws IOException

  Event method which is executed when the signUp button is clicked.

Private Attributes

- Text logIn

  logIn view change button.
- Text signUp

  signUp view change button.
- TextField username

  User name text field.
- PasswordField password

  User password field.
- Label logInResult

  Exception output message label.
- Text signIn

  logIn button text.
- Rectangle signInButton

  logIn button.

6.66.1 Detailed Description

This class represents the scene controller of the LogIn.

Done by Arnau Pujantell

Definition at line 23 of file LogInView.java.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 LogInView()

`view.LogInView.LogInView ( )`

Class creator.

Definition at line 30 of file LogInView.java.

```
                         {
    }
```

6.66.3 Member Function Documentation

6.66.3.1 signIn()

`void view.LogInView.signIn ( ) throws IOException`

Event method which is executed when the logIn button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, scene changes to BotsView.

Definition at line 78 of file LogInView.java.

```
                                              {
        Pair<JSONObject, String> result = ViewCtrl.domainCtrl.login(username.getText(),
    password.getText());
        if (result.second != null) {
            switch (result.second) {
                case "ERR_INVALID_NAME":
                    logInResult.setText("Username can’t be empty!");
                    break;
                    case "ERR_INVALID_PASSWORD":
                    logInResult.setText("Password can’t be empty!");
                    break;
                    case "ERR_INEXISTING_PLAYER":
                    logInResult.setText("The player does not exist!");
                    break;
                    case "ERR_INCORRECT_CREDENTIALS":
                    logInResult.setText("Wrong username or password!");
                    break;
                default:
                logInResult.setText("Something went wrong, try again!");
                    break;
            }
        } else {
            ViewCtrl.changeScene("template/UserView.fxml");
         }
    }
```

6.66.3.2 signUp()

```
void view.LogInView.signUp ( ) throws IOException
```

Event method which is executed when the signUp button is clicked.

Precondition

True

Postcondition

Scene changes to SignUpView.

Definition at line 108 of file LogInView.java.

```
                                     {
        ViewCtrl.changeScene("template/SignUpView.fxml");
    }
```

6.66.4 Member Data Documentation

6.66.4.1 logIn

```
Text view.LogInView.logIn  [private]
```

logIn view change button.

Definition at line 39 of file LogInView.java.

6.66.4.2 signUp

```
Text view.LogInView.signUp  [private]
```

signUp view change button.

Definition at line 44 of file LogInView.java.

6.66.4.3 username

```
TextField view.LogInView.username  [private]
```

User name text field.

Definition at line 49 of file LogInView.java.

6.66.4.4 password

```
PasswordField view.LogInView.password  [private]
```

User password field.

Definition at line 54 of file LogInView.java.

6.66.4.5 logInResult

```
Label view.LogInView.logInResult  [private]
```

Exception output message label.

Definition at line 59 of file LogInView.java.

6.66.4.6 signIn

```
Text view.LogInView.signIn  [private]
```

logIn button text.

Definition at line 64 of file LogInView.java.

6.66.4.7 signInButton

```
Rectangle view.LogInView.signInButton  [private]
```

logIn button.

Definition at line 69 of file LogInView.java.

The documentation for this class was generated from the following file:

- LogInView.java

6.67 cmd.driver.mediumDifficulty Class Reference

MediumDifficulty driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)

  MediumDifficulty driver main function. Creates an instance of the MediumDifficulty driver and starts it.

6.67.1 Detailed Description

MediumDifficulty driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file mediumDifficulty.java.

6.67.2 Member Function Documentation

6.67.2.1 main()

```
static void cmd.driver.mediumDifficulty.main (
             String[ ] args ) [static]
```

MediumDifficulty driver main function. Creates an instance of the MediumDifficulty driver and starts it.

Precondition

True.

Postcondition

The MediumDifficulty driver has started.

Definition at line 22 of file mediumDifficulty.java.

```
                                             {
           new MediumDifficultyDriver().start();
       }
```

The documentation for this class was generated from the following file:

- mediumDifficulty.java

6.68 domain.MediumDifficulty Class Reference

Implements the Minimax algorithm with alpha-beta pruning to get the next best possible position for a given player. By Alex Rodriguez.

Public Member Functions

- MediumDifficulty (Integer difficulty, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally, PieceType pieceType)

  Create a MediumDifficulty instance.
- Pair< Integer, Integer > place (PieceType[ ][ ] playingBoard)

  Get the next best possible position for the implicit player.

Private Member Functions

- int evaluation (Board currentBoard)

  Get the heuristic evaluation for the given Board state.
- int minimax (Board currentBoard, PieceType currentPieceType, int depth, int alpha, int beta)

  Recursive implementation of the Minimax algorithm with alpha-beta pruning.

Additional Inherited Members

6.68.1 Detailed Description

Implements the Minimax algorithm with alpha-beta pruning to get the next best possible position for a given player. By Alex Rodriguez.

Definition at line 18 of file MediumDifficulty.java.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 MediumDifficulty()

```
domain.MediumDifficulty.MediumDifficulty (
            Integer difficulty,
            Boolean canEatHorizontally,
            Boolean canEatVertically,
            Boolean canEatDiagonally,
            PieceType pieceType )
```

Create a MediumDifficulty instance.

Precondition

The given difficulty is a positive number. The given rules are not all false.

Postcondition

A MediumDifficulty instance is created and its implicits difficulty, canEatHorizontally, canEatVertically, can↩ EatDiagonally and pieceType attributes are setted. The implicit maxDepth attribute is setted to the double of the entered difficulty.

Parameters

difficulty	Difficulty for the Minimax algorithm with alpha-beta pruning.
canEatHorizontally	Whether the pieces of the current Game can be eaten horizontally.
canEatVertically	Whether the pieces of the current Game can be eaten vertically.
canEatDiagonally	Whether the pieces of the current Game can be eaten diagonally.
pieceType	Player that wants to be maximized.

Definition at line 34 of file MediumDifficulty.java.

```
                                                    {
        super(difficulty, canEatHorizontally, canEatVertically, canEatDiagonally, pieceType);
    }
```

6.68.3 Member Function Documentation

6.68.3.1 evaluation()

```
int domain.MediumDifficulty.evaluation (
            Board currentBoard )  [private]
```

Get the heuristic evaluation for the given Board state.

Precondition

True

Postcondition

It is returned the heuristic evaluation for the given Board state. The evaluation is the subtraction of the maximized player's control of the board minus the control of the board for the opponent. A player's control of the board is obtained with the number of pieces in his control and adding or subtracting to that based on important positions in the board. Those important positions are corners, positions adjacent to corners, borders of the board which aren't adjacent to corners and positions adjacent to the centre square of the board.

Parameters

currentBoard	Current playing Board to get the heuristic evaluation from.

Returns

The heuristic evaluation for the given Board state.

Definition at line 51 of file MediumDifficulty.java.

```
                                          {
        int player1 = currentBoard.getPiecesPlayer1();
        int player2 = currentBoard.getPiecesPlayer2();

        PieceType[][] board = currentBoard.getBoard();

        // Check corners of the Board
        if (board[0][0] == PieceType.PLAYER1) player1 += 50;
        else if (board[0][0] == PieceType.PLAYER2) player2 += 50;

        if (board[0][7] == PieceType.PLAYER1) player1 += 50;
        else if (board[0][7] == PieceType.PLAYER2) player2 += 50;

        if (board[7][0] == PieceType.PLAYER1) player1 += 50;
        else if (board[7][0] == PieceType.PLAYER2) player2 += 50;

        if (board[7][7] == PieceType.PLAYER1) player1 += 50;
        else if (board[7][7] == PieceType.PLAYER2) player2 += 50;

        // Check borders not next to corner
        for (int k = 2; k < 6; ++k) {
            if (board[k][0] == PieceType.PLAYER1) player1 += 17;
            else if (board[k][0] == PieceType.PLAYER2) player2 += 17;

            if (board[k][7] == PieceType.PLAYER1) player1 += 17;
            else if (board[k][7] == PieceType.PLAYER2) player2 += 17;

            if (board[0][k] == PieceType.PLAYER1) player1 += 17;
            else if (board[0][k] == PieceType.PLAYER2) player2 += 17;

            if (board[7][k] == PieceType.PLAYER1) player1 += 17;
            else if (board[7][k] == PieceType.PLAYER2) player2 += 17;
        }

        // Check next to center of the Board
        for (int i = 2; i < 6; ++i) {
            if (board[i][2] == PieceType.PLAYER1) player1 += 10;
            else if (board[i][2] == PieceType.PLAYER2) player2 += 10;

            if (board[i][5] == PieceType.PLAYER1) player1 += 10;
            else if (board[i][5] == PieceType.PLAYER2) player2 += 10;

            if (board[2][i] == PieceType.PLAYER1) player1 += 10;
            else if (board[2][i] == PieceType.PLAYER2) player2 += 10;

            if (board[5][i] == PieceType.PLAYER1) player1 += 10;
            else if (board[5][i] == PieceType.PLAYER2) player2 += 10;
        }

        // Check next to corners
        for (int j = 0; j < 2; ++j) {
            if (board[1][j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[1][j] == PieceType.PLAYER2) player2 -= 25;

            if (board[1][7 - j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[1][7 - j] == PieceType.PLAYER2) player2 -= 25;

            if (board[6][j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[6][j] == PieceType.PLAYER2) player2 -= 25;

            if (board[6][7 - j] == PieceType.PLAYER1) player1 -= 25;
            else if (board[6][7 - j] == PieceType.PLAYER2) player2 -= 25;
        }

        if (board[0][1] == PieceType.PLAYER1) player1 -= 25;
        else if (board[0][1] == PieceType.PLAYER2) player2 -= 25;

        if (board[7][1] == PieceType.PLAYER1) player1 -= 25;
        else if (board[7][1] == PieceType.PLAYER2) player2 -= 25;

        if (board[0][6] == PieceType.PLAYER1) player1 -= 25;
        else if (board[0][6] == PieceType.PLAYER2) player2 -= 25;

        if (board[7][6] == PieceType.PLAYER1) player1 -= 25;
        else if (board[7][6] == PieceType.PLAYER2) player2 -= 25;

        if (this.pieceType == PieceType.PLAYER1) return player1 - player2;
        else return player2 - player1;
    }
```

6.68.3.2 minimax()

```
int domain.MediumDifficulty.minimax (
            Board currentBoard,
            PieceType currentPieceType,
            int depth,
            int alpha,
            int beta )  [private]
```

Recursive implementation of the Minimax algorithm with alpha-beta pruning.

Precondition

True

Postcondition

It is returned the heuristic evaluation for the current possible position on the tree of possibilities. If there aren't any possible valid positions left or the maximum depth is reached it stops. The implicit player is maximized and the opponent is minimized.

Parameters

currentBoard	current Board in the tree of possibilities.
currentPieceType	current turn in the tree of possibilities.
depth	current depth in the tree of possibilities.
alpha	current alpha in the tree of possibilities.
beta	current beta in the tree of possibilities.

Returns

The heuristic evaluation for the current possible position on the tree of possibilities.

Definition at line 144 of file MediumDifficulty.java.

```
{
    ArrayList<Pair<Integer, Integer>> validPositions = currentBoard.validPositions(currentPieceType,
            this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);

    if (validPositions.isEmpty() || depth == 0)
        return this.evaluation(currentBoard);

    // Maximizer
    if (currentPieceType == this.pieceType) {
        int max = Integer.MIN_VALUE, currentMax = 0;

        for (Pair<Integer, Integer> position : validPositions) {
            // Make a duplicate in order not to work with the same Board pointer!
            Board board = new Board(currentBoard.getBoard());
            board.placePiece(position, currentPieceType, this.canEatHorizontally,
this.canEatVertically,
                    this.canEatDiagonally);

            currentMax = this.minimax(board, MediumDifficulty.inversePieceType(currentPieceType),
depth - 1, alpha, beta);
            max = Integer.max(max, currentMax);
            alpha = Integer.max(alpha, currentMax);
            // Prune
            if (beta <= alpha)
                break;
        }

        return max;
    }

    // Minimizer
    else {
        Integer min = Integer.MAX_VALUE, currentMin = 0;

        for (Pair<Integer, Integer> position : validPositions) {
            // Make a duplicate in order not to work with the same Board pointer!
```

```
                Board board = new Board(currentBoard.getBoard());
                board.placePiece(position, currentPieceType, this.canEatHorizontally,
        this.canEatVertically,
                        this.canEatDiagonally);

                currentMin = this.minimax(board, MediumDifficulty.inversePieceType(currentPieceType),
        depth - 1, alpha, beta);
                min = Integer.min(min, currentMin);
                beta = Integer.min(beta, currentMin);
                // Prune
                if (beta <= alpha)
                    break;
            }

            return min;
        }
    }
```

6.68.3.3 place()

```
Pair<Integer, Integer> domain.MediumDifficulty.place (
        PieceType playingBoard[ ][ ] )
```

Get the next best possible position for the implicit player.

Precondition

True

Postcondition

It is returned the next best possible position for the implicit player, using the Minimax algorithm with alpha-beta pruning with the implicit maximum depth, or null if there isn't any.

Parameters

playingBoard	Current playing Board.

Returns

The next best possible position for the implicit player or null if there isn't any.

Reimplemented from domain.Difficulty.

Definition at line 203 of file MediumDifficulty.java.

```
                                                                    {
        Pair<Integer, Integer> bestPosition = null;

        Board initialBoard = new Board(playingBoard);

        ArrayList<Pair<Integer, Integer>> validPositions = initialBoard.validPositions(this.pieceType,
                this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);

        int max = Integer.MIN_VALUE, currentMax = 0;

        for (Pair<Integer, Integer> position : validPositions) {
            // Make a duplicate in order not to work with the same Board pointer!
            Board board = new Board(initialBoard.getBoard());
            board.placePiece(position, this.pieceType, this.canEatHorizontally, this.canEatVertically,
                    this.canEatDiagonally);

            currentMax = this.minimax(board, MediumDifficulty.inversePieceType(this.pieceType),
      this.maxDepth - 1,
                    Integer.MIN_VALUE, Integer.MAX_VALUE);
            if (currentMax > max) {
                max = currentMax;
                bestPosition = position;
            }
        }

        return bestPosition;
    }
```

The documentation for this class was generated from the following file:

- MediumDifficulty.java

6.69 test.driver.MediumDifficultyDriver Class Reference

Implements the different options for the MediumDifficulty driver application. By Alex Rodriguez.

Public Member Functions

- MediumDifficultyDriver ()
- void start ()

Public Attributes

- MediumDifficulty currentMediumDifficulty
- Board currentBoard
- String nameCurrentBoard
- FixtureRepository fixtureRepository

Private Member Functions

- void mainMenu ()
- void create ()
- void getDifficulty ()
- void getCanEatHorizontally ()
- void getCanEatVertically ()
- void getCanEatDiagonally ()
- void getPieceType ()
- void getMaxDepth ()
- void setMaxDepth ()
- void loadBoard ()
- void printCurrentBoard ()
- void getNextBestPosition ()
- Pair< String, Board > listBoardFixtures ()
- void printBoard (Board board)
- ArrayList< String > transcribeToCharacters (Board board)

Additional Inherited Members

6.69.1 Detailed Description

Implements the different options for the MediumDifficulty driver application. By Alex Rodriguez.

Definition at line 21 of file MediumDifficultyDriver.java.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 MediumDifficultyDriver()

```
test.driver.MediumDifficultyDriver.MediumDifficultyDriver ( )
```

Definition at line 33 of file MediumDifficultyDriver.java.

```
                                 {
        this.currentMediumDifficulty = null;
        this.fixtureRepository = new FixtureRepository();
    }
```

6.69.3 Member Function Documentation

6.69.3.1 start()

```
void test.driver.MediumDifficultyDriver.start ( )
```

Definition at line 40 of file MediumDifficultyDriver.java.

```
                       {
        while (true) {
            this.mainMenu();
        }
    }
```

6.69.3.2 mainMenu()

```
void test.driver.MediumDifficultyDriver.mainMenu ( )  [private]
```

Definition at line 46 of file MediumDifficultyDriver.java.

```
                             {
        String title = null;
        if (this.currentMediumDifficulty != null)
            title = String.format("Current maximum depth: %s\n",
      this.currentMediumDifficulty.getMaxDepth());
        if (this.currentBoard != null)
            title += String.format("Current Board: %s\n", this.nameCurrentBoard);

        switch (Driver.menu(title, "MediumDifficulty (Minimax with alpha-beta pruning) Driver",
                new Pair<String, String>("1", "Create MediumDifficulty"),
                new Pair<String, String>("2", "Get difficulty"),
                new Pair<String, String>("3", "Get canEatHorizontally"),
                new Pair<String, String>("4", "Get canEatVertically"),
                new Pair<String, String>("5", "Get canEatDiagonally"),
                new Pair<String, String>("6", "Get pieceType"),
                new Pair<String, String>("7", "Get maxDepth"),
                new Pair<String, String>("8", "Set maxDepth"),
                new Pair<String, String>("9", "Load Board"),
                new Pair<String, String>("10", "Print Current Board"),
                new Pair<String, String>("11", "Get next best position"))) {
        case "1":
            this.create();
            break;
        case "2":
            this.getDifficulty();
            break;
        case "3":
            this.getCanEatHorizontally();
            break;
        case "4":
            this.getCanEatVertically();
            break;
        case "5":
            this.getCanEatDiagonally();
            break;
        case "6":
            this.getPieceType();
            break;
        case "7":
            this.getMaxDepth();
            break;
        case "8":
            this.setMaxDepth();
            break;
        case "9":
            this.loadBoard();
            break;
        case "10":
            this.printCurrentBoard();
            break;
        case "11":
            this.getNextBestPosition();
            break;
        }
        Driver.pause();
    }
```

6.69.3.3 create()

```
void test.driver.MediumDifficultyDriver.create ( )  [private]
```

Definition at line 102 of file MediumDifficultyDriver.java.

```
                           {
        System.out.println(
                "Take into account that the default maximum depth is the double of the entered
      difficulty.\nMinimax with alpha-beta pruning with higher depths requires more time to execute. A
      value of 3 for the difficulty is reasonable.\n");

        Integer difficulty = Driver.inputInt("Difficulty (positive)?");
        Boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
        Boolean canEatVertically = Driver.inputBool("Can eat vertically?");
        Boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
        PieceType pieceType = null;

        switch (Driver.menu(null, "Select Bot pieces",
                new Pair<String, String>("1", "PLAYER 1 pieces"),
                new Pair<String, String>("2", "PLAYER 2 pieces"))) {
        case "1":
            pieceType = PieceType.PLAYER1;
            break;
        case "2":
            pieceType = PieceType.PLAYER2;
            break;
        }

        this.currentMediumDifficulty = new MediumDifficulty(difficulty, canEatHorizontally,
canEatVertically,
                canEatDiagonally, pieceType);

        System.out.println(String.format("MediumDifficulty with a maximum depth of %s created
successfully!",
                this.currentMediumDifficulty.getMaxDepth()));
    }
```

6.69.3.4 getDifficulty()

```
void test.driver.MediumDifficultyDriver.getDifficulty ( ) [private]
```

Definition at line 130 of file MediumDifficultyDriver.java.

```
                                     {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(
                String.format("MediumDifficulty's difficulty is: %s",
this.currentMediumDifficulty.getDifficulty()));
    }
```

6.69.3.5 getCanEatHorizontally()

```
void test.driver.MediumDifficultyDriver.getCanEatHorizontally ( ) [private]
```

Definition at line 140 of file MediumDifficultyDriver.java.

```
                                             {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(String.format("MediumDifficulty's canEatHorizontally is: %s",
                this.currentMediumDifficulty.getCanEatHorizontally()));
    }
```

6.69.3.6 getCanEatVertically()

```
void test.driver.MediumDifficultyDriver.getCanEatVertically ( ) [private]
```

Definition at line 150 of file MediumDifficultyDriver.java.

```
                                           {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(String.format("MediumDifficulty's canEatVertically is: %s",
                this.currentMediumDifficulty.getCanEatVertically()));
    }
```

6.69.3.7 getCanEatDiagonally()

`void test.driver.MediumDifficultyDriver.getCanEatDiagonally ( ) [private]`

Definition at line 160 of file MediumDifficultyDriver.java.

```
                                        {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(String.format("MediumDifficulty's canEatDiagonally is: %s",
                this.currentMediumDifficulty.getCanEatDiagonally()));
    }
```

6.69.3.8 getPieceType()

`void test.driver.MediumDifficultyDriver.getPieceType ( ) [private]`

Definition at line 170 of file MediumDifficultyDriver.java.

```
                                 {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(
                String.format("MediumDifficulty's pieceType is: %s",
     this.currentMediumDifficulty.getPieceType()));
    }
```

6.69.3.9 getMaxDepth()

`void test.driver.MediumDifficultyDriver.getMaxDepth ( ) [private]`

Definition at line 180 of file MediumDifficultyDriver.java.

```
                                {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(
                String.format("MediumDifficulty's maxDepth is: %s",
     this.currentMediumDifficulty.getMaxDepth()));
    }
```

6.69.3.10 setMaxDepth()

`void test.driver.MediumDifficultyDriver.setMaxDepth ( ) [private]`

Definition at line 190 of file MediumDifficultyDriver.java.

```
                                {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        System.out.println(
                "Take into account that minimax with alpha-beta pruning with higher depths requires more
     time to execute. A value of 7 is reasonable.\n");

        this.currentMediumDifficulty.setMaxDepth(Driver.inputInt("Maximum depth (positive)?"));
        System.out.println("MediumDifficulty's maxDepth changed successfully!");
    }
```

6.69.3.11 loadBoard()

`void test.driver.MediumDifficultyDriver.loadBoard ( ) [private]`

Definition at line 203 of file MediumDifficultyDriver.java.

```
                          {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        Pair<String, Board> selectedBoard = this.listBoardFixtures();

        this.nameCurrentBoard = selectedBoard.first;
        this.currentBoard = selectedBoard.second;

        System.out.println(String.format("Board %s loaded successfully!\n", this.nameCurrentBoard));
        this.printBoard(this.currentBoard);
    }
```

6.69.3.12 printCurrentBoard()

`void test.driver.MediumDifficultyDriver.printCurrentBoard ( ) [private]`

Definition at line 218 of file MediumDifficultyDriver.java.

```
                                  {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        if (this.currentBoard == null) {
            System.out.println("No current Board!");
            return;
        }

        System.out.println(String.format("Board %s printed successfully!\n", this.nameCurrentBoard));
        this.printBoard(this.currentBoard);
    }
```

6.69.3.13 getNextBestPosition()

`void test.driver.MediumDifficultyDriver.getNextBestPosition ( ) [private]`

Definition at line 233 of file MediumDifficultyDriver.java.

```
                                    {
        if (this.currentMediumDifficulty == null) {
            System.out.println("No current MediumDifficulty!");
            return;
        }

        if (this.currentBoard == null) {
            System.out.println("No current Board!");
            return;
        }

        System.out.println("Take into account that the state of the current Board won’t be globally
    modified.\n");

        this.printBoard(this.currentBoard);

        long startTime = System.currentTimeMillis();
        Pair<Integer, Integer> nextBestPosition =
    this.currentMediumDifficulty.place(this.currentBoard.getBoard());
        long durationTime = System.currentTimeMillis() - startTime;

        Board tempBoard = new Board(this.currentBoard.getBoard());

        if (nextBestPosition != null) {
            tempBoard.placePiece(nextBestPosition, this.currentMediumDifficulty.getPieceType(),
                    this.currentMediumDifficulty.getCanEatHorizontally(),
                    this.currentMediumDifficulty.getCanEatVertically(),
                    this.currentMediumDifficulty.getCanEatDiagonally());
            System.out.println(
                    String.format("The best position calculated in %s ms is %s\n", durationTime, nextBestPosition));
            System.out.println("The addition of the piece would look like this:\n");
            this.printBoard(tempBoard);
        } else {
            System.out.println("There isn't any possible position left to place a piece on.");
        }
    }
```

6.69.3.14 listBoardFixtures()

`Pair<String, Board> test.driver.MediumDifficultyDriver.listBoardFixtures ( ) [private]`

Definition at line 268 of file MediumDifficultyDriver.java.

```
                                                    {
        Integer selectedBoard = -1;
        ArrayList<String> listBoards = this.fixtureRepository.listFiles();

        while (selectedBoard < 0 || selectedBoard >= listBoards.size()) {
            Driver.clear();
            System.out.println("==== Available Boards ====\n");

            for (Integer i = 0; i < listBoards.size(); ++i)
                System.out.println(String.format("[%d]\t%s", i, listBoards.get(i)));
            System.out.println("");

            selectedBoard = Driver.inputInt("What Board would you like to load?");
        }

        Driver.clear();

        return new Pair<String, Board>(listBoards.get(selectedBoard),
                new Board(this.fixtureRepository.boardFileToJSON(listBoards.get(selectedBoard))));
    }
```

6.69.3.15 printBoard()

```
void test.driver.MediumDifficultyDriver.printBoard (
              Board board ) [private]
```

Definition at line 289 of file MediumDifficultyDriver.java.

```
                                         {
        ArrayList<String> boardCodified = this.transcribeToCharacters(board);
        System.out.println("       0  1  2  3  4  5  6  7");
        System.out.println("    -------------------------");

        for (Integer i = 0; i < 8; ++i) {
            String row = boardCodified.get(i);
            System.out.println("  " + i + " |  " + row.charAt(0) + "  " + row.charAt(1) + "  " + row.charAt(2) + "  "
                    + row.charAt(3) + "  " + row.charAt(4) + "  " + row.charAt(5) + "  " + row.charAt(6) + "  "
                    + row.charAt(7) + "  ");
        }
        System.out.println("\n");
    }
```

6.69.3.16 transcribeToCharacters()

```
ArrayList<String> test.driver.MediumDifficultyDriver.transcribeToCharacters (
            Board board ) [private]
```

Definition at line 303 of file MediumDifficultyDriver.java.

```
                                                                      {
        ArrayList<String> boardCodified = new ArrayList<String>(8);
        String operational = "";
        PieceType[][] current = board.getBoard();

        for (int i = 0; i < 8; ++i) {
            operational = "";
            for (int j = 0; j < 8; ++j) {
                if (current[i][j] == PieceType.PLAYER1)
                    operational = operational + "B";
                if (current[i][j] == PieceType.PLAYER2)
                    operational = operational + "N";
                if (current[i][j] == null)
                    operational = operational + "?";

            }
            boardCodified.add(operational);
        }

        return boardCodified;
    }
```

6.69.4 Member Data Documentation

6.69.4.1 currentMediumDifficulty

```
MediumDifficulty test.driver.MediumDifficultyDriver.currentMediumDifficulty
```

Definition at line 24 of file MediumDifficultyDriver.java.

6.69.4.2 currentBoard

```
Board test.driver.MediumDifficultyDriver.currentBoard
```

Definition at line 26 of file MediumDifficultyDriver.java.

6.69.4.3 nameCurrentBoard

```
String test.driver.MediumDifficultyDriver.nameCurrentBoard
```

Definition at line 27 of file MediumDifficultyDriver.java.

6.69.4.4 fixtureRepository

```
FixtureRepository test.driver.MediumDifficultyDriver.fixtureRepository
```

Definition at line 29 of file MediumDifficultyDriver.java.

The documentation for this class was generated from the following file:

- MediumDifficultyDriver.java

6.70 view.ModifyInitialBoardView Class Reference

Public Member Functions

- ModifyInitialBoardView ()

  Class creator.
- void initialize ()

  Initialize method which is executed when the scene is shown.
- void transform (MouseEvent mouseEvent)

  Event method which is executed when a piece is clicked.
- void save () throws IOException

  Event method which is executed when the save button is clicked.

Private Member Functions

- void render ()

  Method executed everytime there is a change in the board.
- void drawPiece (Pair< Integer, Integer > pos, char pieceType)

  Painting method executed everytime there is a change in the board.
- Pair< Integer, Integer > getClickedPos (MouseEvent mouseEvent)

  Painting method executed everytime a player clicks on the board.
- Circle getCircle (Pair< Integer, Integer > pos)

  Method executed everytime there is a change in the board.

Private Attributes

- Circle f1c1

  Piece located in (1, 1).
- Circle f1c2

  Piece located in (1, 2).
- Circle f1c3

  Piece located in (1, 3).
- Circle f1c4

  Piece located in (1, 4).
- Circle f1c5

  Piece located in (1, 5).
- Circle f1c6

  Piece located in (1, 6).

- Circle f1c7

  Piece located in (1, 7).
- Circle f1c8

  Piece located in (1, 8).
- Circle f2c1

  Piece located in (2, 1).
- Circle f2c2

  Piece located in (2, 2).
- Circle f2c3

  Piece located in (2, 3).
- Circle f2c4

  Piece located in (2, 4).
- Circle f2c5

  Piece located in (2, 5).
- Circle f2c6

  Piece located in (2, 6).
- Circle f2c7

  Piece located in (2, 7).
- Circle f2c8

  Piece located in (2, 8).
- Circle f3c1

  Piece located in (3, 1).
- Circle f3c2

  Piece located in (3, 2).
- Circle f3c3

  Piece located in (3, 3).
- Circle f3c4

  Piece located in (3, 4).
- Circle f3c5

  Piece located in (3, 5).
- Circle f3c6

  Piece located in (3, 6).
- Circle f3c7

  Piece located in (3, 7).
- Circle f3c8

  Piece located in (3, 8).
- Circle f4c1

  Piece located in (4, 1).
- Circle f4c2

  Piece located in (4, 2).
- Circle f4c3

  Piece located in (4, 3).
- Circle f4c4

  Piece located in (4, 4).
- Circle f4c5

  Piece located in (4, 5).
- Circle f4c6

  Piece located in (4, 6).
- Circle f4c7

  Piece located in (4, 7).
- Circle f4c8

Piece located in (4, 8).
- Circle f5c1

  Piece located in (5, 1).
- Circle f5c2

  Piece located in (5, 2).
- Circle f5c3

  Piece located in (5, 3).
- Circle f5c4

  Piece located in (5, 4).
- Circle f5c5

  Piece located in (5, 5).
- Circle f5c6

  Piece located in (5, 6).
- Circle f5c7

  Piece located in (5, 7).
- Circle f5c8

  Piece located in (5, 8).
- Circle f6c1

  Piece located in (6, 1).
- Circle f6c2

  Piece located in (6, 2).
- Circle f6c3

  Piece located in (6, 3).
- Circle f6c4

  Piece located in (6, 4).
- Circle f6c5

  Piece located in (6, 5).
- Circle f6c6

  Piece located in (6, 6).
- Circle f6c7

  Piece located in (6, 7).
- Circle f6c8

  Piece located in (6, 8).
- Circle f7c1

  Piece located in (7, 1).
- Circle f7c2

  Piece located in (7, 2).
- Circle f7c3

  Piece located in (7, 3).
- Circle f7c4

  Piece located in (7, 4).
- Circle f7c5

  Piece located in (7, 5).
- Circle f7c6

  Piece located in (7, 6).
- Circle f7c7

  Piece located in (7, 7).
- Circle f7c8

  Piece located in (7, 8).
- Circle f8c1

  Piece located in (8, 1).

- Circle f8c2

  Piece located in (8, 2).
- Circle f8c3

  Piece located in (8, 3).
- Circle f8c4

  Piece located in (8, 4).
- Circle f8c5

  Piece located in (8, 5).
- Circle f8c6

  Piece located in (8, 6).
- Circle f8c7

  Piece located in (8, 7).
- Circle f8c8

  Piece located in (8, 8).
- Text save

  Save board button text.
- Rectangle saveButton

  Save board button.
- RadioButton placeWhitePieces

  White colour pieces selector.
- RadioButton placeBlackPieces

  Black colour pieces selector.
- RadioButton quitPieces

  Remove pieces selector.
- Label saveInitialBoardResult

  Exception message output.
- JSONObject board

  Current board.

6.70.1 Detailed Description

This class represents the scene controller of the Initial Board.

By Alex Rodriguez

Definition at line 28 of file ModifyInitialBoardView.java.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 ModifyInitialBoardView()

```
view.ModifyInitialBoardView.ModifyInitialBoardView ( )
```

Class creator.

Definition at line 34 of file ModifyInitialBoardView.java.

```
34                                          {
35     }
```

6.70.3 Member Function Documentation

6.70.3.1 initialize()

`void view.ModifyInitialBoardView.initialize ( )`

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The board is setted.

Definition at line 401 of file ModifyInitialBoardView.java.

```
                             {
        board = ViewCtrl.domainCtrl.viewBoard();
        render();
    }
```

6.70.3.2 transform()

```
void view.ModifyInitialBoardView.transform (
             MouseEvent mouseEvent )
```

Event method which is executed when a piece is clicked.

Precondition

True

Postcondition

The piece changes into white, black or is removed.

Definition at line 411 of file ModifyInitialBoardView.java.

```
                                                 {
        Pair<Integer, Integer> pos = getClickedPos(mouseEvent);
        if (placeWhitePieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos, "PLAYER1");
        else if (placeBlackPieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos,
    "PLAYER2");
        else if (quitPieces.isSelected()) board = ViewCtrl.domainCtrl.removePiece(pos);
        render();
    }
```

6.70.3.3 save()

```
void view.ModifyInitialBoardView.save ( ) throws IOException
```

Event method which is executed when the save button is clicked.

Precondition

True

Postcondition

The game is saved and user can close the game.

Definition at line 424 of file ModifyInitialBoardView.java.

```
424                                     {
425         Stage currentWindow = (Stage) save.getScene().getWindow();
426         currentWindow.close();
427     }
```

6.70.3.4 render()

```
void view.ModifyInitialBoardView.render ( )  [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

The change is setted in the board.

Definition at line 434 of file ModifyInitialBoardView.java.

```
434                           {
435         for (int i = 0; i < 8; i++) {
436             char[] row = board.getString(String.format("row%d", i)).toCharArray();
437             for (int j = 0; j < 8; j++) drawPiece(new Pair<Integer, Integer>(i, j), row[j]);
438         }
439     }
```

6.70.3.5 drawPiece()

```
void view.ModifyInitialBoardView.drawPiece (
            Pair< Integer, Integer > pos,
            char pieceType ) [private]
```

Painting method executed everytime there is a change in the board.

Precondition

True

Postcondition

Pieces change to the correct color.

Definition at line 446 of file ModifyInitialBoardView.java.

```
                                                              {
        Circle circle = getCircle(pos);
        switch (pieceType) {
            case 'B':
                circle.setFill(Color.web("0xFFFFFF", 1.0));
                break;
            case 'N':
                circle.setFill(Color.web("0x000000", 1.0));
                break;
            case '?':
                circle.setFill(Color.web("0x34d399", 1.0));
                break;
            default:
                break;
        }
    }
```

6.70.3.6 getClickedPos()

```
Pair<Integer, Integer> view.ModifyInitialBoardView.getClickedPos (
            MouseEvent mouseEvent ) [private]
```

Painting method executed everytime a player clicks on the board.

Precondition

True

Postcondition

The piece clicked is transformed into a pair.

Definition at line 468 of file ModifyInitialBoardView.java.

```
                                                               {
        Pair<Integer, Integer> pos = new Pair<Integer, Integer>(-1, -1);
        String piece = ((Circle) mouseEvent.getPickResult().getIntersectedNode()).getId();
        pos.first = Character.getNumericValue(piece.charAt(1)) - 1;
        pos.second = Character.getNumericValue(piece.charAt(3)) - 1;
        return pos;
    }
```

6.70.3.7 getCircle()

```
Circle view.ModifyInitialBoardView.getCircle (
            Pair< Integer, Integer > pos )  [private]
```

Method executed everytime there is a change in the board.

Precondition

True

Postcondition

Return the circle which belongs to the position.

Definition at line 481 of file ModifyInitialBoardView.java.

```
                                                   {
         try {
             Field field = this.getClass().getDeclaredField(String.format("f%sc%s", pos.first + 1,
      pos.second + 1));
             field.setAccessible(true);
             return (Circle) field.get(this);
         } catch (Exception e) {
             return new Circle();
         }
     }
```

6.70.4 Member Data Documentation

6.70.4.1 f1c1

```
Circle view.ModifyInitialBoardView.f1c1  [private]
```

Piece located in (1, 1).

Definition at line 43 of file ModifyInitialBoardView.java.

6.70.4.2 f1c2

```
Circle view.ModifyInitialBoardView.f1c2  [private]
```

Piece located in (1, 2).

Definition at line 48 of file ModifyInitialBoardView.java.

6.70.4.3 f1c3

```
Circle view.ModifyInitialBoardView.f1c3 [private]
```

Piece located in (1, 3).

Definition at line 53 of file ModifyInitialBoardView.java.

6.70.4.4 f1c4

```
Circle view.ModifyInitialBoardView.f1c4 [private]
```

Piece located in (1, 4).

Definition at line 58 of file ModifyInitialBoardView.java.

6.70.4.5 f1c5

```
Circle view.ModifyInitialBoardView.f1c5 [private]
```

Piece located in (1, 5).

Definition at line 63 of file ModifyInitialBoardView.java.

6.70.4.6 f1c6

```
Circle view.ModifyInitialBoardView.f1c6 [private]
```

Piece located in (1, 6).

Definition at line 68 of file ModifyInitialBoardView.java.

6.70.4.7 f1c7

```
Circle view.ModifyInitialBoardView.f1c7 [private]
```

Piece located in (1, 7).

Definition at line 73 of file ModifyInitialBoardView.java.

6.70.4.8 f1c8

```
Circle view.ModifyInitialBoardView.f1c8 [private]
```

Piece located in (1, 8).

Definition at line 78 of file ModifyInitialBoardView.java.

6.70.4.9 f2c1

```
Circle view.ModifyInitialBoardView.f2c1 [private]
```

Piece located in (2, 1).

Definition at line 83 of file ModifyInitialBoardView.java.

6.70.4.10 f2c2

```
Circle view.ModifyInitialBoardView.f2c2 [private]
```

Piece located in (2, 2).

Definition at line 88 of file ModifyInitialBoardView.java.

6.70.4.11 f2c3

```
Circle view.ModifyInitialBoardView.f2c3 [private]
```

Piece located in (2, 3).

Definition at line 93 of file ModifyInitialBoardView.java.

6.70.4.12 f2c4

```
Circle view.ModifyInitialBoardView.f2c4 [private]
```

Piece located in (2, 4).

Definition at line 98 of file ModifyInitialBoardView.java.

6.70.4.13 f2c5

```
Circle view.ModifyInitialBoardView.f2c5  [private]
```

Piece located in (2, 5).

Definition at line 103 of file ModifyInitialBoardView.java.

6.70.4.14 f2c6

```
Circle view.ModifyInitialBoardView.f2c6  [private]
```

Piece located in (2, 6).

Definition at line 108 of file ModifyInitialBoardView.java.

6.70.4.15 f2c7

```
Circle view.ModifyInitialBoardView.f2c7  [private]
```

Piece located in (2, 7).

Definition at line 113 of file ModifyInitialBoardView.java.

6.70.4.16 f2c8

```
Circle view.ModifyInitialBoardView.f2c8  [private]
```

Piece located in (2, 8).

Definition at line 118 of file ModifyInitialBoardView.java.

6.70.4.17 f3c1

```
Circle view.ModifyInitialBoardView.f3c1  [private]
```

Piece located in (3, 1).

Definition at line 123 of file ModifyInitialBoardView.java.

6.70.4.18 f3c2

```
Circle view.ModifyInitialBoardView.f3c2  [private]
```

Piece located in (3, 2).

Definition at line 128 of file ModifyInitialBoardView.java.

6.70.4.19 f3c3

```
Circle view.ModifyInitialBoardView.f3c3  [private]
```

Piece located in (3, 3).

Definition at line 133 of file ModifyInitialBoardView.java.

6.70.4.20 f3c4

```
Circle view.ModifyInitialBoardView.f3c4  [private]
```

Piece located in (3, 4).

Definition at line 138 of file ModifyInitialBoardView.java.

6.70.4.21 f3c5

```
Circle view.ModifyInitialBoardView.f3c5  [private]
```

Piece located in (3, 5).

Definition at line 143 of file ModifyInitialBoardView.java.

6.70.4.22 f3c6

```
Circle view.ModifyInitialBoardView.f3c6  [private]
```

Piece located in (3, 6).

Definition at line 148 of file ModifyInitialBoardView.java.

6.70.4.23 f3c7

```
Circle view.ModifyInitialBoardView.f3c7  [private]
```

Piece located in (3, 7).

Definition at line 153 of file ModifyInitialBoardView.java.

6.70.4.24 f3c8

```
Circle view.ModifyInitialBoardView.f3c8  [private]
```

Piece located in (3, 8).

Definition at line 158 of file ModifyInitialBoardView.java.

6.70.4.25 f4c1

```
Circle view.ModifyInitialBoardView.f4c1  [private]
```

Piece located in (4, 1).

Definition at line 163 of file ModifyInitialBoardView.java.

6.70.4.26 f4c2

```
Circle view.ModifyInitialBoardView.f4c2  [private]
```

Piece located in (4, 2).

Definition at line 168 of file ModifyInitialBoardView.java.

6.70.4.27 f4c3

```
Circle view.ModifyInitialBoardView.f4c3  [private]
```

Piece located in (4, 3).

Definition at line 173 of file ModifyInitialBoardView.java.

6.70.4.28 f4c4

```
Circle view.ModifyInitialBoardView.f4c4 [private]
```

Piece located in (4, 4).

Definition at line 178 of file ModifyInitialBoardView.java.

6.70.4.29 f4c5

```
Circle view.ModifyInitialBoardView.f4c5 [private]
```

Piece located in (4, 5).

Definition at line 183 of file ModifyInitialBoardView.java.

6.70.4.30 f4c6

```
Circle view.ModifyInitialBoardView.f4c6 [private]
```

Piece located in (4, 6).

Definition at line 188 of file ModifyInitialBoardView.java.

6.70.4.31 f4c7

```
Circle view.ModifyInitialBoardView.f4c7 [private]
```

Piece located in (4, 7).

Definition at line 193 of file ModifyInitialBoardView.java.

6.70.4.32 f4c8

```
Circle view.ModifyInitialBoardView.f4c8 [private]
```

Piece located in (4, 8).

Definition at line 198 of file ModifyInitialBoardView.java.

6.70.4.33 f5c1

```
Circle view.ModifyInitialBoardView.f5c1  [private]
```

Piece located in (5, 1).

Definition at line 203 of file ModifyInitialBoardView.java.

6.70.4.34 f5c2

```
Circle view.ModifyInitialBoardView.f5c2  [private]
```

Piece located in (5, 2).

Definition at line 208 of file ModifyInitialBoardView.java.

6.70.4.35 f5c3

```
Circle view.ModifyInitialBoardView.f5c3  [private]
```

Piece located in (5, 3).

Definition at line 213 of file ModifyInitialBoardView.java.

6.70.4.36 f5c4

```
Circle view.ModifyInitialBoardView.f5c4  [private]
```

Piece located in (5, 4).

Definition at line 218 of file ModifyInitialBoardView.java.

6.70.4.37 f5c5

```
Circle view.ModifyInitialBoardView.f5c5  [private]
```

Piece located in (5, 5).

Definition at line 223 of file ModifyInitialBoardView.java.

6.70.4.38 f5c6

```
Circle view.ModifyInitialBoardView.f5c6  [private]
```

Piece located in (5, 6).

Definition at line 228 of file ModifyInitialBoardView.java.

6.70.4.39 f5c7

```
Circle view.ModifyInitialBoardView.f5c7  [private]
```

Piece located in (5, 7).

Definition at line 233 of file ModifyInitialBoardView.java.

6.70.4.40 f5c8

```
Circle view.ModifyInitialBoardView.f5c8  [private]
```

Piece located in (5, 8).

Definition at line 238 of file ModifyInitialBoardView.java.

6.70.4.41 f6c1

```
Circle view.ModifyInitialBoardView.f6c1  [private]
```

Piece located in (6, 1).

Definition at line 243 of file ModifyInitialBoardView.java.

6.70.4.42 f6c2

```
Circle view.ModifyInitialBoardView.f6c2  [private]
```

Piece located in (6, 2).

Definition at line 248 of file ModifyInitialBoardView.java.

6.70.4.43 f6c3

```
Circle view.ModifyInitialBoardView.f6c3  [private]
```

Piece located in (6, 3).

Definition at line 253 of file ModifyInitialBoardView.java.

6.70.4.44 f6c4

```
Circle view.ModifyInitialBoardView.f6c4  [private]
```

Piece located in (6, 4).

Definition at line 258 of file ModifyInitialBoardView.java.

6.70.4.45 f6c5

```
Circle view.ModifyInitialBoardView.f6c5  [private]
```

Piece located in (6, 5).

Definition at line 263 of file ModifyInitialBoardView.java.

6.70.4.46 f6c6

```
Circle view.ModifyInitialBoardView.f6c6  [private]
```

Piece located in (6, 6).

Definition at line 268 of file ModifyInitialBoardView.java.

6.70.4.47 f6c7

```
Circle view.ModifyInitialBoardView.f6c7  [private]
```

Piece located in (6, 7).

Definition at line 273 of file ModifyInitialBoardView.java.

6.70.4.48 f6c8

```
Circle view.ModifyInitialBoardView.f6c8 [private]
```

Piece located in (6, 8).

Definition at line 278 of file ModifyInitialBoardView.java.

6.70.4.49 f7c1

```
Circle view.ModifyInitialBoardView.f7c1 [private]
```

Piece located in (7, 1).

Definition at line 283 of file ModifyInitialBoardView.java.

6.70.4.50 f7c2

```
Circle view.ModifyInitialBoardView.f7c2 [private]
```

Piece located in (7, 2).

Definition at line 288 of file ModifyInitialBoardView.java.

6.70.4.51 f7c3

```
Circle view.ModifyInitialBoardView.f7c3 [private]
```

Piece located in (7, 3).

Definition at line 293 of file ModifyInitialBoardView.java.

6.70.4.52 f7c4

```
Circle view.ModifyInitialBoardView.f7c4 [private]
```

Piece located in (7, 4).

Definition at line 298 of file ModifyInitialBoardView.java.

6.70.4.53 f7c5

```
Circle view.ModifyInitialBoardView.f7c5 [private]
```

Piece located in (7, 5).

Definition at line 303 of file ModifyInitialBoardView.java.

6.70.4.54 f7c6

```
Circle view.ModifyInitialBoardView.f7c6 [private]
```

Piece located in (7, 6).

Definition at line 308 of file ModifyInitialBoardView.java.

6.70.4.55 f7c7

```
Circle view.ModifyInitialBoardView.f7c7 [private]
```

Piece located in (7, 7).

Definition at line 313 of file ModifyInitialBoardView.java.

6.70.4.56 f7c8

```
Circle view.ModifyInitialBoardView.f7c8 [private]
```

Piece located in (7, 8).

Definition at line 318 of file ModifyInitialBoardView.java.

6.70.4.57 f8c1

```
Circle view.ModifyInitialBoardView.f8c1 [private]
```

Piece located in (8, 1).

Definition at line 323 of file ModifyInitialBoardView.java.

6.70.4.58 f8c2

```
Circle view.ModifyInitialBoardView.f8c2  [private]
```

Piece located in (8, 2).

Definition at line 328 of file ModifyInitialBoardView.java.

6.70.4.59 f8c3

```
Circle view.ModifyInitialBoardView.f8c3  [private]
```

Piece located in (8, 3).

Definition at line 333 of file ModifyInitialBoardView.java.

6.70.4.60 f8c4

```
Circle view.ModifyInitialBoardView.f8c4  [private]
```

Piece located in (8, 4).

Definition at line 338 of file ModifyInitialBoardView.java.

6.70.4.61 f8c5

```
Circle view.ModifyInitialBoardView.f8c5  [private]
```

Piece located in (8, 5).

Definition at line 343 of file ModifyInitialBoardView.java.

6.70.4.62 f8c6

```
Circle view.ModifyInitialBoardView.f8c6  [private]
```

Piece located in (8, 6).

Definition at line 348 of file ModifyInitialBoardView.java.

6.70.4.63 f8c7

```
Circle view.ModifyInitialBoardView.f8c7  [private]
```

Piece located in (8, 7).

Definition at line 353 of file ModifyInitialBoardView.java.

6.70.4.64 f8c8

```
Circle view.ModifyInitialBoardView.f8c8  [private]
```

Piece located in (8, 8).

Definition at line 358 of file ModifyInitialBoardView.java.

6.70.4.65 save

```
Text view.ModifyInitialBoardView.save  [private]
```

Save board button text.

Definition at line 363 of file ModifyInitialBoardView.java.

6.70.4.66 saveButton

```
Rectangle view.ModifyInitialBoardView.saveButton  [private]
```

Save board button.

Definition at line 368 of file ModifyInitialBoardView.java.

6.70.4.67 placeWhitePieces

```
RadioButton view.ModifyInitialBoardView.placeWhitePieces  [private]
```

White colour pieces selector.

Definition at line 373 of file ModifyInitialBoardView.java.

6.70.4.68 placeBlackPieces

```
RadioButton view.ModifyInitialBoardView.placeBlackPieces  [private]
```

Black colour pieces selector.

Definition at line 378 of file ModifyInitialBoardView.java.

6.70.4.69 quitPieces

```
RadioButton view.ModifyInitialBoardView.quitPieces  [private]
```

Remove pieces selector.

Definition at line 383 of file ModifyInitialBoardView.java.

6.70.4.70 saveInitialBoardResult

```
Label view.ModifyInitialBoardView.saveInitialBoardResult  [private]
```

Exception message output.

Definition at line 388 of file ModifyInitialBoardView.java.

6.70.4.71 board

```
JSONObject view.ModifyInitialBoardView.board  [private]
```

Current board.

Definition at line 392 of file ModifyInitialBoardView.java.

The documentation for this class was generated from the following file:

- ModifyInitialBoardView.java

6.71 domain.Exceptions.NotCreatorException Class Reference

The user that tries to perform an action on a object is not the creator of it. By Alex Rodriguez.

Public Member Functions

- NotCreatorException ()

6.71.1 Detailed Description

The user that tries to perform an action on a object is not the creator of it. By Alex Rodriguez.

Definition at line 96 of file Exceptions.java.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 NotCreatorException()

```
domain.Exceptions.NotCreatorException.NotCreatorException ( )
```

Definition at line 97 of file Exceptions.java.

```
                                {
            super("ERR_NOT_CREATOR");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.72 domain.Exceptions.NotPlayerException Class Reference

The player that wants to perform an action is not part of the game. By Alex Rodriguez.

Public Member Functions

- NotPlayerException ()

6.72.1 Detailed Description

The player that wants to perform an action is not part of the game. By Alex Rodriguez.

Definition at line 206 of file Exceptions.java.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 NotPlayerException()

```
domain.Exceptions.NotPlayerException.NotPlayerException ( )
```

Definition at line 207 of file Exceptions.java.

```
                                    {
            super("ERR_NOT_PLAYER");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.73 domain.Exceptions.NotPlayerPieceException Class Reference

The player that wants to perform an action tries to use an opponent piece. By Alex Rodriguez.

Public Member Functions

- NotPlayerPieceException ()

6.73.1 Detailed Description

The player that wants to perform an action tries to use an opponent piece. By Alex Rodriguez.

Definition at line 217 of file Exceptions.java.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 NotPlayerPieceException()

```
domain.Exceptions.NotPlayerPieceException.NotPlayerPieceException ( )
```

Definition at line 218 of file Exceptions.java.

```
                                         {
            super("ERR_NOT_PLAYER_PIECE");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.74 domain.Exceptions.NotPlayerTurnException Class Reference

It is not the turn of the player that wants to perform an action. By Alex Rodriguez.

Public Member Functions

- NotPlayerTurnException ()

6.74.1 Detailed Description

It is not the turn of the player that wants to perform an action. By Alex Rodriguez.

Definition at line 228 of file Exceptions.java.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 NotPlayerTurnException()

```
domain.Exceptions.NotPlayerTurnException.NotPlayerTurnException ( )
```

Definition at line 229 of file Exceptions.java.

```
                                         {
                super("ERR_NOT_PLAYER_TURN");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.75 domain.Exceptions.NotStartedGameException Class Reference

The game has not yet started. By Alex Rodriguez.

Public Member Functions

- NotStartedGameException ()

6.75.1 Detailed Description

The game has not yet started. By Alex Rodriguez.

Definition at line 250 of file Exceptions.java.

6.75.2 Constructor & Destructor Documentation

6.75.2.1 NotStartedGameException()

```
domain.Exceptions.NotStartedGameException.NotStartedGameException ( )
```

Definition at line 251 of file Exceptions.java.

```
                                         {
            super("ERR_NOT_STARTED_GAME");
        }
```

The documentation for this class was generated from the following file:

- Exceptions.java

6.76 cmd.othello Class Reference

Othello application entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)

  Othello application main function. Creates an instance of the othello application and starts it.

6.76.1 Detailed Description

Othello application entrypoint. By Alex Rodriguez.

Definition at line 50 of file othello.java.

6.76.2 Member Function Documentation

6.76.2.1 main()

```
static void cmd.othello.main (
              String[ ] args )  [static]
```

Othello application main function. Creates an instance of the othello application and starts it.

Precondition

True.

Postcondition

The Othello application has started.

Definition at line 57 of file othello.java.

```
                                             {
        ViewCtrl.main(args);
    }
```

The documentation for this class was generated from the following file:

- othello.java

6.77 cmd.driver.pair Class Reference

Pair driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)

  Pair driver main function. Creates an instance of the Pair driver and starts it.

6.77.1 Detailed Description

Pair driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file pair.java.

6.77.2 Member Function Documentation

6.77.2.1 main()

```
static void cmd.driver.pair.main (
             String[ ] args )  [static]
```

Pair driver main function. Creates an instance of the Pair driver and starts it.

Precondition

True.

Postcondition

The Pair driver has started.

Definition at line 22 of file pair.java.

```
                                        {
           new PairDriver().start();
       }
```

The documentation for this class was generated from the following file:

- pair.java

6.78 util.Pair$<$ F, S $>$ Class Template Reference

Implements a data structure containing two generic types. By Alex Rodriguez.

Public Member Functions

- Pair (F first, S second)

    Create a Pair instance.
- boolean equals (Object object)

    Compare equality of the implicit Pair and another.
- String toString ()

    Get the String representation of the implicit Pair.
- F getFirst ()

    Get the First value of the implicit Pair.
- S getSecond ()

    Get the Second value of the implicit Pair.

Public Attributes

- F first

    First value of the Pair.
- S second

    Second value of the Pair.

6.78.1 Detailed Description

Implements a data structure containing two generic types. By Alex Rodriguez.

Definition at line 15 of file Pair.java.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 Pair()

```
util.Pair< F, S >.Pair (
            F first,
            S second )
```

Create a Pair instance.

Precondition

True

Postcondition

A Pair instance is created and its implicits first and second attributes are setted.

Parameters

first	First value of the Pair.
second	Second value of the Pair.

Definition at line 36 of file Pair.java.

```
                                 {
        this.first = first;
        this.second = second;
    }
```

6.78.3 Member Function Documentation

6.78.3.1 equals()

```
boolean util.Pair< F, S >.equals (
            Object object )
```

Compare equality of the implicit Pair and another.

Precondition

True

Postcondition

It is returned True if the implicit Pair is equal to the given Pair or False if not.

Parameters

object	Pair to be compared.

Returns

Whether the implicit Pair and the given Pair are equal.

Definition at line 51 of file Pair.java.

```
                                      {
        if (!(object instanceof Pair)) {
            return false;
        }
        Pair<?, ?> other = (Pair<?, ?>) object;
        return Objects.equals(other.first, this.first) && Objects.equals(other.second, this.second);
    }
```

6.78.3.2 toString()

```
String util.Pair< F, S >.toString ( )
```

Get the String representation of the implicit Pair.

Precondition

True

Postcondition

An String representing the implicit Pair is returned.

Returns

String representation of the implicit Pair.

Definition at line 66 of file Pair.java.

```
                          {
        return String.format("Pair<%s, %s>", this.first, this.second);
    }
```

6.78.3.3 getFirst()

`F util.Pair< F, S >.getFirst ( )`

Get the First value of the implicit Pair.

Precondition

True

Postcondition

The First value of the implicit Pair is returned.

Returns

First value of the implicit Pair.

Definition at line 76 of file Pair.java.

```
                      {
        return this.first;
    }
```

6.78.3.4 getSecond()

`S util.Pair< F, S >.getSecond ( )`

Get the Second value of the implicit Pair.

Precondition

True

Postcondition

The Second value of the implicit Pair is returned.

Returns

Second value of the implicit Pair.

Definition at line 86 of file Pair.java.

```
                       {
        return this.second;
    }
```

6.78.4 Member Data Documentation

6.78.4.1 first

```
F util.Pair< F, S >.first
```

First value of the Pair.

Definition at line 21 of file Pair.java.

6.78.4.2 second

```
S util.Pair< F, S >.second
```

Second value of the Pair.

Definition at line 25 of file Pair.java.

The documentation for this class was generated from the following file:

- Pair.java

6.79 test.driver.PairDriver Class Reference

Implements the different options for the Pair driver application. By Alex Rodriguez.

Public Member Functions

- PairDriver ()
- void start ()

Public Attributes

- Pair< String, String > currentStrPair
- Pair< Integer, Integer > currentIntPair
- Pair< String, Integer > currentStrIntPair

Private Member Functions

- void mainMenu ()
- Object currentPair ()
- void resetPairs ()
- void createStrPair ()
- void createIntPair ()
- void createStrIntPair ()
- void getFirst ()
- void getSecond ()
- void comparePair ()

Additional Inherited Members

6.79.1 Detailed Description

Implements the different options for the Pair driver application. By Alex Rodriguez.

Definition at line 15 of file PairDriver.java.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 PairDriver()

```
test.driver.PairDriver.PairDriver ( )
```

Definition at line 24 of file PairDriver.java.

```
                         {
        this.currentStrPair = null;
        this.currentIntPair = null;
        this.currentStrIntPair = null;
    }
```

6.79.3 Member Function Documentation

6.79.3.1 start()

```
void test.driver.PairDriver.start ( )
```

Definition at line 32 of file PairDriver.java.

```
                         {
        while (true) {
            this.mainMenu();
        }
    }
```

6.79.3.2 mainMenu()

void test.driver.PairDriver.mainMenu ( ) [private]

Definition at line 38 of file PairDriver.java.

```
                           {
        String title = (this.currentPair() != null ? String.format("Current: %s\n", this.currentPair()) :
    null);
        switch (Driver.menu(title, "Pair Driver",
                new Pair<String, String>("1", "Create String:String Pair"),
                new Pair<String, String>("2", "Create Integer:Integer Pair"),
                new Pair<String, String>("3", "Create String:Integer Pair"),
                new Pair<String, String>("4", "Get first"),
                new Pair<String, String>("5", "Get second"),
                new Pair<String, String>("6", "Compare current Pair with another"))) {
        case "1":
            this.createStrPair();
            break;
        case "2":
            this.createIntPair();
            break;
        case "3":
            this.createStrIntPair();
            break;
        case "4":
            this.getFirst();
            break;
        case "5":
            this.getSecond();
            break;
        case "6":
            this.comparePair();
            break;
        }
        Driver.pause();
    }
```

6.79.3.3 currentPair()

Object test.driver.PairDriver.currentPair ( ) [private]

Definition at line 69 of file PairDriver.java.

```
                                {
        if (this.currentStrPair != null)
            return this.currentStrPair;

        if (this.currentIntPair != null)
            return this.currentIntPair;

        if (this.currentStrIntPair != null)
            return this.currentStrIntPair;

        return null;
    }
```

6.79.3.4 resetPairs()

void test.driver.PairDriver.resetPairs ( ) [private]

Definition at line 82 of file PairDriver.java.

```
                             {
        this.currentStrPair = null;
        this.currentIntPair = null;
        this.currentStrIntPair = null;
    }
```

6.79.3.5 createStrPair()

```
void test.driver.PairDriver.createStrPair ( ) [private]
```

Definition at line 88 of file PairDriver.java.

```
                              {
        this.resetPairs();
        this.currentStrPair = new Pair<String, String>(Driver.input("First value?"), Driver.input("Second
      value?"));
        System.out.println(String.format("%s created successfully!", this.currentStrPair));
    }
```

6.79.3.6 createIntPair()

```
void test.driver.PairDriver.createIntPair ( ) [private]
```

Definition at line 94 of file PairDriver.java.

```
                              {
        this.resetPairs();
        this.currentIntPair = new Pair<Integer, Integer>(Driver.inputInt("First value?"),
                Driver.inputInt("Second value?"));
        System.out.println(String.format("%s created successfully!", this.currentIntPair));
    }
```

6.79.3.7 createStrIntPair()

```
void test.driver.PairDriver.createStrIntPair ( ) [private]
```

Definition at line 101 of file PairDriver.java.

```
                                 {
        this.resetPairs();
        this.currentStrIntPair = new Pair<String, Integer>(Driver.input("First value?"),
                Driver.inputInt("Second value?"));
        System.out.println(String.format("%s created successfully!", this.currentStrIntPair));
    }
```

6.79.3.8 getFirst()

```
void test.driver.PairDriver.getFirst ( ) [private]
```

Definition at line 108 of file PairDriver.java.

```
                          {
        if (this.currentPair() == null) {
            System.out.println("No current Pair!");
            return;
        }

        System.out.print(String.format("%s's first is: ", this.currentPair()));

        if (this.currentStrPair != null)
            System.out.println(this.currentStrPair.first);

        if (this.currentIntPair != null)
            System.out.println(this.currentIntPair.first);

        if (this.currentStrIntPair != null)
            System.out.println(this.currentStrIntPair.first);
    }
```

6.79.3.9 getSecond()

`void test.driver.PairDriver.getSecond ( ) [private]`

Definition at line 126 of file PairDriver.java.

```
                            {
        if (this.currentPair() == null) {
            System.out.println("No current Pair!");
            return;
        }

        System.out.print(String.format("%s’s second is: ", this.currentPair()));

        if (this.currentStrPair != null)
            System.out.println(this.currentStrPair.second);

        if (this.currentIntPair != null)
            System.out.println(this.currentIntPair.second);

        if (this.currentStrIntPair != null)
            System.out.println(this.currentStrIntPair.second);
    }
```

6.79.3.10 comparePair()

`void test.driver.PairDriver.comparePair ( ) [private]`

Definition at line 144 of file PairDriver.java.

```
                              {
        if (this.currentPair() == null) {
            System.out.println("No current Pair!");
            return;
        }

        if (this.currentStrPair != null) {
            Pair<String, String> toCompare = new Pair<String, String>(Driver.input("First value of Pair to compare?"),
                    Driver.input("Second value of Pair to compare?"));
            if (this.currentStrPair.equals(toCompare))
                System.out.println(String.format("%s and %s are equal", this.currentStrPair, toCompare));
            else
                System.out.println(String.format("%s and %s are not equal", this.currentStrPair, toCompare));
        }

        if (this.currentIntPair != null) {
            Pair<Integer, Integer> toCompare = new Pair<Integer, Integer>(
                    Driver.inputInt("First value of Pair to compare?"),
                    Driver.inputInt("Second value of Pair to compare?"));
            if (this.currentIntPair.equals(toCompare))
                System.out.println(String.format("%s and %s are equal", this.currentIntPair, toCompare));
            else
                System.out.println(String.format("%s and %s are not equal", this.currentIntPair, toCompare));
        }

        if (this.currentStrIntPair != null) {
            Pair<String, Integer> toCompare = new Pair<String, Integer>(Driver.input("First value of Pair to compare?"),
                    Driver.inputInt("Second value of Pair to compare?"));
            if (this.currentStrIntPair.equals(toCompare))
                System.out.println(String.format("%s and %s are equal", this.currentStrIntPair, toCompare));
            else
                System.out.println(String.format("%s and %s are not equal", this.currentStrIntPair, toCompare));
        }
    }
```

6.79.4 Member Data Documentation

6.79.4.1 currentStrPair

```
Pair<String, String> test.driver.PairDriver.currentStrPair
```

Definition at line 18 of file PairDriver.java.

6.79.4.2 currentIntPair

```
Pair<Integer, Integer> test.driver.PairDriver.currentIntPair
```

Definition at line 19 of file PairDriver.java.

6.79.4.3 currentStrIntPair

```
Pair<String, Integer> test.driver.PairDriver.currentStrIntPair
```

Definition at line 20 of file PairDriver.java.

The documentation for this class was generated from the following file:

- PairDriver.java

6.80 domain.Board.PieceType Enum Reference

The status of a cell of the Board. An Othello Board is composed of 64 cells with their own unique position and three possible states:

Public Attributes

- PLAYER1
- PLAYER2

6.80.1 Detailed Description

The status of a cell of the Board. An Othello Board is composed of 64 cells with their own unique position and three possible states:

1. PLAYER1 -> PLAYER1 has a piece on that cell.
2. PLAYER2 -> PLAYER2 has a piece on that cell.
3. null -> empty cell (nobody has a piece on that cell).

Definition at line 28 of file Board.java.

6.80.2 Member Data Documentation

6.80.2.1 PLAYER1

```
domain.Board.PieceType.PLAYER1
```

Definition at line 28 of file Board.java.

6.80.2.2 PLAYER2

```
domain.Board.PieceType.PLAYER2
```

Definition at line 28 of file Board.java.

The documentation for this enum was generated from the following file:

- Board.java

6.81 domain.Player Class Reference

Represents a player in our system.

Public Member Functions

- String getName ()
  Consultant that returns the implicit parameter's name.
- UUID getID ()
  Consultant that returns the implicit parameter's ID.
- boolean getIsDeleted ()
  Consultant that returns the implicit parameter's isDeleted value.
- void setName (String name) throws InvalidNameException
  Modifier that, given a name, changes the implicit parameter's name for a new name 'name'.
- void setIsDeleted (boolean isDeleted)
  Modifier that, given an isDeleted value, changes the implicit parameter's state for a new state 'isDeleted'.

Protected Attributes

- UUID id
  Player's ID.
- String name
  Player's name.
- boolean isDeleted
  Player's state.

6.81.1 Detailed Description

Represents a player in our system.

Done by Arnau Pujantell

Class that represents a player. It contains an id, a name, a type and an isDeleted.

Definition at line 18 of file Player.java.

6.81.2 Member Function Documentation

6.81.2.1 getName()

```
String domain.Player.getName ( )
```

Consultant that returns the implicit parameter's name.

CONSULTANTS

Precondition

True

Postcondition

The implicit parameter's name is returned

Definition at line 32 of file Player.java.

```
                              {
          return this.name;
      }
```

6.81.2.2 getID()

```
UUID domain.Player.getID ( )
```

Consultant that returns the implicit parameter's ID.

Precondition

True

Postcondition

The implicit parameter's ID is returned.

Definition at line 40 of file Player.java.

```
                          {
          return this.id;
      }
```

6.81.2.3 getIsDeleted()

```
boolean domain.Player.getIsDeleted ( )
```

Consultant that returns the implicit parameter's isDeleted value.

Precondition

True

Postcondition

The implicit parameter's isDeleted value is returned.

Returns

Definition at line 49 of file Player.java.

```
                                 {
        return this.isDeleted;
    }
```

6.81.2.4 setName()

```
void domain.Player.setName (
             String name ) throws InvalidNameException
```

Modifier that, given a name, changes the implicit parameter's name for a new name 'name'.

MODIFIERS

Precondition

Name is not null

Postcondition

Implicit parameter's name has been changed.

Definition at line 60 of file Player.java.

```
                                                              {
        if(name.isBlank()) {
            throw new InvalidNameException();
        }
        this.name = name;
    }
```

6.81.2.5 setIsDeleted()

```
void domain.Player.setIsDeleted (
             boolean isDeleted )
```

Modifier that, given an isDeleted value, changes the implicit parameter's state for a new state 'isDeleted'.

Precondition

isDeleted is not null

Postcondition

Implicit parameter's state has been changed.

Definition at line 71 of file Player.java.

```
                                          {
         this.isDeleted = isDeleted;
     }
```

6.81.3 Member Data Documentation

6.81.3.1 id

```
UUID domain.Player.id  [protected]
```

Player's ID.

Definition at line 20 of file Player.java.

6.81.3.2 name

```
String domain.Player.name  [protected]
```

Player's name.

Definition at line 22 of file Player.java.

6.81.3.3 isDeleted

```
boolean domain.Player.isDeleted  [protected]
```

Player's state.

Definition at line 24 of file Player.java.

The documentation for this class was generated from the following file:

- Player.java

6.82 domain.PlayerCtrl Class Reference

Player class controller.

Public Member Functions

- PlayerCtrl ()

  Creator method that creates an instance of Player Control.
- User createUser (String name, String password, String confirmation) throws InvalidNameException, Invalid↩ PasswordException, ExistingPlayerException, BadConfirmationException

  Creator that, given a name and a password, creates a new user in the repository.
- Bot createBot (String name, Integer difficulty, UUID creatorID) throws InvalidNameException, Invalid↩ DifficultyException, ExistingPlayerException

  Method that, given a name, a difficulty and an ID, creates a new bot in the repository.
- User login (String name, String password) throws InvalidNameException, InvalidPasswordException, InexistingPlayerException, IncorrectCredentialsException

  Method that, given a name and a password, allows us to log in the system.
- User getUser (UUID userID) throws InexistingPlayerException

  Method that, given an ID, returns a user.
- Bot getBot (UUID botID) throws InexistingPlayerException

  Method that, given an ID, returns a bot.
- ArrayList< Pair< String, UUID > > listUsers ()

  Method that lists all users from repository.
- ArrayList< Pair< String, UUID > > listBots ()

  Method that lists all bots from repository.
- User modifyUser (UUID userID, String name, String password, String confirmation) throws Invalid↩ NameException, InvalidPasswordException, InexistingPlayerException, ExistingPlayerException, Bad↩ ConfirmationException

  Modifier that, given an ID, a name and a password, allows us to modify the user's credentials changing the name, the password or both.
- Bot modifyBot (UUID botID, String name, Integer difficulty, UUID modifierID) throws InvalidNameException, InvalidDifficultyException, ExistingPlayerException, InexistingPlayerException, BotUsedException, Not↩ CreatorException

  Method that, given an ID, a name, a difficulty and an ID, allows us to modify the bot changing the name, the difficulty or both.
- void deleteUser (UUID userID, String password) throws IncorrectCredentialsException, InexistingPlayer↩ Exception

  Method that, given an ID, a name and a password, allows us to delete a user.
- void deleteBot (UUID botID, UUID deleterID) throws NotCreatorException, InexistingPlayerException, Bot↩ UsedException

  Method that, given a name, a botID and a deleterID, allows us to delete a bot.

Private Member Functions

- User saveUser (String name, String password, UUID id) throws InvalidNameException, InvalidPassword↩ Exception

  Method that, given a name and a password, allows us to save a user in the repository.
- Bot saveBot (String name, Integer difficulty, UUID id, UUID creatorID) throws InvalidNameException, Invalid↩ DifficultyException

  Method that, given a name, a difficulty and an ID, allows us to save a bot in repository.
- String hash (String text)

  Method that, given a password, it hashes it using SHA-256.

Private Attributes

- PlayerRepositoryCtrl repositoryCtrl

  Instance of the Player Repository.
- GameRepositoryCtrl gameRepositoryCtrl

  Instance of the Game Repository.

6.82.1 Detailed Description

Player class controller.

```
Done by Manuel Navid

It contains the necessary functions to obtain the information that
needs to send to the presentation layer.
```

See also

domain.Player

Definition at line 38 of file PlayerCtrl.java.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 PlayerCtrl()

```
domain.PlayerCtrl.PlayerCtrl ( )
```

Creator method that creates an instance of Player Control.

Precondition

True

Postcondition

Creates an instance of PlayerCtrl

Definition at line 58 of file PlayerCtrl.java.

```
                           {
        this.repositoryCtrl = new PlayerRepositoryCtrl();
        this.gameRepositoryCtrl = new GameRepositoryCtrl();
    }
```

6.82.3 Member Function Documentation

6.82.3.1 createUser()

```
User domain.PlayerCtrl.createUser (
             String name,
             String password,
             String confirmation ) throws InvalidNameException, InvalidPasswordException,
ExistingPlayerException, BadConfirmationException
```

Creator that, given a name and a password, creates a new user in the repository.

CREATORS

Precondition

True

Parameters

name	Name of a User
password	Password of a User
confirmation	Confirmation of the entered password

Postcondition

saveUser is called and a saved new user is returned.

Definition at line 77 of file PlayerCtrl.java.

```
                {
        if (!password.equals(confirmation))
            throw new BadConfirmationException();
        if (this.repositoryCtrl.getByName(name) != null)
            throw new ExistingPlayerException();
        return this.saveUser(name, hash(password), UUID.randomUUID());
    }
```

6.82.3.2 createBot()

```
Bot domain.PlayerCtrl.createBot (
             String name,
             Integer difficulty,
             UUID creatorID ) throws InvalidNameException, InvalidDifficultyException, ExistingPlayerException
```

Method that, given a name, a difficulty and an ID, creates a new bot in the repository.

Precondition

True

Parameters

name	Name of the Bot
difficulty	Difficulty of the Bot
creatorID	UUID of a Player

Postcondition

saveBot is called and a saved new bot is returned.

Definition at line 96 of file PlayerCtrl.java.

```
                                                                                         {
        if (this.repositoryCtrl.getByName(name) != null)
            throw new ExistingPlayerException();
        return this.saveBot(name, difficulty, UUID.randomUUID(), creatorID);
    }
```

6.82.3.3 saveUser()

```
User domain.PlayerCtrl.saveUser (
            String name,
            String password,
            UUID id ) throws InvalidNameException, InvalidPasswordException  [private]
```

Method that, given a name and a password, allows us to save a user in the repository.

Precondition

True

Parameters

name	Name of a User
password	Password of a User
id	UUID of a User

Postcondition

User is saved in the users' list at repository and returned.

Definition at line 113 of file PlayerCtrl.java.

```
                   {
        if (name.isBlank())
            throw new InvalidNameException();
        if (password.isBlank())
            throw new InvalidPasswordException();
        User user = new User(name, password, id);
        this.repositoryCtrl.save(user.serialize());
        return user;
    }
```

6.82.3.4 saveBot()

```
Bot domain.PlayerCtrl.saveBot (
            String name,
            Integer difficulty,
            UUID id,
            UUID creatorID ) throws InvalidNameException, InvalidDifficultyException  [private]
```

Method that, given a name, a difficulty and an ID, allows us to save a bot in repository.

Precondition

True

Parameters

name	Name of the Bot
difficulty	Difficulty of the Bot
id	UUID of the Bot
creatorID	UUID of a Player

Postcondition

Bot is saved in the bots' list and returned.

Definition at line 134 of file PlayerCtrl.java.

```
                                                                                {
        if (name.isBlank())
            throw new InvalidNameException();
        if (difficulty < 1 || difficulty > 10)
            throw new InvalidDifficultyException();
        Bot bot = new Bot(name, difficulty, id, creatorID);
        this.repositoryCtrl.save(bot.serialize());
        return bot;
    }
```

6.82.3.5 login()

```
User domain.PlayerCtrl.login (
              String name,
              String password ) throws InvalidNameException, InvalidPasswordException, InexistingPlayerExceptio
IncorrectCredentialsException
```

Method that, given a name and a password, allows us to log in the system.

Precondition

True

Parameters

name	Name of a User
password	Password of a User

Postcondition

The user found in the repository is returned.

Definition at line 153 of file PlayerCtrl.java.

```
                                                                                 {
        if (name.isBlank())
            throw new InvalidNameException();
        if (password.isBlank())
```

```
            throw new InvalidPasswordException();

        JSONObject rawUser = this.repositoryCtrl.getByName(name);
        if (rawUser == null)
            throw new InexistingPlayerException();
        if (rawUser.getString("type").equals("BOT"))
            throw new InexistingPlayerException();

        User user = new User(rawUser);
        if (user.getIsDeleted())
            throw new InexistingPlayerException();

        if (!user.getPassword().equals(hash(password)))
            throw new IncorrectCredentialsException();

        return user;
    }
```

6.82.3.6 getUser()

```
User domain.PlayerCtrl.getUser (
                UUID userID ) throws InexistingPlayerException
```

Method that, given an ID, returns a user.

CONSULTANTS

Precondition

userID is not null

Parameters

userID	UUID of a User

Postcondition

User is found in repository and returned.

Definition at line 185 of file PlayerCtrl.java.

```
                                                                {
        JSONObject user = this.repositoryCtrl.get(userID);
        if (user == null)
            throw new InexistingPlayerException();
        if (user.getString("type").equals("BOT"))
            throw new InexistingPlayerException();
        return new User(user);
    }
```

6.82.3.7 getBot()

```
Bot domain.PlayerCtrl.getBot (
                UUID botID ) throws InexistingPlayerException
```

Method that, given an ID, returns a bot.

Precondition

botID is not null

Parameters

botID	UUID of the Bot

Postcondition

Bot is found in repository and returned.

Definition at line 201 of file PlayerCtrl.java.

```
                                                                        {
        JSONObject bot = this.repositoryCtrl.get(botID);
        if (bot == null)
            throw new InexistingPlayerException();
        if (bot.getString("type").equals("USER"))
            throw new InexistingPlayerException();
        return new Bot(bot);
    }
```

6.82.3.8 listUsers()

`ArrayList<Pair<String, UUID> > domain.PlayerCtrl.listUsers ( )`

Method that lists all users from repository.

Precondition

True

Postcondition

All users are listed.

Definition at line 216 of file PlayerCtrl.java.

```
                                                         {
        return this.repositoryCtrl.listUsers();
    }
```

6.82.3.9 listBots()

`ArrayList<Pair<String, UUID> > domain.PlayerCtrl.listBots ( )`

Method that lists all bots from repository.

Precondition

True

Postcondition

All bots are listed.

Definition at line 226 of file PlayerCtrl.java.

```
                                                        {
        return this.repositoryCtrl.listBots();
    }
```

6.82.3.10 modifyUser()

```
User domain.PlayerCtrl.modifyUser (
            UUID userID,
            String name,
            String password,
            String confirmation ) throws InvalidNameException, InvalidPasswordException,
InexistingPlayerException, ExistingPlayerException, BadConfirmationException
```

Modifier that, given an ID, a name and a password, allows us to modify the user's credentials changing the name, the password or both.

MODIFIERS

Precondition

True

Parameters

userid	UUID of a User
name	Name of a User
password	Password of User
confirmation	Confirmation of the entered password

Postcondition

Name, password or both are changed and modified user is returned.

Definition at line 243 of file PlayerCtrl.java.

```
                                                    {
        User original = this.getUser(userID);

        if (name != null) {
            if (name.isBlank())
                throw new InvalidNameException();
            if (!original.getName().equals(name) && this.repositoryCtrl.getByName(name) != null)
                throw new ExistingPlayerException();
            original.setName(name);
        }

        if (password != null) {
            if (password.isBlank())
                throw new InvalidPasswordException();
            if (!password.equals(confirmation))
                throw new BadConfirmationException();
            original.setPassword(hash(password));
        }

        return this.saveUser(original.getName(), original.getPassword(), original.getID());
    }
```

6.82.3.11 modifyBot()

```
Bot domain.PlayerCtrl.modifyBot (
            UUID botID,
            String name,
            Integer difficulty,
            UUID modifierID ) throws InvalidNameException, InvalidDifficultyException, ExistingPlayerExceptio
InexistingPlayerException, BotUsedException, NotCreatorException
```

Method that, given an ID, a name, a difficulty and an ID, allows us to modify the bot changing the name, the difficulty or both.

Precondition

True

Parameters

name	Name of the Bot
difficulty	Difficulty of the Bot
botID	UUID of the Bot
modifierID	UUID of a Player

Postcondition

Bot's name, difficulty or both are modified and modified bot is returned.

Definition at line 278 of file PlayerCtrl.java.

```
                                              {
        Bot original = this.getBot(botID);

        if (!original.getCreatorID().equals(modifierID))
            throw new NotCreatorException();

        if (name != null) {
            if (name.isBlank())
                throw new InvalidNameException();
            if (!original.getName().equals(name) && this.repositoryCtrl.getByName(name) != null)
                throw new ExistingPlayerException();
            original.setName(name);
        }

        if (difficulty != null) {
            if (difficulty < 1 || difficulty > 10)
                throw new InvalidDifficultyException();
            original.setDifficulty(difficulty);
        }

        if (this.gameRepositoryCtrl.existsGameByPlayerID(botID))
            throw new BotUsedException();

        return this.saveBot(original.getName(), original.getDifficulty(), original.getID(), modifierID);
    }
```

6.82.3.12 deleteUser()

```
void domain.PlayerCtrl.deleteUser (
            UUID userID,
            String password ) throws IncorrectCredentialsException, InexistingPlayerException
```

Method that, given an ID, a name and a password, allows us to delete a user.

DELETERS

Precondition

True

Parameters

userID	UUID of a User
password	Passowrd of a User

Postcondition

The user is deleted from the repository.

Definition at line 317 of file PlayerCtrl.java.

```
                                                                                    {
        User user = this.getUser(userID);

        if (!user.getPassword().equals(hash(password)))
            throw new IncorrectCredentialsException();

        this.repositoryCtrl.delete(userID);
    }
```

6.82.3.13 deleteBot()

```
void domain.PlayerCtrl.deleteBot (
              UUID botID,
              UUID deleterID ) throws NotCreatorException, InexistingPlayerException, BotUsedException
```

Method that, given a name, a botID and a deleterID, allows us to delete a bot.

Precondition

True

Parameters

botID	UUID of a bot
deleterID	UUID of a User

Postcondition

The bot is deleted from the repository.

Definition at line 336 of file PlayerCtrl.java.

```
                                                                                            {
        Bot bot = this.getBot(botID);

        if (!bot.getCreatorID().equals(deleterID))
            throw new NotCreatorException();
        if (this.gameRepositoryCtrl.existsGameByPlayerID(botID))
            throw new BotUsedException();

        this.repositoryCtrl.delete(botID);
    }
```

6.82.3.14 hash()

```
String domain.PlayerCtrl.hash (
              String text )  [private]
```

Method that, given a password, it hashes it using SHA-256.

Precondition

True

Parameters

text	String to hash

Postcondition

Returns the hashed password

Definition at line 354 of file PlayerCtrl.java.

```
                                        {
        if (text.isBlank())
            return "";
        try {
            byte[] hash =
      MessageDigest.getInstance("SHA-256").digest(text.getBytes(StandardCharsets.UTF_8));

            StringBuilder hexString = new StringBuilder(2 * hash.length);
            for (int i = 0; i < hash.length; i++) {
                String hex = Integer.toHexString(0xff & hash[i]);
                if (hex.length() == 1)
                    hexString.append('0');
                hexString.append(hex);
            }

            return hexString.toString();
        } catch (Exception e) {
            return "";
        }
    }
```

6.82.4 Member Data Documentation

6.82.4.1 repositoryCtrl

```
PlayerRepositoryCtrl domain.PlayerCtrl.repositoryCtrl  [private]
```

Instance of the Player Repository.

Definition at line 44 of file PlayerCtrl.java.

6.82.4.2 gameRepositoryCtrl

```
GameRepositoryCtrl domain.PlayerCtrl.gameRepositoryCtrl  [private]
```

Instance of the Game Repository.

Definition at line 49 of file PlayerCtrl.java.

The documentation for this class was generated from the following file:

- PlayerCtrl.java

6.83 repository.PlayerRepository Class Reference

Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.

Public Member Functions

- PlayerRepository ()

  Create a PlayerRepository instance.
- void save (JSONObject player)

  Save a Player into the player database.
- void delete (String id)

  Delete a Player by ID from the player database. Soft delete if it is a User and hard delete if it is a Bot.
- JSONObject get (String id)

  Get the Player by ID from the player database or null if it does not exist.
- JSONObject getByName (String name)

  Get the Player by name from the player database or null if it does not exist.
- ArrayList< Pair< String, UUID > > listUsers ()

  List all non-deleted Users of the player database.
- ArrayList< Pair< String, UUID > > listBots ()

  List all non-deleted Bots of the player database.

Additional Inherited Members

6.83.1 Detailed Description

Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.

See also

repository.Repository

Definition at line 21 of file PlayerRepository.java.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 PlayerRepository()

```
repository.PlayerRepository.PlayerRepository ( )
```

Create a PlayerRepository instance.

Precondition

The Player repository JSON files exists.

Postcondition

A PlayerRepository instance is created.

Definition at line 31 of file PlayerRepository.java.

```
                             {
        super(RepositoryType.PLAYER);
    }
```

6.83.3 Member Function Documentation

6.83.3.1 save()

```
void repository.PlayerRepository.save (
             JSONObject player )
```

Save a Player into the player database.

Precondition

The Player repository JSON files exists.

Postcondition

The Player is saved into the player database.

Parameters

player	Player to be saved.

Definition at line 43 of file PlayerRepository.java.

```
                                        {
        String id = player.getString("id");
        this.createOrUpdate(id, player);
    }
```

6.83.3.2 delete()

```
void repository.PlayerRepository.delete (
            String id )
```

Delete a Player by ID from the player database. Soft delete if it is a User and hard delete if it is a Bot.

Precondition

The Player repository JSON files exists.

Postcondition

The Player is soft or hard deleted from the player database by ID depending whether it is a User or a Bot.

Parameters

id	ID of the Player to be deleted.

Definition at line 54 of file PlayerRepository.java.

```
                                  {
        JSONObject player = this.get(id);
        if (player == null)
            return;

        if (player.getString("type").equals("BOT")) {
            this.remove(id);
            return;
        }

        player.put("is_deleted", true);
        this.save(player);
    }
```

6.83.3.3 get()

```
JSONObject repository.PlayerRepository.get (
            String id )
```

Get the Player by ID from the player database or null if it does not exist.

Precondition

The Player repository JSON files exists.

Postcondition

A JSONObject representing the Player by ID from the player database is returned or null if it does not exist.

Parameters

id	ID of the Player to be getted.

Returns

JSONObject that represents the Player by ID from the player database or null if it does not exist.

Reimplemented from repository.Repository.

Definition at line 75 of file PlayerRepository.java.

```
                                       {
        return super.get(id);
    }
```

6.83.3.4 getByName()

```
JSONObject repository.PlayerRepository.getByName (
            String name )
```

Get the Player by name from the player database or null if it does not exist.

Precondition

The Player repository JSON files exists.

Postcondition

A JSONObject representing the Player by name from the player database is returned or null if it does not exist.

Parameters

name	Name of the Player to be getted.

Returns

JSONObject that represents the Player by name from the player database or null if it does not exist.

Definition at line 86 of file PlayerRepository.java.

```
                                               {
        JSONObject all = this.list();

        JSONObject current;
        for (String key : all.keySet()) {
            current = all.getJSONObject(key);
            if (current.getString("name").equals(name))
                return current;
        }

        return null;
    }
```

6.83.3.5 listUsers()

```
ArrayList<Pair<String, UUID> > repository.PlayerRepository.listUsers ( )
```

List all non-deleted Users of the player database.

Precondition

The Player repository JSON files exists.

Postcondition

An ArrayList containing the non-deleted User names and IDs of the player database is returned.

Returns

ArrayList of the non-deleted User names and IDs of the player database.

Definition at line 105 of file PlayerRepository.java.

```
                                                   {
        ArrayList<Pair<String, UUID>> list = new ArrayList<Pair<String, UUID>>();
        JSONObject all = this.list();

        JSONObject current;
        for (String key : all.keySet()) {
            current = all.getJSONObject(key);
            if (current.getString("type").equals("USER") && !current.getBoolean("is_deleted"))
                list.add(new Pair<String, UUID>(current.getString("name"), UUID.fromString(key)));
        }

        return list;
    }
```

6.83.3.6 listBots()

```
ArrayList<Pair<String, UUID> > repository.PlayerRepository.listBots ( )
```

List all non-deleted Bots of the player database.

Precondition

The Player repository JSON files exists.

Postcondition

An ArrayList containing the non-deleted Bots names and IDs of the player database is returned.

Returns

ArrayList of the non-deleted Bots names and IDs of the player database.

Definition at line 125 of file PlayerRepository.java.

```
                                                  {
        ArrayList<Pair<String, UUID>> list = new ArrayList<Pair<String, UUID>>();
        JSONObject all = this.list();

        JSONObject current;
        for (String key : all.keySet()) {
            current = all.getJSONObject(key);
            if (current.getString("type").equals("BOT") && !current.getBoolean("is_deleted"))
                list.add(new Pair<String, UUID>(current.getString("name"), UUID.fromString(key)));
        }

        return list;
    }
```

The documentation for this class was generated from the following file:

- PlayerRepository.java

6.84 repository.PlayerRepositoryCtrl Class Reference

Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.

Public Member Functions

- PlayerRepositoryCtrl ()

  Create a PlayerRepositoryCtrl instance.
- void save (JSONObject player)

  Save a Player into the player database.
- void delete (UUID id)

  Delete a Player by ID from the player database. Soft delete if it is a User and hard delete if it is a Bot.
- JSONObject get (UUID id)

  Get the Player by ID from the player database or null if it does not exist.
- JSONObject getByName (String name)

  Get the Player by name from the player database or null if it does not exist.
- ArrayList< Pair< String, UUID > > listUsers ()

  List all non-deleted Users of the player database.
- ArrayList< Pair< String, UUID > > listBots ()

  List all non-deleted Bots of the player database.

Private Attributes

- PlayerRepository repository

  PlayerRepository instance.

6.84.1 Detailed Description

Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.

See also

repository.PlayerRepository

Definition at line 21 of file PlayerRepositoryCtrl.java.

6.84.2 Constructor & Destructor Documentation

6.84.2.1 PlayerRepositoryCtrl()

```
repository.PlayerRepositoryCtrl.PlayerRepositoryCtrl ( )
```

Create a PlayerRepositoryCtrl instance.

Precondition

The Player repository JSON files exists.

Postcondition

A PlayerRepositoryCtrl instance is created.

Definition at line 36 of file PlayerRepositoryCtrl.java.

```
                                {
        this.repository = new PlayerRepository();
    }
```

6.84.3 Member Function Documentation

6.84.3.1 save()

```
void repository.PlayerRepositoryCtrl.save (
            JSONObject player )
```

Save a Player into the player database.

Precondition

The Player repository JSON files exists.

Postcondition

The Player is saved into the player database.

Parameters

player	Player to be saved.

Definition at line 48 of file PlayerRepositoryCtrl.java.

```
                                        {
        this.repository.save(player);
    }
```

6.84.3.2 delete()

```
void repository.PlayerRepositoryCtrl.delete (
            UUID id )
```

Delete a Player by ID from the player database. Soft delete if it is a User and hard delete if it is a Bot.

Precondition

The Player repository JSON files exists.

Postcondition

The Player is soft or hard deleted from the player database by ID depending whether it is a User or a Bot.

Parameters

id	ID of the Player to be deleted.

Definition at line 58 of file PlayerRepositoryCtrl.java.

```
                             {
        this.repository.delete(id.toString());
    }
```

6.84.3.3 get()

```
JSONObject repository.PlayerRepositoryCtrl.get (
            UUID id )
```

Get the Player by ID from the player database or null if it does not exist.

Precondition

The Player repository JSON files exists.

Postcondition

A JSONObject representing the Player by ID from the player database is returned or null if it does not exist.

Parameters

id	ID of the Player to be getted.

Returns

JSONObject that represents the Player by ID from the player database or null if it does not exist.

Definition at line 69 of file PlayerRepositoryCtrl.java.

```
                              {
        return this.repository.get(id.toString());
    }
```

6.84.3.4 getByName()

```
JSONObject repository.PlayerRepositoryCtrl.getByName (
              String name )
```

Get the Player by name from the player database or null if it does not exist.

Precondition

The Player repository JSON files exists.

Postcondition

A JSONObject representing the Player by name from the player database is returned or null if it does not exist.

Parameters

name	Name of the Player to be getted.

Returns

JSONObject that represents the Player by name from the player database or null if it does not exist.

Definition at line 80 of file PlayerRepositoryCtrl.java.

```
                                       {
        return this.repository.getByName(name);
    }
```

6.84.3.5 listUsers()

```
ArrayList<Pair<String, UUID> > repository.PlayerRepositoryCtrl.listUsers ( )
```

List all non-deleted Users of the player database.

Precondition

The Player repository JSON files exists.

Postcondition

An ArrayList containing the non-deleted User names and IDs of the player database is returned.

Returns

ArrayList of the non-deleted User names and IDs of the player database.

Definition at line 90 of file PlayerRepositoryCtrl.java.

```
                                               {
        return this.repository.listUsers();
    }
```

6.84.3.6 listBots()

```
ArrayList<Pair<String, UUID> > repository.PlayerRepositoryCtrl.listBots ( )
```

List all non-deleted Bots of the player database.

Precondition

The Player repository JSON files exists.

Postcondition

An ArrayList containing the non-deleted Bots names and IDs of the player database is returned.

Returns

ArrayList of the non-deleted Bots names and IDs of the player database.

Definition at line 100 of file PlayerRepositoryCtrl.java.

```
                                                   {
        return this.repository.listBots();
    }
```

6.84.4 Member Data Documentation

6.84.4.1 repository

```
PlayerRepository repository.PlayerRepositoryCtrl.repository  [private]
```

PlayerRepository instance.

Definition at line 27 of file PlayerRepositoryCtrl.java.

The documentation for this class was generated from the following file:

- PlayerRepositoryCtrl.java

6.85 view.PlayView Class Reference

Public Member Functions

- PlayView ()

  Class creator.
- void initialize ()
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void onChangeGameChooser () throws IOException

  Event method which is executed when the Game chooser is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void goToGame () throws IOException

  Event method which is executed when the goToMenu button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Label player1

  Player 1 label.
- Label player2

  Player 2 label.
- Label configuration

  Configuration label.
- Label creator

  Creator label.
- Label createdAt

  Created At label.
- Label state

  State label.

- Label info

  Info label.
- Label playResult

  Exception message output.
- Text playGame

  Play game button text.
- Rectangle playGameButton

  Play game button text.
- Label goToGame

  goToGame button label.
- Rectangle goToGameButton

  goToGame button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.
- ChoiceBox gameChooser

  Configuration choiceBox.

6.85.1 Detailed Description

This class represents the scene controller of the Play Game .

By Alex Rodriguez

Definition at line 32 of file PlayView.java.

6.85.2 Constructor & Destructor Documentation

6.85.2.1 PlayView()

```
view.PlayView.PlayView ( )
```

Class creator.

Definition at line 39 of file PlayView.java.

```
39                          {
40      }
```

6.85.3 Member Function Documentation

6.85.3.1 initialize()

```
void view.PlayView.initialize ( )
```

Definition at line 152 of file PlayView.java.

```
                          {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        Pair<ArrayList<String>, String> gameList = ViewCtrl.domainCtrl.listGames();
        for(String gameName : gameList.first) gameChooser.getItems().add(gameName);
    }
```

6.85.3.2 user()

```
void view.PlayView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 163 of file PlayView.java.

```
                                       {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.85.3.3 bots()

```
void view.PlayView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 172 of file PlayView.java.

```
                                       {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.85.3.4 config()

```
void view.PlayView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 181 of file PlayView.java.

```
                                          {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.85.3.5 games()

```
void view.PlayView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 190 of file PlayView.java.

```
                                         {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.85.3.6 onChangeGameChooser()

```
void view.PlayView.onChangeGameChooser ( ) throws IOException
```

Event method which is executed when the Game chooser is clicked.

Precondition

True

Postcondition

Game information is shown.

Definition at line 199 of file PlayView.java.

```
                                                         {
        String chosenGame = (String) gameChooser.getValue();
        if (chosenGame != null) {
            Pair<JSONObject, String> result = ViewCtrl.domainCtrl.getGame(chosenGame);
            if (result.second != null) {
                switch (result.second) {
                    case "ERR_NOT_PLAYER":
                        playResult.setText("You are not part of this game!");
                        break;
                    default:
                        playResult.setText("Something went wrong, try again!");
                        break;
                }
            }
            else {
                playResult.setText("");
                ViewCtrl.domainCtrl.selectGame(result.first.getString("name")); // Load onto memory the chosen game Board
                Pair<JSONObject, JSONObject> players = ViewCtrl.domainCtrl.viewPlayers();
                player1.setText((players.first != null ? players.first.getString("name") : "Unknown"));
                player2.setText((players.second != null ? players.second.getString("name") : "Unknown"));
                configuration.setText(result.first.getString("configuration_name"));
                Pair<JSONObject, String> userCreator = ViewCtrl.domainCtrl.getUser(UUID.fromString(result.first.getString("creator_id")));
                creator.setText((userCreator.first != null ? userCreator.first.getString("name") : "Unknown"));
                if (creator.getText().equals(currentUserName.getText()) && !player1.getText().equals(currentUserName.getText()) && !player2.getText().equals(currentUserName.getText())) {
                    goToGame.setText("SPECTATE");
                } else {
                    goToGame.setText("PLAY");
                }
                DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
                createdAt.setText(LocalDateTime.parse(result.first.getString("created_at")).format(dateFormat));
                state.setText(result.first.get("state").toString());
                if (result.first.get("state").toString().equals("FINISHED")) {
                    goToGame.setText("CONSULT");
                    String winner = null;
                    winner = result.first.optString("winner_id", null);
                    if (winner == null) {
                        info.setText("The game has ended in a draw.");
                    } else {
                        if (winner.equals(players.first.getString("id"))) {
                            info.setText(String.format("%s has won the game.", player1.getText()));
                        } else {
                            info.setText(String.format("%s has won the game.", player2.getText()));
                        }
                    }
                } else {
                    if (result.first.get("turn").toString().equals("PLAYER1")) {
                        info.setText(String.format("%s has the current turn.", player1.getText()));
                    } else {
                        info.setText(String.format("%s has the current turn.", player2.getText()));
                    }
                }
            }
        }
    }
```

6.85.3.7 ranking()

```
void view.PlayView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 259 of file PlayView.java.

```
                                   {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.85.3.8 goToGame()

```
void view.PlayView.goToGame ( ) throws IOException
```

Event method which is executed when the goToMenu button is clicked.

Precondition

True

Postcondition

The scene is changed to GameBoardView.

Definition at line 268 of file PlayView.java.

```
                                    {
        String chosenGame = (String) gameChooser.getValue();
        if (chosenGame != null) {
            Pair<JSONObject, String> result = ViewCtrl.domainCtrl.play();
            if (result.second != null) {
                switch (result.second) {
                    case "ERR_FINISHED_GAME":
                        // playResult.setText("The game has already finished!");
                        playResult.setText("");
                        ViewCtrl.changeScene("template/GameBoardView.fxml");
                        break;
                    default:
                        playResult.setText("Something went wrong, try again!");
                        break;
                }
            }
            else {
                playResult.setText("");
                ViewCtrl.changeScene("template/GameBoardView.fxml");
            }
        }
    }
```

6.85.3.9 logOut()

```
void view.PlayView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 296 of file PlayView.java.

```
                                         {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.85.4 Member Data Documentation

6.85.4.1 user

```
Text view.PlayView.user  [private]
```

Menu User tab.

Definition at line 48 of file PlayView.java.

6.85.4.2 bots

```
Text view.PlayView.bots  [private]
```

Menu Bots tab.

Definition at line 53 of file PlayView.java.

6.85.4.3 config

```
Text view.PlayView.config  [private]
```

Menu Configuration tab.

Definition at line 58 of file PlayView.java.

6.85.4.4 games

```
Text view.PlayView.games  [private]
```

Menu Games tab.

Definition at line 63 of file PlayView.java.

6.85.4.5 ranking

```
Text view.PlayView.ranking  [private]
```

Menu Ranking tab.

Definition at line 68 of file PlayView.java.

6.85.4.6 play

```
Text view.PlayView.play  [private]
```

Menu Play tab.

Definition at line 73 of file PlayView.java.

6.85.4.7 player1

```
Label view.PlayView.player1  [private]
```

Player 1 label.

Definition at line 78 of file PlayView.java.

6.85.4.8 player2

```
Label view.PlayView.player2  [private]
```

Player 2 label.

Definition at line 83 of file PlayView.java.

6.85.4.9 configuration

```
Label view.PlayView.configuration  [private]
```

Configuration label.

Definition at line 88 of file PlayView.java.

6.85.4.10 creator

```
Label view.PlayView.creator  [private]
```

Creator label.

Definition at line 93 of file PlayView.java.

6.85.4.11 createdAt

```
Label view.PlayView.createdAt  [private]
```

Created At label.

Definition at line 98 of file PlayView.java.

6.85.4.12 state

```
Label view.PlayView.state  [private]
```

State label.

Definition at line 103 of file PlayView.java.

6.85.4.13 info

```
Label view.PlayView.info  [private]
```

Info label.

Definition at line 108 of file PlayView.java.

6.85.4.14 playResult

```
Label view.PlayView.playResult  [private]
```

Exception message output.

Definition at line 113 of file PlayView.java.

6.85.4.15 playGame

```
Text view.PlayView.playGame  [private]
```

Play game button text.

Definition at line 118 of file PlayView.java.

6.85.4.16 playGameButton

```
Rectangle view.PlayView.playGameButton  [private]
```

Play game button text.

Definition at line 123 of file PlayView.java.

6.85.4.17 goToGame

```
Label view.PlayView.goToGame  [private]
```

goToGame button label.

Definition at line 128 of file PlayView.java.

6.85.4.18 goToGameButton

```
Rectangle view.PlayView.goToGameButton  [private]
```

goToGame button.

Definition at line 133 of file PlayView.java.

6.85.4.19 currentUserName

```
Label view.PlayView.currentUserName  [private]
```

Current user name.

Definition at line 138 of file PlayView.java.

6.85.4.20 logOut

```
Text view.PlayView.logOut  [private]
```

LogOut button.

Definition at line 143 of file PlayView.java.

6.85.4.21 gameChooser

```
ChoiceBox view.PlayView.gameChooser  [private]
```

Configuration choiceBox.

Definition at line 148 of file PlayView.java.

The documentation for this class was generated from the following file:

- PlayView.java

6.86 cmd.unitary.ranking Class Reference

JUnit Ranking tests entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)

  JUnit Ranking tests main function. Calls the JUnitCore main entrypoint and runs the Ranking unitary tests.

6.86.1 Detailed Description

JUnit Ranking tests entrypoint. By Alex Rodriguez.

Definition at line 17 of file ranking.java.

6.86.2 Member Function Documentation

6.86.2.1 main()

```
static void cmd.unitary.ranking.main (
            String[ ] args )  [static]
```

JUnit Ranking tests main function. Calls the JUnitCore main entrypoint and runs the Ranking unitary tests.

Precondition

True.

Postcondition

The JUnit Ranking tests have started.

Definition at line 24 of file ranking.java.

```
                                       {
         JUnitCore.main(new RankingJUnit().getClass().getName());
     }
```

The documentation for this class was generated from the following file:

- ranking.java

6.87 domain.Ranking Class Reference

Representation of a ranking table.

Classes

- enum RankingType

Public Member Functions

- Ranking (String name)
  Builder operation that gets a name for a new Ranking as a parameter and creates an empty Ranking.
- Ranking (JSONObject ranking)
  Builder operation that creates a new Ranking based on parameter ranking.
- JSONObject serialize ()
  Operation that translates a Ranking into a JSONObject.
- Entry getRecord (UUID playerID)
  Consulting operation that returns the record of the player with the playerID passed as a parameter.
- void removePlayer (UUID playerID)
  Modifying operation that removes a player's entries from the implicit ranking.
- void addEntry (Entry entry)
  Modifying operation that adds the parameter entry to the ranking table.
- String getName ()
  Consulting operation that returns the implicit Ranking's name.
- ArrayList< Entry > getEntries ()
  Consulting operation that returns the implicit Ranking's ArrayList.

Private Member Functions

- int whereInsert (int value, int start, int end)
  Private method that returns where to place parameter value in the ranking table based on a binary search.

Private Attributes

- String name
  Name of the table.
- ArrayList< Entry > entries
  Ranking table.

6.87.1 Detailed Description

Representation of a ranking table.

Created by Roger Mollon

Class that represents a ranking table. Contains the tableName and its Entries. The table is ordered by values

Definition at line 21 of file Ranking.java.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 Ranking() [1/2]

```
domain.Ranking.Ranking (
              String name )
```

Builder operation that gets a name for a new Ranking as a parameter and creates an empty Ranking.

Precondition

True

Postcondition

An empty Ranking of name entriesName has been created

Parameters

name	Name of the table to be created

Definition at line 36 of file Ranking.java.

```
                                {
        this.entries = new ArrayList<Entry>();
        this.name = name;
    }
```

6.87.2.2 Ranking() [2/2]

```
domain.Ranking.Ranking (
              JSONObject ranking )
```

Builder operation that creates a new Ranking based on parameter ranking.

Precondition

True

Postcondition

A Ranking with its attributes based on ranking has been created

Parameters

ranking	JSONObject with the attributes of the implicit Ranking

Definition at line 46 of file Ranking.java.

```
                                       {
        this.name = ranking.getString("name");
        this.entries = new ArrayList<Entry>();
        JSONArray entries = ranking.getJSONArray("entries");
        for(int i=0; i<entries.length(); ++i) this.entries.add(i, new Entry(entries.getJSONObject(i)));
    }
```

6.87.3 Member Function Documentation

6.87.3.1 serialize()

```
JSONObject domain.Ranking.serialize ( )
```

Operation that translates a Ranking into a JSONObject.

Precondition

True

Postcondition

A new JSONObject with information from implicit Ranking has been returned

Returns

JSONObject with attributes from implicit Ranking

Definition at line 58 of file Ranking.java.

```
                                  {
        JSONObject ranking = new JSONObject();
        ranking.put("name", this.name);
        JSONArray entries = new JSONArray();
        for(Entry e: this.entries) entries.put(e.serialize());
        ranking.put("entries", entries);
        return ranking;
    }
```

6.87.3.2 getRecord()

```
Entry domain.Ranking.getRecord (
              UUID playerID )
```

Consulting operation that returns the record of the player with the playerID passed as a parameter.

Precondition

True

Postcondition

The first entry from the player has been returned if possible

Parameters

playerID	ID of the player whose record will be returned

Returns

First Entry of the requested player in case the player has at least 1 instance. If not it returns null

Definition at line 73 of file Ranking.java.

```
                                     {
        for(Entry entry : this.entries)
            if(entry.getPlayerID().equals(playerID))
                return entry;
        return null;
    }
```

6.87.3.3 removePlayer()

```
void domain.Ranking.removePlayer (
             UUID playerID )
```

Modifying operation that removes a player's entries from the implicit ranking.

Precondition

True

Postcondition

All the player's entries have been removed from the implicit ranking.

Parameters

playerID	Player ID to remove from.

Definition at line 85 of file Ranking.java.

```
                                       {
        ArrayList<Entry> newEntries = new ArrayList<Entry>();

        for(Entry entry : this.entries)
            if(!entry.getPlayerID().equals(playerID))
                newEntries.add(entry);

        this.entries = newEntries;
    }
```

6.87.3.4 whereInsert()

```
int domain.Ranking.whereInsert (
             int value,
             int start,
             int end )  [private]
```

Private method that returns where to place parameter value in the ranking table based on a binary search.

Precondition

value $>$ 0, start $>=$ 0 and end $<=$ this.entries.size() The position in which to insert in the ranking table has been returned

Parameters

value	Value of the Entry to be placed in the ranking table
start	Starting place of the segment of the ranking table to check
end	Ending place of the segment of the ranking table to check

Returns

Position of the value in the ranking table

Definition at line 103 of file Ranking.java.

```
                                                              {
        if(start == end) {
            if(start == this.entries.size()) return start;
            if(this.entries.get(start).getValue() > value) return start+1;
            else return start;
        }
        if(start > end) return start;

        int middle = (start+end)/2;
        int current = this.entries.get(middle).getValue();

        if(current > value) return whereInsert(value, middle+1, end);
        else if(current < value) return whereInsert(value, start, middle-1);
        else return middle;
    }
```

6.87.3.5 addEntry()

```
void domain.Ranking.addEntry (
              Entry entry )
```

Modifying operation that adds the parameter entry to the ranking table.

Precondition

True

Postcondition

A new Entry has been correctly added to the ranking table

Parameters

entry	Entry added to Ranking

Definition at line 124 of file Ranking.java.

```
                                        {
        this.entries.add(this.whereInsert(entry.getValue(), 0, this.entries.size()), entry);
    }
```

6.87.3.6 getName()

```
String domain.Ranking.getName ( )
```

Consulting operation that returns the implicit Ranking's name.

Precondition

True

Postcondition

String name from the implicit Ranking has been returned

Returns

String name

Definition at line 133 of file Ranking.java.

```
                                 {
        return this.name;
    }
```

6.87.3.7 getEntries()

```
ArrayList<Entry> domain.Ranking.getEntries ( )
```

Consulting operation that returns the implicit Ranking's ArrayList.

Precondition

True

Postcondition

ArrayList<Entry> entries has been returned

Returns

ArrayList<Entry> entries

Definition at line 142 of file Ranking.java.

```
                                               {
        return this.entries;
    }
```

6.87.4 Member Data Documentation

6.87.4.1 name

```
String domain.Ranking.name  [private]
```

Name of the table.

Definition at line 27 of file Ranking.java.

6.87.4.2 entries

```
ArrayList<Entry> domain.Ranking.entries  [private]
```

Ranking table.

Definition at line 29 of file Ranking.java.

The documentation for this class was generated from the following file:

- Ranking.java

6.88 view.RankingConsultView Class Reference

Public Member Functions

- RankingConsultView ()

  Class creator.
- void initialize ()

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void onChangeRankingChooser () throws IOException

  Event method which is executed when the Ranking chooser is clicked.
- void consultRankings () throws IOException

  Event method which is executed when the Ranking consult button is clicked.
- void consultRecords () throws IOException

  Event method which is executed when the Record consult button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

    Menu User tab.
- Text bots

    Menu Bots tab.
- Text config

    Menu Configuration tab.
- Text games

    Menu Games tab.
- Text ranking

    Menu Ranking tab.
- Text play

    Menu Play tab.
- Text consultRankings

    Ranking consult button text.
- Rectangle consultRankingsButton

    Ranking consult button.
- Text consultRankingsConfirm

    Ranking consult confirm text button.
- Rectangle consultRankingsConfirmButton

    Ranking consult confirm button.
- Text consultRecords

    Records consult button text.
- Rectangle consultRecordsButton

    Records consult button.
- ChoiceBox rankingChooser

    Ranking choiceBox.
- Label rankingInfo

    Ranking information label.
- Label currentUserName

    Current user name.
- Text logOut

    LogOut button.
- TableView table

    Ranking table.
- TableColumn playerColumn

    Player column.
- TableColumn valueColumn

    Value column.

6.88.1 Detailed Description

This class represents the scene controller of consult function of a ranking.

By Alex Rodriguez

Definition at line 32 of file RankingConsultView.java.

6.88.2 Constructor & Destructor Documentation

6.88.2.1 RankingConsultView()

`view.RankingConsultView.RankingConsultView ( )`

Class creator.

Definition at line 39 of file RankingConsultView.java.

```
                              {
    }
```

6.88.3 Member Function Documentation

6.88.3.1 initialize()

`void view.RankingConsultView.initialize ( )`

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. All ranking names are inserted in the Ranking choiceBox.

Definition at line 146 of file RankingConsultView.java.

```
                              {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        ArrayList<String> rankingList = ViewCtrl.domainCtrl.listRankings();
        for(String rankingName : rankingList) rankingChooser.getItems().add(rankingName);
        playerColumn.setCellValueFactory(new PropertyValueFactory<>("first"));
        valueColumn.setCellValueFactory(new PropertyValueFactory<>("second"));
    }
```

6.88.3.2 user()

`void view.RankingConsultView.user ( ) throws IOException`

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 159 of file RankingConsultView.java.

```
                                        {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.88.3.3 bots()

`void view.RankingConsultView.bots ( ) throws IOException`

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 168 of file RankingConsultView.java.

```
                                    {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.88.3.4 config()

`void view.RankingConsultView.config ( ) throws IOException`

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 177 of file RankingConsultView.java.

```
                                      {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.88.3.5 games()

`void view.RankingConsultView.games ( ) throws IOException`

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 186 of file RankingConsultView.java.

```
                                     {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.88.3.6 play()

```
void view.RankingConsultView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 195 of file RankingConsultView.java.

```
                                            {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.88.3.7 onChangeRankingChooser()

```
void view.RankingConsultView.onChangeRankingChooser ( ) throws IOException
```

Event method which is executed when the Ranking chooser is clicked.

Precondition

True

Postcondition

Ranking information is shown.

Definition at line 204 of file RankingConsultView.java.

```
                                                              {
        String chosenRanking = (String) rankingChooser.getValue();
        if (rankingChooser != null) {
            table.getItems().clear();
            JSONObject ranking = ViewCtrl.domainCtrl.getRanking(chosenRanking);
            JSONArray entries = ranking.getJSONArray("entries");
            for (int i = 0; i < entries.length(); ++i) {
                JSONObject entry = entries.getJSONObject(i);
                Pair<JSONObject, String> player =
    ViewCtrl.domainCtrl.getPlayer(UUID.fromString(entry.getString("player_id")));
                Pair<String, Integer> pairEntry = new Pair<String,
    Integer>(player.first.getString("name"), entry.getInt("value"));
                if (player.first.getString("type") == "BOT")
                    pairEntry.first = pairEntry.first + " (bot)";
                if (player.first.getBoolean("is_deleted"))
                    pairEntry.first = pairEntry.first + " (deleted)";
                table.getItems().add(pairEntry);
            }
        }
    }
```

6.88.3.8 consultRankings()

`void view.RankingConsultView.consultRankings ( ) throws IOException`

Event method which is executed when the Ranking consult button is clicked.

Precondition

True

Postcondition

The current scene is changed to RankingConsultView.

Definition at line 228 of file RankingConsultView.java.

```
                                                    {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.88.3.9 consultRecords()

`void view.RankingConsultView.consultRecords ( ) throws IOException`

Event method which is executed when the Record consult button is clicked.

Precondition

True

Postcondition

The current scene is changed to RecordConsultView.

Definition at line 237 of file RankingConsultView.java.

```
                                                   {
        ViewCtrl.changeScene("template/RecordConsultView.fxml");
    }
```

6.88.3.10 logOut()

`void view.RankingConsultView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 246 of file RankingConsultView.java.

```
                                            {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.88.4 Member Data Documentation

6.88.4.1 user

```
Text view.RankingConsultView.user  [private]
```

Menu User tab.

Definition at line 47 of file RankingConsultView.java.

6.88.4.2 bots

```
Text view.RankingConsultView.bots  [private]
```

Menu Bots tab.

Definition at line 52 of file RankingConsultView.java.

6.88.4.3 config

```
Text view.RankingConsultView.config  [private]
```

Menu Configuration tab.

Definition at line 57 of file RankingConsultView.java.

6.88.4.4 games

```
Text view.RankingConsultView.games  [private]
```

Menu Games tab.

Definition at line 62 of file RankingConsultView.java.

6.88.4.5 ranking

```
Text view.RankingConsultView.ranking  [private]
```

Menu Ranking tab.

Definition at line 67 of file RankingConsultView.java.

6.88.4.6 play

```
Text view.RankingConsultView.play  [private]
```

Menu Play tab.

Definition at line 72 of file RankingConsultView.java.

6.88.4.7 consultRankings

```
Text view.RankingConsultView.consultRankings  [private]
```

Ranking consult button text.

Definition at line 77 of file RankingConsultView.java.

6.88.4.8 consultRankingsButton

```
Rectangle view.RankingConsultView.consultRankingsButton  [private]
```

Ranking consult button.

Definition at line 82 of file RankingConsultView.java.

6.88.4.9 consultRankingsConfirm

```
Text view.RankingConsultView.consultRankingsConfirm  [private]
```

Ranking consult confirm text button.

Definition at line 87 of file RankingConsultView.java.

6.88.4.10 consultRankingsConfirmButton

```
Rectangle view.RankingConsultView.consultRankingsConfirmButton  [private]
```

Ranking consult confirm button.

Definition at line 92 of file RankingConsultView.java.

6.88.4.11 consultRecords

```
Text view.RankingConsultView.consultRecords  [private]
```

Records consult button text.

Definition at line 97 of file RankingConsultView.java.

6.88.4.12 consultRecordsButton

```
Rectangle view.RankingConsultView.consultRecordsButton  [private]
```

Records consult button.

Definition at line 102 of file RankingConsultView.java.

6.88.4.13 rankingChooser

```
ChoiceBox view.RankingConsultView.rankingChooser  [private]
```

Ranking choiceBox.

Definition at line 107 of file RankingConsultView.java.

6.88.4.14 rankingInfo

```
Label view.RankingConsultView.rankingInfo  [private]
```

Ranking information label.

Definition at line 112 of file RankingConsultView.java.

6.88.4.15 currentUserName

```
Label view.RankingConsultView.currentUserName  [private]
```

Current user name.

Definition at line 117 of file RankingConsultView.java.

6.88.4.16 logOut

```
Text view.RankingConsultView.logOut  [private]
```

LogOut button.

Definition at line 122 of file RankingConsultView.java.

6.88.4.17 table

```
TableView view.RankingConsultView.table  [private]
```

Ranking table.

Definition at line 127 of file RankingConsultView.java.

6.88.4.18 playerColumn

```
TableColumn view.RankingConsultView.playerColumn  [private]
```

Player column.

Definition at line 132 of file RankingConsultView.java.

6.88.4.19 valueColumn

```
TableColumn view.RankingConsultView.valueColumn  [private]
```

Value column.

Definition at line 137 of file RankingConsultView.java.

The documentation for this class was generated from the following file:

- RankingConsultView.java

6.89 domain.RankingCtrl Class Reference

Ranking domain sub-controller. It communicates with the main domain controller and the ranking repository controller. By Alex Rodriguez.

Public Member Functions

- RankingCtrl ()

  Creator method that creates an instance of Ranking Controller.
- Ranking getRanking (String name)

  Returns the ranking identified by name.
- ArrayList< String > listRankings ()

  Returns a list of all ranking names in the system.
- ArrayList< Pair< String, Entry > > listRecords (UUID playerID)

  Returns the entries with the highest score of the current user for each ranking in the system.
- Entry createEntry (String rankingName, UUID playerID, Integer value, RankingType rankingType)

  Lets the system to automatically create an entry of the associated ranking when the current user finishes a game.

Private Attributes

- RankingRepositoryCtrl repositoryCtrl

  Ranking repository controller.

6.89.1 Detailed Description

Ranking domain sub-controller. It communicates with the main domain controller and the ranking repository controller. By Alex Rodriguez.

See also

domain.Ranking

Definition at line 25 of file RankingCtrl.java.

6.89.2 Constructor & Destructor Documentation

6.89.2.1 RankingCtrl()

```
domain.RankingCtrl.RankingCtrl ( )
```

Creator method that creates an instance of Ranking Controller.

Precondition

True

Postcondition

An instsance of Ranking Control is created.

Definition at line 41 of file RankingCtrl.java.

```
                        {
        this.repositoryCtrl = new RankingRepositoryCtrl();
    }
```

6.89.3 Member Function Documentation

6.89.3.1 getRanking()

```
Ranking domain.RankingCtrl.getRanking (
            String name )
```

Returns the ranking identified by name.

Precondition

The Ranking repository JSON files and the Ranking identified by name exists.

Postcondition

The Ranking identified by name is returned

Parameters

name	Name of a ranking

Returns

Ranking identified by name

Definition at line 54 of file RankingCtrl.java.

```
                                              {
        JSONObject ranking = this.repositoryCtrl.get(name);
        if (ranking == null)
            return null;

        return new Ranking(ranking);
    }
```

6.89.3.2 listRankings()

```
ArrayList<String> domain.RankingCtrl.listRankings ( )
```

Returns a list of all ranking names in the system.

Precondition

The Ranking repository JSON files and the default Rankings exists.

Postcondition

The list of names of rankings are returned in an ArrayList of Strings.

Returns

ArrayList of Strings

Definition at line 68 of file RankingCtrl.java.

```
                                    {
        return this.repositoryCtrl.listRankings();
    }
```

6.89.3.3 listRecords()

```
ArrayList<Pair<String, Entry> > domain.RankingCtrl.listRecords (
            UUID playerID )
```

Returns the entries with the highest score of the current user for each ranking in the system.

Precondition

PlayerID is not null

Postcondition

Returns a list of entries which corresponds to the records of the playerID

Parameters

playerID	UUID of a Player

Returns

List of records of a Player

Definition at line 79 of file RankingCtrl.java.

```
                                                                 {
        ArrayList<String> rankings = this.repositoryCtrl.listRankings();
        ArrayList<Pair<String, Entry» records = new ArrayList<Pair<String, Entry»();

        for (String name : rankings) {
            Ranking ranking = this.getRanking(name);
            if (ranking != null) {
                Entry record = ranking.getRecord(playerID);
                if (record != null)
                    records.add(new Pair<String, Entry>(name, record));
            }
        }

        return records;
    }
```

6.89.3.4 createEntry()

```
Entry domain.RankingCtrl.createEntry (
            String rankingName,
            UUID playerID,
            Integer value,
            RankingType rankingType )
```

Lets the system to automatically create an entry of the associated ranking when the current user finishes a game.

Precondition

Parameters aren't null

Postcondition

Returns the created entry

Parameters

rankingName	Name of a Ranking
playerID	UUID of a Player
value	Value of an entry
rankingType	Type of Ranking

Returns

Entry

Definition at line 105 of file RankingCtrl.java.

```
{
    Ranking ranking = this.getRanking(rankingName);
    if (ranking == null)
        ranking = new Ranking(rankingName);

    switch (rankingType) {
        case INCREMENTAL:
            Entry oldRecord = ranking.getRecord(playerID);
            if (oldRecord != null)
                value += oldRecord.getValue();
        case UNIQUE:
            ranking.removePlayer(playerID);
        case MULTIPLE:
        default:
            break;
    }

    Entry entry = new Entry(playerID, value);
    ranking.addEntry(entry);

    this.repositoryCtrl.save(ranking.serialize());

    return entry;
}
```

6.89.4 Member Data Documentation

6.89.4.1 repositoryCtrl

```
RankingRepositoryCtrl domain.RankingCtrl.repositoryCtrl  [private]
```

Ranking repository controller.

Definition at line 32 of file RankingCtrl.java.

The documentation for this class was generated from the following file:

- RankingCtrl.java

6.90 test.unitary.RankingJUnit Class Reference

Allows JUnit testing of class Ranking.

Public Member Functions

- void Ranking ()
- void deserialize ()
- void serialize ()
- void getRecord ()
- void addEntry ()
- void getName ()
- void getEntries ()

Private Member Functions

- int randomInt (Integer min, Integer max)

6.90.1 Detailed Description

Allows JUnit testing of class Ranking.

Created by Roger Mollon

Class that represents a testing of class Ranking. It contains tester methods for all public Ranking methods

Definition at line 25 of file RankingJUnit.java.

6.90.2 Member Function Documentation

6.90.2.1 Ranking()

```
void test.unitary.RankingJUnit.Ranking ( )
```

Definition at line 28 of file RankingJUnit.java.

```
28                          {
29          Ranking r = new Ranking("win_percentage");
30          assertEquals("Ranking failed because", "win_percentage", r.getName());
31          assertEquals("Ranking failed because", new ArrayList<Entry>(), r.getEntries());
32      }
```

6.90.2.2 randomInt()

```
int test.unitary.RankingJUnit.randomInt (
                Integer min,
                Integer max )  [private]
```

Definition at line 34 of file RankingJUnit.java.

```
34                                                      {
35          return min+(int)(Math.random()*((max-min)+1));
36      }
```

6.90.2.3 deserialize()

```
void test.unitary.RankingJUnit.deserialize ( )
```

Definition at line 39 of file RankingJUnit.java.

```
39                              {
40          Ranking r = new Ranking("number_of_games");
41          for(int i=0; i<5; ++i) r.addEntry(new Entry(UUID.randomUUID(), randomInt(0,i)));
42          Ranking r1 = new Ranking(r.serialize());
43
44          assertEquals("deserialize failed because", r.getName(), r1.getName());
45          for(int j=0; j<5; ++j) {
46              Entry entry = r.getEntries().get(j);
47              Entry entry1 = r1.getEntries().get(j);
48              assertEquals("deserialize failed because", entry.getPlayerID(), entry1.getPlayerID());
49              assertEquals("deserialize failed because", entry.getValue(), entry1.getValue());
50          }
51      }
```

6.90.2.4 serialize()

```
void test.unitary.RankingJUnit.serialize ( )
```

Definition at line 54 of file RankingJUnit.java.

```
54                            {
55          Ranking r = new Ranking("number_of_pieces");
56          for(int i=0; i<4; ++i) r.addEntry(new Entry(UUID.randomUUID(), randomInt(0,i)));
57          JSONObject jranking = r.serialize();
58          assertEquals("serialize failed because", r.getName(), jranking.getString("name"));
59          JSONArray entries = jranking.getJSONArray("entries");
60          for(int i=0; i<entries.length(); ++i) {
61              assertEquals("serialize failed because", entries.getJSONObject(i).getString("player_id"),
        r.getEntries().get(i).getPlayerID().toString());
62              assertEquals("serialize failed because", entries.getJSONObject(i).getInt("value"),
        r.getEntries().get(i).getValue());
63          }
64      }
```

6.90.2.5 getRecord()

```
void test.unitary.RankingJUnit.getRecord ( )
```

Definition at line 67 of file RankingJUnit.java.

```
                             {
        Ranking r = new Ranking("time");
        UUID playerID = UUID.randomUUID();

        // Case 1
        assertEquals("Record failed because", null, r.getRecord(playerID));

        // Case 2
        for(int i=0; i<4; ++i) r.addEntry(new Entry(UUID.randomUUID(), randomInt(0,i)));
        assertEquals("Record failed because", null, r.getRecord(playerID));

        // Case 3
        r.addEntry(new Entry(playerID, 6));
        assertEquals("Record failed because", new Entry(playerID, 6).getValue(), r.getRecord(playerID).getValue());

        // Case 4
        r.addEntry(new Entry(playerID, 6));
        r.addEntry(new Entry(playerID, 7));
        r.addEntry(new Entry(playerID, 5));
        r.addEntry(new Entry(playerID, 7));
        assertEquals("Record failed because", new Entry(playerID, 7).getValue(), r.getRecord(playerID).getValue());
    }
```

6.90.2.6 addEntry()

```
void test.unitary.RankingJUnit.addEntry ( )
```

Definition at line 91 of file RankingJUnit.java.

```
                            {
        Ranking r = new Ranking("number_of_wins");
        UUID id = UUID.randomUUID();
        UUID id1 = UUID.randomUUID();
        UUID id2 = UUID.randomUUID();
        UUID id3 = UUID.randomUUID();

        ArrayList<UUID> player_ids = new ArrayList<UUID>();
        ArrayList<Integer> player_values = new ArrayList<Integer>();

         // Case 1
         player_ids.add(0, id);
         player_values.add(0, 5);
         r.addEntry(new Entry(id, 5));
         for(int i=0; i < r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i), r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i), r.getEntries().get(i).getValue());
         }

         // Case 2
         player_ids.add(1, id1);
         player_values.add(1, 2);
         r.addEntry(new Entry(id1, 2));
         for(int i=0; i < r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i), r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i), r.getEntries().get(i).getValue());
         }

         // Case 3
         player_values.add(0, 25);
         player_ids.add(0, id2);
         r.addEntry(new Entry(id2, 25));
         for(int i=0; i < r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i), r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i), r.getEntries().get(i).getValue());
         }

         // Case 4
         player_ids.add(1, id3);
         player_values.add(1, 10);
         r.addEntry(new Entry(id3, 10));
         for(int i=0; i<r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i),
    r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i),
    r.getEntries().get(i).getValue());
         }

         // Case 5
         player_ids.add(0, id3);
         player_values.add(0, 25);
         r.addEntry(new Entry(id3, 25));
         for(int i=0; i<r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i),
    r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i),
    r.getEntries().get(i).getValue());
         }

         // Case 6
         player_ids.add(5, id3);
         player_values.add(5, 0);
         r.addEntry(new Entry(id3, 0));
         for(int i=0; i<r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i),
    r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i),
    r.getEntries().get(i).getValue());
         }

         // Case 7
         player_ids.add(0, id3);
         player_values.add(0, 69);
         r.addEntry(new Entry(id3, 69));
         for(int i=0; i<r.getEntries().size(); ++i) {
             assertEquals("addEntry failed because", player_ids.get(i),
    r.getEntries().get(i).getPlayerID());
             assertEquals("addEntry failed because",(int) player_values.get(i),
    r.getEntries().get(i).getValue());
         }

    }
```

6.90.2.7 getName()

`void test.unitary.RankingJUnit.getName ( )`

Definition at line 167 of file RankingJUnit.java.

```
                       {
        Ranking r = new Ranking("number_of_ties");
        assertEquals("getName failed because", "number_of_ties", r.getName());
    }
```

6.90.2.8 getEntries()

`void test.unitary.RankingJUnit.getEntries ( )`

Definition at line 173 of file RankingJUnit.java.

```
                          {
        Ranking r = new Ranking("number_of_losses");
        ArrayList<UUID> expectedIDs = new ArrayList<UUID> ();
        UUID id = UUID.randomUUID();
        UUID id1 = UUID.randomUUID();
        UUID id2 = UUID.randomUUID();
        UUID id3 = UUID.randomUUID();
        expectedIDs.add(id);
        expectedIDs.add(id3);
        expectedIDs.add(id2);
        expectedIDs.add(id1);

        r.addEntry(new Entry(id, 52));
        r.addEntry(new Entry(id1, 7));
        r.addEntry(new Entry(id2, 15));
        r.addEntry(new Entry(id3, 40));

        ArrayList<Integer> expectedValue = new ArrayList<Integer>();
        expectedValue.add(52);
        expectedValue.add(40);
        expectedValue.add(15);
        expectedValue.add(7);

        for(int i=0; i<4; ++i) {
            assertEquals("getEntries failed because", expectedIDs.get(i),
    r.getEntries().get(i).getPlayerID());
            assertEquals("getEntries failed because",(int) expectedValue.get(i),
    r.getEntries().get(i).getValue());
        }
    }
```

The documentation for this class was generated from the following file:

- RankingJUnit.java

6.91 repository.RankingRepository Class Reference

Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.

Public Member Functions

- RankingRepository ()

  Create a RankingRepository instance.
- void save (JSONObject ranking)

  Save a Ranking into the ranking database.
- JSONObject get (String name)

  Get the Ranking by name from the ranking database or null if it does not exist.
- ArrayList< String > listRankings ()

  List all Rankings of the ranking database.

Additional Inherited Members

6.91.1 Detailed Description

Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.

See also

repository.Repository

Definition at line 18 of file RankingRepository.java.

6.91.2 Constructor & Destructor Documentation

6.91.2.1 RankingRepository()

```
repository.RankingRepository.RankingRepository ( )
```

Create a RankingRepository instance.

Precondition

The Ranking repository JSON files and all the default Rankings exists.

Postcondition

A RankingRepository instance is created.

Definition at line 28 of file RankingRepository.java.

```
28                              {
29          super(RepositoryType.RANKING);
30      }
```

6.91.3 Member Function Documentation

6.91.3.1 save()

```
void repository.RankingRepository.save (
             JSONObject ranking )
```

Save a Ranking into the ranking database.

Precondition

The Ranking repository JSON files and the Ranking to be saved already exists, so it's only updated.

Postcondition

The Ranking is saved into the ranking database.

Parameters

ranking	Ranking to be saved.

Definition at line 40 of file RankingRepository.java.

```
                                          {
        String name = ranking.getString("name");
        this.createOrUpdate(name, ranking);
    }
```

6.91.3.2 get()

```
JSONObject repository.RankingRepository.get (
            String name )
```

Get the Ranking by name from the ranking database or null if it does not exist.

Precondition

The Ranking repository JSON files and the Ranking identified by name exists.

Postcondition

A JSONObject representing the Ranking by name from the ranking database is returned or null if it does not exist.

Parameters

name	Name of the Ranking to be getted.

Returns

JSONObject that represents the Ranking by name from the ranking database or null if it does not exist.

Reimplemented from repository.Repository.

Definition at line 52 of file RankingRepository.java.

```
                                        {
        return super.get(name);
    }
```

6.91.3.3 listRankings()

```
ArrayList<String> repository.RankingRepository.listRankings ( )
```

List all Rankings of the ranking database.

Precondition

The Ranking repository JSON files and all the default Rankings exists.

Postcondition

An ArrayList containing the Ranking names of the ranking database is returned.

Returns

ArrayList of the Ranking names of the ranking database.

Definition at line 62 of file RankingRepository.java.

```
                                                        {
        JSONObject all = this.list();
        return new ArrayList<String>(all.keySet());
    }
```

The documentation for this class was generated from the following file:

- RankingRepository.java

6.92 repository.RankingRepositoryCtrl Class Reference

Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.

Public Member Functions

- RankingRepositoryCtrl ()

  Create a RankingRepositoryCtrl instance.
- void save (JSONObject ranking)

  Save a Ranking into the ranking database.
- JSONObject get (String name)

  Get the Ranking by name from the ranking database or null if it does not exist.
- ArrayList< String > listRankings ()

  List all Rankings of the ranking database.

Private Attributes

- RankingRepository repository

  RankingRepository instance.

6.92.1 Detailed Description

Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.

See also

repository.RankingRepository

Definition at line 18 of file RankingRepositoryCtrl.java.

6.92.2 Constructor & Destructor Documentation

6.92.2.1 RankingRepositoryCtrl()

```
repository.RankingRepositoryCtrl.RankingRepositoryCtrl ( )
```

Create a RankingRepositoryCtrl instance.

Precondition

The Ranking repository JSON files and all the default Rankings exists.

Postcondition

A RankingRepositoryCtrl instance is created.

Definition at line 33 of file RankingRepositoryCtrl.java.

```
                              {
        this.repository = new RankingRepository();
    }
```

6.92.3 Member Function Documentation

6.92.3.1 save()

```
void repository.RankingRepositoryCtrl.save (
            JSONObject ranking )
```

Save a Ranking into the ranking database.

Precondition

The Ranking repository JSON files and the Ranking to be saved already exists, so it's only updated.

Postcondition

The Ranking is saved into the ranking database.

Parameters

ranking	Ranking to be saved.

Definition at line 45 of file RankingRepositoryCtrl.java.

```
                                        {
        this.repository.save(ranking);
    }
```

6.92.3.2 get()

```
JSONObject repository.RankingRepositoryCtrl.get (
              String name )
```

Get the Ranking by name from the ranking database or null if it does not exist.

Precondition

The Ranking repository JSON files and the Ranking identified by name exists.

Postcondition

A JSONObject representing the Ranking by name from the ranking database is returned or null if it does not exist.

Parameters

name	Name of the Ranking to be getted.

Returns

JSONObject that represents the Ranking by name from the ranking database or null if it does not exist.

Definition at line 56 of file RankingRepositoryCtrl.java.

```
                                      {
        return this.repository.get(name);
    }
```

6.92.3.3 listRankings()

```
ArrayList<String> repository.RankingRepositoryCtrl.listRankings ( )
```

List all Rankings of the ranking database.

Precondition

The Ranking repository JSON files and all the default Rankings exists.

Postcondition

An ArrayList containing the Ranking names of the ranking database is returned.

Returns

ArrayList of the Ranking names of the ranking database.

Definition at line 66 of file RankingRepositoryCtrl.java.

```
                                          {
        return this.repository.listRankings();
    }
```

6.92.4 Member Data Documentation

6.92.4.1 repository

`RankingRepository repository.RankingRepositoryCtrl.repository [private]`

RankingRepository instance.

Definition at line 24 of file RankingRepositoryCtrl.java.

The documentation for this class was generated from the following file:

- RankingRepositoryCtrl.java

6.93 domain.Ranking.RankingType Enum Reference

Public Attributes

- MULTIPLE
- UNIQUE
- INCREMENTAL

6.93.1 Detailed Description

Definition at line 22 of file Ranking.java.

6.93.2 Member Data Documentation

6.93.2.1 MULTIPLE

`domain.Ranking.RankingType.MULTIPLE`

Definition at line 23 of file Ranking.java.

6.93.2.2 UNIQUE

`domain.Ranking.RankingType.UNIQUE`

Definition at line 23 of file Ranking.java.

6.93.2.3 INCREMENTAL

```
domain.Ranking.RankingType.INCREMENTAL
```

Definition at line 24 of file Ranking.java.

The documentation for this enum was generated from the following file:

- Ranking.java

6.94 view.RankingView Class Reference

Public Member Functions

- RankingView ()

  Class creator.
- void initialize () throws Exception

  Initialize method which is executed when the scene is shown.
- void user () throws IOException

  Event method which is executed when the User tab is clicked.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void consultRankings () throws IOException

  Event method which is executed when the Ranking consult button is clicked.
- void consultRecords () throws IOException

  Event method which is executed when the Record consult button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

Menu Play tab.

- Text consultRanking

  Ranking consult button text.
- Rectangle consultRankingButton

  Ranking consult button.
- Label currentUserName

  Current user name.
- Text consultRecord

  Records consult button text.
- Rectangle consultRecordButton

  Records consult button.
- Text logOut

  LogOut button.

6.94.1 Detailed Description

This class represents the scene controller of the Ranking Menu .

Done by Arnau Pujantell

Definition at line 22 of file RankingView.java.

6.94.2 Constructor & Destructor Documentation

6.94.2.1 RankingView()

```
view.RankingView.RankingView ( )
```

Class creator.

Definition at line 29 of file RankingView.java.

```
29                              {
30     }
```

6.94.3 Member Function Documentation

6.94.3.1 initialize()

```
void view.RankingView.initialize ( ) throws Exception
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 102 of file RankingView.java.

```
                                        {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
    }
```

6.94.3.2 user()

```
void view.RankingView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 111 of file RankingView.java.

```
                                    {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.94.3.3 bots()

```
void view.RankingView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 120 of file RankingView.java.

```
                                    {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.94.3.4 config()

`void view.RankingView.config ( ) throws IOException`

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 129 of file RankingView.java.

```
                                        {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.94.3.5 games()

`void view.RankingView.games ( ) throws IOException`

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 138 of file RankingView.java.

```
                                       {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.94.3.6 play()

`void view.RankingView.play ( ) throws IOException`

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 147 of file RankingView.java.

```
                                      {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.94.3.7 consultRankings()

`void view.RankingView.consultRankings ( ) throws IOException`

Event method which is executed when the Ranking consult button is clicked.

Precondition

True

Postcondition

The current scene is changed to RankingConsultView.

Definition at line 156 of file RankingView.java.

```
                                               {
        ViewCtrl.changeScene("template/RankingConsultView.fxml");
    }
```

6.94.3.8 consultRecords()

`void view.RankingView.consultRecords ( ) throws IOException`

Event method which is executed when the Record consult button is clicked.

Precondition

True

Postcondition

The current scene is changed to RecordConsultView.

Definition at line 165 of file RankingView.java.

```
                                              {
        ViewCtrl.changeScene("template/RecordConsultView.fxml");
    }
```

6.94.3.9 logOut()

`void view.RankingView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 174 of file RankingView.java.

```
                                       {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.94.4 Member Data Documentation

6.94.4.1 user

```
Text view.RankingView.user [private]
```

Menu User tab.

Definition at line 38 of file RankingView.java.

6.94.4.2 bots

```
Text view.RankingView.bots [private]
```

Menu Bots tab.

Definition at line 43 of file RankingView.java.

6.94.4.3 config

```
Text view.RankingView.config [private]
```

Menu Configuration tab.

Definition at line 48 of file RankingView.java.

6.94.4.4 games

```
Text view.RankingView.games [private]
```

Menu Games tab.

Definition at line 53 of file RankingView.java.

6.94.4.5 ranking

```
Text view.RankingView.ranking [private]
```

Menu Ranking tab.

Definition at line 58 of file RankingView.java.

6.94.4.6 play

```
Text view.RankingView.play  [private]
```

Menu Play tab.

Definition at line 63 of file RankingView.java.

6.94.4.7 consultRanking

```
Text view.RankingView.consultRanking  [private]
```

Ranking consult button text.

Definition at line 68 of file RankingView.java.

6.94.4.8 consultRankingButton

```
Rectangle view.RankingView.consultRankingButton  [private]
```

Ranking consult button.

Definition at line 73 of file RankingView.java.

6.94.4.9 currentUserName

```
Label view.RankingView.currentUserName  [private]
```

Current user name.

Definition at line 78 of file RankingView.java.

6.94.4.10 consultRecord

```
Text view.RankingView.consultRecord  [private]
```

Records consult button text.

Definition at line 83 of file RankingView.java.

6.94.4.11 consultRecordButton

```
Rectangle view.RankingView.consultRecordButton [private]
```

Records consult button.

Definition at line 88 of file RankingView.java.

6.94.4.12 logOut

```
Text view.RankingView.logOut [private]
```

LogOut button.

Definition at line 93 of file RankingView.java.

The documentation for this class was generated from the following file:

- RankingView.java

6.95 view.RecordConsultView Class Reference

Public Member Functions

- RecordConsultView ()
  Class creator.
- void initialize () throws Exception
  Initialize method which is executed when the scene is shown.
- void user () throws IOException
  Event method which is executed when the User tab is clicked.
- void bots () throws IOException
  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException
  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException
  Event method which is executed when the Games tab is clicked.
- void play () throws IOException
  Event method which is executed when the Play tab is clicked.
- void consultRankings () throws IOException
  Event method which is executed when the Ranking consult button is clicked.
- void consultRecords () throws IOException
  Event method which is executed when the Record consult button is clicked.
- void logOut () throws IOException
  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text consultRanking

  Ranking consult button text.
- Rectangle consultRankingButton

  Ranking consult button.
- Text consultRecord

  Records consult button text.
- Rectangle consultRecordButton

  Records consult button.
- Label currentUserName

  Current user name.
- TableView table

  Record table.
- TableColumn rankingColumn

  Ranking column.
- TableColumn valueColumn

  Value column.

6.95.1 Detailed Description

This class represents the scene controller of consult function of a record.

By Alex Rodriguez

Definition at line 30 of file RecordConsultView.java.

6.95.2 Constructor & Destructor Documentation

6.95.2.1 RecordConsultView()

```
view.RecordConsultView.RecordConsultView ( )
```

Class creator.

Definition at line 37 of file RecordConsultView.java.

```
37                                    {
38      }
```

6.95.3 Member Function Documentation

6.95.3.1 initialize()

```
void view.RecordConsultView.initialize ( ) throws Exception
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown. The columns are setted and the records are shown.

Definition at line 119 of file RecordConsultView.java.

```
                                         {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
        ArrayList<Pair<String, JSONObject» recordList = ViewCtrl.domainCtrl.listRecords();
        rankingColumn.setCellValueFactory(new PropertyValueFactory<>("first"));
        valueColumn.setCellValueFactory(new PropertyValueFactory<>("second"));
        for (Pair<String, JSONObject> record : recordList) {
            Pair<String, Integer> pairRecord = new Pair<String, Integer>(record.first,
      record.second.getInt("value"));
            table.getItems().add(pairRecord);
        }
    }
```

6.95.3.2 user()

```
void view.RecordConsultView.user ( ) throws IOException
```

Event method which is executed when the User tab is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 135 of file RecordConsultView.java.

```
                                     {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.95.3.3 bots()

```
void view.RecordConsultView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 144 of file RecordConsultView.java.

```
                                         {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.95.3.4 config()

```
void view.RecordConsultView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 153 of file RecordConsultView.java.

```
                                           {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.95.3.5 games()

```
void view.RecordConsultView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 162 of file RecordConsultView.java.

```
                                          {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.95.3.6 play()

`void view.RecordConsultView.play ( ) throws IOException`

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 171 of file RecordConsultView.java.

```
                              {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.95.3.7 consultRankings()

`void view.RecordConsultView.consultRankings ( ) throws IOException`

Event method which is executed when the Ranking consult button is clicked.

Precondition

True

Postcondition

The current scene is changed to RankingConsultView.

Definition at line 180 of file RecordConsultView.java.

```
                                          {
        ViewCtrl.changeScene("template/RankingConsultView.fxml");
    }
```

6.95.3.8 consultRecords()

`void view.RecordConsultView.consultRecords ( ) throws IOException`

Event method which is executed when the Record consult button is clicked.

Precondition

True

Postcondition

The current scene is changed to RecordConsultView.

Definition at line 189 of file RecordConsultView.java.

```
                                         {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.95.3.9 logOut()

```
void view.RecordConsultView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 198 of file RecordConsultView.java.

```
                                    {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
     ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.95.4 Member Data Documentation

6.95.4.1 user

```
Text view.RecordConsultView.user  [private]
```

Menu User tab.

Definition at line 45 of file RecordConsultView.java.

6.95.4.2 bots

```
Text view.RecordConsultView.bots  [private]
```

Menu Bots tab.

Definition at line 50 of file RecordConsultView.java.

6.95.4.3 config

```
Text view.RecordConsultView.config [private]
```

Menu Configuration tab.

Definition at line 55 of file RecordConsultView.java.

6.95.4.4 games

```
Text view.RecordConsultView.games [private]
```

Menu Games tab.

Definition at line 60 of file RecordConsultView.java.

6.95.4.5 ranking

```
Text view.RecordConsultView.ranking [private]
```

Menu Ranking tab.

Definition at line 65 of file RecordConsultView.java.

6.95.4.6 play

```
Text view.RecordConsultView.play [private]
```

Menu Play tab.

Definition at line 70 of file RecordConsultView.java.

6.95.4.7 consultRanking

```
Text view.RecordConsultView.consultRanking [private]
```

Ranking consult button text.

Definition at line 75 of file RecordConsultView.java.

6.95.4.8 consultRankingButton

```
Rectangle view.RecordConsultView.consultRankingButton  [private]
```

Ranking consult button.

Definition at line 80 of file RecordConsultView.java.

6.95.4.9 consultRecord

```
Text view.RecordConsultView.consultRecord  [private]
```

Records consult button text.

Definition at line 85 of file RecordConsultView.java.

6.95.4.10 consultRecordButton

```
Rectangle view.RecordConsultView.consultRecordButton  [private]
```

Records consult button.

Definition at line 90 of file RecordConsultView.java.

6.95.4.11 currentUserName

```
Label view.RecordConsultView.currentUserName  [private]
```

Current user name.

Definition at line 95 of file RecordConsultView.java.

6.95.4.12 table

```
TableView view.RecordConsultView.table  [private]
```

Record table.

Definition at line 100 of file RecordConsultView.java.

6.95.4.13 rankingColumn

```
TableColumn view.RecordConsultView.rankingColumn  [private]
```

Ranking column.

Definition at line 105 of file RecordConsultView.java.

6.95.4.14 valueColumn

```
TableColumn view.RecordConsultView.valueColumn  [private]
```

Value column.

Definition at line 110 of file RecordConsultView.java.

The documentation for this class was generated from the following file:

- RecordConsultView.java

6.96 repository.Repository Class Reference

Implements various CRUD operations to work with the local file system JSON databases and TXT fixtures. By Alex Rodriguez.

Classes

- enum RepositoryType

  Different types for the accessed repository.

Protected Member Functions

- Repository (RepositoryType repositoryType)

  Create a Repository instance.
- JSONObject list ()

  Obtain all entries of the database. For JSON repositories.
- JSONObject get (String key)

  Obtain an entry of the database by key. For JSON repositories.
- JSONObject createOrUpdate (String key, JSONObject value)

  Create an entry in the database by key or update it if it does exist. For JSON repositories.
- JSONObject remove (String key)

  Remove an entry in the database by key if it does exist. For JSON repositories.

Protected Attributes

- String path

  Relative path of the accessed repository.
- RepositoryType type

  Type of the accessed repository.

Static Private Attributes

- static final String databasesPath = "./res/databases/"

  Relative root path of the local JSON databases.
- static final String fixturesPath = "./res/fixtures/"

  Relative root path of the local TXT fixtures.

6.96.1 Detailed Description

Implements various CRUD operations to work with the local file system JSON databases and TXT fixtures. By Alex Rodriguez.

Definition at line 22 of file Repository.java.

6.96.2 Constructor & Destructor Documentation

6.96.2.1 Repository()

```
repository.Repository.Repository (
            RepositoryType repositoryType )  [protected]
```

Create a Repository instance.

Precondition

The accessed repository JSON or TXT files exists.

Postcondition

A Repository instance of the given type is created and binded with the correct path.

Parameters

repositoryType	type of the accessed repository.

Definition at line 62 of file Repository.java.

```
62                                                         {
        String realPath = "";
        try {
            realPath =
    Paths.get(Repository.class.getProtectionDomain().getCodeSource().getLocation().toURI().getPath())
                    .getParent().toString();
        } catch (Exception e) {
            e.printStackTrace();
        }

        this.type = repositoryType;
        switch (repositoryType) {
            case CONFIGURATION:
                this.path = Paths.get(realPath, Repository.databasesPath,
    "configurations.json").toString();
                break;
            case GAME:
                this.path = Paths.get(realPath, Repository.databasesPath, "games.json").toString();
                break;
            case PLAYER:
                this.path = Paths.get(realPath, Repository.databasesPath, "players.json").toString();
                break;
            case RANKING:
                this.path = Paths.get(realPath, Repository.databasesPath, "rankings.json").toString();
                break;
            case FIXTURE:
                this.path = Paths.get(realPath, Repository.fixturesPath).toString();
                break;
            default:
                this.path = null;
        }
    }
```

6.96.3 Member Function Documentation

6.96.3.1 list()

`JSONObject repository.Repository.list ( ) [protected]`

Obtain all entries of the database. For JSON repositories.

Precondition

The accessed repository JSON or TXT files exists.

Postcondition

A JSONObject representing the accessed database is returned.

Returns

JSONObject that represents the accessed database.

Definition at line 102 of file Repository.java.

```
                                    {
        JSONObject database = new JSONObject();

        try {
            File file = new File(this.path);
            InputStream reader = new FileInputStream(file);
            JSONTokener tokener = new JSONTokener(reader);
            database = new JSONObject(tokener);
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return database;
    }
```

6.96.3.2 get()

```
JSONObject repository.Repository.get (
            String key )  [protected]
```

Obtain an entry of the database by key. For JSON repositories.

Precondition

The accessed repository JSON or TXT files exists.

Postcondition

A JSONObject representing the key entry of the accessed database is returned or null if it does not exist.

Parameters

key	Key of the entry in the accessed database.

Returns

JSONObject that represents the key entry of the accessed database or null if it does not exist.

Reimplemented in repository.RankingRepository, and repository.PlayerRepository.

Definition at line 126 of file Repository.java.

```
                                      {
        JSONObject database = this.list();
        return database.optJSONObject(key);
    }
```

6.96.3.3 createOrUpdate()

```
JSONObject repository.Repository.createOrUpdate (
            String key,
            JSONObject value )  [protected]
```

Create an entry in the database by key or update it if it does exist. For JSON repositories.

Precondition

The accessed repository JSON or TXT files exists.

Postcondition

The key entry is created in the accessed database or it is updated if it already exists. A JSONObject representing the accessed database is returned.

Parameters

key	Key of the entry in the accessed database.
value	Value to be inserted in the accessed database by the key.

Returns

JSONObject that represents the accessed database.

Definition at line 140 of file Repository.java.

```
                                                                          {
        JSONObject database = this.list();
        database.put(key, value);

        try {
            File file = new File(this.path);
            FileWriter writer = new FileWriter(file);
            database.write(writer, 2, 0);
            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return database;
    }
```

6.96.3.4 remove()

```
JSONObject repository.Repository.remove (
              String key )  [protected]
```

Remove an entry in the database by key if it does exist. For JSON repositories.

Precondition

The accessed repository JSON or TXT files exists.

Postcondition

The key entry is removed in the accessed database if it does exist. A JSONObject representing the accessed database is returned.

Parameters

key	Key of the entry in the accessed database.

Returns

JSONObject that represents the accessed database.

Definition at line 164 of file Repository.java.

```
                                                {
        JSONObject database = this.list();
        database.remove(key);

        try {
            File file = new File(this.path);
            FileWriter writer = new FileWriter(file);
            database.write(writer, 2, 0);
            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return database;
    }
```

6.96.4 Member Data Documentation

6.96.4.1 databasesPath

`final String repository.Repository.databasesPath = "./res/databases/" [static], [private]`

Relative root path of the local JSON databases.

Definition at line 35 of file Repository.java.

6.96.4.2 fixturesPath

`final String repository.Repository.fixturesPath = "./res/fixtures/" [static], [private]`

Relative root path of the local TXT fixtures.

Definition at line 40 of file Repository.java.

6.96.4.3 path

`String repository.Repository.path [protected]`

Relative path of the accessed repository.

Definition at line 47 of file Repository.java.

6.96.4.4 type

`RepositoryType repository.Repository.type [protected]`

Type of the accessed repository.

Definition at line 52 of file Repository.java.

The documentation for this class was generated from the following file:

- Repository.java

6.97 repository.Repository.RepositoryType Enum Reference

Different types for the accessed repository.

Public Attributes

- CONFIGURATION
- GAME
- PLAYER
- RANKING
- FIXTURE

6.97.1 Detailed Description

Different types for the accessed repository.

Definition at line 26 of file Repository.java.

6.97.2 Member Data Documentation

6.97.2.1 CONFIGURATION

```
repository.Repository.RepositoryType.CONFIGURATION
```

Definition at line 27 of file Repository.java.

6.97.2.2 GAME

```
repository.Repository.RepositoryType.GAME
```

Definition at line 27 of file Repository.java.

6.97.2.3 PLAYER

```
repository.Repository.RepositoryType.PLAYER
```

Definition at line 27 of file Repository.java.

6.97.2.4 RANKING

```
repository.Repository.RepositoryType.RANKING
```

Definition at line 27 of file Repository.java.

6.97.2.5 FIXTURE

```
repository.Repository.RepositoryType.FIXTURE
```

Definition at line 28 of file Repository.java.

The documentation for this enum was generated from the following file:

- Repository.java

6.98 view.SignUpView Class Reference

Public Member Functions

- SignUpView ()

  Class creator.
- void signUp () throws IOException

  Event method which is executed when the signUp button is clicked.
- void logIn () throws IOException

  Event method which is executed when the logIn button is clicked.

Private Attributes

- Text logIn

  logIn view change button.
- Text signUp

  signUp view change button.
- TextField nusername

  New user name text field.
- PasswordField npassword

  New password field.
- PasswordField rpassword

  Repeat password field.
- Label signUpResult

  Exception output message label.
- Text signUp2

  signUp button text.
- Rectangle signUpButton

  signUp button.

6.98.1 Detailed Description

This class represents the scene controller of the SignUp.

Done by Arnau Pujantell

Definition at line 23 of file SignUpView.java.

6.98.2 Constructor & Destructor Documentation

6.98.2.1 SignUpView()

```
view.SignUpView.SignUpView ( )
```

Class creator.

Definition at line 30 of file SignUpView.java.

```
                          {
    }
```

6.98.3 Member Function Documentation

6.98.3.1 signUp()

```
void view.SignUpView.signUp ( ) throws IOException
```

Event method which is executed when the signUp button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, scene changes to LogInView.

Definition at line 83 of file SignUpView.java.

```
                                         {
        Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createUser(nusername.getText(),
    npassword.getText(),
                rpassword.getText());
        if (result.second != null) {
            switch (result.second) {
                case "ERR_INVALID_NAME":
                    signUpResult.setText("Username can’t be empty!");
                    break;
                case "ERR_INVALID_PASSWORD":
                    signUpResult.setText("Password can’t be empty!");
                    break;
                case "ERR_EXISTING_PLAYER":
                    signUpResult.setText("The username is already taken!");
                    break;
                case "ERR_BAD_CONFIRMATION":
                    signUpResult.setText("Confirmation password doesn’t match!");
                    break;
                default:
                    signUpResult.setText("Something went wrong, try again!");
                    break;
            }
        } else {
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.98.3.2 logIn()

```
void view.SignUpView.logIn ( ) throws IOException
```

Event method which is executed when the logIn button is clicked.

Precondition

True

Postcondition

Scene changes to logInView.

Definition at line 114 of file SignUpView.java.

```
                                        {
        ViewCtrl.changeScene("template/LogInView.fxml");
    }
```

6.98.4 Member Data Documentation

6.98.4.1 logIn

```
Text view.SignUpView.logIn  [private]
```

logIn view change button.

Definition at line 39 of file SignUpView.java.

6.98.4.2 signUp

```
Text view.SignUpView.signUp  [private]
```

signUp view change button.

Definition at line 44 of file SignUpView.java.

6.98.4.3 nusername

```
TextField view.SignUpView.nusername  [private]
```

New user name text field.

Definition at line 49 of file SignUpView.java.

6.98.4.4 npassword

```
PasswordField view.SignUpView.npassword  [private]
```

New password field.

Definition at line 54 of file SignUpView.java.

6.98.4.5 rpassword

```
PasswordField view.SignUpView.rpassword  [private]
```

Repeat password field.

Definition at line 59 of file SignUpView.java.

6.98.4.6 signUpResult

```
Label view.SignUpView.signUpResult  [private]
```

Exception output message label.

Definition at line 64 of file SignUpView.java.

6.98.4.7 signUp2

```
Text view.SignUpView.signUp2  [private]
```

signUp button text.

Definition at line 69 of file SignUpView.java.

6.98.4.8 signUpButton

```
Rectangle view.SignUpView.signUpButton  [private]
```

signUp button.

Definition at line 74 of file SignUpView.java.

The documentation for this class was generated from the following file:

- SignUpView.java

6.99 domain.HardDifficulty.TreeNode Class Reference

Public Member Functions

- TreeNode (PieceType rootType, PieceType pieceType, Board currentBoard, boolean canEatHorizontally, boolean canEatVertically, boolean canEatDiagonally, Pair< Integer, Integer > selectedPosition)

  Create a TreeNode instance.
- boolean isLeaf ()

  Returns whether the implicit TreeNode has possible future moves or not.
- Pair< Integer, Integer > getSelectedPosition ()

  Returns the selectedPosition attribute of the implicit TreeNode. Since the initial board state doesn't have a selected←Position, this method can return null.
- double getWinRatio ()

  Returns the win ratio of a TreeNode, which is the result of the division of attribute totValue, which represents the number of wins in the implicit TreeNode and the attribute nVisits, which represents the number of times the implicit TreeNode has been visited. Since nVisits is initialized with value 0 we use the attribute epsilon to prevent division by 0.
- ArrayList< TreeNode > getChildren ()

  Returns the implicit TreeNode's private attribute children, which represents future board states obtained from the current state.
- void play ()

  Simulation of a game used as the basis of the Monte Carlo Tree Search algorithm. Given the root TreeNode of the stats tree, which represents the initial board state, it traverses the tree using the UCT formula to select, in every TreeNode, which of its future states is best. Once it reaches an unexplored TreeNode, it generates its children Tree←Nodes and picks the best out of them using the UCT formula once more. After this, based on the number of pieces of each player, it returns whether there has been a win or not, and every single TreeNode that was traversed to get to that state is updated based on the outcome.

Private Member Functions

- TreeNode select ()

  Method that gets the best move to play out of the implicit TreeNode's children attribute. This is done using the UCT formula to compare each TreeNode and get the best one of them. UCT takes into consideration the percentage of wins of the TreeNode and if it has been explored very few times. In the case of a tie between two different candidates, the random attribute is used to break the tie. Since a TreeNode could have no possible future states, this method can return null.
- void expand ()

  Generates the next board states of a game given the implicit TreeNode's currentBoard and saves them in the implicit TreeNode's children attribute. Since a board state of a TreeNode could have no possible future moves, then it could occur that children is left unchanged after the method.
- double rollOut ()

  Returns whether the implicit TreeNode's currentBoard would result in a win or a loss for the indicated player. The indicated player is known using the rootType attribute.
- void updateStats (double value)

  Updates information on the stats tree when a simulation is finished.

Private Attributes

- ArrayList< TreeNode > children

  Possible future moves and board states that can be obtained given the currrent board state.
- double nVisits

  Number of times a TreeNode has been traversed.

- double totValue

    Number of wins obtained in this TreeNode.
- Pair< Integer, Integer > selectedPosition

    Move that produces the current board state. It can be null since you can have the first board state, which isn't produced by a move.
- PieceType pieceType

    PieceType used to identify whose turn to make a move is between Player1 (White) and Player2 (Black)
- PieceType rootType

    PieceType used to identify whose turn it is in the initial board state.
- Board currentBoard

    Current board state in an instance of a game.
- boolean canEatHorizontally

    Whether the pieces of the current Game can be eaten horizontally.
- boolean canEatVertically

    Whether the pieces of the current Game can be eaten vertically.
- boolean canEatDiagonally

    Whether the pieces of the current Game can be eaten diagonally.

6.99.1 Detailed Description

Definition at line 87 of file HardDifficulty.java.

6.99.2 Constructor & Destructor Documentation

6.99.2.1 TreeNode()

```
domain.HardDifficulty.TreeNode.TreeNode (
            PieceType rootType,
            PieceType pieceType,
            Board currentBoard,
            boolean canEatHorizontally,
            boolean canEatVertically,
            boolean canEatDiagonally,
            Pair< Integer, Integer > selectedPosition )
```

Create a TreeNode instance.

Precondition

The given rules are not all false.

Postcondition

A TreeNode instance is created and its implicits rootType, pieceType, currentBoard, canEatHorizontally, can↩ EatVertically, canEatDiagonally and selectedPosition attributes are setted.

Parameters

rootType	PieceType of the root TreeNode of a tree.
pieceType	PieceType used to know whose turn it is to make a move.
currentBoard	Current state of the board.
canEatHorizontally	Whether the pieces of the current Game can be eaten horizontally.
canEatVertically	Whether the pieces of the current Game can be eaten vertically.
canEatDiagonally	Whether the pieces of the current Game can be eaten diagonally.
selectedPosition	Position in the board that resulted in the current board state (it can be null).

Definition at line 145 of file HardDifficulty.java.

```
    {
            this.rootType = rootType;
            this.pieceType = pieceType;
            this.currentBoard = currentBoard;
            this.canEatHorizontally = canEatHorizontally;
            this.canEatVertically = canEatVertically;
            this.canEatDiagonally = canEatDiagonally;
            this.children = new ArrayList<TreeNode> ();
            this.nVisits = 0;
            this.totValue = 0;
            this.selectedPosition = selectedPosition;
        }
```

6.99.3 Member Function Documentation

6.99.3.1 isLeaf()

`boolean domain.HardDifficulty.TreeNode.isLeaf ( )`

Returns whether the implicit TreeNode has possible future moves or not.

Precondition

True

Postcondition

A boolean which has value true if the implicit TreeNode has future moves or false otherwise is returned.

Returns

Boolean which tells whether the children attribute of the implicit TreeNode is empty or not.

Definition at line 167 of file HardDifficulty.java.

```
                            {
            return this.children.isEmpty();
        }
```

6.99.3.2 getSelectedPosition()

```
Pair<Integer, Integer> domain.HardDifficulty.TreeNode.getSelectedPosition ( )
```

Returns the selectedPosition attribute of the implicit TreeNode. Since the initial board state doesn't have a selectedPosition, this method can return null.

Precondition

True

Postcondition

The implicit TreeNode's selectedPosition, which can be either a position of the board or null, is returned.

Returns

Pair of Integers which represents a position inside of a board.

Definition at line 178 of file HardDifficulty.java.

```
                                                         {
            return this.selectedPosition;
        }
```

6.99.3.3 getWinRatio()

```
double domain.HardDifficulty.TreeNode.getWinRatio ( )
```

Returns the win ratio of a TreeNode, which is the result of the division of attribute totValue, which represents the number of wins in the implicit TreeNode and the attribute nVisits, which represents the number of times the implicit TreeNode has been visited. Since nVisits is initialized with value 0 we use the attribute epsilon to prevent division by 0.

Precondition

True

Postcondition

The implicit TreeNode's win ratio is returned.

Returns

Double equal to the division between totValue and nVisits of the implicit TreeNode.

Definition at line 191 of file HardDifficulty.java.

```
                                   {
            return (this.totValue/(this.nVisits + HardDifficulty.epsilon));
        }
```

6.99.3.4 getChildren()

`ArrayList<TreeNode> domain.HardDifficulty.TreeNode.getChildren ( )`

Returns the implicit TreeNode's private attribute children, which represents future board states obtained from the current state.

Precondition

True

Postcondition

The implicit TreeNode's children attribute is returned.

Returns

ArrayList which acts as a representation of the possible future states of a board.

Definition at line 201 of file HardDifficulty.java.

```
                                                {
            return this.children;
        }
```

6.99.3.5 play()

`void domain.HardDifficulty.TreeNode.play ( )`

Simulation of a game used as the basis of the Monte Carlo Tree Search algorithm. Given the root TreeNode of the stats tree, which represents the initial board state, it traverses the tree using the UCT formula to select, in every TreeNode, which of its future states is best. Once it reaches an unexplored TreeNode, it generates its children TreeNodes and picks the best out of them using the UCT formula once more. After this, based on the number of pieces of each player, it returns whether there has been a win or not, and every single TreeNode that was traversed to get to that state is updated based on the outcome.

Precondition

True

Postcondition

The simulation of the game is done and the tree is updated based on the outcome of that simulation.

Definition at line 215 of file HardDifficulty.java.

```
                          {
            List<TreeNode> visited = new LinkedList<TreeNode>();
            TreeNode current = this;
            visited.add(this);

            while (!current.isLeaf()) {
                current = current.select();
                visited.add(current);
            }

            current.expand();

            if(current.isLeaf()) {
                double value = current.rollOut();
                for(TreeNode node : visited) node.updateStats(value);
            }

            else {
                TreeNode bestChild = current.select();
                visited.add(bestChild);
                double value = bestChild.rollOut();
                for (TreeNode node : visited) node.updateStats(value);
            }
        }
```

6.99.3.6 select()

```
TreeNode domain.HardDifficulty.TreeNode.select ( )  [private]
```

Method that gets the best move to play out of the implicit TreeNode's children attribute. This is done using the UCT formula to compare each TreeNode and get the best one of them. UCT takes into consideration the percentage of wins of the TreeNode and if it has been explored very few times. In the case of a tie between two different candidates, the random attribute is used to break the tie. Since a TreeNode could have no possible future states, this method can return null.

Precondition

True

Postcondition

The best Node of the next board states of the implicit TreeNode or null is returned.

Returns

TreeNode with the best TreeNode value based on the UCT formula out of all the TreeNodes in attribute children of the implicit TreeNode.

Definition at line 251 of file HardDifficulty.java.

```
                                        {
            TreeNode selected = null;
            double bestValue = Double.NEGATIVE_INFINITY;

            for (TreeNode child : this.children) {
                double uctValue = child.totValue / (child.nVisits + HardDifficulty.epsilon) +
                            Math.sqrt(Math.log(this.nVisits+1) / (child.nVisits +
        HardDifficulty.epsilon)) +
                            HardDifficulty.random.nextDouble() * HardDifficulty.epsilon;

                if (uctValue > bestValue) {
                    bestValue = uctValue;
                    selected = child;
                }
            }

            return selected;
        }
```

6.99.3.7 expand()

```
void domain.HardDifficulty.TreeNode.expand ( )  [private]
```

Generates the next board states of a game given the implicit TreeNode's currentBoard and saves them in the implicit TreeNode's children attribute. Since a board state of a TreeNode could have no possible future moves, then it could occur that children is left unchanged after the method.

Precondition

True

Postcondition

If a board state has next states that can be obtained, TreeNodes that represent them will be generated and saved in the implicit TreeNode's children attribute. If that isn't the case, then children remains the same as before calling this function.

Definition at line 278 of file HardDifficulty.java.

```
                                {
            ArrayList<Pair<Integer,Integer» validPositions = this.currentBoard.validPositions(
                this.pieceType, this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);

            for (int i = 0; i < validPositions.size(); ++i) {
                Board board = new Board(this.currentBoard.getBoard());
                board.placePiece(validPositions.get(i), this.pieceType, this.canEatHorizontally,
                    this.canEatVertically, this.canEatDiagonally);

                this.children.add(i, new TreeNode(this.rootType,
       HardDifficulty.inversePieceType(this.pieceType), board, this.canEatHorizontally,
                    this.canEatVertically, this.canEatDiagonally, validPositions.get(i)));
            }
        }
```

6.99.3.8 rollOut()

`double domain.HardDifficulty.TreeNode.rollOut ( )  [private]`

Returns whether the implicit TreeNode's currentBoard would result in a win or a loss for the indicated player. The indicated player is known using the rootType attribute.

Precondition

True

Postcondition

If rootType is equal to PLAYER1 and the number of pieces of PLAYER1 is greater than the number of pieces of PLAYER2 this method returns 1, if not it returns 0. If instead rootType is equal to PLAYER2 and the number of pieces of PLAYER2 is greater than the number of pieces of PLAYER1 it returns 1, and if not it returns 0.

Returns

Double which is equal to 1 if the rootType's number of pieces is greater than the opposing PieceType's number of pieces, and 0 otherwise.

Definition at line 301 of file HardDifficulty.java.

```
                                   {
            int piecesPlayer1 = this.currentBoard.getPiecesPlayer1();
            int piecesPlayer2 = this.currentBoard.getPiecesPlayer2();
            if(this.rootType == PieceType.PLAYER1) {
                if(piecesPlayer1 > piecesPlayer2) return 1;
                else return 0;
            }
            else {
                if(piecesPlayer2 > piecesPlayer1) return 1;
                else return 0;
            }
        }
```

6.99.3.9 updateStats()

```
void domain.HardDifficulty.TreeNode.updateStats (
             double value )  [private]
```

Updates information on the stats tree when a simulation is finished.

Precondition

True

Postcondition

For every single one of the TreeNodes traversed to get to the ending of a simulation, its number of visits is increased by 1 and its number of wins changes based on the parameter value.

Parameters

value	Double which equals either 1 or 0 and represents whether the final board state of a simulation ended in a victory or in a loss.

Definition at line 322 of file HardDifficulty.java.

```
                                             {
                ++this.nVisits;
                this.totValue += value;
            }
```

6.99.4 Member Data Documentation

6.99.4.1 children

```
ArrayList<TreeNode> domain.HardDifficulty.TreeNode.children  [private]
```

Possible future moves and board states that can be obtained given the currrent board state.

Definition at line 93 of file HardDifficulty.java.

6.99.4.2 nVisits

```
double domain.HardDifficulty.TreeNode.nVisits  [private]
```

Number of times a TreeNode has been traversed.

Definition at line 97 of file HardDifficulty.java.

6.99.4.3 totValue

```
double domain.HardDifficulty.TreeNode.totValue [private]
```

Number of wins obtained in this TreeNode.

Definition at line 101 of file HardDifficulty.java.

6.99.4.4 selectedPosition

```
Pair<Integer, Integer> domain.HardDifficulty.TreeNode.selectedPosition [private]
```

Move that produces the current board state. It can be null since you can have the first board state, which isn't produced by a move.

Definition at line 105 of file HardDifficulty.java.

6.99.4.5 pieceType

```
PieceType domain.HardDifficulty.TreeNode.pieceType [private]
```

PieceType used to identify whose turn to make a move is between Player1 (White) and Player2 (Black)

Definition at line 109 of file HardDifficulty.java.

6.99.4.6 rootType

```
PieceType domain.HardDifficulty.TreeNode.rootType [private]
```

PieceType used to identify whose turn it is in the initial board state.

Definition at line 113 of file HardDifficulty.java.

6.99.4.7 currentBoard

```
Board domain.HardDifficulty.TreeNode.currentBoard [private]
```

Current board state in an instance of a game.

Definition at line 117 of file HardDifficulty.java.

6.99.4.8 canEatHorizontally

```
boolean domain.HardDifficulty.TreeNode.canEatHorizontally  [private]
```

Whether the pieces of the current Game can be eaten horizontally.

Definition at line 121 of file HardDifficulty.java.

6.99.4.9 canEatVertically

```
boolean domain.HardDifficulty.TreeNode.canEatVertically  [private]
```

Whether the pieces of the current Game can be eaten vertically.

Definition at line 125 of file HardDifficulty.java.

6.99.4.10 canEatDiagonally

```
boolean domain.HardDifficulty.TreeNode.canEatDiagonally  [private]
```

Whether the pieces of the current Game can be eaten diagonally.

Definition at line 129 of file HardDifficulty.java.

The documentation for this class was generated from the following file:

- HardDifficulty.java

6.100 cmd.driver.user Class Reference

User driver entrypoint. By Alex Rodriguez.

Static Public Member Functions

- static void main (String[ ] args)
  User driver main function. Creates an instance of the User driver and starts it.

6.100.1 Detailed Description

User driver entrypoint. By Alex Rodriguez.

Definition at line 15 of file user.java.

6.100.2 Member Function Documentation

6.100.2.1 main()

```
static void cmd.driver.user.main (
             String[] args )  [static]
```

User driver main function. Creates an instance of the User driver and starts it.

Precondition

True.

Postcondition

The User driver has started.

Definition at line 22 of file user.java.

```
                                        {
        new UserDriver().start();
    }
```

The documentation for this class was generated from the following file:

- user.java

6.101 domain.User Class Reference

Represents a human user in our system.

Public Member Functions

- User (String name, String password, UUID id)
  Creator that, given a username 'name', a password 'password' and an id 'id'.
- User (JSONObject user)
  Creator that, given a JSONObject user, deserializes it.
- JSONObject serialize ()
  Creator that serializes the current object to a JSON Object.
- String getPassword ()
  Consultant that returns the implicit parameter's password.
- void setPassword (String password) throws InvalidPasswordException
  Modifier that, given a password, changes the implicit parameter's password for a new password 'password'.

Private Attributes

- String password
  User's password.

Additional Inherited Members

6.101.1 Detailed Description

Represents a human user in our system.

Done by Arnau Pujantell

Subclass that represents a human. It contains a password.

Definition at line 21 of file User.java.

6.101.2 Constructor & Destructor Documentation

6.101.2.1 User() [1/2]

```
domain.User.User (
             String name,
             String password,
             UUID id )
```

Creator that, given a username 'name', a password 'password' and an id 'id'.

CREATORS

Precondition

None of the elements is null

Postcondition

It creates a new User with name 'name', password 'password', id 'id', type 'USER' and isDeleted as 'false'.

Definition at line 32 of file User.java.

```
    {
        this.name = name;
        this.password = password;
        this.id = id;
        this.isDeleted = false;
    }
```

6.101.2.2 User() [2/2]

```
domain.User.User (
              JSONObject user )
```

Creator that, given a JSONObject user, deserializes it.

Precondition

user is not null

Postcondition

user is not a JSONObject anymore

Definition at line 44 of file User.java.

```
                                 {
          this.name = user.getString("name");
          this.id = UUID.fromString(user.getString("id"));
          this.password = user.getString("password");
          this.isDeleted = user.getBoolean("is_deleted");
      }
```

6.101.3 Member Function Documentation

6.101.3.1 serialize()

```
JSONObject domain.User.serialize ( )
```

Creator that serializes the current object to a JSON Object.

Precondition

True

Postcondition

The current user becomes a JSON Object

Definition at line 55 of file User.java.

```
                                  {
          JSONObject user = new JSONObject();
          user.put("name", this.name);
          user.put("id", this.id.toString());
          user.put("password", this.password);
          user.put("type", "USER");
          user.put("is_deleted", this.isDeleted);

          return user;
      }
```

6.101.3.2 getPassword()

```
String domain.User.getPassword ( )
```

Consultant that returns the implicit parameter's password.

CONSULTANTS

Precondition

True

Postcondition

Implicit parameter's password is returned.

Definition at line 73 of file User.java.

```
                                    {
        return this.password;
    }
```

6.101.3.3 setPassword()

```
void domain.User.setPassword (
             String password ) throws InvalidPasswordException
```

Modifier that, given a password, changes the implicit parameter's password for a new password 'password'.

MODIFIERS

Precondition

password is not null

Postcondition

Implicit parameter's password has been changed.

Definition at line 84 of file User.java.

```
                                                                             {
        if(password.isBlank()) {
            throw new InvalidPasswordException();
        }
        this.password = password;
    }
```

6.101.4 Member Data Documentation

6.101.4.1 password

```
String domain.User.password  [private]
```

User's password.

Definition at line 23 of file User.java.

The documentation for this class was generated from the following file:

- User.java

6.102 view.UserDeleteView Class Reference

Public Member Functions

- UserDeleteView ()

  Class creator.
- void initialize () throws Exception

  Initialize method which is executed when the scene is shown.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void modifyUser () throws IOException

  Event method which is executed when the modifyUser button is clicked.
- void deleteUser () throws IOException

  Event method which is executed when the deleteUser button is clicked.
- void deleteUserConfirm () throws IOException

  Event method which is executed when the delete button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text modifyUser

  User modify button text.
- Rectangle modifyUserButton

  User modify button.
- Text deleteUser

  User modify button text.
- Rectangle deleteUserButton

  User modify button.
- PasswordField password

  New User password field.
- Label deleteUserResult

  Exception output message label.
- Text deleteUserConfirm

  User delete confirm text button.
- Rectangle deleteUserConfirmButton

  User delete confirm button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.102.1 Detailed Description

This class represents the scene controller of delete function of a user.

Done by Arnau Pujantell

Definition at line 23 of file UserDeleteView.java.

6.102.2 Constructor & Destructor Documentation

6.102.2.1 UserDeleteView()

```
view.UserDeleteView.UserDeleteView ( )
```

Class creator.

Definition at line 30 of file UserDeleteView.java.

```
                             {
    }
```

6.102.3 Member Function Documentation

6.102.3.1 initialize()

```
void view.UserDeleteView.initialize ( ) throws Exception
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 123 of file UserDeleteView.java.

```
                                          {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
    }
```

6.102.3.2 bots()

```
void view.UserDeleteView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 132 of file UserDeleteView.java.

```
                                      {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.102.3.3 config()

```
void view.UserDeleteView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 141 of file UserDeleteView.java.

```
                                          {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.102.3.4 games()

```
void view.UserDeleteView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 150 of file UserDeleteView.java.

```
                                         {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.102.3.5 ranking()

```
void view.UserDeleteView.ranking ( ) throws IOException
```

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 159 of file UserDeleteView.java.

```
                                           {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.102.3.6 play()

```
void view.UserDeleteView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 168 of file UserDeleteView.java.

```
                                    {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.102.3.7 modifyUser()

```
void view.UserDeleteView.modifyUser ( ) throws IOException
```

Event method which is executed when the modifyUser button is clicked.

Precondition

True

Postcondition

Scene changes to UserModifyView.

Definition at line 177 of file UserDeleteView.java.

```
                                          {
        ViewCtrl.changeScene("template/UserModifyView.fxml");
    }
```

6.102.3.8 deleteUser()

```
void view.UserDeleteView.deleteUser ( ) throws IOException
```

Event method which is executed when the deleteUser button is clicked.

Precondition

True

Postcondition

Scene changes to UserModifyView.

Definition at line 186 of file UserDeleteView.java.

```
                                          {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.102.3.9 deleteUserConfirm()

`void view.UserDeleteView.deleteUserConfirm ( ) throws IOException`

Event method which is executed when the delete button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the current user is deleted. Finally, scene changes to LogInView.

Definition at line 195 of file UserDeleteView.java.

```
                                                    {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "Your account will be deleted. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            String result = ViewCtrl.domainCtrl.deleteUser(password.getText());
            if (result != null) {
                switch (result) {
                    case "ERR_INCORRECT_CREDENTIALS":
                        deleteUserResult.setText("Wrong password!");
                        break;
                    case "ERR_INEXISTING_PLAYER":
                        deleteUserResult.setText("The player doesn't exist!");
                        break;
                    default:
                        deleteUserResult.setText("Something went wrong, try again!");
                        break;
                }
            }
            else {
                ViewCtrl.changeScene("template/LogInView.fxml");
            }
        }
    }
```

6.102.3.10 logOut()

`void view.UserDeleteView.logOut ( ) throws IOException`

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 225 of file UserDeleteView.java.

```
                                         {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
    ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            ViewCtrl.domainCtrl.logout();
            ViewCtrl.changeScene("template/LogInView.fxml");
        }
    }
```

6.102.4 Member Data Documentation

6.102.4.1 user

```
Text view.UserDeleteView.user  [private]
```

Menu User tab.

Definition at line 39 of file UserDeleteView.java.

6.102.4.2 bots

```
Text view.UserDeleteView.bots  [private]
```

Menu Bots tab.

Definition at line 44 of file UserDeleteView.java.

6.102.4.3 config

```
Text view.UserDeleteView.config  [private]
```

Menu Configuration tab.

Definition at line 49 of file UserDeleteView.java.

6.102.4.4 games

```
Text view.UserDeleteView.games  [private]
```

Menu Games tab.

Definition at line 54 of file UserDeleteView.java.

6.102.4.5 ranking

```
Text view.UserDeleteView.ranking  [private]
```

Menu Ranking tab.

Definition at line 59 of file UserDeleteView.java.

6.102.4.6 play

```
Text view.UserDeleteView.play  [private]
```

Menu Play tab.

Definition at line 64 of file UserDeleteView.java.

6.102.4.7 modifyUser

```
Text view.UserDeleteView.modifyUser  [private]
```

User modify button text.

Definition at line 69 of file UserDeleteView.java.

6.102.4.8 modifyUserButton

```
Rectangle view.UserDeleteView.modifyUserButton  [private]
```

User modify button.

Definition at line 74 of file UserDeleteView.java.

6.102.4.9 deleteUser

```
Text view.UserDeleteView.deleteUser  [private]
```

User modify button text.

Definition at line 79 of file UserDeleteView.java.

6.102.4.10 deleteUserButton

```
Rectangle view.UserDeleteView.deleteUserButton  [private]
```

User modify button.

Definition at line 84 of file UserDeleteView.java.

6.102.4.11 password

```
PasswordField view.UserDeleteView.password  [private]
```

New User password field.

Definition at line 89 of file UserDeleteView.java.

6.102.4.12 deleteUserResult

```
Label view.UserDeleteView.deleteUserResult  [private]
```

Exception output message label.

Definition at line 94 of file UserDeleteView.java.

6.102.4.13 deleteUserConfirm

```
Text view.UserDeleteView.deleteUserConfirm  [private]
```

User delete confirm text button.

Definition at line 99 of file UserDeleteView.java.

6.102.4.14 deleteUserConfirmButton

```
Rectangle view.UserDeleteView.deleteUserConfirmButton  [private]
```

User delete confirm button.

Definition at line 104 of file UserDeleteView.java.

6.102.4.15 currentUserName

```
Label view.UserDeleteView.currentUserName  [private]
```

Current user name.

Definition at line 109 of file UserDeleteView.java.

6.102.4.16 logOut

```
Text view.UserDeleteView.logOut  [private]
```

LogOut button.

Definition at line 114 of file UserDeleteView.java.

The documentation for this class was generated from the following file:

- UserDeleteView.java

6.103 test.driver.UserDriver Class Reference

Public Member Functions

- UserDriver ()
- void start ()

Public Attributes

- User currentUser

Private Member Functions

- void mainMenu ()
- void createUser ()
- void serialize ()
- void deserialize ()
- void getName ()
- void getPassword ()
- void getIsDeleted ()
- void getType ()
- void getID ()
- void setName ()
- void setPassword ()
- void setIsDeleted ()

Additional Inherited Members

6.103.1 Detailed Description

Definition at line 12 of file UserDriver.java.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 UserDriver()

```
test.driver.UserDriver.UserDriver ( )
```

Definition at line 16 of file UserDriver.java.

```
                          {
        this.currentUser = null;
    }
```

6.103.3 Member Function Documentation

6.103.3.1 start()

```
void test.driver.UserDriver.start ( )
```

Definition at line 20 of file UserDriver.java.

```
                          {
        while(true) {
            this.mainMenu();
        }
    }
```

6.103.3.2 mainMenu()

```
void test.driver.UserDriver.mainMenu ( )  [private]
```

Definition at line 26 of file UserDriver.java.

```
                              {
        String title = (this.currentUser != null ? String.format("Current: %s\n",
    this.currentUser.getName()) : null);
        switch (Driver.menu(title, "User Driver",
                new Pair<String, String>("1", "Create User"),
                new Pair<String, String>("2", "Get name"),
                new Pair<String, String>("3", "Set name"),
                new Pair<String, String>("4", "Get password"),
                new Pair<String, String>("5", "Set password"),
                new Pair<String, String>("6", "Get state"),
                new Pair<String, String>("7", "Set state"),
                new Pair<String, String>("8", "Get type"),
                new Pair<String, String>("9", "Get ID"),
                new Pair<String, String>("10", "Serialize User to JSON"),
                new Pair<String, String>("11", "Deserialize User from JSON"))) {
            case "1":
                this.createUser();
                break;
            case "2":
                this.getName();
                break;
            case "3":
                this.setName();
                break;
            case "4":
                this.getPassword();
                break;
            case "5":
                this.setPassword();
                break;
            case "6":
                this.getIsDeleted();
                break;
            case "7":
                this.setIsDeleted();
                break;
            case "8":
                this.getType();
                break;
            case "9":
                this.getID();
                break;
            case "10":
                this.serialize();
                break;
            case "11":
                this.deserialize();
                break;
        }
        Driver.pause();
    }
```

6.103.3.3 createUser()

`void test.driver.UserDriver.createUser ( ) [private]`

Definition at line 77 of file UserDriver.java.

```
                                {
        System.out.println("Take into account that UUIDs will be randomly generated because typing them
    in will be a hassle.\n");
        String name = Driver.input("Name?");
        String password = Driver.input("Password?");
        try {
            User user = new User("Default name", "Default password", UUID.randomUUID());
            user.setName(name);
            user.setPassword(password);
            this.currentUser = user;
            System.out.println(String.format("%s created successfully!", this.currentUser.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
    e.getMessage()));
        }
    }
```

6.103.3.4 serialize()

`void test.driver.UserDriver.serialize ( ) [private]`

Definition at line 92 of file UserDriver.java.

```
                               {
        if(this.currentUser == null) {
            System.out.println("No current User");
            return;
        }
        System.out.println(String.format("%s's serialized to JSON is: %s", this.currentUser.getName(),
                this.currentUser.serialize().toString(2)));
    }
```

6.103.3.5 deserialize()

`void test.driver.UserDriver.deserialize ( ) [private]`

Definition at line 101 of file UserDriver.java.

```
                                 {
        if(this.currentUser == null) {
            System.out.println("No current User");
            return;
        }
        System.out.println(this.currentUser.serialize().toString(2));
        this.currentUser = new User(this.currentUser.serialize());
        System.out.println(String.format("\n%s's deserialized from the above JSON successfully!\n",
                this.currentUser.getName()));
        System.out.println(String.format("name:\t\t\t%s", this.currentUser.getName()));
        System.out.println(String.format("id:\t\t\t%s", this.currentUser.getID()));
        System.out.println(String.format("password:\t\t%s", this.currentUser.getPassword()));
        System.out.println(String.format("is_deleted:\t\t%s", this.currentUser.getIsDeleted()));
    }
```

6.103.3.6 getName()

`void test.driver.UserDriver.getName ( )  [private]`

Definition at line 116 of file UserDriver.java.

```
                          {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        System.out.println(String.format("%s’s name is: %s", this.currentUser.getName(),
     this.currentUser.getName()));
    }
```

6.103.3.7 getPassword()

`void test.driver.UserDriver.getPassword ( )  [private]`

Definition at line 124 of file UserDriver.java.

```
                              {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        System.out.println(String.format("%s’s password is: %s", this.currentUser.getName(),
     this.currentUser.getPassword()));
    }
```

6.103.3.8 getIsDeleted()

`void test.driver.UserDriver.getIsDeleted ( )  [private]`

Definition at line 132 of file UserDriver.java.

```
                               {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        System.out.print(String.format("%s’s state is: ", this.currentUser.getName()));
        if(this.currentUser.getIsDeleted())
            System.out.println("DELETED");
        else
            System.out.println("ACTIVE");
    }
```

6.103.3.9 getType()

`void test.driver.UserDriver.getType ( )  [private]`

Definition at line 144 of file UserDriver.java.

```
                          {
        System.out.println("User’s type attribute was removed because of professor’s feedback. However,
     this option is still here to have backwards compatibility with old delivered documents.");
    }
```

6.103.3.10 getID()

void test.driver.UserDriver.getID ( ) [private]

Definition at line 148 of file UserDriver.java.

```
                            {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        System.out.println(String.format("%s's ID is: %s", this.currentUser.getName(),
    this.currentUser.getID()));
    }
```

6.103.3.11 setName()

void test.driver.UserDriver.setName ( ) [private]

Definition at line 156 of file UserDriver.java.

```
                              {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        try {
            this.currentUser.setName(Driver.input("New name?"));
            System.out.println(String.format("%s name changed successfully!",
    this.currentUser.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
    e.getMessage()));
        }
    }
```

6.103.3.12 setPassword()

void test.driver.UserDriver.setPassword ( ) [private]

Definition at line 169 of file UserDriver.java.

```
                                  {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        try {
            this.currentUser.setPassword(Driver.input("New password?"));
            System.out.println(String.format("%s password changed successfully!",
    this.currentUser.getName()));
        } catch (Exception e) {
            System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
    e.getMessage()));
        }
    }
```

6.103.3.13 setIsDeleted()

```
void test.driver.UserDriver.setIsDeleted ( )  [private]
```

Definition at line 182 of file UserDriver.java.

```
                                    {
        if(this.currentUser == null) {
            System.out.println("No current user!");
            return;
        }
        if(Driver.inputBool("Do you want to delete the current user?")) {
            this.currentUser.setIsDeleted(true);
            System.out.println(String.format("%s’s state has changed to DELETED!",
   this.currentUser.getName()));
        }
        else {
            System.out.println(String.format("%s’s state has not changed!",
   this.currentUser.getName()));
        }
    }
```

6.103.4 Member Data Documentation

6.103.4.1 currentUser

```
User test.driver.UserDriver.currentUser
```

Definition at line 14 of file UserDriver.java.

The documentation for this class was generated from the following file:

- UserDriver.java

6.104 view.UserModifyView Class Reference

Public Member Functions

- UserModifyView ()

    Class creator.
- void initialize ()

    Initialize method which is executed when the scene is shown.
- void bots () throws IOException

    Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

    Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

    Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

    Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

    Event method which is executed when the Play tab is clicked.
- void modifyUser () throws IOException

    Event method which is executed when the modifyUser button is clicked.
- void deleteUser () throws IOException

    Event method which is executed when the deleteUser button is clicked.
- void modifyUserConfirm () throws IOException

    Event method which is executed when the modify button is clicked.
- void logOut () throws IOException

    Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Text modifyUser

  User modify button text.
- Rectangle modifyUserButton

  User modify button.
- Text deleteUser

  User modify button text.
- Rectangle deleteUserButton

  User modify button.
- TextField nusername

  New User name text field.
- PasswordField npassword

  New User name text field.
- PasswordField rpassword

  New User name text field.
- Label modifyUserResult

  Exception output message label.
- Text modifyUserConfirm

  User create confirm text button.
- Rectangle modifyUserConfirmButton

  User create confirm button.
- Label currentUserName

  Current user name.
- Text logOut

  LogOut button.

6.104.1 Detailed Description

This class represents the scene controller of modify function of a user.

Done by Arnau Pujantell

Definition at line 26 of file UserModifyView.java.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 UserModifyView()

```
view.UserModifyView.UserModifyView ( )
```

Class creator.

Definition at line 33 of file UserModifyView.java.

```
                            {
    }
```

6.104.3 Member Function Documentation

6.104.3.1 initialize()

```
void view.UserModifyView.initialize ( )
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

The current username is shown.

Definition at line 136 of file UserModifyView.java.

```
                            {
        JSONObject user = ViewCtrl.domainCtrl.viewUser();
        currentUserName.setText(user.getString("name"));
        nusername.setText(user.getString("name"));
    }
```

6.104.3.2 bots()

```
void view.UserModifyView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 147 of file UserModifyView.java.

```
                                          {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.104.3.3 config()

`void view.UserModifyView.config ( ) throws IOException`

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 156 of file UserModifyView.java.

```
                                         {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.104.3.4 games()

`void view.UserModifyView.games ( ) throws IOException`

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 165 of file UserModifyView.java.

```
                                        {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.104.3.5 ranking()

`void view.UserModifyView.ranking ( ) throws IOException`

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 174 of file UserModifyView.java.

```
                                          {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.104.3.6 play()

```
void view.UserModifyView.play ( ) throws IOException
```

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 183 of file UserModifyView.java.

```
                                    {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.104.3.7 modifyUser()

```
void view.UserModifyView.modifyUser ( ) throws IOException
```

Event method which is executed when the modifyUser button is clicked.

Precondition

True

Postcondition

Scene changes to UserView.

Definition at line 192 of file UserModifyView.java.

```
                                          {
        ViewCtrl.changeScene("template/UserView.fxml");
    }
```

6.104.3.8 deleteUser()

```
void view.UserModifyView.deleteUser ( ) throws IOException
```

Event method which is executed when the deleteUser button is clicked.

Precondition

True

Postcondition

Scene changes to UserModifyView.

Definition at line 201 of file UserModifyView.java.

```
                                          {
        ViewCtrl.changeScene("template/UserDeleteView.fxml");
    }
```

6.104.3.9 modifyUserConfirm()

```
void view.UserModifyView.modifyUserConfirm ( ) throws IOException
```

Event method which is executed when the modify button is clicked.

Precondition

True

Postcondition

If there is an exception, it's name is shown. If not, the credentials introduced modify the current User. Finally, scene changes to UserView.

Definition at line 210 of file UserModifyView.java.

```
                                                        {
        Alert confirm = new Alert(AlertType.CONFIRMATION, "Your account will be modified. Are you
    sure?", ButtonType.YES, ButtonType.NO);
        confirm.showAndWait();

        if (confirm.getResult() == ButtonType.YES) {
            String newPassword = (!npassword.getText().isBlank() ? npassword.getText() : null);

            Pair<JSONObject, String> result = ViewCtrl.domainCtrl.modifyUser(nusername.getText(),
    newPassword, rpassword.getText());
            if (result.second != null) {
                switch (result.second) {
                    case "ERR_INVALID_NAME":
                        modifyUserResult.setText("Username can’t be empty!");
                        break;
                    case "ERR_INVALID_PASSWORD":
                        modifyUserResult.setText("Password can’t be empty!");
                        break;
                    case "ERR_BAD_CONFIRMATION":
                        modifyUserResult.setText("Confirmation password doesn’t match!");
                        break;
                    case "ERR_INEXISTING_PLAYER":
                        modifyUserResult.setText("The player doesn’t exist!");
                        break;
                    case "ERR_EXISTING_PLAYER":
                        modifyUserResult.setText("The username is already taken!");
                        break;
                    default:
                        modifyUserResult.setText("Something went wrong, try again!");
                        break;
                }
            }
            else {
                initialize();
                npassword.clear();
                rpassword.clear();
                modifyUserResult.setText("Success!");
            }
        }
    }
```

6.104.3.10 logOut()

```
void view.UserModifyView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 254 of file UserModifyView.java.

```
                                            {
            Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
       ButtonType.YES, ButtonType.NO);
            confirm.showAndWait();

            if (confirm.getResult() == ButtonType.YES) {
                ViewCtrl.domainCtrl.logout();
                ViewCtrl.changeScene("template/LogInView.fxml");
            }
        }
```

6.104.4 Member Data Documentation

6.104.4.1 user

`Text view.UserModifyView.user [private]`

Menu User tab.

Definition at line 42 of file UserModifyView.java.

6.104.4.2 bots

`Text view.UserModifyView.bots [private]`

Menu Bots tab.

Definition at line 47 of file UserModifyView.java.

6.104.4.3 config

`Text view.UserModifyView.config [private]`

Menu Configuration tab.

Definition at line 52 of file UserModifyView.java.

6.104.4.4 games

```
Text view.UserModifyView.games [private]
```

Menu Games tab.

Definition at line 57 of file UserModifyView.java.

6.104.4.5 ranking

```
Text view.UserModifyView.ranking [private]
```

Menu Ranking tab.

Definition at line 62 of file UserModifyView.java.

6.104.4.6 play

```
Text view.UserModifyView.play [private]
```

Menu Play tab.

Definition at line 67 of file UserModifyView.java.

6.104.4.7 modifyUser

```
Text view.UserModifyView.modifyUser [private]
```

User modify button text.

Definition at line 72 of file UserModifyView.java.

6.104.4.8 modifyUserButton

```
Rectangle view.UserModifyView.modifyUserButton [private]
```

User modify button.

Definition at line 77 of file UserModifyView.java.

6.104.4.9 deleteUser

```
Text view.UserModifyView.deleteUser [private]
```

User modify button text.

Definition at line 82 of file UserModifyView.java.

6.104.4.10 deleteUserButton

```
Rectangle view.UserModifyView.deleteUserButton [private]
```

User modify button.

Definition at line 87 of file UserModifyView.java.

6.104.4.11 nusername

```
TextField view.UserModifyView.nusername [private]
```

New User name text field.

Definition at line 92 of file UserModifyView.java.

6.104.4.12 npassword

```
PasswordField view.UserModifyView.npassword [private]
```

New User name text field.

Definition at line 97 of file UserModifyView.java.

6.104.4.13 rpassword

```
PasswordField view.UserModifyView.rpassword [private]
```

New User name text field.

Definition at line 102 of file UserModifyView.java.

6.104.4.14 modifyUserResult

```
Label view.UserModifyView.modifyUserResult  [private]
```

Exception output message label.

Definition at line 107 of file UserModifyView.java.

6.104.4.15 modifyUserConfirm

```
Text view.UserModifyView.modifyUserConfirm  [private]
```

User create confirm text button.

Definition at line 112 of file UserModifyView.java.

6.104.4.16 modifyUserConfirmButton

```
Rectangle view.UserModifyView.modifyUserConfirmButton  [private]
```

User create confirm button.

Definition at line 117 of file UserModifyView.java.

6.104.4.17 currentUserName

```
Label view.UserModifyView.currentUserName  [private]
```

Current user name.

Definition at line 122 of file UserModifyView.java.

6.104.4.18 logOut

```
Text view.UserModifyView.logOut  [private]
```

LogOut button.

Definition at line 127 of file UserModifyView.java.

The documentation for this class was generated from the following file:

- UserModifyView.java

6.105 view.UserView Class Reference

Public Member Functions

- UserView ()

  Class creator.
- void initialize () throws Exception

  Initialize method which is executed when the scene is shown.
- void bots () throws IOException

  Event method which is executed when the Bots tab is clicked.
- void config () throws IOException

  Event method which is executed when the Configuration tab is clicked.
- void games () throws IOException

  Event method which is executed when the Games tab is clicked.
- void ranking () throws IOException

  Event method which is executed when the Ranking tab is clicked.
- void play () throws IOException

  Event method which is executed when the Play tab is clicked.
- void modifyUser () throws IOException

  Event method which is executed when the modifyUser button is clicked.
- void deleteUser () throws IOException

  Event method which is executed when the deleteUser button is clicked.
- void logOut () throws IOException

  Event method which is executed when the LogOut button is clicked.

Private Attributes

- Text user

  Menu User tab.
- Text bots

  Menu Bots tab.
- Text config

  Menu Configuration tab.
- Text games

  Menu Games tab.
- Text ranking

  Menu Ranking tab.
- Text play

  Menu Play tab.
- Label currentUserName

  User name label.
- Text modifyUser

  User modify button text.
- Rectangle modifyUserButton

  User modify button.
- Text deleteUser

  User delete button text.
- Rectangle deleteUserButton

  User delete button.
- Text logOut

  LogOut button.

6.105.1 Detailed Description

This class represents the scene controller of the User Menu .

Done by Arnau Pujantell

Definition at line 22 of file UserView.java.

6.105.2 Constructor & Destructor Documentation

6.105.2.1 UserView()

```
view.UserView.UserView ( )
```

Class creator.

Definition at line 29 of file UserView.java.

```
                     {
    }
```

6.105.3 Member Function Documentation

6.105.3.1 initialize()

```
void view.UserView.initialize ( ) throws Exception
```

Initialize method which is executed when the scene is shown.

Precondition

True

Postcondition

All user names are inserted in the User choiceBox.

Definition at line 102 of file UserView.java.

```
                                        {
        currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
    }
```

6.105.3.2 bots()

```
void view.UserView.bots ( ) throws IOException
```

Event method which is executed when the Bots tab is clicked.

Precondition

True

Postcondition

Scene changes to BotsView.

Definition at line 111 of file UserView.java.

```
                                        {
        ViewCtrl.changeScene("template/BotsView.fxml");
    }
```

6.105.3.3 config()

```
void view.UserView.config ( ) throws IOException
```

Event method which is executed when the Configuration tab is clicked.

Precondition

True

Postcondition

Scene changes to ConfigView.

Definition at line 120 of file UserView.java.

```
                                          {
        ViewCtrl.changeScene("template/ConfigView.fxml");
    }
```

6.105.3.4 games()

```
void view.UserView.games ( ) throws IOException
```

Event method which is executed when the Games tab is clicked.

Precondition

True

Postcondition

Scene changes to GamesView.

Definition at line 129 of file UserView.java.

```
                                         {
        ViewCtrl.changeScene("template/GamesView.fxml");
    }
```

6.105.3.5 ranking()

`void view.UserView.ranking ( ) throws IOException`

Event method which is executed when the Ranking tab is clicked.

Precondition

True

Postcondition

Scene changes to RankingView.

Definition at line 138 of file UserView.java.

```
                                        {
        ViewCtrl.changeScene("template/RankingView.fxml");
    }
```

6.105.3.6 play()

`void view.UserView.play ( ) throws IOException`

Event method which is executed when the Play tab is clicked.

Precondition

True

Postcondition

Scene changes to PlayView.

Definition at line 147 of file UserView.java.

```
                                     {
        ViewCtrl.changeScene("template/PlayView.fxml");
    }
```

6.105.3.7 modifyUser()

`void view.UserView.modifyUser ( ) throws IOException`

Event method which is executed when the modifyUser button is clicked.

Precondition

True

Postcondition

Scene changes to UserModifyView.

Definition at line 156 of file UserView.java.

```
                                           {
        ViewCtrl.changeScene("template/UserModifyView.fxml");
    }
```

6.105.3.8 deleteUser()

```
void view.UserView.deleteUser ( ) throws IOException
```

Event method which is executed when the deleteUser button is clicked.

Precondition

True

Postcondition

Scene changes to UserDeleteView.

Definition at line 165 of file UserView.java.

```
                                            {
            ViewCtrl.changeScene("template/UserDeleteView.fxml");
        }
```

6.105.3.9 logOut()

```
void view.UserView.logOut ( ) throws IOException
```

Event method which is executed when the LogOut button is clicked.

Precondition

True

Postcondition

The current user is logged out and the scene is changed to LogInView.

Definition at line 174 of file UserView.java.

```
                                        {
            Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
        ButtonType.YES, ButtonType.NO);
            confirm.showAndWait();

            if (confirm.getResult() == ButtonType.YES) {
                ViewCtrl.domainCtrl.logout();
                ViewCtrl.changeScene("template/LogInView.fxml");
            }
        }
```

6.105.4 Member Data Documentation

6.105.4.1 user

```
Text view.UserView.user [private]
```

Menu User tab.

Definition at line 38 of file UserView.java.

6.105.4.2 bots

```
Text view.UserView.bots [private]
```

Menu Bots tab.

Definition at line 43 of file UserView.java.

6.105.4.3 config

```
Text view.UserView.config [private]
```

Menu Configuration tab.

Definition at line 48 of file UserView.java.

6.105.4.4 games

```
Text view.UserView.games [private]
```

Menu Games tab.

Definition at line 53 of file UserView.java.

6.105.4.5 ranking

```
Text view.UserView.ranking [private]
```

Menu Ranking tab.

Definition at line 58 of file UserView.java.

6.105.4.6 play

```
Text view.UserView.play  [private]
```

Menu Play tab.

Definition at line 63 of file UserView.java.

6.105.4.7 currentUserName

```
Label view.UserView.currentUserName  [private]
```

User name label.

Definition at line 68 of file UserView.java.

6.105.4.8 modifyUser

```
Text view.UserView.modifyUser  [private]
```

User modify button text.

Definition at line 73 of file UserView.java.

6.105.4.9 modifyUserButton

```
Rectangle view.UserView.modifyUserButton  [private]
```

User modify button.

Definition at line 78 of file UserView.java.

6.105.4.10 deleteUser

```
Text view.UserView.deleteUser  [private]
```

User delete button text.

Definition at line 83 of file UserView.java.

6.105.4.11 deleteUserButton

```
Rectangle view.UserView.deleteUserButton  [private]
```

User delete button.

Definition at line 88 of file UserView.java.

6.105.4.12 logOut

```
Text view.UserView.logOut  [private]
```

LogOut button.

Definition at line 93 of file UserView.java.

The documentation for this class was generated from the following file:

- UserView.java

6.106 view.ViewCtrl Class Reference

Public Member Functions

- ViewCtrl ()
  Class creator.
- void start (Stage primaryStage) throws Exception
  Event method which is executed when the program is executed.

Static Public Member Functions

- static void changeScene (String fxml) throws IOException
  Change scene. Event method which is executed when an fxml is recieved.
- static void newWindow (String fxml) throws IOException
  Create a new window and hide the previous one. Event method which is executed when an fxml is recieved.
- static void main (String[ ] args)
  Main method.

Static Public Attributes

- static DomainCtrl domainCtrl
  Domain Controller.

Static Private Attributes

- static Stage stage

  Main stage.

6.106.1 Detailed Description

This class represents tha main class controller.

By Arnau Pujantell

Definition at line 23 of file ViewCtrl.java.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 ViewCtrl()

```
view.ViewCtrl.ViewCtrl ( )
```

Class creator.

Definition at line 41 of file ViewCtrl.java.

```
                      {
    }
```

6.106.3 Member Function Documentation

6.106.3.1 start()

```
void view.ViewCtrl.start (
             Stage primaryStage ) throws Exception
```

Event method which is executed when the program is executed.

Precondition

True

Postcondition

All stage parameters are set and the LogInView scene is shown.

Definition at line 52 of file ViewCtrl.java.

```
                                                     {
        ViewCtrl.stage = primaryStage;
        primaryStage.setResizable(false);
        Parent root = FXMLLoader.load(getClass().getResource("template/LogInView.fxml"));
        primaryStage.setTitle("OTHELLO");
        primaryStage.setScene(new Scene(root, 1464, 824));
        primaryStage.show();
    }
```

6.106.3.2 changeScene()

```
static void view.ViewCtrl.changeScene (
             String fxml ) throws IOException  [static]
```

Change scene. Event method which is executed when an fxml is recieved.

Precondition

True

Postcondition

The scene is changed.

Definition at line 66 of file ViewCtrl.java.

```
                                                           {
        Parent pane = FXMLLoader.load(ViewCtrl.class.getResource(fxml));
        stage.getScene().setRoot(pane);
    }
```

6.106.3.3 newWindow()

```
static void view.ViewCtrl.newWindow (
             String fxml ) throws IOException  [static]
```

Create a new window and hide the previous one. Event method which is executed when an fxml is recieved.

Precondition

True

Postcondition

A new window is created and the previous one is hidden.

Definition at line 76 of file ViewCtrl.java.

```
                                                         {
        Parent pane = FXMLLoader.load(ViewCtrl.class.getResource(fxml));
        Stage windowStage = new Stage();
        windowStage.setScene(new Scene(pane, 1464, 824));
        windowStage.setTitle("OTHELLO");
        windowStage.setResizable(false);
        windowStage.initModality(Modality.APPLICATION_MODAL);
        stage.hide();
        windowStage.showAndWait();
        stage.show();
    }
```

6.106.3.4 main()

```
static void view.ViewCtrl.main (
             String[] args )  [static]
```

Main method.

Definition at line 91 of file ViewCtrl.java.

```
                                      {
        ViewCtrl.domainCtrl = new DomainCtrl();
        ViewCtrl.launch(args);
    }
```

6.106.4 Member Data Documentation

6.106.4.1 domainCtrl

```
DomainCtrl view.ViewCtrl.domainCtrl  [static]
```

Domain Controller.

Definition at line 29 of file ViewCtrl.java.

6.106.4.2 stage

```
Stage view.ViewCtrl.stage  [static], [private]
```

Main stage.

Definition at line 34 of file ViewCtrl.java.

The documentation for this class was generated from the following file:

- ViewCtrl.java

Chapter 7

File Documentation

7.1 board.java File Reference

BoardDriver entrypoint class specification.

Classes

- class cmd.driver.board

  Board driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.1.1 Detailed Description

BoardDriver entrypoint class specification.

Author

Alex Rodriguez

7.2 Board.java File Reference

Board class specification.

Classes

- class domain.Board
- enum domain.Board.PieceType

  The status of a cell of the Board. An Othello Board is composed of 64 cells with their own unique position and three possible states:

Packages

- package domain

7.2.1 Detailed Description

Board class specification.

Author

Manuel Navid

7.3 BoardCtrl.java File Reference

BoardCtrl class specification.

Classes

- class domain.BoardCtrl

  This class represents the controller of the Board class, which is the classs that will be used to communicate with the other controllers.

Packages

- package domain

7.3.1 Detailed Description

BoardCtrl class specification.

Author

Manuel Navid

7.4 BoardDriver.java File Reference

Classes

- class test.driver.BoardDriver

Packages

- package test.driver

7.5 bot.java File Reference

BotDriver entrypoint class specification.

Classes

- class cmd.driver.bot

  Bot driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.5.1 Detailed Description

BotDriver entrypoint class specification.

Author

Alex Rodriguez

7.6 Bot.java File Reference

Bot subclass specification.

Classes

- class domain.Bot

  Represents a bot in our system.

Packages

- package domain

7.6.1 Detailed Description

Bot subclass specification.

7.7 BotDriver.java File Reference

Bot driver specification Done by Arnau Pujantell.

Classes

- class test.driver.BotDriver

Packages

- package test.driver

7.7.1 Detailed Description

Bot driver specification Done by Arnau Pujantell.

7.8 BotsConsultView.java File Reference

Bot consult View controller specification.

Classes

- class view.BotsConsultView

Packages

- package view

7.8.1 Detailed Description

Bot consult View controller specification.

Author

Arnau pujantell

7.9 BotsCreateView.java File Reference

Bot create View controller specification.

Classes

- class view.BotsCreateView

Packages

- package view

7.9.1 Detailed Description

Bot create View controller specification.

Author

Arnau pujantell

7.10 BotsModifyView.java File Reference

Bot modify View controller specification.

Classes

- class view.BotsModifyView

Packages

- package view

7.10.1 Detailed Description

Bot modify View controller specification.

Author

Arnau pujantell

7.11 BotsView.java File Reference

BotsView controller specification.

Classes

- class view.BotsView

Packages

- package view

7.11.1 Detailed Description

BotsView controller specification.

Author

Arnau pujantell

7.12 ConfigConsultView.java File Reference

Configuration Consult View controller specification.

Classes

- class view.ConfigConsultView

Packages

- package view

7.12.1 Detailed Description

Configuration Consult View controller specification.

Author

Arnau pujantell

7.13 ConfigCreateView.java File Reference

Configuration Create View controller specification.

Classes

- class view.ConfigCreateView

Packages

- package view

7.13.1 Detailed Description

Configuration Create View controller specification.

Author

Arnau pujantell

7.14 ConfigModifyView.java File Reference

Configuration Modify View controller specification.

Classes

- class view.ConfigModifyView

Packages

- package view

7.14.1 Detailed Description

Configuration Modify View controller specification.

Author

Arnau pujantell

7.15 Configuration.java File Reference

Configuration class specification.

Classes

- class domain.Configuration

  Represents the rules of an Othello game including its name, whether the pieces can be eaten horizontally, vertically or diagonally, and its creator. By Alex Rodriguez.

Packages

- package domain

7.15.1 Detailed Description

Configuration class specification.

Author

Alex Rodriguez

7.16 configuration.java File Reference

ConfigurationDriver entrypoint class specification.

Classes

- class cmd.driver.configuration

  Configuration driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.16.1 Detailed Description

ConfigurationDriver entrypoint class specification.

Author

Alex Rodriguez

7.17 ConfigurationCtrl.java File Reference

ConfigurationCtrl class specification.

Classes

- class domain.ConfigurationCtrl

  Configuration domain sub-controller. It communicates with the main domain controller, the configuration repository controller and the game repository controller for certain integrity checks. It is also in charge of retrieving the initial boards associated with the configurations.

Packages

- package domain

7.17.1 Detailed Description

ConfigurationCtrl class specification.

Author

Alex Rodriguez

7.18 ConfigurationDriver.java File Reference

ConfigurationDriver class specification.

Classes

- class test.driver.ConfigurationDriver

  Implements the different options for the Configuration driver application. By Alex Rodriguez.

Packages

- package test.driver

7.18.1 Detailed Description

ConfigurationDriver class specification.

Author

Alex Rodriguez

7.19 ConfigurationRepository.java File Reference

ConfigurationRepository class specification.

Classes

- class repository.ConfigurationRepository

  Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.

Packages

- package repository

7.19.1 Detailed Description

ConfigurationRepository class specification.

Author

Alex Rodriguez

7.20 ConfigurationRepositoryCtrl.java File Reference

ConfigurationRepositoryCtrl class specification.

Classes

- class repository.ConfigurationRepositoryCtrl

  Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez.

Packages

- package repository

7.20.1 Detailed Description

ConfigurationRepositoryCtrl class specification.

Author

Alex Rodriguez

7.21 ConfigView.java File Reference

ConfigView controller specification.

Classes

- class view.ConfigView

Packages

- package view

7.21.1 Detailed Description

ConfigView controller specification.

Author

Arnau pujantell

7.22 ConsultInitialBoardView.java File Reference

ConsultInitialBoardView controller specification.

Classes

- class view.ConsultInitialBoardView

Packages

- package view

7.22.1 Detailed Description

ConsultInitialBoardView controller specification.

Author

Alex Rodriguez

7.23 Difficulty.java File Reference

Difficulty class specification.

Classes

- class domain.Difficulty

  Implements the abstract class and methods of all the difficulty implementations. By Arnau Pujantell.

Packages

- package domain

7.23.1 Detailed Description

Difficulty class specification.

Author

Arnau Pujantell

7.24 DifficultyCtrl.java File Reference

DifficultyCtrl class specification.

Classes

- class domain.DifficultyCtrl

  Difficulty domain sub-controller. Is in charge of EasyDifficulty, MediumDifficulty and HardDifficulty. It communicates with the main domain controller. It forwards the current bot's placePiece request to the correct algorithm depending on the current game's difficulty: 1 to 3: EasyDifficulty (Minimax). 4 to 6: MediumDifficulty (Minimax alpha beta pruning). 7 to 10: HardDifficulty (Montecarlo).

Packages

- package domain

7.24.1 Detailed Description

DifficultyCtrl class specification.

Author

Alex Rodriguez

7.25 DomainCtrl.java File Reference

DomainCtrl class specification.

Classes

- class domain.DomainCtrl

  Is the main domain controller. It keeps the current state of all the game and application. It serves as a forwarder for the specific domain class controllers, that's why most of the sub-controller methods receive as a parameter the current instance of the associated class. Moreover, it implements a few methods that do not have a specific domain class binded. It also gathers all the thrown exceptions in the sub-controllers and transforms them into string messages in order to pass them to the view layer without coupling any domain specific logic. By Manuel Navid.

Packages

- package domain

7.25.1 Detailed Description

DomainCtrl class specification.

Author

Manuel Navid

7.26 Driver.java File Reference

Driver class specification.

Classes

- class test.driver.Driver

  Implements various utilities to create a driver application. By Alex Rodriguez.

Packages

- package test.driver

7.26.1 Detailed Description

Driver class specification.

Author

Alex Rodriguez

7.27 EasyDifficulty.java File Reference

EasyDifficulty class specification.

Classes

- class domain.EasyDifficulty

  Implements the Minimax algorithm to get the next best possible position for a given player. By Manuel Navid.

Packages

- package domain

7.27.1 Detailed Description

EasyDifficulty class specification.

Author

Manuel Navid

7.28 easyDifficulty.java File Reference

EasyDifficultyDriver entrypoint class specification.

Classes

- class cmd.driver.easyDifficulty

  EasyDifficulty driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.28.1 Detailed Description

EasyDifficultyDriver entrypoint class specification.

Author

Alex Rodriguez

7.29 EasyDifficultyDriver.java File Reference

EasyDifficulty class specification.

Classes

- class test.driver.EasyDifficultyDriver

  Implements the different options for the EasyDifficulty driver application. By Manuel Navid.

Packages

- package test.driver

7.29.1 Detailed Description

EasyDifficulty class specification.

Author

Manuel Navid

7.30 Entry.java File Reference

Specification of class Entry.

Classes

- class domain.Entry

  Represents an entry in a Ranking table.

Packages

- package domain

7.30.1 Detailed Description

Specification of class Entry.

7.31 entry.java File Reference

JUnit Entry tests entrypoint class specification.

Classes

- class cmd.unitary.entry

  JUnit Entry tests entrypoint. By Alex Rodriguez.

Packages

- package cmd.unitary

7.31.1 Detailed Description

JUnit Entry tests entrypoint class specification.

Author

Alex Rodriguez

7.32 EntryJUnit.java File Reference

Specification of class EntryJUnit.

Classes

- class test.unitary.EntryJUnit

  Allows JUnit testing of class Entry.

Packages

- package test.unitary

7.32.1 Detailed Description

Specification of class EntryJUnit.

7.33 Exceptions.java File Reference

Exceptions classes specifications.

Classes

- class domain.Exceptions

  Holds all the different custom Exceptions used in the whole project. By Alex Rodriguez.
- class domain.Exceptions.ExistingPlayerException

  There is already a player with the same name in the system. By Alex Rodriguez.
- class domain.Exceptions.InvalidNameException

  The entered name is null, empty or blank. By Alex Rodriguez.
- class domain.Exceptions.InvalidPasswordException

  The entered password is null, empty or blank. By Alex Rodriguez.
- class domain.Exceptions.BadConfirmationException

  The entered confirmation password doesn't match the user's password. By Alex Rodriguez.
- class domain.Exceptions.InexistingPlayerException

  There isn't any player with the entered name. By Alex Rodriguez.
- class domain.Exceptions.InexistingConfigurationException

There isn't any configuration with the entered name. By Alex Rodriguez.
- class domain.Exceptions.IncorrectCredentialsException

  Wrong password or name. By Alex Rodriguez.
- class domain.Exceptions.NotCreatorException

  The user that tries to perform an action on a object is not the creator of it. By Alex Rodriguez.
- class domain.Exceptions.BotUsedException

  A bot cannot be modified or deleted if it is already part of a game. By Alex Rodriguez.
- class domain.Exceptions.InvalidDifficultyException

  The entered difficulty is null, empty, blank, less than 0 or greater than 10. By Alex Rodriguez.
- class domain.Exceptions.ExistingConfigurationException

  There is already a configuration with the same name and creator ID in the system. By Alex Rodriguez.
- class domain.Exceptions.ConfigurationUsedException

  A configuration cannot be modified or deleted if it is already used in a game. By Alex Rodriguez.
- class domain.Exceptions.InvalidBoardException

  The current board is in an illegal state. By Alex Rodriguez.
- class domain.Exceptions.InvalidRulesException

  The entered configuration rules are all deactivated. By Alex Rodriguez.
- class domain.Exceptions.InvalidPositionException

  The entered position is null, empty, blank or ends up with an invalid board. By Alex Rodriguez.
- class domain.Exceptions.InvalidPlayersException

  The entered players are the same, null, empty, blank or both bots have the same difficulty. By Alex Rodriguez.
- class domain.Exceptions.InvalidConfigurationException

  The entered configuration is null, empty or blank. By Alex Rodriguez.
- class domain.Exceptions.NotPlayerException

  The player that wants to perform an action is not part of the game. By Alex Rodriguez.
- class domain.Exceptions.NotPlayerPieceException

  The player that wants to perform an action tries to use an opponent piece. By Alex Rodriguez.
- class domain.Exceptions.NotPlayerTurnException

  It is not the turn of the player that wants to perform an action. By Alex Rodriguez.
- class domain.Exceptions.FinishedGameException

  The game is already finished. By Alex Rodriguez.
- class domain.Exceptions.NotStartedGameException

  The game has not yet started. By Alex Rodriguez.

Packages

- package domain

7.33.1 Detailed Description

Exceptions classes specifications.

Author

Alex Rodriguez

7.34 FixtureRepository.java File Reference

FixtureRepository class specification.

Classes

- class repository.FixtureRepository

  Implements various CRUD operations to work with the Fixture repository. By Alex Rodriguez.

Packages

- package repository

7.34.1 Detailed Description

FixtureRepository class specification.

Author

Alex Rodriguez

7.35 Game.java File Reference

Game class specification.

Classes

- class domain.Game

  Represents the state of an Othello game including its name, players, the current turn, the state, the configuration used, the winner if any, its creator and the creation timestamp. By Alex Rodriguez.
- enum domain.Game.GameState

  State of a Game. Whether it has not started, it is currently being played or it has already finished.

Packages

- package domain

7.35.1 Detailed Description

Game class specification.

Author

Alex Rodriguez

7.36 game.java File Reference

GameDriver entrypoint class specification.

Classes

- class cmd.driver.game

    Game driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.36.1 Detailed Description

GameDriver entrypoint class specification.

Author

Alex Rodriguez

7.37 GameBoardView.java File Reference

GameBoardView controller specification.

Classes

- class view.GameBoardView

Packages

- package view

7.37.1 Detailed Description

GameBoardView controller specification.

Author

Alex Rodriguez

7.38 GameCtrl.java File Reference

GameCtrl class specification.

Classes

- class domain.GameCtrl

  Game domain sub-controller. It communicates with the main domain controller, the game repository controller, the configuration repository controller and the player repository controller in order to source the necessary components to manage games.

Packages

- package domain

7.38.1 Detailed Description

GameCtrl class specification.

Author

Alex Rodriguez

7.39 GameDriver.java File Reference

GameDriver class specification.

Classes

- class test.driver.GameDriver

  Implements the different options for the Game driver application. By Alex Rodriguez.

Packages

- package test.driver

7.39.1 Detailed Description

GameDriver class specification.

Author

Alex Rodriguez

7.40 GameRepository.java File Reference

GameRepository class specification.

Classes

- class repository.GameRepository

  Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.

Packages

- package repository

7.40.1 Detailed Description

GameRepository class specification.

Author

Alex Rodriguez

7.41 GameRepositoryCtrl.java File Reference

GameRepositoryCtrl class specification.

Classes

- class repository.GameRepositoryCtrl

  Implements various CRUD operations to work with the Game repository. By Alex Rodriguez.

Packages

- package repository

7.41.1 Detailed Description

GameRepositoryCtrl class specification.

Author

Alex Rodriguez

7.42 GamesCreateView.java File Reference

Game Create View controller specification.

Classes

- class view.GamesCreateView

Packages

- package view

7.42.1 Detailed Description

Game Create View controller specification.

Author

Arnau pujantell

7.43 GamesView.java File Reference

GamesView controller specification.

Classes

- class view.GamesView

Packages

- package view

7.43.1 Detailed Description

GamesView controller specification.

Author

Arnau pujantell

7.44 hardDifficulty.java File Reference

HardDifficultyDriver entrypoint class specification.

Classes

- class cmd.driver.hardDifficulty

  HardDifficulty driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.44.1 Detailed Description

HardDifficultyDriver entrypoint class specification.

Author

Alex Rodriguez

7.45 HardDifficulty.java File Reference

HardDifficulty class specification.

Classes

- class domain.HardDifficulty

  Implements the Monte Carlo Tree Search algorithm to get the next best possible position for a given player. By Roger Mollon.
- class domain.HardDifficulty.TreeNode

Packages

- package domain

7.45.1 Detailed Description

HardDifficulty class specification.

Author

Roger Mollon

7.46 HardDifficultyDriver.java File Reference

HardDifficultyDriver class specification.

Classes

- class test.driver.HardDifficultyDriver

  Implements the different options for the HardDifficulty driver application. By Roger Mollon.

Packages

- package test.driver

7.46.1 Detailed Description

HardDifficultyDriver class specification.

Author

Roger Mollon

7.47 InitialBoardView.java File Reference

InitialBoardView controller specification.

Classes

- class view.InitialBoardView

Packages

- package view

7.47.1 Detailed Description

InitialBoardView controller specification.

Author

Alex Rodriguez

7.48 LogInView.java File Reference

LogInView controller specification.

Classes

- class view.LogInView

Packages

- package view

7.48.1 Detailed Description

LogInView controller specification.

Author

Arnau pujantell

7.49 MediumDifficulty.java File Reference

MediumDifficulty class specification.

Classes

- class domain.MediumDifficulty

  Implements the Minimax algorithm with alpha-beta pruning to get the next best possible position for a given player. By Alex Rodriguez.

Packages

- package domain

7.49.1 Detailed Description

MediumDifficulty class specification.

Author

Alex Rodriguez

7.50 mediumDifficulty.java File Reference

MediumDifficultyDriver entrypoint class specification.

Classes

- class cmd.driver.mediumDifficulty

  MediumDifficulty driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.50.1 Detailed Description

MediumDifficultyDriver entrypoint class specification.

Author

Alex Rodriguez

7.51 MediumDifficultyDriver.java File Reference

MediumDifficulty class specification.

Classes

- class test.driver.MediumDifficultyDriver

  Implements the different options for the MediumDifficulty driver application. By Alex Rodriguez.

Packages

- package test.driver

7.51.1 Detailed Description

MediumDifficulty class specification.

Author

Alex Rodriguez

7.52 ModifyInitialBoardView.java File Reference

ModifyInitialBoardView controller specification.

Classes

- class view.ModifyInitialBoardView

Packages

- package view

7.52.1 Detailed Description

ModifyInitialBoardView controller specification.

Author

Alex Rodriguez

7.53 othello.java File Reference

Othello entrypoint class specification.

Classes

- class cmd.othello

  Othello application entrypoint. By Alex Rodriguez.

Packages

- package cmd

7.53.1 Detailed Description

Othello entrypoint class specification.

Author

Alex Rodriguez

7.54 Pair.java File Reference

Pair class specification.

Classes

- class util.Pair< F, S >

  Implements a data structure containing two generic types. By Alex Rodriguez.

Packages

- package util

7.54.1 Detailed Description

Pair class specification.

Author

Alex Rodriguez

7.55 pair.java File Reference

PairDriver entrypoint class specification.

Classes

- class cmd.driver.pair

  Pair driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.55.1 Detailed Description

PairDriver entrypoint class specification.

Author

Alex Rodriguez

7.56 PairDriver.java File Reference

PairDriver class specification.

Classes

- class test.driver.PairDriver

  Implements the different options for the Pair driver application. By Alex Rodriguez.

Packages

- package test.driver

7.56.1 Detailed Description

PairDriver class specification.

Author

Alex Rodriguez

7.57 Player.java File Reference

Player class specification.

Classes

- class domain.Player

  Represents a player in our system.

Packages

- package domain

7.57.1 Detailed Description

Player class specification.

7.58 PlayerCtrl.java File Reference

PlayerCtrl controller specification.

Classes

- class domain.PlayerCtrl

  Player class controller.

Packages

- package domain

7.58.1 Detailed Description

PlayerCtrl controller specification.

7.59 PlayerRepository.java File Reference

PlayerRepository class specification.

Classes

- class repository.PlayerRepository

  Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.

Packages

- package repository

7.59.1 Detailed Description

PlayerRepository class specification.

Author

Alex Rodriguez

7.60 PlayerRepositoryCtrl.java File Reference

PlayerRepositoryCtrl class specification.

Classes

- class repository.PlayerRepositoryCtrl

  Implements various CRUD operations to work with the Player repository. By Alex Rodriguez.

Packages

- package repository

7.60.1 Detailed Description

PlayerRepositoryCtrl class specification.

Author

Alex Rodriguez

7.61 PlayView.java File Reference

PlayView controller specification.

Classes

- class view.PlayView

Packages

- package view

7.61.1 Detailed Description

PlayView controller specification.

Author

Alex Rodriguez

7.62 ranking.java File Reference

JUnit Ranking tests entrypoint class specification.

Classes

- class cmd.unitary.ranking

  JUnit Ranking tests entrypoint. By Alex Rodriguez.

Packages

- package cmd.unitary

7.62.1 Detailed Description

JUnit Ranking tests entrypoint class specification.

Author

Alex Rodriguez

7.63 Ranking.java File Reference

Specification of class Ranking.

Classes

- class domain.Ranking

  Representation of a ranking table.
- enum domain.Ranking.RankingType

Packages

- package domain

7.63.1 Detailed Description

Specification of class Ranking.

7.64 RankingConsultView.java File Reference

Ranking Consult View controller specification.

Classes

- class view.RankingConsultView

Packages

- package view

7.64.1 Detailed Description

Ranking Consult View controller specification.

Author

Alex Rodriguez

7.65 RankingCtrl.java File Reference

RankingCtrl class specification.

Classes

- class domain.RankingCtrl

  Ranking domain sub-controller. It communicates with the main domain controller and the ranking repository controller. By Alex Rodriguez.

Packages

- package domain

7.65.1 Detailed Description

RankingCtrl class specification.

Author

Alex Rodriguez

7.66 RankingJUnit.java File Reference

Specification of class RankingJUnit.

Classes

- class test.unitary.RankingJUnit

  Allows JUnit testing of class Ranking.

Packages

- package test.unitary

7.66.1 Detailed Description

Specification of class RankingJUnit.

7.67 RankingRepository.java File Reference

RankingRepository class specification.

Classes

- class repository.RankingRepository

  Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.

Packages

- package repository

7.67.1 Detailed Description

RankingRepository class specification.

Author

Alex Rodriguez

7.68 RankingRepositoryCtrl.java File Reference

RankingRepositoryCtrl class specification.

Classes

- class repository.RankingRepositoryCtrl

  Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez.

Packages

- package repository

7.68.1 Detailed Description

RankingRepositoryCtrl class specification.

Author

Alex Rodriguez

7.69 RankingView.java File Reference

RankingView controller specification.

Classes

- class view.RankingView

Packages

- package view

7.69.1 Detailed Description

RankingView controller specification.

Author

Arnau pujantell

7.70 RecordConsultView.java File Reference

Record Consult View controller specification.

Classes

- class view.RecordConsultView

Packages

- package view

7.70.1 Detailed Description

Record Consult View controller specification.

Author

Alex Rodriguez

7.71 Repository.java File Reference

Repository class specification.

Classes

- class repository.Repository

  Implements various CRUD operations to work with the local file system JSON databases and TXT fixtures. By Alex Rodriguez.
- enum repository.Repository.RepositoryType

  Different types for the accessed repository.

Packages

- package repository

7.71.1 Detailed Description

Repository class specification.

Author

Alex Rodriguez

7.72 SignUpView.java File Reference

SignUpView controller specification.

Classes

- class view.SignUpView

Packages

- package view

7.72.1 Detailed Description

SignUpView controller specification.

Author

Arnau pujantell

7.73 User.java File Reference

User subclass specification.

Classes

- class domain.User

  Represents a human user in our system.

Packages

- package domain

7.73.1 Detailed Description

User subclass specification.

7.74 user.java File Reference

UserDriver entrypoint class specification.

Classes

- class cmd.driver.user

  User driver entrypoint. By Alex Rodriguez.

Packages

- package cmd.driver

7.74.1 Detailed Description

UserDriver entrypoint class specification.

Author

Alex Rodriguez

7.75 UserDeleteView.java File Reference

User delete View controller specification.

Classes

- class view.UserDeleteView

Packages

- package view

7.75.1 Detailed Description

User delete View controller specification.

Author

Arnau pujantell

7.76 UserDriver.java File Reference

User driver specification Done by Arnau Pujantell.

Classes

- class test.driver.UserDriver

Packages

- package test.driver

7.76.1 Detailed Description

User driver specification Done by Arnau Pujantell.

7.77 UserModifyView.java File Reference

User modify View controller specification.

Classes

- class view.UserModifyView

Packages

- package view

7.77.1 Detailed Description

User modify View controller specification.

Author

Arnau pujantell

7.78 UserView.java File Reference

UserView controller specification.

Classes

- class view.UserView

Packages

- package view

7.78.1 Detailed Description

UserView controller specification.

Author

Arnau pujantell

7.79 ViewCtrl.java File Reference

ViewCtrl class specification.

Classes

- class view.ViewCtrl

Packages

- package view

7.79.1 Detailed Description

ViewCtrl class specification.

Author

Arnau Pujantell

Index