

Othello, PROP 20-21 Q2  
Third assignment. Group 13-3.2

Generated by Doxygen 1.8.18



<b>1 Othello, PROP 20-21 Q2.</b>	<b>1</b>
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy . . . . .	3
<b>3 Class Index</b>	<b>7</b>
3.1 Class List . . . . .	7
<b>4 File Index</b>	<b>13</b>
4.1 File List . . . . .	13
<b>5 Namespace Documentation</b>	<b>17</b>
5.1 Package cmd . . . . .	17
5.2 Package cmd.driver . . . . .	17
5.3 Package cmd.unitary . . . . .	18
5.4 Package domain . . . . .	18
5.5 Package repository . . . . .	19
5.6 Package test . . . . .	19
5.7 Package test.driver . . . . .	20
5.8 Package test.unitary . . . . .	20
5.9 Package util . . . . .	20
5.10 Package view . . . . .	21
<b>6 Class Documentation</b>	<b>23</b>
6.1 domain.Exceptions.BadConfirmationException Class Reference . . . . .	23
6.1.1 Detailed Description . . . . .	23
6.1.2 Constructor & Destructor Documentation . . . . .	23
6.1.2.1 BadConfirmationException() . . . . .	23
6.2 cmd.driver.board Class Reference . . . . .	24
6.2.1 Detailed Description . . . . .	24
6.2.2 Member Function Documentation . . . . .	24
6.2.2.1 main() . . . . .	24
6.3 domain.Board Class Reference . . . . .	24
6.3.1 Detailed Description . . . . .	26
6.3.2 Constructor & Destructor Documentation . . . . .	26
6.3.2.1 Board() [1/3] . . . . .	26
6.3.2.2 Board() [2/3] . . . . .	27
6.3.2.3 Board() [3/3] . . . . .	27
6.3.3 Member Function Documentation . . . . .	28
6.3.3.1 serialize() . . . . .	28
6.3.3.2 getBoard() . . . . .	29
6.3.3.3 getPiecesPlayer1() . . . . .	29
6.3.3.4 getPiecesPlayer2() . . . . .	30
6.3.3.5 isValid() . . . . .	30

6.3.3.6 validPositions()	31
6.3.3.7 removePiece()	32
6.3.3.8 placePiece()	32
6.3.3.9 placePieceConfig()	34
6.3.3.10 surroundingPieces()	34
6.3.3.11 transcribeToCharacters()	36
6.3.3.12 transcribeToPieceType()	37
6.3.3.13 inversePlayer()	37
6.3.3.14 canPlaceHorizontal()	38
6.3.3.15 canPlaceVertical()	39
6.3.3.16 canPlaceDiagonal()	40
6.3.3.17 changePieces()	42
6.3.4 Member Data Documentation	43
6.3.4.1 board	44
6.3.4.2 piecesPlayer1	44
6.3.4.3 piecesPlayer2	44
6.4 domain.BoardCtrl Class Reference	44
6.4.1 Detailed Description	45
6.4.2 Constructor & Destructor Documentation	45
6.4.2.1 BoardCtrl()	45
6.4.3 Member Function Documentation	45
6.4.3.1 placePiece()	45
6.4.3.2 placePieceConfig()	46
6.4.3.3 removePiece()	47
6.4.3.4 getNumPieces()	47
6.4.3.5 validPositions()	48
6.4.3.6 isValid()	48
6.5 test.driver.BoardDriver Class Reference	49
6.5.1 Detailed Description	50
6.5.2 Constructor & Destructor Documentation	50
6.5.2.1 BoardDriver()	50
6.5.3 Member Function Documentation	50
6.5.3.1 start()	50
6.5.3.2 defaultBoard()	51
6.5.3.3 createBoard()	52
6.5.3.4 loadBoard()	52
6.5.3.5 serializeBoard()	53
6.5.3.6 getPiecesPlayers()	53
6.5.3.7 getCurrentBoard()	53
6.5.3.8 placePieceBoard()	54
6.5.3.9 removePieceBoard()	54
6.5.3.10 placePieceInitialBoard()	55

6.5.3.11 validBoard()	55
6.5.3.12 addPositions()	56
6.5.3.13 deserialize()	57
6.5.3.14 printCurrentBoard()	57
6.5.3.15 transcribeToCharacters()	58
6.5.3.16 correctNumber()	58
6.5.3.17 correctPieceType()	59
6.5.3.18 playAgain()	59
6.5.3.19 correctBoolean()	60
6.5.4 Member Data Documentation	60
6.5.4.1 currentBoard	60
6.5.4.2 nameCurrentBoard	61
6.6 domain.Bot Class Reference	61
6.6.1 Detailed Description	61
6.6.2 Constructor & Destructor Documentation	62
6.6.2.1 Bot() [1/2]	62
6.6.2.2 Bot() [2/2]	62
6.6.3 Member Function Documentation	63
6.6.3.1 serialize()	63
6.6.3.2 getDifficulty()	63
6.6.3.3 getCreatorID()	64
6.6.3.4 setDifficulty()	64
6.6.4 Member Data Documentation	64
6.6.4.1 difficulty	65
6.6.4.2 creatorID	65
6.7 cmd.driver.bot Class Reference	65
6.7.1 Detailed Description	65
6.7.2 Member Function Documentation	65
6.7.2.1 main()	66
6.8 test.driver.BotDriver Class Reference	66
6.8.1 Detailed Description	67
6.8.2 Constructor & Destructor Documentation	67
6.8.2.1 BotDriver()	67
6.8.3 Member Function Documentation	67
6.8.3.1 start()	67
6.8.3.2 mainMenu()	68
6.8.3.3 createBot()	68
6.8.3.4 serialize()	69
6.8.3.5 deserialize()	69
6.8.3.6 getName()	69
6.8.3.7 getDifficulty()	70
6.8.3.8 getIsDeleted()	70

6.8.3.9 getType()	70
6.8.3.10 getID()	70
6.8.3.11 getCreatorID()	71
6.8.3.12 setName()	71
6.8.3.13 setDifficulty()	71
6.8.3.14 setIsDeleted()	72
6.8.4 Member Data Documentation	72
6.8.4.1 currentBot	72
6.9 view.BotsConsultView Class Reference	72
6.9.1 Detailed Description	74
6.9.2 Constructor & Destructor Documentation	74
6.9.2.1 BotsConsultView()	74
6.9.3 Member Function Documentation	74
6.9.3.1 initialize()	74
6.9.3.2 user()	75
6.9.3.3 onChangeBotChooser()	75
6.9.3.4 config()	76
6.9.3.5 games()	76
6.9.3.6 ranking()	76
6.9.3.7 play()	77
6.9.3.8 createBot()	77
6.9.3.9 modifyBot()	77
6.9.3.10 consultBot()	78
6.9.3.11 logOut()	78
6.9.4 Member Data Documentation	78
6.9.4.1 user	79
6.9.4.2 bots	79
6.9.4.3 config	79
6.9.4.4 games	79
6.9.4.5 ranking	79
6.9.4.6 play	80
6.9.4.7 createBot	80
6.9.4.8 createBotButton	80
6.9.4.9 modifyBot	80
6.9.4.10 modifyBotButton	80
6.9.4.11 consultBot	81
6.9.4.12 consultBotButton	81
6.9.4.13 botChooser	81
6.9.4.14 name	81
6.9.4.15 difficultyNumber	81
6.9.4.16 consultConfigResult	82
6.9.4.17 consultBotResult	82

6.9.4.18	currentUserName	82
6.9.4.19	creator	82
6.9.4.20	logOut	82
6.9.4.21	botMap	83
6.10	view.BotsCreateView Class Reference	83
6.10.1	Detailed Description	84
6.10.2	Constructor & Destructor Documentation	85
6.10.2.1	BotsCreateView()	85
6.10.3	Member Function Documentation	85
6.10.3.1	initialize()	85
6.10.3.2	user()	85
6.10.3.3	config()	86
6.10.3.4	games()	86
6.10.3.5	ranking()	86
6.10.3.6	play()	87
6.10.3.7	createBot()	87
6.10.3.8	modifyBot()	87
6.10.3.9	consultBot()	88
6.10.3.10	showLevel()	88
6.10.3.11	createBotConfirm()	89
6.10.3.12	logOut()	89
6.10.4	Member Data Documentation	90
6.10.4.1	user	90
6.10.4.2	bots	90
6.10.4.3	config	90
6.10.4.4	games	90
6.10.4.5	ranking	90
6.10.4.6	play	91
6.10.4.7	createBot	91
6.10.4.8	createBotButton	91
6.10.4.9	modifyBot	91
6.10.4.10	modifyBotButton	91
6.10.4.11	consultBot	92
6.10.4.12	consultBotButton	92
6.10.4.13	nbotname	92
6.10.4.14	difficultyLevel	92
6.10.4.15	difficultyNumber	92
6.10.4.16	createBotResult	93
6.10.4.17	createBotConfirm	93
6.10.4.18	createBotConfirmButton	93
6.10.4.19	currentUserName	93
6.10.4.20	logOut	93

6.11 view.BotsModifyView Class Reference . . . . .	94
6.11.1 Detailed Description . . . . .	95
6.11.2 Constructor & Destructor Documentation . . . . .	95
6.11.2.1 BotsModifyView() . . . . .	96
6.11.3 Member Function Documentation . . . . .	96
6.11.3.1 initialize() . . . . .	96
6.11.3.2 user() . . . . .	96
6.11.3.3 onChangeBotChooser() . . . . .	97
6.11.3.4 config() . . . . .	97
6.11.3.5 games() . . . . .	98
6.11.3.6 ranking() . . . . .	98
6.11.3.7 play() . . . . .	98
6.11.3.8 createBot() . . . . .	99
6.11.3.9 modifyBot() . . . . .	99
6.11.3.10 consultBot() . . . . .	99
6.11.3.11 showLevel() . . . . .	100
6.11.3.12 modifyBotConfirm() . . . . .	100
6.11.3.13 deleteBot() . . . . .	101
6.11.3.14 logOut() . . . . .	102
6.11.4 Member Data Documentation . . . . .	102
6.11.4.1 user . . . . .	102
6.11.4.2 bots . . . . .	102
6.11.4.3 config . . . . .	103
6.11.4.4 games . . . . .	103
6.11.4.5 ranking . . . . .	103
6.11.4.6 play . . . . .	103
6.11.4.7 createBot . . . . .	103
6.11.4.8 createBotButton . . . . .	104
6.11.4.9 modifyBot . . . . .	104
6.11.4.10 modifyBotButton . . . . .	104
6.11.4.11 consultBot . . . . .	104
6.11.4.12 consultBotButton . . . . .	104
6.11.4.13 botChooser . . . . .	105
6.11.4.14 nbotname . . . . .	105
6.11.4.15 difficultyLevel . . . . .	105
6.11.4.16 difficultyNumber . . . . .	105
6.11.4.17 modifyBotResult . . . . .	105
6.11.4.18 modifyBotConfirm . . . . .	106
6.11.4.19 modifyBotConfirmButton . . . . .	106
6.11.4.20 deleteBot . . . . .	106
6.11.4.21 deleteBotButton . . . . .	106
6.11.4.22 currentUserName . . . . .	106



6.11.4.23 logOut	107
6.11.4.24 botMap	107
6.12 view.BotsView Class Reference	107
6.12.1 Detailed Description	108
6.12.2 Constructor & Destructor Documentation	108
6.12.2.1 BotsView()	108
6.12.3 Member Function Documentation	109
6.12.3.1 initialize()	109
6.12.3.2 user()	109
6.12.3.3 config()	110
6.12.3.4 games()	110
6.12.3.5 ranking()	110
6.12.3.6 play()	111
6.12.3.7 createBot()	111
6.12.3.8 modifyBot()	111
6.12.3.9 consultBot()	112
6.12.3.10 logOut()	112
6.12.4 Member Data Documentation	112
6.12.4.1 user	113
6.12.4.2 bots	113
6.12.4.3 config	113
6.12.4.4 games	113
6.12.4.5 ranking	113
6.12.4.6 play	114
6.12.4.7 createBot	114
6.12.4.8 createBotButton	114
6.12.4.9 modifyBot	114
6.12.4.10 modifyBotButton	114
6.12.4.11 consultBot	115
6.12.4.12 consultBotButton	115
6.12.4.13 currentUserName	115
6.12.4.14 logOut	115
6.13 domain.Exceptions.BotUsedException Class Reference	115
6.13.1 Detailed Description	116
6.13.2 Constructor & Destructor Documentation	116
6.13.2.1 BotUsedException()	116
6.14 view.ConfigConsultView Class Reference	116
6.14.1 Detailed Description	118
6.14.2 Constructor & Destructor Documentation	118
6.14.2.1 ConfigConsultView()	118
6.14.3 Member Function Documentation	118
6.14.3.1 initialize()	118

---

6.14.3.2 user()	119
6.14.3.3 createConfig()	119
6.14.3.4 modifyConfig()	119
6.14.3.5 consultConfig()	120
6.14.3.6 bots()	120
6.14.3.7 onChangeConfigChooser()	121
6.14.3.8 games()	121
6.14.3.9 ranking()	122
6.14.3.10 play()	122
6.14.3.11 checkInitialBoard()	122
6.14.3.12 logOut()	123
6.14.4 Member Data Documentation	123
6.14.4.1 user	123
6.14.4.2 bots	123
6.14.4.3 config	124
6.14.4.4 games	124
6.14.4.5 ranking	124
6.14.4.6 play	124
6.14.4.7 createConfig	124
6.14.4.8 createConfigButton	125
6.14.4.9 modifyConfig	125
6.14.4.10 modifyConfigButton	125
6.14.4.11 consultConfig	125
6.14.4.12 consultConfigButton	125
6.14.4.13 consultConfigConfirm	126
6.14.4.14 consultConfigConfirmButton	126
6.14.4.15 configChooser	126
6.14.4.16 name	126
6.14.4.17 ceh	126
6.14.4.18 cev	127
6.14.4.19 ced	127
6.14.4.20 checkInitialBoard	127
6.14.4.21 checkInitialBoardButton	127
6.14.4.22 creator	127
6.14.4.23 consultConfigResult	128
6.14.4.24 currentUserName	128
6.14.4.25 logOut	128
6.15 view.ConfigCreateView Class Reference	128
6.15.1 Detailed Description	130
6.15.2 Constructor & Destructor Documentation	130
6.15.2.1 ConfigCreateView()	130
6.15.3 Member Function Documentation	130

---

6.15.3.1 initialize()	130
6.15.3.2 user()	131
6.15.3.3 bots()	131
6.15.3.4 games()	131
6.15.3.5 ranking()	132
6.15.3.6 play()	132
6.15.3.7 createConfig()	132
6.15.3.8 modifyConfig()	133
6.15.3.9 consultConfig()	133
6.15.3.10 createInitialBoard()	133
6.15.3.11 createConfigConfirm()	134
6.15.3.12 logOut()	134
6.15.4 Member Data Documentation	135
6.15.4.1 user	135
6.15.4.2 bots	135
6.15.4.3 config	135
6.15.4.4 games	135
6.15.4.5 ranking	135
6.15.4.6 play	136
6.15.4.7 createConfig	136
6.15.4.8 createConfigButton	136
6.15.4.9 modifyConfig	136
6.15.4.10 modifyConfigButton	136
6.15.4.11 consultConfig	137
6.15.4.12 consultConfigButton	137
6.15.4.13 createInitialBoard	137
6.15.4.14 createInitialBoardButton	137
6.15.4.15 nconfname	137
6.15.4.16 canEatHorizontally	138
6.15.4.17 canEatVertically	138
6.15.4.18 canEatDiagonally	138
6.15.4.19 createConfigResult	138
6.15.4.20 createConfigConfirm	138
6.15.4.21 createConfigConfirmButton	139
6.15.4.22 currentUsername	139
6.15.4.23 logOut	139
6.16 view.ConfigModifyView Class Reference	139
6.16.1 Detailed Description	141
6.16.2 Constructor & Destructor Documentation	141
6.16.2.1 ConfigModifyView()	141
6.16.3 Member Function Documentation	141
6.16.3.1 initialize()	142

6.16.3.2 user()	142
6.16.3.3 bots()	142
6.16.3.4 onChangeConfigChooser()	143
6.16.3.5 games()	143
6.16.3.6 ranking()	144
6.16.3.7 play()	144
6.16.3.8 createConfig()	144
6.16.3.9 modifyConfig()	145
6.16.3.10 consultConfig()	145
6.16.3.11 modifyInitialBoard()	145
6.16.3.12 modifyConfigConfirm()	146
6.16.3.13 deleteConfig()	146
6.16.3.14 logOut()	147
6.16.4 Member Data Documentation	148
6.16.4.1 user	148
6.16.4.2 bots	148
6.16.4.3 config	148
6.16.4.4 games	148
6.16.4.5 ranking	148
6.16.4.6 play	149
6.16.4.7 createConfig	149
6.16.4.8 createConfigButton	149
6.16.4.9 modifyConfig	149
6.16.4.10 modifyConfigButton	149
6.16.4.11 consultConfig	150
6.16.4.12 consultConfigButton	150
6.16.4.13 canEatHorizontally	150
6.16.4.14 canEatVertically	150
6.16.4.15 canEatDiagonally	150
6.16.4.16 modifyInitialBoard	151
6.16.4.17 modifyInitialBoardButton	151
6.16.4.18 modifyConfigConfirm	151
6.16.4.19 modifyConfigConfirmButton	151
6.16.4.20 deleteConfig	151
6.16.4.21 deleteConfigButton	152
6.16.4.22 configChooser	152
6.16.4.23 modifyConfigResult	152
6.16.4.24 currentUserName	152
6.16.4.25 logOut	152
6.17 domain.Configuration Class Reference	153
6.17.1 Detailed Description	154
6.17.2 Constructor & Destructor Documentation	154

6.17.2.1 Configuration() [1/2]	154
6.17.2.2 Configuration() [2/2]	155
6.17.3 Member Function Documentation	155
6.17.3.1 serialize()	155
6.17.3.2 getName()	156
6.17.3.3 setName()	156
6.17.3.4 getCreatorID()	157
6.17.3.5 getCanEatHorizontally()	157
6.17.3.6 setCanEatHorizontally()	158
6.17.3.7 getCanEatVertically()	158
6.17.3.8 setCanEatVertically()	159
6.17.3.9 getCanEatDiagonally()	159
6.17.3.10 setCanEatDiagonally()	160
6.17.4 Member Data Documentation	160
6.17.4.1 name	160
6.17.4.2 creatorID	160
6.17.4.3 canEatHorizontally	161
6.17.4.4 canEatVertically	161
6.17.4.5 canEatDiagonally	161
6.18 cmd.driver.configuration Class Reference	161
6.18.1 Detailed Description	161
6.18.2 Member Function Documentation	162
6.18.2.1 main()	162
6.19 domain.ConfigurationCtrl Class Reference	162
6.19.1 Detailed Description	163
6.19.2 Constructor & Destructor Documentation	163
6.19.2.1 ConfigurationCtrl()	163
6.19.3 Member Function Documentation	164
6.19.3.1 create()	164
6.19.3.2 modify()	165
6.19.3.3 save()	166
6.19.3.4 delete()	167
6.19.3.5 getConfiguration()	167
6.19.3.6 getInitialBoard()	168
6.19.3.7 list()	168
6.19.4 Member Data Documentation	169
6.19.4.1 repositoryCtrl	169
6.19.4.2 gameRepositoryCtrl	169
6.20 test.driver.ConfigurationDriver Class Reference	169
6.20.1 Detailed Description	170
6.20.2 Constructor & Destructor Documentation	170
6.20.2.1 ConfigurationDriver()	170

6.20.3 Member Function Documentation	170
6.20.3.1 start()	171
6.20.3.2 mainMenu()	171
6.20.3.3 create()	172
6.20.3.4 getName()	172
6.20.3.5 setName()	172
6.20.3.6 getCreatorID()	173
6.20.3.7 getCanEatHorizontally()	173
6.20.3.8 setCanEatHorizontally()	173
6.20.3.9 getCanEatVertically()	173
6.20.3.10 setCanEatVertically()	174
6.20.3.11 getCanEatDiagonally()	174
6.20.3.12 setCanEatDiagonally()	174
6.20.3.13 serialize()	175
6.20.3.14 deserialize()	175
6.20.4 Member Data Documentation	175
6.20.4.1 currentConfiguration	175
6.21 repository.ConfigurationRepository Class Reference	175
6.21.1 Detailed Description	176
6.21.2 Constructor & Destructor Documentation	176
6.21.2.1 ConfigurationRepository()	176
6.21.3 Member Function Documentation	177
6.21.3.1 save()	177
6.21.3.2 delete()	177
6.21.3.3 getConfiguration()	178
6.21.3.4 getBoard()	178
6.21.3.5 listConfigurations()	179
6.22 repository.ConfigurationRepositoryCtrl Class Reference	180
6.22.1 Detailed Description	180
6.22.2 Constructor & Destructor Documentation	180
6.22.2.1 ConfigurationRepositoryCtrl()	180
6.22.3 Member Function Documentation	181
6.22.3.1 save()	181
6.22.3.2 delete()	181
6.22.3.3 getConfiguration()	182
6.22.3.4 getBoard()	182
6.22.3.5 listConfigurations()	183
6.22.4 Member Data Documentation	183
6.22.4.1 repository	184
6.23 domain.Exceptions.ConfigurationUsedException Class Reference	184
6.23.1 Detailed Description	184
6.23.2 Constructor & Destructor Documentation	184

6.23.2.1 ConfigurationUsedException()	184
6.24 view.ConfigView Class Reference	185
6.24.1 Detailed Description	186
6.24.2 Constructor & Destructor Documentation	186
6.24.2.1 ConfigView()	186
6.24.3 Member Function Documentation	186
6.24.3.1 initialize()	186
6.24.3.2 user()	187
6.24.3.3 bots()	187
6.24.3.4 games()	187
6.24.3.5 ranking()	188
6.24.3.6 play()	188
6.24.3.7 createConfig()	188
6.24.3.8 modifyConfig()	189
6.24.3.9 consultConfig()	189
6.24.3.10 logOut()	189
6.24.4 Member Data Documentation	190
6.24.4.1 user	190
6.24.4.2 bots	190
6.24.4.3 config	190
6.24.4.4 games	190
6.24.4.5 ranking	190
6.24.4.6 play	191
6.24.4.7 createConfig	191
6.24.4.8 createConfigButton	191
6.24.4.9 modifyConfig	191
6.24.4.10 modifyConfigButton	191
6.24.4.11 consultConfig	192
6.24.4.12 consultConfigButton	192
6.24.4.13 currentUserName	192
6.24.4.14 logOut	192
6.25 view.ConsultInitialBoardView Class Reference	192
6.25.1 Detailed Description	195
6.25.2 Constructor & Destructor Documentation	196
6.25.2.1 ConsultInitialBoardView()	196
6.25.3 Member Function Documentation	196
6.25.3.1 initialize()	196
6.25.3.2 goToMenu()	196
6.25.3.3 render()	197
6.25.3.4 drawPiece()	197
6.25.3.5 getCircle()	198
6.25.4 Member Data Documentation	198

6.25.4.1 goToMenu . . . . .	198
6.25.4.2 f1c1 . . . . .	198
6.25.4.3 f1c2 . . . . .	199
6.25.4.4 f1c3 . . . . .	199
6.25.4.5 f1c4 . . . . .	199
6.25.4.6 f1c5 . . . . .	199
6.25.4.7 f1c6 . . . . .	199
6.25.4.8 f1c7 . . . . .	200
6.25.4.9 f1c8 . . . . .	200
6.25.4.10 f2c1 . . . . .	200
6.25.4.11 f2c2 . . . . .	200
6.25.4.12 f2c3 . . . . .	200
6.25.4.13 f2c4 . . . . .	201
6.25.4.14 f2c5 . . . . .	201
6.25.4.15 f2c6 . . . . .	201
6.25.4.16 f2c7 . . . . .	201
6.25.4.17 f2c8 . . . . .	201
6.25.4.18 f3c1 . . . . .	202
6.25.4.19 f3c2 . . . . .	202
6.25.4.20 f3c3 . . . . .	202
6.25.4.21 f3c4 . . . . .	202
6.25.4.22 f3c5 . . . . .	202
6.25.4.23 f3c6 . . . . .	203
6.25.4.24 f3c7 . . . . .	203
6.25.4.25 f3c8 . . . . .	203
6.25.4.26 f4c1 . . . . .	203
6.25.4.27 f4c2 . . . . .	203
6.25.4.28 f4c3 . . . . .	204
6.25.4.29 f4c4 . . . . .	204
6.25.4.30 f4c5 . . . . .	204
6.25.4.31 f4c6 . . . . .	204
6.25.4.32 f4c7 . . . . .	204
6.25.4.33 f4c8 . . . . .	205
6.25.4.34 f5c1 . . . . .	205
6.25.4.35 f5c2 . . . . .	205
6.25.4.36 f5c3 . . . . .	205
6.25.4.37 f5c4 . . . . .	205
6.25.4.38 f5c5 . . . . .	206
6.25.4.39 f5c6 . . . . .	206
6.25.4.40 f5c7 . . . . .	206
6.25.4.41 f5c8 . . . . .	206
6.25.4.42 f6c1 . . . . .	206



6.25.4.43 f6c2 . . . . .	207
6.25.4.44 f6c3 . . . . .	207
6.25.4.45 f6c4 . . . . .	207
6.25.4.46 f6c5 . . . . .	207
6.25.4.47 f6c6 . . . . .	207
6.25.4.48 f6c7 . . . . .	208
6.25.4.49 f6c8 . . . . .	208
6.25.4.50 f7c1 . . . . .	208
6.25.4.51 f7c2 . . . . .	208
6.25.4.52 f7c3 . . . . .	208
6.25.4.53 f7c4 . . . . .	209
6.25.4.54 f7c5 . . . . .	209
6.25.4.55 f7c6 . . . . .	209
6.25.4.56 f7c7 . . . . .	209
6.25.4.57 f7c8 . . . . .	209
6.25.4.58 f8c1 . . . . .	210
6.25.4.59 f8c2 . . . . .	210
6.25.4.60 f8c3 . . . . .	210
6.25.4.61 f8c4 . . . . .	210
6.25.4.62 f8c5 . . . . .	210
6.25.4.63 f8c6 . . . . .	211
6.25.4.64 f8c7 . . . . .	211
6.25.4.65 f8c8 . . . . .	211
6.25.4.66 board . . . . .	211
6.26 domain.Difficulty Class Reference . . . . .	211
6.26.1 Detailed Description . . . . .	212
6.26.2 Constructor & Destructor Documentation . . . . .	213
6.26.2.1 Difficulty() . . . . .	213
6.26.3 Member Function Documentation . . . . .	213
6.26.3.1 getDifficulty() . . . . .	213
6.26.3.2 getCanEatHorizontally() . . . . .	214
6.26.3.3 getCanEatVertically() . . . . .	214
6.26.3.4 getCanEatDiagonally() . . . . .	215
6.26.3.5 getPieceType() . . . . .	215
6.26.3.6 getMaxDepth() . . . . .	216
6.26.3.7 setMaxDepth() . . . . .	216
6.26.3.8 inversePieceType() . . . . .	217
6.26.3.9 place() . . . . .	217
6.26.4 Member Data Documentation . . . . .	218
6.26.4.1 maxDepth . . . . .	218
6.26.4.2 difficulty . . . . .	218
6.26.4.3 canEatHorizontally . . . . .	218

6.26.4.4 canEatVertically . . . . .	218
6.26.4.5 canEatDiagonally . . . . .	219
6.26.4.6 pieceType . . . . .	219
6.27 domain.DifficultyCtrl Class Reference . . . . .	219
6.27.1 Detailed Description . . . . .	220
6.27.2 Constructor & Destructor Documentation . . . . .	220
6.27.2.1 DifficultyCtrl() . . . . .	220
6.27.3 Member Function Documentation . . . . .	220
6.27.3.1 getBestPositionByBot() . . . . .	220
6.27.3.2 getBestPosition() . . . . .	221
6.28 domain.DomainCtrl Class Reference . . . . .	222
6.28.1 Detailed Description . . . . .	225
6.28.2 Constructor & Destructor Documentation . . . . .	226
6.28.2.1 DomainCtrl() . . . . .	226
6.28.3 Member Function Documentation . . . . .	226
6.28.3.1 exitApplication() . . . . .	226
6.28.3.2 logout() . . . . .	227
6.28.3.3 exitGame() . . . . .	227
6.28.3.4 createUser() . . . . .	227
6.28.3.5 createBot() . . . . .	228
6.28.3.6 login() . . . . .	229
6.28.3.7 getUser() . . . . .	229
6.28.3.8 getBot() . . . . .	230
6.28.3.9 getPlayer() . . . . .	231
6.28.3.10 listUsers() . . . . .	231
6.28.3.11 listBots() . . . . .	232
6.28.3.12 modifyUser() . . . . .	232
6.28.3.13 modifyBot() . . . . .	233
6.28.3.14 deleteUser() . . . . .	233
6.28.3.15 deleteBot() . . . . .	234
6.28.3.16 viewUser() . . . . .	235
6.28.3.17 viewPlayers() . . . . .	235
6.28.3.18 createConfiguration() . . . . .	235
6.28.3.19 createInitialBoard() . . . . .	236
6.28.3.20 modifyConfiguration() . . . . .	237
6.28.3.21 modifyInitialBoard() . . . . .	237
6.28.3.22 deleteConfiguration() . . . . .	238
6.28.3.23 getConfiguration() . . . . .	239
6.28.3.24 listConfigurations() . . . . .	239
6.28.3.25 getInitialBoard() . . . . .	240
6.28.3.26 viewConfiguration() . . . . .	240
6.28.3.27 createGame() . . . . .	241

6.28.3.28 saveGame()	241
6.28.3.29 getGame()	242
6.28.3.30 getPlayingBoard()	242
6.28.3.31 listGames()	243
6.28.3.32 selectGame()	243
6.28.3.33 play()	244
6.28.3.34 surrender()	245
6.28.3.35 viewGame()	245
6.28.3.36 stringToPieceType()	246
6.28.3.37 getBestPosition()	246
6.28.3.38 placePieceConfig()	247
6.28.3.39 removePiece()	247
6.28.3.40 getNumPieces()	248
6.28.3.41 validPositions()	248
6.28.3.42 isValidBoard()	249
6.28.3.43 placePiece()	249
6.28.3.44 inversePieceType()	250
6.28.3.45 finishGame()	251
6.28.3.46 nextTurn()	252
6.28.3.47 currentTurn()	252
6.28.3.48 viewBoard()	253
6.28.3.49 getRanking()	253
6.28.3.50 listRankings()	253
6.28.3.51 listRecords()	254
6.28.3.52 createEntries()	254
6.28.4 Member Data Documentation	255
6.28.4.1 playerCtrl	255
6.28.4.2 configurationCtrl	256
6.28.4.3 boardCtrl	256
6.28.4.4 gameCtrl	256
6.28.4.5 rankingCtrl	256
6.28.4.6 difficultyCtrl	256
6.28.4.7 currentUser	257
6.28.4.8 currentPlayer1	257
6.28.4.9 currentPlayer2	257
6.28.4.10 currentBoard	257
6.28.4.11 currentConfiguration	257
6.28.4.12 currentGame	258
6.29 test.driver.Driver Class Reference	258
6.29.1 Detailed Description	258
6.29.2 Member Function Documentation	258
6.29.2.1 menu()	258

6.29.2.2 pause()	259
6.29.2.3 clear()	260
6.29.2.4 input()	260
6.29.2.5 inputInt()	261
6.29.2.6 inputBool()	261
6.30 cmd.driver.easyDifficulty Class Reference	262
6.30.1 Detailed Description	262
6.30.2 Member Function Documentation	262
6.30.2.1 main()	263
6.31 domain.EasyDifficulty Class Reference	263
6.31.1 Detailed Description	263
6.31.2 Constructor & Destructor Documentation	264
6.31.2.1 EasyDifficulty()	264
6.31.3 Member Function Documentation	264
6.31.3.1 evaluation()	264
6.31.3.2 minimax()	266
6.31.3.3 place()	267
6.32 test.driver.EasyDifficultyDriver Class Reference	268
6.32.1 Detailed Description	269
6.32.2 Constructor & Destructor Documentation	269
6.32.2.1 EasyDifficultyDriver()	269
6.32.3 Member Function Documentation	269
6.32.3.1 start()	270
6.32.3.2 mainMenu()	270
6.32.3.3 create()	271
6.32.3.4 getDifficulty()	271
6.32.3.5 getCanEatHorizontally()	271
6.32.3.6 getCanEatVertically()	272
6.32.3.7 getCanEatDiagonally()	272
6.32.3.8 getPieceType()	272
6.32.3.9 getMaxDepth()	272
6.32.3.10 setMaxDepth()	273
6.32.3.11 loadBoard()	273
6.32.3.12 printCurrentBoard()	273
6.32.3.13 getNextBestPosition()	274
6.32.3.14 listBoardFixtures()	274
6.32.3.15 printBoard()	275
6.32.3.16 transcribeToCharacters()	275
6.32.4 Member Data Documentation	275
6.32.4.1 currentEasyDifficulty	275
6.32.4.2 currentBoard	276
6.32.4.3 nameCurrentBoard	276

6.32.4.4 fixtureRepository	276
6.33 cmd.unitary.entry Class Reference	276
6.33.1 Detailed Description	276
6.33.2 Member Function Documentation	276
6.33.2.1 main()	277
6.34 domain.Entry Class Reference	277
6.34.1 Detailed Description	278
6.34.2 Constructor & Destructor Documentation	278
6.34.2.1 Entry() [1/2]	278
6.34.2.2 Entry() [2/2]	278
6.34.3 Member Function Documentation	279
6.34.3.1 serialize()	279
6.34.3.2 getPlayerID()	280
6.34.3.3 getValue()	280
6.34.3.4 setPlayerID()	280
6.34.3.5 setValue()	281
6.34.4 Member Data Documentation	281
6.34.4.1 playerId	281
6.34.4.2 value	282
6.35 test.unitary.EntryJUnit Class Reference	282
6.35.1 Detailed Description	282
6.35.2 Member Function Documentation	282
6.35.2.1 Entry()	282
6.35.2.2 deserialize()	283
6.35.2.3 serialize()	283
6.35.2.4 getPlayerID()	283
6.35.2.5 getValue()	283
6.35.2.6 setPlayerID()	284
6.35.2.7 setValue()	284
6.36 domain.Exceptions Class Reference	284
6.36.1 Detailed Description	285
6.37 domain.Exceptions.ExistingConfigurationException Class Reference	285
6.37.1 Detailed Description	286
6.37.2 Constructor & Destructor Documentation	286
6.37.2.1 ExistingConfigurationException()	286
6.38 domain.Exceptions.ExistingPlayerException Class Reference	286
6.38.1 Detailed Description	286
6.38.2 Constructor & Destructor Documentation	286
6.38.2.1 ExistingPlayerException()	287
6.39 domain.Exceptions.FinishedGameException Class Reference	287
6.39.1 Detailed Description	287
6.39.2 Constructor & Destructor Documentation	287

6.39.2.1 FinishedGameException()	287
6.40 repository.FixtureRepository Class Reference	287
6.40.1 Detailed Description	288
6.40.2 Constructor & Destructor Documentation	288
6.40.2.1 FixtureRepository()	288
6.40.3 Member Function Documentation	288
6.40.3.1 listFiles()	289
6.40.3.2 getLines()	289
6.40.3.3 boardFileToJSON()	290
6.41 domain.Game Class Reference	291
6.41.1 Detailed Description	292
6.41.2 Constructor & Destructor Documentation	292
6.41.2.1 Game() [1/2]	293
6.41.2.2 Game() [2/2]	293
6.41.3 Member Function Documentation	295
6.41.3.1 serialize()	295
6.41.3.2 getName()	296
6.41.3.3 setName()	296
6.41.3.4 getPlayer1ID()	297
6.41.3.5 getPlayer2ID()	297
6.41.3.6 getConfigurationName()	298
6.41.3.7 setConfigurationName()	298
6.41.3.8 getTurn()	299
6.41.3.9 setTurn()	299
6.41.3.10 getState()	300
6.41.3.11 setState()	300
6.41.3.12 getWinnerID()	301
6.41.3.13 getCreatorID()	301
6.41.3.14 getCreatedAt()	302
6.41.3.15 play()	302
6.41.3.16 surrender()	303
6.41.3.17 finish()	303
6.41.3.18 checkPlaceRights()	304
6.41.3.19 nextTurn()	305
6.41.4 Member Data Documentation	305
6.41.4.1 name	305
6.41.4.2 player1ID	306
6.41.4.3 player2ID	306
6.41.4.4 configurationName	306
6.41.4.5 turn	306
6.41.4.6 state	306
6.41.4.7 winnerID	307

6.41.4.8 creatorID . . . . .	307
6.41.4.9 createdAt . . . . .	307
6.42 cmd.driver.game Class Reference . . . . .	307
6.42.1 Detailed Description . . . . .	307
6.42.2 Member Function Documentation . . . . .	308
6.42.2.1 main() . . . . .	308
6.43 view.GameBoardView Class Reference . . . . .	308
6.43.1 Detailed Description . . . . .	312
6.43.2 Constructor & Destructor Documentation . . . . .	312
6.43.2.1 GameBoardView() . . . . .	313
6.43.3 Member Function Documentation . . . . .	313
6.43.3.1 initialize() . . . . .	313
6.43.3.2 goToMenu() . . . . .	314
6.43.3.3 save() . . . . .	314
6.43.3.4 surrender() . . . . .	315
6.43.3.5 transform() . . . . .	315
6.43.3.6 renderState() . . . . .	316
6.43.3.7 renderResult() . . . . .	317
6.43.3.8 render() . . . . .	318
6.43.3.9 drawPiece() . . . . .	319
6.43.3.10 getClickedPos() . . . . .	319
6.43.3.11 getCircle() . . . . .	320
6.43.3.12 onChangeAssistedMode() . . . . .	320
6.43.4 Member Data Documentation . . . . .	321
6.43.4.1 goToMenu . . . . .	321
6.43.4.2 f1c1 . . . . .	321
6.43.4.3 f1c2 . . . . .	321
6.43.4.4 f1c3 . . . . .	321
6.43.4.5 f1c4 . . . . .	321
6.43.4.6 f1c5 . . . . .	322
6.43.4.7 f1c6 . . . . .	322
6.43.4.8 f1c7 . . . . .	322
6.43.4.9 f1c8 . . . . .	322
6.43.4.10 f2c1 . . . . .	322
6.43.4.11 f2c2 . . . . .	323
6.43.4.12 f2c3 . . . . .	323
6.43.4.13 f2c4 . . . . .	323
6.43.4.14 f2c5 . . . . .	323
6.43.4.15 f2c6 . . . . .	323
6.43.4.16 f2c7 . . . . .	324
6.43.4.17 f2c8 . . . . .	324
6.43.4.18 f3c1 . . . . .	324

6.43.4.19 f3c2 . . . . .	324
6.43.4.20 f3c3 . . . . .	324
6.43.4.21 f3c4 . . . . .	325
6.43.4.22 f3c5 . . . . .	325
6.43.4.23 f3c6 . . . . .	325
6.43.4.24 f3c7 . . . . .	325
6.43.4.25 f3c8 . . . . .	325
6.43.4.26 f4c1 . . . . .	326
6.43.4.27 f4c2 . . . . .	326
6.43.4.28 f4c3 . . . . .	326
6.43.4.29 f4c4 . . . . .	326
6.43.4.30 f4c5 . . . . .	326
6.43.4.31 f4c6 . . . . .	327
6.43.4.32 f4c7 . . . . .	327
6.43.4.33 f4c8 . . . . .	327
6.43.4.34 f5c1 . . . . .	327
6.43.4.35 f5c2 . . . . .	327
6.43.4.36 f5c3 . . . . .	328
6.43.4.37 f5c4 . . . . .	328
6.43.4.38 f5c5 . . . . .	328
6.43.4.39 f5c6 . . . . .	328
6.43.4.40 f5c7 . . . . .	328
6.43.4.41 f5c8 . . . . .	329
6.43.4.42 f6c1 . . . . .	329
6.43.4.43 f6c2 . . . . .	329
6.43.4.44 f6c3 . . . . .	329
6.43.4.45 f6c4 . . . . .	329
6.43.4.46 f6c5 . . . . .	330
6.43.4.47 f6c6 . . . . .	330
6.43.4.48 f6c7 . . . . .	330
6.43.4.49 f6c8 . . . . .	330
6.43.4.50 f7c1 . . . . .	330
6.43.4.51 f7c2 . . . . .	331
6.43.4.52 f7c3 . . . . .	331
6.43.4.53 f7c4 . . . . .	331
6.43.4.54 f7c5 . . . . .	331
6.43.4.55 f7c6 . . . . .	331
6.43.4.56 f7c7 . . . . .	332
6.43.4.57 f7c8 . . . . .	332
6.43.4.58 f8c1 . . . . .	332
6.43.4.59 f8c2 . . . . .	332
6.43.4.60 f8c3 . . . . .	332



6.43.4.61 f8c4 . . . . .	333
6.43.4.62 f8c5 . . . . .	333
6.43.4.63 f8c6 . . . . .	333
6.43.4.64 f8c7 . . . . .	333
6.43.4.65 f8c8 . . . . .	333
6.43.4.66 save . . . . .	334
6.43.4.67 saveButton . . . . .	334
6.43.4.68 surrender . . . . .	334
6.43.4.69 surrenderButton . . . . .	334
6.43.4.70 tielcon . . . . .	334
6.43.4.71 winIcon . . . . .	335
6.43.4.72 gameResult . . . . .	335
6.43.4.73 player2 . . . . .	335
6.43.4.74 player2Turn . . . . .	335
6.43.4.75 player2Pieces . . . . .	335
6.43.4.76 player2Type . . . . .	336
6.43.4.77 player1 . . . . .	336
6.43.4.78 player1Turn . . . . .	336
6.43.4.79 player1Pieces . . . . .	336
6.43.4.80 player1Type . . . . .	336
6.43.4.81 assistedMode . . . . .	337
6.43.4.82 board . . . . .	337
6.43.4.83 game . . . . .	337
6.43.4.84 players . . . . .	337
6.43.4.85 user . . . . .	337
6.43.4.86 turnPlayerID . . . . .	338
6.43.4.87 turnPlayerIsBot . . . . .	338
6.43.4.88 turnPieceType . . . . .	338
6.43.4.89 isSpectating . . . . .	338
6.43.4.90 isVsBot . . . . .	338
6.43.4.91 timer . . . . .	339
6.44 domain.GameCtrl Class Reference . . . . .	339
6.44.1 Detailed Description . . . . .	340
6.44.2 Constructor & Destructor Documentation . . . . .	340
6.44.2.1 GameCtrl() . . . . .	340
6.44.3 Member Function Documentation . . . . .	340
6.44.3.1 create() . . . . .	341
6.44.3.2 getGame() . . . . .	342
6.44.3.3 getPlayingBoard() . . . . .	343
6.44.3.4 list() . . . . .	343
6.44.3.5 save() . . . . .	344
6.44.3.6 play() . . . . .	345

6.44.3.7	surrender()	345
6.44.3.8	finish()	346
6.44.3.9	checkPlaceRights()	346
6.44.3.10	nextTurn()	347
6.44.4	Member Data Documentation	348
6.44.4.1	repositoryCtrl	348
6.44.4.2	configurationRepositoryCtrl	348
6.44.4.3	playerRepositoryCtrl	348
6.45	test.driver.GameDriver Class Reference	348
6.45.1	Detailed Description	349
6.45.2	Constructor & Destructor Documentation	349
6.45.2.1	GameDriver()	349
6.45.3	Member Function Documentation	350
6.45.3.1	start()	350
6.45.3.2	mainMenu()	350
6.45.3.3	create()	351
6.45.3.4	getName()	351
6.45.3.5	setName()	352
6.45.3.6	getPlayer1ID()	352
6.45.3.7	getPlayer2ID()	352
6.45.3.8	getConfigurationName()	353
6.45.3.9	setConfigurationName()	353
6.45.3.10	getTurn()	353
6.45.3.11	setTurn()	354
6.45.3.12	getState()	354
6.45.3.13	setState()	354
6.45.3.14	getWinnerID()	355
6.45.3.15	getCreatorID()	355
6.45.3.16	getCreatedAt()	355
6.45.3.17	serialize()	355
6.45.3.18	deserialize()	356
6.45.3.19	play()	356
6.45.3.20	surrender()	356
6.45.3.21	finish()	357
6.45.3.22	checkPlaceRights()	357
6.45.3.23	nextTurn()	358
6.45.4	Member Data Documentation	358
6.45.4.1	currentGame	358
6.46	repository.GameRepository Class Reference	359
6.46.1	Detailed Description	359
6.46.2	Constructor & Destructor Documentation	359
6.46.2.1	GameRepository()	359

6.46.3 Member Function Documentation	360
6.46.3.1 save()	360
6.46.3.2 getGame()	360
6.46.3.3 getBoard()	361
6.46.3.4 existsGameByConfigurationName()	362
6.46.3.5 existsGameByPlayerID()	362
6.46.3.6 listGames()	363
6.47 repository.GameRepositoryCtrl Class Reference	364
6.47.1 Detailed Description	364
6.47.2 Constructor & Destructor Documentation	365
6.47.2.1 GameRepositoryCtrl()	365
6.47.3 Member Function Documentation	365
6.47.3.1 save()	365
6.47.3.2 getGame()	366
6.47.3.3 getBoard()	366
6.47.3.4 existsGameByConfigurationName()	367
6.47.3.5 existsGameByPlayerID()	367
6.47.3.6 listGames()	368
6.47.4 Member Data Documentation	369
6.47.4.1 repository	369
6.48 view.GamesCreateView Class Reference	369
6.48.1 Detailed Description	371
6.48.2 Constructor & Destructor Documentation	371
6.48.2.1 GamesCreateView()	371
6.48.3 Member Function Documentation	371
6.48.3.1 initialize()	371
6.48.3.2 user()	372
6.48.3.3 bots()	372
6.48.3.4 config()	372
6.48.3.5 ranking()	373
6.48.3.6 play()	373
6.48.3.7 createGame()	373
6.48.3.8 userChooser1()	374
6.48.3.9 userChooser2()	374
6.48.3.10 botChooser1()	374
6.48.3.11 botChooser2()	375
6.48.3.12 createGameConfirm()	375
6.48.3.13 logOut()	376
6.48.4 Member Data Documentation	376
6.48.4.1 user	376
6.48.4.2 bots	377
6.48.4.3 config	377

6.48.4.4 games	377
6.48.4.5 ranking	377
6.48.4.6 play	377
6.48.4.7 createGame	378
6.48.4.8 createGameButton	378
6.48.4.9 configChooser	378
6.48.4.10 pl1User	378
6.48.4.11 pl1Bot	378
6.48.4.12 userChooser1	379
6.48.4.13 botChooser1	379
6.48.4.14 pl2User	379
6.48.4.15 pl2Bot	379
6.48.4.16 userChooser2	379
6.48.4.17 botChooser2	380
6.48.4.18 createGameConfirm	380
6.48.4.19 createGameConfirmButton	380
6.48.4.20 createGameResult	380
6.48.4.21 currentUserName	380
6.48.4.22 logOut	381
6.48.4.23 userMap	381
6.48.4.24 botMap	381
6.49 domain.Game.GameState Enum Reference	381
6.49.1 Detailed Description	381
6.49.2 Member Data Documentation	382
6.49.2.1 NOT_STARTED	382
6.49.2.2 IN_PROGRESS	382
6.49.2.3 FINISHED	382
6.50 view.GamesView Class Reference	382
6.50.1 Detailed Description	383
6.50.2 Constructor & Destructor Documentation	383
6.50.2.1 GamesView()	383
6.50.3 Member Function Documentation	383
6.50.3.1 initialize()	384
6.50.3.2 user()	384
6.50.3.3 bots()	384
6.50.3.4 config()	385
6.50.3.5 ranking()	385
6.50.3.6 play()	385
6.50.3.7 createGame()	386
6.50.3.8 logOut()	386
6.50.4 Member Data Documentation	386
6.50.4.1 user	387

6.50.4.2 bots	387
6.50.4.3 config	387
6.50.4.4 games	387
6.50.4.5 ranking	387
6.50.4.6 play	388
6.50.4.7 createGame	388
6.50.4.8 createGameButton	388
6.50.4.9 currentUserName	388
6.50.4.10 logOut	388
6.51 domain.HardDifficulty Class Reference	389
6.51.1 Detailed Description	389
6.51.2 Constructor & Destructor Documentation	389
6.51.2.1 HardDifficulty()	390
6.51.3 Member Function Documentation	390
6.51.3.1 place()	390
6.51.4 Member Data Documentation	391
6.51.4.1 random	391
6.51.4.2 epsilon	391
6.52 cmd.driver.hardDifficulty Class Reference	392
6.52.1 Detailed Description	392
6.52.2 Member Function Documentation	392
6.52.2.1 main()	392
6.53 test.driver.HardDifficultyDriver Class Reference	392
6.53.1 Detailed Description	393
6.53.2 Constructor & Destructor Documentation	393
6.53.2.1 HardDifficultyDriver()	393
6.53.3 Member Function Documentation	394
6.53.3.1 start()	394
6.53.3.2 mainMenu()	394
6.53.3.3 create()	395
6.53.3.4 getDifficulty()	395
6.53.3.5 getCanEatHorizontally()	395
6.53.3.6 getCanEatVertically()	396
6.53.3.7 getCanEatDiagonally()	396
6.53.3.8 getPieceType()	396
6.53.3.9 getMaxDepth()	396
6.53.3.10 setMaxDepth()	397
6.53.3.11 loadBoard()	397
6.53.3.12 printCurrentBoard()	397
6.53.3.13 getNextBestPosition()	398
6.53.3.14 listBoardFixtures()	398
6.53.3.15 printBoard()	399

6.53.3.16 transcribeToCharacters()	399
6.53.4 Member Data Documentation	399
6.53.4.1 currentHardDifficulty	399
6.53.4.2 currentBoard	400
6.53.4.3 nameCurrentBoard	400
6.53.4.4 fixtureRepository	400
6.54 domain.Exceptions.IncorrectCredentialsException Class Reference	400
6.54.1 Detailed Description	400
6.54.2 Constructor & Destructor Documentation	400
6.54.2.1 IncorrectCredentialsException()	401
6.55 domain.Exceptions.InexistingConfigurationException Class Reference	401
6.55.1 Detailed Description	401
6.55.2 Constructor & Destructor Documentation	401
6.55.2.1 InexistingConfigurationException()	401
6.56 domain.Exceptions.InexistingPlayerException Class Reference	402
6.56.1 Detailed Description	402
6.56.2 Constructor & Destructor Documentation	402
6.56.2.1 InexistingPlayerException()	402
6.57 view.InitialBoardView Class Reference	402
6.57.1 Detailed Description	406
6.57.2 Constructor & Destructor Documentation	406
6.57.2.1 InitialBoardView()	406
6.57.3 Member Function Documentation	406
6.57.3.1 initialize()	406
6.57.3.2 transform()	407
6.57.3.3 save()	407
6.57.3.4 render()	408
6.57.3.5 drawPiece()	408
6.57.3.6 getClickedPos()	409
6.57.3.7 getCircle()	409
6.57.4 Member Data Documentation	409
6.57.4.1 f1c1	410
6.57.4.2 f1c2	410
6.57.4.3 f1c3	410
6.57.4.4 f1c4	410
6.57.4.5 f1c5	410
6.57.4.6 f1c6	411
6.57.4.7 f1c7	411
6.57.4.8 f1c8	411
6.57.4.9 f2c1	411
6.57.4.10 f2c2	411
6.57.4.11 f2c3	412

6.57.4.12 f2c4 . . . . .	412
6.57.4.13 f2c5 . . . . .	412
6.57.4.14 f2c6 . . . . .	412
6.57.4.15 f2c7 . . . . .	412
6.57.4.16 f2c8 . . . . .	413
6.57.4.17 f3c1 . . . . .	413
6.57.4.18 f3c2 . . . . .	413
6.57.4.19 f3c3 . . . . .	413
6.57.4.20 f3c4 . . . . .	413
6.57.4.21 f3c5 . . . . .	414
6.57.4.22 f3c6 . . . . .	414
6.57.4.23 f3c7 . . . . .	414
6.57.4.24 f3c8 . . . . .	414
6.57.4.25 f4c1 . . . . .	414
6.57.4.26 f4c2 . . . . .	415
6.57.4.27 f4c3 . . . . .	415
6.57.4.28 f4c4 . . . . .	415
6.57.4.29 f4c5 . . . . .	415
6.57.4.30 f4c6 . . . . .	415
6.57.4.31 f4c7 . . . . .	416
6.57.4.32 f4c8 . . . . .	416
6.57.4.33 f5c1 . . . . .	416
6.57.4.34 f5c2 . . . . .	416
6.57.4.35 f5c3 . . . . .	416
6.57.4.36 f5c4 . . . . .	417
6.57.4.37 f5c5 . . . . .	417
6.57.4.38 f5c6 . . . . .	417
6.57.4.39 f5c7 . . . . .	417
6.57.4.40 f5c8 . . . . .	417
6.57.4.41 f6c1 . . . . .	418
6.57.4.42 f6c2 . . . . .	418
6.57.4.43 f6c3 . . . . .	418
6.57.4.44 f6c4 . . . . .	418
6.57.4.45 f6c5 . . . . .	418
6.57.4.46 f6c6 . . . . .	419
6.57.4.47 f6c7 . . . . .	419
6.57.4.48 f6c8 . . . . .	419
6.57.4.49 f7c1 . . . . .	419
6.57.4.50 f7c2 . . . . .	419
6.57.4.51 f7c3 . . . . .	420
6.57.4.52 f7c4 . . . . .	420
6.57.4.53 f7c5 . . . . .	420

6.57.4.54 f7c6 . . . . .	420
6.57.4.55 f7c7 . . . . .	420
6.57.4.56 f7c8 . . . . .	421
6.57.4.57 f8c1 . . . . .	421
6.57.4.58 f8c2 . . . . .	421
6.57.4.59 f8c3 . . . . .	421
6.57.4.60 f8c4 . . . . .	421
6.57.4.61 f8c5 . . . . .	422
6.57.4.62 f8c6 . . . . .	422
6.57.4.63 f8c7 . . . . .	422
6.57.4.64 f8c8 . . . . .	422
6.57.4.65 save . . . . .	422
6.57.4.66 saveButton . . . . .	423
6.57.4.67 placeWhitePieces . . . . .	423
6.57.4.68 placeBlackPieces . . . . .	423
6.57.4.69 quitPieces . . . . .	423
6.57.4.70 saveInitialBoardResult . . . . .	423
6.57.4.71 board . . . . .	424
6.58 domain.Exceptions.InvalidBoardException Class Reference . . . . .	424
6.58.1 Detailed Description . . . . .	424
6.58.2 Constructor & Destructor Documentation . . . . .	424
6.58.2.1 InvalidBoardException() . . . . .	424
6.59 domain.Exceptions.InvalidConfigurationException Class Reference . . . . .	424
6.59.1 Detailed Description . . . . .	425
6.59.2 Constructor & Destructor Documentation . . . . .	425
6.59.2.1 InvalidConfigurationException() . . . . .	425
6.60 domain.Exceptions.InvalidDifficultyException Class Reference . . . . .	425
6.60.1 Detailed Description . . . . .	425
6.60.2 Constructor & Destructor Documentation . . . . .	425
6.60.2.1 InvalidDifficultyException() . . . . .	426
6.61 domain.Exceptions.InvalidNameException Class Reference . . . . .	426
6.61.1 Detailed Description . . . . .	426
6.61.2 Constructor & Destructor Documentation . . . . .	426
6.61.2.1 InvalidNameException() . . . . .	426
6.62 domain.Exceptions.InvalidPasswordException Class Reference . . . . .	426
6.62.1 Detailed Description . . . . .	427
6.62.2 Constructor & Destructor Documentation . . . . .	427
6.62.2.1 InvalidPasswordException() . . . . .	427
6.63 domain.Exceptions.InvalidPlayersException Class Reference . . . . .	427
6.63.1 Detailed Description . . . . .	427
6.63.2 Constructor & Destructor Documentation . . . . .	427
6.63.2.1 InvalidPlayersException() . . . . .	428



6.64 domain.Exceptions.InvalidPositionException Class Reference	428
6.64.1 Detailed Description	428
6.64.2 Constructor & Destructor Documentation	428
6.64.2.1 InvalidPositionException()	428
6.65 domain.Exceptions.InvalidRulesException Class Reference	428
6.65.1 Detailed Description	429
6.65.2 Constructor & Destructor Documentation	429
6.65.2.1 InvalidRulesException()	429
6.66 view.LoginView Class Reference	429
6.66.1 Detailed Description	430
6.66.2 Constructor & Destructor Documentation	430
6.66.2.1 LoginView()	430
6.66.3 Member Function Documentation	430
6.66.3.1 signIn()	430
6.66.3.2 signUp()	431
6.66.4 Member Data Documentation	431
6.66.4.1 login	431
6.66.4.2 signUp	431
6.66.4.3 username	431
6.66.4.4 password	432
6.66.4.5 loginResult	432
6.66.4.6 signIn	432
6.66.4.7 signInButton	432
6.67 cmd.driver.mediumDifficulty Class Reference	432
6.67.1 Detailed Description	433
6.67.2 Member Function Documentation	433
6.67.2.1 main()	433
6.68 domain.MediumDifficulty Class Reference	433
6.68.1 Detailed Description	434
6.68.2 Constructor & Destructor Documentation	434
6.68.2.1 MediumDifficulty()	434
6.68.3 Member Function Documentation	435
6.68.3.1 evaluation()	435
6.68.3.2 minimax()	436
6.68.3.3 place()	438
6.69 test.driver.MediumDifficultyDriver Class Reference	439
6.69.1 Detailed Description	440
6.69.2 Constructor & Destructor Documentation	440
6.69.2.1 MediumDifficultyDriver()	440
6.69.3 Member Function Documentation	440
6.69.3.1 start()	440
6.69.3.2 mainMenu()	441

6.69.3.3 create()	441
6.69.3.4 getDifficulty()	442
6.69.3.5 getCanEatHorizontally()	442
6.69.3.6 getCanEatVertically()	442
6.69.3.7 getCanEatDiagonally()	443
6.69.3.8 getPieceType()	443
6.69.3.9 getMaxDepth()	443
6.69.3.10 setMaxDepth()	443
6.69.3.11 loadBoard()	444
6.69.3.12 printCurrentBoard()	444
6.69.3.13 getNextBestPosition()	444
6.69.3.14 listBoardFixtures()	445
6.69.3.15 printBoard()	445
6.69.3.16 transcribeToCharacters()	446
6.69.4 Member Data Documentation	446
6.69.4.1 currentMediumDifficulty	446
6.69.4.2 currentBoard	446
6.69.4.3 nameCurrentBoard	446
6.69.4.4 fixtureRepository	447
6.70 view.ModifyInitialBoardView Class Reference	447
6.70.1 Detailed Description	450
6.70.2 Constructor & Destructor Documentation	450
6.70.2.1 ModifyInitialBoardView()	450
6.70.3 Member Function Documentation	451
6.70.3.1 initialize()	451
6.70.3.2 transform()	451
6.70.3.3 save()	452
6.70.3.4 render()	452
6.70.3.5 drawPiece()	453
6.70.3.6 getClickedPos()	453
6.70.3.7 getCircle()	454
6.70.4 Member Data Documentation	454
6.70.4.1 f1c1	454
6.70.4.2 f1c2	454
6.70.4.3 f1c3	455
6.70.4.4 f1c4	455
6.70.4.5 f1c5	455
6.70.4.6 f1c6	455
6.70.4.7 f1c7	455
6.70.4.8 f1c8	456
6.70.4.9 f2c1	456
6.70.4.10 f2c2	456

6.70.4.11 f2c3 . . . . .	456
6.70.4.12 f2c4 . . . . .	456
6.70.4.13 f2c5 . . . . .	457
6.70.4.14 f2c6 . . . . .	457
6.70.4.15 f2c7 . . . . .	457
6.70.4.16 f2c8 . . . . .	457
6.70.4.17 f3c1 . . . . .	457
6.70.4.18 f3c2 . . . . .	458
6.70.4.19 f3c3 . . . . .	458
6.70.4.20 f3c4 . . . . .	458
6.70.4.21 f3c5 . . . . .	458
6.70.4.22 f3c6 . . . . .	458
6.70.4.23 f3c7 . . . . .	459
6.70.4.24 f3c8 . . . . .	459
6.70.4.25 f4c1 . . . . .	459
6.70.4.26 f4c2 . . . . .	459
6.70.4.27 f4c3 . . . . .	459
6.70.4.28 f4c4 . . . . .	460
6.70.4.29 f4c5 . . . . .	460
6.70.4.30 f4c6 . . . . .	460
6.70.4.31 f4c7 . . . . .	460
6.70.4.32 f4c8 . . . . .	460
6.70.4.33 f5c1 . . . . .	461
6.70.4.34 f5c2 . . . . .	461
6.70.4.35 f5c3 . . . . .	461
6.70.4.36 f5c4 . . . . .	461
6.70.4.37 f5c5 . . . . .	461
6.70.4.38 f5c6 . . . . .	462
6.70.4.39 f5c7 . . . . .	462
6.70.4.40 f5c8 . . . . .	462
6.70.4.41 f6c1 . . . . .	462
6.70.4.42 f6c2 . . . . .	462
6.70.4.43 f6c3 . . . . .	463
6.70.4.44 f6c4 . . . . .	463
6.70.4.45 f6c5 . . . . .	463
6.70.4.46 f6c6 . . . . .	463
6.70.4.47 f6c7 . . . . .	463
6.70.4.48 f6c8 . . . . .	464
6.70.4.49 f7c1 . . . . .	464
6.70.4.50 f7c2 . . . . .	464
6.70.4.51 f7c3 . . . . .	464
6.70.4.52 f7c4 . . . . .	464

6.70.4.53 f7c5 . . . . .	465
6.70.4.54 f7c6 . . . . .	465
6.70.4.55 f7c7 . . . . .	465
6.70.4.56 f7c8 . . . . .	465
6.70.4.57 f8c1 . . . . .	465
6.70.4.58 f8c2 . . . . .	466
6.70.4.59 f8c3 . . . . .	466
6.70.4.60 f8c4 . . . . .	466
6.70.4.61 f8c5 . . . . .	466
6.70.4.62 f8c6 . . . . .	466
6.70.4.63 f8c7 . . . . .	467
6.70.4.64 f8c8 . . . . .	467
6.70.4.65 save . . . . .	467
6.70.4.66 saveButton . . . . .	467
6.70.4.67 placeWhitePieces . . . . .	467
6.70.4.68 placeBlackPieces . . . . .	468
6.70.4.69 quitPieces . . . . .	468
6.70.4.70 saveInitialBoardResult . . . . .	468
6.70.4.71 board . . . . .	468
6.71 domain.Exceptions.NotCreatorException Class Reference . . . . .	468
6.71.1 Detailed Description . . . . .	469
6.71.2 Constructor & Destructor Documentation . . . . .	469
6.71.2.1 NotCreatorException() . . . . .	469
6.72 domain.Exceptions.NotPlayerException Class Reference . . . . .	469
6.72.1 Detailed Description . . . . .	469
6.72.2 Constructor & Destructor Documentation . . . . .	469
6.72.2.1 NotPlayerException() . . . . .	470
6.73 domain.Exceptions.NotPlayerPieceException Class Reference . . . . .	470
6.73.1 Detailed Description . . . . .	470
6.73.2 Constructor & Destructor Documentation . . . . .	470
6.73.2.1 NotPlayerPieceException() . . . . .	470
6.74 domain.Exceptions.NotPlayerTurnException Class Reference . . . . .	470
6.74.1 Detailed Description . . . . .	471
6.74.2 Constructor & Destructor Documentation . . . . .	471
6.74.2.1 NotPlayerTurnException() . . . . .	471
6.75 domain.Exceptions.NotStartedGameException Class Reference . . . . .	471
6.75.1 Detailed Description . . . . .	471
6.75.2 Constructor & Destructor Documentation . . . . .	471
6.75.2.1 NotStartedGameException() . . . . .	472
6.76 cmd.othello Class Reference . . . . .	472
6.76.1 Detailed Description . . . . .	472
6.76.2 Member Function Documentation . . . . .	472

6.76.2.1 main()	472
6.77 cmd.driver.pair Class Reference	473
6.77.1 Detailed Description	473
6.77.2 Member Function Documentation	473
6.77.2.1 main()	473
6.78 util.Pair< F, S > Class Template Reference	473
6.78.1 Detailed Description	474
6.78.2 Constructor & Destructor Documentation	474
6.78.2.1 Pair()	474
6.78.3 Member Function Documentation	475
6.78.3.1 equals()	475
6.78.3.2 toString()	475
6.78.3.3 getFirst()	476
6.78.3.4 getSecond()	476
6.78.4 Member Data Documentation	477
6.78.4.1 first	477
6.78.4.2 second	477
6.79 test.driver.PairDriver Class Reference	477
6.79.1 Detailed Description	478
6.79.2 Constructor & Destructor Documentation	478
6.79.2.1 PairDriver()	478
6.79.3 Member Function Documentation	478
6.79.3.1 start()	478
6.79.3.2 mainMenu()	479
6.79.3.3 currentPair()	479
6.79.3.4 resetPairs()	479
6.79.3.5 createStrPair()	480
6.79.3.6 createIntPair()	480
6.79.3.7 createStrIntPair()	480
6.79.3.8 getFirst()	480
6.79.3.9 getSecond()	481
6.79.3.10 comparePair()	481
6.79.4 Member Data Documentation	481
6.79.4.1 currentStrPair	482
6.79.4.2 currentIntPair	482
6.79.4.3 currentStrIntPair	482
6.80 domain.Board.PieceType Enum Reference	482
6.80.1 Detailed Description	482
6.80.2 Member Data Documentation	483
6.80.2.1 PLAYER1	483
6.80.2.2 PLAYER2	483
6.81 domain.Player Class Reference	483

6.81.1 Detailed Description	484
6.81.2 Member Function Documentation	484
6.81.2.1 getName()	484
6.81.2.2 getID()	484
6.81.2.3 getIsDeleted()	485
6.81.2.4 setName()	485
6.81.2.5 setIsDeleted()	486
6.81.3 Member Data Documentation	486
6.81.3.1 id	486
6.81.3.2 name	486
6.81.3.3 isDeleted	486
6.82 domain.PlayerCtrl Class Reference	487
6.82.1 Detailed Description	488
6.82.2 Constructor & Destructor Documentation	488
6.82.2.1 PlayerCtrl()	488
6.82.3 Member Function Documentation	488
6.82.3.1 createUser()	489
6.82.3.2 createBot()	489
6.82.3.3 saveUser()	490
6.82.3.4 saveBot()	490
6.82.3.5 login()	491
6.82.3.6 getUser()	492
6.82.3.7 getBot()	492
6.82.3.8 listUsers()	493
6.82.3.9 listBots()	493
6.82.3.10 modifyUser()	494
6.82.3.11 modifyBot()	494
6.82.3.12 deleteUser()	495
6.82.3.13 deleteBot()	496
6.82.3.14 hash()	497
6.82.4 Member Data Documentation	497
6.82.4.1 repositoryCtrl	497
6.82.4.2 gameRepositoryCtrl	498
6.83 repository.PlayerRepository Class Reference	498
6.83.1 Detailed Description	498
6.83.2 Constructor & Destructor Documentation	499
6.83.2.1 PlayerRepository()	499
6.83.3 Member Function Documentation	499
6.83.3.1 save()	499
6.83.3.2 delete()	500
6.83.3.3 get()	500
6.83.3.4 getByName()	501

6.83.3.5 listUsers()	502
6.83.3.6 listBots()	502
6.84 repository.PlayerRepositoryCtrl Class Reference	503
6.84.1 Detailed Description	503
6.84.2 Constructor & Destructor Documentation	503
6.84.2.1 PlayerRepositoryCtrl()	504
6.84.3 Member Function Documentation	504
6.84.3.1 save()	504
6.84.3.2 delete()	505
6.84.3.3 get()	505
6.84.3.4 getByName()	506
6.84.3.5 listUsers()	506
6.84.3.6 listBots()	507
6.84.4 Member Data Documentation	507
6.84.4.1 repository	507
6.85 view.PlayView Class Reference	508
6.85.1 Detailed Description	509
6.85.2 Constructor & Destructor Documentation	509
6.85.2.1 PlayView()	509
6.85.3 Member Function Documentation	509
6.85.3.1 initialize()	510
6.85.3.2 user()	510
6.85.3.3 bots()	510
6.85.3.4 config()	511
6.85.3.5 games()	511
6.85.3.6 onChangeGameChooser()	512
6.85.3.7 ranking()	513
6.85.3.8 goToGame()	513
6.85.3.9 logOut()	514
6.85.4 Member Data Documentation	514
6.85.4.1 user	514
6.85.4.2 bots	514
6.85.4.3 config	515
6.85.4.4 games	515
6.85.4.5 ranking	515
6.85.4.6 play	515
6.85.4.7 player1	515
6.85.4.8 player2	516
6.85.4.9 configuration	516
6.85.4.10 creator	516
6.85.4.11 createdAt	516
6.85.4.12 state	516

6.85.4.13 info	517
6.85.4.14 playResult	517
6.85.4.15 playGame	517
6.85.4.16 playGameButton	517
6.85.4.17 goToGame	517
6.85.4.18 goToGameButton	518
6.85.4.19 currentUserName	518
6.85.4.20 logOut	518
6.85.4.21 gameChooser	518
6.86 cmd.unitary.ranking Class Reference	518
6.86.1 Detailed Description	519
6.86.2 Member Function Documentation	519
6.86.2.1 main()	519
6.87 domain.Ranking Class Reference	519
6.87.1 Detailed Description	520
6.87.2 Constructor & Destructor Documentation	520
6.87.2.1 Ranking() [1/2]	521
6.87.2.2 Ranking() [2/2]	521
6.87.3 Member Function Documentation	522
6.87.3.1 serialize()	522
6.87.3.2 getRecord()	522
6.87.3.3 removePlayer()	523
6.87.3.4 whereInsert()	523
6.87.3.5 addEntry()	524
6.87.3.6 getName()	525
6.87.3.7 getEntries()	525
6.87.4 Member Data Documentation	525
6.87.4.1 name	526
6.87.4.2 entries	526
6.88 view.RankingConsultView Class Reference	526
6.88.1 Detailed Description	527
6.88.2 Constructor & Destructor Documentation	528
6.88.2.1 RankingConsultView()	528
6.88.3 Member Function Documentation	528
6.88.3.1 initialize()	528
6.88.3.2 user()	528
6.88.3.3 bots()	529
6.88.3.4 config()	529
6.88.3.5 games()	529
6.88.3.6 play()	530
6.88.3.7 onChangeRankingChooser()	530
6.88.3.8 consultRankings()	531



6.88.3.9 consultRecords()	531
6.88.3.10 logOut()	531
6.88.4 Member Data Documentation	532
6.88.4.1 user	532
6.88.4.2 bots	532
6.88.4.3 config	532
6.88.4.4 games	532
6.88.4.5 ranking	532
6.88.4.6 play	533
6.88.4.7 consultRankings	533
6.88.4.8 consultRankingsButton	533
6.88.4.9 consultRankingsConfirm	533
6.88.4.10 consultRankingsConfirmButton	533
6.88.4.11 consultRecords	534
6.88.4.12 consultRecordsButton	534
6.88.4.13 rankingChooser	534
6.88.4.14 rankingInfo	534
6.88.4.15 currentUserName	534
6.88.4.16 logOut	535
6.88.4.17 table	535
6.88.4.18 playerColumn	535
6.88.4.19 valueColumn	535
6.89 domain.RankingCtrl Class Reference	535
6.89.1 Detailed Description	536
6.89.2 Constructor & Destructor Documentation	536
6.89.2.1 RankingCtrl()	536
6.89.3 Member Function Documentation	537
6.89.3.1 getRanking()	537
6.89.3.2 listRankings()	537
6.89.3.3 listRecords()	538
6.89.3.4 createEntry()	538
6.89.4 Member Data Documentation	539
6.89.4.1 repositoryCtrl	540
6.90 test.unitary.RankingJUnit Class Reference	540
6.90.1 Detailed Description	540
6.90.2 Member Function Documentation	540
6.90.2.1 Ranking()	541
6.90.2.2 randomInt()	541
6.90.2.3 deserialize()	541
6.90.2.4 serialize()	541
6.90.2.5 getRecord()	542
6.90.2.6 addEntry()	542

6.90.2.7 getName()	543
6.90.2.8 getEntries()	543
6.91 repository.RankingRepository Class Reference	544
6.91.1 Detailed Description	544
6.91.2 Constructor & Destructor Documentation	545
6.91.2.1 RankingRepository()	545
6.91.3 Member Function Documentation	545
6.91.3.1 save()	545
6.91.3.2 get()	546
6.91.3.3 listRankings()	546
6.92 repository.RankingRepositoryCtrl Class Reference	547
6.92.1 Detailed Description	547
6.92.2 Constructor & Destructor Documentation	548
6.92.2.1 RankingRepositoryCtrl()	548
6.92.3 Member Function Documentation	548
6.92.3.1 save()	548
6.92.3.2 get()	549
6.92.3.3 listRankings()	549
6.92.4 Member Data Documentation	550
6.92.4.1 repository	550
6.93 domain.Ranking.RankingType Enum Reference	550
6.93.1 Detailed Description	550
6.93.2 Member Data Documentation	550
6.93.2.1 MULTIPLE	550
6.93.2.2 UNIQUE	550
6.93.2.3 INCREMENTAL	551
6.94 view.RankingView Class Reference	551
6.94.1 Detailed Description	552
6.94.2 Constructor & Destructor Documentation	552
6.94.2.1 RankingView()	552
6.94.3 Member Function Documentation	552
6.94.3.1 initialize()	553
6.94.3.2 user()	553
6.94.3.3 bots()	553
6.94.3.4 config()	554
6.94.3.5 games()	554
6.94.3.6 play()	554
6.94.3.7 consultRankings()	555
6.94.3.8 consultRecords()	555
6.94.3.9 logOut()	555
6.94.4 Member Data Documentation	556
6.94.4.1 user	556

6.94.4.2 bots	556
6.94.4.3 config	556
6.94.4.4 games	556
6.94.4.5 ranking	556
6.94.4.6 play	557
6.94.4.7 consultRanking	557
6.94.4.8 consultRankingButton	557
6.94.4.9 currentUserName	557
6.94.4.10 consultRecord	557
6.94.4.11 consultRecordButton	558
6.94.4.12 logOut	558
6.95 view.RecordConsultView Class Reference	558
6.95.1 Detailed Description	559
6.95.2 Constructor & Destructor Documentation	559
6.95.2.1 RecordConsultView()	559
6.95.3 Member Function Documentation	560
6.95.3.1 initialize()	560
6.95.3.2 user()	560
6.95.3.3 bots()	561
6.95.3.4 config()	561
6.95.3.5 games()	561
6.95.3.6 play()	562
6.95.3.7 consultRankings()	562
6.95.3.8 consultRecords()	562
6.95.3.9 logOut()	563
6.95.4 Member Data Documentation	563
6.95.4.1 user	563
6.95.4.2 bots	563
6.95.4.3 config	564
6.95.4.4 games	564
6.95.4.5 ranking	564
6.95.4.6 play	564
6.95.4.7 consultRanking	564
6.95.4.8 consultRankingButton	565
6.95.4.9 consultRecord	565
6.95.4.10 consultRecordButton	565
6.95.4.11 currentUserName	565
6.95.4.12 table	565
6.95.4.13 rankingColumn	566
6.95.4.14 valueColumn	566
6.96 repository.Repository Class Reference	566
6.96.1 Detailed Description	567

6.96.2 Constructor & Destructor Documentation	567
6.96.2.1 Repository()	567
6.96.3 Member Function Documentation	568
6.96.3.1 list()	568
6.96.3.2 get()	569
6.96.3.3 createOrUpdate()	569
6.96.3.4 remove()	570
6.96.4 Member Data Documentation	571
6.96.4.1 databasesPath	571
6.96.4.2 fixturesPath	571
6.96.4.3 path	571
6.96.4.4 type	571
6.97 repository.Repository.RepositoryType Enum Reference	572
6.97.1 Detailed Description	572
6.97.2 Member Data Documentation	572
6.97.2.1 CONFIGURATION	572
6.97.2.2 GAME	572
6.97.2.3 PLAYER	572
6.97.2.4 RANKING	573
6.97.2.5 FIXTURE	573
6.98 view.SignUpView Class Reference	573
6.98.1 Detailed Description	574
6.98.2 Constructor & Destructor Documentation	574
6.98.2.1 SignUpView()	574
6.98.3 Member Function Documentation	574
6.98.3.1 signUp()	574
6.98.3.2 login()	575
6.98.4 Member Data Documentation	575
6.98.4.1 login	575
6.98.4.2 signUp	575
6.98.4.3 nusername	575
6.98.4.4 npassword	576
6.98.4.5 rpassword	576
6.98.4.6 signUpResult	576
6.98.4.7 signUp2	576
6.98.4.8 signUpButton	576
6.99 domain.HardDifficulty.TreeNode Class Reference	577
6.99.1 Detailed Description	578
6.99.2 Constructor & Destructor Documentation	578
6.99.2.1 TreeNode()	578
6.99.3 Member Function Documentation	579
6.99.3.1 isLeaf()	579

6.99.3.2	<a href="#">getSelectedPosition()</a>	580
6.99.3.3	<a href="#">getWinRatio()</a>	580
6.99.3.4	<a href="#">getChildren()</a>	581
6.99.3.5	<a href="#">play()</a>	581
6.99.3.6	<a href="#">select()</a>	582
6.99.3.7	<a href="#">expand()</a>	582
6.99.3.8	<a href="#">rollOut()</a>	583
6.99.3.9	<a href="#">updateStats()</a>	584
6.99.4	<a href="#">Member Data Documentation</a>	584
6.99.4.1	<a href="#">children</a>	584
6.99.4.2	<a href="#">nVisits</a>	584
6.99.4.3	<a href="#">totValue</a>	585
6.99.4.4	<a href="#">selectedPosition</a>	585
6.99.4.5	<a href="#">pieceType</a>	585
6.99.4.6	<a href="#">rootType</a>	585
6.99.4.7	<a href="#">currentBoard</a>	585
6.99.4.8	<a href="#">canEatHorizontally</a>	586
6.99.4.9	<a href="#">canEatVertically</a>	586
6.99.4.10	<a href="#">canEatDiagonally</a>	586
6.100	<a href="#">cmd.driver.user Class Reference</a>	586
6.100.1	<a href="#">Detailed Description</a>	586
6.100.2	<a href="#">Member Function Documentation</a>	587
6.100.2.1	<a href="#">main()</a>	587
6.101	<a href="#">domain.User Class Reference</a>	587
6.101.1	<a href="#">Detailed Description</a>	588
6.101.2	<a href="#">Constructor &amp; Destructor Documentation</a>	588
6.101.2.1	<a href="#">User() [1/2]</a>	588
6.101.2.2	<a href="#">User() [2/2]</a>	589
6.101.3	<a href="#">Member Function Documentation</a>	589
6.101.3.1	<a href="#">serialize()</a>	589
6.101.3.2	<a href="#">getPassword()</a>	590
6.101.3.3	<a href="#">setPassword()</a>	590
6.101.4	<a href="#">Member Data Documentation</a>	590
6.101.4.1	<a href="#">password</a>	591
6.102	<a href="#">view.UserDeleteView Class Reference</a>	591
6.102.1	<a href="#">Detailed Description</a>	592
6.102.2	<a href="#">Constructor &amp; Destructor Documentation</a>	592
6.102.2.1	<a href="#">UserDeleteView()</a>	593
6.102.3	<a href="#">Member Function Documentation</a>	593
6.102.3.1	<a href="#">initialize()</a>	593
6.102.3.2	<a href="#">bots()</a>	593
6.102.3.3	<a href="#">config()</a>	594

6.102.3.4 games()	594
6.102.3.5 ranking()	594
6.102.3.6 play()	595
6.102.3.7 modifyUser()	595
6.102.3.8 deleteUser()	595
6.102.3.9 deleteUserConfirm()	596
6.102.3.10 logOut()	596
6.102.4 Member Data Documentation	597
6.102.4.1 user	597
6.102.4.2 bots	597
6.102.4.3 config	597
6.102.4.4 games	597
6.102.4.5 ranking	597
6.102.4.6 play	598
6.102.4.7 modifyUser	598
6.102.4.8 modifyUserButton	598
6.102.4.9 deleteUser	598
6.102.4.10 deleteUserButton	598
6.102.4.11 password	599
6.102.4.12 deleteUserResult	599
6.102.4.13 deleteUserConfirm	599
6.102.4.14 deleteUserConfirmButton	599
6.102.4.15 currentUserName	599
6.102.4.16 logOut	600
6.103 test.driver.UserDriver Class Reference	600
6.103.1 Detailed Description	600
6.103.2 Constructor & Destructor Documentation	600
6.103.2.1 UserDriver()	601
6.103.3 Member Function Documentation	601
6.103.3.1 start()	601
6.103.3.2 mainMenu()	601
6.103.3.3 createUser()	602
6.103.3.4 serialize()	602
6.103.3.5 deserialize()	602
6.103.3.6 getName()	603
6.103.3.7 getPassword()	603
6.103.3.8 getIsDeleted()	603
6.103.3.9 getType()	603
6.103.3.10 getID()	604
6.103.3.11 setName()	604
6.103.3.12 setPassword()	604
6.103.3.13 setIsDeleted()	605

6.103.4 Member Data Documentation	605
6.103.4.1 currentUser	605
6.104 view.UserModifyView Class Reference	605
6.104.1 Detailed Description	606
6.104.2 Constructor & Destructor Documentation	606
6.104.2.1 UserModifyView()	607
6.104.3 Member Function Documentation	607
6.104.3.1 initialize()	607
6.104.3.2 bots()	607
6.104.3.3 config()	608
6.104.3.4 games()	608
6.104.3.5 ranking()	608
6.104.3.6 play()	609
6.104.3.7 modifyUser()	609
6.104.3.8 deleteUser()	609
6.104.3.9 modifyUserConfirm()	610
6.104.3.10 logOut()	610
6.104.4 Member Data Documentation	611
6.104.4.1 user	611
6.104.4.2 bots	611
6.104.4.3 config	611
6.104.4.4 games	612
6.104.4.5 ranking	612
6.104.4.6 play	612
6.104.4.7 modifyUser	612
6.104.4.8 modifyUserButton	612
6.104.4.9 deleteUser	613
6.104.4.10 deleteUserButton	613
6.104.4.11 nusername	613
6.104.4.12 npassword	613
6.104.4.13 rpassword	613
6.104.4.14 modifyUserResult	614
6.104.4.15 modifyUserConfirm	614
6.104.4.16 modifyUserConfirmButton	614
6.104.4.17 currentUserName	614
6.104.4.18 logOut	614
6.105 view.UserView Class Reference	615
6.105.1 Detailed Description	616
6.105.2 Constructor & Destructor Documentation	616
6.105.2.1 UserView()	616
6.105.3 Member Function Documentation	616
6.105.3.1 initialize()	616

6.105.3.2 bots()	617
6.105.3.3 config()	617
6.105.3.4 games()	617
6.105.3.5 ranking()	618
6.105.3.6 play()	618
6.105.3.7 modifyUser()	618
6.105.3.8 deleteUser()	619
6.105.3.9 logOut()	619
6.105.4 Member Data Documentation	619
6.105.4.1 user	620
6.105.4.2 bots	620
6.105.4.3 config	620
6.105.4.4 games	620
6.105.4.5 ranking	620
6.105.4.6 play	621
6.105.4.7 currentUserName	621
6.105.4.8 modifyUser	621
6.105.4.9 modifyUserButton	621
6.105.4.10 deleteUser	621
6.105.4.11 deleteUserButton	622
6.105.4.12 logOut	622
6.106 view.ViewCtrl Class Reference	622
6.106.1 Detailed Description	623
6.106.2 Constructor & Destructor Documentation	623
6.106.2.1 ViewCtrl()	623
6.106.3 Member Function Documentation	623
6.106.3.1 start()	623
6.106.3.2 changeScene()	624
6.106.3.3 newWindow()	624
6.106.3.4 main()	625
6.106.4 Member Data Documentation	625
6.106.4.1 domainCtrl	625
6.106.4.2 stage	625
<b>7 File Documentation</b>	<b>627</b>
7.1 board.java File Reference	627
7.1.1 Detailed Description	627
7.2 Board.java File Reference	627
7.2.1 Detailed Description	628
7.3 BoardCtrl.java File Reference	628
7.3.1 Detailed Description	628
7.4 BoardDriver.java File Reference	628



7.5 bot.java File Reference . . . . .	629
7.5.1 Detailed Description . . . . .	629
7.6 Bot.java File Reference . . . . .	629
7.6.1 Detailed Description . . . . .	629
7.7 BotDriver.java File Reference . . . . .	629
7.7.1 Detailed Description . . . . .	630
7.8 BotsConsultView.java File Reference . . . . .	630
7.8.1 Detailed Description . . . . .	630
7.9 BotsCreateView.java File Reference . . . . .	630
7.9.1 Detailed Description . . . . .	631
7.10 BotsModifyView.java File Reference . . . . .	631
7.10.1 Detailed Description . . . . .	631
7.11 BotsView.java File Reference . . . . .	631
7.11.1 Detailed Description . . . . .	632
7.12 ConfigConsultView.java File Reference . . . . .	632
7.12.1 Detailed Description . . . . .	632
7.13 ConfigCreateView.java File Reference . . . . .	632
7.13.1 Detailed Description . . . . .	633
7.14 ConfigModifyView.java File Reference . . . . .	633
7.14.1 Detailed Description . . . . .	633
7.15 Configuration.java File Reference . . . . .	633
7.15.1 Detailed Description . . . . .	634
7.16 configuration.java File Reference . . . . .	634
7.16.1 Detailed Description . . . . .	634
7.17 ConfigurationCtrl.java File Reference . . . . .	634
7.17.1 Detailed Description . . . . .	635
7.18 ConfigurationDriver.java File Reference . . . . .	635
7.18.1 Detailed Description . . . . .	635
7.19 ConfigurationRepository.java File Reference . . . . .	635
7.19.1 Detailed Description . . . . .	636
7.20 ConfigurationRepositoryCtrl.java File Reference . . . . .	636
7.20.1 Detailed Description . . . . .	636
7.21 ConfigView.java File Reference . . . . .	636
7.21.1 Detailed Description . . . . .	637
7.22 ConsultInitialBoardView.java File Reference . . . . .	637
7.22.1 Detailed Description . . . . .	637
7.23 Difficulty.java File Reference . . . . .	637
7.23.1 Detailed Description . . . . .	638
7.24 DifficultyCtrl.java File Reference . . . . .	638
7.24.1 Detailed Description . . . . .	638
7.25 DomainCtrl.java File Reference . . . . .	638
7.25.1 Detailed Description . . . . .	639

7.26 Driver.java File Reference . . . . .	639
7.26.1 Detailed Description . . . . .	639
7.27 EasyDifficulty.java File Reference . . . . .	639
7.27.1 Detailed Description . . . . .	640
7.28 easyDifficulty.java File Reference . . . . .	640
7.28.1 Detailed Description . . . . .	640
7.29 EasyDifficultyDriver.java File Reference . . . . .	640
7.29.1 Detailed Description . . . . .	641
7.30 Entry.java File Reference . . . . .	641
7.30.1 Detailed Description . . . . .	641
7.31 entry.java File Reference . . . . .	641
7.31.1 Detailed Description . . . . .	642
7.32 EntryJUnit.java File Reference . . . . .	642
7.32.1 Detailed Description . . . . .	642
7.33 Exceptions.java File Reference . . . . .	642
7.33.1 Detailed Description . . . . .	643
7.34 FixtureRepository.java File Reference . . . . .	643
7.34.1 Detailed Description . . . . .	644
7.35 Game.java File Reference . . . . .	644
7.35.1 Detailed Description . . . . .	644
7.36 game.java File Reference . . . . .	644
7.36.1 Detailed Description . . . . .	645
7.37 GameBoardView.java File Reference . . . . .	645
7.37.1 Detailed Description . . . . .	645
7.38 GameCtrl.java File Reference . . . . .	645
7.38.1 Detailed Description . . . . .	646
7.39 GameDriver.java File Reference . . . . .	646
7.39.1 Detailed Description . . . . .	646
7.40 GameRepository.java File Reference . . . . .	646
7.40.1 Detailed Description . . . . .	647
7.41 GameRepositoryCtrl.java File Reference . . . . .	647
7.41.1 Detailed Description . . . . .	647
7.42 GamesCreateView.java File Reference . . . . .	647
7.42.1 Detailed Description . . . . .	648
7.43 GamesView.java File Reference . . . . .	648
7.43.1 Detailed Description . . . . .	648
7.44 hardDifficulty.java File Reference . . . . .	648
7.44.1 Detailed Description . . . . .	649
7.45 HardDifficulty.java File Reference . . . . .	649
7.45.1 Detailed Description . . . . .	649
7.46 HardDifficultyDriver.java File Reference . . . . .	649
7.46.1 Detailed Description . . . . .	650

7.47 InitialBoardView.java File Reference . . . . .	650
7.47.1 Detailed Description . . . . .	650
7.48 LogInView.java File Reference . . . . .	650
7.48.1 Detailed Description . . . . .	651
7.49 MediumDifficulty.java File Reference . . . . .	651
7.49.1 Detailed Description . . . . .	651
7.50 mediumDifficulty.java File Reference . . . . .	651
7.50.1 Detailed Description . . . . .	652
7.51 MediumDifficultyDriver.java File Reference . . . . .	652
7.51.1 Detailed Description . . . . .	652
7.52 ModifyInitialBoardView.java File Reference . . . . .	652
7.52.1 Detailed Description . . . . .	653
7.53 othello.java File Reference . . . . .	653
7.53.1 Detailed Description . . . . .	653
7.54 Pair.java File Reference . . . . .	653
7.54.1 Detailed Description . . . . .	654
7.55 pair.java File Reference . . . . .	654
7.55.1 Detailed Description . . . . .	654
7.56 PairDriver.java File Reference . . . . .	654
7.56.1 Detailed Description . . . . .	655
7.57 Player.java File Reference . . . . .	655
7.57.1 Detailed Description . . . . .	655
7.58 PlayerCtrl.java File Reference . . . . .	655
7.58.1 Detailed Description . . . . .	655
7.59 PlayerRepository.java File Reference . . . . .	656
7.59.1 Detailed Description . . . . .	656
7.60 PlayerRepositoryCtrl.java File Reference . . . . .	656
7.60.1 Detailed Description . . . . .	656
7.61 PlayView.java File Reference . . . . .	657
7.61.1 Detailed Description . . . . .	657
7.62 ranking.java File Reference . . . . .	657
7.62.1 Detailed Description . . . . .	657
7.63 Ranking.java File Reference . . . . .	657
7.63.1 Detailed Description . . . . .	658
7.64 RankingConsultView.java File Reference . . . . .	658
7.64.1 Detailed Description . . . . .	658
7.65 RankingCtrl.java File Reference . . . . .	658
7.65.1 Detailed Description . . . . .	659
7.66 RankingJUnit.java File Reference . . . . .	659
7.66.1 Detailed Description . . . . .	659
7.67 RankingRepository.java File Reference . . . . .	659
7.67.1 Detailed Description . . . . .	660

---

7.68 RankingRepositoryCtrl.java File Reference . . . . .	660
7.68.1 Detailed Description . . . . .	660
7.69 RankingView.java File Reference . . . . .	660
7.69.1 Detailed Description . . . . .	661
7.70 RecordConsultView.java File Reference . . . . .	661
7.70.1 Detailed Description . . . . .	661
7.71 Repository.java File Reference . . . . .	661
7.71.1 Detailed Description . . . . .	662
7.72 SignUpView.java File Reference . . . . .	662
7.72.1 Detailed Description . . . . .	662
7.73 User.java File Reference . . . . .	662
7.73.1 Detailed Description . . . . .	662
7.74 user.java File Reference . . . . .	663
7.74.1 Detailed Description . . . . .	663
7.75 UserDeleteView.java File Reference . . . . .	663
7.75.1 Detailed Description . . . . .	663
7.76 UserDriver.java File Reference . . . . .	663
7.76.1 Detailed Description . . . . .	664
7.77 UserModifyView.java File Reference . . . . .	664
7.77.1 Detailed Description . . . . .	664
7.78 UserView.java File Reference . . . . .	664
7.78.1 Detailed Description . . . . .	665
7.79 ViewCtrl.java File Reference . . . . .	665
7.79.1 Detailed Description . . . . .	665
<b>Index</b>	<b>667</b>

# 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:

cmd.driver.board . . . . .	24
domain.Board . . . . .	24
domain.BoardCtrl . . . . .	44
test.driver.BoardDriver . . . . .	49
cmd.driver.bot . . . . .	65
view.BotsConsultView . . . . .	72
view.BotsCreateView . . . . .	83
view.BotsModifyView . . . . .	94
view.BotsView . . . . .	107
view.ConfigConsultView . . . . .	116
view.ConfigCreateView . . . . .	128
view.ConfigModifyView . . . . .	139
domain.Configuration . . . . .	153
cmd.driver.configuration . . . . .	161
domain.ConfigurationCtrl . . . . .	162
repository.ConfigurationRepositoryCtrl . . . . .	180
view.ConfigView . . . . .	185
view.ConsultInitialBoardView . . . . .	192
domain.Difficulty . . . . .	211
domain.EasyDifficulty . . . . .	263
domain.HardDifficulty . . . . .	389
domain.MediumDifficulty . . . . .	433
domain.DifficultyCtrl . . . . .	219
domain.DomainCtrl . . . . .	222
test.driver.Driver . . . . .	258
test.driver.BotDriver . . . . .	66
test.driver.ConfigurationDriver . . . . .	169
test.driver.EasyDifficultyDriver . . . . .	268
test.driver.GameDriver . . . . .	348
test.driver.HardDifficultyDriver . . . . .	392
test.driver.MediumDifficultyDriver . . . . .	439
test.driver.PairDriver . . . . .	477
test.driver.UserDriver . . . . .	600
cmd.driver.easyDifficulty . . . . .	262
cmd.unitary.entry . . . . .	276

domain.Entry	277
test.unitary.EntryJUnit	282
Exception	
domain.Exceptions.BadConfirmationException	23
domain.Exceptions.BotUsedException	115
domain.Exceptions.ConfigurationUsedException	184
domain.Exceptions.ExistingConfigurationException	285
domain.Exceptions.ExistingPlayerException	286
domain.Exceptions.FinishedGameException	287
domain.Exceptions.IncorrectCredentialsException	400
domain.Exceptions.InexistingConfigurationException	401
domain.Exceptions.InexistingPlayerException	402
domain.Exceptions.InvalidBoardException	424
domain.Exceptions.InvalidConfigurationException	424
domain.Exceptions.InvalidDifficultyException	425
domain.Exceptions.InvalidNameException	426
domain.Exceptions.InvalidPasswordException	426
domain.Exceptions.InvalidPlayersException	427
domain.Exceptions.InvalidPositionException	428
domain.Exceptions.InvalidRulesException	428
domain.Exceptions.NotCreatorException	468
domain.Exceptions.NotPlayerException	469
domain.Exceptions.NotPlayerPieceException	470
domain.Exceptions.NotPlayerTurnException	470
domain.Exceptions.NotStartedGameException	471
domain.Exceptions	284
domain.Game	291
cmd.driver.game	307
view.GameBoardView	308
domain.GameCtrl	339
repository.GameRepositoryCtrl	364
view.GamesCreateView	369
domain.Game.GameState	381
view.GamesView	382
cmd.driver.hardDifficulty	392
view.InitialBoardView	402
view.LogInView	429
cmd.driver.mediumDifficulty	432
view.ModifyInitialBoardView	447
cmd.othello	472
cmd.driver.pair	473
util.Pair< F, S >	473
util.Pair< Integer, Integer >	473
util.Pair< JSONObject, JSONObject >	473
util.Pair< String, Integer >	473
util.Pair< String, String >	473
domain.Board.PieceType	482
domain.Player	483
domain.Bot	61
domain.User	587
domain.PlayerCtrl	487
repository.PlayerRepositoryCtrl	503
view.PlayView	508
cmd.unitary.ranking	518
domain.Ranking	519
view.RankingConsultView	526
domain.RankingCtrl	535
test.unitary.RankingJUnit	540



repository.RankingRepositoryCtrl . . . . .	547
domain.Ranking.RankingType . . . . .	550
view.RankingView . . . . .	551
view.RecordConsultView . . . . .	558
repository.Repository . . . . .	566
repository.ConfigurationRepository . . . . .	175
repository.FixtureRepository . . . . .	287
repository.GameRepository . . . . .	359
repository.PlayerRepository . . . . .	498
repository.RankingRepository . . . . .	544
repository.Repository.RepositoryType . . . . .	572
view.SignUpView . . . . .	573
domain.HardDifficulty.TreeNode . . . . .	577
cmd.driver.user . . . . .	586
view.UserDeleteView . . . . .	591
view.UserModifyView . . . . .	605
view.UserView . . . . .	615
Application	
view.ViewCtrl . . . . .	622



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">domain.Exceptions.BadConfirmationException</a>	
The entered confirmation password doesn't match the user's password. By Alex Rodriguez . . .	23
<a href="#">cmd.driver.board</a>	
Board driver entryptoint. By Alex Rodriguez . . . . .	24
<a href="#">domain.Board</a> . . . . .	24
<a href="#">domain.BoardCtrl</a>	
This class represents the controller of the <a href="#">Board</a> class, which is the classs that will be used to communicate with the other controllers . . . . .	44
<a href="#">test.driver.BoardDriver</a> . . . . .	49
<a href="#">domain.Bot</a>	
Represents a bot in our system . . . . .	61
<a href="#">cmd.driver.bot</a>	
Bot driver entryptoint. By Alex Rodriguez . . . . .	65
<a href="#">test.driver.BotDriver</a> . . . . .	66
<a href="#">view.BotsConsultView</a> . . . . .	72
<a href="#">view.BotsCreateView</a> . . . . .	83
<a href="#">view.BotsModifyView</a> . . . . .	94
<a href="#">view.BotsView</a> . . . . .	107
<a href="#">domain.Exceptions.BotUsedException</a>	
A bot cannot be modified or deleted if it is already part of a game. By Alex Rodriguez . . . . .	115
<a href="#">view.ConfigConsultView</a> . . . . .	116
<a href="#">view.ConfigCreateView</a> . . . . .	128
<a href="#">view.ConfigModifyView</a> . . . . .	139
<a href="#">domain.Configuration</a>	
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 . . . . .	153
<a href="#">cmd.driver.configuration</a>	
Configuration driver entryptoint. By Alex Rodriguez . . . . .	161
<a href="#">domain.ConfigurationCtrl</a>	
<a href="#">Configuration</a> 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 . . . . .	162
<a href="#">test.driver.ConfigurationDriver</a>	
Implements the different options for the Configuration driver application. By Alex Rodriguez . . .	169

<a href="#">repository.ConfigurationRepository</a>	Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez . . . . .	175
<a href="#">repository.ConfigurationRepositoryCtrl</a>	Implements various CRUD operations to work with the Configuration repository. By Alex Rodriguez . . . . .	180
<a href="#">domain.Exceptions.ConfigurationUsedException</a>	A configuration cannot be modified or deleted if it is already used in a game. By Alex Rodriguez . . . . .	184
<a href="#">view.ConfigView</a>	. . . . .	185
<a href="#">view.ConsultInitialBoardView</a>	. . . . .	192
<a href="#">domain.Difficulty</a>	Implements the abstract class and methods of all the difficulty implementations. By Arnau Pu-jantell . . . . .	211
<a href="#">domain.DifficultyCtrl</a>	<a href="#">Difficulty</a> domain sub-controller. Is in charge of <a href="#">EasyDifficulty</a> , <a href="#">MediumDifficulty</a> and <a href="#">HardDifficulty</a> . 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: <a href="#">EasyDifficulty</a> (Minimax). 4 to 6: <a href="#">MediumDifficulty</a> (Minimax alpha beta pruning). 7 to 10: <a href="#">HardDifficulty</a> (Montecarlo) . . . . .	219
<a href="#">domain.DomainCtrl</a>	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 . . . . .	222
<a href="#">test.driver.Driver</a>	Implements various utilities to create a driver application. By Alex Rodriguez . . . . .	258
<a href="#">cmd.driver.easyDifficulty</a>	<a href="#">EasyDifficulty</a> driver entrypoint. By Alex Rodriguez . . . . .	262
<a href="#">domain.EasyDifficulty</a>	Implements the Minimax algorithm to get the next best possible position for a given player. By Manuel Navid . . . . .	263
<a href="#">test.driver.EasyDifficultyDriver</a>	Implements the different options for the <a href="#">EasyDifficulty</a> driver application. By Manuel Navid . . . . .	268
<a href="#">cmd.unitary.entry</a>	JUnit Entry tests entrypoint. By Alex Rodriguez . . . . .	276
<a href="#">domain.Entry</a>	Represents an entry in a <a href="#">Ranking</a> table . . . . .	277
<a href="#">test.unitary.EntryJUnit</a>	Allows JUnit testing of class <a href="#">Entry</a> . . . . .	282
<a href="#">domain.Exceptions</a>	Holds all the different custom <a href="#">Exceptions</a> used in the whole project. By Alex Rodriguez . . . . .	284
<a href="#">domain.Exceptions.ExistingConfigurationException</a>	There is already a configuration with the same name and creator ID in the system. By Alex Rodriguez . . . . .	285
<a href="#">domain.Exceptions.ExistingPlayerException</a>	There is already a player with the same name in the system. By Alex Rodriguez . . . . .	286
<a href="#">domain.Exceptions.FinishedGameException</a>	The game is already finished. By Alex Rodriguez . . . . .	287
<a href="#">repository.FixtureRepository</a>	Implements various CRUD operations to work with the Fixture repository. By Alex Rodriguez . . . . .	287
<a href="#">domain.Game</a>	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 . . . . .	291
<a href="#">cmd.driver.game</a>	Game driver entrypoint. By Alex Rodriguez . . . . .	307

<a href="#">view.GameBoardView</a>	308
<a href="#">domain.GameCtrl</a>	
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	339
<a href="#">test.driver.GameDriver</a>	
Implements the different options for the Game driver application. By Alex Rodriguez	348
<a href="#">repository.GameRepository</a>	
Implements various CRUD operations to work with the Game repository. By Alex Rodriguez	359
<a href="#">repository.GameRepositoryCtrl</a>	
Implements various CRUD operations to work with the Game repository. By Alex Rodriguez	364
<a href="#">view.GamesCreateView</a>	369
<a href="#">domain.Game.GameState</a>	
State of a <a href="#">Game</a> . Whether it has not started, it is currently being played or it has already finished	381
<a href="#">view.GamesView</a>	382
<a href="#">domain.HardDifficulty</a>	
Implements the Monte Carlo Tree Search algorithm to get the next best possible position for a given player. By Roger Mollon	389
<a href="#">cmd.driver.hardDifficulty</a>	
HardDifficulty driver entrypoint. By Alex Rodriguez	392
<a href="#">test.driver.HardDifficultyDriver</a>	
Implements the different options for the HardDifficulty driver application. By Roger Mollon	392
<a href="#">domain.Exceptions.IncorrectCredentialsException</a>	
Wrong password or name. By Alex Rodriguez	400
<a href="#">domain.Exceptions.InexistingConfigurationException</a>	
There isn't any configuration with the entered name. By Alex Rodriguez	401
<a href="#">domain.Exceptions.InexistingPlayerException</a>	
There isn't any player with the entered name. By Alex Rodriguez	402
<a href="#">view.InitialBoardView</a>	402
<a href="#">domain.Exceptions.InvalidBoardException</a>	
The current board is in an illegal state. By Alex Rodriguez	424
<a href="#">domain.Exceptions.InvalidConfigurationException</a>	
The entered configuration is null, empty or blank. By Alex Rodriguez	424
<a href="#">domain.Exceptions.InvalidDifficultyException</a>	
The entered difficulty is null, empty, blank, less than 0 or greater than 10. By Alex Rodriguez	425
<a href="#">domain.Exceptions.InvalidNameException</a>	
The entered name is null, empty or blank. By Alex Rodriguez	426
<a href="#">domain.Exceptions.InvalidPasswordException</a>	
The entered password is null, empty or blank. By Alex Rodriguez	426
<a href="#">domain.Exceptions.InvalidPlayersException</a>	
The entered players are the same, null, empty, blank or both bots have the same difficulty. By Alex Rodriguez	427
<a href="#">domain.Exceptions.InvalidPositionException</a>	
The entered position is null, empty, blank or ends up with an invalid board. By Alex Rodriguez	428
<a href="#">domain.Exceptions.InvalidRulesException</a>	
The entered configuration rules are all deactivated. By Alex Rodriguez	428
<a href="#">view.LoginView</a>	429
<a href="#">cmd.driver.mediumDifficulty</a>	
MediumDifficulty driver entrypoint. By Alex Rodriguez	432
<a href="#">domain.MediumDifficulty</a>	
Implements the Minimax algorithm with alpha-beta pruning to get the next best possible position for a given player. By Alex Rodriguez	433
<a href="#">test.driver.MediumDifficultyDriver</a>	
Implements the different options for the MediumDifficulty driver application. By Alex Rodriguez	439
<a href="#">view.ModifyInitialBoardView</a>	447
<a href="#">domain.Exceptions.NotCreatorException</a>	
The user that tries to perform an action on a object is not the creator of it. By Alex Rodriguez	468

<a href="#">domain.Exceptions.NotPlayerException</a>	
The player that wants to perform an action is not part of the game. By Alex Rodriguez . . . . .	469
<a href="#">domain.Exceptions.NotPlayerPieceException</a>	
The player that wants to perform an action tries to use an opponent piece. By Alex Rodriguez . . . . .	470
<a href="#">domain.Exceptions.NotPlayerTurnException</a>	
It is not the turn of the player that wants to perform an action. By Alex Rodriguez . . . . .	470
<a href="#">domain.Exceptions.NotStartedGameException</a>	
The game has not yet started. By Alex Rodriguez . . . . .	471
<a href="#">cmd.othello</a>	
Othello application entrypoint. By Alex Rodriguez . . . . .	472
<a href="#">cmd.driver.pair</a>	
Pair driver entrypoint. By Alex Rodriguez . . . . .	473
<a href="#">util.Pair&lt; F, S &gt;</a>	
Implements a data structure containing two generic types. By Alex Rodriguez . . . . .	473
<a href="#">test.driver.PairDriver</a>	
Implements the different options for the Pair driver application. By Alex Rodriguez . . . . .	477
<a href="#">domain.Board.PieceType</a>	
The status of a cell of the <a href="#">Board</a> . An Othello <a href="#">Board</a> is composed of 64 cells with their own unique position and three possible states: . . . . .	482
<a href="#">domain.Player</a>	
Represents a player in our system . . . . .	483
<a href="#">domain.PlayerCtrl</a>	
<a href="#">Player</a> class controller . . . . .	487
<a href="#">repository.PlayerRepository</a>	
Implements various CRUD operations to work with the Player repository. By Alex Rodriguez . . . . .	498
<a href="#">repository.PlayerRepositoryCtrl</a>	
Implements various CRUD operations to work with the Player repository. By Alex Rodriguez . . . . .	503
<a href="#">view.PlayView</a> . . . . .	508
<a href="#">cmd.unitary.ranking</a>	
JUnit Ranking tests entrypoint. By Alex Rodriguez . . . . .	518
<a href="#">domain.Ranking</a>	
Representation of a ranking table . . . . .	519
<a href="#">view.RankingConsultView</a> . . . . .	526
<a href="#">domain.RankingCtrl</a>	
<a href="#">Ranking</a> domain sub-controller. It communicates with the main domain controller and the ranking repository controller. By Alex Rodriguez . . . . .	535
<a href="#">test.unitary.RankingJUnit</a>	
Allows JUnit testing of class Ranking . . . . .	540
<a href="#">repository.RankingRepository</a>	
Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez . . . . .	544
<a href="#">repository.RankingRepositoryCtrl</a>	
Implements various CRUD operations to work with the Ranking repository. By Alex Rodriguez . . . . .	547
<a href="#">domain.Ranking.RankingType</a> . . . . .	550
<a href="#">view.RankingView</a> . . . . .	551
<a href="#">view.RecordConsultView</a> . . . . .	558
<a href="#">repository.Repository</a>	
Implements various CRUD operations to work with the local file system JSON databases and TXT fixtures. By Alex Rodriguez . . . . .	566
<a href="#">repository.Repository.RepositoryType</a>	
Different types for the accessed repository . . . . .	572
<a href="#">view.SignUpView</a> . . . . .	573
<a href="#">domain.HardDifficulty.TreeNode</a> . . . . .	577
<a href="#">cmd.driver.user</a>	
User driver entrypoint. By Alex Rodriguez . . . . .	586
<a href="#">domain.User</a>	
Represents a human user in our system . . . . .	587
<a href="#">view.UserDeleteView</a> . . . . .	591
<a href="#">test.driver.UserDriver</a> . . . . .	600

<a href="#">view.UserModifyView</a>	605
<a href="#">view.UserView</a>	615
<a href="#">view.ViewCtrl</a>	622





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">board.java</a>	BoardDriver entrypoint class specification . . . . .	627
<a href="#">Board.java</a>	Board class specification . . . . .	627
<a href="#">BoardCtrl.java</a>	BoardCtrl class specification . . . . .	628
<a href="#">BoardDriver.java</a>	. . . . .	628
<a href="#">bot.java</a>	BotDriver entrypoint class specification . . . . .	629
<a href="#">Bot.java</a>	Bot subclass specification . . . . .	629
<a href="#">BotDriver.java</a>	Bot driver specification Done by Arnau Pujantell . . . . .	629
<a href="#">BotsConsultView.java</a>	Bot consult View controller specification . . . . .	630
<a href="#">BotsCreateView.java</a>	Bot create View controller specification . . . . .	630
<a href="#">BotsModifyView.java</a>	Bot modify View controller specification . . . . .	631
<a href="#">BotsView.java</a>	BotsView controller specification . . . . .	631
<a href="#">ConfigConsultView.java</a>	Configuration Consult View controller specification . . . . .	632
<a href="#">ConfigCreateView.java</a>	Configuration Create View controller specification . . . . .	632
<a href="#">ConfigModifyView.java</a>	Configuration Modify View controller specification . . . . .	633
<a href="#">Configuration.java</a>	Configuration class specification . . . . .	633
<a href="#">configuration.java</a>	ConfigurationDriver entrypoint class specification . . . . .	634
<a href="#">ConfigurationCtrl.java</a>	ConfigurationCtrl class specification . . . . .	634
<a href="#">ConfigurationDriver.java</a>	ConfigurationDriver class specification . . . . .	635

<a href="#">ConfigurationRepository.java</a>	
ConfigurationRepository class specification	635
<a href="#">ConfigurationRepositoryCtrl.java</a>	
ConfigurationRepositoryCtrl class specification	636
<a href="#">ConfigView.java</a>	
ConfigView controller specification	636
<a href="#">ConsultInitialBoardView.java</a>	
ConsultInitialBoardView controller specification	637
<a href="#">Difficulty.java</a>	
Difficulty class specification	637
<a href="#">DifficultyCtrl.java</a>	
DifficultyCtrl class specification	638
<a href="#">DomainCtrl.java</a>	
DomainCtrl class specification	638
<a href="#">Driver.java</a>	
Driver class specification	639
<a href="#">EasyDifficulty.java</a>	
EasyDifficulty class specification	639
<a href="#">easyDifficulty.java</a>	
EasyDifficultyDriver entrypoint class specification	640
<a href="#">EasyDifficultyDriver.java</a>	
EasyDifficulty class specification	640
<a href="#">Entry.java</a>	
Specification of class Entry	641
<a href="#">entry.java</a>	
JUnit Entry tests entrypoint class specification	641
<a href="#">EntryJUnit.java</a>	
Specification of class EntryJUnit	642
<a href="#">Exceptions.java</a>	
Exceptions classes specifications	642
<a href="#">FixtureRepository.java</a>	
FixtureRepository class specification	643
<a href="#">Game.java</a>	
Game class specification	644
<a href="#">game.java</a>	
GameDriver entrypoint class specification	644
<a href="#">GameBoardView.java</a>	
GameBoardView controller specification	645
<a href="#">GameCtrl.java</a>	
GameCtrl class specification	645
<a href="#">GameDriver.java</a>	
GameDriver class specification	646
<a href="#">GameRepository.java</a>	
GameRepository class specification	646
<a href="#">GameRepositoryCtrl.java</a>	
GameRepositoryCtrl class specification	647
<a href="#">GamesCreateView.java</a>	
Game Create View controller specification	647
<a href="#">GamesView.java</a>	
GamesView controller specification	648
<a href="#">hardDifficulty.java</a>	
HardDifficultyDriver entrypoint class specification	648
<a href="#">HardDifficulty.java</a>	
HardDifficulty class specification	649
<a href="#">HardDifficultyDriver.java</a>	
HardDifficultyDriver class specification	649
<a href="#">InitialBoardView.java</a>	
InitialBoardView controller specification	650

<a href="#">LoginView.java</a>	
LoginView controller specification	650
<a href="#">MediumDifficulty.java</a>	
MediumDifficulty class specification	651
<a href="#">mediumDifficulty.java</a>	
MediumDifficultyDriver entrypoint class specification	651
<a href="#">MediumDifficultyDriver.java</a>	
MediumDifficulty class specification	652
<a href="#">ModifyInitialBoardView.java</a>	
ModifyInitialBoardView controller specification	652
<a href="#">othello.java</a>	
Othello entrypoint class specification	653
<a href="#">Pair.java</a>	
Pair class specification	653
<a href="#">pair.java</a>	
PairDriver entrypoint class specification	654
<a href="#">PairDriver.java</a>	
PairDriver class specification	654
<a href="#">Player.java</a>	
Player class specification	655
<a href="#">PlayerCtrl.java</a>	
PlayerCtrl controller specification	655
<a href="#">PlayerRepository.java</a>	
PlayerRepository class specification	656
<a href="#">PlayerRepositoryCtrl.java</a>	
PlayerRepositoryCtrl class specification	656
<a href="#">PlayView.java</a>	
PlayView controller specification	657
<a href="#">ranking.java</a>	
JUnit Ranking tests entrypoint class specification	657
<a href="#">Ranking.java</a>	
Specification of class Ranking	657
<a href="#">RankingConsultView.java</a>	
Ranking Consult View controller specification	658
<a href="#">RankingCtrl.java</a>	
RankingCtrl class specification	658
<a href="#">RankingJUnit.java</a>	
Specification of class RankingJUnit	659
<a href="#">RankingRepository.java</a>	
RankingRepository class specification	659
<a href="#">RankingRepositoryCtrl.java</a>	
RankingRepositoryCtrl class specification	660
<a href="#">RankingView.java</a>	
RankingView controller specification	660
<a href="#">RecordConsultView.java</a>	
Record Consult View controller specification	661
<a href="#">Repository.java</a>	
Repository class specification	661
<a href="#">SignUpView.java</a>	
SignUpView controller specification	662
<a href="#">User.java</a>	
User subclass specification	662
<a href="#">user.java</a>	
UserDriver entrypoint class specification	663
<a href="#">UserDeleteView.java</a>	
User delete View controller specification	663
<a href="#">UserDriver.java</a>	
User driver specification Done by Arnau Pujantell	663

<a href="#">UserModifyView.java</a>	
User modify View controller specification . . . . .	664
<a href="#">UserView.java</a>	
UserView controller specification . . . . .	664
<a href="#">ViewCtrl.java</a>	
ViewCtrl class specification . . . . .	665

## Chapter 5

# Namespace Documentation

### 5.1 Package cmd

#### Packages

- package [driver](#)
- package [unitary](#)

#### Classes

- class [othello](#)  
*Othello application endpoint. By Alex Rodriguez.*

### 5.2 Package cmd.driver

#### Classes

- class [board](#)  
*Board driver endpoint. By Alex Rodriguez.*
- class [bot](#)  
*Bot driver endpoint. By Alex Rodriguez.*
- class [configuration](#)  
*Configuration driver endpoint. By Alex Rodriguez.*
- class [easyDifficulty](#)  
*EasyDifficulty driver endpoint. By Alex Rodriguez.*
- class [game](#)  
*Game driver endpoint. By Alex Rodriguez.*
- class [hardDifficulty](#)  
*HardDifficulty driver endpoint. By Alex Rodriguez.*
- class [mediumDifficulty](#)  
*MediumDifficulty driver endpoint. By Alex Rodriguez.*
- class [pair](#)  
*Pair driver endpoint. By Alex Rodriguez.*
- class [user](#)  
*User driver endpoint. By Alex Rodriguez.*

## 5.3 Package cmd.unitary

### Classes

- class [entry](#)  
*JUnit Entry tests endpoint. By Alex Rodriguez.*
- class [ranking](#)  
*JUnit Ranking tests endpoint. By Alex Rodriguez.*

## 5.4 Package domain

### Classes

- class [Board](#)
- class [BoardCtrl](#)  
*This class represents the controller of the [Board](#) class, which is the class 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 endpoint. 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 endpoint. 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.

```
22                                     {  
23     new BoardDriver().start();  
24 }
```

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 canEatHorizontally, 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 numRows)  
*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 myPieceType)  
*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 instantiated 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.

```
52     {
53         this.board = new PieceType[8][8];
54
55         //Initial Pieces
56         this.board[3][3] = PieceType.PLAYER1;
57         this.board[4][4] = PieceType.PLAYER1;
58         this.board[4][3] = PieceType.PLAYER2;
59         this.board[3][4] = PieceType.PLAYER2;
60
61         this.piecesPlayer1 = 2;
62         this.piecesPlayer2 = 2;
63     }
```

### 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

<i>jsonBoard</i>	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.

```
73     {
74         this.board = new PieceType[8][8];
75         this.piecesPlayer1 = 0;
76         this.piecesPlayer2 = 0;
77
78         String row0 = jsonBoard.getString("row0");
79         String row1 = jsonBoard.getString("row1");
80         String row2 = jsonBoard.getString("row2");
81         String row3 = jsonBoard.getString("row3");
82         String row4 = jsonBoard.getString("row4");
83         String row5 = jsonBoard.getString("row5");
84         String row6 = jsonBoard.getString("row6");
85         String row7 = jsonBoard.getString("row7");
86
87         this.transcribeToPieceType(row0, 0);
88         this.transcribeToPieceType(row1, 1);
89         this.transcribeToPieceType(row2, 2);
90         this.transcribeToPieceType(row3, 3);
91         this.transcribeToPieceType(row4, 4);
92         this.transcribeToPieceType(row5, 5);
93         this.transcribeToPieceType(row6, 6);
94         this.transcribeToPieceType(row7, 7);
95     }
```

### 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 instantiated 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**

<i>board</i>	An 8x8 <a href="#">PieceType</a> matrix that represents a state of an Othello board.
--------------	--

Definition at line 104 of file Board.java.

```

105     {
106         this.board = new PieceType[8][8];
107         this.piecesPlayer1 = 0;
108         this.piecesPlayer2 = 0;
109
110         for(int i = 0; i < 8; ++i)
111         {
112             for(int j = 0; j < 8; j++)
113             {
114                 this.board[i][j] = board[i][j];
115                 if(this.board[i][j] == PieceType.PLAYER1) this.piecesPlayer1 += 1;
116                 if(this.board[i][j] == PieceType.PLAYER2) this.piecesPlayer2 += 1;
117             }
118         }
119     }

```

**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.

```
129     {
130         ArrayList<String> boardCodified = this.transcribeToCharacters();
131         JSONObject jsonBoard = new JSONObject();
132
133         jsonBoard.put("row0", boardCodified.get(0));
134         jsonBoard.put("row1", boardCodified.get(1));
135         jsonBoard.put("row2", boardCodified.get(2));
136         jsonBoard.put("row3", boardCodified.get(3));
137         jsonBoard.put("row4", boardCodified.get(4));
138         jsonBoard.put("row5", boardCodified.get(5));
139         jsonBoard.put("row6", boardCodified.get(6));
140         jsonBoard.put("row7", boardCodified.get(7));
141
142         return jsonBoard;
143     }
```

**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.

```
153     {
154         return this.board;
155     }
```

**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.

```
163     {
164         return this.piecesPlayer1;
165     }
```

#### 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.

```
173 {  
174     return this.piecesPlayer2;  
175 }
```

#### 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.

```
184 {  
185     ArrayList<Pair<Integer, Integer> player1 = validPositions(PieceType.PLAYER1, canEatHorizontally,  
    canEatVertically, canEatDiagonally);  
186     ArrayList<Pair<Integer, Integer> player2 = validPositions(PieceType.PLAYER2, canEatHorizontally,  
    canEatVertically, canEatDiagonally);  
187     //If there is no possible movements == Board Invalid  
188     if(player1.isEmpty() && player2.isEmpty()) throw new InvalidBoardException();  
189 }
```

### 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

<i>myPieceType</i>	<a href="#">PieceType</a> variable that represents the player in a cell.
<i>canEatHorizontally</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Horizontal manner.
<i>canEatVertically</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Vertical manner.
<i>canEatDiagonally</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Diagonal manner.

Definition at line 203 of file Board.java.

```
204 {
205     ArrayList<Pair<Integer,Integer> availablePos = new ArrayList<Pair<Integer,Integer>();
206     boolean posValid = false;
207
208     for(int i = 0; i < 8; ++i)
209     {
210         for(int j = 0; j < 8; ++j)
211         {
212             Pair<Integer, Integer> iterator = new Pair<Integer,Integer>(i,j);
213             posValid = false;
214
215             if(this.board[i][j] == null && surroundingPieces(iterator, myPieceType,
canEatHorizontally, canEatVertically, canEatDiagonally))
216             {
217                 if(canEatHorizontally)
218                 {
219                     ArrayList<Pair<Integer,Integer> horizontal = canPlaceHorizontal(iterator,
myPieceType);
220                     //IF NOT EMPTY
221                     if(!horizontal.isEmpty()) posValid = true;
222                 }
223                 if(canEatVertically)
224                 {
225                     ArrayList<Pair<Integer,Integer> vertical = canPlaceVertical(iterator,
myPieceType);
226                     //IF NOT EMPTY
227                     if(!vertical.isEmpty()) posValid = true;
```

```

228         }
229         if (canEatDiagonally)
230         {
231             ArrayList<Pair<Integer,Integer> diagonal = canPlaceDiagonal(iterator,
myPieceType);
232             //IF NOT EMPTY
233             if (!diagonal.isEmpty()) posValid = true;
234         }
235         //It's a valid position to add a Piece
236         if (posValid) availablePos.add(iterator);
237     }
238 }
239 }
240
241     return availablePos;
242 }

```

### 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

<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.
-----------------	--

Definition at line 252 of file Board.java.

```

253     {
254         Integer row = position.first;
255         Integer column = position.second;
256
257         if (this.board[row][column] == PieceType.PLAYER1) this.piecesPlayer1 -= 1;
258         if (this.board[row][column] == PieceType.PLAYER2) this.piecesPlayer2 -= 1;
259
260         this.board[row][column] = null;
261     }

```

### 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 *Piectype* 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

<i>myPieceType</i>	<a href="#">PieceType</a> variable that represents the player in a cell.
<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.
<i>canEatHorizontally</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Horizontal manner..
<i>canEatVertically</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Vertical manner.
<i>canEatDiagonally</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Diagonal manner.

Definition at line 277 of file Board.java.

```

278     {
279         ArrayList<Pair<Integer,Integer> horizontal = new ArrayList<Pair<Integer,Integer>();
280         ArrayList<Pair<Integer,Integer> vertical = new ArrayList<Pair<Integer,Integer>();
281         ArrayList<Pair<Integer,Integer> diagonal = new ArrayList<Pair<Integer,Integer>();
282
283         //if the position given to us is not null, it means it's owned by PLAYER1 or PLAYER2. Therefore,
284         //we won't add a Piece there and we will return.
285         //Although this will never happen when we use this method (because we will make sure it's a
286         //valid position),
287         // we added this so this method is more reusable for other future projects.
288         if(this.board[position.first][position.second] != null) return;
289
290         if(canEatHorizontally) //Includes eating HORIZONTALLY activated
291         {
292             horizontal = canPlaceHorizontal(position, myPieceType);
293             for(int i = 0; i < horizontal.size(); i++)
294                 changePieces(position, horizontal.get(i), myPieceType);
295         }
296
297         if (canEatVertically) //Includes eating VERTICALLY activated
298         {
299             vertical = canPlaceVertical(position, myPieceType);
300             for(int i = 0; i < vertical.size(); i++) {
301                 changePieces(position, vertical.get(i), myPieceType);
302             }
303         }
304
305         if (canEatDiagonally) //Includes eating DIAGONALLY activated
306         {
307             diagonal = canPlaceDiagonal(position, myPieceType);

```

```

306         for(int i = 0; i < diagonal.size(); i++)
307             changePieces(position,diagonal.get(i),myPieceType);
308     }
309     //If we added a piece to the board, we must add this to the piecesPlayerx attribute
310     if((canEatHorizontally && !horizontal.isEmpty()) || (canEatVertically && !vertical.isEmpty())
    || (canEatDiagonally && !diagonal.isEmpty()))
311     {
312         if(myPieceType == PieceType.PLAYER1) this.piecesPlayer1++;
313         if(myPieceType == PieceType.PLAYER2) this.piecesPlayer2++;
314     }
315 }

```

### 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

<i>myPieceType</i>	<a href="#">PieceType</a> variable that represents the player in a cell.
<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.

Definition at line 324 of file Board.java.

```

325     {
326         Integer row = position.first;
327         Integer column = position.second;
328
329         if(this.board[row][column] == PieceType.PLAYER1 && myPieceType == PieceType.PLAYER2)
330         {
331             this.piecesPlayer2 += 1;
332             this.piecesPlayer1 -= 1;
333         }
334
335         if(this.board[row][column] == PieceType.PLAYER2 && myPieceType == PieceType.PLAYER1)
336         {
337             this.piecesPlayer1 += 1;
338             this.piecesPlayer2 -= 1;
339         }
340
341         if(this.board[row][column] == null && myPieceType == PieceType.PLAYER1) this.piecesPlayer1 += 1;
342         if(this.board[row][column] == null && myPieceType == PieceType.PLAYER2) this.piecesPlayer2 += 1;
343
344         this.board[row][column] = myPieceType;
345     }

```

### 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 subtracting 1 to the y or x value (taking into consideration the board's limits obviously).

#### Parameters

<i>myPieceType</i>	<a href="#">PieceType</a> variable that represents the player in a cell.
<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.
<i>canEatHorizontally</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Horizontal manner.
<i>canEatVertically</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Vertical manner.
<i>canEatDiagonally</i>	Boolean value from <a href="#">Configuration</a> that determines if we can capture pieces of the <i>myPieceType</i> opponent in a Diagonal manner.

Definition at line 362 of file Board.java.

```

363     {
364         PieceType opponentPiece = inversePlayer(myPieceType);
365
366         if(canEatDiagonally)
367         {
368             //TOP LEFT
369             if(position.first > 0 && position.second > 0 &&
this.board[position.first-1][position.second-1] == opponentPiece) return true;
370             //TOP RIGHT
371             if(position.first > 0 && position.second < 7 &&
this.board[position.first-1][position.second+1] == opponentPiece) return true;
372             //BOTTOM RIGHT
373             if(position.first < 7 && position.second < 7 &&
this.board[position.first+1][position.second+1] == opponentPiece) return true;
374             //BOTTOM LEFT
375             if(position.first < 7 && position.second > 0 &&
this.board[position.first+1][position.second-1] == opponentPiece) return true;
376         }
377
378         if(canEatVertically)
379         {
380             //BOTTOM
381             if(position.first < 7 && this.board[position.first+1][position.second] == opponentPiece)
return true;

```

```

382         //TOP
383         if(position.first > 0 && this.board[position.first-1][position.second] == opponentPiece)
384             return true;
385     }
386     if(canEatHorizontally)
387     {
388         //RIGHT
389         if(position.second < 7 && this.board[position.first][position.second+1] == opponentPiece)
390             return true;
391         //LEFT
392         if(position.second > 0 && this.board[position.first][position.second-1] == opponentPiece)
393             return true;
394     }
395     //If none are true
396     return false;
397 }

```

### 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.

```

404     {
405         ArrayList<String> boardCodified = new ArrayList<String>(8);
406         String operational = "";
407         for(int i = 0; i < 8; ++i)
408         {
409             operational = "";
410             for(int j = 0; j < 8; ++j)
411             {
412                 if(this.board[i][j] == PieceType.PLAYER1) operational = operational + "B";
413                 if(this.board[i][j] == PieceType.PLAYER2) operational = operational + "N";
414                 if(this.board[i][j] == null) operational = operational + "?";
415             }
416             boardCodified.add(operational);
417         }
418     }
419     return boardCodified;
420 }
421

```



### 6.3.3.12 transcribeToPieceType()

```
void domain.Board.transcribeToPieceType (
    String row,
    Integer numRows ) [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 numRows 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

<i>row</i>	String of a row of a board in storing format.
<i>numRow</i>	Number of the row in the board its transcribing.

Definition at line 432 of file Board.java.

```
433     {
434         for(int i = 0; i < 8; ++i)
435         {
436             if(row.charAt(i) == '?') this.board[numRow][i] = null;
437             if(row.charAt(i) == 'B')
438             {
439                 this.board[numRow][i] = PieceType.PLAYER1;
440                 this.piecesPlayer1++;
441             }
442             if(row.charAt(i) == 'N')
443             {
444                 this.board[numRow][i] = PieceType.PLAYER2;
445                 this.piecesPlayer2++;
446             }
447         }
448     }
```

### 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

<i>myPieceType</i>	<a href="#">PieceType</a> variable that represents the player in a cell.
--------------------	--

Definition at line 457 of file Board.java.

```

458     {
459         if(myPieceType == PieceType.PLAYER1) return PieceType.PLAYER2;
460         else return PieceType.PLAYER1;
461     }

```

### 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

<i>position</i>	<a href="#">Pair</a> <Integer,Integer> that represents a position in a board.
<i>myPieceType</i>	<a href="#">PieceType</a> variable that represents the player in a cell.

Definition at line 471 of file Board.java.

```

472     {
473         ArrayList< Pair<Integer,Integer> > result = new ArrayList<Pair<Integer, Integer>>();
474
475         Integer row = position.first;
476         Integer column = position.second;
477         PieceType opponentPiece = inversePlayer(myPieceType);
478
479         if(column > 0) //To not go out of the boards boundaries
480         {
481             //to check if we can eat LEFT SIDE
482             if(this.board[row][column-1] == opponentPiece)
483             {
484                 Integer it1 = column-1;
485                 Boolean found1 = false;
486
487                 while(it1 >= 0 && this.board[row][it1] != null && found1 == false)
488                     //go through the line of the board to see if we can place the piece we want in
489                     "position". If so, we add the position of the piece that closes in the result array.
490                     {
491                         if(this.board[row][it1] == myPieceType) //found a piece that's mine = CAN PLACE
492                         {
493                             result.add(new Pair<Integer,Integer>(row,it1));
494                             found1 = true;
495                         }
496                     }
497             }
498         }
499     }

```

```

495         else //found another piece of the opponent = CONTINUE THE HUNT
496             it1 -= 1;
497     }
498 }
499 }
500
501 if(column < 7) //To not go out of the boards boundaries
502 {
503     //to check if we can eat RIGHT SIDE
504     if(this.board[row][column+1] == opponentPiece)
505     {
506         Integer it2 = column+1;
507         Boolean found2 = false;
508
509         while(it2 <= 7 && this.board[row][it2] != null && found2 == false)
510             //go through the line of the board to see if we can place the piece we want in
511             "position". If so, we add the position of the piece that closes in the result array.
512             {
513                 if(this.board[row][it2] == myPieceType) //found a piece that's mine = CAN PLACE
514                 {
515                     result.add(new Pair<Integer,Integer>(row,it2));
516                     found2 = true;
517                 }
518                 else //found another piece of the opponent = CONTINUE THE HUNT
519                 {
520                     it2 += 1;
521                 }
522             }
523     }
524
525     return result;
526 }

```

### 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

<i>position</i>	<i>Pair&lt;Integer,Integer&gt;</i> that represents a position in a board.
<i>myPieceType</i>	<i>PieceType</i> variable that represents the player in a cell.

Definition at line 536 of file `Board.java`.

```

537 {
538     ArrayList< Pair<Integer,Integer> > result = new ArrayList<Pair<Integer, Integer>>();
539
540     Integer row = position.first;

```

```

541     Integer column = position.second;
542     PieceType opponentPiece = inversePlayer(myPieceType);
543
544     if(row > 0) //To not go out of the boards boundaries
545     {
546         if(this.board[row-1][column] == opponentPiece) //to check if left side can be eaten
547         {
548             Integer it1 = row-1;
549             Boolean found1 = false;
550
551             while(it1 >= 0 && this.board[it1][column] != null && found1 == false)
552                 //go through the line of the board to see if we can place the piece we want in
553                 "position". If so, we add the position of the piece that closes in the result array.
554                 {
555                     if(this.board[it1][column] == myPieceType) //found a piece that's mine = CAN PLACE
556                     {
557                         result.add(new Pair<Integer,Integer>(it1,column));
558                         found1 = true;
559                     }
560                     else //found another piece of the opponent = CONTINUE THE HUNT
561                         it1 -= 1;
562                 }
563             }
564
565     if(row < 7)
566     {
567         if(this.board[row+1][column] == opponentPiece) //to check if right side can be eaten
568         {
569             Integer it2 = row+1;
570             Boolean found2 = false;
571
572             while(it2 <= 7 && this.board[it2][column] != null && found2 == false)
573                 //go through the line of the board to see if we can place the piece we want in
574                 "position". If so, we add the position of the piece that closes in the result array.
575                 {
576                     if(this.board[it2][column] == myPieceType) //found a piece that's mine = CAN PLACE
577                     {
578                         result.add(new Pair<Integer,Integer>(it2,column));
579                         found2 = true;
580                     }
581                     else //found another piece of the opponent = CONTINUE THE HUNT
582                     {
583                         it2 += 1;
584                     }
585                 }
586             }
587
588     return result;
589 }

```

### 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

<i>position</i>	<i>Pair&lt;Integer,Integer&gt; that represents a position in a board.</i>
<i>myPieceType</i>	<i>PieceType variable that represents the player in a cell.</i>

Definition at line 599 of file Board.java.

```

600     {
601         ArrayList< Pair<Integer,Integer> > result = new ArrayList<Pair<Integer, Integer>>();
602
603         Integer row = position.first;
604         Integer column = position.second;
605         PieceType opponentPiece = inversePlayer(myPieceType);
606
607         //DIAGONAL UP LEFT
608         if(row > 0 && column > 0) //To not go out of the boards boundaries
609         {
610             if(this.board[row-1][column-1] == opponentPiece) //to check if we can eat some opponent
pieces in the upper left diagonal
611             {
612                 Integer itRow = row-1;
613                 Integer itCol = column-1;
614                 Boolean found = false;
615
616                 while(itRow >= 0 && itCol >= 0 && this.board[itRow][itCol] != null && found == false)
617                     //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.
618                 {
619                     if(this.board[itRow][itCol] == myPieceType) //found a piece that's mine in the
diagonal line = CAN PLACE
620                     {
621                         result.add(new Pair<Integer,Integer>(itRow,itCol));
622                         found = true;
623                     }
624                     else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
625                     {
626                         itRow -= 1;
627                         itCol -= 1;
628                     }
629                 }
630
631             }
632         }
633
634         //DIAGONAL UP RIGHT
635         if(row > 0 && column < 7) //To not go out of the boards boundaries
636         {
637             if(this.board[row-1][column+1] == opponentPiece) //to check if we can eat some opponents
pieces in the upper right diagonal
638             {
639                 Integer itRow = row-1;
640                 Integer itCol = column+1;
641                 Boolean found = false;
642
643                 while(itRow >= 0 && itCol <= 7 && this.board[itRow][itCol] != null && found == false)
644                     //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.
645                 {
646                     if(this.board[itRow][itCol] == myPieceType)
647                     {
648                         result.add(new Pair<Integer,Integer>(itRow,itCol));
649                         found = true;
650                     }
651                     else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
652                     {
653                         itRow -= 1;
654                         itCol += 1;
655                     }
656                 }
657
658             }
659         }
660
661         //DIAGONAL DOWN LEFT
662         if(row < 7 && column > 0) //To not go out of the boards boundaries
663         {
664             if(this.board[row+1][column-1] == opponentPiece) //to check if we can eat some opponents
pieces in the bottom left diagonal
665             {
666                 Integer itRow = row+1;
667                 Integer itCol = column-1;
668                 Boolean found = false;
669
670                 while(itRow <= 7 && itCol >= 0 && this.board[itRow][itCol] != null && found == false)

```

```

671         //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.
672         {
673             if(this.board[itRow][itCol] == myPieceType)
674             {
675                 result.add(new Pair<Integer,Integer>(itRow,itCol));
676                 found = true;
677             }
678             else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
679             {
680                 itRow += 1;
681                 itCol -= 1;
682             }
683         }
684     }
685 }
686 }
687
688 //DIAGONAL DOWN RIGHT
689 if(row < 7 && column < 7) //To not go out of the boards boundaries
690 {
691     if(this.board[row+1][column+1] == opponentPiece) //to check if we can eat some opponent
        pieces in the bottom right diagonal
692     {
693         Integer itRow = row+1;
694         Integer itCol = column+1;
695         Boolean found = false;
696
697         while(itRow <= 7 && itCol <= 7 && this.board[itRow][itCol] != null && found == false)
698             //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.
699         {
700             if(this.board[itRow][itCol] == myPieceType)
701             {
702                 result.add(new Pair<Integer,Integer>(itRow,itCol));
703                 found = true;
704             }
705             else //found another piece of the opponent in the diagonal line = CONTINUE THE HUNT
706             {
707                 itRow += 1;
708                 itCol += 1;
709             }
710         }
711     }
712 }
713 }
714
715 return result;
716
717 }

```

### 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

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

Definition at line 727 of file Board.java.

```

728     {
729         Pair<Integer,Integer> position = new Pair<Integer,Integer>(addPiece.first,addPiece.second);
730         Integer diffRow = lastPiece.first - addPiece.first;
731         Integer diffCol = lastPiece.second - addPiece.second;
732         Integer dirRow = 0, dirCol = 0;
733         PieceType opponent = inversePlayer(myPieceType);
734
735         if(diffRow == 0) //HORIZONTAL
736         {
737             if(diffCol > 0) //RIGHT
738                 dirCol = 1;
739             else //LEFT
740                 dirCol = -1;
741         }
742
743         if(diffCol == 0) //VERTICAL
744         {
745             if(diffRow > 0) //UP
746                 dirRow = 1;
747             else //DOWN
748                 dirRow = -1;
749         }
750
751         if(diffCol != 0 && diffRow != 0) //DIAGONAL
752         {
753             if(diffRow > 0 && diffCol > 0) //DIAGONAL BOTTOM RIGHT
754             {
755                 dirRow = 1;
756                 dirCol = 1;
757             }
758             if(diffRow > 0 && diffCol < 0) //DIAGONAL BOTTOM LEFT
759             {
760                 dirRow = 1;
761                 dirCol = -1;
762             }
763             if(diffRow < 0 && diffCol > 0) //DIAGONAL TOP RIGHT
764             {
765                 dirRow = -1;
766                 dirCol = 1;
767             }
768             if(diffRow < 0 && diffCol < 0) //DIAGONAL TOP LEFT
769             {
770                 dirRow = -1;
771                 dirCol = -1;
772             }
773         }
774
775
776         while((position.first != lastPiece.first) || (position.second != lastPiece.second)) {
777             if(this.board[position.first][position.second] == opponent && opponent == PieceType.PLAYER1)
778             {
779                 this.piecesPlayer1--;
780                 this.piecesPlayer2++;
781             }
782             if(this.board[position.first][position.second] == opponent && opponent == PieceType.PLAYER2)
783             {
784                 this.piecesPlayer1++;
785                 this.piecesPlayer2--;
786             }
787
788             this.board[position.first][position.second] = myPieceType;
789             position.first = position.first + dirRow;
790             position.second = position.second + dirCol;
791         }
792     }

```

### 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 class 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 class 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.

```
26         {  
27     }
```

### 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. *Piectype* 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

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

Definition at line 44 of file BoardCtrl.java.

```

45         {
46             board.placePiece(position, myPieceType, configuration.getCanEatHorizontally(),
47                             configuration.getCanEatVertically(), configuration.getCanEatDiagonally());
48
49             return board;
50         }

```

### 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

<i>board</i>	Instance of a <a href="#">Board</a> class which is the one we will modify and return.
<i>myPieceType</i>	PieceType variable that represents the player in a cell.
<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.

Definition at line 60 of file BoardCtrl.java.

```

60         {
61             board.placePieceConfig(position, myPieceType);
62
63             return board;
64         }

```

### 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

<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.
<i>board</i>	Instance of a <a href="#">Board</a> class which is the one we will modify and return.

Definition at line 73 of file BoardCtrl.java.

```
73                                     {
74         board.removePiece(position);
75
76         return board;
77     }
```

### 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

<i>board</i>	Instance of a <a href="#">Board</a> class which is the one we will modify and return.
--------------	---

Definition at line 85 of file BoardCtrl.java.

```

85         {
86         Pair<Integer, Integer> totalPieces = new Pair<Integer, Integer>(board.getPiecesPlayer1(),
87             board.getPiecesPlayer2());
88
89         return totalPieces;
90     }

```

#### 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 its 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

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

Definition at line 102 of file BoardCtrl.java.

```

103     {
104     ArrayList<Pair<Integer, Integer>> validPos = new ArrayList<Pair<Integer, Integer>>();
105
106     validPos = board.validPositions(myPieceType, configuration.getCanEatHorizontally(),
107         configuration.getCanEatVertically(), configuration.getCanEatDiagonally());
108
109     return validPos;
110 }

```

#### 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`.

```
118                                     {
119         board.isValid(configuration.getCanEatHorizontally(), configuration.getCanEatVertically(),
120                        configuration.getCanEatDiagonally());
121     }
```

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.

```
23 {}
```

### 6.5.3 Member Function Documentation

#### 6.5.3.1 start()

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

Definition at line 24 of file BoardDriver.java.

```

25     {
26         String title = new String();
27         while(true)
28         {
29             title = (this.currentBoard != null ? String.format("\tcurrent Board %s selected\n",
this.nameCurrentBoard): "\tNo current Board is selected\n");
30             switch (Driver.menu(title, "Board Driver Menu",
31                 new Pair<String, String>("0", "Create a Default Board"),
32                 new Pair<String, String>("1", "Create a Board based on the board of the currentBoard"),
33                 new Pair<String, String>("2", "Load a Board from a file"),
34                 new Pair<String, String>("3", "Serialize the current Board to JSON"),
35                 new Pair<String, String>("4", "Get the number of pieces of both players of the current
Board"),
36                 new Pair<String, String>("5", "Get the whole current Board"),
37                 new Pair<String, String>("6", "Place a piece on the current Board"),
38                 new Pair<String, String>("7", "Remove a piece of the current Board"),
39                 new Pair<String, String>("8", "Place a piece on the current Board as if it was an initial
board of a configuration"),
40                 new Pair<String, String>("9", "Check if the current Board is valid to play"),
41                 new Pair<String, String>("10", "Check which positions you can add a piece to in the
current Board"),
42                 new Pair<String, String>("11", "Deserialize the current Board"),
43                 new Pair<String, String>("12", "Print the current Board"))

```

```

44         {
45             case "0":
46                 defaultBoard();
47                 break;
48             case "1":
49                 createBoard();
50                 break;
51             case "2":
52                 loadBoard();
53                 break;
54             case "3":
55                 serializeBoard();
56                 break;
57             case "4":
58                 getPiecesPlayers();
59                 break;
60             case "5":
61                 getCurrentBoard();
62                 break;
63             case "6":
64                 placePieceBoard();
65                 break;
66             case "7":
67                 removePieceBoard();
68                 break;
69             case "8":
70                 placePieceInitialBoard();
71                 break;
72             case "9":
73                 validBoard();
74                 break;
75             case "10":
76                 addPositions();
77                 break;
78             case "11":
79                 deserialize();
80                 break;
81             case "12":
82                 {
83                     if (this.currentBoard != null)
84                     {
85                         System.out.println(String.format("==== Printing Board %s ====\n",
86 this.nameCurrentBoard));
87                         System.out.println(String.format("Board %s printed below!\n",
88 this.nameCurrentBoard));
89                     }
90                     printCurrentBoard();
91                 }
92                 break;
93             }
94         }
95         Driver.pause();
96     }
97 }

```

### 6.5.3.2 defaultBoard()

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

Definition at line 98 of file BoardDriver.java.

```

99     {
100         Driver.clear();
101         System.out.println("==== Creating a Default Board ====\n");
102         this.currentBoard = new Board();
103         this.nameCurrentBoard = "Default";
104
105         System.out.println("Default Board was created succesfully, printed below:");
106         System.out.println("\n");
107         printCurrentBoard();
108         System.out.println("\n");
109         System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
110         System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2());
111         System.out.println("\n");
112     }

```

### 6.5.3.3 createBoard()

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

Definition at line 114 of file BoardDriver.java.

```

115     {
116         Driver.clear();
117         if(this.currentBoard == null)
118         {
119             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");
120         }
121         else
122         {
123             PieceType[][] board = this.currentBoard.getBoard();
124             System.out.println("Getting the currentBoard board attribute and creating a new Board
instance based off of it.\n");
125             this.currentBoard = new Board(board);
126             printCurrentBoard();
127             System.out.println("\n");
128             System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
129             System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
130             System.out.println("Created the Board class instance successfully\n");
131         }
132     }

```

### 6.5.3.4 loadBoard()

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

Definition at line 134 of file BoardDriver.java.

```

135     {
136         Driver.clear();
137         FixtureRepository fixtureRepo = new FixtureRepository();
138         System.out.println("==== What Board would you like to load? =====\n");
139         ArrayList<String> listBoards = fixtureRepo.listFiles();
140
141         for(Integer i = 0; i < listBoards.size(); ++i)
142         {
143             System.out.println(String.format("[%d] ", i) + listBoards.get(i));
144         }
145
146         System.out.println("\n");
147
148         Boolean checkValid = false;
149         Integer nameBoard = null;
150
151         while(checkValid == false)
152         {
153             try {
154                 nameBoard = Integer.parseInt(Driver.input("What Board would you like to load?"));
155                 if(nameBoard < 0 || nameBoard >= listBoards.size() )
156                     System.out.println("Incorrect Index! Try again :D");
157                 else checkValid = true;
158             } catch (Exception e) {
159                 System.out.println("You didn't add an Integer! Try again :D");
160             }
161         }
162
163         Driver.clear();
164         String path = listBoards.get(nameBoard);
165         JSONObject newBoard = fixtureRepo.boardFileToJSON(path);
166
167         this.nameCurrentBoard = path;
168         this.currentBoard = new Board(newBoard);
169
170         System.out.println(String.format("%s was loaded succesfully. \n\nIt's printed below!\n",
path));
171         printCurrentBoard();
172         System.out.println("\n");
173         System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
174         System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
175     }

```



### 6.5.3.5 serializeBoard()

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

Definition at line 177 of file BoardDriver.java.

```
178     {
179         Driver.clear();
180         if(this.currentBoard == null)
181         {
182             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");
183         }
184         else
185         {
186             System.out.println(String.format("==== Serializing Board %s ====\n",
this.nameCurrentBoard));
187             printCurrentBoard();
188             System.out.println("\n" + this.nameCurrentBoard + " Board was serialized correctly into JSON
format.\n");
189             System.out.println(this.currentBoard.serialize().toString(2) + "\n");
190         }
191     }
```

### 6.5.3.6 getPiecesPlayers()

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

Definition at line 193 of file BoardDriver.java.

```
194     {
195         Driver.clear();
196         if(this.currentBoard == null)
197         {
198             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");
199         }
200         else
201         {
202             System.out.println(String.format("==== Getting the number of pieces of both players of the
current Board %s ====\n", this.nameCurrentBoard));
203             printCurrentBoard();
204             System.out.println("\nPieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
205             System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
206         }
207     }
```

### 6.5.3.7 getCurrentBoard()

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

Definition at line 209 of file BoardDriver.java.

```
210     {
211         if(this.currentBoard == null)
212         {
213             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");
214         }
215         else
216         {
217             System.out.println(String.format("==== Getting the current Board %s ====\n",
this.nameCurrentBoard));
218             PieceType[][] gotBoard = currentBoard.getBoard();
219             System.out.println("Get Board was executed correctly and it's saved in a PieceType[][]
called gotBoard. gotBoard is printed below:\n");
220             currentBoard = new Board(gotBoard);
221             printCurrentBoard();
222         }
223     }
```

### 6.5.3.8 placePieceBoard()

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

Definition at line 225 of file BoardDriver.java.

```

226     {
227         Driver.clear();
228         if(this.currentBoard == null)
229         {
230             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");
231         }
232         else
233         {
234             System.out.println(String.format("==== Placing a piece on %s Board ==== \n",
this.nameCurrentBoard));
235
236             printCurrentBoard();
237             System.out.println("\n");
238             System.out.println("Write what position you would like to add a piece in the board\n");
239
240             Integer row = correctNumber("row");
241             Integer col = correctNumber("column");
242             Pair<Integer, Integer> pos = new Pair<Integer, Integer>(row,col);
243             PieceType myPieceType = correctPieceType();
244             Boolean horizontal = correctBoolean("Horizontal"), vertical = correctBoolean("Vertical"),
diagonal = correctBoolean("Diagonal");
245
246             Integer sumPieces = this.currentBoard.getPiecesPlayer1() +
this.currentBoard.getPiecesPlayer2();
247             this.currentBoard.placePiece(pos, myPieceType, horizontal, vertical, diagonal);
248
249             if(sumPieces == (this.currentBoard.getPiecesPlayer1() +
this.currentBoard.getPiecesPlayer2()))
250                 System.out.println("\nNo pieces were added, as the parameters you inserted didn't give a
valid position to place a piece.");
251             else
252                 System.out.println("\nSuccesfully added a piece in position: " + pos);
253
254             System.out.println("\n");
255             printCurrentBoard();
256             System.out.println("\n");
257             System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
258             System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
259
260             playAgain("placePieceBoard");
261         }
262     }

```

### 6.5.3.9 removePieceBoard()

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

Definition at line 264 of file BoardDriver.java.

```

265     {
266         Driver.clear();
267         if(this.currentBoard == null)
268         {
269             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");
270         }
271         else
272         {
273             System.out.println(String.format("==== Removing a piece on %s Board ==== \n",
this.nameCurrentBoard));
274
275             printCurrentBoard();
276             System.out.println("\n");
277             System.out.println("Write what position you would like to remove a piece from\n");
278             Integer row = correctNumber("row");
279             Integer col = correctNumber("column");
280             Pair<Integer, Integer> pos = new Pair<Integer, Integer>(row,col);
281
282             this.currentBoard.removePiece(pos);
283

```

```

284         System.out.println("\n");
285         printCurrentBoard();
286         System.out.println("\n");
287         System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
288         System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
289
290         playAgain("removePieceBoard");
291     }
292 }

```

### 6.5.3.10 placePieceInitialBoard()

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

Definition at line 294 of file BoardDriver.java.

```

295 {
296     Driver.clear();
297     if(this.currentBoard == null)
298     {
299         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");
300     }
301     else
302     {
303         System.out.println(String.format("==== Placing a piece on %s INITIAL Board ==== \n",
this.nameCurrentBoard));
304
305         printCurrentBoard();
306         System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
307         System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
308         System.out.println("Write what position you would like to add a piece to the board\n");
309
310         Integer row = correctNumber("row");
311         Integer col = correctNumber("column");
312         Pair<Integer, Integer> pos = new Pair<Integer, Integer>(row,col);
313
314         PieceType myPieceType = correctPieceType();
315
316         this.currentBoard.placePieceConfig(pos, myPieceType);
317         System.out.println("\n");
318         printCurrentBoard();
319         System.out.println("\n");
320         System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
321         System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
322
323         playAgain("placePieceInitialBoard");
324     }
325 }

```

### 6.5.3.11 validBoard()

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

Definition at line 327 of file BoardDriver.java.

```

328 {
329     Driver.clear();
330     if(this.currentBoard == null)
331     {
332         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");
333     }
334     else
335     {
336         System.out.println(String.format("==== Checking if %s Board is valid ==== \n",
this.nameCurrentBoard));
337         printCurrentBoard();
338         System.out.println("\n");
339
340         try {

```

```

341         Boolean horizontal = correctBoolean("Horizontal"), vertical =
correctBoolean("Vertical"), diagonal = correctBoolean("Diagonal");
342
343         this.currentBoard.isValid(horizontal, vertical, diagonal);
344
345         System.out.println("\nThe board is valid with the configuration you've inserted!\n");
346         ArrayList<Pair<Integer,Integer>> validPosPlayer1 =
this.currentBoard.validPositions(PieceType.PLAYER1, horizontal, vertical, diagonal);
347         ArrayList<Pair<Integer,Integer>> validPosPlayer2 =
this.currentBoard.validPositions(PieceType.PLAYER1, horizontal, vertical, diagonal);
348
349         if(!validPosPlayer1.isEmpty()) System.out.println("For example, PLAYER1(B) can add a
piece in " + validPosPlayer1.get(0) + "\n");
350         else if(!validPosPlayer2.isEmpty()) System.out.println("For example, PLAYER2(B) can add
a piece in " + validPosPlayer2.get(0) + "\n");
351
352     } catch (Exception e) {
353         System.out.println("\nThe board is invalid with the configuration you've inserted,
meaning neither of both players can add a piece.\n");
354     }
355 }
356 }

```

### 6.5.3.12 addPositions()

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

Definition at line 358 of file BoardDriver.java.

```

359 {
360     Driver.clear();
361     if(this.currentBoard == null)
362     {
363         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");
364     }
365     else
366     {
367         System.out.println(String.format("==== Getting all available positions of a PLAYER in %s
Board ====\n", this.nameCurrentBoard));
368         printCurrentBoard();
369         System.out.println("\n");
370
371         Boolean checkValues = false;
372         Boolean both = false;
373         PieceType myPieceType = null;
374
375         while(!checkValues)
376         {
377             try {
378                 String typePiece = new String(Driver.input("Write which color piece you want to see
its valid positions (B or N or BN)"));
379                 if(typePiece.equals("B"))
380                 {
381                     myPieceType = PieceType.PLAYER1;
382                     checkValues = true;
383                 }
384                 else if (typePiece.equals("N"))
385                 {
386                     myPieceType = PieceType.PLAYER2;
387                     checkValues = true;
388                 }
389                 else if (typePiece.equals("BN"))
390                 {
391                     myPieceType = PieceType.PLAYER2;
392                     checkValues = true;
393                     both = true;
394                 }
395                 else {
396                     System.out.println("\nERROR! Piece is neither B nor N nor BN\n");
397                 }
398             } catch (Exception e) {
399                 System.out.println("\nERROR! Piece is neither B nor N nor BN\n");
400             }
401         }
402
403         Boolean horizontal = correctBoolean("Horizontal"), vertical = correctBoolean("Vertical"),
diagonal = correctBoolean("Diagonal");
404

```

```

405         ArrayList<Pair<Integer,Integer> validPosPlayer1 =
this.currentBoard.validPositions(PieceType.PLAYER1, horizontal, vertical, diagonal);
406         ArrayList<Pair<Integer,Integer> validPosPlayer2 =
this.currentBoard.validPositions(PieceType.PLAYER2, horizontal, vertical, diagonal);
407
408         if(both || myPieceType == PieceType.PLAYER1)
409         {
410             for(int i = 0; i < validPosPlayer1.size(); ++i)
411             {
412                 if(i == 0) System.out.println("\nValid positions for PLAYER1(B):");
413                 System.out.println("Position: " + validPosPlayer1.get(i));
414             }
415         }
416
417         System.out.println("\n");
418
419         if(both || myPieceType == PieceType.PLAYER2)
420         {
421             for(int i = 0; i < validPosPlayer2.size(); ++i)
422             {
423                 if(i == 0) System.out.println("Valid positions for PLAYER2(N):");
424                 System.out.println("Position: " + validPosPlayer2.get(i));
425             }
426         }
427         System.out.println("\n");
428     }
429 }

```

### 6.5.3.13 deserialize()

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

Definition at line 431 of file BoardDriver.java.

```

432     {
433         Driver.clear();
434         if(this.currentBoard == null)
435         {
436             System.out.println("YOU MUST INITIALIZE A BOARD BEFORE DESERIALIZING IT!\nGo back to the
main menu and create a Default Board or Load a preexisting one\n");
437         }
438         else
439         {
440             System.out.println(String.format("==== Deserializing Board %s ==== \n",
this.nameCurrentBoard));
441
442             System.out.println(this.currentBoard.serialize().toString(2) + "\n");
443             this.currentBoard = new Board(this.currentBoard.serialize());
444
445             System.out.println("The currentBoard has been deserialized from the JSON format
successfully, as we can see below:\n");
446             printCurrentBoard();
447             System.out.println("\n");
448             System.out.println("Pieces PLAYER1(B): " + this.currentBoard.getPiecesPlayer1());
449             System.out.println("Pieces PLAYER2(N): " + this.currentBoard.getPiecesPlayer2() + "\n");
450         }
451     }

```

### 6.5.3.14 printCurrentBoard()

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

Definition at line 452 of file BoardDriver.java.

```

453     {
454         if(this.currentBoard == null)
455         {
456             System.out.println("YOU MUST INITIALIZE A BOARD BEFORE PRINTING IT!\nGo back to the main
menu and create a Default Board or Load a preexisting one\n");
457         }
458         else
459         {

```

```

460
461     ArrayList<String> boardCodified = transcribeToCharacters();
462     System.out.println("      0 1 2 3 4 5 6 7");
463     System.out.println("      -----");
464
465     for(Integer i = 0; i < 8; ++i)
466     {
467         String row = boardCodified.get(i);
468         System.out.println("  " + i + " | " + row.charAt(0) + " " + row.charAt(1) + " " +
row.charAt(2) + " " + row.charAt(3) +
469         " " + row.charAt(4) + " " + row.charAt(5) + " " + row.charAt(6) + "
" + row.charAt(7) + " ");
470     }
471
472     System.out.println("\n");
473 }
474 }

```

### 6.5.3.15 transcribeToCharacters()

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

Definition at line 478 of file BoardDriver.java.

```

480 {
481     ArrayList<String> boardCodified = new ArrayList<String>(8);
482     String operational = "";
483     PieceType[][] current = this.currentBoard.getBoard();
484
485     if (current != null)
486     {
487         for(int i = 0; i < 8; ++i)
488         {
489             operational = "";
490             for(int j = 0; j < 8; ++j)
491             {
492                 if(current[i][j] == PieceType.PLAYER1) operational = operational + "B";
493                 if(current[i][j] == PieceType.PLAYER2) operational = operational + "N";
494                 if(current[i][j] == null) operational = operational + "?";
495             }
496             boardCodified.add(operational);
497         }
498     }
499
500     return boardCodified;
501 }
502 }

```

### 6.5.3.16 correctNumber()

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

Definition at line 504 of file BoardDriver.java.

```

505 {
506     Boolean checkValues = false;
507     Integer returnNumber = -1;
508
509     while(!checkValues)
510     {
511         try {
512             Integer input = Integer.parseInt(Driver.input(String.format("Write the %s number",
rowOrColumn)));
513             if(input < 0 || input > 7) System.out.println(String.format("\nERROR! %s is not
between 0 and 7\n", rowOrColumn));
514             else
515             {
516                 checkValues = true;
517                 returnNumber = input;
518             }
519         } catch (Exception e) {

```

```

520             System.out.println(String.format("\nERROR! %s is not between 0 and 7\n",
rowOrColumn));
521         }
522     }
523     return returnNumber;
524 }

```

### 6.5.3.17 correctPieceType()

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

Definition at line 526 of file BoardDriver.java.

```

527 {
528     Boolean checkValues = false;
529     PieceType myPieceType = null;
530
531     while(!checkValues)
532     {
533         try {
534             char typePiece = new String(Driver.input("Write which color plays (B or N)")).charAt(0);
535             if(typePiece == 'B')
536             {
537                 myPieceType = PieceType.PLAYER1;
538                 checkValues = true;
539             }
540             else if (typePiece == 'N')
541             {
542                 myPieceType = PieceType.PLAYER2;
543                 checkValues = true;
544             }
545             else {
546                 System.out.println("\nERROR! Piece is not B or N\n");
547             }
548         } catch (Exception e) {
549             System.out.println("\nERROR! Piece is not B or N\n");
550         }
551     }
552     return myPieceType;
553 }

```

### 6.5.3.18 playAgain()

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

Definition at line 555 of file BoardDriver.java.

```

556 {
557     Boolean checkValues = false;
558     String methodDescription = null;
559
560     if(method.equals("placePieceInitialBoard") || method.equals("placePieceBoard"))
561         methodDescription = "add";
562     else if (method.equals("removePieceBoard"))
563         methodDescription = "remove";
564
565     while(!checkValues)
566     {
567         try {
568             char cont = new String(Driver.input(String.format("Do you want to %s another piece?
Write y or n", methodDescription))).charAt(0);
569             if(cont == 'y')
570             {
571                 checkValues = true;
572                 if(method.equals("placePieceInitialBoard"))
573                     placePieceInitialBoard();
574                 else if (method.equals("placePieceBoard"))
575                     placePieceBoard();
576                 else if (method.equals("removePieceBoard"))
577                     removePieceBoard();
578             }

```

```

579         else if(cont == 'n')
580         {
581             checkValues = true;
582             return;
583         }
584         else System.out.println("\nERROR! You didn't write y or n. Try again! :D\n");
585     } catch (Exception e) {
586         System.out.println("You didn't write y or n! Try again :D");
587     }
588 }
589 }

```

### 6.5.3.19 correctBoolean()

```

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

```

Definition at line 591 of file BoardDriver.java.

```

592 {
593     Boolean checkValues = false;
594     Boolean eating = null;
595
596     while(!checkValues)
597     {
598         try {
599             char input = new String(Driver.input(String.format("Can we eat in %s? Write y or n",
600 eatingMethod))).charAt(0);
601             if(input == 'y')
602             {
603                 eating = true;
604                 checkValues = true;
605             }
606             else if(input == 'n')
607             {
608                 eating = false;
609                 checkValues = true;
610             }
611             else System.out.println(String.format("\nERROR! %s eating was neither affirmative nor
612 negative\n", eatingMethod));
613         } catch (Exception e) {
614             System.out.println(String.format("\nERROR! %s eating was neither affirmative nor
615 negative\n", eatingMethod));
616         }
617     }
618     return eating;
619 }

```

## 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.

```
35     {
36         this.name = name;
37         this.difficulty = difficulty;
38         this.id = id;
39         this.isDeleted = false;
40         this.creatorID = creatorID;
41     }
```

### 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.

```
47     {
48         this.name = bot.getString("name");
49         this.id = UUID.fromString(bot.getString("id"));
50         this.difficulty = bot.getInt("difficulty");
51         this.isDeleted = bot.getBoolean("is_deleted");
52         this.creatorID = UUID.fromString(bot.getString("creator_id"));
53     }
```

## 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.

```
59         {
60             JSONObject bot = new JSONObject();
61             bot.put("name", this.name);
62             bot.put("id", this.id.toString());
63             bot.put("difficulty", this.difficulty);
64             bot.put("type", "BOT");
65             bot.put("is_deleted", this.isDeleted);
66             bot.put("creator_id", this.creatorID.toString());
67
68             return bot;
69         }
```

### 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.

```
79         {
80             return this.difficulty;
81         }
```

### 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.

```
88 {  
89     return this.creatorID;  
90 }
```

### 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.

```
98 {  
99     if(difficulty < 1 || difficulty > 10) {  
100         throw new InvalidDifficultyException();  
101     }  
102     this.difficulty = difficulty;  
103 }
```

## 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 endpoint. 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 endpoint. 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.

```
22                                     {
23         new BotDriver().start();
24     }
```

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.

```
17         {  
18             this.currentBot = null;  
19         }
```

### 6.8.3 Member Function Documentation

#### 6.8.3.1 start()

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

Definition at line 21 of file BotDriver.java.

```
21         {  
22             while(true) {  
23                 this.mainMenu();  
24             }  
25         }
```

### 6.8.3.2 mainMenu()

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

Definition at line 27 of file BotDriver.java.

```

27         {
28             String title = (this.currentBot != null ? String.format("Current: %s\n",
this.currentBot.getName()) : null);
29             switch (Driver.menu(title, "Bot Driver",
30                 new Pair<String, String>("1", "Create Bot"),
31                 new Pair<String, String>("2", "Get name"),
32                 new Pair<String, String>("3", "Set name"),
33                 new Pair<String, String>("4", "Get difficulty"),
34                 new Pair<String, String>("5", "Set difficulty"),
35                 new Pair<String, String>("6", "Get state"),
36                 new Pair<String, String>("7", "Set state"),
37                 new Pair<String, String>("8", "Get type"),
38                 new Pair<String, String>("9", "Get ID"),
39                 new Pair<String, String>("10", "Get creatorID"),
40                 new Pair<String, String>("11", "Serialize Bot to JSON"),
41                 new Pair<String, String>("12", "Deserialize Bot from JSON"))) {
42                 case "1":
43                     this.createBot();
44                     break;
45                 case "2":
46                     this.getName();
47                     break;
48                 case "3":
49                     this.setName();
50                     break;
51                 case "4":
52                     this.getDifficulty();
53                     break;
54                 case "5":
55                     this.setDifficulty();
56                     break;
57                 case "6":
58                     this.getIsDeleted();
59                     break;
60                 case "7":
61                     this.setIsDeleted();
62                     break;
63                 case "8":
64                     this.getType();
65                     break;
66                 case "9":
67                     this.getID();
68                     break;
69                 case "10":
70                     this.getCreatorID();
71                     break;
72                 case "11":
73                     this.serialize();
74                     break;
75                 case "12":
76                     this.deserialize();
77                     break;
78             }
79             Driver.pause();
80         }
81     }

```

### 6.8.3.3 createBot()

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

Definition at line 83 of file BotDriver.java.

```

83         {
84             System.out.println("Take into account that UUIDs will be randomly generated because typing them
in will be a hassle.\n");
85             String name = Driver.input("Name?");
86             Integer difficulty = Driver.inputInt("Difficulty? (From 1 to 10)");
87             try {
88                 Bot bot = new Bot("Default name", 0, UUID.randomUUID(), UUID.randomUUID());
89                 bot.setName(name);
90                 bot.setDifficulty(difficulty);

```



```

91         this.currentBot = bot;
92         System.out.println(String.format("%s created successfully!", this.currentBot.getName()));
93     } catch (Exception e) {
94         System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
95             e.getMessage()));
96     }

```

### 6.8.3.4 serialize()

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

Definition at line 98 of file BotDriver.java.

```

98     {
99         if(this.currentBot == null) {
100             System.out.println("No current Bot");
101             return;
102         }
103         System.out.println(String.format("%s's serialized to JSON is: %s", this.currentBot.getName(),
104             this.currentBot.serialize().toString(2)));
105     }

```

### 6.8.3.5 deserialize()

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

Definition at line 108 of file BotDriver.java.

```

108     {
109         if(this.currentBot == null) {
110             System.out.println("No current Bot");
111             return;
112         }
113         System.out.println(this.currentBot.serialize().toString(2));
114         this.currentBot = new Bot(this.currentBot.serialize());
115         System.out.println(String.format("\n%s's deserialized from the above JSON successfully!\n",
116             this.currentBot.getName()));
117         System.out.println(String.format("name:\t\t\t%s", this.currentBot.getName()));
118         System.out.println(String.format("id:\t\t\t%s", this.currentBot.getID()));
119         System.out.println(String.format("difficulty:\t\t\t%s", this.currentBot.getDifficulty()));
120         System.out.println(String.format("is_deleted:\t\t\t%s", this.currentBot.getIsDeleted()));
121         System.out.println(String.format("creator_id:\t\t\t%s", this.currentBot.getCreatorID()));
122     }
123 }

```

### 6.8.3.6 getName()

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

Definition at line 125 of file BotDriver.java.

```

125     {
126         if(this.currentBot == null) {
127             System.out.println("No current bot!");
128             return;
129         }
130         System.out.println(String.format("%s's bot name is: %s", this.currentBot.getName(),
131             this.currentBot.getName()));
132     }

```

### 6.8.3.7 getDifficulty()

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

Definition at line 133 of file BotDriver.java.

```
133     {
134         if(this.currentBot == null) {
135             System.out.println("No current bot!");
136             return;
137         }
138         System.out.println(String.format("%s's difficulty is: %s", this.currentBot.getName(),
139             this.currentBot.getDifficulty()));
139     }
```

### 6.8.3.8 getIsDeleted()

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

Definition at line 141 of file BotDriver.java.

```
141     {
142         if(this.currentBot == null) {
143             System.out.println("No current bot!");
144             return;
145         }
146         System.out.print(String.format("%s's state is: ", this.currentBot.getName()));
147         if(this.currentBot.getIsDeleted())
148             System.out.println("DELETED");
149         else
150             System.out.println("ACTIVE");
151     }
```

### 6.8.3.9 getType()

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

Definition at line 153 of file BotDriver.java.

```
153     {
154         System.out.println("Bot's type attribute was removed because of professor's feedback. However,
155             this option is still here to have backwards compatibility with old delivered documents.");
155     }
```

### 6.8.3.10 getID()

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

Definition at line 157 of file BotDriver.java.

```
157     {
158         if(this.currentBot == null) {
159             System.out.println("No current bot!");
160             return;
161         }
162         System.out.println(String.format("%s's ID is: %s", this.currentBot.getName(),
163             this.currentBot.getID()));
163     }
```

### 6.8.3.11 getCreatorID()

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

Definition at line 165 of file BotDriver.java.

```
165         {
166             if(this.currentBot == null) {
167                 System.out.println("No current bot!");
168                 return;
169             }
170             System.out.println(String.format("%s's CreatorID is: %s", this.currentBot.getName(),
171                 this.currentBot.getCreatorID()));
172         }
```

### 6.8.3.12 setName()

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

Definition at line 174 of file BotDriver.java.

```
174         {
175             if (this.currentBot == null) {
176                 System.out.println("No current bot!");
177                 return;
178             }
179             try {
180                 this.currentBot.setName(Driver.input("New name?"));
181                 System.out.println(String.format("%s name changed successfully!",
182                     this.currentBot.getName()));
183             } catch (Exception e) {
184                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
185                     e.getMessage()));
186             }
187         }
```

### 6.8.3.13 setDifficulty()

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

Definition at line 187 of file BotDriver.java.

```
187         {
188             if (this.currentBot == null) {
189                 System.out.println("No current bot!");
190                 return;
191             }
192             try {
193                 this.currentBot.setDifficulty(Driver.inputInt("Choose a difficulty level from 1 to 10"));
194                 System.out.println(String.format("%s's difficulty has been changed successfully!",
195                     this.currentBot.getName()));
196             } catch (Exception e) {
197                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
198                     e.getMessage()));
199                 setDifficulty();
200             }
201         }
```

### 6.8.3.14 setIsDeleted()

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

Definition at line 201 of file BotDriver.java.

```
201         {
202             if(this.currentBot == null) {
203                 System.out.println("No current bot!");
204                 return;
205             }
206             if(Driver.inputBool("Do you want to delete the current bot?")) {
207                 this.currentBot.setIsDeleted(true);
208                 System.out.println(String.format("%s's state has changed to DELETED!",
209                     this.currentBot.getName()));
210             }
211             else {
212                 System.out.println(String.format("%s's state has not changed!", this.currentBot.getName()));
213             }
214         }
```

## 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.

```
36         {  
37     }
```

### 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.

```
153     {  
154         currentUser.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));  
155         botMap = new HashMap<String, UUID>();  
156         ArrayList<Pair<String, UUID> botList = ViewCtrl.domainCtrl.listBots();  
157         for(Pair<String, UUID> bot : botList) {  
158             botChooser.getItems().add(bot.first);  
159             botMap.put(bot.first, bot.second);  
160         }  
161     }
```

### 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.

```
168                                     {
169         ViewController.changeScene("template/UserView.fxml");
170     }
```

### 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.

```
177                                     {
178         String chosenBot = (String) botChooser.getValue();
179         if (chosenBot != null) {
180             Pair<JSONObject, String> result = ViewController.domainCtrl.getBot(botMap.get(chosenBot));
181             if (result.second != null) {
182                 switch (result.second) {
183                     case "ERR_INEXISTING_PLAYER":
184                         consultBotResult.setText("The bot doesn't exist!");
185                         break;
186                     default:
187                         consultBotResult.setText("Something went wrong, try again!");
188                         break;
189                 }
190             }
191             else {
192                 consultBotResult.setText("");
193                 name.setText(result.first.getString("name"));
194                 difficultyNumber.setText(Integer.toString(result.first.getInt("difficulty")));
195                 Pair<JSONObject, String> user =
196                 ViewController.domainCtrl.getUser(UUID.fromString(result.first.getString("creator_id")));
197                 creator.setText((user.first != null ? user.first.getString("name") : "Unknown"));
198             }
199         }
```

#### 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.

```
206      {
207          ViewCtrl.changeScene("template/ConfigView.fxml");
208      }
```

#### 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.

```
215      {
216          ViewCtrl.changeScene("template/GamesView.fxml");
217      }
```

#### 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.

```
224      {
225          ViewCtrl.changeScene("template/RankingView.fxml");
226      }
```



### 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.

```
233         {
234             ViewCtrl.changeScene("template/PlayView.fxml");
235         }
```

### 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.

```
242         {
243             ViewCtrl.changeScene("template/BotsCreateView.fxml");
244         }
```

### 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.

```
251         {
252             ViewCtrl.changeScene("template/BotsModifyView.fxml");
253         }
```

#### 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.

```
260                                     {
261     ViewCtrl.changeScene("template/BotsView.fxml");
262 }
```

#### 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.

```
269                                     {
270     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
        ButtonType.YES, ButtonType.NO);
271     confirm.showAndWait();
272
273     if (confirm.getResult() == ButtonType.YES) {
274         ViewCtrl.domainCtrl.logout();
275         ViewCtrl.changeScene("template/LoginView.fxml");
276     }
277 }
```

### 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.

```
34     {  
35     }
```

## 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.

```
147     {  
148         currentUserName.setText (ViewCtrl.domainCtrl.viewUser().getString("name"));  
149     }
```

### 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.

```
156     {  
157         ViewCtrl.changeScene ("template/UserView.fxml");  
158     }
```

### 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.

```
165      {  
166          ViewCtrl.changeScene("template/ConfigView.fxml");  
167      }
```

### 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.

```
174      {  
175          ViewCtrl.changeScene("template/GamesView.fxml");  
176      }
```

### 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.

```
183      {  
184          ViewCtrl.changeScene("template/RankingView.fxml");  
185      }
```

### 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.

```
192         {
193             ViewCtrl.changeScene("template/PlayView.fxml");
194         }
```

### 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.

```
201         {
202             ViewCtrl.changeScene("template/BotsView.fxml");
203         }
```

### 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.

```
210         {
211             ViewCtrl.changeScene("template/BotsModifyView.fxml");
212         }
```

### 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.

```
219                                     {
220         ViewCtrl.changeScene("template/BotsConsultView.fxml");
221     }
```

### 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.

```
228                                     {
229         difficultyNumber.setText(String.valueOf((int) difficultyLevel.getValue()));
230     }
```

### 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.

```

237     {
238         Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createBot(nbotname.getText(), (int)
difficultyLevel.getValue());
239         if (result.second != null) {
240             switch (result.second) {
241                 case "ERR_INVALID_NAME":
242                     createBotResult.setText("Bot name can't be empty!");
243                     break;
244                 case "ERR_INVALID_DIFFICULTY":
245                     createBotResult.setText("This is an invalid difficulty!");
246                     break;
247                 case "ERR_EXISTING_PLAYER":
248                     createBotResult.setText("The name is already taken!");
249                     break;
250                 default:
251                     createBotResult.setText("Something went wrong, try again!");
252                     break;
253             }
254         }
255         else {
256             nbotname.clear();
257             difficultyLevel.setValue(1);
258             showLevel();
259             createBotResult.setText("Success!");
260         }
261     }

```

### 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.

```

268     {
269         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
ButtonType.YES, ButtonType.NO);
270         confirm.showAndWait();
271
272         if (confirm.getResult() == ButtonType.YES) {
273             ViewCtrl.domainCtrl.logout();
274             ViewCtrl.changeScene("template/LoginView.fxml");
275         }
276     }

```

## 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.

```
37     {
38 }
```

## 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.

```
169     {
170         currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
171         botMap = new HashMap<String, UUID>();
172         ArrayList<Pair<String, UUID> botList = ViewCtrl.domainCtrl.listBots();
173         for(Pair<String, UUID> bot : botList) {
174             botChooser.getItems().add(bot.first);
175             botMap.put(bot.first, bot.second);
176         }
177     }
```

### 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.

```
184     {
185         ViewCtrl.changeScene("template/UserView.fxml");
186     }
```

### 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.

```
193                                     {
194     String chosenBot = (String) botChooser.getValue();
195     if (chosenBot != null) {
196         Pair<JSONObject, String> bot = ViewCtrl.domainCtrl.getBot(botMap.get(chosenBot));
197         if (bot.second == null) {
198             nbotname.setText(bot.first.getString("name"));
199             difficultyLevel.setValue((double) bot.first.getInt("difficulty"));
200             showLevel();
201         }
202     }
203 }
```

### 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.

```
210                                     {
211     ViewCtrl.changeScene("template/ConfigView.fxml");
212 }
```

### 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.

```
219         {
220             ViewCtrl.changeScene("template/GamesView.fxml");
221         }
```

### 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.

```
228         {
229             ViewCtrl.changeScene("template/RankingView.fxml");
230         }
```

### 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.

```
237         {
238             ViewCtrl.changeScene("template/PlayView.fxml");
239         }
```

### 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.

```
273         {
274             difficultyNumber.setText(String.valueOf((int) difficultyLevel.getValue()));
275         }
```

### 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.

```
282         {
283             Alert confirm = new Alert(AlertType.CONFIRMATION, "This bot will be modified. Are you sure?",
                ButtonType.YES, ButtonType.NO);
284             confirm.showAndWait();
285
286             if (confirm.getResult() == ButtonType.YES) {
287                 String chosenBot = (String) botChooser.getValue();
288                 if (chosenBot != null) {
289                     Pair<JSONObject, String> result = ViewCtrl.domainCtrl.modifyBot(botMap.get(chosenBot),
                        nbotname.getText(), (int) difficultyLevel.getValue());
290                     if (result.second != null) {
291                         switch (result.second) {
292                             case "ERR_INVALID_NAME":
293                                 modifyBotResult.setText("Bot name can't be empty!");
294                                 break;
295                             case "ERR_INVALID_DIFFICULTY":
296                                 modifyBotResult.setText("This is an invalid difficulty!");
297                                 break;
298                             case "ERR_EXISTING_PLAYER":
299                                 modifyBotResult.setText("The name is already taken!");
300                                 break;
301                             case "ERR_INEXISTING_PLAYER":
302                                 modifyBotResult.setText("The player doesn't exist!");
303                                 break;
304                             case "ERR_BOT_USED":
305                                 modifyBotResult.setText("This bot is already part of a game!");
```



```

306             break;
307         case "ERR_NOT_CREATOR":
308             modifyBotResult.setText("You are not the creator of this bot!");
309             break;
310         default:
311             modifyBotResult.setText("Something went wrong, try again!");
312             break;
313     }
314 }
315 else {
316     botChooser.getItems().clear();
317     initialize();
318     botChooser.getSelectionModel().select(nbotname.getText());
319     modifyBotResult.setText("Success!");
320 }
321 }
322 }
323 }

```

### 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.

```

330     {
331         Alert confirm = new Alert(AlertType.CONFIRMATION, "This bot will be deleted. Are you sure?",
332             ButtonType.YES, ButtonType.NO);
333         confirm.showAndWait();
334         if (confirm.getResult() == ButtonType.YES) {
335             String chosenBot = (String) botChooser.getValue();
336             if (chosenBot != null) {
337                 String result = ViewCtrl.domainCtrl.deleteBot(botMap.get(chosenBot));
338                 if (result != null) {
339                     switch (result) {
340                         case "ERR_INEXISTING_PLAYER":
341                             modifyBotResult.setText("The player doesn't exist!");
342                             break;
343                         case "ERR_BOT_USED":
344                             modifyBotResult.setText("This bot is already part of a game!");
345                             break;
346                         case "ERR_NOT_CREATOR":
347                             modifyBotResult.setText("You are not the creator of this bot!");
348                             break;
349                         default:
350                             modifyBotResult.setText("Something went wrong, try again!");
351                             break;
352                     }
353                 }
354             }
355             else {
356                 nbotname.clear();
357                 botChooser.getItems().clear();
358                 difficultyLevel.setValue(1);
359                 initialize();
360                 showLevel();
361                 modifyBotResult.setText("Success!");
362             }
363         }
364     }

```

#### 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.

```
371 {
372     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
373         ButtonType.YES, ButtonType.NO);
374     confirm.showAndWait();
375     if (confirm.getResult() == ButtonType.YES) {
376         ViewCtrl.domainCtrl.logout();
377         ViewCtrl.changeScene("template/LoginView.fxml");
378     }
379 }
```

### 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.

```
29         {
30     }
```



## 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.

```
111                                     {
112     currentUserName.setText (ViewCtrl.domainCtrl.viewUser().getString("name"));
113 }
```

### 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.

```
120                                     {
121     ViewCtrl.changeScene("template/UserView.fxml");
122 }
```

### 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.

```
129      {
130          ViewCtrl.changeScene("template/ConfigView.fxml");
131      }
```

### 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.

```
138      {
139          ViewCtrl.changeScene("template/GamesView.fxml");
140      }
```

### 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.

```
147      {
148          ViewCtrl.changeScene("template/RankingView.fxml");
149      }
```

### 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.

```
156                                     {
157     ViewCtrl.changeScene("template/PlayView.fxml");
158 }
```

### 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.

```
165                                     {
166     ViewCtrl.changeScene("template/BotsCreateView.fxml");
167 }
```

### 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.

```
174                                     {
175     ViewCtrl.changeScene("template/BotsModifyView.fxml");
176 }
```

### 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.

```
183                                     {
184     ViewController.changeScene("template/BotsConsultView.fxml");
185 }
```

### 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.

```
192                                     {
193     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
194                             ButtonType.YES, ButtonType.NO);
195     confirm.showAndWait();
196     if (confirm.getResult() == ButtonType.YES) {
197         ViewController.domainCtrl.logout();
198         ViewController.changeScene("template/LoginView.fxml");
199     }
200 }
```

## 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.

```
108         {
109             super("ERR_BOT_USED");
110         }
```

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.

```
34     {  
35     }
```

### 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.

```
171     {  
172         currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));  
173         ArrayList<String> configList = ViewCtrl.domainCtrl.listConfigurations().first;  
174         for(String configName : configList) configChooser.getItems().add(configName);  
175     }
```

### 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.

```
182         {  
183             ViewCtrl.changeScene("template/UserView.fxml");  
184         }
```

### 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.

```
191         {  
192             ViewCtrl.changeScene("template/ConfigCreateView.fxml");  
193         }
```

### 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.

```
200         {  
201             ViewCtrl.changeScene("template/ConfigModifyView.fxml");  
202         }
```

#### 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.

```
209                                     {
210         ViewCtrl.changeScene("template/ConfigView.fxml");
211     }
```

#### 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.

```
218                                     {
219         ViewCtrl.changeScene("template/BotsView.fxml");
220     }
```

### 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.

```
268         {  
269             ViewCtrl.changeScene("template/RankingView.fxml");  
270         }
```

#### 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.

```
277         {  
278             ViewCtrl.changeScene("template/PlayView.fxml");  
279         }
```

#### 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.

```
286         {  
287             String chosenConfig = (String) configChooser.getValue();  
288             if (chosenConfig != null) {  
289                 ViewCtrl.newWindow("template/ConsultInitialBoardView.fxml");  
290             }  
291         }
```

### 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.

```
298 {
299     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
300         ButtonType.YES, ButtonType.NO);
301     confirm.showAndWait();
302     if (confirm.getResult() == ButtonType.YES) {
303         ViewCtrl.domainCtrl.logout();
304         ViewCtrl.changeScene("template/LoginView.fxml");
305     }
306 }
```

## 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.

```
33         {
34     }
```

### 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.

```
160         {
161             currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
162             ViewCtrl.domainCtrl.createInitialBoard();
163     }
```

### 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.

```
170         {
171             ViewCtrl.changeScene("template/UserView.fxml");
172         }
```

### 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.

```
179         {
180             ViewCtrl.changeScene("template/BotsView.fxml");
181         }
```

### 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.

```
188         {
189             ViewCtrl.changeScene("template/GamesView.fxml");
190         }
```

### 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.

```
197         {
198             ViewCtrl.changeScene("template/RankingView.fxml");
199         }
```

### 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.

```
206         {
207             ViewCtrl.changeScene("template/PlayView.fxml");
208         }
```

### 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.

```
215         {
216             ViewCtrl.changeScene("template/ConfigView.fxml");
217         }
```



### 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.

```
224         {
225             ViewCtrl.changeScene("template/ConfigModifyView.fxml");
226         }
```

### 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.

```
233         {
234             ViewCtrl.changeScene("template/ConfigConsultView.fxml");
235         }
```

### 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.

```
242         {
243             ViewCtrl.newWindow("template/InitialBoardView.fxml");
244         }
```

### 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.

```

251     {
252         Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createConfiguration(nconfname.getText(),
canEatHorizontally.isSelected(), canEatVertically.isSelected(), canEatDiagonally.isSelected());
253         if (result.second != null) {
254             switch (result.second) {
255                 case "ERR_INVALID_NAME":
256                     createConfigResult.setText("Configuration name can't be empty!");
257                     break;
258                 case "ERR_EXISTING_CONFIGURATION":
259                     createConfigResult.setText("The configuration name is already taken!");
260                     break;
261                 case "ERR_INVALID_BOARD":
262                     createConfigResult.setText("The initial board is invalid!");
263                     break;
264                 case "ERR_INVALID_RULES":
265                     createConfigResult.setText("You must select at least one rule!");
266                     break;
267                 default:
268                     createConfigResult.setText("Something went wrong, try again!");
269                     break;
270             }
271         }
272         else {
273             nconfname.clear();
274             canEatHorizontally.setSelected(false);
275             canEatVertically.setSelected(false);
276             canEatDiagonally.setSelected(false);
277             initialize();
278             createConfigResult.setText("Success!");
279         }
280     }

```

### 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.

```

287     {
288         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
ButtonType.YES, ButtonType.NO);
289         confirm.showAndWait();
290
291         if (confirm.getResult() == ButtonType.YES) {
292             ViewCtrl.domainCtrl.logout();
293             ViewCtrl.changeScene("template/LogInView.fxml");
294         }
295     }

```

## 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.

```
36     {  
37 }
```

### 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.

```
174         {
175             currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
176             ArrayList<String> configList = ViewCtrl.domainCtrl.listConfigurations().first;
177             for(String configName : configList) configChooser.getItems().add(configName);
178         }
```

### 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.

```
185         {
186             ViewCtrl.changeScene("template/UserView.fxml");
187         }
```

### 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.

```
194         {
195             ViewCtrl.changeScene("template/BotsView.fxml");
196         }
```

### 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.

```

203                                     {
204         String chosenConfig = (String) configChooser.getValue();
205         if (chosenConfig != null) {
206             Pair<JSONObject, String> config = ViewCtrl.domainCtrl.getConfiguration(chosenConfig);
207             if (config.second == null) {
208                 ViewCtrl.domainCtrl.modifyInitialBoard(config.first.getString("name")); // Load onto
memory the chosen config Board
209                 canEatHorizontally.setSelected(config.first.getBoolean("can_eat_horizontally"));
210                 canEatVertically.setSelected(config.first.getBoolean("can_eat_vertically"));
211                 canEatDiagonally.setSelected(config.first.getBoolean("can_eat_diagonally"));
212             }
213         }
214     }

```

### 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.

```

221                                     {
222         ViewCtrl.changeScene("template/GamesView.fxml");
223     }

```

### 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.

```
230         {
231             ViewCtrl.changeScene("template/RankingView.fxml");
232         }
```

### 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.

```
239         {
240             ViewCtrl.changeScene("template/PlayView.fxml");
241         }
```

### 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.

```
248         {
249             ViewCtrl.changeScene("template/ConfigCreateView.fxml");
250         }
```

### 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.

```
257                                     {
258         ViewCtrl.changeScene("template/ConfigView.fxml");
259     }
```

### 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.

```
266                                     {
267         ViewCtrl.changeScene("template/ConfigConsultView.fxml");
268     }
```

### 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.

```
275                                     {
276         String chosenConfig = (String) configChooser.getValue();
277         if (chosenConfig != null) {
278             ViewCtrl.newWindow("template/ModifyInitialBoardView.fxml");
279         }
280     }
```

### 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.

```

332     {
333         Alert confirm = new Alert(AlertType.CONFIRMATION, "This configuration will be deleted. Are you
        sure?", ButtonType.YES, ButtonType.NO);
334         confirm.showAndWait();
335
336         if (confirm.getResult() == ButtonType.YES) {
337             String chosenConfig = (String) configChooser.getValue();
338             if (chosenConfig != null) {
339                 String result = ViewCtrl.domainCtrl.deleteConfiguration(chosenConfig);
340                 if (result != null) {
341                     switch (result) {
342                         case "ERR_INEXISTING_CONFIGURATION":
343                             modifyConfigResult.setText("This configuration doesn't exist!");
344                             break;
345                         case "ERR_NOT_CREATOR":
346                             modifyConfigResult.setText("You are not the creator of this
        configuration!");
347                             break;
348                         case "ERR_CONFIGURATION_USED":
349                             modifyConfigResult.setText("This configuration has been already used in a
        game!");
350                             break;
351                         default:
352                             modifyConfigResult.setText("Something went wrong, try again!");
353                             break;
354                     }
355                 }
356             } else {
357                 configChooser.getItems().clear();
358                 canEatHorizontally.setSelected(false);
359                 canEatVertically.setSelected(false);
360                 canEatDiagonally.setSelected(false);
361                 initialize();
362                 modifyConfigResult.setText("Success!");
363             }
364         }
365     }
366 }

```

**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.

```

373     {
374         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
        ButtonType.YES, ButtonType.NO);
375         confirm.showAndWait();
376
377         if (confirm.getResult() == ButtonType.YES) {
378             ViewCtrl.domainCtrl.logout();
379             ViewCtrl.changeScene("template/LoginView.fxml");
380         }
381     }

```

## 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

<i>name</i>	Name of the <a href="#">Configuration</a> .
<i>creatorID</i>	<a href="#">Player</a> ID of the <a href="#">Configuration</a> 's creator.
<i>canEatHorizontally</i>	Whether the pieces of a <a href="#">Game</a> can be eaten horizontally.
<i>canEatVertically</i>	Whether the pieces of a <a href="#">Game</a> can be eaten vertically.
<i>canEatDiagonally</i>	Whether the pieces of a <a href="#">Game</a> can be eaten diagonally.

Definition at line 58 of file Configuration.java.

```
59         {
60             this.name = name;
61             this.creatorID = creatorID;
62             this.canEatHorizontally = canEatHorizontally;
63             this.canEatVertically = canEatVertically;
64             this.canEatDiagonally = canEatDiagonally;
65         }
```

### 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

<i>configuration</i>	JSONObject representation of a <a href="#">Configuration</a> .
----------------------	--

Definition at line 74 of file Configuration.java.

```
74      {
75          this.name = configuration.getString("name");
76          this.creatorID = UUID.fromString(configuration.getString("creator_id"));
77          this.canEatHorizontally = configuration.getBoolean("can_eat_horizontally");
78          this.canEatVertically = configuration.getBoolean("can_eat_vertically");
79          this.canEatDiagonally = configuration.getBoolean("can_eat_diagonally");
80      }
```

## 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.

```
90         {
91             JSONObject configuration = new JSONObject();
92
93             configuration.put("name", this.name);
94             configuration.put("creator_id", this.creatorID.toString());
95             configuration.put("can_eat_horizontally", this.canEatHorizontally);
96             configuration.put("can_eat_vertically", this.canEatVertically);
97             configuration.put("can_eat_diagonally", this.canEatDiagonally);
98
99             return configuration;
100         }
```

**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.

```
108         {
109             return this.name;
110         }
```

**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**

<i>name</i>	Name of the <a href="#">Configuration</a> .
-------------	---

Definition at line 119 of file Configuration.java.

```
119                                     {
120         if (name.isBlank())
121             throw new InvalidNameException();
122
123         this.name = name;
124     }
```

**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.

```
132     {
133         return this.creatorID;
134     }
```

**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.

```
142     {
143         return this.canEatHorizontally;
144     }
```

### 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

<i>canEatHorizontally</i>	Whether the pieces of a <a href="#">Game</a> can be eaten horizontally.
---------------------------	---

Definition at line 153 of file Configuration.java.

```
153
154         if (canEatHorizontally == false && this.canEatVertically == false && this.canEatDiagonally ==
    false)
155             throw new InvalidRulesException();
156
157         this.canEatHorizontally = canEatHorizontally;
158     }
```

### 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.

```
166         {
167             return this.canEatVertically;
168         }
```

**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**

<i>canEatVertically</i>	Whether the pieces of a <a href="#">Game</a> can be eaten vertically.
-------------------------	---

Definition at line 177 of file Configuration.java.

```
177
178         if (this.canEatHorizontally == false && canEatVertically == false && this.canEatDiagonally ==
    false)
179             throw new InvalidRulesException();
180
181         this.canEatVertically = canEatVertically;
182     }
```

**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.

```
190
191         return this.canEatDiagonally;
192     }
```

### 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

<i>canEatDiagonally</i>	Whether the pieces of a <a href="#">Game</a> can be eaten diagonally.
-------------------------	---

Definition at line 201 of file Configuration.java.

```
201                                     {
202     if (this.canEatHorizontally == false && this.canEatVertically == false && canEatDiagonally ==
    false)
203         throw new InvalidRulesException();
204
205     this.canEatDiagonally = canEatDiagonally;
206 }
```

## 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 endpoint. 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 endpoint. 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.

```
22
23     new ConfigurationDriver().start();
24 }
```

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 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.*
- [Configuration modify](#) (String name, Boolean canEatHorizontally, Boolean canEatVertically, Boolean canEatDiagonally, [Board](#) initialBoard, UUID modifierID) 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.*
- 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 ConfigurationCtrl is created.

Definition at line 50 of file ConfigurationCtrl.java.

```
50         {
51             this.repositoryCtrl = new ConfigurationRepositoryCtrl();
52             this.gameRepositoryCtrl = new GameRepositoryCtrl();
53         }
```

## 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

<i>name</i>	name of a <a href="#">Configuration</a>
<i>canEatHorizontally</i>	Boolean that represents if you can capture pieces in a horizontal manner.
<i>canEatVertically</i>	Boolean that represents if you can capture pieces in a vertical manner.
<i>canEatDiagonally</i>	Boolean that represents if you can capture pieces in a diagonal manner.
<i>initialBoard</i>	Instance of a <a href="#">Board</a>
<i>creatorID</i>	UUID of the creator.

#### Returns

[Configuration](#).

Definition at line 69 of file ConfigurationCtrl.java.

```
71
72     {
73         if (name.isBlank())
74             throw new InvalidNameException();
75         if (this.repositoryCtrl.getConfiguration(name) != null)
76             throw new ExistingConfigurationException();
77         return this.save(name, canEatHorizontally, canEatVertically, canEatDiagonally, initialBoard,
78             creatorID);
79     }
```



### 6.19.3.2 modify()

```
Configuration domain.ConfigurationCtrl.modify (
    String name,
    Boolean canEatHorizontally,
    Boolean canEatVertically,
    Boolean canEatDiagonally,
    Board initialBoard,
    UUID modifierID ) 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

<i>name</i>	name of a <a href="#">Configuration</a>
<i>canEatHorizontally</i>	Boolean that represents if you can capture pieces in a horizontal manner.
<i>canEatVertically</i>	Boolean that represents if you can capture pieces in a vertical manner.
<i>canEatDiagonally</i>	Boolean that represents if you can capture pieces in a diagonal manner.
<i>initialBoard</i>	Instance of a <a href="#">Board</a>
<i>modifierID</i>	Modifier <a href="#">Player</a> UUID

#### Returns

[Configuration](#)

Definition at line 93 of file ConfigurationCtrl.java.

```
95
96         {
97             Configuration original = this.getConfiguration(name);
98             Board originalInitialBoard = this.getInitialBoard(name);
99             if (!original.getCreatorID().equals(modifierID))
100                 throw new NotCreatorException();
101
102             if (this.gameRepositoryCtrl.existsGameByConfigurationName(name))
103                 throw new ConfigurationUsedException();
104
105             if (canEatHorizontally != null)
106                 original.setCanEatHorizontally(canEatHorizontally);
107
108             if (canEatVertically != null)
109                 original.setCanEatVertically(canEatVertically);
110
111             if (canEatDiagonally != null)
112                 original.setCanEatDiagonally(canEatDiagonally);
113
114             if (initialBoard != null)
115                 originalInitialBoard = initialBoard;
116
117             return this.save(original.getName(), original.getCanEatHorizontally(),
original.getCanEatVertically(),
```

```

118         original.getCanEatDiagonally(), originalInitialBoard, original.getCreatorID());
119     }

```

### 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

<i>name</i>	name of a <a href="#">Configuration</a>
<i>canEatHorizontally</i>	Boolean that represents if you can capture pieces in a horizontal manner.
<i>canEatVertically</i>	Boolean that represents if you can capture pieces in a vertical manner.
<i>canEatDiagonally</i>	Boolean that represents if you can capture pieces in a diagonal manner.
<i>initialBoard</i>	Instance of a <a href="#">Board</a>
<i>creatorID</i>	UUID of the creator.

#### Returns

[Configuration](#)

Definition at line 133 of file ConfigurationCtrl.java.

```

135     {
136         if (initialBoard == null)
137             throw new InvalidBoardException();
138
139         if (canEatHorizontally == false && canEatVertically == false && canEatDiagonally == false)
140             throw new InvalidRulesException();
141
142         Configuration configuration = new Configuration(name, creatorID, canEatHorizontally,
143             canEatVertically,
144             canEatDiagonally);
145         this.repositoryCtrl.save(configuration.serialize(), initialBoard.serialize());
146         return configuration;
147     }

```

#### 6.19.3.4 delete()

```
void domain.ConfigurationCtrl.delete (
    String name,
    UUID deleterID ) throws NotCreatorException, ConfigurationUsedException, InexistingConfigurationException
```

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

<i>name</i>	Name of a <a href="#">Configuration</a>
<i>deleterID</i>	UUID of a <a href="#">Player</a> .

Definition at line 156 of file ConfigurationCtrl.java.

```
157                                     {
158         Configuration original = this.getConfiguration (name);
159
160         if (!original.getCreatorID().equals(deleterID))
161             throw new NotCreatorException();
162
163         if (this.gameRepositoryCtrl.existsGameByConfigurationName (name))
164             throw new ConfigurationUsedException();
165
166         this.repositoryCtrl.delete (name);
167     }
```

#### 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

<i>name</i>	Name of a <a href="#">Configuration</a>
-------------	---

**Returns**

[Configuration](#)

Definition at line 176 of file ConfigurationCtrl.java.

```
176                                                     {
177     JSONObject rawConfiguration = this.repositoryCtrl.getConfiguration(name);
178     if (rawConfiguration == null)
179         throw new InexistingConfigurationException();
180
181     return new Configuration(rawConfiguration);
182 }
```

**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**

<i>name</i>	Name of a <a href="#">Configuration</a> .
-------------	---

**Returns**

[Board](#)

Definition at line 191 of file ConfigurationCtrl.java.

```
191                                                     {
192     JSONObject rawInitialBoard = this.repositoryCtrl.getBoard(name);
193     if (rawInitialBoard == null)
194         throw new InexistingConfigurationException();
195
196     return new Board(rawInitialBoard);
197 }
```

**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.

```
205         {  
206             return this.repositoryCtrl.listConfigurations();  
207         }
```

## 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.

```
25         {
26             this.currentConfiguration = null;
27         }
```

### 6.20.3 Member Function Documentation

### 6.20.3.1 start()

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

Definition at line 31 of file ConfigurationDriver.java.

```
31         {
32             while (true) {
33                 this.mainMenu();
34             }
35         }
```

### 6.20.3.2 mainMenu()

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

Definition at line 37 of file ConfigurationDriver.java.

```
37         {
38             String title = (this.currentConfiguration != null
39                 ? String.format("Current: %s\n", this.currentConfiguration.getName())
40                 : null);
41             switch (Driver.menu(title, "Configuration Driver",
42                 new Pair<String, String>("1", "Create Configuration"),
43                 new Pair<String, String>("2", "Get name"),
44                 new Pair<String, String>("3", "Set name"),
45                 new Pair<String, String>("4", "Get creatorID"),
46                 new Pair<String, String>("5", "Get canEatHorizontally"),
47                 new Pair<String, String>("6", "Set canEatHorizontally"),
48                 new Pair<String, String>("7", "Get canEatVertically"),
49                 new Pair<String, String>("8", "Set canEatVertically"),
50                 new Pair<String, String>("9", "Get canEatDiagonally"),
51                 new Pair<String, String>("10", "Set canEatDiagonally"),
52                 new Pair<String, String>("11", "Serialize to JSON"),
53                 new Pair<String, String>("12", "Deserialize from JSON"))) {
54                 case "1":
55                     this.create();
56                     break;
57                 case "2":
58                     this.getName();
59                     break;
60                 case "3":
61                     this.setName();
62                     break;
63                 case "4":
64                     this.getCreatorID();
65                     break;
66                 case "5":
67                     this.getCanEatHorizontally();
68                     break;
69                 case "6":
70                     this.setCanEatHorizontally();
71                     break;
72                 case "7":
73                     this.getCanEatVertically();
74                     break;
75                 case "8":
76                     this.setCanEatVertically();
77                     break;
78                 case "9":
79                     this.getCanEatDiagonally();
80                     break;
81                 case "10":
82                     this.setCanEatDiagonally();
83                     break;
84                 case "11":
85                     this.serialize();
86                     break;
87                 case "12":
88                     this.deserialize();
89                     break;
90             }
91             Driver.pause();
92         }
```

### 6.20.3.3 create()

```
void test.driver.ConfigurationDriver.create ( ) [private]
```

Definition at line 94 of file ConfigurationDriver.java.

```

94         {
95             System.out.println(
96                 "Take into account that UUIDs will be randomly generated because typing them in will be a
             hassle.\n");
97             String name = Driver.input("Name?");
98             boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
99             boolean canEatVertically = Driver.inputBool("Can eat vertically?");
100            boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
101            try {
102                Configuration configuration = new Configuration("Default name", UUID.randomUUID(), true,
            true, true);
103                configuration.setName(name);
104                configuration.setCanEatHorizontally(canEatHorizontally);
105                configuration.setCanEatVertically(canEatVertically);
106                configuration.setCanEatDiagonally(canEatDiagonally);
107                this.currentConfiguration = configuration;
108                System.out.println(String.format("%s created successfully!",
            this.currentConfiguration.getName()));
109            } catch (Exception e) {
110                System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
            e.getMessage()));
111            }
112        }

```

### 6.20.3.4 getName()

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

Definition at line 114 of file ConfigurationDriver.java.

```

114        {
115            if (this.currentConfiguration == null) {
116                System.out.println("No current Configuration!");
117                return;
118            }
119
120            System.out.println(String.format("%s's name is: %s", this.currentConfiguration.getName(),
            this.currentConfiguration.getName()));
121        }
122    }

```

### 6.20.3.5 setName()

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

Definition at line 124 of file ConfigurationDriver.java.

```

124        {
125            if (this.currentConfiguration == null) {
126                System.out.println("No current Configuration!");
127                return;
128            }
129
130            try {
131                this.currentConfiguration.setName(Driver.input("Name?"));
132                System.out.println(String.format("%s's name changed successfully!",
            this.currentConfiguration.getName()));
133            } catch (Exception e) {
134                System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
            e.getMessage()));
135            }
136        }

```



### 6.20.3.6 getCreatorID()

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

Definition at line 138 of file ConfigurationDriver.java.

```
138         {
139             if (this.currentConfiguration == null) {
140                 System.out.println("No current Configuration!");
141                 return;
142             }
143
144             System.out.println(String.format("%s's creatorID is: %s", this.currentConfiguration.getName(),
145                 this.currentConfiguration.getCreatorID()));
146         }
```

### 6.20.3.7 getCanEatHorizontally()

```
void test.driver.ConfigurationDriver.getCanEatHorizontally ( ) [private]
```

Definition at line 148 of file ConfigurationDriver.java.

```
148         {
149             if (this.currentConfiguration == null) {
150                 System.out.println("No current Configuration!");
151                 return;
152             }
153
154             System.out.println(String.format("%s's canEatHorizontally is: %s",
155                 this.currentConfiguration.getName(),
156                 this.currentConfiguration.getCanEatHorizontally()));
157         }
```

### 6.20.3.8 setCanEatHorizontally()

```
void test.driver.ConfigurationDriver.setCanEatHorizontally ( ) [private]
```

Definition at line 158 of file ConfigurationDriver.java.

```
158         {
159             if (this.currentConfiguration == null) {
160                 System.out.println("No current Configuration!");
161                 return;
162             }
163
164             try {
165                 this.currentConfiguration.setCanEatHorizontally(Driver.inputBool("Can eat horizontally?"));
166                 System.out.println(String.format("%s's canEatHorizontally changed successfully!",
167                     this.currentConfiguration.getName()));
168             } catch (Exception e) {
169                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
170                     e.getMessage()));
171             }
```

### 6.20.3.9 getCanEatVertically()

```
void test.driver.ConfigurationDriver.getCanEatVertically ( ) [private]
```

Definition at line 173 of file ConfigurationDriver.java.

```
173         {
174             if (this.currentConfiguration == null) {
175                 System.out.println("No current Configuration!");
176                 return;
177             }
178
179             System.out.println(String.format("%s's canEatVertically is: %s",
180                 this.currentConfiguration.getName(),
181                 this.currentConfiguration.getCanEatVertically()));
182         }
```

### 6.20.3.10 setCanEatVertically()

```
void test.driver.ConfigurationDriver.setCanEatVertically ( ) [private]
```

Definition at line 183 of file ConfigurationDriver.java.

```
183         {
184             if (this.currentConfiguration == null) {
185                 System.out.println("No current Configuration!");
186                 return;
187             }
188
189             try {
190                 this.currentConfiguration.setCanEatVertically(Driver.inputBool("Can eat vertically?"));
191                 System.out.println(
192                     String.format("%s's canEatVertically changed successfully!",
193                         this.currentConfiguration.getName()));
194             } catch (Exception e) {
195                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
196                     e.getMessage()));
197             }
198         }
```

### 6.20.3.11 getCanEatDiagonally()

```
void test.driver.ConfigurationDriver.getCanEatDiagonally ( ) [private]
```

Definition at line 198 of file ConfigurationDriver.java.

```
198         {
199             if (this.currentConfiguration == null) {
200                 System.out.println("No current Configuration!");
201                 return;
202             }
203
204             System.out.println(String.format("%s's canEatDiagonally is: %s",
205                 this.currentConfiguration.getName(),
206                 this.currentConfiguration.getCanEatDiagonally()));
207         }
```

### 6.20.3.12 setCanEatDiagonally()

```
void test.driver.ConfigurationDriver.setCanEatDiagonally ( ) [private]
```

Definition at line 208 of file ConfigurationDriver.java.

```
208         {
209             if (this.currentConfiguration == null) {
210                 System.out.println("No current Configuration!");
211                 return;
212             }
213
214             try {
215                 this.currentConfiguration.setCanEatDiagonally(Driver.inputBool("Can eat diagonally?"));
216                 System.out.println(
217                     String.format("%s's canEatDiagonally changed successfully!",
218                         this.currentConfiguration.getName()));
219             } catch (Exception e) {
220                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
221                     e.getMessage()));
222             }
223         }
```

### 6.20.3.13 serialize()

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

Definition at line 223 of file ConfigurationDriver.java.

```
223         {
224             if (this.currentConfiguration == null) {
225                 System.out.println("No current Configuration!");
226                 return;
227             }
228
229             System.out.println(String.format("%s's serialized to JSON is: %s",
230                 this.currentConfiguration.getName(),
231                 this.currentConfiguration.serialize().toString(2)));
232         }
```

### 6.20.3.14 deserialize()

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

Definition at line 233 of file ConfigurationDriver.java.

```
233         {
234             if (this.currentConfiguration == null) {
235                 System.out.println("No current Configuration!");
236                 return;
237             }
238
239             System.out.println(this.currentConfiguration.serialize().toString(2));
240             this.currentConfiguration = new Configuration(this.currentConfiguration.serialize());
241             System.out.println(String.format("\n%s's deserialized from the above JSON successfully!\n",
242                 this.currentConfiguration.getName()));
243             System.out.println(String.format("name:\t\t%s", this.currentConfiguration.getName()));
244             System.out.println(String.format("creatorID:\t\t%s", this.currentConfiguration.getCreatorID()));
245             System.out.println(String.format("canEatHorizontally:\t%s",
246                 this.currentConfiguration.getCanEatHorizontally()));
247             System.out.println(String.format("canEatVertically:\t%s",
248                 this.currentConfiguration.getCanEatVertically()));
249             System.out.println(String.format("canEatDiagonally:\t%s",
250                 this.currentConfiguration.getCanEatDiagonally()));
251         }
```

## 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.

```
28         {  
29             super (RepositoryType.CONFIGURATION);  
30         }
```

## 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

<i>configuration</i>	Configuration to be saved.
<i>board</i>	Initial Board of the Configuration to be saved.

Definition at line 41 of file ConfigurationRepository.java.

```
41                                     {
42     String name = configuration.getString("name");
43     configuration.put("board", board);
44     this.createOrUpdate(name, configuration);
45 }
```

### 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**

<i>name</i>	Name of the Configuration to be deleted.
-------------	--

Definition at line 53 of file ConfigurationRepository.java.

```
53         {  
54             this.remove(name);  
55         }
```

**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**

<i>name</i>	Name of the Configuration to be gotten.
-------------	---

**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.

```
64     {  
65         JSONObject configuration = this.get(name);  
66         if (configuration == null)  
67             return null;  
68         configuration.remove("board");  
69         return configuration;  
70     }  
71 }
```

**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**

<i>name</i>	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.

```
80      {
81          JSONObject configuration = this.get(name);
82          if (configuration == null)
83              return null;
84
85          return configuration.getJSONObject("board");
86      }
```

**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.

```
94      {
95          return new ArrayList<String>(this.list().keySet());
96      }
```

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.

```
33         {  
34             this.repository = new ConfigurationRepository();  
35         }
```



### 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

<i>configuration</i>	Configuration to be saved.
<i>board</i>	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**

<i>name</i>	Name of the Configuration to be deleted.
-------------	--

Definition at line 56 of file ConfigurationRepositoryCtrl.java.

```
56         {  
57             this.repository.delete(name);  
58         }
```

**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**

<i>name</i>	Name of the Configuration to be gotten.
-------------	---

**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.

```
67         {  
68             return this.repository.getConfiguration(name);  
69         }
```

**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**

<i>name</i>	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.

```
78      {  
79          return this.repository.getBoard(name);  
80      }
```

**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.

```
88      {  
89          return this.repository.listConfigurations();  
90      }
```

**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`.

```
141 {  
142     super("ERR_CONFIGURATION_USED");  
143 }
```

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.

```
121         {
122     ViewCtrl.changeScene("template/UserView.fxml");
123     }
```

### 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.

```
130         {
131     ViewCtrl.changeScene("template/BotsView.fxml");
132     }
```

### 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.

```
139         {
140     ViewCtrl.changeScene("template/GamesView.fxml");
141     }
```

#### 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.

```
149         {
150             ViewCtrl.changeScene("template/RankingView.fxml");
151         }
```

#### 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.

```
158         {
159             ViewCtrl.changeScene("template/PlayView.fxml");
160         }
```

#### 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.

```
167         {
168             ViewCtrl.changeScene("template/ConfigCreateView.fxml");
169         }
```



### 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.

```
176         {
177         ViewCtrl.changeScene("template/ConfigModifyView.fxml");
178     }
```

### 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.

```
185         {
186         ViewCtrl.changeScene("template/ConfigConsultView.fxml");
187     }
```

### 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.

```
194         {
195         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
196             ButtonType.YES, ButtonType.NO);
197         confirm.showAndWait();
198         if (confirm.getResult() == ButtonType.YES) {
199             ViewCtrl.domainCtrl.logout();
200             ViewCtrl.changeScene("template/LogInView.fxml");
201         }
202     }
```

## 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`.

```
29         {  
30     }
```

## 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`.

```
371         {  
372             board = ViewCtrl.domainCtrl.viewBoard();  
373             render();  
374     }
```

### 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`.

```
381         {  
382             Stage currentWindow = (Stage) goToMenu.getScene().getWindow();  
383             currentWindow.close();  
384     }
```



### 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.

```
391         {
392             for (int i = 0; i < 8; i++) {
393                 char[] row = board.getString(String.format("row%d", i)).toCharArray();
394                 for (int j = 0; j < 8; j++) drawPiece(new Pair<Integer, Integer>(i, j), row[j]);
395             }
396         }
```

### 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.

```
403         {
404             Circle circle = getCircle(pos);
405             switch (pieceType) {
406                 case 'B':
407                     circle.setFill(Color.web("0xFFFFFF", 1.0));
408                     break;
409                 case 'N':
410                     circle.setFill(Color.web("0x000000", 1.0));
411                     break;
412                 case '?':
413                     circle.setFill(Color.web("0x34d399", 1.0));
414                     break;
415                 default:
416                     break;
417             }
418         }
```

### 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.

```
425                                     {
426         try {
427             Field field = this.getClass().getDeclaredField(String.format("f%s%c", pos.first + 1,
pos.second + 1));
428             field.setAccessible(true);
429             return (Circle) field.get(this);
430         } catch (Exception e) {
431             return new Circle();
432         }
433     }
```

## 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, canEatDiagonally and pieceType attributes are setted. The implicit maxDepth attribute is setted to the double of the entered difficulty.

#### Parameters

<i>difficulty</i>	<a href="#">Difficulty</a> for the chosen algorithm.
<i>canEatHorizontally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten horizontally.
<i>canEatVertically</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten vertically.
<i>canEatDiagonally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten diagonally.
<i>pieceType</i>	<a href="#">Player</a> that wants to be maximized.

Definition at line 57 of file Difficulty.java.

```
58                                     {
59     this.difficulty = difficulty;
60     this.canEatHorizontally = canEatHorizontally;
61     this.canEatVertically = canEatVertically;
62     this.canEatDiagonally = canEatDiagonally;
63     this.pieceType = pieceType;
64     this.maxDepth = difficulty * 2;
65 }
```

## 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.

```
75     {  
76         return this.difficulty;  
77     }
```

**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.

```
85     {  
86         return this.canEatHorizontally;  
87     }
```

**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.

```
95     {  
96         return this.canEatVertically;  
97     }
```

#### 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.

```
105 {  
106     return this.canEatDiagonally;  
107 }
```

#### 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.

```
115 {  
116     return this.pieceType;  
117 }
```

### 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.

```
125 {  
126     return this.maxDepth;  
127 }
```

### 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

<i>maxDepth</i>	Max depth for the heuristics of the chosen algorithm.
-----------------	---

Definition at line 135 of file Difficulty.java.

```
135 {  
136     this.maxDepth = maxDepth;  
137 }
```

### 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

<i>pieceType</i>	<a href="#">Player</a> to be inverted.
------------------	--

#### Returns

The opponent player.

Definition at line 146 of file Difficulty.java.

```
146                                     {  
147     return (pieceType == PieceType.PLAYER2 ? PieceType.PLAYER1 : PieceType.PLAYER2);  
148 }
```

### 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

<i>playingBoard</i>	Current playing <a href="#">Board</a> .
---------------------	---

### 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

<i>bot</i>	An instance of the <a href="#">Bot</a> Class
<i>configuration</i>	An instance of the <a href="#">Configuration</a> Class
<i>board</i>	An instance of the <a href="#">Board</a> Class
<i>myPieceType</i>	PieceType variable that represents a <a href="#">Player</a> in a <a href="#">Board</a>

## Returns

The best position for the bot is returned.

Definition at line 48 of file DifficultyCtrl.java.

```
49     {  
50         return this.getBestPosition(bot.getDifficulty(), configuration, board, myPieceType);  
51     }
```

### 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

<i>difficulty</i>	Integer that represents the level of the assisted mode.
<i>configuration</i>	An instance of the <a href="#">Configuration</a> Class
<i>board</i>	An instance of the <a href="#">Board</a> Class
<i>myPieceType</i>	PieceType variable that represents a <a href="#">Player</a> in a <a href="#">Board</a>

## Returns

The best position is returned.

Definition at line 64 of file DifficultyCtrl.java.

```

65         {
66             Pair<Integer, Integer> bestPosition = null;
67             boolean cH = configuration.getCanEatHorizontally();
68             boolean cV = configuration.getCanEatVertically();
69             boolean cD = configuration.getCanEatDiagonally();
70             PieceType[][] b = board.getBoard();
71
72             switch (difficulty) {
73                 case 1:
74                     bestPosition = new HardDifficulty(7, cH, cV, cD, myPieceType).place(b);
75                     break;
76                 case 2:
77                     bestPosition = new HardDifficulty(8, cH, cV, cD, myPieceType).place(b);
78                     break;
79                 case 3:
80                     bestPosition = new HardDifficulty(9, cH, cV, cD, myPieceType).place(b);
81                     break;
82                 case 4:
83                     bestPosition = new HardDifficulty(10, cH, cV, cD, myPieceType).place(b);
84                     break;
85                 case 5:
86                     EasyDifficulty ed5 = new EasyDifficulty(1, cH, cV, cD, myPieceType); ed5.setMaxDepth(1);
87                     bestPosition = ed5.place(b);
88                     break;
89                 case 6:
90                     bestPosition = new EasyDifficulty(1, cH, cV, cD, myPieceType).place(b);
91                     break;
92                 case 7:
93                     MediumDifficulty md7 = new MediumDifficulty(1, cH, cV, cD, myPieceType);
94                     md7.setMaxDepth(3);
95                     bestPosition = md7.place(b);
96                     break;
97                 case 8:
98                     bestPosition = new MediumDifficulty(2, cH, cV, cD, myPieceType).place(b);
99                     break;
100                 case 9:
101                     MediumDifficulty md9 = new MediumDifficulty(2, cH, cV, cD, myPieceType);
102                     md9.setMaxDepth(5);
103                     bestPosition = md9.place(b);
104                     break;
105                 case 10:
106                     bestPosition = new MediumDifficulty(3, cH, cV, cD, myPieceType).place(b);
107                     break;
108             }
109             return bestPosition;
110         }

```

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 canEatVertically, 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 canEatVertically, 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 knowledge.*
- [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.

```
88         {
89             this.playerCtrl = new PlayerCtrl();
90             this.configurationCtrl = new ConfigurationCtrl();
91             this.boardCtrl = new BoardCtrl();
92             this.gameCtrl = new GameCtrl();
93             this.rankingCtrl = new RankingCtrl();
94             this.difficultyCtrl = new DifficultyCtrl();
95             this.currentUser = null;
96             this.currentPlayer1 = null;
97             this.currentPlayer2 = null;
98             this.currentBoard = null;
99             this.currentConfiguration = null;
100             this.currentGame = null;
101         }
```

## 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.

```
110         {
111             this.logout();
112             System.exit(0);
113         }
```

### 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.

```
120      {  
121          this.exitGame();  
122          this.currentUser = null;  
123      }
```

### 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.

```
130      {  
131          this.currentPlayer1 = null;  
132          this.currentPlayer2 = null;  
133          this.currentBoard = null;  
134          this.currentConfiguration = null;  
135          this.currentGame = null;  
136      }
```

### 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**

<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of a <a href="#">User</a>
<i>confirmation</i>	Confirmation of the entered password

**Postcondition**

The user is saved and is returned in JSON format if no exceptions where triggered, else the exception will be returned in a string.

Definition at line 149 of file DomainCtrl.java.

```
149                                     {
150     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
151
152     try {
153         User user = this.playerCtrl.createUser(name, password, confirmation);
154         result.first = user.serialize();
155     } catch (Exception e) {
156         return new Pair<JSONObject, String>(null, e.getMessage());
157     }
158
159     return result;
160 }
```

**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**

<i>name</i>	Name of a <a href="#">Bot</a>
<i>difficulty</i>	<a href="#">Difficulty</a> of a <a href="#">Bot</a>

**Postcondition**

The created bot is saved and is returned in JSON format if no exceptions were triggered, else the exception will be returned in a string.

Definition at line 170 of file DomainCtrl.java.

```
170                                     {
171     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
172
173     try {
174         Bot bot = this.playerCtrl.createBot(name, difficulty, this.currentUser.getID());
175         result.first = bot.serialize();
176     } catch (Exception e) {
```



```

177         return new Pair<JSONObject, String>(null, e.getMessage());
178     }
179
180     return result;
181 }

```

### 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

<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of a <a href="#">User</a>

#### 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.

```

191     {
192         Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
193
194         try {
195             User user = this.playerCtrl.login(name, password);
196             this.currentUser = user;
197             result.first = user.serialize();
198         } catch (Exception e) {
199             return new Pair<JSONObject, String>(null, e.getMessage());
200         }
201
202         return result;
203     }

```

### 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

<i>userID</i>	UUID of a <a href="#">User</a>
---------------	--------------------------------

## 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.

```

211                                     {
212     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
213
214     try {
215         User user = this.playerCtrl.getUser(userID);
216         result.first = user.serialize();
217     } catch (Exception e) {
218         return new Pair<JSONObject, String>(null, e.getMessage());
219     }
220
221     return result;
222 }
```

## 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

<i>botID</i>	UUID of a <a href="#">Bot</a>
--------------	-------------------------------

## 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.

```

230                                     {
231     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
232
233     try {
234         Bot bot = this.playerCtrl.getBot(botID);
235         result.first = bot.serialize();
236     } catch (Exception e) {
237         return new Pair<JSONObject, String>(null, e.getMessage());
238     }
239
240     return result;
241 }
```

### 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

<i>playerID</i>	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.

```
249                                     {
250     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
251
252     try {
253         try {
254             User user = this.playerCtrl.getUser(playerID);
255             result.first = user.serialize();
256         } catch (Exception e) {
257             Bot bot = this.playerCtrl.getBot(playerID);
258             result.first = bot.serialize();
259         }
260     } catch (Exception e) {
261         return new Pair<JSONObject, String>(null, e.getMessage());
262     }
263
264     return result;
265 }
```

### 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.

```
272                                     {
273     return this.playerCtrl.listUsers();
274 }
```

### 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.

```
281                                     {
282         return this.playerCtrl.listBots();
283     }
```

### 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

<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of a <a href="#">User</a>
<i>confirmation</i>	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.

```
294                                     {
295         Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
296
297         try {
298             User user = this.playerCtrl.modifyUser(this.currentUser.getID(), name, password,
299             confirmation);
299             this.currentUser = user;
```

```

300         result.first = user.serialize();
301     } catch (Exception e) {
302         return new Pair<JSONObject, String>(null, e.getMessage());
303     }
304
305     return result;
306 }

```

### 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

<i>botID</i>	ID of a <a href="#">Bot</a>
<i>name</i>	Name
<i>difficulty</i>	The difficulty of a <a href="#">Bot</a>

#### 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.

```

319                                     {
320         Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
321
322         try {
323             Bot bot = this.playerCtrl.modifyBot(botID, name, difficulty, this.currentUser.getID());
324             result.first = bot.serialize();
325         } catch (Exception e) {
326             return new Pair<JSONObject, String>(null, e.getMessage());
327         }
328
329         return result;
330     }

```

### 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**

<i>password</i>	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.

```
339                                     {
340         String result = null;
341
342         try {
343             this.playerCtrl.deleteUser(this.currentUser.getID(), password);
344             this.logout();
345         } catch (Exception e) {
346             return e.getMessage();
347         }
348
349         return result;
350     }
```

**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**

<i>botID</i>	ID of a bot
--------------	-------------

Definition at line 359 of file DomainCtrl.java.

```
359                                     {
360         String result = null;
361
362         try {
363             this.playerCtrl.deleteBot(botID, this.currentUser.getID());
364         } catch (Exception e) {
365             return e.getMessage();
366         }
367
368         return result;
369     }
```

**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.

```

376         {
377             if (this.currentUser == null)
378                 return null;
379
380             return this.currentUser.serialize();
381         }

```

**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.

```

388         {
389             Pair<JSONObject, JSONObject> result = new Pair<JSONObject, JSONObject>(null, null);
390
391             if (this.currentPlayer1 instanceof User)
392                 result.first = ((User) this.currentPlayer1).serialize();
393             else if (this.currentPlayer1 instanceof Bot)
394                 result.first = ((Bot) this.currentPlayer1).serialize();
395
396             if (this.currentPlayer2 instanceof User)
397                 result.second = ((User) this.currentPlayer2).serialize();
398             else if (this.currentPlayer2 instanceof Bot)
399                 result.second = ((Bot) this.currentPlayer2).serialize();
400
401             return result;
402         }

```

**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

<i>name</i>	Name of the <a href="#">Configuration</a>
<i>canEatHorizontally</i>	Boolean that determines if you can capture pieces in a horizontal manner.
<i>canEatVertically</i>	Boolean that determines if you can capture pieces in a vertical manner.
<i>canEatDiagonally</i>	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.

```

416                                     {
417     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
418
419     try {
420         Configuration configuration = new Configuration(name, this.currentUser.getID(),
canEatHorizontally,
421             canEatVertically, canEatDiagonally);
422         configuration.setCanEatHorizontally(canEatHorizontally); // To ensure raising the rules
exception
423         configuration.setCanEatVertically(canEatVertically);
424         configuration.setCanEatDiagonally(canEatDiagonally);
425         this.boardCtrl.isValid(this.currentBoard, configuration);
426         configuration = this.configurationCtrl.create(name, canEatHorizontally, canEatVertically,
canEatDiagonally,
427             this.currentBoard, this.currentUser.getID());
428         result.first = configuration.serialize();
429     } catch (Exception e) {
430         return new Pair<JSONObject, String>(null, e.getMessage());
431     }
432
433     return result;
434 }
```

## 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.

```

441                                     {
442     String result = null;
443
444     try {
445         this.currentBoard = new Board();
446     } catch (Exception e) {
447         return e.getMessage();
448     }
449
450     return result;
451 }
```



**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**

<i>canEatHorizontally</i>	Boolean that determines if you can capture pieces in a horizontal manner.
<i>canEatVertically</i>	Boolean that determines if you can capture pieces in a vertical manner.
<i>canEatDiagonally</i>	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.

```
462                                     {
463     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
464
465     try {
466         Configuration configuration = new Configuration(name, this.currentUser.getID(),
467             canEatHorizontally,
468             canEatVertically, canEatDiagonally);
469         configuration.setCanEatHorizontally(canEatHorizontally); // To ensure raising the rules
470     exception configuration.setCanEatVertically(canEatVertically);
471         configuration.setCanEatDiagonally(canEatDiagonally);
472         this.boardCtrl.isValid(this.currentBoard, configuration);
473         configuration = this.configurationCtrl.modify(name, canEatHorizontally, canEatVertically,
474             canEatDiagonally,
475             this.currentBoard, this.currentUser.getID());
476         result.first = configuration.serialize();
477     } catch (Exception e) {
478         return new Pair<JSONObject, String>(null, e.getMessage());
479     }
480     return result;
481 }
```

**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**

<i>name</i>	Name of the <a href="#">Configuration</a>
-------------	---

**Postcondition**

The exception is returned in a String if any are triggered.

Definition at line 488 of file DomainCtrl.java.

```
488                                     {
489     String result = null;
490
491     try {
492         this.currentBoard = this.configurationCtrl.getInitialBoard(name);
493     } catch (Exception e) {
494         return e.getMessage();
495     }
496
497     return result;
498 }
```

**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**

<i>name</i>	Name of the <a href="#">Configuration</a>
-------------	---

**Postcondition**

The exception is returned in a String if any is triggered.

Definition at line 506 of file DomainCtrl.java.

```
506                                     {
507     String result = null;
508
509     try {
510         this.configurationCtrl.delete(name, this.currentUser.getID());
511     } catch (Exception e) {
512         return e.getMessage();
513     }
514
515     return result;
516 }
```

**6.28.3.23 getConfiguration()**

```
Pair<JSONObject, String> domain.DomainCtrl.getConfiguration (
    String name )
```

Returns the configuration identified by the name.

**Precondition**

*True*

**Parameters**

<i>name</i>	Name of the <a href="#">Configuration</a>
-------------	---

**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.

```
524                                     {
525     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
526
527     try {
528         Configuration configuration = this.configurationCtrl.getConfiguration(name);
529         result.first = configuration.serialize();
530     } catch (Exception e) {
531         return new Pair<JSONObject, String>(null, e.getMessage());
532     }
533
534     return result;
535 }
```

**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.

```
542                                     {
543     Pair<ArrayList<String>, String> result = new Pair<ArrayList<String>, String>(null, null);
544
545     try {
546         result.first = this.configurationCtrl.list();
547     } catch (Exception e) {
548         return new Pair<ArrayList<String>, String>(null, e.getMessage());
549     }
550
551     return result;
552 }
```

### 6.28.3.25 getInitialBoard()

```
Pair<JSONObject, String> domain.DomainCtrl.getInitialBoard (
    String name )
```

Get an Initial [Board](#) of a [Configuration](#).

#### Precondition

*True*

#### Parameters

<i>name</i>	Name of the <a href="#">Configuration</a>
-------------	---

#### 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.

```
560                                     {
561     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
562
563     try {
564         Board initialBoard = this.configurationCtrl.getInitialBoard(name);
565         result.first = initialBoard.serialize();
566     } catch (Exception e) {
567         return new Pair<JSONObject, String>(null, e.getMessage());
568     }
569
570     return result;
571 }
```

### 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.

```
578                                     {
579     if (this.currentConfiguration == null)
580         return null;
581
582     return this.currentConfiguration.serialize();
583 }
```

### 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

<i>player1ID</i>	UUID of Player1
<i>player2ID</i>	UUID of Player1
<i>configurationName</i>	Name of a <a href="#">Configuration</a>

#### 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.

```
595 {
596     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
597
598     try {
599         Game game = this.gameCtrl.create(player1ID, player2ID, configurationName,
        this.currentUser.getID());
600         result.first = game.serialize();
601     } catch (Exception e) {
602         return new Pair<JSONObject, String>(null, e.getMessage());
603     }
604
605     return result;
606 }
```

### 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.

```

613         {
614             Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
615
616             try {
617                 Game game = this.gameCtrl.save(this.currentGame, this.currentBoard,
        this.currentUser.getID());
618                 result.first = game.serialize();
619             } catch (Exception e) {
620                 return new Pair<JSONObject, String>(null, e.getMessage());
621             }
622
623             return result;
624         }

```

**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**

<i>name</i>	Name of a <a href="#">Game</a>
-------------	--------------------------------

**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.

```

632         {
633             Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
634
635             try {
636                 Game game = this.gameCtrl.getGame(name, this.currentUser.getID());
637                 result.first = game.serialize();
638             } catch (Exception e) {
639                 return new Pair<JSONObject, String>(null, e.getMessage());
640             }
641
642             return result;
643         }

```

**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**

<i>name</i>	Name of a <a href="#">Game</a>
-------------	--------------------------------

**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.

```

651                                     {
652     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
653
654     try {
655         Board playingBoard = this.gameCtrl.getPlayingBoard(name, this.currentUser.getID());
656         result.first = playingBoard.serialize();
657     } catch (Exception e) {
658         return new Pair<JSONObject, String>(null, e.getMessage());
659     }
660
661     return result;
662 }
```

**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.

```

669                                     {
670     Pair<ArrayList<String>, String> result = new Pair<ArrayList<String>, String>(null, null);
671
672     try {
673         result.first = this.gameCtrl.list(this.currentUser.getID());
674     } catch (Exception e) {
675         return new Pair<ArrayList<String>, String>(null, e.getMessage());
676     }
677
678     return result;
679 }
```

**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

<i>name</i>	Name of a <a href="#">Game</a>
-------------	--------------------------------

## 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.

```

687         {
688             Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
689
690             try {
691                 Game game = this.gameCtrl.getGame(name, this.currentUser.getID());
692                 Board board = this.gameCtrl.getPlayingBoard(name, this.currentUser.getID());
693                 Configuration configuration =
694                     this.configurationCtrl.getConfiguration(game.getConfigurationName());
695                 Player player1 = null;
696                 Player player2 = null;
697                 try { player1 = this.playerCtrl.getUser(game.getPlayer1ID()); } catch (Exception e) {
698                     player1 = this.playerCtrl.getBot(game.getPlayer1ID()); }
699                 try { player2 = this.playerCtrl.getUser(game.getPlayer2ID()); } catch (Exception e) {
700                     player2 = this.playerCtrl.getBot(game.getPlayer2ID()); }
701                 this.currentGame = game;
702                 this.currentBoard = board;
703                 this.currentConfiguration = configuration;
704                 this.currentPlayer1 = player1;
705                 this.currentPlayer2 = player2;
706                 result.first = game.serialize();
707             } catch (Exception e) {
708                 return new Pair<JSONObject, String>(null, e.getMessage());
709             }
710         }
711     }
712     return result;
713 }
```

## 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.

```

716         {
717             Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
718
719             try {
720                 this.currentGame = this.gameCtrl.play(this.currentGame);
721                 result.first = this.currentGame.serialize();
722             } catch (Exception e) {
723                 return new Pair<JSONObject, String>(null, e.getMessage());
724             }
725         }
726     }
727     return result;
728 }
```



### 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 <a href="#">User</a>
-------------	----------------------------------

#### 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.

```
735                                     {
736     Pair<JSONObject, String> result = new Pair<JSONObject, String>(null, null);
737
738     try {
739         this.currentGame = this.gameCtrl.surrender(this.currentGame, surrendeeID);
740         result.first = this.currentGame.serialize();
741         this.createEntries();
742     } catch (Exception e) {
743         return new Pair<JSONObject, String>(null, e.getMessage());
744     }
745
746     return result;
747 }
```

### 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.

```
754                                     {
755     if (this.currentGame == null)
756         return null;
757
758     return this.currentGame.serialize();
759 }
```

### 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 knowledge.

#### Precondition

*True*

#### Parameters

<i>pieceType</i>	String that represents a pieceType
------------------	------------------------------------

#### Postcondition

The corresponding PieceType will be returned.

Definition at line 769 of file DomainCtrl.java.

```
769                                     {
770         if (pieceType == null)
771             return null;
772
773         if (pieceType.equals(PieceType.PLAYER1.toString()))
774             return PieceType.PLAYER1;
775
776         if (pieceType.equals(PieceType.PLAYER2.toString()))
777             return PieceType.PLAYER2;
778
779         return null;
780     }
```

### 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

<i>difficulty</i>	Difficulty of the Bot
<i>myPieceType</i>	String that represents a pieceType

**Postcondition**

The best position will be returned.

Definition at line 790 of file DomainCtrl.java.

```
790
791     return this.difficultyCtrl.getBestPosition(difficulty, this.currentConfiguration, {
792         this.currentBoard,
793         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**

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

Definition at line 805 of file DomainCtrl.java.

```
805
806     this.currentBoard = this.boardCtrl.placePieceConfig(this.currentBoard, position, {
807         this.stringToPieceType(myPieceType));
808     return this.currentBoard.serialize();
809 }
```

**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

<i>position</i>	<a href="#">Pair&lt;Integer,Integer&gt;</a> that represents a position in a board.
-----------------	--

Definition at line 819 of file DomainCtrl.java.

```

819                                     {
820     this.currentBoard = this.boardCtrl.removePiece(this.currentBoard, position);
821     return this.currentBoard.serialize();
822 }
```

#### 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.

```

832                                     {
833     return this.boardCtrl.getNumPieces(this.currentBoard);
834 }
```

#### 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 its 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

<i>myPieceType</i>	PieceType variable that represents the player in a cell.
--------------------	--

Definition at line 851 of file DomainCtrl.java.

```

851                                     {
852         return this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
853             this.stringToPieceType(myPieceType));
854     }
```

## 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.

```

867                                     {
868         String result = null;
869
870         try {
871             this.boardCtrl.isValid(this.currentBoard, this.currentConfiguration);
872         } catch (Exception e) {
873             return e.getMessage();
874         }
875
876         return result;
877     }
```

## 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

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

Definition at line 908 of file DomainCtrl.java.

```

909         {
910             ArrayList<Pair<Integer, Integer>> validPos1 = new ArrayList<Pair<Integer, Integer>>();
911             ArrayList<Pair<Integer, Integer>> validPos2 = new ArrayList<Pair<Integer, Integer>>();
912
913             PieceType myPieceType = this.stringToPieceType(pieceType);
914
915             try {
916                 this.gameCtrl.checkPlaceRights(this.currentGame, playerID, myPieceType);
917
918                 validPos1 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
myPieceType);
919                 if (validPos1.isEmpty()) {
920                     validPos2 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
this.inversePieceType(myPieceType));
921                     if (validPos2.isEmpty())
922                         return this.finishGame();
923                     return this.nextTurn();
924                 }
925             }
926
927             Player player = (this.currentPlayer1.getID().equals(playerID) ? this.currentPlayer1 :
this.currentPlayer2);
928             if (player instanceof Bot) {
929                 Bot bot = ((Bot) player);
930                 position = this.difficultyCtrl.getBestPositionByBot(bot, this.currentConfiguration,
this.currentBoard,
931                     myPieceType);
932             }
933
934             if (!validPos1.contains(position))
935                 throw new InvalidPositionException();
936
937             this.currentBoard = this.boardCtrl.placePiece(this.currentBoard, this.currentConfiguration,
myPieceType,
938                 position);
939
940             validPos2 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
this.inversePieceType(myPieceType));
941             if (!validPos2.isEmpty())
942                 return this.nextTurn();
943
944             validPos1 = this.boardCtrl.validPositions(this.currentBoard, this.currentConfiguration,
myPieceType);
945             if (validPos1.isEmpty())
946                 return this.finishGame();
947
948             return this.currentTurn();
949         } catch (Exception e) {
950             return new Pair<Pair<JSONObject, String>, String>(null, e.getMessage());
951         }
952     }
953 }
```

### 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**

<i>PieceType</i>	PieceType variable that represents the player in a cell.
------------------	--

**Postcondition**

The opponent's PieceType is returned

Definition at line 961 of file DomainCtrl.java.

```

961                                     {
962         return (pieceType == PieceType.PLAYER2 ? PieceType.PLAYER1 : PieceType.PLAYER2);
963     }

```

**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.

```

970                                     {
971         Pair<Integer, Integer> numPieces = this.boardCtrl.getNumPieces(this.currentBoard);
972
973         try {
974             if (numPieces.first > numPieces.second) {
975                 this.currentGame = this.gameCtrl.finish(this.currentGame,
976 this.currentGame.getPlayer1ID());
977                 this.createEntries();
978                 return new Pair<Pair<JSONObject, String>, String>(
979                     new Pair<JSONObject, String>(this.currentBoard.serialize(), null),
980                     new FinishedGameException().getMessage());
981             } else if (numPieces.second > numPieces.first) {
982                 this.currentGame = this.gameCtrl.finish(this.currentGame,
983 this.currentGame.getPlayer2ID());
984                 this.createEntries();
985                 return new Pair<Pair<JSONObject, String>, String>(
986                     new Pair<JSONObject, String>(this.currentBoard.serialize(), null),
987                     new FinishedGameException().getMessage());
988             } else {
989                 this.currentGame = this.gameCtrl.finish(this.currentGame, null);
990                 this.createEntries();
991                 return new Pair<Pair<JSONObject, String>, String>(
992                     new Pair<JSONObject, String>(this.currentBoard.serialize(), null),
993                     new FinishedGameException().getMessage());
994             }
995         } catch (Exception e) {
996             return new Pair<Pair<JSONObject, String>, String>(null, e.getMessage());
997         }
998     }

```

### 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.

```
1003                                     {
1004         try {
1005             this.currentGame = this.gameCtrl.nextTurn(this.currentGame);
1006         } catch (Exception e) {
1007             return new Pair<Pair<JSONObject, String>, String>(null, e.getMessage());
1008         }
1009         return this.currentTurn();
1010     }
```

### 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.

```
1017                                     {
1018         return new Pair<Pair<JSONObject, String>, String>(
1019             new Pair<JSONObject, String>(this.currentBoard.serialize(),
1020             this.currentGame.getTurn().toString()),
1021             null);
1021     }
```



### 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.

```
1028         {  
1029             if (this.currentBoard == null)  
1030                 return null;  
1031  
1032             return this.currentBoard.serialize();  
1033         }
```

### 6.28.3.49 getRanking()

```
JSONObject domain.DomainCtrl.getRanking (  
    String name )
```

Returns the ranking identified by name.

#### Precondition

*True*

#### Parameters

<i>name</i>	of a <a href="#">Ranking</a>
-------------	------------------------------

#### Postcondition

The ranking identified by name is returned in JSON format.

Definition at line 1043 of file DomainCtrl.java.

```
1043         {  
1044             return this.rankingCtrl.getRanking(name).serialize();  
1045         }
```

### 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.

```
1052                                     {
1053     return thisrankingCtrl.listRankings();
1054 }
```

**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 JSONObject and Strings.

Definition at line 1061 of file DomainCtrl.java.

```
1061                                     {
1062     ArrayList<Pair<String, JSONObject> result = new ArrayList<Pair<String, JSONObject>();
1063     ArrayList<Pair<String, Entry> records = thisrankingCtrl.listRecords(this.currentUser.getID());
1064
1065     for (Pair<String, Entry> record : records)
1066         result.add(new Pair<String, JSONObject>(record.first, record.second.serialize()));
1067
1068     return result;
1069 }
```

**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.

```

1076         {
1077             ArrayList<String> rules = new ArrayList<String>();
1078
1079             if (this.currentConfiguration.getCanEatHorizontally())
1080                 rules.add("horizontally");
1081
1082             if (this.currentConfiguration.getCanEatVertically())
1083                 rules.add("vertically");
1084
1085             if (this.currentConfiguration.getCanEatDiagonally())
1086                 rules.add("diagonally");
1087
1088             String rankingName;
1089
1090             if (this.currentGame.getWinnerID() != null) {
1091                 Player winner = null;
1092                 Player loser = null;
1093
1094                 if (this.currentGame.getWinnerID().equals(this.currentPlayer1.getID())) {
1095                     winner = this.currentPlayer1;
1096                     loser = this.currentPlayer2;
1097                 } else {
1098                     winner = this.currentPlayer2;
1099                     loser = this.currentPlayer1;
1100                 }
1101
1102                 // Games won vs <Player>
1103                 rankingName = String.format("Games won vs %s", loser.getName());
1104                 this.rankingCtrl.createEntry(rankingName, winner.getID(), 1, RankingType.INCREMENTAL);
1105
1106                 // Games won with <Rules>
1107                 rankingName = String.format("Games won with %s rules", String.join(", ", rules));
1108                 this.rankingCtrl.createEntry(rankingName, winner.getID(), 1, RankingType.INCREMENTAL);
1109
1110                 // Games lost with <Rules>
1111                 rankingName = String.format("Games lost with %s rules", String.join(", ", rules));
1112                 this.rankingCtrl.createEntry(rankingName, loser.getID(), 1, RankingType.INCREMENTAL);
1113             } else {
1114                 // Games tied with <Rules>
1115                 rankingName = String.format("Games tied with %s rules", String.join(", ", rules));
1116                 this.rankingCtrl.createEntry(rankingName, this.currentPlayer1.getID(), 1,
RankingType.INCREMENTAL);
1117                 this.rankingCtrl.createEntry(rankingName, this.currentPlayer2.getID(), 1,
RankingType.INCREMENTAL);
1118             }
1119
1120             // Maximum pieces obtained in a game with <Rules>
1121             rankingName = String.format("Maximum pieces obtained in a game with %s rules", String.join(",
", rules));
1122             this.rankingCtrl.createEntry(rankingName, currentPlayer1.getID(),
this.currentBoard.getPiecesPlayer1(),
RankingType.UNIQUE);
1123             this.rankingCtrl.createEntry(rankingName, currentPlayer2.getID(),
this.currentBoard.getPiecesPlayer2(),
RankingType.UNIQUE);
1124             this.rankingCtrl.createEntry(rankingName, currentPlayer2.getID(),
this.currentBoard.getPiecesPlayer2(),
RankingType.UNIQUE);
1125             this.rankingCtrl.createEntry(rankingName, currentPlayer2.getID(),
RankingType.UNIQUE);
1126         }

```

**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**

<i>title</i>	A text inserted before the printed menu.
<i>name</i>	The name of the shown menu.
<i>options</i>	List of options to show.

**Returns**

The identifier of the selected option.

Definition at line 29 of file Driver.java.

```

29
30     String selected = new String();
31
32     if (name == null)
33         name = "Options";
34
35     do {
36         Driver.clear();
37         if (title != null)
38             System.out.println(title);
39         System.out.println(String.format("==== %s ====", name));
40         for (Pair<String, String> option : options)
41             System.out.println(String.format("[%s]\t%s", option.first, option.second));
42         System.out.println("[e]\tExit driver\n");
43         selected = Driver.input("What do you want to do?");
44         Driver.clear();
45
46         for (Pair<String, String> option : options)
47             if (selected.equals(option.first))
48                 return selected;
49
50     } while (!selected.equals("e") && !selected.equals("E"));
51
52     System.exit(0);
53
54     return null;
55 }
```

**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.

```

62     {
63         Driver.input("Press enter to continue");
64     }
```

### 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.

```
71         {  
72             System.out.print("\033\143");  
73         }
```

### 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

<i>prompt</i>	The text of the shown prompt.
---------------	-------------------------------

#### Returns

The entered value as String.

Definition at line 82 of file Driver.java.

```
82         {  
83             String in = new String();  
84             System.out.print(String.format("» %s: ", prompt));  
85             Scanner stdin = new Scanner(System.in);  
86  
87             try {  
88                 in = stdin.nextLine();  
89             } catch (Exception e) {  
90                 stdin.close();
```



```
91         }
92
93         return in;
94     }
```

### 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

<i>prompt</i>	The text of the shown prompt.
---------------	-------------------------------

#### Returns

The entered value as Integer.

Definition at line 103 of file Driver.java.

```
103         {
104             boolean trick = true; // Necessary or Java won't compile...
105             do {
106                 try {
107                     return Integer.parseInt(Driver.input(prompt));
108                 } catch (NumberFormatException e) {
109                     System.out.println("That is not an integer!");
110                 }
111             } while (trick);
112
113             return 0;
114         }
```

### 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

<i>prompt</i>	The text of the shown prompt.
---------------	-------------------------------

### Returns

The entered value as boolean.

Definition at line 123 of file Driver.java.

```
123                                     {
124     boolean trick = true; // Necessary or Java won't compile...
125     String in = new String();
126     do {
127         in = Driver.input(String.format("%s [y/n]", prompt));
128         if (in.toLowerCase().equals("yes") || in.toLowerCase().equals("y"))
129             return true;
130         else if (in.toLowerCase().equals("no") || in.toLowerCase().equals("n"))
131             return false;
132         System.out.println("That is not a yes or no!");
133     } while (trick);
134
135     return false;
136 }
```

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

- [Driver.java](#)

## 6.30 cmd.driver.easyDifficulty Class Reference

EasyDifficulty driver endpoint. 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 endpoint. 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.

```
22
23     new EasyDifficultyDriver().start();
24 }
```

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, canEatDiagonally and pieceType attributes are setted. The implicit maxDepth attribute is setted to the double of the entered difficulty.

#### Parameters

<i>difficulty</i>	<a href="#">Difficulty</a> for the Minimax algorithm.
<i>canEatHorizontally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten horizontally.
<i>canEatVertically</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten vertically.
<i>canEatDiagonally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten diagonally.
<i>pieceType</i>	<a href="#">Player</a> that wants to be maximized.

Definition at line 34 of file EasyDifficulty.java.

```
35                                     {
36         super(difficulty, canEatHorizontally, canEatVertically, canEatDiagonally, pieceType);
37     }
```

## 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**

<i>currentBoard</i>	Current playing <a href="#">Board</a> to get the heuristic evaluation from.
---------------------	---

**Returns**

The heuristic evaluation for the given [Board](#) state.

Definition at line 51 of file EasyDifficulty.java.

```

51         {
52             int player1 = currentBoard.getPiecesPlayer1();
53             int player2 = currentBoard.getPiecesPlayer2();
54
55             PieceType[][] board = currentBoard.getBoard();
56
57             // Check corners of the Board
58             if (board[0][0] == PieceType.PLAYER1) player1 += 50;
59             else if (board[0][0] == PieceType.PLAYER2) player2 += 50;
60
61             if (board[0][7] == PieceType.PLAYER1) player1 += 50;
62             else if (board[0][7] == PieceType.PLAYER2) player2 += 50;
63
64             if (board[7][0] == PieceType.PLAYER1) player1 += 50;
65             else if (board[7][0] == PieceType.PLAYER2) player2 += 50;
66
67             if (board[7][7] == PieceType.PLAYER1) player1 += 50;
68             else if (board[7][7] == PieceType.PLAYER2) player2 += 50;
69
70             // Check borders not next to corner
71             for (int k = 2; k < 6; ++k) {
72                 if (board[k][0] == PieceType.PLAYER1) player1 += 17;
73                 else if (board[k][0] == PieceType.PLAYER2) player2 += 17;
74
75                 if (board[k][7] == PieceType.PLAYER1) player1 += 17;
76                 else if (board[k][7] == PieceType.PLAYER2) player2 += 17;
77
78                 if (board[0][k] == PieceType.PLAYER1) player1 += 17;
79                 else if (board[0][k] == PieceType.PLAYER2) player2 += 17;
80
81                 if (board[7][k] == PieceType.PLAYER1) player1 += 17;
82                 else if (board[7][k] == PieceType.PLAYER2) player2 += 17;
83             }
84
85             // Check next to center of the Board
86             for (int i = 2; i < 6; ++i) {
87                 if (board[i][2] == PieceType.PLAYER1) player1 += 10;
88                 else if (board[i][2] == PieceType.PLAYER2) player2 += 10;
89
90                 if (board[i][5] == PieceType.PLAYER1) player1 += 10;
91                 else if (board[i][5] == PieceType.PLAYER2) player2 += 10;
92
93                 if (board[2][i] == PieceType.PLAYER1) player1 += 10;
94                 else if (board[2][i] == PieceType.PLAYER2) player2 += 10;
95
96                 if (board[5][i] == PieceType.PLAYER1) player1 += 10;
97                 else if (board[5][i] == PieceType.PLAYER2) player2 += 10;
98             }
99
100            // Check next to corners
101            for (int j = 0; j < 2; ++j) {
102                if (board[1][j] == PieceType.PLAYER1) player1 -= 25;
```

```

103         else if (board[1][j] == PieceType.PLAYER2) player2 -= 25;
104
105         if (board[1][7 - j] == PieceType.PLAYER1) player1 -= 25;
106         else if (board[1][7 - j] == PieceType.PLAYER2) player2 -= 25;
107
108         if (board[6][j] == PieceType.PLAYER1) player1 -= 25;
109         else if (board[6][j] == PieceType.PLAYER2) player2 -= 25;
110
111         if (board[6][7 - j] == PieceType.PLAYER1) player1 -= 25;
112         else if (board[6][7 - j] == PieceType.PLAYER2) player2 -= 25;
113     }
114
115     if (board[0][1] == PieceType.PLAYER1) player1 -= 25;
116     else if (board[0][1] == PieceType.PLAYER2) player2 -= 25;
117
118     if (board[7][1] == PieceType.PLAYER1) player1 -= 25;
119     else if (board[7][1] == PieceType.PLAYER2) player2 -= 25;
120
121     if (board[0][6] == PieceType.PLAYER1) player1 -= 25;
122     else if (board[0][6] == PieceType.PLAYER2) player2 -= 25;
123
124     if (board[7][6] == PieceType.PLAYER1) player1 -= 25;
125     else if (board[7][6] == PieceType.PLAYER2) player2 -= 25;
126
127     if (this.pieceType == PieceType.PLAYER1) return player1 - player2;
128     else return player2 - player1;
129 }

```

### 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

<i>currentBoard</i>	current <a href="#">Board</a> in the tree of possibilities.
<i>currentPieceType</i>	current turn in the tree of possibilities.
<i>depth</i>	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.

142

{

```

143     ArrayList<Pair<Integer, Integer>> validPositions = currentBoard.validPositions(currentPieceType,
144         this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);
145
146     if (validPositions.isEmpty() || depth == 0)
147         return this.evaluation(currentBoard);
148
149     // Maximizer
150     if (currentPieceType == this.pieceType) {
151         int max = Integer.MIN_VALUE, currentMax = 0;
152
153         for (Pair<Integer, Integer> position : validPositions) {
154             // Make a duplicate in order not to work with the same Board pointer!
155             Board board = new Board(currentBoard.getBoard());
156             board.placePiece(position, currentPieceType, this.canEatHorizontally,
this.canEatVertically,
157                 this.canEatDiagonally);
158
159             currentMax = this.minimax(board, EasyDifficulty.inversePieceType(currentPieceType),
depth - 1);
160             if (currentMax > max)
161                 max = currentMax;
162         }
163
164         return max;
165     }
166
167     // Minimizer
168     else {
169         Integer min = Integer.MAX_VALUE, currentMin = 0;
170
171         for (Pair<Integer, Integer> position : validPositions) {
172             // Make a duplicate in order not to work with the same Board pointer!
173             Board board = new Board(currentBoard.getBoard());
174             board.placePiece(position, currentPieceType, this.canEatHorizontally,
this.canEatVertically,
175                 this.canEatDiagonally);
176
177             currentMin = this.minimax(board, EasyDifficulty.inversePieceType(currentPieceType),
depth - 1);
178             if (currentMin < min)
179                 min = currentMin;
180         }
181
182         return min;
183     }
184 }

```

### 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

<i>playingBoard</i>	Current playing <a href="#">Board</a> .
---------------------	---

### 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.

```

195                                     {
196         Pair<Integer, Integer> bestPosition = null;
197
198         Board initialBoard = new Board(playingBoard);
199
200         ArrayList<Pair<Integer, Integer>> validPositions = initialBoard.validPositions(this.pieceType,
201             this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);
202
203         int max = Integer.MIN_VALUE, currentMax = 0;
204
205         for (Pair<Integer, Integer> position : validPositions) {
206             // Make a duplicate in order not to work with the same Board pointer!
207             Board board = new Board(initialBoard.getBoard());
208             board.placePiece(position, this.pieceType, this.canEatHorizontally, this.canEatVertically,
209                 this.canEatDiagonally);
210
211             currentMax = this.minimax(board, EasyDifficulty.inversePieceType(this.pieceType),
212                 this.maxDepth - 1);
213             if (currentMax > max) {
214                 max = currentMax;
215                 bestPosition = position;
216             }
217         }
218         return bestPosition;
219     }

```

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.

```
33         {
34             this.currentEasyDifficulty = null;
35             this.fixtureRepository = new FixtureRepository();
36         }
```

### 6.32.3 Member Function Documentation

### 6.32.3.1 start()

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

Definition at line 40 of file EasyDifficultyDriver.java.

```
40         {
41             while (true) {
42                 this.mainMenu();
43             }
44         }
```

### 6.32.3.2 mainMenu()

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

Definition at line 46 of file EasyDifficultyDriver.java.

```
46         {
47             String title = null;
48             if (this.currentEasyDifficulty != null)
49                 title = String.format("Current maximum depth: %s\n",
this.currentEasyDifficulty.getMaxDepth());
50             if (this.currentBoard != null)
51                 title += String.format("Current Board: %s\n", this.nameCurrentBoard);
52
53             switch (Driver.menu(title, "EasyDifficulty (Minimax) Driver",
54                 new Pair<String, String>("1", "Create EasyDifficulty"),
55                 new Pair<String, String>("2", "Get difficulty"),
56                 new Pair<String, String>("3", "Get canEatHorizontally"),
57                 new Pair<String, String>("4", "Get canEatVertically"),
58                 new Pair<String, String>("5", "Get canEatDiagonally"),
59                 new Pair<String, String>("6", "Get pieceType"),
60                 new Pair<String, String>("7", "Get maxDepth"),
61                 new Pair<String, String>("8", "Set maxDepth"),
62                 new Pair<String, String>("9", "Load Board"),
63                 new Pair<String, String>("10", "Print Current Board"),
64                 new Pair<String, String>("11", "Get next best position"))) {
65             case "1":
66                 this.create();
67                 break;
68             case "2":
69                 this.getDifficulty();
70                 break;
71             case "3":
72                 this.getCanEatHorizontally();
73                 break;
74             case "4":
75                 this.getCanEatVertically();
76                 break;
77             case "5":
78                 this.getCanEatDiagonally();
79                 break;
80             case "6":
81                 this.getPieceType();
82                 break;
83             case "7":
84                 this.getMaxDepth();
85                 break;
86             case "8":
87                 this.setMaxDepth();
88                 break;
89             case "9":
90                 this.loadBoard();
91                 break;
92             case "10":
93                 this.printCurrentBoard();
94                 break;
95             case "11":
96                 this.getNextBestPosition();
97                 break;
98             }
99             Driver.pause();
100         }
```

### 6.32.3.3 create()

```
void test.driver.EasyDifficultyDriver.create ( ) [private]
```

Definition at line 102 of file EasyDifficultyDriver.java.

```

102     {
103         System.out.println(
104             "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");
105
106         Integer difficulty = Driver.inputInt("Difficulty (positive)?");
107         Boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
108         Boolean canEatVertically = Driver.inputBool("Can eat vertically?");
109         Boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
110         PieceType pieceType = null;
111
112         switch (Driver.menu(null, "Select Bot pieces",
113             new Pair<String, String>("1", "PLAYER 1 pieces"),
114             new Pair<String, String>("2", "PLAYER 2 pieces"))) {
115             case "1":
116                 pieceType = PieceType.PLAYER1;
117                 break;
118             case "2":
119                 pieceType = PieceType.PLAYER2;
120                 break;
121         }
122
123         this.currentEasyDifficulty = new EasyDifficulty(difficulty, canEatHorizontally,
canEatVertically,
canEatDiagonally, pieceType);
124
125         System.out.println(String.format("EasyDifficulty with a maximum depth of %s created
successfully!",
126             this.currentEasyDifficulty.getMaxDepth()));
127     }
128 }
```

### 6.32.3.4 getDifficulty()

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

Definition at line 130 of file EasyDifficultyDriver.java.

```

130     {
131         if (this.currentEasyDifficulty == null) {
132             System.out.println("No current EasyDifficulty!");
133             return;
134         }
135
136         System.out.println(
137             String.format("EasyDifficulty's difficulty is: %s",
this.currentEasyDifficulty.getDifficulty()));
138     }
```

### 6.32.3.5 getCanEatHorizontally()

```
void test.driver.EasyDifficultyDriver.getCanEatHorizontally ( ) [private]
```

Definition at line 140 of file EasyDifficultyDriver.java.

```

140     {
141         if (this.currentEasyDifficulty == null) {
142             System.out.println("No current EasyDifficulty!");
143             return;
144         }
145
146         System.out.println(String.format("EasyDifficulty's canEatHorizontally is: %s",
147             this.currentEasyDifficulty.getCanEatHorizontally()));
148     }
```

### 6.32.3.6 getCanEatVertically()

```
void test.driver.EasyDifficultyDriver.getCanEatVertically ( ) [private]
```

Definition at line 150 of file EasyDifficultyDriver.java.

```
150         {
151             if (this.currentEasyDifficulty == null) {
152                 System.out.println("No current EasyDifficulty!");
153                 return;
154             }
155
156             System.out.println(String.format("EasyDifficulty's canEatVertically is: %s",
157                 this.currentEasyDifficulty.getCanEatVertically()));
158         }
```

### 6.32.3.7 getCanEatDiagonally()

```
void test.driver.EasyDifficultyDriver.getCanEatDiagonally ( ) [private]
```

Definition at line 160 of file EasyDifficultyDriver.java.

```
160         {
161             if (this.currentEasyDifficulty == null) {
162                 System.out.println("No current EasyDifficulty!");
163                 return;
164             }
165
166             System.out.println(String.format("EasyDifficulty's canEatDiagonally is: %s",
167                 this.currentEasyDifficulty.getCanEatDiagonally()));
168         }
```

### 6.32.3.8 getPieceType()

```
void test.driver.EasyDifficultyDriver.getPieceType ( ) [private]
```

Definition at line 170 of file EasyDifficultyDriver.java.

```
170         {
171             if (this.currentEasyDifficulty == null) {
172                 System.out.println("No current EasyDifficulty!");
173                 return;
174             }
175
176             System.out
177                 .println(String.format("EasyDifficulty's pieceType is: %s",
178                 this.currentEasyDifficulty.getPieceType()));
179         }
```

### 6.32.3.9 getMaxDepth()

```
void test.driver.EasyDifficultyDriver.getMaxDepth ( ) [private]
```

Definition at line 180 of file EasyDifficultyDriver.java.

```
180         {
181             if (this.currentEasyDifficulty == null) {
182                 System.out.println("No current EasyDifficulty!");
183                 return;
184             }
185
186             System.out.println(String.format("EasyDifficulty's maxDepth is: %s",
187                 this.currentEasyDifficulty.getMaxDepth()));
188         }
```

**6.32.3.10 setMaxDepth()**

```
void test.driver.EasyDifficultyDriver.setMaxDepth ( ) [private]
```

Definition at line 189 of file EasyDifficultyDriver.java.

```
189         {
190             if (this.currentEasyDifficulty == null) {
191                 System.out.println("No current EasyDifficulty!");
192                 return;
193             }
194
195             System.out.println(
196                 "Take into account that minimax with higher depths requires more time to execute. A
value of 5 is reasonable.\n");
197
198             this.currentEasyDifficulty.setMaxDepth(Driver.inputInt("Maximum depth (positive)?"));
199             System.out.println("EasyDifficulty's maxDepth changed successfully!");
200         }
```

**6.32.3.11 loadBoard()**

```
void test.driver.EasyDifficultyDriver.loadBoard ( ) [private]
```

Definition at line 202 of file EasyDifficultyDriver.java.

```
202         {
203             if (this.currentEasyDifficulty == null) {
204                 System.out.println("No current EasyDifficulty!");
205                 return;
206             }
207
208             Pair<String, Board> selectedBoard = this.listBoardFixtures();
209
210             this.nameCurrentBoard = selectedBoard.first;
211             this.currentBoard = selectedBoard.second;
212
213             System.out.println(String.format("Board %s loaded successfully!\n", this.nameCurrentBoard));
214             this.printBoard(this.currentBoard);
215         }
```

**6.32.3.12 printCurrentBoard()**

```
void test.driver.EasyDifficultyDriver.printCurrentBoard ( ) [private]
```

Definition at line 217 of file EasyDifficultyDriver.java.

```
217         {
218             if (this.currentEasyDifficulty == null) {
219                 System.out.println("No current EasyDifficulty!");
220                 return;
221             }
222
223             if (this.currentBoard == null) {
224                 System.out.println("No current Board!");
225                 return;
226             }
227
228             System.out.println(String.format("Board %s printed successfully!\n", this.nameCurrentBoard));
229             this.printBoard(this.currentBoard);
230         }
```

### 6.32.3.13 getNextBestPosition()

```
void test.driver.EasyDifficultyDriver.getNextBestPosition ( ) [private]
```

Definition at line 232 of file EasyDifficultyDriver.java.

```

232         {
233             if (this.currentEasyDifficulty == null) {
234                 System.out.println("No current EasyDifficulty!");
235                 return;
236             }
237
238             if (this.currentBoard == null) {
239                 System.out.println("No current Board!");
240                 return;
241             }
242
243             System.out.println("Take into account that the state of the current Board won't be globally
modified.\n");
244
245             this.printBoard(this.currentBoard);
246
247             long startTime = System.currentTimeMillis();
248             Pair<Integer, Integer> nextBestPosition =
this.currentEasyDifficulty.place(this.currentBoard.getBoard());
249             long durationTime = System.currentTimeMillis() - startTime;
250
251             Board tempBoard = new Board(this.currentBoard.getBoard());
252
253             if (nextBestPosition != null) {
254                 tempBoard.placePiece(nextBestPosition, this.currentEasyDifficulty.getPieceType(),
255                                     this.currentEasyDifficulty.getCanEatHorizontally(),
256                                     this.currentEasyDifficulty.getCanEatVertically(),
257                                     this.currentEasyDifficulty.getCanEatDiagonally());
258                 System.out.println(
String.format("The best position calculated in %s ms is %s\n", durationTime,
nextBestPosition));
259                 System.out.println("The addition of the piece would look like this:\n");
260                 this.printBoard(tempBoard);
261             } else {
262                 System.out.println("There isn't any possible position left to place a piece on.");
263             }
264         }

```

### 6.32.3.14 listBoardFixtures()

```
Pair<String, Board> test.driver.EasyDifficultyDriver.listBoardFixtures ( ) [private]
```

Definition at line 266 of file EasyDifficultyDriver.java.

```

266         {
267             Integer selectedBoard = -1;
268             ArrayList<String> listBoards = this.fixtureRepository.listFiles();
269
270             while (selectedBoard < 0 || selectedBoard >= listBoards.size()) {
271                 Driver.clear();
272                 System.out.println("==== Available Boards ====\n");
273
274                 for (Integer i = 0; i < listBoards.size(); ++i)
275                     System.out.println(String.format("[%d]\t%s", i, listBoards.get(i)));
276                 System.out.println("");
277
278                 selectedBoard = Driver.inputInt("What Board would you like to load?");
279             }
280
281             Driver.clear();
282
283             return new Pair<String, Board>(listBoards.get(selectedBoard),
284                                           new Board(this.fixtureRepository.boardFileToJSON(listBoards.get(selectedBoard))));
285         }

```

### 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)
+ " "
+ row.charAt(7) + " " );
296         }
297         System.out.println("\n");
298     }
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             boardCodified.add(operational);
317         }
318     }
319     return boardCodified;
320 }
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 entripoint. By Alex Rodriguez.

#### Static Public Member Functions

- static void `main` (String[] args)  
*JUnit Entry tests main function. Calls the JUnitCore main entripoint and runs the Entry unitary tests.*

#### 6.33.1 Detailed Description

JUnit Entry tests entripoint. 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 entryptoint and runs the Entry unitary tests.

#### Precondition

*True.*

#### Postcondition

The JUnit Entry tests have started.

Definition at line 24 of file entry.java.

```
24                                     {
25     JUnitCore.main(new EntryJUnit().getClass().getName());
26 }
```

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

<i>playerID</i>	ID of the player about to be created
<i>value</i>	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**

<code>entry</code>	JSONObject which contains information to create an <a href="#">Entry</a>
--------------------	--

Definition at line 41 of file Entry.java.

```
41         {
42             this.playerID = UUID.fromString(entry.getString("player_id"));
43             this.value = entry.getInt("value");
44         }
```

### 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.

```
51         {
52             JSONObject entry = new JSONObject();
53
54             entry.put("player_id", this.playerID.toString());
55             entry.put("value", this.value);
56
57             return entry;
58         }
```

### 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.

```
65      {  
66          return this.playerID;  
67      }
```

### 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.

```
74      {  
75          return this.value;  
76      }
```

### 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**

<i>newPlayerID</i>	New ID of the player
--------------------	----------------------

Definition at line 83 of file Entry.java.

```
83                                     {
84         this.playerID = newPlayerID;
85     }
```

**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**

<i>newValue</i>	New value of the player
-----------------	-------------------------

Definition at line 92 of file Entry.java.

```
92                                     {
93         this.value = newValue;
94     }
```

**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.

```
25     {
26         UUID playerID = UUID.randomUUID();
27         Entry e = new Entry(playerID, 25);
28         assertEquals("Entry failed because", playerID, e.getPlayerID());
29         assertEquals("Entry failed because", 25, e.getValue());
30     }
```

### 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(),
46             jobj.getString("player_id"));
47         assertEquals("serialize failed because", 100, jobj.getInt("value"));
48     }
```

### 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.

```
63         {
64             UUID playerId = UUID.randomUUID();
65             Entry e = new Entry(UUID.randomUUID(), 150);
66             e.setPlayerID(playerID);
67             assertEquals("setPlayerID failed because", playerId, e.getPlayerID());
68         }
```

### 6.35.2.7 setValue()

```
void test.unitary.EntryJUnit.setValue ( )
```

Definition at line 71 of file EntryJUnit.java.

```
71         {
72             Entry e = new Entry(UUID.randomUUID(), 180);
73             e.setValue(150);
74             assertEquals("setValue failed because", 150, e.getValue());
75         }
```

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.

```
130                                     {  
131         super ("ERR_EXISTING_CONFIGURATION");  
132     }
```

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.

```
20 {  
21     super("ERR_EXISTING_PLAYER");  
22 }
```

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.

```
240 {  
241     super("ERR_FINISHED_GAME");  
242 }
```

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`.

```
30         {  
31             super (RepositoryType.FIXTURE);  
32         }
```

### 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.

```
42     {  
43         try {  
44             ArrayList<String> list = new  
45                 ArrayList<String>(Files.walk(Paths.get(this.path)).filter(Files::isRegularFile)  
46                     .map(file -> file.toString()).collect(Collectors.toList()));  
47             return list;  
48         } catch (Exception e) {  
49             e.printStackTrace();  
50         }  
51         return new ArrayList<String>();  
52     }
```

### 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

<i>path</i>	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**

<i>path</i>	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 surrendeelD) throws NotPlayerException, FinishedGameException, NotStartedGameException  
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, NotPlayerPieceException, 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 implicit 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**

<i>name</i>	Name of the <a href="#">Game</a> .
<i>player1ID</i>	First player ID of the <a href="#">Game</a> .
<i>player2ID</i>	Second player ID of the <a href="#">Game</a> .
<i>configurationName</i>	Name of the <a href="#">Configuration</a> used to create the <a href="#">Game</a> .
<i>creatorID</i>	<a href="#">Player</a> ID of the <a href="#">Game</a> 's creator.

Definition at line 88 of file Game.java.

```
88                                     {
89     this.name = name;
90     this.player1ID = player1ID;
91     this.player2ID = player2ID;
92     this.configurationName = configurationName;
93     this.turn = PieceType.PLAYER1;
94     this.state = GameState.NOT_STARTED;
95     this.winnerID = null;
96     this.creatorID = creatorID;
97     this.createdAt = LocalDateTime.now();
98 }
```

**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 implicit name, player1ID, player2ID, configurationName and creatorID attributes are set. The current turn is set to PLAYER1, the state to NOT\_STARTED, the winnerID to null and the createdAt to the current timestamp.

## Parameters

<i>game</i>	JSONObject representation of a <a href="#">Game</a> .
-------------	---

Definition at line 107 of file Game.java.

```

107         {
108             this.name = game.getString("name");
109             this.player1ID = UUID.fromString(game.getString("player1_id"));
110             this.player2ID = UUID.fromString(game.getString("player2_id"));
111             this.configurationName = game.getString("configuration_name");
112             this.turn = game.getEnum(PieceType.class, "turn");
113             this.state = game.getEnum(GameState.class, "state");
114
115             this.winnerID = null;
116             String winnerID = game.optString("winner_id", null);
117             if (winnerID != null)
118                 this.winnerID = UUID.fromString(winnerID);
119
120             this.creatorID = UUID.fromString(game.getString("creator_id"));
121             this.createdAt = LocalDateTime.parse(game.getString("created_at"));
122         }

```

### 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.

```

132         {
133             JSONObject game = new JSONObject();
134
135             game.put("name", this.name);
136             game.put("player1_id", this.player1ID.toString());
137             game.put("player2_id", this.player2ID.toString());
138             game.put("configuration_name", this.configurationName);
139             game.put("turn", this.turn);
140             game.put("state", this.state);
141
142             if (this.winnerID != null)
143                 game.put("winner_id", this.winnerID.toString());
144             else
145                 game.put("winner_id", JSONObject.NULL);
146
147             game.put("creator_id", this.creatorID.toString());
148             game.put("created_at", this.createdAt.toString());
149
150             return game;
151         }

```

### 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.

```
159 {
160     return this.name;
161 }
```

### 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

<i>name</i>	Name of the <a href="#">Game</a> .
-------------	------------------------------------

Definition at line 170 of file Game.java.

```
170 {
171     if (name.isBlank())
172         throw new InvalidNameException();
173
174     this.name = name;
175 }
```

#### 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.

```
183 {  
184     return this.player1ID;  
185 }
```

#### 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.

```
193 {  
194     return this.player2ID;  
195 }
```

#### 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.

```
203                                     {
204     return this.configurationName;
205 }
```

#### 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 InvalidNameException is thrown.

##### Parameters

<i>configurationName</i>	Name of the <a href="#">Configuration</a> used to create the <a href="#">Game</a> .
--------------------------	---

Definition at line 214 of file Game.java.

```
214                                     {
215     if (configurationName.isBlank())
216         throw new InvalidConfigurationException();
217     this.configurationName = configurationName;
218 }
219 }
```

### 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.

```
227     {  
228         return this.turn;  
229     }
```

### 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

<i>turn</i>	Current turn of the <a href="#">Game</a> .
-------------	--

Definition at line 237 of file Game.java.

```
237     {  
238         this.turn = turn;  
239     }
```

#### 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.

```
247 {  
248     return this.state;  
249 }
```

#### 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

<i>state</i>	Current state of the <a href="#">Game</a> .
--------------	---

Definition at line 257 of file Game.java.

```
257 {  
258     this.state = state;  
259 }
```



### 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.

```
267     {  
268         return this.winnerID;  
269     }
```

### 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.

```
277     {  
278         return this.creatorID;  
279     }
```

#### 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.

```
287  
288         return this.createdAt;  
289     }
```

#### 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.

```
297  
298         if (this.state == GameState.FINISHED)  
299             throw new FinishedGameException();  
300  
301         this.state = GameState.IN_PROGRESS;  
302     }
```

**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**

<i>surrendeeID</i>	ID of the <a href="#">Player</a> that surrends.
--------------------	---

Definition at line 314 of file Game.java.

```
314
    {
315         if (surrendeeID.equals(this.player1ID))
316             this.finish(this.player2ID);
317         else if (surrendeeID.equals(this.player2ID))
318             this.finish(this.player1ID);
319         else
320             throw new NotPlayerException();
321     }
```

**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

<i>winnerID</i>	ID of the <a href="#">Player</a> that wins or null if the implicit <a href="#">Game</a> ended in a draw.
-----------------	--

Definition at line 333 of file Game.java.

```

333
334     {
335         if (this.state == GameState.NOT_STARTED)
336             throw new NotStartedGameException();
337         if (this.state == GameState.FINISHED)
338             throw new FinishedGameException();
339         if (winnerID != null && !winnerID.equals(this.player1ID) && !winnerID.equals(this.player2ID))
340             throw new NotPlayerException();
341         this.state = GameState.FINISHED;
342         this.winnerID = winnerID;
343     }
344
345

```

### 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

<i>playerID</i>	ID of the <a href="#">Player</a> that wants to place a piece in the implicit <a href="#">Game</a> .
<i>pieceType</i>	Type of the piece that the <a href="#">Player</a> wants to place in the implicit <a href="#">Game</a> .

Definition at line 359 of file Game.java.

```

360
361     if (this.state == GameState.NOT_STARTED)
362         throw new NotStartedGameException();
363     if (this.state == GameState.FINISHED)
364         throw new FinishedGameException();
365
366

```

```

366
367     if (playerID.equals(this.player1ID)) {
368         if (pieceType != PieceType.PLAYER1)
369             throw new NotPlayerPieceException();
370     } else if (playerID.equals(this.player2ID)) {
371         if (pieceType != PieceType.PLAYER2)
372             throw new NotPlayerPieceException();
373     } else {
374         throw new NotPlayerException();
375     }
376
377     if (pieceType != this.turn)
378         throw new NotPlayerTurnException();
379 }

```

### 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.

```

388
389     if (this.state == GameState.NOT_STARTED) {
390         throw new NotStartedGameException();
391     }
392     if (this.state == GameState.FINISHED)
393         throw new FinishedGameException();
394
395     this.turn = (this.turn == PieceType.PLAYER1 ? PieceType.PLAYER2 : PieceType.PLAYER1);
396 }

```

## 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.

```
22                                     {
23         new GameDriver().start();
24     }
```

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 [tielcon](#)
- 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") ||
519                     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.

```

546         {
547             if (!game.get("state").toString().equals("FINISHED")) {
548                 Alert confirm = new Alert(AlertType.CONFIRMATION, "You will exit without saving. Are you
                    sure?",
549                     ButtonType.YES, ButtonType.NO);
550                 confirm.showAndWait();
551                 if (confirm.getResult() == ButtonType.YES) {
552                     ViewCtrl.domainCtrl.exitGame();
553                     ViewCtrl.changeScene("template/PlayView.fxml");
554                 }
555             } else {
556                 ViewCtrl.domainCtrl.exitGame();
557                 ViewCtrl.changeScene("template/PlayView.fxml");
558             }
559         }

```

### 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.

```

566         {
567             Pair<JSONObject, String> result = ViewCtrl.domainCtrl.saveGame();
568             if (result.second != null) {
569                 switch (result.second) {
570                     case "ERR_NOT_PLAYER":
571                         gameResult.setText("You are not part of this game!");
572                         break;
573                     default:
574                         gameResult.setText("Something went wrong, try again!");
575                         break;
576                 }
577             } else {
578                 gameResult.setText("");
579                 Alert confirm = new Alert(AlertType.CONFIRMATION, "Do you also want to exit the current
                    game?",
580                     ButtonType.YES, ButtonType.NO);
581                 confirm.showAndWait();
582                 if (confirm.getResult() == ButtonType.YES) {
583                     ViewCtrl.changeScene("template/PlayView.fxml");
584                 }
585             }
586         }

```

### 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.

```
593                                     {
594         if (!isSpectating) {
595             if (!isVsBot || turnPlayerID.equals(UUID.fromString(user.getString("id")))) {
596                 Alert confirm = new Alert(AlertType.CONFIRMATION,
597                     "You will surrender and the game will be saved. Are you sure?", ButtonType.YES,
598                     ButtonType.NO);
599                 confirm.showAndWait();
600                 if (confirm.getResult() == ButtonType.YES) {
601                     Pair<JSONObject, String> result = ViewCtrl.domainCtrl.surrender(turnPlayerID);
602                     if (result.second != null) {
603                         switch (result.second) {
604                             case "ERR_NOT_PLAYER":
605                                 gameResult.setText("You are not part of this game!");
606                                 break;
607                             case "ERR_FINISHED_GAME":
608                                 gameResult.setText("This game is already finished!");
609                                 break;
610                             case "ERR_NOT_STARTED_GAME":
611                                 gameResult.setText("This game has not yet started!");
612                                 break;
613                             default:
614                                 gameResult.setText("Something went wrong, try again!");
615                                 break;
616                         }
617                     } else {
618                         gameResult.setText("");
619                         game = result.first;
620                         renderState();
621                         renderResult(UUID.fromString(game.getString("winner_id")));
622                         ViewCtrl.domainCtrl.saveGame();
623                         confirm = new Alert(AlertType.CONFIRMATION, "Do you also want to exit the
624                             current game?",
625                             ButtonType.YES, ButtonType.NO);
626                         confirm.showAndWait();
627                         if (confirm.getResult() == ButtonType.YES) {
628                             ViewCtrl.changeScene("template/PlayView.fxml");
629                         }
630                     }
631                 }
632             }
633         }
634     }
```

### 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.

```

639         {
640             if (mouseEvent == null || !isSpectating) {
641                 if (mouseEvent == null || !isVsBot ||
turnPlayerID.equals(UUID.fromString(user.getString("id")))) {
642                     Pair<Integer, Integer> pos = (!turnPlayerIsBot ? getClickedPos(mouseEvent) : null);
643                     Pair<Pair<JSONObject, String>, String> result = ViewCtrl.domainCtrl.placePiece(pos,
turnPlayerID,
644                                     turnPieceType);
645                     if (result.second != null) {
646                         switch (result.second) {
647                             case "ERR_NOT_PLAYER":
648                                 gameResult.setText("You are not part of this game!");
649                                 break;
650                             case "ERR_NOT_PLAYER_PIECE":
651                                 gameResult.setText("Is not your turn!");
652                                 break;
653                             case "ERR_NOT_PLAYER_TURN":
654                                 gameResult.setText("Is not your turn!");
655                                 break;
656                             case "ERR_NOT_STARTED_GAME":
657                                 gameResult.setText("This game has not yet started!");
658                                 break;
659                             case "ERR_INVALID_POSITION":
660                                 gameResult.setText("Can't place there!");
661                                 break;
662                             case "ERR_FINISHED_GAME":
663                                 gameResult.setText("");
664                                 board = result.first.first;
665                                 turnPieceType = result.first.second;
666                                 game = ViewCtrl.domainCtrl.viewGame();
667                                 renderState();
668                                 String winner = game.optString("winner_id", null);
669                                 UUID winnerID = (winner != null ? UUID.fromString(winner) : null);
670                                 ViewCtrl.domainCtrl.saveGame();
671                                 renderResult(winnerID);
672                                 break;
673                             default:
674                                 gameResult.setText("Something went wrong, try again!");
675                                 break;
676                         }
677                     } else {
678                         try {
679                             board = result.first.first;
680                             turnPieceType = result.first.second;
681                             gameResult.setText("");
682                             renderState();
683                             if (turnPlayerIsBot) {
684                                 timer = new Timer();
685                                 timer.schedule(new BotTask(), 500);
686                             }
687                         } catch (Exception e) {
688                         }
689                     }
690                 }
691             }
692         }

```

**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.

```

737     {
738         surrender.setVisible(false);
739         surrenderButton.setVisible(false);
740         save.setVisible(false);
741         saveButton.setVisible(false);
742         assistedMode.setVisible(false);
743         player1Turn.setVisible(false);
744         player2Turn.setVisible(false);
745         isSpectating = true;
746
747         if (winnerID == null) {
748             gameResult.setText("The game has ended in a draw.");
749             tieIcon.setVisible(true);
750         } else if (winnerID.equals(UUID.fromString(players.first.getString("id")))) {
751             gameResult.setTextFill(Color.web("0xFFFFFFFF", 1.0));
752             gameResult.setText(String.format("%s has won the game.", players.first.getString("name")));
753             winIcon.setVisible(true);
754         } else {
755             gameResult.setTextFill(Color.web("0x000000", 1.0));
756             gameResult.setText(String.format("%s has won the game.", players.second.getString("name")));
757             winIcon.setVisible(true);
758         }
759     }

```

**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.

```

766     {
767         for (int i = 0; i < 8; i++) {
768             char[] row = board.getString(String.format("row%d", i)).toCharArray();
769             for (int j = 0; j < 8; j++)
770                 drawPiece(new Pair<Integer, Integer>(i, j), row[j], false);
771         }
772     }

```

**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.

```
779                                     {
780         Circle circle = getCircle(pos);
781         switch (pieceType) {
782             case 'B':
783                 if (!stroke) {
784                     circle.setFill(Color.web("0xFFFFFF", 1.0));
785                     circle.setStrokeWidth(0);
786                 } else {
787                     circle.setStrokeWidth(3);
788                     circle.setStrokeType(StrokeType.INSIDE);
789                     circle.setStroke(Color.web("0xFFFFFF", 1.0));
790                 }
791                 break;
792             case 'N':
793                 if (!stroke) {
794                     circle.setFill(Color.web("0x000000", 1.0));
795                     circle.setStrokeWidth(0);
796                 } else {
797                     circle.setStrokeWidth(3);
798                     circle.setStrokeType(StrokeType.INSIDE);
799                     circle.setStroke(Color.web("0x000000", 1.0));
800                 }
801                 break;
802             case 'X':
803                 circle.setStrokeWidth(3);
804                 circle.setStrokeType(StrokeType.INSIDE);
805                 circle.setStroke(Color.web("0x0059ff", 1.0));
806                 break;
807             case '?':
808                 circle.setFill(Color.web("0x34d399", 1.0));
809                 circle.setStrokeWidth(0);
810                 break;
811             default:
812                 break;
813         }
814     }
```

**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.

```

821
822     Pair<Integer, Integer> pos = new Pair<Integer, Integer>(-1, -1);
823     String piece = ((Circle) mouseEvent.getPickResult().getIntersectedNode()).getId();
824     pos.first = Character.getNumericValue(piece.charAt(1)) - 1;
825     pos.second = Character.getNumericValue(piece.charAt(3)) - 1;
826     return pos;
827 }
```

**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.

```

834
835     try {
836         Field field = this.getClass().getDeclaredField(String.format("f%s%s", pos.first + 1,
pos.second + 1));
837         field.setAccessible(true);
838         return (Circle) field.get(this);
839     } catch (Exception e) {
840         return new Circle();
841     }
842 }
```

**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.

```

859
860     renderState();
861 }
```

## 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 InvalidPlayersException, 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 surrendeelID) 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, 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.*
- 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

<i>player1ID</i>	UUID of Player1
<i>player2ID</i>	UUID of Player2
<i>configuration</i>	Instance of a <a href="#">Configuration</a>
<i>creatorID</i>	UUID of the creator <a href="#">User</a> .

## Returns

[Game](#)

Definition at line 79 of file GameCtrl.java.

```
80
81
82     if (player1ID.equals(player2ID))
83         throw new InvalidPlayersException();
84
85     JSONObject rawPlayer1 = this.playerRepositoryCtrl.get(player1ID);
86     if (rawPlayer1 == null)
87         throw new InvalidPlayersException();
88
89     JSONObject rawPlayer2 = this.playerRepositoryCtrl.get(player2ID);
90     if (rawPlayer2 == null)
91         throw new InvalidPlayersException();
92
93     if (rawPlayer1.getBoolean("is_deleted") || rawPlayer2.getBoolean("is_deleted"))
94         throw new InvalidPlayersException();
95
96     if (rawPlayer1.getString("type").equals("BOT") && rawPlayer2.getString("type").equals("BOT"))
97         if (rawPlayer1.getInt("difficulty") == rawPlayer2.getInt("difficulty"))
98             throw new InvalidPlayersException();
99
100     JSONObject rawConfiguration =
101     this.configurationRepositoryCtrl.getConfiguration(configurationName);
102     if (rawConfiguration == null)
103         throw new InvalidConfigurationException();
104
105     JSONObject rawBoard = this.configurationRepositoryCtrl.getBoard(configurationName);
106     if (rawBoard == null)
107         throw new InvalidBoardException();
108
109     Board playingBoard = new Board(rawBoard);
110
111     LocalDateTime now = LocalDateTime.now();
112     DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
```

```

112         String name = String.format("%s VS %s | %s", rawPlayer1.getString("name"),
rawPlayer2.getString("name"),
113             now.format(dateFormat));
114
115         Game game = new Game(name, player1ID, player2ID, configurationName, creatorID);
116
117         this.repositoryCtrl.save(game.serialize(), playingBoard.serialize());
118         return game;
119     }

```

### 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

<i>name</i>	Name of a <code>Game</code>
<i>playerID</i>	UUID of a <code>Player</code>

#### Returns

`Game`

Definition at line 129 of file GameCtrl.java.

```

129                                     {
130         if (name.isBlank())
131             throw new NotPlayerException();
132
133         JSONObject rawGame = this.repositoryCtrl.getGame(name);
134
135         if (rawGame == null)
136             throw new NotPlayerException();
137
138         Game game = new Game(rawGame);
139
140         if (!game.getPlayer1ID().equals(playerID) && !game.getPlayer2ID().equals(playerID)
141             && !game.getCreatorID().equals(playerID))
142             throw new NotPlayerException();
143
144         return game;
145     }

```

### 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

<i>name</i>	Name of a <a href="#">Game</a>
<i>playerID</i>	UUID of a <a href="#">Player</a>

#### Returns

[Board](#)

Definition at line 155 of file GameCtrl.java.

```
155                                     {
156     if (name.isBlank())
157         throw new NotPlayerException();
158
159     JSONObject rawPlayingBoard = this.repositoryCtrl.getBoard(name);
160
161     if (rawPlayingBoard == null)
162         throw new NotPlayerException();
163
164     return new Board(rawPlayingBoard);
165 }
```

### 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

<i>playerID</i>	UUID of a <a href="#">Player</a> .
-----------------	------------------------------------

## Returns

`ArrayList<String>`

Definition at line 174 of file GameCtrl.java.

```

174                                     {
175         return this.repositoryCtrl.listGames(playerID);
176     }
```

### 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

<i>game</i>	<a href="#">Game</a> instance
<i>playingBoard</i>	<a href="#">Board</a> instance
<i>playerID</i>	UUID instance.

## Returns

[Game](#).

Definition at line 187 of file GameCtrl.java.

```

187                                     {
188         if (!game.getPlayer1ID().equals(playerID) && !game.getPlayer2ID().equals(playerID)
189             && !game.getCreatorID().equals(playerID))
190             throw new NotPlayerException();
191
192         this.repositoryCtrl.save(game.serialize(), playingBoard.serialize());
193         return game;
194     }
```

### 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

<i>game</i>	<a href="#">Game</a> instance
-------------	-------------------------------

#### Returns

Playing [Game](#)

Definition at line 203 of file GameCtrl.java.

```
203                                     {
204     game.play();
205     return game;
206 }
```

### 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

<i>game</i>	<a href="#">Game</a> instance
<i>surrendeeID</i>	UUID of <a href="#">Player</a>

**Returns**

[Game](#)

Definition at line 216 of file GameCtrl.java.

```
217                                     {
218         game.surrender(surrendeeID);
219         return game;
220     }
```

**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**

<i>game</i>	<a href="#">Game</a> instance
<i>winnerID</i>	UUID of <a href="#">Player</a>

**Returns**

[Game](#)

Definition at line 231 of file GameCtrl.java.

```
232                                     {
233         game.finish(winnerID);
234         return game;
235     }
```

**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**

<i>game</i>	<a href="#">Game</a> instance
<i>winnerID</i>	UUID of <a href="#">Player</a>
<i>pieceType</i>	PieceType

Definition at line 246 of file GameCtrl.java.

```
247
248         {
249             game.checkPlaceRights(playerID, pieceType);
250         }
```

**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**

<i>game</i>	Instance of a <a href="#">Game</a>
-------------	------------------------------------

**Returns**

[Game](#)

Definition at line 258 of file GameCtrl.java.

```
258
259         game.nextTurn();
260         return game;
261     }
```

## 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`.

```
28         {
29             this.currentGame = null;
30         }
```

## 6.45.3 Member Function Documentation

### 6.45.3.1 start()

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

Definition at line 34 of file GameDriver.java.

```
34         {
35             while (true) {
36                 this.mainMenu();
37             }
38         }
```

### 6.45.3.2 mainMenu()

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

Definition at line 40 of file GameDriver.java.

```
40         {
41             String title = (this.currentGame != null ? String.format("Current: %s\n",
this.currentGame.getName()) : null);
42             switch (Driver.menu(title, "Game Driver",
43                 new Pair<String, String>("1", "Create Game"),
44                 new Pair<String, String>("2", "Get name"),
45                 new Pair<String, String>("3", "Set name"),
46                 new Pair<String, String>("4", "Get player1ID"),
47                 new Pair<String, String>("5", "Get player2ID"),
48                 new Pair<String, String>("6", "Get configurationName"),
49                 new Pair<String, String>("7", "Set configurationName"),
50                 new Pair<String, String>("8", "Get turn"),
51                 new Pair<String, String>("9", "Set turn"),
52                 new Pair<String, String>("10", "Get state"),
53                 new Pair<String, String>("11", "Set state"),
54                 new Pair<String, String>("12", "Get winnerID"),
55                 new Pair<String, String>("13", "Get creatorID"),
56                 new Pair<String, String>("14", "Get createdAt"),
57                 new Pair<String, String>("15", "Serialize to JSON"),
58                 new Pair<String, String>("16", "Deserialize from JSON"),
59                 new Pair<String, String>("17", "Execute play"),
60                 new Pair<String, String>("18", "Execute surrender"),
61                 new Pair<String, String>("19", "Execute finish"),
62                 new Pair<String, String>("20", "Execute checkPlaceRights"),
63                 new Pair<String, String>("21", "Execute nextTurn"))) {
64                 case "1":
65                     this.create();
66                     break;
67                 case "2":
68                     this.getName();
69                     break;
70                 case "3":
71                     this.setName();
72                     break;
73                 case "4":
74                     this.getPlayer1ID();
75                     break;
76                 case "5":
77                     this.getPlayer2ID();
78                     break;
79                 case "6":
80                     this.getConfigurationName();
81                     break;
82                 case "7":
83                     this.setConfigurationName();
84                     break;
85                 case "8":
86                     this.getTurn();
87                     break;
88                 case "9":
89                     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`.

```
148         {
149             if (this.currentGame == null) {
150                 System.out.println("No current Game!");
151                 return;
152             }
153
154             System.out.println(String.format("%s's name is: %s", this.currentGame.getName(),
155                 this.currentGame.getName()));
156         }
```

#### 6.45.3.5 setName()

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

Definition at line 157 of file `GameDriver.java`.

```
157         {
158             if (this.currentGame == null) {
159                 System.out.println("No current Game!");
160                 return;
161             }
162
163             try {
164                 this.currentGame.setName(Driver.input("Name?"));
165                 System.out.println(String.format("%s's name changed successfully!",
166                     this.currentGame.getName()));
167             } catch (Exception e) {
168                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
169                     e.getMessage()));
170             }
171         }
```

#### 6.45.3.6 getPlayer1ID()

```
void test.driver.GameDriver.getPlayer1ID ( ) [private]
```

Definition at line 171 of file `GameDriver.java`.

```
171         {
172             if (this.currentGame == null) {
173                 System.out.println("No current Game!");
174                 return;
175             }
176
177             System.out.println(
178                 String.format("%s's player1ID is: %s", this.currentGame.getName(),
179                     this.currentGame.getPlayer1ID()));
180         }
```

#### 6.45.3.7 getPlayer2ID()

```
void test.driver.GameDriver.getPlayer2ID ( ) [private]
```

Definition at line 181 of file `GameDriver.java`.

```
181         {
182             if (this.currentGame == null) {
183                 System.out.println("No current Game!");
184                 return;
185             }
186
187             System.out.println(
188                 String.format("%s's player2ID is: %s", this.currentGame.getName(),
189                     this.currentGame.getPlayer2ID()));
190         }
```

### 6.45.3.8 getConfigurationName()

```
void test.driver.GameDriver.getConfigurationName ( ) [private]
```

Definition at line 191 of file GameDriver.java.

```
191 {
192     if (this.currentGame == null) {
193         System.out.println("No current Game!");
194         return;
195     }
196
197     System.out.println(String.format("%s's configurationName is: %s", this.currentGame.getName(),
198                                     this.currentGame.getConfigurationName()));
199 }
```

### 6.45.3.9 setConfigurationName()

```
void test.driver.GameDriver.setConfigurationName ( ) [private]
```

Definition at line 201 of file GameDriver.java.

```
201 {
202     if (this.currentGame == null) {
203         System.out.println("No current Game!");
204         return;
205     }
206
207     try {
208         this.currentGame.setConfigurationName(Driver.input("Configuration name?"));
209         System.out
210             .println(String.format("%s's configurationName changed successfully!",
211                                   this.currentGame.getName()));
212     } catch (Exception e) {
213         System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
214                                         e.getMessage()));
215     }
216 }
```

### 6.45.3.10 getTurn()

```
void test.driver.GameDriver.getTurn ( ) [private]
```

Definition at line 216 of file GameDriver.java.

```
216 {
217     if (this.currentGame == null) {
218         System.out.println("No current Game!");
219         return;
220     }
221
222     System.out.println(String.format("%s's turn is: %s", this.currentGame.getName(),
223                                     this.currentGame.getTurn()));
224 }
```

### 6.45.3.11 setTurn()

```
void test.driver.GameDriver.setTurn ( ) [private]
```

Definition at line 225 of file GameDriver.java.

```
225         {
226             if (this.currentGame == null) {
227                 System.out.println("No current Game!");
228                 return;
229             }
230
231             switch (Driver.menu(null, "Select Turn",
232                 new Pair<String, String>("1", "PLAYER 1"),
233                 new Pair<String, String>("2", "PLAYER 2"))) {
234             case "1":
235                 this.currentGame.setTurn(PieceType.PLAYER1);
236                 break;
237             case "2":
238                 this.currentGame.setTurn(PieceType.PLAYER2);
239                 break;
240             }
241             System.out.println(String.format("%s's turn changed successfully!",
242                 this.currentGame.getName()));
242         }
```

### 6.45.3.12 getState()

```
void test.driver.GameDriver.getState ( ) [private]
```

Definition at line 244 of file GameDriver.java.

```
244         {
245             if (this.currentGame == null) {
246                 System.out.println("No current Game!");
247                 return;
248             }
249
250             System.out.println(String.format("%s's state is: %s", this.currentGame.getName(),
251                 this.currentGame.getState()));
251         }
```

### 6.45.3.13 setState()

```
void test.driver.GameDriver.setState ( ) [private]
```

Definition at line 253 of file GameDriver.java.

```
253         {
254             if (this.currentGame == null) {
255                 System.out.println("No current Game!");
256                 return;
257             }
258
259             switch (Driver.menu(null, "Select State",
260                 new Pair<String, String>("1", "NOT_STARTED"),
261                 new Pair<String, String>("2", "IN_PROGRESS"),
262                 new Pair<String, String>("3", "FINISHED"))) {
263             case "1":
264                 this.currentGame.setState(GameState.NOT_STARTED);
265                 break;
266             case "2":
267                 this.currentGame.setState(GameState.IN_PROGRESS);
268                 break;
269             case "3":
270                 this.currentGame.setState(GameState.FINISHED);
271                 break;
272             }
273             System.out.println(String.format("%s's state changed successfully!",
274                 this.currentGame.getName()));
274         }
```

### 6.45.3.14 getWinnerID()

```
void test.driver.GameDriver.getWinnerID ( ) [private]
```

Definition at line 276 of file GameDriver.java.

```
276         {
277             if (this.currentGame == null) {
278                 System.out.println("No current Game!");
279                 return;
280             }
281
282             System.out.println(
283                 String.format("%s's winnerID is: %s", this.currentGame.getName(),
284                     this.currentGame.getWinnerID()));
285         }
```

### 6.45.3.15 getCreatorID()

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

Definition at line 286 of file GameDriver.java.

```
286         {
287             if (this.currentGame == null) {
288                 System.out.println("No current Game!");
289                 return;
290             }
291
292             System.out.println(
293                 String.format("%s's creatorID is: %s", this.currentGame.getName(),
294                     this.currentGame.getCreatorID()));
295         }
```

### 6.45.3.16 getCreatedAt()

```
void test.driver.GameDriver.getCreatedAt ( ) [private]
```

Definition at line 296 of file GameDriver.java.

```
296         {
297             if (this.currentGame == null) {
298                 System.out.println("No current Game!");
299                 return;
300             }
301
302             DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
303             System.out.println(String.format("%s's createdAt is: %s", this.currentGame.getName(),
304                 this.currentGame.getCreatedAt().format(dateFormat)));
305         }
```

### 6.45.3.17 serialize()

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

Definition at line 307 of file GameDriver.java.

```
307         {
308             if (this.currentGame == null) {
309                 System.out.println("No current Game!");
310                 return;
311             }
312
313             System.out.println(String.format("%s's serialized to JSON is: %s", this.currentGame.getName(),
314                 this.currentGame.serialize().toString(2)));
315         }
```

### 6.45.3.18 deserialize()

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

Definition at line 317 of file GameDriver.java.

```
317         {
318             if (this.currentGame == null) {
319                 System.out.println("No current Game!");
320                 return;
321             }
322
323             System.out.println(this.currentGame.serialize().toString(2));
324             this.currentGame = new Game(this.currentGame.serialize());
325             System.out.println(
326                 String.format("\n%s's deserialized from the above JSON successfully!\n",
327                     this.currentGame.getName()));
328             System.out.println(String.format("name:\t\t\t%s", this.currentGame.getName()));
329             System.out.println(String.format("player1ID:\t\t\t%s", this.currentGame.getPlayer1ID()));
330             System.out.println(String.format("player2ID:\t\t\t%s", this.currentGame.getPlayer2ID()));
331             System.out.println(String.format("configurationName:\t\t\t%s",
332                 this.currentGame.getConfigurationName()));
333             System.out.println(String.format("turn:\t\t\t\t%s", this.currentGame.getTurn()));
334             System.out.println(String.format("state:\t\t\t\t%s", this.currentGame.getState()));
335             System.out.println(String.format("winnerID:\t\t\t\t%s", this.currentGame.getWinnerID()));
336             System.out.println(String.format("creatorID:\t\t\t\t%s", this.currentGame.getCreatorID()));
337             DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
338             System.out.println(String.format("createdAt:\t\t\t\t%s",
339                 this.currentGame.getCreatedAt().format(dateFormat)));
340         }
```

### 6.45.3.19 play()

```
void test.driver.GameDriver.play ( ) [private]
```

Definition at line 339 of file GameDriver.java.

```
339         {
340             if (this.currentGame == null) {
341                 System.out.println("No current Game!");
342                 return;
343             }
344
345             try {
346                 this.currentGame.play();
347                 System.out.println(String.format("The Game state has changed to %s",
348                     this.currentGame.getState()));
349             } catch (Exception e) {
350                 System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
351                     e.getMessage()));
352             }
353         }
```

### 6.45.3.20 surrender()

```
void test.driver.GameDriver.surrender ( ) [private]
```

Definition at line 353 of file GameDriver.java.

```
353         {
354             if (this.currentGame == null) {
355                 System.out.println("No current Game!");
356                 return;
357             }
358
359             try {
360                 switch (Driver.menu(null, "Select who surrenders",
361                     new Pair<String, String>("1", "PLAYER 1"),
362                     new Pair<String, String>("2", "PLAYER 2"))) {
363                     case "1":
```



```

364         this.currentGame.surrender(this.currentGame.getPlayer1ID());
365         System.out.println("PLAYER 2 has won the Game");
366         break;
367     case "2":
368         this.currentGame.surrender(this.currentGame.getPlayer2ID());
369         System.out.println("PLAYER 1 has won the Game");
370         break;
371     }
372     System.out.println(String.format("The Game winnerID has changed to %s",
this.currentGame.getWinnerID()));
373     System.out.println(String.format("The Game state has changed to %s",
this.currentGame.getState()));
374     } catch (Exception e) {
375         System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
376     }
377 }

```

### 6.45.3.21 finish()

```
void test.driver.GameDriver.finish ( ) [private]
```

Definition at line 379 of file GameDriver.java.

```

379     {
380         if (this.currentGame == null) {
381             System.out.println("No current Game!");
382             return;
383         }
384
385         try {
386             switch (Driver.menu(null, "Select who wins",
387                 new Pair<String, String>("1", "PLAYER 1"),
388                 new Pair<String, String>("2", "PLAYER 2"),
389                 new Pair<String, String>("3", "DRAW"))) {
390             case "1":
391                 this.currentGame.finish(this.currentGame.getPlayer1ID());
392                 System.out.println("PLAYER 1 has won the Game");
393                 break;
394             case "2":
395                 this.currentGame.finish(this.currentGame.getPlayer2ID());
396                 System.out.println("PLAYER 2 has won the Game");
397                 break;
398             case "3":
399                 this.currentGame.finish(null);
400                 System.out.println("The Game has resulted in a draw");
401                 break;
402             }
403             System.out.println(String.format("The Game winnerID has changed to %s",
this.currentGame.getWinnerID()));
404             System.out.println(String.format("The Game state has changed to %s",
this.currentGame.getState()));
405         } catch (Exception e) {
406             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
407         }
408     }

```

### 6.45.3.22 checkPlaceRights()

```
void test.driver.GameDriver.checkPlaceRights ( ) [private]
```

Definition at line 410 of file GameDriver.java.

```

410     {
411         if (this.currentGame == null) {
412             System.out.println("No current Game!");
413             return;
414         }
415
416         UUID playerId = null;
417         switch (Driver.menu(null, "Select who places",

```

```

418         new Pair<String, String>("1", "PLAYER 1"),
419         new Pair<String, String>("2", "PLAYER 2")))) {
420     case "1":
421         playerID = this.currentGame.getPlayer1ID();
422         break;
423     case "2":
424         playerID = this.currentGame.getPlayer2ID();
425         break;
426     }
427     try {
428         switch (Driver.menu(null, "Select piece type",
429             new Pair<String, String>("1", "PLAYER 1 pieces"),
430             new Pair<String, String>("2", "PLAYER 2 pieces"))) {
431             case "1":
432                 this.currentGame.checkPlaceRights(playerID, PieceType.PLAYER1);
433                 break;
434             case "2":
435                 this.currentGame.checkPlaceRights(playerID, PieceType.PLAYER2);
436                 break;
437             }
438             System.out.println("The player did place the piece successfully");
439         } catch (Exception e) {
440             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
441                 e.getMessage()));
442         }
443     }

```

### 6.45.3.23 nextTurn()

```
void test.driver.GameDriver.nextTurn ( ) [private]
```

Definition at line 444 of file GameDriver.java.

```

444     {
445         if (this.currentGame == null) {
446             System.out.println("No current Game!");
447             return;
448         }
449
450         try {
451             this.currentGame.nextTurn();
452             System.out.println(String.format("The Game turn has changed to %s",
453                 this.currentGame.getTurn()));
454         } catch (Exception e) {
455             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
456                 e.getMessage()));
457         }
458     }

```

## 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.

```
28         {  
29             super (RepositoryType.GAME);  
30         }
```

### 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

<i>game</i>	Game to be saved.
<i>board</i>	Playing Board of the Game to be saved.

Definition at line 41 of file GameRepository.java.

```
41                                     {
42     String name = game.getString("name");
43     game.put("board", board);
44     this.createOrUpdate(name, game);
45 }
```

#### 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**

<i>name</i>	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.

```
54 {
55     JSONObject game = this.get(name);
56     if (game == null)
57         return null;
58
59     game.remove("board");
60     return game;
61 }
```

**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**

<i>name</i>	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.

```
70 {
71     JSONObject game = this.get(name);
72     if (game == null)
73         return null;
74
75     return game.getJSONObject("board");
76 }
```

#### 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

<i>configurationName</i>	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.

```
85                                     {
86     JSONObject all = this.list();
87
88     JSONObject current;
89     for (String key : all.keySet()) {
90         current = all.getJSONObject(key);
91         if (current.getString("configuration_name").equals(configurationName))
92             return true;
93     }
94
95     return false;
96 }
```

#### 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**

<i>playerID</i>	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.

```
105                                     {
106     JSONObject all = this.list();
107
108     JSONObject current;
109     for (String key : all.keySet()) {
110         current = all.getJSONObject(key);
111         if (current.getString("player1_id").equals(playerID) ||
112             current.getString("player2_id").equals(playerID)
113             || current.getString("creator_id").equals(playerID))
114             return true;
115     }
116     return false;
117 }
```

**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**

<i>playerID</i>	Player ID of a player in the Games to be gotten.
-----------------	--

**Returns**

ArrayList of the Game names by Player ID of the game database.

Definition at line 126 of file GameRepository.java.

```
126                                     {
127     ArrayList<String> list = new ArrayList<String>();
128     JSONObject all = this.list();
129
130     JSONObject current;
131     for (String key : all.keySet()) {
132         current = all.getJSONObject(key);
```

```

133         if (current.getString("player1_id").equals(playerID) ||
            current.getString("player2_id").equals(playerID)
134             || current.getString("creator_id").equals(playerID))
135             list.add(key);
136     }
137
138     return list;
139 }

```

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.

```
34     {  
35         this.repository = new GameRepository();  
36     }
```

## 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

<i>game</i>	Game to be saved.
<i>board</i>	Playing Board of the Game to be saved.

Definition at line 47 of file GameRepositoryCtrl.java.

```
47                                     {
48         this.repository.save(game, board);
49     }
```

### 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

<i>name</i>	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.

```
58                                     {
59         return this.repository.getGame(name);
60     }
```

### 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

<i>name</i>	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.

```
69      {  
70          return this.repository.getBoard(name);  
71      }
```

#### 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

<i>configurationName</i>	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.

```
80      {  
81          return this.repository.existsGameByConfigurationName(configurationName);  
82      }
```

#### 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**

<i>playerID</i>	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.

```
91                                     {
92     return this.repository.existsGameByPlayerID(playerID.toString());
93 }
```

**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**

<i>playerID</i>	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.

```
102                                     {
103     return this.repository.listGames(playerID.toString());
104 }
```

## 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.GameCreateView 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.GameCreateView.GameCreateView ( )
```

Class creator.

Definition at line 37 of file GamesCreateView.java.

```
37         {
38     }
```

## 6.48.3 Member Function Documentation

### 6.48.3.1 initialize()

```
void view.GameCreateView.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.

```
168     {
169         currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
170         userMap = new HashMap<String, UUID>();
171         botMap = new HashMap<String, UUID>();
172
173         ArrayList<Pair<String, UUID> userList = ViewCtrl.domainCtrl.listUsers();
174         for(Pair<String, UUID> user : userList) {
175             userChooser1.getItems().add(user.first);
176             userChooser2.getItems().add(user.first);
177             userMap.put(user.first, user.second);
178         }
179
180         ArrayList<Pair<String, UUID> botList = ViewCtrl.domainCtrl.listBots();
181         for(Pair<String, UUID> bot : botList) {
182             botChooser1.getItems().add(bot.first);
183             botChooser2.getItems().add(bot.first);
184             botMap.put(bot.first, bot.second);
185         }
186
187         ArrayList<String> configList = ViewCtrl.domainCtrl.listConfigurations().first;
188         for(String configName : configList) configChooser.getItems().add(configName);
189
190         userChooser1();
191         botChooser2();
192     }
```

### 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.

```
199         {
200             ViewCtrl.changeScene("template/UserView.fxml");
201         }
```

### 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.

```
208         {
209             ViewCtrl.changeScene("template/BotsView.fxml");
210         }
```

### 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.

```
217         {
218             ViewCtrl.changeScene("template/ConfigView.fxml");
219         }
```



### 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.

```
226         {
227             ViewCtrl.changeScene("template/RankingView.fxml");
228         }
```

### 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.

```
235         {
236             ViewCtrl.changeScene("template/PlayView.fxml");
237         }
```

### 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.

```
244         {
245             ViewCtrl.changeScene("template/GamesView.fxml");
246         }
```

#### 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.

```
253         {  
254             userChooser1.setDisable(false);  
255             botChooser1.setDisable(true);  
256         }
```

#### 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.

```
263         {  
264             userChooser2.setDisable(false);  
265             botChooser2.setDisable(true);  
266         }
```

#### 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.

```
273         {  
274             userChooser1.setDisable(true);  
275             botChooser1.setDisable(false);  
276         }
```

**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.

```
283         {
284             userChooser2.setDisable(true);
285             botChooser2.setDisable(false);
286         }
```

**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.

```
293         {
294             String chosenConfig = (String) configChooser.getValue();
295             String chosenUser1 = (String) userChooser1.getValue();
296             String chosenBot1 = (String) botChooser1.getValue();
297             String chosenUser2 = (String) userChooser2.getValue();
298             String chosenBot2 = (String) botChooser2.getValue();
299
300             if (chosenConfig != null) {
301                 UUID player1ID = null;
302                 UUID player2ID = null;
303
304                 if (p11User.isSelected() && chosenUser1 != null) player1ID = userMap.get(chosenUser1);
305                 if (p11Bot.isSelected() && chosenBot1 != null) player1ID = botMap.get(chosenBot1);
306                 if (p12User.isSelected() && chosenUser2 != null) player2ID = userMap.get(chosenUser2);
307                 if (p12Bot.isSelected() && chosenBot2 != null) player2ID = botMap.get(chosenBot2);
308
309                 if (player1ID != null && player2ID != null) {
310                     Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createGame(player1ID, player2ID,
311                     chosenConfig);
312                     if (result.second != null) {
313                         switch (result.second) {
314                             case "ERR_INVALID_PLAYERS":
315                                 createGameResult.setText("The player selection is invalid!");
316                                 break;
317                             case "ERR_INVALID_CONFIGURATION":
318                                 createGameResult.setText("The selected configuration is invalid!");
319                                 break;
320                         }
321                     }
322                 }
323             }
```

```

319         case "ERR_INVALID_BOARD":
320             createGameResult.setText("The playing board is invalid!");
321             break;
322         default:
323             createGameResult.setText("Something went wrong, try again!");
324             break;
325     }
326 }
327 else {
328     userChooser1.getSelectionModel().select(null);
329     botChooser1.getSelectionModel().select(null);
330     userChooser2.getSelectionModel().select(null);
331     botChooser2.getSelectionModel().select(null);
332     createGameResult.setText("Success!");
333 }
334 }
335 }
336 }

```

### 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?",
345             ButtonType.YES, ButtonType.NO);
346         confirm.showAndWait();
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.GameCreateView.bots [private]`

Menu Bots tab.

Definition at line 51 of file GameCreateView.java.

#### 6.48.4.3 config

`Text view.GameCreateView.config [private]`

Menu Configuration tab.

Definition at line 56 of file GameCreateView.java.

#### 6.48.4.4 games

`Text view.GameCreateView.games [private]`

Menu Games tab.

Definition at line 61 of file GameCreateView.java.

#### 6.48.4.5 ranking

`Text view.GameCreateView.ranking [private]`

Menu Ranking tab.

Definition at line 66 of file GameCreateView.java.

#### 6.48.4.6 play

`Text view.GameCreateView.play [private]`

Menu Play tab.

Definition at line 71 of file GameCreateView.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.GameAndView Class Reference

### Public Member Functions

- [GameAndView](#) ()  
*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.

```
91         {
92     currentUserName.setText (ViewCtrl.domainCtrl.viewUser().getString("name"));
93     }
```

### 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.

```
100         {
101     ViewCtrl.changeScene("template/UserView.fxml");
102     }
```

### 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.

```
109         {
110     ViewCtrl.changeScene("template/BotsView.fxml");
111     }
```

#### 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.

```
118         {
119             ViewCtrl.changeScene("template/ConfigView.fxml");
120         }
```

#### 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.

```
127         {
128             ViewCtrl.changeScene("template/RankingView.fxml");
129         }
```

#### 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.

```
136         {
137             ViewCtrl.changeScene("template/PlayView.fxml");
138         }
```

### 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.

```
145         {
146             ViewCtrl.changeScene("template/GamesCreateView.fxml");
147         }
```

### 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.

```
154         {
155             Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
156                                     ButtonType.YES, ButtonType.NO);
157             confirm.showAndWait();
158             if (confirm.getResult() == ButtonType.YES) {
159                 ViewCtrl.domainCtrl.logout();
160                 ViewCtrl.changeScene("template/LoginView.fxml");
161             }
162         }
```

## 6.50.4 Member Data Documentation

#### 6.50.4.1 user

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

Menu User tab.

Definition at line 38 of file GamesView.java.

#### 6.50.4.2 bots

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

Menu Bots tab.

Definition at line 43 of file GamesView.java.

#### 6.50.4.3 config

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

Menu Configuration tab.

Definition at line 48 of file GamesView.java.

#### 6.50.4.4 games

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

Menu Games tab.

Definition at line 53 of file GamesView.java.

#### 6.50.4.5 ranking

```
Text view.GameView.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, canEatDiagonally and pieceType attributes are set. The implicit maxDepth attribute is set to 1000 times the entered difficulty.

#### Parameters

<i>difficulty</i>	<a href="#">Difficulty</a> for the Monte Carlo Tree Search algorithm.
<i>canEatHorizontally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten horizontally.
<i>canEatVertically</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten vertically.
<i>canEatDiagonally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten diagonally.
<i>pieceType</i>	<a href="#">Player</a> that wants to be maximized.

Definition at line 46 of file HardDifficulty.java.

```
47         {
48         super(difficulty, canEatHorizontally, canEatVertically, canEatDiagonally, pieceType);
49         this.maxDepth = difficulty * 1000;
50     }
```

## 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

<i>playingBoard</i>	Current playing <a href="#">Board</a> .
---------------------	---

## 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.

```

64                                     {
65         Pair<Integer, Integer> bestPosition = null;
66
67         Board initialBoard = new Board(playingBoard);
68
69         TreeNode rootGame = new TreeNode(
70             this.pieceType, this.pieceType, initialBoard, this.canEatHorizontally, this.canEatVertically,
71             this.canEatDiagonally, null);
72
73         for (int i = 0; i < this.maxDepth; ++i) rootGame.play();
74
75         ArrayList<TreeNode> playedGames = rootGame.getChildren();
76
77         double maxWinRatio = Double.NEGATIVE_INFINITY;
78         for (TreeNode game: playedGames) {
79             if (game.getWinRatio() > maxWinRatio) {
80                 maxWinRatio = game.getWinRatio();
81                 bestPosition = game.getSelectedPosition();
82             }
83         }
84         return bestPosition;
85     }

```

## 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 entrypt. 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 entrypt. 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.

```
22  
23     new HardDifficultyDriver().start();  
24 }
```

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.

```
33         {
34             this.currentHardDifficulty = null;
35             this.fixtureRepository = new FixtureRepository();
36         }
```

### 6.53.3 Member Function Documentation

#### 6.53.3.1 start()

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

Definition at line 40 of file `HardDifficultyDriver.java`.

```

40         {
41             while (true) {
42                 this.mainMenu();
43             }
44         }

```

#### 6.53.3.2 mainMenu()

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

Definition at line 46 of file `HardDifficultyDriver.java`.

```

46         {
47             String title = null;
48             if (this.currentHardDifficulty != null)
49                 title = String.format("Current maximum depth: %s\n",
this.currentHardDifficulty.getMaxDepth());
50             if (this.currentBoard != null)
51                 title += String.format("Current Board: %s\n", this.nameCurrentBoard);
52
53             switch (Driver.menu(title, "HardDifficulty (Montecarlo) Driver",
54                 new Pair<String, String>("1", "Create HardDifficulty"),
55                 new Pair<String, String>("2", "Get difficulty"),
56                 new Pair<String, String>("3", "Get canEatHorizontally"),
57                 new Pair<String, String>("4", "Get canEatVertically"),
58                 new Pair<String, String>("5", "Get canEatDiagonally"),
59                 new Pair<String, String>("6", "Get pieceType"),
60                 new Pair<String, String>("7", "Get maxDepth"),
61                 new Pair<String, String>("8", "Set maxDepth"),
62                 new Pair<String, String>("9", "Load Board"),
63                 new Pair<String, String>("10", "Print Current Board"),
64                 new Pair<String, String>("11", "Get next best position"))) {
65             case "1":
66                 this.create();
67                 break;
68             case "2":
69                 this.getDifficulty();
70                 break;
71             case "3":
72                 this.getCanEatHorizontally();
73                 break;
74             case "4":
75                 this.getCanEatVertically();
76                 break;
77             case "5":
78                 this.getCanEatDiagonally();
79                 break;
80             case "6":
81                 this.getPieceType();
82                 break;
83             case "7":
84                 this.getMaxDepth();
85                 break;
86             case "8":
87                 this.setMaxDepth();
88                 break;
89             case "9":
90                 this.loadBoard();
91                 break;
92             case "10":
93                 this.printCurrentBoard();
94                 break;
95             case "11":
96                 this.getNextBestPosition();
97                 break;
98             }
99             Driver.pause();
100         }

```

### 6.53.3.3 create()

```
void test.driver.HardDifficultyDriver.create ( ) [private]
```

Definition at line 102 of file HardDifficultyDriver.java.

```

102     {
103         System.out.println(
104             "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");
105
106         Integer difficulty = Driver.inputInt("Difficulty (positive)?");
107         Boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
108         Boolean canEatVertically = Driver.inputBool("Can eat vertically?");
109         Boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
110         PieceType pieceType = null;
111
112         switch (Driver.menu(null, "Select Bot pieces",
113             new Pair<String, String>("1", "PLAYER 1 pieces"),
114             new Pair<String, String>("2", "PLAYER 2 pieces"))) {
115             case "1":
116                 pieceType = PieceType.PLAYER1;
117                 break;
118             case "2":
119                 pieceType = PieceType.PLAYER2;
120                 break;
121         }
122
123         this.currentHardDifficulty = new HardDifficulty(difficulty, canEatHorizontally,
            canEatVertically,
            canEatDiagonally, pieceType);
124
125         System.out.println(String.format("HardDifficulty with a maximum depth of %s created
            successfully!",
126             this.currentHardDifficulty.getMaxDepth()));
127     }
128 }
```

### 6.53.3.4 getDifficulty()

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

Definition at line 130 of file HardDifficultyDriver.java.

```

130     {
131         if (this.currentHardDifficulty == null) {
132             System.out.println("No current HardDifficulty!");
133             return;
134         }
135
136         System.out.println(
137             String.format("HardDifficulty's difficulty is: %s",
138                 this.currentHardDifficulty.getDifficulty()));
139     }
```

### 6.53.3.5 getCanEatHorizontally()

```
void test.driver.HardDifficultyDriver.getCanEatHorizontally ( ) [private]
```

Definition at line 140 of file HardDifficultyDriver.java.

```

140     {
141         if (this.currentHardDifficulty == null) {
142             System.out.println("No current HardDifficulty!");
143             return;
144         }
145
146         System.out.println(String.format("HardDifficulty's canEatHorizontally is: %s",
147             this.currentHardDifficulty.getCanEatHorizontally()));
148     }
```

### 6.53.3.6 getCanEatVertically()

```
void test.driver.HardDifficultyDriver.getCanEatVertically ( ) [private]
```

Definition at line 150 of file HardDifficultyDriver.java.

```
150         {
151             if (this.currentHardDifficulty == null) {
152                 System.out.println("No current HardDifficulty!");
153                 return;
154             }
155             System.out.println(String.format("HardDifficulty's canEatVertically is: %s",
156                 this.currentHardDifficulty.getCanEatVertically()));
157         }
158     }
```

### 6.53.3.7 getCanEatDiagonally()

```
void test.driver.HardDifficultyDriver.getCanEatDiagonally ( ) [private]
```

Definition at line 160 of file HardDifficultyDriver.java.

```
160         {
161             if (this.currentHardDifficulty == null) {
162                 System.out.println("No current HardDifficulty!");
163                 return;
164             }
165             System.out.println(String.format("HardDifficulty's canEatDiagonally is: %s",
166                 this.currentHardDifficulty.getCanEatDiagonally()));
167         }
168     }
```

### 6.53.3.8 getPieceType()

```
void test.driver.HardDifficultyDriver.getPieceType ( ) [private]
```

Definition at line 170 of file HardDifficultyDriver.java.

```
170         {
171             if (this.currentHardDifficulty == null) {
172                 System.out.println("No current HardDifficulty!");
173                 return;
174             }
175             System.out.println(
176                 String.format("HardDifficulty's pieceType is: %s",
177                     this.currentHardDifficulty.getPieceType()));
178         }
```

### 6.53.3.9 getMaxDepth()

```
void test.driver.HardDifficultyDriver.getMaxDepth ( ) [private]
```

Definition at line 180 of file HardDifficultyDriver.java.

```
180         {
181             if (this.currentHardDifficulty == null) {
182                 System.out.println("No current HardDifficulty!");
183                 return;
184             }
185             System.out.println(
186                 String.format("HardDifficulty's maxDepth is: %s",
187                     this.currentHardDifficulty.getMaxDepth()));
188         }
```



### 6.53.3.10 setMaxDepth()

```
void test.driver.HardDifficultyDriver.setMaxDepth ( ) [private]
```

Definition at line 190 of file HardDifficultyDriver.java.

```
190         {
191             if (this.currentHardDifficulty == null) {
192                 System.out.println("No current HardDifficulty!");
193                 return;
194             }
195
196             System.out.println(
197                 "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");
198
199             this.currentHardDifficulty.setMaxDepth(Driver.inputInt("Maximum depth (positive)?"));
200             System.out.println("HardDifficulty's maxDepth changed successfully!");
201         }
```

### 6.53.3.11 loadBoard()

```
void test.driver.HardDifficultyDriver.loadBoard ( ) [private]
```

Definition at line 203 of file HardDifficultyDriver.java.

```
203         {
204             if (this.currentHardDifficulty == null) {
205                 System.out.println("No current HardDifficulty!");
206                 return;
207             }
208
209             Pair<String, Board> selectedBoard = this.listBoardFixtures();
210
211             this.nameCurrentBoard = selectedBoard.first;
212             this.currentBoard = selectedBoard.second;
213
214             System.out.println(String.format("Board %s loaded successfully!\n", this.nameCurrentBoard));
215             this.printBoard(this.currentBoard);
216         }
```

### 6.53.3.12 printCurrentBoard()

```
void test.driver.HardDifficultyDriver.printCurrentBoard ( ) [private]
```

Definition at line 218 of file HardDifficultyDriver.java.

```
218         {
219             if (this.currentHardDifficulty == null) {
220                 System.out.println("No current HardDifficulty!");
221                 return;
222             }
223
224             if (this.currentBoard == null) {
225                 System.out.println("No current Board!");
226                 return;
227             }
228
229             System.out.println(String.format("Board %s printed successfully!\n", this.nameCurrentBoard));
230             this.printBoard(this.currentBoard);
231         }
```

### 6.53.3.13 getNextBestPosition()

```
void test.driver.HardDifficultyDriver.getNextBestPosition ( ) [private]
```

Definition at line 233 of file HardDifficultyDriver.java.

```

233         {
234             if (this.currentHardDifficulty == null) {
235                 System.out.println("No current HardDifficulty!");
236                 return;
237             }
238
239             if (this.currentBoard == null) {
240                 System.out.println("No current Board!");
241                 return;
242             }
243
244             System.out.println("Take into account that the state of the current Board won't be globally
modified.\n");
245
246             this.printBoard(this.currentBoard);
247
248             long startTime = System.currentTimeMillis();
249             Pair<Integer, Integer> nextBestPosition =
this.currentHardDifficulty.place(this.currentBoard.getBoard());
250             long durationTime = System.currentTimeMillis() - startTime;
251
252             Board tempBoard = new Board(this.currentBoard.getBoard());
253
254             if (nextBestPosition != null) {
255                 tempBoard.placePiece(nextBestPosition, this.currentHardDifficulty.getPieceType(),
256                     this.currentHardDifficulty.getCanEatHorizontally(),
257                     this.currentHardDifficulty.getCanEatVertically(),
258                     this.currentHardDifficulty.getCanEatDiagonally());
259                 System.out.println(
260                     String.format("The best position calculated in %s ms is %s\n", durationTime,
nextBestPosition));
261                 System.out.println("The addition of the piece would look like this:\n");
262                 this.printBoard(tempBoard);
263             } else {
264                 System.out.println("There isn't any possible position left to place a piece on.");
265             }
266         }

```

### 6.53.3.14 listBoardFixtures()

```
Pair<String, Board> test.driver.HardDifficultyDriver.listBoardFixtures ( ) [private]
```

Definition at line 268 of file HardDifficultyDriver.java.

```

268         {
269             Integer selectedBoard = -1;
270             ArrayList<String> listBoards = this.fixtureRepository.listFiles();
271
272             while (selectedBoard < 0 || selectedBoard >= listBoards.size()) {
273                 Driver.clear();
274                 System.out.println("==== Available Boards ==== \n");
275
276                 for (Integer i = 0; i < listBoards.size(); ++i)
277                     System.out.println(String.format("[%d]\t%s", i, listBoards.get(i)));
278                 System.out.println("");
279
280                 selectedBoard = Driver.inputInt("What Board would you like to load?");
281             }
282
283             Driver.clear();
284
285             return new Pair<String, Board>(listBoards.get(selectedBoard),
286                 new Board(this.fixtureRepository.boardFileToJSON(listBoards.get(selectedBoard))));
287         }

```

### 6.53.3.15 printBoard()

```
void test.driver.HardDifficultyDriver.printBoard (
    Board board ) [private]
```

Definition at line 289 of file HardDifficultyDriver.java.

```
289     {
290         ArrayList<String> boardCodified = this.transcribeToCharacters(board);
291         System.out.println("      0 1 2 3 4 5 6 7");
292         System.out.println("      -----");
293
294         for (Integer i = 0; i < 8; ++i) {
295             String row = boardCodified.get(i);
296             System.out.println("    " + i + " | " + row.charAt(0) + " " + row.charAt(1) + " " +
row.charAt(2) + " "
297 + " "
+ row.charAt(3) + " " + row.charAt(4) + " " + row.charAt(5) + " " + row.charAt(6)
+ " "
+ row.charAt(7) + " " );
298         }
299         System.out.println("\n");
300     }
301 }
```

### 6.53.3.16 transcribeToCharacters()

```
ArrayList<String> test.driver.HardDifficultyDriver.transcribeToCharacters (
    Board board ) [private]
```

Definition at line 303 of file HardDifficultyDriver.java.

```
303     {
304         ArrayList<String> boardCodified = new ArrayList<String>(8);
305         String operational = "";
306         PieceType[][] current = board.getBoard();
307
308         for (int i = 0; i < 8; ++i) {
309             operational = "";
310             for (int j = 0; j < 8; ++j) {
311                 if (current[i][j] == PieceType.PLAYER1)
312                     operational = operational + "B";
313                 if (current[i][j] == PieceType.PLAYER2)
314                     operational = operational + "N";
315                 if (current[i][j] == null)
316                     operational = operational + "?";
317             }
318             boardCodified.add(operational);
319         }
320     }
321
322     return boardCodified;
323 }
```

## 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.

```
86  
87         super("ERR_INCORRECT_CREDENTIALS");  
88     }
```

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.

```
75  
76         super("ERR_INEXISTING_CONFIGURATION");  
77     }
```

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.

```
64         {
65             super("ERR_INEXISTING_PLAYER");
66         }
```

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.

```
36         {  
37     }
```

### 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.

```
403         {  
404             board = ViewCtrl.domainCtrl.viewBoard();  
405             render();  
406     }
```

### 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.

```
413                                     {
414     Pair<Integer, Integer> pos = getClickedPos(mouseEvent);
415     if (placeWhitePieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos, "PLAYER1");
416     else if (placeBlackPieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos,
417     "PLAYER2");
418     else if (quitPieces.isSelected()) board = ViewCtrl.domainCtrl.removePiece(pos);
419     render();
420 }
```

### 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.

```
426                                     {
427     Stage currentWindow = (Stage) save.getScene().getWindow();
428     currentWindow.close();
429 }
```

### 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.

```
436         {
437             for (int i = 0; i < 8; i++) {
438                 char[] row = board.getString(String.format("row%d", i)).toCharArray();
439                 for (int j = 0; j < 8; j++) drawPiece(new Pair<Integer, Integer>(i, j), row[j]);
440             }
441         }
```

### 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.

```
448         {
449             Circle circle = getCircle(pos);
450             switch (pieceType) {
451                 case 'B':
452                     circle.setFill(Color.web("0xFFFFFF", 1.0));
453                     break;
454                 case 'N':
455                     circle.setFill(Color.web("0x000000", 1.0));
456                     break;
457                 case '?':
458                     circle.setFill(Color.web("0x34d399", 1.0));
459                     break;
460                 default:
461                     break;
462             }
463         }
```

### 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.

```
470 {
471     Pair<Integer, Integer> pos = new Pair<Integer, Integer>(-1, -1);
472     String piece = ((Circle) mouseEvent.getPickResult().getIntersectedNode()).getId();
473     pos.first = Character.getNumericValue(piece.charAt(1)) - 1;
474     pos.second = Character.getNumericValue(piece.charAt(3)) - 1;
475     return pos;
476 }
```

### 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.

```
483 {
484     try {
485         Field field = this.getClass().getDeclaredField(String.format("f%s%s", pos.first + 1,
pos.second + 1));
486         field.setAccessible(true);
487         return (Circle) field.get(this);
488     } catch (Exception e) {
489         return new Circle();
490     }
491 }
```

## 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.

```
152 {  
153     super ("ERR_INVALID_BOARD");  
154 }
```

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.

```
196 {  
197     super ("ERR_INVALID_CONFIGURATION");  
198 }
```

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.

```
119 {  
120     super("ERR_INVALID_DIFFICULTY");  
121 }
```

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.

```
31 {  
32     super("ERR_INVALID_NAME");  
33 }
```

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.

```
42
43         super("ERR_INVALID_PASSWORD");
44     }
```

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.

```
185 {  
186     super ("ERR_INVALID_PLAYERS");  
187 }
```

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.

```
174 {  
175     super ("ERR_INVALID_POSITION");  
176 }
```

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.

```
163
164         super ("ERR_INVALID_RULES");
165     }
```

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.

```
30         {
31     }
```

### 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.

```
78         {
79             Pair<JSONObject, String> result = ViewCtrl.domainCtrl.login(username.getText(),
password.getText());
80             if (result.second != null) {
81                 switch (result.second) {
82                     case "ERR_INVALID_NAME":
83                         logInResult.setText("Username can't be empty!");
84                         break;
85                     case "ERR_INVALID_PASSWORD":
86                         logInResult.setText("Password can't be empty!");
87                         break;
88                     case "ERR_INEXISTING_PLAYER":
89                         logInResult.setText("The player does not exist!");
90                         break;
91                     case "ERR_INCORRECT_CREDENTIALS":
92                         logInResult.setText("Wrong username or password!");
93                         break;
94                     default:
95                         logInResult.setText("Something went wrong, try again!");
96                         break;
97                 }
98             } else {
99                 ViewCtrl.changeScene("template/UserView.fxml");
100             }
101     }
```

### 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.

```
108                                     {  
109         ViewCtrl.changeScene("template/SignUpView.fxml");  
110     }
```

## 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 entrypt. 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 entryptoint. 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.

```
22 {  
23     new MediumDifficultyDriver().start();  
24 }
```

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 implicit difficulty, `canEatHorizontally`, `canEatVertically`, `canEatDiagonally` and `pieceType` attributes are setted. The implicit `maxDepth` attribute is setted to the double of the entered difficulty.

## Parameters

<i>difficulty</i>	<a href="#">Difficulty</a> for the Minimax algorithm with alpha-beta pruning.
<i>canEatHorizontally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten horizontally.
<i>canEatVertically</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten vertically.
<i>canEatDiagonally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten diagonally.
<i>pieceType</i>	<a href="#">Player</a> that wants to be maximized.

Definition at line 34 of file MediumDifficulty.java.

```
35                                     {
36         super(difficulty, canEatHorizontally, canEatVertically, canEatDiagonally, pieceType);
37     }
```

## 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

<i>currentBoard</i>	Current playing <a href="#">Board</a> to get the heuristic evaluation from.
---------------------	---

## Returns

The heuristic evaluation for the given [Board](#) state.

Definition at line 51 of file MediumDifficulty.java.

```
51                                     {
52     int player1 = currentBoard.getPiecesPlayer1();
53     int player2 = currentBoard.getPiecesPlayer2();
54
55     PieceType[][] board = currentBoard.getBoard();
56 }
```

```

57     // Check corners of the Board
58     if (board[0][0] == PieceType.PLAYER1) player1 += 50;
59     else if (board[0][0] == PieceType.PLAYER2) player2 += 50;
60
61     if (board[0][7] == PieceType.PLAYER1) player1 += 50;
62     else if (board[0][7] == PieceType.PLAYER2) player2 += 50;
63
64     if (board[7][0] == PieceType.PLAYER1) player1 += 50;
65     else if (board[7][0] == PieceType.PLAYER2) player2 += 50;
66
67     if (board[7][7] == PieceType.PLAYER1) player1 += 50;
68     else if (board[7][7] == PieceType.PLAYER2) player2 += 50;
69
70     // Check borders not next to corner
71     for (int k = 2; k < 6; ++k) {
72         if (board[k][0] == PieceType.PLAYER1) player1 += 17;
73         else if (board[k][0] == PieceType.PLAYER2) player2 += 17;
74
75         if (board[k][7] == PieceType.PLAYER1) player1 += 17;
76         else if (board[k][7] == PieceType.PLAYER2) player2 += 17;
77
78         if (board[0][k] == PieceType.PLAYER1) player1 += 17;
79         else if (board[0][k] == PieceType.PLAYER2) player2 += 17;
80
81         if (board[7][k] == PieceType.PLAYER1) player1 += 17;
82         else if (board[7][k] == PieceType.PLAYER2) player2 += 17;
83     }
84
85     // Check next to center of the Board
86     for (int i = 2; i < 6; ++i) {
87         if (board[i][2] == PieceType.PLAYER1) player1 += 10;
88         else if (board[i][2] == PieceType.PLAYER2) player2 += 10;
89
90         if (board[i][5] == PieceType.PLAYER1) player1 += 10;
91         else if (board[i][5] == PieceType.PLAYER2) player2 += 10;
92
93         if (board[2][i] == PieceType.PLAYER1) player1 += 10;
94         else if (board[2][i] == PieceType.PLAYER2) player2 += 10;
95
96         if (board[5][i] == PieceType.PLAYER1) player1 += 10;
97         else if (board[5][i] == PieceType.PLAYER2) player2 += 10;
98     }
99
100    // Check next to corners
101    for (int j = 0; j < 2; ++j) {
102        if (board[1][j] == PieceType.PLAYER1) player1 -= 25;
103        else if (board[1][j] == PieceType.PLAYER2) player2 -= 25;
104
105        if (board[1][7 - j] == PieceType.PLAYER1) player1 -= 25;
106        else if (board[1][7 - j] == PieceType.PLAYER2) player2 -= 25;
107
108        if (board[6][j] == PieceType.PLAYER1) player1 -= 25;
109        else if (board[6][j] == PieceType.PLAYER2) player2 -= 25;
110
111        if (board[6][7 - j] == PieceType.PLAYER1) player1 -= 25;
112        else if (board[6][7 - j] == PieceType.PLAYER2) player2 -= 25;
113    }
114
115    if (board[0][1] == PieceType.PLAYER1) player1 -= 25;
116    else if (board[0][1] == PieceType.PLAYER2) player2 -= 25;
117
118    if (board[7][1] == PieceType.PLAYER1) player1 -= 25;
119    else if (board[7][1] == PieceType.PLAYER2) player2 -= 25;
120
121    if (board[0][6] == PieceType.PLAYER1) player1 -= 25;
122    else if (board[0][6] == PieceType.PLAYER2) player2 -= 25;
123
124    if (board[7][6] == PieceType.PLAYER1) player1 -= 25;
125    else if (board[7][6] == PieceType.PLAYER2) player2 -= 25;
126
127    if (this.pieceType == PieceType.PLAYER1) return player1 - player2;
128    else return player2 - player1;
129 }

```

### 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

<i>currentBoard</i>	current <a href="#">Board</a> in the tree of possibilities.
<i>currentPieceType</i>	current turn in the tree of possibilities.
<i>depth</i>	current depth in the tree of possibilities.
<i>alpha</i>	current alpha in the tree of possibilities.
<i>beta</i>	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.

```

144
145 {
146     ArrayList<Pair<Integer, Integer> validPositions = currentBoard.validPositions(currentPieceType,
147     this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);
148
149     if (validPositions.isEmpty() || depth == 0)
150         return this.evaluation(currentBoard);
151
152     // Maximizer
153     if (currentPieceType == this.pieceType) {
154         int max = Integer.MIN_VALUE, currentMax = 0;
155
156         for (Pair<Integer, Integer> position : validPositions) {
157             // Make a duplicate in order not to work with the same Board pointer!
158             Board board = new Board(currentBoard.getBoard());
159             board.placePiece(position, currentPieceType, this.canEatHorizontally,
160             this.canEatVertically,
161             this.canEatDiagonally);
162             currentMax = this.minimax(board, MediumDifficulty.inversePieceType(currentPieceType),
163             depth - 1, alpha, beta);
164             max = Integer.max(max, currentMax);
165             alpha = Integer.max(alpha, currentMax);
166             // Prune
167             if (beta <= alpha)
168                 break;
169         }
170     }
171     return max;
172
173     // Minimizer
174     else {
175         Integer min = Integer.MAX_VALUE, currentMin = 0;
176
177         for (Pair<Integer, Integer> position : validPositions) {
178             // Make a duplicate in order not to work with the same Board pointer!

```

```

178         Board board = new Board(currentBoard.getBoard());
179         board.placePiece(position, currentPieceType, this.canEatHorizontally,
180             this.canEatVertically,
181                 this.canEatDiagonally);
182         currentMin = this.minimax(board, MediumDifficulty.inversePieceType(currentPieceType),
183             depth - 1, alpha, beta);
183         min = Integer.min(min, currentMin);
184         beta = Integer.min(beta, currentMin);
185         // Prune
186         if (beta <= alpha)
187             break;
188     }
189     return min;
190 }
191 }
192 }

```

### 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

<i>playingBoard</i>	Current playing <a href="#">Board</a> .
---------------------	---

#### 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.

```

203
204     Pair<Integer, Integer> bestPosition = null;
205
206     Board initialBoard = new Board(playingBoard);
207
208     ArrayList<Pair<Integer, Integer>> validPositions = initialBoard.validPositions(this.pieceType,
209         this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);
210
211     int max = Integer.MIN_VALUE, currentMax = 0;
212
213     for (Pair<Integer, Integer> position : validPositions) {
214         // Make a duplicate in order not to work with the same Board pointer!
215         Board board = new Board(initialBoard.getBoard());
216         board.placePiece(position, this.pieceType, this.canEatHorizontally, this.canEatVertically,
217             this.canEatDiagonally);
218     }

```

```

219         currentMax = this.minimax(board, MediumDifficulty.inversePieceType(this.pieceType),
220             this.maxDepth - 1, Integer.MIN_VALUE, Integer.MAX_VALUE);
221         if (currentMax > max) {
222             max = currentMax;
223             bestPosition = position;
224         }
225     }
226
227     return bestPosition;
228 }

```

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.

```
33         {  
34             this.currentMediumDifficulty = null;  
35             this.fixtureRepository = new FixtureRepository();  
36         }
```

### 6.69.3 Member Function Documentation

#### 6.69.3.1 start()

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

Definition at line 40 of file MediumDifficultyDriver.java.

```
40         {  
41             while (true) {  
42                 this.mainMenu();  
43             }  
44         }
```

### 6.69.3.2 mainMenu()

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

Definition at line 46 of file MediumDifficultyDriver.java.

```

46     {
47         String title = null;
48         if (this.currentMediumDifficulty != null)
49             title = String.format("Current maximum depth: %s\n",
this.currentMediumDifficulty.getMaxDepth());
50         if (this.currentBoard != null)
51             title += String.format("Current Board: %s\n", this.nameCurrentBoard);
52
53         switch (Driver.menu(title, "MediumDifficulty (Minimax with alpha-beta pruning) Driver",
54             new Pair<String, String>("1", "Create MediumDifficulty"),
55             new Pair<String, String>("2", "Get difficulty"),
56             new Pair<String, String>("3", "Get canEatHorizontally"),
57             new Pair<String, String>("4", "Get canEatVertically"),
58             new Pair<String, String>("5", "Get canEatDiagonally"),
59             new Pair<String, String>("6", "Get pieceType"),
60             new Pair<String, String>("7", "Get maxDepth"),
61             new Pair<String, String>("8", "Set maxDepth"),
62             new Pair<String, String>("9", "Load Board"),
63             new Pair<String, String>("10", "Print Current Board"),
64             new Pair<String, String>("11", "Get next best position"))) {
65             case "1":
66                 this.create();
67                 break;
68             case "2":
69                 this.getDifficulty();
70                 break;
71             case "3":
72                 this.getCanEatHorizontally();
73                 break;
74             case "4":
75                 this.getCanEatVertically();
76                 break;
77             case "5":
78                 this.getCanEatDiagonally();
79                 break;
80             case "6":
81                 this.getPieceType();
82                 break;
83             case "7":
84                 this.getMaxDepth();
85                 break;
86             case "8":
87                 this.setMaxDepth();
88                 break;
89             case "9":
90                 this.loadBoard();
91                 break;
92             case "10":
93                 this.printCurrentBoard();
94                 break;
95             case "11":
96                 this.getNextBestPosition();
97                 break;
98             }
99         Driver.pause();
100     }
```

### 6.69.3.3 create()

```
void test.driver.MediumDifficultyDriver.create ( ) [private]
```

Definition at line 102 of file MediumDifficultyDriver.java.

```

102     {
103         System.out.println(
104             "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");
105
106         Integer difficulty = Driver.inputInt("Difficulty (positive)?");
107         Boolean canEatHorizontally = Driver.inputBool("Can eat horizontally?");
108         Boolean canEatVertically = Driver.inputBool("Can eat vertically?");
```

```

109         Boolean canEatDiagonally = Driver.inputBool("Can eat diagonally?");
110         PieceType pieceType = null;
111
112         switch (Driver.menu(null, "Select Bot pieces",
113             new Pair<String, String>("1", "PLAYER 1 pieces"),
114             new Pair<String, String>("2", "PLAYER 2 pieces"))) {
115             case "1":
116                 pieceType = PieceType.PLAYER1;
117                 break;
118             case "2":
119                 pieceType = PieceType.PLAYER2;
120                 break;
121         }
122
123         this.currentMediumDifficulty = new MediumDifficulty(difficulty, canEatHorizontally,
124             canEatVertically,
125             canEatDiagonally, pieceType);
126         System.out.println(String.format("MediumDifficulty with a maximum depth of %s created
127             successfully!",
128             this.currentMediumDifficulty.getMaxDepth()));
129     }

```

#### 6.69.3.4 getDifficulty()

void test.driver.MediumDifficultyDriver.getDifficulty ( ) [private]

Definition at line 130 of file MediumDifficultyDriver.java.

```

130         {
131             if (this.currentMediumDifficulty == null) {
132                 System.out.println("No current MediumDifficulty!");
133                 return;
134             }
135
136             System.out.println(
137                 String.format("MediumDifficulty's difficulty is: %s",
138                     this.currentMediumDifficulty.getDifficulty()));
139         }

```

#### 6.69.3.5 getCanEatHorizontally()

void test.driver.MediumDifficultyDriver.getCanEatHorizontally ( ) [private]

Definition at line 140 of file MediumDifficultyDriver.java.

```

140         {
141             if (this.currentMediumDifficulty == null) {
142                 System.out.println("No current MediumDifficulty!");
143                 return;
144             }
145
146             System.out.println(String.format("MediumDifficulty's canEatHorizontally is: %s",
147                 this.currentMediumDifficulty.getCanEatHorizontally()));
148         }

```

#### 6.69.3.6 getCanEatVertically()

void test.driver.MediumDifficultyDriver.getCanEatVertically ( ) [private]

Definition at line 150 of file MediumDifficultyDriver.java.

```

150         {
151             if (this.currentMediumDifficulty == null) {
152                 System.out.println("No current MediumDifficulty!");
153                 return;
154             }
155
156             System.out.println(String.format("MediumDifficulty's canEatVertically is: %s",
157                 this.currentMediumDifficulty.getCanEatVertically()));
158         }

```

### 6.69.3.7 getCanEatDiagonally()

```
void test.driver.MediumDifficultyDriver.getCanEatDiagonally ( ) [private]
```

Definition at line 160 of file MediumDifficultyDriver.java.

```
160         {
161             if (this.currentMediumDifficulty == null) {
162                 System.out.println("No current MediumDifficulty!");
163                 return;
164             }
165             System.out.println(String.format("MediumDifficulty's canEatDiagonally is: %s",
166                 this.currentMediumDifficulty.getCanEatDiagonally()));
167         }
168     }
```

### 6.69.3.8 getPieceType()

```
void test.driver.MediumDifficultyDriver.getPieceType ( ) [private]
```

Definition at line 170 of file MediumDifficultyDriver.java.

```
170         {
171             if (this.currentMediumDifficulty == null) {
172                 System.out.println("No current MediumDifficulty!");
173                 return;
174             }
175             System.out.println(
176                 String.format("MediumDifficulty's pieceType is: %s",
177                     this.currentMediumDifficulty.getPieceType()));
178         }
```

### 6.69.3.9 getMaxDepth()

```
void test.driver.MediumDifficultyDriver.getMaxDepth ( ) [private]
```

Definition at line 180 of file MediumDifficultyDriver.java.

```
180         {
181             if (this.currentMediumDifficulty == null) {
182                 System.out.println("No current MediumDifficulty!");
183                 return;
184             }
185             System.out.println(
186                 String.format("MediumDifficulty's maxDepth is: %s",
187                     this.currentMediumDifficulty.getMaxDepth()));
188         }
```

### 6.69.3.10 setMaxDepth()

```
void test.driver.MediumDifficultyDriver.setMaxDepth ( ) [private]
```

Definition at line 190 of file MediumDifficultyDriver.java.

```
190         {
191             if (this.currentMediumDifficulty == null) {
192                 System.out.println("No current MediumDifficulty!");
193                 return;
194             }
195             System.out.println(
196                 "Take into account that minimax with alpha-beta pruning with higher depths requires more
197                 time to execute. A value of 7 is reasonable.\n");
198             this.currentMediumDifficulty.setMaxDepth(Driver.inputInt("Maximum depth (positive)?"));
199             System.out.println("MediumDifficulty's maxDepth changed successfully!");
200         }
201     }
```

### 6.69.3.11 loadBoard()

```
void test.driver.MediumDifficultyDriver.loadBoard ( ) [private]
```

Definition at line 203 of file MediumDifficultyDriver.java.

```
203         {
204             if (this.currentMediumDifficulty == null) {
205                 System.out.println("No current MediumDifficulty!");
206                 return;
207             }
208
209             Pair<String, Board> selectedBoard = this.listBoardFixtures();
210
211             this.nameCurrentBoard = selectedBoard.first;
212             this.currentBoard = selectedBoard.second;
213
214             System.out.println(String.format("Board %s loaded successfully!\n", this.nameCurrentBoard));
215             this.printBoard(this.currentBoard);
216         }
```

### 6.69.3.12 printCurrentBoard()

```
void test.driver.MediumDifficultyDriver.printCurrentBoard ( ) [private]
```

Definition at line 218 of file MediumDifficultyDriver.java.

```
218         {
219             if (this.currentMediumDifficulty == null) {
220                 System.out.println("No current MediumDifficulty!");
221                 return;
222             }
223
224             if (this.currentBoard == null) {
225                 System.out.println("No current Board!");
226                 return;
227             }
228
229             System.out.println(String.format("Board %s printed successfully!\n", this.nameCurrentBoard));
230             this.printBoard(this.currentBoard);
231         }
```

### 6.69.3.13 getNextBestPosition()

```
void test.driver.MediumDifficultyDriver.getNextBestPosition ( ) [private]
```

Definition at line 233 of file MediumDifficultyDriver.java.

```
233         {
234             if (this.currentMediumDifficulty == null) {
235                 System.out.println("No current MediumDifficulty!");
236                 return;
237             }
238
239             if (this.currentBoard == null) {
240                 System.out.println("No current Board!");
241                 return;
242             }
243
244             System.out.println("Take into account that the state of the current Board won't be globally
modified.\n");
245
246             this.printBoard(this.currentBoard);
247
248             long startTime = System.currentTimeMillis();
249             Pair<Integer, Integer> nextBestPosition =
this.currentMediumDifficulty.place(this.currentBoard.getBoard());
250             long durationTime = System.currentTimeMillis() - startTime;
251
252             Board tempBoard = new Board(this.currentBoard.getBoard());
```



```

253
254     if (nextBestPosition != null) {
255         tempBoard.placePiece(nextBestPosition, this.currentMediumDifficulty.getPieceType(),
256             this.currentMediumDifficulty.getCanEatHorizontally(),
257             this.currentMediumDifficulty.getCanEatVertically(),
258             this.currentMediumDifficulty.getCanEatDiagonally());
259         System.out.println(
260             String.format("The best position calculated in %s ms is %s\n", durationTime,
nextBestPosition));
261         System.out.println("The addition of the piece would look like this:\n");
262         this.printBoard(tempBoard);
263     } else {
264         System.out.println("There isn't any possible position left to place a piece on.");
265     }
266 }

```

### 6.69.3.14 listBoardFixtures()

```
Pair<String, Board> test.driver.MediumDifficultyDriver.listBoardFixtures ( ) [private]
```

Definition at line 268 of file MediumDifficultyDriver.java.

```

268     {
269         Integer selectedBoard = -1;
270         ArrayList<String> listBoards = this.fixtureRepository.listFiles();
271
272         while (selectedBoard < 0 || selectedBoard >= listBoards.size()) {
273             Driver.clear();
274             System.out.println("==== Available Boards ====\n");
275
276             for (Integer i = 0; i < listBoards.size(); ++i)
277                 System.out.println(String.format("[%d]\t%s", i, listBoards.get(i)));
278             System.out.println("");
279
280             selectedBoard = Driver.inputInt("What Board would you like to load?");
281         }
282
283         Driver.clear();
284
285         return new Pair<String, Board>(listBoards.get(selectedBoard),
286             new Board(this.fixtureRepository.boardFileToJSON(listBoards.get(selectedBoard))));
287     }

```

### 6.69.3.15 printBoard()

```
void test.driver.MediumDifficultyDriver.printBoard (
    Board board ) [private]
```

Definition at line 289 of file MediumDifficultyDriver.java.

```

289     {
290         ArrayList<String> boardCodified = this.transcribeToCharacters(board);
291         System.out.println("      0 1 2 3 4 5 6 7");
292         System.out.println("-----");
293
294         for (Integer i = 0; i < 8; ++i) {
295             String row = boardCodified.get(i);
296             System.out.println(" " + i + " | " + row.charAt(0) + " " + row.charAt(1) + " " +
row.charAt(2) + " "
297                 + row.charAt(3) + " " + row.charAt(4) + " " + row.charAt(5) + " " + row.charAt(6)
+ " "
298                 + row.charAt(7) + " ");
299         }
300         System.out.println("\n");
301     }

```

### 6.69.3.16 transcribeToCharacters()

```
ArrayList<String> test.driver.MediumDifficultyDriver.transcribeToCharacters (
    Board board ) [private]
```

Definition at line 303 of file MediumDifficultyDriver.java.

```
303 {
304     ArrayList<String> boardCodified = new ArrayList<String>(8);
305     String operational = "";
306     PieceType[][] current = board.getBoard();
307
308     for (int i = 0; i < 8; ++i) {
309         operational = "";
310         for (int j = 0; j < 8; ++j) {
311             if (current[i][j] == PieceType.PLAYER1)
312                 operational = operational + "B";
313             if (current[i][j] == PieceType.PLAYER2)
314                 operational = operational + "N";
315             if (current[i][j] == null)
316                 operational = operational + "?";
317         }
318         boardCodified.add(operational);
319     }
320 }
321
322 return boardCodified;
323 }
```

## 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.

```
401         {  
402             board = ViewCtrl.domainCtrl.viewBoard();  
403             render();  
404         }
```

### 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.

```
411         {  
412             Pair<Integer, Integer> pos = getClickedPos(mouseEvent);  
413             if (placeWhitePieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos, "PLAYER1");  
414             else if (placeBlackPieces.isSelected()) board = ViewCtrl.domainCtrl.placePieceConfig(pos,  
415                 "PLAYER2");  
416             else if (quitPieces.isSelected()) board = ViewCtrl.domainCtrl.removePiece(pos);  
417             render();  
418         }
```

### 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.

```
446
447     Circle circle = getCircle(pos);
448     switch (pieceType) {
449         case 'B':
450             circle.setFill(Color.web("0xFFFFFF", 1.0));
451             break;
452         case 'N':
453             circle.setFill(Color.web("0x000000", 1.0));
454             break;
455         case '?':
456             circle.setFill(Color.web("0x34d399", 1.0));
457             break;
458         default:
459             break;
460     }
461 }
```

**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.

```
468
469     Pair<Integer, Integer> pos = new Pair<Integer, Integer>(-1, -1);
470     String piece = ((Circle) mouseEvent.getPickResult().getIntersectedNode()).getId();
471     pos.first = Character.getNumericValue(piece.charAt(1)) - 1;
472     pos.second = Character.getNumericValue(piece.charAt(3)) - 1;
473     return pos;
474 }
```

### 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.

```
481                                     {
482         try {
483             Field field = this.getClass().getDeclaredField(String.format("f%s%c", pos.first + 1,
pos.second + 1));
484             field.setAccessible(true);
485             return (Circle) field.get(this);
486         } catch (Exception e) {
487             return new Circle();
488         }
489     }
```

## 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.

```
97 {  
98     super("ERR_NOT_CREATOR");  
99 }
```

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.

```
207     {  
208         super ("ERR_NOT_PLAYER");  
209     }
```

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.

```
218     {  
219         super ("ERR_NOT_PLAYER_PIECE");  
220     }
```

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.

```
229                                     {  
230         super ("ERR_NOT_PLAYER_TURN");  
231     }
```

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.

```
251 {  
252     super("ERR_NOT_STARTED_GAME");  
253 }
```

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.

```
57 {  
58     ViewCtrl.main(args);  
59 }
```

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

- [othello.java](#)

## 6.77 cmd.driver.pair Class Reference

Pair driver endpoint. 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 endpoint. 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.

```
22                                     {  
23         new PairDriver().start();  
24     }
```

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

<i>first</i>	First value of the <a href="#">Pair</a> .
<i>second</i>	Second value of the <a href="#">Pair</a> .

Definition at line 36 of file Pair.java.

```
36      {
37          this.first = first;
38          this.second = second;
39      }
```

## 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

<i>object</i>	<a href="#">Pair</a> to be compared.
---------------	--------------------------------------

## Returns

Whether the implicit [Pair](#) and the given [Pair](#) are equal.

Definition at line 51 of file Pair.java.

```
51      {
52          if (!(object instanceof Pair)) {
53              return false;
54          }
55          Pair<?, ?> other = (Pair<?, ?>) object;
56          return Objects.equals(other.first, this.first) && Objects.equals(other.second, this.second);
57      }
```

### 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.

```
66      {
67          return String.format("Pair<%s, %s>", this.first, this.second);
68      }
```

**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.

```
76      {
77          return this.first;
78      }
```

**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.

```
86      {
87          return this.second;
88      }
```



## 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.

```
24     {
25         this.currentStrPair = null;
26         this.currentIntPair = null;
27         this.currentStrIntPair = null;
28     }
```

### 6.79.3 Member Function Documentation

#### 6.79.3.1 start()

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

Definition at line 32 of file PairDriver.java.

```
32     {
33         while (true) {
34             this.mainMenu();
35         }
36     }
```

### 6.79.3.2 mainMenu()

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

Definition at line 38 of file PairDriver.java.

```
38         {
39             String title = (this.currentPair() != null ? String.format("Current: %s\n", this.currentPair()) :
        null);
40             switch (Driver.menu(title, "Pair Driver",
41                 new Pair<String, String>("1", "Create String:String Pair"),
42                 new Pair<String, String>("2", "Create Integer:Integer Pair"),
43                 new Pair<String, String>("3", "Create String:Integer Pair"),
44                 new Pair<String, String>("4", "Get first"),
45                 new Pair<String, String>("5", "Get second"),
46                 new Pair<String, String>("6", "Compare current Pair with another"))) {
47                 case "1":
48                     this.createStrPair();
49                     break;
50                 case "2":
51                     this.createIntPair();
52                     break;
53                 case "3":
54                     this.createStrIntPair();
55                     break;
56                 case "4":
57                     this.getFirst();
58                     break;
59                 case "5":
60                     this.getSecond();
61                     break;
62                 case "6":
63                     this.comparePair();
64                     break;
65             }
66             Driver.pause();
67     }
```

### 6.79.3.3 currentPair()

```
Object test.driver.PairDriver.currentPair ( ) [private]
```

Definition at line 69 of file PairDriver.java.

```
69         {
70             if (this.currentStrPair != null)
71                 return this.currentStrPair;
72
73             if (this.currentIntPair != null)
74                 return this.currentIntPair;
75
76             if (this.currentStrIntPair != null)
77                 return this.currentStrIntPair;
78
79             return null;
80     }
```

### 6.79.3.4 resetPairs()

```
void test.driver.PairDriver.resetPairs ( ) [private]
```

Definition at line 82 of file PairDriver.java.

```
82         {
83             this.currentStrPair = null;
84             this.currentIntPair = null;
85             this.currentStrIntPair = null;
86     }
```

### 6.79.3.5 createStrPair()

```
void test.driver.PairDriver.createStrPair ( ) [private]
```

Definition at line 88 of file PairDriver.java.

```
88         {
89             this.resetPairs();
90             this.currentStrPair = new Pair<String, String>(Driver.input("First value?"), Driver.input("Second
value?"));
91             System.out.println(String.format("%s created successfully!", this.currentStrPair));
92         }
```

### 6.79.3.6 createIntPair()

```
void test.driver.PairDriver.createIntPair ( ) [private]
```

Definition at line 94 of file PairDriver.java.

```
94         {
95             this.resetPairs();
96             this.currentIntPair = new Pair<Integer, Integer>(Driver.inputInt("First value?"),
97                 Driver.inputInt("Second value?"));
98             System.out.println(String.format("%s created successfully!", this.currentIntPair));
99         }
```

### 6.79.3.7 createStrIntPair()

```
void test.driver.PairDriver.createStrIntPair ( ) [private]
```

Definition at line 101 of file PairDriver.java.

```
101        {
102            this.resetPairs();
103            this.currentStrIntPair = new Pair<String, Integer>(Driver.input("First value?"),
104                Driver.inputInt("Second value?"));
105            System.out.println(String.format("%s created successfully!", this.currentStrIntPair));
106        }
```

### 6.79.3.8 getFirst()

```
void test.driver.PairDriver.getFirst ( ) [private]
```

Definition at line 108 of file PairDriver.java.

```
108        {
109            if (this.currentPair() == null) {
110                System.out.println("No current Pair!");
111                return;
112            }
113
114            System.out.print(String.format("%s's first is: ", this.currentPair()));
115
116            if (this.currentStrPair != null)
117                System.out.println(this.currentStrPair.first);
118
119            if (this.currentIntPair != null)
120                System.out.println(this.currentIntPair.first);
121
122            if (this.currentStrIntPair != null)
123                System.out.println(this.currentStrIntPair.first);
124        }
```

### 6.79.3.9 getSecond()

```
void test.driver.PairDriver.getSecond ( ) [private]
```

Definition at line 126 of file PairDriver.java.

```

126         {
127             if (this.currentPair() == null) {
128                 System.out.println("No current Pair!");
129                 return;
130             }
131
132             System.out.print(String.format("%s's second is: ", this.currentPair()));
133
134             if (this.currentStrPair != null)
135                 System.out.println(this.currentStrPair.second);
136
137             if (this.currentIntPair != null)
138                 System.out.println(this.currentIntPair.second);
139
140             if (this.currentStrIntPair != null)
141                 System.out.println(this.currentStrIntPair.second);
142         }

```

### 6.79.3.10 comparePair()

```
void test.driver.PairDriver.comparePair ( ) [private]
```

Definition at line 144 of file PairDriver.java.

```

144         {
145             if (this.currentPair() == null) {
146                 System.out.println("No current Pair!");
147                 return;
148             }
149
150             if (this.currentStrPair != null) {
151                 Pair<String, String> toCompare = new Pair<String, String>(Driver.input("First value of Pair
to compare?"),
152                                     Driver.input("Second value of Pair to compare?"));
153                 if (this.currentStrPair.equals(toCompare))
154                     System.out.println(String.format("%s and %s are equal", this.currentStrPair,
toCompare));
155                 else
156                     System.out.println(String.format("%s and %s are not equal", this.currentStrPair,
toCompare));
157             }
158
159             if (this.currentIntPair != null) {
160                 Pair<Integer, Integer> toCompare = new Pair<Integer, Integer>(
161                     Driver.inputInt("First value of Pair to compare?"),
162                     Driver.inputInt("Second value of Pair to compare?"));
163                 if (this.currentIntPair.equals(toCompare))
164                     System.out.println(String.format("%s and %s are equal", this.currentIntPair,
toCompare));
165                 else
166                     System.out.println(String.format("%s and %s are not equal", this.currentIntPair,
toCompare));
167             }
168
169             if (this.currentStrIntPair != null) {
170                 Pair<String, Integer> toCompare = new Pair<String, Integer>(Driver.input("First value of
Pair to compare?"),
171                                     Driver.inputInt("Second value of Pair to compare?"));
172                 if (this.currentStrIntPair.equals(toCompare))
173                     System.out.println(String.format("%s and %s are equal", this.currentStrIntPair,
toCompare));
174                 else
175                     System.out.println(String.format("%s and %s are not equal", this.currentStrIntPair,
toCompare));
176             }
177         }

```

## 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.

```
32     {  
33         return this.name;  
34     }
```

#### 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.

```
40     {  
41         return this.id;  
42     }
```



### 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.

```
49                                     {
50         return this.isDeleted;
51     }
```

### 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.

```
60                                     {
61         if(name.isBlank()) {
62             throw new InvalidNameException();
63         }
64         this.name = name;
65     }
```

### 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.

```
71                                     {
72         this.isDeleted = isDeleted;
73     }
```

## 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](#), [InvalidPasswordException](#), [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](#), [InvalidDifficultyException](#), [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 [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.*
- [Bot modifyBot](#) (UUID botID, String name, Integer difficulty, UUID modifierID) throws [InvalidNameException](#), [InvalidDifficultyException](#), [ExistingPlayerException](#), [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.*
- [void deleteUser](#) (UUID userID, String password) throws [IncorrectCredentialsException](#), [InexistingPlayerException](#)  
*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](#), [BotUsedException](#)  
*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](#), [InvalidPasswordException](#)  
*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](#), [InvalidDifficultyException](#)  
*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.

```
58     {
59         this.repositoryCtrl = new PlayerRepositoryCtrl();
60         this.gameRepositoryCtrl = new GameRepositoryCtrl();
61     }
```

### 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

<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of a <a href="#">User</a>
<i>confirmation</i>	Confirmation of the entered password

##### Postcondition

`saveUser` is called and a saved new user is returned.

Definition at line 77 of file `PlayerCtrl.java`.

```
78
79         {
80             if (!password.equals(confirmation))
81                 throw new BadConfirmationException();
82             if (this.repositoryCtrl.getByName(name) != null)
83                 throw new ExistingPlayerException();
84             return this.saveUser(name, hash(password), UUID.randomUUID());
85         }
```

### 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

<i>name</i>	Name of the <a href="#">Bot</a>
<i>difficulty</i>	Difficulty of the <a href="#">Bot</a>
<i>creatorID</i>	UUID of a <a href="#">Player</a>

**Postcondition**

saveBot is called and a saved new bot is returned.

Definition at line 96 of file PlayerCtrl.java.

```

97                                     {
98         if (this.repositoryCtrl.getByName(name) != null)
99             throw new ExistingPlayerException();
100        return this.saveBot(name, difficulty, UUID.randomUUID(), creatorID);
101    }
```

**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**

<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of a <a href="#">User</a>
<i>id</i>	UUID of a <a href="#">User</a>

**Postcondition**

[User](#) is saved in the users' list at repository and returned.

Definition at line 113 of file PlayerCtrl.java.

```

113
114     {
115         if (name.isBlank())
116             throw new InvalidNameException();
117         if (password.isBlank())
118             throw new InvalidPasswordException();
119         User user = new User(name, password, id);
120         this.repositoryCtrl.save(user.serialize());
121         return user;
122     }
```

**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**

<i>name</i>	Name of the <a href="#">Bot</a>
<i>difficulty</i>	<a href="#">Difficulty</a> of the <a href="#">Bot</a>
<i>id</i>	UUID of the <a href="#">Bot</a>
<i>creatorID</i>	UUID of a <a href="#">Player</a>

**Postcondition**[Bot](#) is saved in the bots' list and returned.

Definition at line 134 of file PlayerCtrl.java.

```
135                                     {
136         if (name.isBlank())
137             throw new InvalidNameException();
138         if (difficulty < 1 || difficulty > 10)
139             throw new InvalidDifficultyException();
140         Bot bot = new Bot(name, difficulty, id, creatorID);
141         this.repositoryCtrl.save(bot.serialize());
142         return bot;
143     }
```

**6.82.3.5 login()**

```
User domain.PlayerCtrl.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.

**Precondition***True***Parameters**

<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of a <a href="#">User</a>

**Postcondition**

The user found in the repository is returned.

Definition at line 153 of file PlayerCtrl.java.

```
154                                     {
155         if (name.isBlank())
156             throw new InvalidNameException();
157         if (password.isBlank())
```

```

158         throw new InvalidPasswordException();
159
160         JSONObject rawUser = this.repositoryCtrl.getByName(name);
161         if (rawUser == null)
162             throw new InexistingPlayerException();
163         if (rawUser.getString("type").equals("BOT"))
164             throw new InexistingPlayerException();
165
166         User user = new User(rawUser);
167         if (user.getIsDeleted())
168             throw new InexistingPlayerException();
169
170         if (!user.getPassword().equals(hash(password)))
171             throw new IncorrectCredentialsException();
172
173         return user;
174     }

```

### 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

<i>userID</i>	UUID of a <code>User</code>
---------------	-----------------------------

##### Postcondition

`User` is found in repository and returned.

Definition at line 185 of file PlayerCtrl.java.

```

185
186     JSONObject user = this.repositoryCtrl.get(userID);
187     if (user == null)
188         throw new InexistingPlayerException();
189     if (user.getString("type").equals("BOT"))
190         throw new InexistingPlayerException();
191     return new User(user);
192 }

```

### 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

<i>botID</i>	UUID of the <a href="#">Bot</a>
--------------	---------------------------------

## Postcondition

[Bot](#) is found in repository and returned.

Definition at line 201 of file PlayerCtrl.java.

```
201
202     JSONObject bot = this.repositoryCtrl.get(botID);
203     if (bot == null)
204         throw new InexistingPlayerException();
205     if (bot.getString("type").equals("USER"))
206         throw new InexistingPlayerException();
207     return new Bot(bot);
208 }
```

### 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.

```
216
217     return this.repositoryCtrl.listUsers();
218 }
```

### 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.

```
226
227     return this.repositoryCtrl.listBots();
228 }
```

### 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

<i>userid</i>	UUID of a <a href="#">User</a>
<i>name</i>	Name of a <a href="#">User</a>
<i>password</i>	Password of <a href="#">User</a>
<i>confirmation</i>	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.

```
244
245         User original = this.getUser(userID);
246
247         if (name != null) {
248             if (name.isBlank())
249                 throw new InvalidNameException();
250             if (!original.getName().equals(name) && this.repositoryCtrl.getByName(name) != null)
251                 throw new ExistingPlayerException();
252             original.setName(name);
253         }
254
255         if (password != null) {
256             if (password.isBlank())
257                 throw new InvalidPasswordException();
258             if (!password.equals(confirmation))
259                 throw new BadConfirmationException();
260             original.setPassword(hash(password));
261         }
262
263         return this.saveUser(original.getName(), original.getPassword(), original.getID());
264     }
```

### 6.82.3.11 modifyBot()

```
Bot domain.PlayerCtrl.modifyBot (
    UUID botID,
    String name,
```

```
Integer difficulty,
UUID modifierID ) throws InvalidNameException, InvalidDifficultyException, ExistingPlayerException,
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

<i>name</i>	Name of the Bot
<i>difficulty</i>	Difficulty of the Bot
<i>botID</i>	UUID of the Bot
<i>modifierID</i>	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.

```
280                                     {
281     Bot original = this.getBot(botID);
282
283     if (!original.getCreatorID().equals(modifierID))
284         throw new NotCreatorException();
285
286     if (name != null) {
287         if (name.isBlank())
288             throw new InvalidNameException();
289         if (!original.getName().equals(name) && this.repositoryCtrl.getByName(name) != null)
290             throw new ExistingPlayerException();
291         original.setName(name);
292     }
293
294     if (difficulty != null) {
295         if (difficulty < 1 || difficulty > 10)
296             throw new InvalidDifficultyException();
297         original.setDifficulty(difficulty);
298     }
299
300     if (this.gameRepositoryCtrl.existsGameByPlayerID(botID))
301         throw new BotUsedException();
302
303     return this.saveBot(original.getName(), original.getDifficulty(), original.getID(), modifierID);
304 }
```

#### 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.

#### DELETTERS

#### Precondition

*True*

**Parameters**

<i>userID</i>	UUID of a <a href="#">User</a>
<i>password</i>	Passowrd of a <a href="#">User</a>

**Postcondition**

The user is deleted from the repository.

Definition at line 317 of file PlayerCtrl.java.

```

318                                     {
319         User user = this.getUser(userID);
320
321         if (!user.getPassword().equals(hash(password)))
322             throw new IncorrectCredentialsException();
323
324         this.repositoryCtrl.delete(userID);
325     }
```

**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**

<i>botID</i>	UUID of a bot
<i>deleterID</i>	UUID of a <a href="#">User</a>

**Postcondition**

The bot is deleted from the repository.

Definition at line 336 of file PlayerCtrl.java.

```

337                                     {
338         Bot bot = this.getBot(botID);
339
340         if (!bot.getCreatorID().equals(deleterID))
341             throw new NotCreatorException();
342         if (this.gameRepositoryCtrl.existsGameByPlayerID(botID))
343             throw new BotUsedException();
344
345         this.repositoryCtrl.delete(botID);
346     }
```

### 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

<i>text</i>	String to hash
-------------	----------------

#### Postcondition

Returns the hashed password

Definition at line 354 of file PlayerCtrl.java.

```
354     {  
355         if (text.isBlank())  
356             return "";  
357         try {  
358             byte[] hash =  
359                 MessageDigest.getInstance("SHA-256").digest(text.getBytes(StandardCharsets.UTF_8));  
360             StringBuilder hexString = new StringBuilder(2 * hash.length);  
361             for (int i = 0; i < hash.length; i++) {  
362                 String hex = Integer.toHexString(0xff & hash[i]);  
363                 if (hex.length() == 1)  
364                     hexString.append('0');  
365                 hexString.append(hex);  
366             }  
367             return hexString.toString();  
368         } catch (Exception e) {  
369             return "";  
370         }  
371     }  
372 }
```

## 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.

```
31     {  
32         super (RepositoryType.PLAYER);  
33     }
```

## 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

<i>player</i>	Player to be saved.
---------------	---------------------

Definition at line 43 of file PlayerRepository.java.

```
43         {
44             String id = player.getString("id");
45             this.createOrUpdate(id, player);
46         }
```

### 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

<i>id</i>	ID of the Player to be deleted.
-----------	---------------------------------

Definition at line 54 of file PlayerRepository.java.

```
54         {
55             JSONObject player = this.get(id);
56             if (player == null)
57                 return;
58
59             if (player.getString("type").equals("BOT")) {
60                 this.remove(id);
61                 return;
62             }
63
64             player.put("is_deleted", true);
65             this.save(player);
66         }
```

### 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**

<i>id</i>	ID of the Player to be gotten.
-----------	--------------------------------

**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.

```
75         {  
76             return super.get(id);  
77         }
```

**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**

<i>name</i>	Name of the Player to be gotten.
-------------	----------------------------------

**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.

```
86         {  
87             JSONObject all = this.list();  
88               
89             JSONObject current;  
90             for (String key : all.keySet()) {  
91                 current = all.getJSONObject(key);  
92                 if (current.getString("name").equals(name))  
93                     return current;  
94             }  
95               
96             return null;  
97         }
```

### 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.

```
105         {
106             ArrayList<Pair<String, UUID> list = new ArrayList<Pair<String, UUID>();
107             JSONObject all = this.list();
108
109             JSONObject current;
110             for (String key : all.keySet()) {
111                 current = all.getJSONObject(key);
112                 if (current.getString("type").equals("USER") && !current.getBoolean("is_deleted"))
113                     list.add(new Pair<String, UUID>(current.getString("name"), UUID.fromString(key)));
114             }
115
116             return list;
117         }
```

### 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.

```
125         {
126             ArrayList<Pair<String, UUID> list = new ArrayList<Pair<String, UUID>();
127             JSONObject all = this.list();
128
129             JSONObject current;
130             for (String key : all.keySet()) {
131                 current = all.getJSONObject(key);
132                 if (current.getString("type").equals("BOT") && !current.getBoolean("is_deleted"))
133                     list.add(new Pair<String, UUID>(current.getString("name"), UUID.fromString(key)));
134             }
135
136             return list;
137         }
```

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.

```
36      {  
37          this.repository = new PlayerRepository();  
38      }
```

## 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

<i>player</i>	Player to be saved.
---------------	---------------------

Definition at line 48 of file PlayerRepositoryCtrl.java.

```
48      {  
49          this.repository.save(player);  
50      }
```

### 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

<i>id</i>	ID of the Player to be deleted.
-----------	---------------------------------

Definition at line 58 of file PlayerRepositoryCtrl.java.

```
58         {
59             this.repository.delete(id.toString());
60         }
```

### 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

<i>id</i>	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.

```
69         {
70         return this.repository.get(id.toString());
71     }
```

#### 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

<i>name</i>	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.

```
80         {
81         return this.repository.getByName(name);
82     }
```

#### 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.

```
90         {
91         return this.repository.listUsers();
92     }
```

### 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.

```
100                                     {  
101     return this.repository.listBots();  
102 }
```

## 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.

```
152         {
153             currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
154             Pair<ArrayList<String>, String> gameList = ViewCtrl.domainCtrl.listGames();
155             for(String gameName : gameList.first) gameChooser.getItems().add(gameName);
156         }
```

### 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.

```
163         {
164             ViewCtrl.changeScene("template/UserView.fxml");
165         }
```

### 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.

```
172         {
173             ViewCtrl.changeScene("template/BotsView.fxml");
174         }
```

#### 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.

```
181         {  
182             ViewCtrl.changeScene("template/ConfigView.fxml");  
183         }
```

#### 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.

```
190         {  
191             ViewCtrl.changeScene("template/GamesView.fxml");  
192         }
```

### 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.

```

199                                     {
200     String chosenGame = (String) gameChooser.getValue();
201     if (chosenGame != null) {
202         Pair<JSONObject, String> result = ViewCtrl.domainCtrl.getGame(chosenGame);
203         if (result.second != null) {
204             switch (result.second) {
205                 case "ERR_NOT_PLAYER":
206                     playResult.setText("You are not part of this game!");
207                     break;
208                 default:
209                     playResult.setText("Something went wrong, try again!");
210                     break;
211             }
212         }
213         else {
214             playResult.setText("");
215             ViewCtrl.domainCtrl.selectGame(result.first.getString("name")); // Load onto memory the
chosen game Board
216             Pair<JSONObject, JSONObject> players = ViewCtrl.domainCtrl.viewPlayers();
217             player1.setText((players.first != null ? players.first.getString("name") : "Unknown"));
218             player2.setText((players.second != null ? players.second.getString("name") :
"Unknown"));
219             configuration.setText(result.first.getString("configuration_name"));
220             Pair<JSONObject, String> userCreator =
ViewCtrl.domainCtrl.getUser(UUID.fromString(result.first.getString("creator_id")));
221             creator.setText((userCreator.first != null ? userCreator.first.getString("name") :
"Unknown"));
222             if (creator.getText().equals(currentUserName.getText()) &&
!player1.getText().equals(currentUserName.getText()) &&
!player2.getText().equals(currentUserName.getText())) {
223                 goToGame.setText("SPECTATE");
224             } else {
225                 goToGame.setText("PLAY");
226             }
227             DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
228
229             createdAt.setText(LocalDate.parse(result.first.getString("created_at")).format(dateFormat));
230             state.setText(result.first.get("state").toString());
231             if (result.first.get("state").toString().equals("FINISHED")) {
232                 goToGame.setText("CONSULT");
233                 String winner = null;
234                 winner = result.first.optString("winner_id", null);
235                 if (winner == null) {
236                     info.setText("The game has ended in a draw.");
237                 } else {
238                     if (winner.equals(players.first.getString("id"))) {
239                         info.setText(String.format("%s has won the game.", player1.getText()));
240                     } else {
241                         info.setText(String.format("%s has won the game.", player2.getText()));
242                     }
243                 }
244             } else {
245                 if (result.first.get("turn").toString().equals("PLAYER1")) {
246                     info.setText(String.format("%s has the current turn.", player1.getText()));
247                 } else {
248                     info.setText(String.format("%s has the current turn.", player2.getText()));
249                 }
250             }
251         }
252     }

```

### 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.

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

### 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.

```
268         {
269             String chosenGame = (String) gameChooser.getValue();
270             if (chosenGame != null) {
271                 Pair<JSONObject, String> result = ViewCtrl.domainCtrl.play();
272                 if (result.second != null) {
273                     switch (result.second) {
274                         case "ERR_FINISHED_GAME":
275                             // playResult.setText("The game has already finished!");
276                             playResult.setText("");
277                             ViewCtrl.changeScene("template/GameBoardView.fxml");
278                             break;
279                         default:
280                             playResult.setText("Something went wrong, try again!");
281                             break;
282                     }
283                 }
284             } else {
285                 playResult.setText("");
286                 ViewCtrl.changeScene("template/GameBoardView.fxml");
287             }
288         }
289     }
```

### 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.

```
296                                     {
297     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
298                             ButtonType.YES, ButtonType.NO);
299     confirm.showAndWait();
300     if (confirm.getResult() == ButtonType.YES) {
301         ViewCtrl.domainCtrl.logout();
302         ViewCtrl.changeScene("template/LogInView.fxml");
303     }
304 }
```

## 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 endpoint. 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.

```
24                                     {  
25     JUnitCore.main(new RankingJUnit().getClass().getName());  
26 }
```

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

<i>name</i>	Name of the table to be created
-------------	---------------------------------

Definition at line 36 of file Ranking.java.

```
36      {
37          this.entries = new ArrayList<Entry>();
38          this.name = name;
39      }
```

### 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

<i>ranking</i>	JSONObject with the attributes of the implicit <a href="#">Ranking</a>
----------------	--

Definition at line 46 of file Ranking.java.

```
46      {
47          this.name = ranking.getString("name");
48          this.entries = new ArrayList<Entry>();
49          JSONArray entries = ranking.getJSONArray("entries");
50          for(int i=0; i<entries.length(); ++i) this.entries.add(i, new Entry(entries.getJSONObject(i)));
51      }
```

### 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`.

```
58         {
59             JSONObject ranking = new JSONObject();
60             ranking.put("name", this.name);
61             JSONArray entries = new JSONArray();
62             for (Entry e: this.entries) entries.put(e.serialize());
63             ranking.put("entries", entries);
64             return ranking;
65         }
```

#### 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

<i>playerID</i>	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.

```

73         {
74             for(Entry entry : this.entries)
75                 if(entry.getPlayerID().equals(playerID))
76                     return entry;
77             return null;
78         }

```

**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**

<i>playerID</i>	<a href="#">Player</a> ID to remove from.
-----------------	---

Definition at line 85 of file Ranking.java.

```

85         {
86             ArrayList<Entry> newEntries = new ArrayList<Entry>();
87
88             for(Entry entry : this.entries)
89                 if(!entry.getPlayerID().equals(playerID))
90                     newEntries.add(entry);
91
92             this.entries = newEntries;
93         }

```

**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**

<i>value</i>	Value of the <a href="#">Entry</a> to be placed in the ranking table
<i>start</i>	Starting place of the segment of the ranking table to check
<i>end</i>	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.

```

103                                     {
104         if(start == end) {
105             if(start == this.entries.size()) return start;
106             if(this.entries.get(start).getValue() > value) return start+1;
107             else return start;
108         }
109         if(start > end) return start;
110
111         int middle = (start+end)/2;
112         int current = this.entries.get(middle).getValue();
113
114         if(current > value) return whereInsert(value, middle+1, end);
115         else if(current < value) return whereInsert(value, start, middle-1);
116         else return middle;
117     }
```

**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**

<i>entry</i>	<a href="#">Entry</a> added to <a href="#">Ranking</a>
--------------	--

Definition at line 124 of file Ranking.java.

```

124                                     {
125         this.entries.add(this.whereInsert(entry.getValue(), 0, this.entries.size()), entry);
126     }
```



### 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.

```
133 {  
134     return this.name;  
135 }
```

### 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.

```
142 {  
143     return this.entries;  
144 }
```

## 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.

```
39      {
40      }
```

## 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.

```
146      {
147          currentUser.setName(ViewCtrl.domainCtrl.viewUser().getString("name"));
148          ArrayList<String> rankingList = ViewCtrl.domainCtrl.listRankings();
149          for(String rankingName : rankingList) rankingChooser.getItems().add(rankingName);
150          playerColumn.setCellValueFactory(new PropertyValueFactory<>("first"));
151          valueColumn.setCellValueFactory(new PropertyValueFactory<>("second"));
152      }
```

### 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.

```
159      {
160          ViewCtrl.changeScene("template/UserView.fxml");
161      }
```

### 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.

```
168         {
169             ViewCtrl.changeScene("template/BotsView.fxml");
170         }
```

### 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.

```
177         {
178             ViewCtrl.changeScene("template/ConfigView.fxml");
179         }
```

### 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.

```
186         {
187             ViewCtrl.changeScene("template/GamesView.fxml");
188         }
```

### 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.

```
195         {
196             ViewCtrl.changeScene("template/PlayView.fxml");
197         }
```

### 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.

```
204         {
205             String chosenRanking = (String) rankingChooser.getValue();
206             if (rankingChooser != null) {
207                 table.getItems().clear();
208                 JSONObject ranking = ViewCtrl.domainCtrl.getRanking(chosenRanking);
209                 JSONArray entries = ranking.getJSONArray("entries");
210                 for (int i = 0; i < entries.length(); ++i) {
211                     JSONObject entry = entries.getJSONObject(i);
212                     Pair<JSONObject, String> player =
ViewCtrl.domainCtrl.getPlayer(UUID.fromString(entry.getString("player_id")));
213                     Pair<String, Integer> pairEntry = new Pair<String,
Integer>(player.first.getString("name"), entry.getInt("value"));
214                     if (player.first.getString("type") == "BOT")
pairEntry.first = pairEntry.first + " (bot)";
215                     if (player.first.getBoolean("is_deleted"))
pairEntry.first = pairEntry.first + " (deleted)";
216                     table.getItems().add(pairEntry);
217                 }
218             }
219         }
```

### 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.

```
228                                     {
229         ViewCtrl.changeScene("template/RankingView.fxml");
230     }
```

### 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.

```
237                                     {
238         ViewCtrl.changeScene("template/RecordConsultView.fxml");
239     }
```

### 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.

```
246                                     {
247         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
248         ButtonType.YES, ButtonType.NO);
249         confirm.showAndWait();
250         if (confirm.getResult() == ButtonType.YES) {
251             ViewCtrl.domainCtrl.logout();
252             ViewCtrl.changeScene("template/LogInView.fxml");
253         }
254     }
```

## 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 instance of [Ranking](#) Control is created.

Definition at line 41 of file [RankingCtrl.java](#).

```
41         {
42             this.repositoryCtrl = new RankingRepositoryCtrl();
43         }
```

## 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

<i>name</i>	Name of a ranking
-------------	-------------------

#### Returns

[Ranking](#) identified by name

Definition at line 54 of file RankingCtrl.java.

```
54                                     {
55     JSONObject ranking = this.repositoryCtrl.get(name);
56     if (ranking == null)
57         return null;
58
59     return new Ranking(ranking);
60 }
```

### 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.

```

68      {
69          return this.repositoryCtrl.listRankings();
70      }

```

**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**

<i>playerID</i>	UUID of a <a href="#">Player</a>
-----------------	----------------------------------

**Returns**

List of records of a [Player](#)

Definition at line 79 of file RankingCtrl.java.

```

79      {
80          ArrayList<String> rankings = this.repositoryCtrl.listRankings();
81          ArrayList<Pair<String, Entry>> records = new ArrayList<Pair<String, Entry>>();
82
83          for (String name : rankings) {
84              Ranking ranking = this.getRanking(name);
85              if (ranking != null) {
86                  Entry record = ranking.getRecord(playerID);
87                  if (record != null)
88                      records.add(new Pair<String, Entry>(name, record));
89              }
90          }
91
92          return records;
93      }

```

**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

<i>rankingName</i>	Name of a <a href="#">Ranking</a>
<i>playerID</i>	UUID of a <a href="#">Player</a>
<i>value</i>	Value of an entry
<i>rankingType</i>	Type of <a href="#">Ranking</a>

#### Returns

[Entry](#)

Definition at line 105 of file RankingCtrl.java.

```

105
106     {
107         Ranking ranking = this.getRanking(rankingName);
108         if (ranking == null)
109             ranking = new Ranking(rankingName);
110
111         switch (rankingType) {
112             case INCREMENTAL:
113                 Entry oldRecord = ranking.getRecord(playerID);
114                 if (oldRecord != null)
115                     value += oldRecord.getValue();
116             case UNIQUE:
117                 ranking.removePlayer(playerID);
118             case MULTIPLE:
119             default:
120                 break;
121         }
122
123         Entry entry = new Entry(playerID, value);
124         ranking.addEntry(entry);
125
126         this.repositoryCtrl.save(ranking.serialize());
127
128         return entry;
129     }

```

## 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.

```

67     {
68         Ranking r = new Ranking("time");
69         UUID playerId = UUID.randomUUID();
70
71         // Case 1
72         assertEquals("Record failed because", null, r.getRecord(playerID));
73
74         // Case 2
75         for(int i=0; i<4; ++i) r.addEntry(new Entry(UUID.randomUUID(), randomInt(0,i)));
76         assertEquals("Record failed because", null, r.getRecord(playerID));
77
78         // Case 3
79         r.addEntry(new Entry(playerID, 6));
80         assertEquals("Record failed because", new Entry(playerID, 6).getValue(),
81             r.getRecord(playerID).getValue());
82
83         // Case 4
84         r.addEntry(new Entry(playerID, 6));
85         r.addEntry(new Entry(playerID, 7));
86         r.addEntry(new Entry(playerID, 5));
87         r.addEntry(new Entry(playerID, 7));
88         assertEquals("Record failed because", new Entry(playerID, 7).getValue(),
89             r.getRecord(playerID).getValue());
90     }

```

### 6.90.2.6 addEntry()

```
void test.unitary.RankingJUnit.addEntry ( )
```

Definition at line 91 of file RankingJUnit.java.

```

91     {
92         Ranking r = new Ranking("number_of_wins");
93         UUID id = UUID.randomUUID();
94         UUID id1 = UUID.randomUUID();
95         UUID id2 = UUID.randomUUID();
96         UUID id3 = UUID.randomUUID();
97
98         ArrayList<UUID> player_ids = new ArrayList<UUID>();
99         ArrayList<Integer> player_values = new ArrayList<Integer>();
100
101         // Case 1
102         player_ids.add(0, id);
103         player_values.add(0, 5);
104         r.addEntry(new Entry(id, 5));
105         for(int i=0; i < r.getEntries().size(); ++i) {
106             assertEquals("addEntry failed because", player_ids.get(i),
107                 r.getEntries().get(i).getPlayerID());
108             assertEquals("addEntry failed because", (int) player_values.get(i),
109                 r.getEntries().get(i).getValue());
110         }
111
112         // Case 2
113         player_ids.add(1, id1);
114         player_values.add(1, 2);
115         r.addEntry(new Entry(id1, 2));
116         for(int i=0; i < r.getEntries().size(); ++i) {
117             assertEquals("addEntry failed because", player_ids.get(i),
118                 r.getEntries().get(i).getPlayerID());
119             assertEquals("addEntry failed because", (int) player_values.get(i),
120                 r.getEntries().get(i).getValue());
121         }
122
123         // Case 3
124         player_values.add(0, 25);
125         player_ids.add(0, id2);
126         r.addEntry(new Entry(id2, 25));
127         for(int i=0; i < r.getEntries().size(); ++i) {
128             assertEquals("addEntry failed because", player_ids.get(i),
129                 r.getEntries().get(i).getPlayerID());
130             assertEquals("addEntry failed because", (int) player_values.get(i),
131                 r.getEntries().get(i).getValue());
132         }
133     }

```

```

126     }
127
128     // Case 4
129     player_ids.add(1, id3);
130     player_values.add(1, 10);
131     r.addEntry(new Entry(id3, 10));
132     for(int i=0; i<r.getEntries().size(); ++i) {
133         assertEquals("addEntry failed because", player_ids.get(i),
134             r.getEntries().get(i).getPlayerID());
135         assertEquals("addEntry failed because", (int) player_values.get(i),
136             r.getEntries().get(i).getValue());
137     }
138
139     // Case 5
140     player_ids.add(0, id3);
141     player_values.add(0, 25);
142     r.addEntry(new Entry(id3, 25));
143     for(int i=0; i<r.getEntries().size(); ++i) {
144         assertEquals("addEntry failed because", player_ids.get(i),
145             r.getEntries().get(i).getPlayerID());
146         assertEquals("addEntry failed because", (int) player_values.get(i),
147             r.getEntries().get(i).getValue());
148     }
149
150     // Case 6
151     player_ids.add(5, id3);
152     player_values.add(5, 0);
153     r.addEntry(new Entry(id3, 0));
154     for(int i=0; i<r.getEntries().size(); ++i) {
155         assertEquals("addEntry failed because", player_ids.get(i),
156             r.getEntries().get(i).getPlayerID());
157         assertEquals("addEntry failed because", (int) player_values.get(i),
158             r.getEntries().get(i).getValue());
159     }
160
161     // Case 7
162     player_ids.add(0, id3);
163     player_values.add(0, 69);
164     r.addEntry(new Entry(id3, 69));
165     for(int i=0; i<r.getEntries().size(); ++i) {
166         assertEquals("addEntry failed because", player_ids.get(i),
167             r.getEntries().get(i).getPlayerID());
168         assertEquals("addEntry failed because", (int) player_values.get(i),
169             r.getEntries().get(i).getValue());
170     }
171 }

```

### 6.90.2.7 getName()

```
void test.unitary.RankingJUnit.getName ( )
```

Definition at line 167 of file RankingJUnit.java.

```

167     {
168         Ranking r = new Ranking("number_of_ties");
169         assertEquals("getName failed because", "number_of_ties", r.getName());
170     }

```

### 6.90.2.8 getEntries()

```
void test.unitary.RankingJUnit.getEntries ( )
```

Definition at line 173 of file RankingJUnit.java.

```

173     {
174         Ranking r = new Ranking("number_of_losses");
175         ArrayList<UUID> expectedIDs = new ArrayList<UUID> ();
176         UUID id = UUID.randomUUID();
177         UUID id1 = UUID.randomUUID();
178         UUID id2 = UUID.randomUUID();
179         UUID id3 = UUID.randomUUID();

```

```

180         expectedIDs.add(id);
181         expectedIDs.add(id3);
182         expectedIDs.add(id2);
183         expectedIDs.add(id1);
184
185         r.addEntry(new Entry(id, 52));
186         r.addEntry(new Entry(id1, 7));
187         r.addEntry(new Entry(id2, 15));
188         r.addEntry(new Entry(id3, 40));
189
190         ArrayList<Integer> expectedValue = new ArrayList<Integer>();
191         expectedValue.add(52);
192         expectedValue.add(40);
193         expectedValue.add(15);
194         expectedValue.add(7);
195
196         for(int i=0; i<4; ++i) {
197             assertEquals("getEntries failed because", expectedIDs.get(i),
198                 r.getEntries().get(i).getPlayerID());
199             assertEquals("getEntries failed because", (int) expectedValue.get(i),
200                 r.getEntries().get(i).getValue());
201         }

```

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

<i>ranking</i>	Ranking to be saved.
----------------	----------------------

Definition at line 40 of file RankingRepository.java.

```
40         {
41             String name = ranking.getString("name");
42             this.createOrUpdate(name, ranking);
43         }
```

### 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

<i>name</i>	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.

```
52         {
53             return super.get(name);
54         }
```

### 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.

```
62      {  
63          JSONObject all = this.list();  
64          return new ArrayList<String>(all.keySet());  
65      }
```

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.

```
33         {  
34             this.repository = new RankingRepository();  
35         }
```

## 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

<i>ranking</i>	Ranking to be saved.
----------------	----------------------

Definition at line 45 of file RankingRepositoryCtrl.java.



```
45         {
46             this.repository.save(ranking);
47         }
```

### 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

<i>name</i>	Name of the Ranking to be gotten.
-------------	-----------------------------------

#### 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.

```
56         {
57             return this.repository.get(name);
58         }
```

### 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.

```
66         {
67             return this.repository.listRankings();
68         }
```

## 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.

```
102         {
103         currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
104     }
```

### 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.

```
111         {
112         ViewCtrl.changeScene("template/UserView.fxml");
113     }
```

### 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.

```
120         {
121         ViewCtrl.changeScene("template/BotsView.fxml");
122     }
```

#### 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.

```
129      {
130          ViewCtrl.changeScene("template/ConfigView.fxml");
131      }
```

#### 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.

```
138      {
139          ViewCtrl.changeScene("template/GamesView.fxml");
140      }
```

#### 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.

```
147      {
148          ViewCtrl.changeScene("template/PlayView.fxml");
149      }
```

### 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.

```
156                                     {
157         ViewCtrl.changeScene("template/RankingConsultView.fxml");
158     }
```

### 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.

```
165                                     {
166         ViewCtrl.changeScene("template/RecordConsultView.fxml");
167     }
```

### 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.

```
174                                     {
175         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
176             ButtonType.YES, ButtonType.NO);
177         confirm.showAndWait();
178         if (confirm.getResult() == ButtonType.YES) {
179             ViewCtrl.domainCtrl.logout();
180             ViewCtrl.changeScene("template/LogInView.fxml");
181         }
182     }
```

## 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.

```
119         {
120             currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));
121             ArrayList<Pair<String, JSONObject> recordList = ViewCtrl.domainCtrl.listRecords();
122             rankingColumn.setCellValueFactory(new PropertyValueFactory<>("first"));
123             valueColumn.setCellValueFactory(new PropertyValueFactory<>("second"));
124             for (Pair<String, JSONObject> record : recordList) {
125                 Pair<String, Integer> pairRecord = new Pair<String, Integer>(record.first,
126                     record.second.getInt("value"));
127                 table.getItems().add(pairRecord);
128             }
129         }
```

#### 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.

```
135         {
136             ViewCtrl.changeScene("template/UserView.fxml");
137         }
```

### 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.

```
144         {  
145             ViewCtrl.changeScene("template/BotsView.fxml");  
146         }
```

### 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.

```
153         {  
154             ViewCtrl.changeScene("template/ConfigView.fxml");  
155         }
```

### 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.

```
162         {  
163             ViewCtrl.changeScene("template/GamesView.fxml");  
164         }
```

### 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.

```
171         {
172             ViewCtrl.changeScene("template/PlayView.fxml");
173         }
```

### 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.

```
180         {
181             ViewCtrl.changeScene("template/RankingConsultView.fxml");
182         }
```

### 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.

```
189         {
190             ViewCtrl.changeScene("template/RankingView.fxml");
191         }
```

### 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.

```
198 {
199     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
200     ButtonType.YES, ButtonType.NO);
201     confirm.showAndWait();
202     if (confirm.getResult() == ButtonType.YES) {
203         ViewCtrl.domainCtrl.logout();
204         ViewCtrl.changeScene("template/LoginView.fxml");
205     }
206 }
```

## 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

<i>repositoryType</i>	type of the accessed repository.
-----------------------	----------------------------------

Definition at line 62 of file Repository.java.

62

{

```

63         String realPath = "";
64         try {
65             realPath =
Paths.get(Repository.class.getProtectionDomain().getCodeSource().getLocation().getPath())
66                 .getParent().toString();
67         } catch (Exception e) {
68             e.printStackTrace();
69         }
70
71         this.type = repositoryType;
72         switch (repositoryType) {
73             case CONFIGURATION:
74                 this.path = Paths.get(realPath, Repository.databasesPath,
"configurations.json").toString();
75                 break;
76             case GAME:
77                 this.path = Paths.get(realPath, Repository.databasesPath, "games.json").toString();
78                 break;
79             case PLAYER:
80                 this.path = Paths.get(realPath, Repository.databasesPath, "players.json").toString();
81                 break;
82             case RANKING:
83                 this.path = Paths.get(realPath, Repository.databasesPath, "rankings.json").toString();
84                 break;
85             case FIXTURE:
86                 this.path = Paths.get(realPath, Repository.fixturesPath).toString();
87                 break;
88             default:
89                 this.path = null;
90         }
91     }

```

### 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.

```

102         {
103             JSONObject database = new JSONObject();
104
105             try {
106                 File file = new File(this.path);
107                 InputStream reader = new FileInputStream(file);
108                 JSNTokener tokenener = new JSNTokener(reader);
109                 database = new JSONObject(tokenener);
110                 reader.close();
111             } catch (Exception e) {
112                 e.printStackTrace();
113             }
114
115             return database;
116         }

```

### 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

<i>key</i>	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.

```
126     {
127         JSONObject database = this.list();
128         return database.optJSONObject(key);
129     }
```

### 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**

<i>key</i>	Key of the entry in the accessed database.
<i>value</i>	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.

```
140                                     {
141     JSONObject database = this.list();
142     database.put(key, value);
143
144     try {
145         File file = new File(this.path);
146         FileWriter writer = new FileWriter(file);
147         database.write(writer, 2, 0);
148         writer.close();
149     } catch (Exception e) {
150         e.printStackTrace();
151     }
152
153     return database;
154 }
```

**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**

<i>key</i>	Key of the entry in the accessed database.
------------	--

**Returns**

JSONObject that represents the accessed database.

Definition at line 164 of file Repository.java.

```
164                                     {
165     JSONObject database = this.list();
166     database.remove(key);
```

```
167
168     try {
169         File file = new File(this.path);
170         FileWriter writer = new FileWriter(file);
171         database.write(writer, 2, 0);
172         writer.close();
173     } catch (Exception e) {
174         e.printStackTrace();
175     }
176
177     return database;
178 }
```

## 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.

```
30     {
31 }
```

### 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.

```
83     {
84         Pair<JSONObject, String> result = ViewCtrl.domainCtrl.createUser(nusername.getText(),
npassword.getText(),
85             rpassword.getText());
86         if (result.second != null) {
87             switch (result.second) {
88                 case "ERR_INVALID_NAME":
89                     signUpResult.setText("Username can't be empty!");
90                     break;
91                 case "ERR_INVALID_PASSWORD":
92                     signUpResult.setText("Password can't be empty!");
93                     break;
94                 case "ERR_EXISTING_PLAYER":
95                     signUpResult.setText("The username is already taken!");
96                     break;
97                 case "ERR_BAD_CONFIRMATION":
98                     signUpResult.setText("Confirmation password doesn't match!");
99                     break;
100                 default:
101                     signUpResult.setText("Something went wrong, try again!");
102                     break;
103             }
104         } else {
105             ViewCtrl.changeScene("template/LoginView.fxml");
106         }
107     }
```

### 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.

```
114         {  
115             ViewCtrl.changeScene("template/LogInView.fxml");  
116         }
```

## 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 selectedPosition, 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 [TreeNode](#)s 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 current 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`, `canEatVertically`, `canEatDiagonally` and `selectedPosition` attributes are setted.

## Parameters

<i>rootType</i>	PieceType of the root <a href="#">TreeNode</a> of a tree.
<i>pieceType</i>	PieceType used to know whose turn it is to make a move.
<i>currentBoard</i>	Current state of the board.
<i>canEatHorizontally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten horizontally.
<i>canEatVertically</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten vertically.
<i>canEatDiagonally</i>	Whether the pieces of the current <a href="#">Game</a> can be eaten diagonally.
<i>selectedPosition</i>	Position in the board that resulted in the current board state (it can be null).

Definition at line 145 of file HardDifficulty.java.

```

146
147     {
148         this.rootType = rootType;
149         this.pieceType = pieceType;
150         this.currentBoard = currentBoard;
151         this.canEatHorizontally = canEatHorizontally;
152         this.canEatVertically = canEatVertically;
153         this.canEatDiagonally = canEatDiagonally;
154         this.children = new ArrayList<TreeNode> ();
155         this.nVisits = 0;
156         this.totValue = 0;
157         this.selectedPosition = selectedPosition;
158     }

```

## 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.

```

167     {
168         return this.children.isEmpty();
169     }

```

### 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`.

```
178                                     {
179         return this.selectedPosition;
180     }
```

### 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`.

```
191         {
192         return (this.totValue / (this.nVisits + HardDifficulty.epsilon));
193     }
```



### 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.

```
201                                     {
202         return this.children;
203     }
```

### 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.

```
215     {
216         List<TreeNode> visited = new LinkedList<TreeNode>();
217         TreeNode current = this;
218         visited.add(this);
219
220         while (!current.isLeaf()) {
221             current = current.select();
222             visited.add(current);
223         }
224
225         current.expand();
226
227         if(current.isLeaf()) {
228             double value = current.rollOut();
229             for(TreeNode node : visited) node.updateStats(value);
230         }
231
232         else {
233             TreeNode bestChild = current.select();
234             visited.add(bestChild);
235             double value = bestChild.rollOut();
236             for (TreeNode node : visited) node.updateStats(value);
237         }
238     }
```

### 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.

```
251     {
252         TreeNode selected = null;
253         double bestValue = Double.NEGATIVE_INFINITY;
254
255         for (TreeNode child : this.children) {
256             double uctValue = child.totValue / (child.nVisits + HardDifficulty.epsilon) +
257                 Math.sqrt(Math.log(this.nVisits+1) / (child.nVisits +
258                     HardDifficulty.epsilon)) +
259                 HardDifficulty.random.nextDouble() * HardDifficulty.epsilon;
260
261             if (uctValue > bestValue) {
262                 bestValue = uctValue;
263                 selected = child;
264             }
265
266         return selected;
267     }
```

### 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, `TreeNode`s 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`.

```

278         {
279             ArrayList<Pair<Integer,Integer> validPositions = this.currentBoard.validPositions (
280                 this.pieceType, this.canEatHorizontally, this.canEatVertically, this.canEatDiagonally);
281
282             for (int i = 0; i < validPositions.size(); ++i) {
283                 Board board = new Board(this.currentBoard.getBoard());
284                 board.placePiece(validPositions.get(i), this.pieceType, this.canEatHorizontally,
285                     this.canEatVertically, this.canEatDiagonally);
286
287                 this.children.add(i, new TreeNode(this.rootType,
288                     HardDifficulty.inversePieceType(this.pieceType), board, this.canEatHorizontally,
289                     this.canEatVertically, this.canEatDiagonally, validPositions.get(i)));
290             }

```

**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`.

```

301         {
302             int piecesPlayer1 = this.currentBoard.getPiecesPlayer1();
303             int piecesPlayer2 = this.currentBoard.getPiecesPlayer2();
304             if(this.rootType == PieceType.PLAYER1) {
305                 if(piecesPlayer1 > piecesPlayer2) return 1;
306                 else return 0;
307             }
308             else {
309                 if(piecesPlayer2 > piecesPlayer1) return 1;
310                 else return 0;
311             }
312         }

```

### 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

<i>value</i>	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`.

```
322                                     {
323         ++this.nVisits;
324         this.totValue += value;
325     }
```

## 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 current 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.

```
22                                     {
23         new UserDriver().start();
24     }
```

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.

```
33     {
34         this.name = name;
35         this.password = password;
36         this.id = id;
37         this.isDeleted = false;
38     }
```



### 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.

```
44         {
45         this.name = user.getString("name");
46         this.id = UUID.fromString(user.getString("id"));
47         this.password = user.getString("password");
48         this.isDeleted = user.getBoolean("is_deleted");
49     }
```

## 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.

```
55     {
56     JSONObject user = new JSONObject();
57     user.put("name", this.name);
58     user.put("id", this.id.toString());
59     user.put("password", this.password);
60     user.put("type", "USER");
61     user.put("is_deleted", this.isDeleted);
62
63     return user;
64 }
```

### 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.

```
73 {  
74     return this.password;  
75 }
```

### 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.

```
84 {  
85     if(password.isBlank()) {  
86         throw new InvalidPasswordException();  
87     }  
88     this.password = password;  
89 }
```

## 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.

```
30         {  
31     }
```

## 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.

```
123         {  
124     currentUserName.setText(ViewCtrl.domainCtrl.viewUser().getString("name"));  
125     }
```

### 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.

```
132         {  
133     ViewCtrl.changeScene("template/BotsView.fxml");  
134     }
```

### 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.

```
141      {
142          ViewCtrl.changeScene("template/ConfigView.fxml");
143      }
```

### 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.

```
150      {
151          ViewCtrl.changeScene("template/GamesView.fxml");
152      }
```

### 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.

```
159      {
160          ViewCtrl.changeScene("template/RankingView.fxml");
161      }
```

### 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.

```
168         {
169             ViewCtrl.changeScene("template/PlayView.fxml");
170         }
```

### 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.

```
177         {
178             ViewCtrl.changeScene("template/UserModifyView.fxml");
179         }
```

### 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.

```
186         {
187             ViewCtrl.changeScene("template/UserView.fxml");
188         }
```

### 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.

```

195                                     {
196     Alert confirm = new Alert(AlertType.CONFIRMATION, "Your account will be deleted. Are you sure?",
        ButtonType.YES, ButtonType.NO);
197     confirm.showAndWait();
198
199     if (confirm.getResult() == ButtonType.YES) {
200         String result = ViewCtrl.domainCtrl.deleteUser(password.getText());
201         if (result != null) {
202             switch (result) {
203                 case "ERR_INCORRECT_CREDENTIALS":
204                     deleteUserResult.setText("Wrong password!");
205                     break;
206                 case "ERR_INEXISTING_PLAYER":
207                     deleteUserResult.setText("The player doesn't exist!");
208                     break;
209                 default:
210                     deleteUserResult.setText("Something went wrong, try again!");
211                     break;
212             }
213         }
214         else {
215             ViewCtrl.changeScene("template/LoginView.fxml");
216         }
217     }
218 }

```

### 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.

```

225                                     {
226     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
        ButtonType.YES, ButtonType.NO);
227     confirm.showAndWait();
228
229     if (confirm.getResult() == ButtonType.YES) {
230         ViewCtrl.domainCtrl.logout();
231         ViewCtrl.changeScene("template/LoginView.fxml");
232     }
233 }

```



## 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.

```
16      {
17          this.currentUser = null;
18      }
```

## 6.103.3 Member Function Documentation

### 6.103.3.1 start()

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

Definition at line 20 of file UserDriver.java.

```
20      {
21          while(true) {
22              this.mainMenu();
23          }
24      }
```

### 6.103.3.2 mainMenu()

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

Definition at line 26 of file UserDriver.java.

```
26      {
27          String title = (this.currentUser != null ? String.format("Current: %s\n",
this.currentUser.getName()) : null);
28          switch (Driver.menu(title, "User Driver",
29              new Pair<String, String>("1", "Create User"),
30              new Pair<String, String>("2", "Get name"),
31              new Pair<String, String>("3", "Set name"),
32              new Pair<String, String>("4", "Get password"),
33              new Pair<String, String>("5", "Set password"),
34              new Pair<String, String>("6", "Get state"),
35              new Pair<String, String>("7", "Set state"),
36              new Pair<String, String>("8", "Get type"),
37              new Pair<String, String>("9", "Get ID"),
38              new Pair<String, String>("10", "Serialize User to JSON"),
39              new Pair<String, String>("11", "Deserialize User from JSON"))) {
40              case "1":
41                  this.createUser();
42                  break;
43              case "2":
44                  this.getName();
45                  break;
46              case "3":
47                  this.setName();
48                  break;
49              case "4":
50                  this.getPassword();
51                  break;
52              case "5":
53                  this.setPassword();
54                  break;
55              case "6":
56                  this.getIsDeleted();
57                  break;
58              case "7":
59                  this.setIsDeleted();
60                  break;
```

```

61         case "8":
62             this.getType();
63             break;
64         case "9":
65             this.getID();
66             break;
67         case "10":
68             this.serialize();
69             break;
70         case "11":
71             this.deserialize();
72             break;
73     }
74     Driver.pause();
75 }

```

### 6.103.3.3 createUser()

void test.driver.UserDriver.createUser ( ) [private]

Definition at line 77 of file UserDriver.java.

```

77     {
78         System.out.println("Take into account that UUIDs will be randomly generated because typing them
in will be a hassle.\n");
79         String name = Driver.input("Name?");
80         String password = Driver.input("Password?");
81         try {
82             User user = new User("Default name", "Default password", UUID.randomUUID());
83             user.setName(name);
84             user.setPassword(password);
85             this.currentUser = user;
86             System.out.println(String.format("%s created successfully!", this.currentUser.getName()));
87         } catch (Exception e) {
88             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
e.getMessage()));
89         }
90     }

```

### 6.103.3.4 serialize()

void test.driver.UserDriver.serialize ( ) [private]

Definition at line 92 of file UserDriver.java.

```

92     {
93         if(this.currentUser == null) {
94             System.out.println("No current User");
95             return;
96         }
97         System.out.println(String.format("%s's serialized to JSON is: %s", this.currentUser.getName(),
this.currentUser.serialize().toString(2)));
98     }
99 }

```

### 6.103.3.5 deserialize()

void test.driver.UserDriver.deserialize ( ) [private]

Definition at line 101 of file UserDriver.java.

```

101     {
102         if(this.currentUser == null) {
103             System.out.println("No current User");
104             return;
105         }
106         System.out.println(this.currentUser.serialize().toString(2));
107         this.currentUser = new User(this.currentUser.serialize());
108         System.out.println(String.format("\n%s's deserialized from the above JSON successfully!\n",
this.currentUser.getName()));
109         System.out.println(String.format("name:\t\t%s", this.currentUser.getName()));
110         System.out.println(String.format("id:\t\t%s", this.currentUser.getID()));
111         System.out.println(String.format("password:\t\t%s", this.currentUser.getPassword()));
112         System.out.println(String.format("is_deleted:\t\t%s", this.currentUser.getIsDeleted()));
113     }
114 }

```

### 6.103.3.6 getName()

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

Definition at line 116 of file UserDriver.java.

```
116         {
117             if(this.currentUser == null) {
118                 System.out.println("No current user!");
119                 return;
120             }
121             System.out.println(String.format("%s's name is: %s", this.currentUser.getName(),
122                                     this.currentUser.getName()));
122         }
```

### 6.103.3.7 getPassword()

```
void test.driver.UserDriver.getPassword ( ) [private]
```

Definition at line 124 of file UserDriver.java.

```
124         {
125             if(this.currentUser == null) {
126                 System.out.println("No current user!");
127                 return;
128             }
129             System.out.println(String.format("%s's password is: %s", this.currentUser.getName(),
130                                     this.currentUser.getPassword()));
130         }
```

### 6.103.3.8 getIsDeleted()

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

Definition at line 132 of file UserDriver.java.

```
132         {
133             if(this.currentUser == null) {
134                 System.out.println("No current user!");
135                 return;
136             }
137             System.out.print(String.format("%s's state is: ", this.currentUser.getName()));
138             if(this.currentUser.getIsDeleted())
139                 System.out.println("DELETED");
140             else
141                 System.out.println("ACTIVE");
142         }
```

### 6.103.3.9 getType()

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

Definition at line 144 of file UserDriver.java.

```
144         {
145             System.out.println("User's type attribute was removed because of professor's feedback. However,
146                                     this option is still here to have backwards compatibility with old delivered documents.");
146         }
```

### 6.103.3.10 getID()

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

Definition at line 148 of file UserDriver.java.

```
148     {
149         if(this.currentUser == null) {
150             System.out.println("No current user!");
151             return;
152         }
153         System.out.println(String.format("%s's ID is: %s", this.currentUser.getName(),
154             this.currentUser.getID()));
155     }
```

### 6.103.3.11 setName()

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

Definition at line 156 of file UserDriver.java.

```
156     {
157         if(this.currentUser == null) {
158             System.out.println("No current user!");
159             return;
160         }
161         try {
162             this.currentUser.setName(Driver.input("New name?"));
163             System.out.println(String.format("%s name changed successfully!",
164                 this.currentUser.getName()));
165         } catch (Exception e) {
166             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
167                 e.getMessage()));
168         }
169     }
```

### 6.103.3.12 setPassword()

```
void test.driver.UserDriver.setPassword ( ) [private]
```

Definition at line 169 of file UserDriver.java.

```
169     {
170         if(this.currentUser == null) {
171             System.out.println("No current user!");
172             return;
173         }
174         try {
175             this.currentUser.setPassword(Driver.input("New password?"));
176             System.out.println(String.format("%s password changed successfully!",
177                 this.currentUser.getName()));
178         } catch (Exception e) {
179             System.out.println(String.format("Oh no! The execution threw an exception (EXPECTED): %s",
180                 e.getMessage()));
181         }
182     }
```



### 6.103.3.13 setIsDeleted()

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

Definition at line 182 of file UserDriver.java.

```

182         {
183             if(this.currentUser == null) {
184                 System.out.println("No current user!");
185                 return;
186             }
187             if(Driver.inputBool("Do you want to delete the current user?")) {
188                 this.currentUser.setIsDeleted(true);
189                 System.out.println(String.format("%s's state has changed to DELETED!",
190                     this.currentUser.getName()));
191             }
192             else {
193                 System.out.println(String.format("%s's state has not changed!",
194                     this.currentUser.getName()));
195             }
196         }

```

## 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.

```
33         {  
34     }
```

## 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.

```
136         {  
137             JSONObject user = ViewCtrl.domainCtrl.viewUser();  
138             currentUserName.setText(user.getString("name"));  
139             nusername.setText(user.getString("name"));  
140         }
```

### 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.

```
147         {  
148             ViewCtrl.changeScene("template/BotsView.fxml");  
149         }
```

### 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.

```
156      {  
157          ViewCtrl.changeScene("template/ConfigView.fxml");  
158      }
```

### 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.

```
165      {  
166          ViewCtrl.changeScene("template/GamesView.fxml");  
167      }
```

### 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.

```
174      {  
175          ViewCtrl.changeScene("template/RankingView.fxml");  
176      }
```

### 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.

```
183         {
184             ViewCtrl.changeScene("template/PlayView.fxml");
185         }
```

### 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.

```
192         {
193             ViewCtrl.changeScene("template/UserView.fxml");
194         }
```

### 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.

```
201         {
202             ViewCtrl.changeScene("template/UserDeleteView.fxml");
203         }
```

### 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.

```

210                                     {
211     Alert confirm = new Alert(AlertType.CONFIRMATION, "Your account will be modified. Are you
    sure?", ButtonType.YES, ButtonType.NO);
212     confirm.showAndWait();
213
214     if (confirm.getResult() == ButtonType.YES) {
215         String newPassword = (!npassword.getText().isBlank() ? npassword.getText() : null);
216
217         Pair<JSONObject, String> result = ViewCtrl.domainCtrl.modifyUser(nusername.getText(),
    newPassword, rpassword.getText());
218         if (result.second != null) {
219             switch (result.second) {
220                 case "ERR_INVALID_NAME":
221                     modifyUserResult.setText("Username can't be empty!");
222                     break;
223                 case "ERR_INVALID_PASSWORD":
224                     modifyUserResult.setText("Password can't be empty!");
225                     break;
226                 case "ERR_BAD_CONFIRMATION":
227                     modifyUserResult.setText("Confirmation password doesn't match!");
228                     break;
229                 case "ERR_INEXISTING_PLAYER":
230                     modifyUserResult.setText("The player doesn't exist!");
231                     break;
232                 case "ERR_EXISTING_PLAYER":
233                     modifyUserResult.setText("The username is already taken!");
234                     break;
235                 default:
236                     modifyUserResult.setText("Something went wrong, try again!");
237                     break;
238             }
239         }
240         else {
241             initialize();
242             npassword.clear();
243             rpassword.clear();
244             modifyUserResult.setText("Success!");
245         }
246     }
247 }
```

### 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.

```
254     {
255         Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
                                   ButtonType.YES, ButtonType.NO);
256         confirm.showAndWait();
257
258         if (confirm.getResult() == ButtonType.YES) {
259             ViewCtrl.domainCtrl.logout();
260             ViewCtrl.changeScene("template/LogInView.fxml");
261         }
262     }
```

## 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.

```
29         {  
30     }
```

### 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.

```
102         {  
103             currentUserName.setText (ViewCtrl.domainCtrl.viewUser().getString("name"));  
104     }
```

### 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.

```
111         {
112     ViewCtrl.changeScene("template/BotsView.fxml");
113     }
```

### 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.

```
120         {
121     ViewCtrl.changeScene("template/ConfigView.fxml");
122     }
```

### 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.

```
129         {
130     ViewCtrl.changeScene("template/GamesView.fxml");
131     }
```

### 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`.

```
138         {
139             ViewCtrl.changeScene("template/RankingView.fxml");
140         }
```

### 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`.

```
147         {
148             ViewCtrl.changeScene("template/PlayView.fxml");
149         }
```

### 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`.

```
156         {
157             ViewCtrl.changeScene("template/UserModifyView.fxml");
158         }
```

### 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.

```
165                                     {
166     ViewController.changeScene("template/UserDeleteView.fxml");
167 }
```

### 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.

```
174                                     {
175     Alert confirm = new Alert(AlertType.CONFIRMATION, "You are going to log out. Are you sure?",
176                             ButtonType.YES, ButtonType.NO);
177     confirm.showAndWait();
178     if (confirm.getResult() == ButtonType.YES) {
179         ViewController.domainCtrl.logout();
180         ViewController.changeScene("template/LoginView.fxml");
181     }
182 }
```

## 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 the 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.

```
41         {  
42     }
```

### 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.

```
52         {  
53     ViewCtrl.stage = primaryStage;  
54     primaryStage.setResizable(false);  
55     Parent root = FXMLLoader.load(getClass().getResource("template/LogInView.fxml"));  
56     primaryStage.setTitle("OTHELLO");  
57     primaryStage.setScene(new Scene(root, 1464, 824));  
58     primaryStage.show();  
59 }
```

### 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.

```
66                                     {
67     Parent pane = FXMLLoader.load(ViewCtrl.class.getResource(fxml));
68     stage.setScene().setRoot(pane);
69 }
```

### 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.

```
76                                     {
77     Parent pane = FXMLLoader.load(ViewCtrl.class.getResource(fxml));
78     Stage windowStage = new Stage();
79     windowStage.setScene(new Scene(pane, 1464, 824));
80     windowStage.setTitle("OTHELLO");
81     windowStage.setResizable(false);
82     windowStage.initModality(Modality.APPLICATION_MODAL);
83     stage.hide();
84     windowStage.showAndWait();
85     stage.show();
86 }
```

### 6.106.3.4 main()

```
static void view.ViewCtrl.main (
    String[] args ) [static]
```

Main method.

Definition at line 91 of file ViewCtrl.java.

```
91     {
92         ViewCtrl.domainCtrl = new DomainCtrl();
93         ViewCtrl.launch(args);
94     }
```

## 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 class 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 endpoint class specification.

### Classes

- class [cmd.driver.configuration](#)  
*Configuration driver endpoint. By Alex Rodriguez.*

### Packages

- package [cmd.driver](#)

### 7.16.1 Detailed Description

ConfigurationDriver endpoint 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 entripoint class specification.

### Classes

- class [cmd.unitary.entry](#)  
*JUnit Entry tests entripoint. By Alex Rodriguez.*

## Packages

- package [cmd.unitary](#)

### 7.31.1 Detailed Description

JUnit Entry tests entripoint 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 endpoint. By Alex Rodriguez.*

## Packages

- package [cmd.driver](#)

### 7.36.1 Detailed Description

GameDriver endpoint 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.GameCreateView](#)

## 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.GameView](#)

## 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 entryptoint class specification.

### Classes

- class [cmd.driver.user](#)  
*User driver entryptoint. By Alex Rodriguez.*

### Packages

- package [cmd.driver](#)

#### 7.74.1 Detailed Description

UserDriver entryptoint 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

- addEntry
  - domain.Ranking, [524](#)
  - test.unitary.RankingJUnit, [542](#)
- addPositions
  - test.driver.BoardDriver, [56](#)
- assistedMode
  - view.GameBoardView, [336](#)
- BadConfirmationException
  - domain.Exceptions.BadConfirmationException, [23](#)
- Board
  - domain.Board, [26](#), [27](#)
- board
  - domain.Board, [43](#)
  - view.ConsultInitialBoardView, [211](#)
  - view.GameBoardView, [337](#)
  - view.InitialBoardView, [423](#)
  - view.ModifyInitialBoardView, [468](#)
- Board.java, [627](#)
- board.java, [627](#)
- BoardCtrl
  - domain.BoardCtrl, [45](#)
- boardCtrl
  - domain.DomainCtrl, [256](#)
- BoardCtrl.java, [628](#)
- BoardDriver
  - test.driver.BoardDriver, [50](#)
- BoardDriver.java, [628](#)
- boardFileToJson
  - repository.FixtureRepository, [290](#)
- Bot
  - domain.Bot, [62](#)
- Bot.java, [629](#)
- bot.java, [629](#)
- botChooser
  - view.BotsConsultView, [81](#)
  - view.BotsModifyView, [104](#)
- botChooser1
  - view.GamesCreateView, [374](#), [379](#)
- botChooser2
  - view.GamesCreateView, [374](#), [379](#)
- BotDriver
  - test.driver.BotDriver, [67](#)
- BotDriver.java, [629](#)
- botMap
  - view.BotsConsultView, [82](#)
  - view.BotsModifyView, [107](#)
  - view.GamesCreateView, [381](#)
- bots
  - view.BotsConsultView, [79](#)
  - view.BotsCreateView, [90](#)
  - view.BotsModifyView, [102](#)
  - view.BotsView, [113](#)
  - view.ConfigConsultView, [120](#), [123](#)
  - view.ConfigCreateView, [131](#), [135](#)
  - view.ConfigModifyView, [142](#), [148](#)
  - view.ConfigView, [187](#), [190](#)
  - view.GamesCreateView, [372](#), [376](#)
  - view.GamesView, [384](#), [387](#)
  - view.PlayView, [510](#), [514](#)
  - view.RankingConsultView, [528](#), [532](#)
  - view.RankingView, [553](#), [556](#)
  - view.RecordConsultView, [560](#), [563](#)
  - view.UserDeleteView, [593](#), [597](#)
  - view.UserModifyView, [607](#), [611](#)
  - view.UserView, [616](#), [620](#)
- BotsConsultView
  - view.BotsConsultView, [74](#)
- BotsConsultView.java, [630](#)
- BotsCreateView
  - view.BotsCreateView, [85](#)
- BotsCreateView.java, [630](#)
- BotsModifyView
  - view.BotsModifyView, [95](#)
- BotsModifyView.java, [631](#)
- BotsView
  - view.BotsView, [108](#)
- BotsView.java, [631](#)
- BotUsedException
  - domain.Exceptions.BotUsedException, [116](#)
- canEatDiagonally
  - domain.Configuration, [161](#)
  - domain.Difficulty, [218](#)
  - domain.HardDifficulty.TreeNode, [586](#)
  - view.ConfigCreateView, [138](#)
  - view.ConfigModifyView, [150](#)
- canEatHorizontally
  - domain.Configuration, [160](#)
  - domain.Difficulty, [218](#)
  - domain.HardDifficulty.TreeNode, [585](#)
  - view.ConfigCreateView, [137](#)
  - view.ConfigModifyView, [150](#)
- canEatVertically
  - domain.Configuration, [161](#)
  - domain.Difficulty, [218](#)
  - domain.HardDifficulty.TreeNode, [586](#)
  - view.ConfigCreateView, [138](#)
  - view.ConfigModifyView, [150](#)
- canPlaceDiagonal

- domain.Board, 40
- canPlaceHorizontal
  - domain.Board, 38
- canPlaceVertical
  - domain.Board, 39
- ced
  - view.ConfigConsultView, 127
- ceh
  - view.ConfigConsultView, 126
- cev
  - view.ConfigConsultView, 126
- changePieces
  - domain.Board, 42
- changeScene
  - view.ViewCtrl, 623
- checkInitialBoard
  - view.ConfigConsultView, 122, 127
- checkInitialBoardButton
  - view.ConfigConsultView, 127
- checkPlaceRights
  - domain.Game, 304
  - domain.GameCtrl, 346
  - test.driver.GameDriver, 357
- children
  - domain.HardDifficulty.TreeNode, 584
- clear
  - test.driver.Driver, 259
- cmd, 17
- cmd.driver, 17
- cmd.driver.board, 24
  - main, 24
- cmd.driver.bot, 65
  - main, 65
- cmd.driver.configuration, 161
  - main, 162
- cmd.driver.easyDifficulty, 262
  - main, 262
- cmd.driver.game, 307
  - main, 308
- cmd.driver.hardDifficulty, 392
  - main, 392
- cmd.driver.mediumDifficulty, 432
  - main, 433
- cmd.driver.pair, 473
  - main, 473
- cmd.driver.user, 586
  - main, 587
- cmd.othello, 472
  - main, 472
- cmd.unitary, 18
- cmd.unitary.entry, 276
  - main, 276
- cmd.unitary.ranking, 518
  - main, 519
- comparePair
  - test.driver.PairDriver, 481
- config
  - view.BotsConsultView, 75, 79
  - view.BotsCreateView, 85, 90
  - view.BotsModifyView, 97, 102
  - view.BotsView, 109, 113
  - view.ConfigConsultView, 123
  - view.ConfigCreateView, 135
  - view.ConfigModifyView, 148
  - view.ConfigView, 190
  - view.GamesCreateView, 372, 377
  - view.GamesView, 384, 387
  - view.PlayView, 510, 514
  - view.RankingConsultView, 529, 532
  - view.RankingView, 553, 556
  - view.RecordConsultView, 561, 563
  - view.UserDeleteView, 593, 597
  - view.UserModifyView, 607, 611
  - view.UserView, 617, 620
- configChooser
  - view.ConfigConsultView, 126
  - view.ConfigModifyView, 152
  - view.GamesCreateView, 378
- ConfigConsultView
  - view.ConfigConsultView, 118
- ConfigConsultView.java, 632
- ConfigCreateView
  - view.ConfigCreateView, 130
- ConfigCreateView.java, 632
- ConfigModifyView
  - view.ConfigModifyView, 141
- ConfigModifyView.java, 633
- CONFIGURATION
  - repository.Repository.RepositoryType, 572
- Configuration
  - domain.Configuration, 154
- configuration
  - view.PlayView, 516
- Configuration.java, 633
- configuration.java, 634
- ConfigurationCtrl
  - domain.ConfigurationCtrl, 163
- configurationCtrl
  - domain.DomainCtrl, 255
- ConfigurationCtrl.java, 634
- ConfigurationDriver
  - test.driver.ConfigurationDriver, 170
- ConfigurationDriver.java, 635
- configurationName
  - domain.Game, 306
- ConfigurationRepository
  - repository.ConfigurationRepository, 176
- ConfigurationRepository.java, 635
- ConfigurationRepositoryCtrl
  - repository.ConfigurationRepositoryCtrl, 180
- configurationRepositoryCtrl
  - domain.GameCtrl, 348
- ConfigurationRepositoryCtrl.java, 636
- ConfigurationUsedException
  - domain.Exceptions.ConfigurationUsedException, 184



- ConfigView
  - view.ConfigView, 186
- ConfigView.java, 636
- consultBot
  - view.BotsConsultView, 77, 80
  - view.BotsCreateView, 87, 91
  - view.BotsModifyView, 99, 104
  - view.BotsView, 111, 114
- consultBotButton
  - view.BotsConsultView, 81
  - view.BotsCreateView, 92
  - view.BotsModifyView, 104
  - view.BotsView, 115
- consultBotResult
  - view.BotsConsultView, 82
- consultConfig
  - view.ConfigConsultView, 119, 125
  - view.ConfigCreateView, 133, 136
  - view.ConfigModifyView, 145, 149
  - view.ConfigView, 189, 191
- consultConfigButton
  - view.ConfigConsultView, 125
  - view.ConfigCreateView, 137
  - view.ConfigModifyView, 150
  - view.ConfigView, 192
- consultConfigConfirm
  - view.ConfigConsultView, 125
- consultConfigConfirmButton
  - view.ConfigConsultView, 126
- consultConfigResult
  - view.BotsConsultView, 81
  - view.ConfigConsultView, 127
- ConsultInitialBoardView
  - view.ConsultInitialBoardView, 196
- ConsultInitialBoardView.java, 637
- consultRanking
  - view.RankingView, 557
  - view.RecordConsultView, 564
- consultRankingButton
  - view.RankingView, 557
  - view.RecordConsultView, 564
- consultRankings
  - view.RankingConsultView, 530, 533
  - view.RankingView, 554
  - view.RecordConsultView, 562
- consultRankingsButton
  - view.RankingConsultView, 533
- consultRankingsConfirm
  - view.RankingConsultView, 533
- consultRankingsConfirmButton
  - view.RankingConsultView, 533
- consultRecord
  - view.RankingView, 557
  - view.RecordConsultView, 565
- consultRecordButton
  - view.RankingView, 557
  - view.RecordConsultView, 565
- consultRecords
  - view.RankingConsultView, 531, 533
  - view.RankingView, 555
  - view.RecordConsultView, 562
- consultRecordsButton
  - view.RankingConsultView, 534
- correctBoolean
  - test.driver.BoardDriver, 60
- correctNumber
  - test.driver.BoardDriver, 58
- correctPieceType
  - test.driver.BoardDriver, 59
- create
  - domain.ConfigurationCtrl, 164
  - domain.GameCtrl, 340
  - test.driver.ConfigurationDriver, 171
  - test.driver.EasyDifficultyDriver, 270
  - test.driver.GameDriver, 351
  - test.driver.HardDifficultyDriver, 394
  - test.driver.MediumDifficultyDriver, 441
- createBoard
  - test.driver.BoardDriver, 51
- createBot
  - domain.DomainCtrl, 228
  - domain.PlayerCtrl, 489
  - test.driver.BotDriver, 68
  - view.BotsConsultView, 77, 80
  - view.BotsCreateView, 87, 91
  - view.BotsModifyView, 98, 103
  - view.BotsView, 111, 114
- createBotButton
  - view.BotsConsultView, 80
  - view.BotsCreateView, 91
  - view.BotsModifyView, 103
  - view.BotsView, 114
- createBotConfirm
  - view.BotsCreateView, 88, 93
- createBotConfirmButton
  - view.BotsCreateView, 93
- createBotResult
  - view.BotsCreateView, 92
- createConfig
  - view.ConfigConsultView, 119, 124
  - view.ConfigCreateView, 132, 136
  - view.ConfigModifyView, 144, 149
  - view.ConfigView, 188, 191
- createConfigButton
  - view.ConfigConsultView, 124
  - view.ConfigCreateView, 136
  - view.ConfigModifyView, 149
  - view.ConfigView, 191
- createConfigConfirm
  - view.ConfigCreateView, 133, 138
- createConfigConfirmButton
  - view.ConfigCreateView, 138
- createConfigResult
  - view.ConfigCreateView, 138
- createConfiguration
  - domain.DomainCtrl, 235

- createdAt
  - domain.Game, 307
  - view.PlayView, 516
- createEntries
  - domain.DomainCtrl, 254
- createEntry
  - domain.RankingCtrl, 538
- createGame
  - domain.DomainCtrl, 240
  - view.GamesCreateView, 373, 377
  - view.GamesView, 385, 388
- createGameButton
  - view.GamesCreateView, 378
  - view.GamesView, 388
- createGameConfirm
  - view.GamesCreateView, 375, 380
- createGameConfirmButton
  - view.GamesCreateView, 380
- createGameResult
  - view.GamesCreateView, 380
- createInitialBoard
  - domain.DomainCtrl, 236
  - view.ConfigCreateView, 133, 137
- createInitialBoardButton
  - view.ConfigCreateView, 137
- createIntPair
  - test.driver.PairDriver, 480
- createOrUpdate
  - repository.Repository, 569
- createStrIntPair
  - test.driver.PairDriver, 480
- createStrPair
  - test.driver.PairDriver, 479
- createUser
  - domain.DomainCtrl, 227
  - domain.PlayerCtrl, 488
  - test.driver.UserDriver, 602
- creator
  - view.BotsConsultView, 82
  - view.ConfigConsultView, 127
  - view.PlayView, 516
- creatorID
  - domain.Bot, 65
  - domain.Configuration, 160
  - domain.Game, 307
- currentBoard
  - domain.DomainCtrl, 257
  - domain.HardDifficulty.TreeNode, 585
  - test.driver.BoardDriver, 60
  - test.driver.EasyDifficultyDriver, 275
  - test.driver.HardDifficultyDriver, 399
  - test.driver.MediumDifficultyDriver, 446
- currentBot
  - test.driver.BotDriver, 72
- currentConfiguration
  - domain.DomainCtrl, 257
  - test.driver.ConfigurationDriver, 175
- currentEasyDifficulty
  - test.driver.EasyDifficultyDriver, 275
- currentGame
  - domain.DomainCtrl, 257
  - test.driver.GameDriver, 358
- currentHardDifficulty
  - test.driver.HardDifficultyDriver, 399
- currentIntPair
  - test.driver.PairDriver, 482
- currentMediumDifficulty
  - test.driver.MediumDifficultyDriver, 446
- currentPair
  - test.driver.PairDriver, 479
- currentPlayer1
  - domain.DomainCtrl, 257
- currentPlayer2
  - domain.DomainCtrl, 257
- currentStrIntPair
  - test.driver.PairDriver, 482
- currentStrPair
  - test.driver.PairDriver, 481
- currentTurn
  - domain.DomainCtrl, 252
- currentUser
  - domain.DomainCtrl, 256
  - test.driver.UserDriver, 605
- currentUserName
  - view.BotsConsultView, 82
  - view.BotsCreateView, 93
  - view.BotsModifyView, 106
  - view.BotsView, 115
  - view.ConfigConsultView, 128
  - view.ConfigCreateView, 139
  - view.ConfigModifyView, 152
  - view.ConfigView, 192
  - view.GamesCreateView, 380
  - view.GamesView, 388
  - view.PlayView, 518
  - view.RankingConsultView, 534
  - view.RankingView, 557
  - view.RecordConsultView, 565
  - view.UserDeleteView, 599
  - view.UserModifyView, 614
  - view.UserView, 621
- databasesPath
  - repository.Repository, 571
- defaultBoard
  - test.driver.BoardDriver, 51
- delete
  - domain.ConfigurationCtrl, 166
  - repository.ConfigurationRepository, 177
  - repository.ConfigurationRepositoryCtrl, 181
  - repository.PlayerRepository, 500
  - repository.PlayerRepositoryCtrl, 504
- deleteBot
  - domain.DomainCtrl, 234
  - domain.PlayerCtrl, 496
  - view.BotsModifyView, 101, 106
- deleteBotButton

- view.BotsModifyView, 106
- deleteConfig
  - view.ConfigModifyView, 146, 151
- deleteConfigButton
  - view.ConfigModifyView, 151
- deleteConfiguration
  - domain.DomainCtrl, 238
- deleteUser
  - domain.DomainCtrl, 233
  - domain.PlayerCtrl, 495
  - view.UserDeleteView, 595, 598
  - view.UserModifyView, 609, 612
  - view.UserView, 618, 621
- deleteUserButton
  - view.UserDeleteView, 598
  - view.UserModifyView, 613
  - view.UserView, 621
- deleteUserConfirm
  - view.UserDeleteView, 595, 599
- deleteUserConfirmButton
  - view.UserDeleteView, 599
- deleteUserResult
  - view.UserDeleteView, 599
- deserialize
  - test.driver.BoardDriver, 57
  - test.driver.BotDriver, 69
  - test.driver.ConfigurationDriver, 175
  - test.driver.GameDriver, 355
  - test.driver.UserDriver, 602
  - test.unitary.EntryJUnit, 282
  - test.unitary.RankingJUnit, 541
- Difficulty
  - domain.Difficulty, 213
- difficulty
  - domain.Bot, 64
  - domain.Difficulty, 218
- Difficulty.java, 637
- DifficultyCtrl
  - domain.DifficultyCtrl, 220
- difficultyCtrl
  - domain.DomainCtrl, 256
- DifficultyCtrl.java, 638
- difficultyLevel
  - view.BotsCreateView, 92
  - view.BotsModifyView, 105
- difficultyNumber
  - view.BotsConsultView, 81
  - view.BotsCreateView, 92
  - view.BotsModifyView, 105
- domain, 18
- domain.Board, 24
  - Board, 26, 27
  - board, 43
  - canPlaceDiagonal, 40
  - canPlaceHorizontal, 38
  - canPlaceVertical, 39
  - changePieces, 42
  - getBoard, 29
  - getPiecesPlayer1, 29
  - getPiecesPlayer2, 29
  - inversePlayer, 37
  - isValid, 30
  - piecesPlayer1, 44
  - piecesPlayer2, 44
  - placePiece, 32
  - placePieceConfig, 34
  - removePiece, 32
  - serialize, 28
  - surroundingPieces, 34
  - transcribeToCharacters, 36
  - transcribeToPieceType, 36
  - validPositions, 30
- domain.Board.PieceType, 482
  - PLAYER1, 483
  - PLAYER2, 483
- domain.BoardCtrl, 44
  - BoardCtrl, 45
  - getNumPieces, 47
  - isValid, 48
  - placePiece, 45
  - placePieceConfig, 46
  - removePiece, 46
  - validPositions, 48
- domain.Bot, 61
  - Bot, 62
  - creatorID, 65
  - difficulty, 64
  - getCreatorID, 63
  - getDifficulty, 63
  - serialize, 63
  - setDifficulty, 64
- domain.Configuration, 153
  - canEatDiagonally, 161
  - canEatHorizontally, 160
  - canEatVertically, 161
  - Configuration, 154
  - creatorID, 160
  - getCanEatDiagonally, 159
  - getCanEatHorizontally, 157
  - getCanEatVertically, 158
  - getCreatorID, 157
  - getName, 156
  - name, 160
  - serialize, 155
  - setCanEatDiagonally, 159
  - setCanEatHorizontally, 157
  - setCanEatVertically, 158
  - setName, 156
- domain.ConfigurationCtrl, 162
  - ConfigurationCtrl, 163
  - create, 164
  - delete, 166
  - gameRepositoryCtrl, 169
  - getConfiguration, 167
  - getInitialBoard, 168
  - list, 168

- modify, 164
  - repositoryCtrl, 169
  - save, 166
- domain.Difficulty, 211
  - canEatDiagonally, 218
  - canEatHorizontally, 218
  - canEatVertically, 218
  - Difficulty, 213
  - difficulty, 218
  - getCanEatDiagonally, 214
  - getCanEatHorizontally, 214
  - getCanEatVertically, 214
  - getDifficulty, 213
  - getMaxDepth, 215
  - getPieceType, 215
  - inversePieceType, 216
  - maxDepth, 218
  - pieceType, 219
  - place, 217
  - setMaxDepth, 216
- domain.DifficultyCtrl, 219
  - DifficultyCtrl, 220
  - getBestPosition, 221
  - getBestPositionByBot, 220
- domain.DomainCtrl, 222
  - boardCtrl, 256
  - configurationCtrl, 255
  - createBot, 228
  - createConfiguration, 235
  - createEntries, 254
  - createGame, 240
  - createInitialBoard, 236
  - createUser, 227
  - currentBoard, 257
  - currentConfiguration, 257
  - currentGame, 257
  - currentPlayer1, 257
  - currentPlayer2, 257
  - currentTurn, 252
  - currentUser, 256
  - deleteBot, 234
  - deleteConfiguration, 238
  - deleteUser, 233
  - difficultyCtrl, 256
  - DomainCtrl, 226
  - exitApplication, 226
  - exitGame, 227
  - finishGame, 251
  - gameCtrl, 256
  - getBestPosition, 246
  - getBot, 230
  - getConfiguration, 238
  - getGame, 242
  - getInitialBoard, 239
  - getNumPieces, 248
  - getPlayer, 230
  - getPlayingBoard, 242
  - getRanking, 253
  - getUser, 229
  - inversePieceType, 250
  - isValidBoard, 249
  - listBots, 231
  - listConfigurations, 239
  - listGames, 243
  - listRankings, 253
  - listRecords, 254
  - listUsers, 231
  - login, 229
  - logout, 226
  - modifyBot, 233
  - modifyConfiguration, 236
  - modifyInitialBoard, 237
  - modifyUser, 232
  - nextTurn, 251
  - placePiece, 249
  - placePieceConfig, 247
  - play, 244
  - playerCtrl, 255
  - rankingCtrl, 256
  - removePiece, 247
  - saveGame, 241
  - selectGame, 243
  - stringToPieceType, 245
  - surrender, 244
  - validPositions, 248
  - viewBoard, 252
  - viewConfiguration, 240
  - viewGame, 245
  - viewPlayers, 235
  - viewUser, 234
- domain.EasyDifficulty, 263
  - EasyDifficulty, 264
  - evaluation, 264
  - minimax, 266
  - place, 267
- domain.Entry, 277
  - Entry, 278
  - getPlayerID, 279
  - getValue, 280
  - playerID, 281
  - serialize, 279
  - setPlayerID, 280
  - setValue, 281
  - value, 281
- domain.Exceptions, 284
- domain.Exceptions.BadConfirmationException, 23
  - BadConfirmationException, 23
- domain.Exceptions.BotUsedException, 115
  - BotUsedException, 116
- domain.Exceptions.ConfigurationUsedException, 184
  - ConfigurationUsedException, 184
- domain.Exceptions.ExistingConfigurationException, 285
  - ExistingConfigurationException, 286
- domain.Exceptions.ExistingPlayerException, 286
  - ExistingPlayerException, 286
- domain.Exceptions.FinishedGameException, 287

- FinishedGameException, 287
- domain.Exceptions.IncorrectCredentialsException, 400
  - IncorrectCredentialsException, 400
- domain.Exceptions.InexistingConfigurationException, 401
  - InexistingConfigurationException, 401
- domain.Exceptions.InexistingPlayerException, 402
  - InexistingPlayerException, 402
- domain.Exceptions.InvalidBoardException, 424
  - InvalidBoardException, 424
- domain.Exceptions.InvalidConfigurationException, 424
  - InvalidConfigurationException, 425
- domain.Exceptions.InvalidDifficultyException, 425
  - InvalidDifficultyException, 425
- domain.Exceptions.InvalidNameException, 426
  - InvalidNameException, 426
- domain.Exceptions.InvalidPasswordException, 426
  - InvalidPasswordException, 427
- domain.Exceptions.InvalidPlayersException, 427
  - InvalidPlayersException, 427
- domain.Exceptions.InvalidPositionException, 428
  - InvalidPositionException, 428
- domain.Exceptions.InvalidRulesException, 428
  - InvalidRulesException, 429
- domain.Exceptions.NotCreatorException, 468
  - NotCreatorException, 469
- domain.Exceptions.NotPlayerException, 469
  - NotPlayerException, 469
- domain.Exceptions.NotPlayerPieceException, 470
  - NotPlayerPieceException, 470
- domain.Exceptions.NotPlayerTurnException, 470
  - NotPlayerTurnException, 471
- domain.Exceptions.NotStartedGameException, 471
  - NotStartedGameException, 471
- domain.Game, 291
  - checkPlaceRights, 304
  - configurationName, 306
  - createdAt, 307
  - creatorID, 307
  - finish, 303
  - Game, 292, 293
  - getConfigurationName, 297
  - getCreatedAt, 301
  - getCreatorID, 301
  - getName, 295
  - getPlayer1ID, 296
  - getPlayer2ID, 297
  - getState, 299
  - getTurn, 298
  - getWinnerID, 300
  - name, 305
  - nextTurn, 305
  - play, 302
  - player1ID, 305
  - player2ID, 306
  - serialize, 295
  - setConfigurationName, 298
  - setName, 296
  - setState, 300
  - setTurn, 299
  - state, 306
  - surrender, 302
  - turn, 306
  - winnerID, 306
- domain.Game.GameState, 381
  - FINISHED, 382
  - IN\_PROGRESS, 382
  - NOT\_STARTED, 382
- domain.GameCtrl, 339
  - checkPlaceRights, 346
  - configurationRepositoryCtrl, 348
  - create, 340
  - finish, 346
  - GameCtrl, 340
  - getGame, 342
  - getPlayingBoard, 342
  - list, 343
  - nextTurn, 347
  - play, 344
  - playerRepositoryCtrl, 348
  - repositoryCtrl, 348
  - save, 344
  - surrender, 345
- domain.HardDifficulty, 389
  - epsilon, 391
  - HardDifficulty, 389
  - place, 390
  - random, 391
- domain.HardDifficulty.TreeNode, 577
  - canEatDiagonally, 586
  - canEatHorizontally, 585
  - canEatVertically, 586
  - children, 584
  - currentBoard, 585
  - expand, 582
  - getChildren, 580
  - getSelectedPosition, 579
  - getWinRatio, 580
  - isLeaf, 579
  - nVisits, 584
  - pieceType, 585
  - play, 581
  - rollOut, 583
  - rootType, 585
  - select, 581
  - selectedPosition, 585
  - totValue, 584
  - TreeNode, 578
  - updateStats, 583
- domain.MediumDifficulty, 433
  - evaluation, 435
  - MediumDifficulty, 434
  - minimax, 436
  - place, 438
- domain.Player, 483
  - getID, 484

- getIsDeleted, 484
- getName, 484
- id, 486
- isDeleted, 486
- name, 486
- setIsDeleted, 485
- setName, 485
- domain.PlayerCtrl, 487
  - createBot, 489
  - createUser, 488
  - deleteBot, 496
  - deleteUser, 495
  - gameRepositoryCtrl, 497
  - getBot, 492
  - getUser, 492
  - hash, 496
  - listBots, 493
  - listUsers, 493
  - login, 491
  - modifyBot, 494
  - modifyUser, 493
  - PlayerCtrl, 488
  - repositoryCtrl, 497
  - saveBot, 490
  - saveUser, 490
- domain.Ranking, 519
  - addEntry, 524
  - entries, 526
  - getEntries, 525
  - getName, 524
  - getRecord, 522
  - name, 525
  - Ranking, 520, 521
  - removePlayer, 523
  - serialize, 522
  - whereInsert, 523
- domain.Ranking.RankingType, 550
  - INCREMENTAL, 550
  - MULTIPLE, 550
  - UNIQUE, 550
- domain.RankingCtrl, 535
  - createEntry, 538
  - getRanking, 537
  - listRankings, 537
  - listRecords, 538
  - RankingCtrl, 536
  - repositoryCtrl, 539
- domain.User, 587
  - getPassword, 589
  - password, 590
  - serialize, 589
  - setPassword, 590
  - User, 588
- DomainCtrl
  - domain.DomainCtrl, 226
- domainCtrl
  - view.ViewCtrl, 625
- DomainCtrl.java, 638
- drawPiece
  - view.ConsultInitialBoardView, 197
  - view.GameBoardView, 318
  - view.InitialBoardView, 408
  - view.ModifyInitialBoardView, 452
- Driver.java, 639
- EasyDifficulty
  - domain.EasyDifficulty, 264
- EasyDifficulty.java, 639
- easyDifficulty.java, 640
- EasyDifficultyDriver
  - test.driver.EasyDifficultyDriver, 269
- EasyDifficultyDriver.java, 640
- entries
  - domain.Ranking, 526
- Entry
  - domain.Entry, 278
  - test.unitary.EntryJUnit, 282
- Entry.java, 641
- entry.java, 641
- EntryJUnit.java, 642
- epsilon
  - domain.HardDifficulty, 391
- equals
  - util.Pair< F, S >, 475
- evaluation
  - domain.EasyDifficulty, 264
  - domain.MediumDifficulty, 435
- Exceptions.java, 642
- ExistingConfigurationException
  - domain.Exceptions.ExistingConfigurationException, 286
- ExistingPlayerException
  - domain.Exceptions.ExistingPlayerException, 286
- existsGameByConfigurationName
  - repository.GameRepository, 361
  - repository.GameRepositoryCtrl, 367
- existsGameByPlayerID
  - repository.GameRepository, 362
  - repository.GameRepositoryCtrl, 367
- exitApplication
  - domain.DomainCtrl, 226
- exitGame
  - domain.DomainCtrl, 227
- expand
  - domain.HardDifficulty.TreeNode, 582
- f1c1
  - view.ConsultInitialBoardView, 198
  - view.GameBoardView, 321
  - view.InitialBoardView, 409
  - view.ModifyInitialBoardView, 454
- f1c2
  - view.ConsultInitialBoardView, 198
  - view.GameBoardView, 321
  - view.InitialBoardView, 410
  - view.ModifyInitialBoardView, 454
- f1c3



[illegible]





- view.InitialBoardView, [419](#)
  - view.ModifyInitialBoardView, [464](#)
- f7c2
  - view.ConsultInitialBoardView, [208](#)
  - view.GameBoardView, [330](#)
  - view.InitialBoardView, [419](#)
  - view.ModifyInitialBoardView, [464](#)
- f7c3
  - view.ConsultInitialBoardView, [208](#)
  - view.GameBoardView, [331](#)
  - view.InitialBoardView, [419](#)
  - view.ModifyInitialBoardView, [464](#)
- f7c4
  - view.ConsultInitialBoardView, [208](#)
  - view.GameBoardView, [331](#)
  - view.InitialBoardView, [420](#)
  - view.ModifyInitialBoardView, [464](#)
- f7c5
  - view.ConsultInitialBoardView, [209](#)
  - view.GameBoardView, [331](#)
  - view.InitialBoardView, [420](#)
  - view.ModifyInitialBoardView, [464](#)
- f7c6
  - view.ConsultInitialBoardView, [209](#)
  - view.GameBoardView, [331](#)
  - view.InitialBoardView, [420](#)
  - view.ModifyInitialBoardView, [465](#)
- f7c7
  - view.ConsultInitialBoardView, [209](#)
  - view.GameBoardView, [331](#)
  - view.InitialBoardView, [420](#)
  - view.ModifyInitialBoardView, [465](#)
- f7c8
  - view.ConsultInitialBoardView, [209](#)
  - view.GameBoardView, [332](#)
  - view.InitialBoardView, [420](#)
  - view.ModifyInitialBoardView, [465](#)
- f8c1
  - view.ConsultInitialBoardView, [209](#)
  - view.GameBoardView, [332](#)
  - view.InitialBoardView, [421](#)
  - view.ModifyInitialBoardView, [465](#)
- f8c2
  - view.ConsultInitialBoardView, [210](#)
  - view.GameBoardView, [332](#)
  - view.InitialBoardView, [421](#)
  - view.ModifyInitialBoardView, [465](#)
- f8c3
  - view.ConsultInitialBoardView, [210](#)
  - view.GameBoardView, [332](#)
  - view.InitialBoardView, [421](#)
  - view.ModifyInitialBoardView, [466](#)
- f8c4
  - view.ConsultInitialBoardView, [210](#)
  - view.GameBoardView, [332](#)
  - view.InitialBoardView, [421](#)
  - view.ModifyInitialBoardView, [466](#)
- f8c5
  - view.ConsultInitialBoardView, [210](#)
  - view.GameBoardView, [333](#)
  - view.InitialBoardView, [421](#)
  - view.ModifyInitialBoardView, [466](#)
- f8c6
  - view.ConsultInitialBoardView, [210](#)
  - view.GameBoardView, [333](#)
  - view.InitialBoardView, [422](#)
  - view.ModifyInitialBoardView, [466](#)
- f8c7
  - view.ConsultInitialBoardView, [211](#)
  - view.GameBoardView, [333](#)
  - view.InitialBoardView, [422](#)
  - view.ModifyInitialBoardView, [466](#)
- f8c8
  - view.ConsultInitialBoardView, [211](#)
  - view.GameBoardView, [333](#)
  - view.InitialBoardView, [422](#)
  - view.ModifyInitialBoardView, [467](#)
- finish
  - domain.Game, [303](#)
  - domain.GameCtrl, [346](#)
  - test.driver.GameDriver, [357](#)
- FINISHED
  - domain.Game.GameState, [382](#)
- FinishedGameException
  - domain.Exceptions.FinishedGameException, [287](#)
- finishGame
  - domain.DomainCtrl, [251](#)
- first
  - util.Pair< F, S >, [477](#)
- FIXTURE
  - repository.Repository.RepositoryType, [573](#)
- FixtureRepository
  - repository.FixtureRepository, [288](#)
- fixtureRepository
  - test.driver.EasyDifficultyDriver, [276](#)
  - test.driver.HardDifficultyDriver, [400](#)
  - test.driver.MediumDifficultyDriver, [446](#)
- FixtureRepository.java, [643](#)
- fixturesPath
  - repository.Repository, [571](#)
- GAME
  - repository.Repository.RepositoryType, [572](#)
- Game
  - domain.Game, [292](#), [293](#)
- game
  - view.GameBoardView, [337](#)
- Game.java, [644](#)
- game.java, [644](#)
- GameBoardView
  - view.GameBoardView, [312](#)
- GameBoardView.java, [645](#)
- gameChooser
  - view.PlayView, [518](#)
- GameCtrl
  - domain.GameCtrl, [340](#)
- gameCtrl

- domain.DomainCtrl, 256
- GameCtrl.java, 645
- GameDriver
  - test.driver.GameDriver, 349
- GameDriver.java, 646
- GameRepository
  - repository.GameRepository, 359
- GameRepository.java, 646
- GameRepositoryCtrl
  - repository.GameRepositoryCtrl, 365
- gameRepositoryCtrl
  - domain.ConfigurationCtrl, 169
  - domain.PlayerCtrl, 497
- GameRepositoryCtrl.java, 647
- gameResult
  - view.GameBoardView, 335
- games
  - view.BotsConsultView, 76, 79
  - view.BotsCreateView, 86, 90
  - view.BotsModifyView, 97, 103
  - view.BotsView, 110, 113
  - view.ConfigConsultView, 121, 124
  - view.ConfigCreateView, 131, 135
  - view.ConfigModifyView, 143, 148
  - view.ConfigView, 187, 190
  - view.GamesCreateView, 377
  - view.GamesView, 387
  - view.PlayView, 511, 515
  - view.RankingConsultView, 529, 532
  - view.RankingView, 554, 556
  - view.RecordConsultView, 561, 564
  - view.UserDeleteView, 594, 597
  - view.UserModifyView, 608, 611
  - view.UserView, 617, 620
- GamesCreateView
  - view.GamesCreateView, 371
- GamesCreateView.java, 647
- GamesView
  - view.GamesView, 383
- GamesView.java, 648
- get
  - repository.PlayerRepository, 500
  - repository.PlayerRepositoryCtrl, 505
  - repository.RankingRepository, 546
  - repository.RankingRepositoryCtrl, 549
  - repository.Repository, 568
- getBestPosition
  - domain.DifficultyCtrl, 221
  - domain.DomainCtrl, 246
- getBestPositionByBot
  - domain.DifficultyCtrl, 220
- getBoard
  - domain.Board, 29
  - repository.ConfigurationRepository, 178
  - repository.ConfigurationRepositoryCtrl, 182
  - repository.GameRepository, 361
  - repository.GameRepositoryCtrl, 366
- getBot
  - domain.DomainCtrl, 230
  - domain.PlayerCtrl, 492
- getByName
  - repository.PlayerRepository, 501
  - repository.PlayerRepositoryCtrl, 506
- getCanEatDiagonally
  - domain.Configuration, 159
  - domain.Difficulty, 214
  - test.driver.ConfigurationDriver, 174
  - test.driver.EasyDifficultyDriver, 272
  - test.driver.HardDifficultyDriver, 396
  - test.driver.MediumDifficultyDriver, 442
- getCanEatHorizontally
  - domain.Configuration, 157
  - domain.Difficulty, 214
  - test.driver.ConfigurationDriver, 173
  - test.driver.EasyDifficultyDriver, 271
  - test.driver.HardDifficultyDriver, 395
  - test.driver.MediumDifficultyDriver, 442
- getCanEatVertically
  - domain.Configuration, 158
  - domain.Difficulty, 214
  - test.driver.ConfigurationDriver, 173
  - test.driver.EasyDifficultyDriver, 271
  - test.driver.HardDifficultyDriver, 395
  - test.driver.MediumDifficultyDriver, 442
- getChildren
  - domain.HardDifficulty.TreeNode, 580
- getCircle
  - view.ConsultInitialBoardView, 197
  - view.GameBoardView, 320
  - view.InitialBoardView, 409
  - view.ModifyInitialBoardView, 453
- getClickedPos
  - view.GameBoardView, 319
  - view.InitialBoardView, 408
  - view.ModifyInitialBoardView, 453
- getConfiguration
  - domain.ConfigurationCtrl, 167
  - domain.DomainCtrl, 238
  - repository.ConfigurationRepository, 178
  - repository.ConfigurationRepositoryCtrl, 182
- getConfigurationName
  - domain.Game, 297
  - test.driver.GameDriver, 352
- getCreatedAt
  - domain.Game, 301
  - test.driver.GameDriver, 355
- getCreatorID
  - domain.Bot, 63
  - domain.Configuration, 157
  - domain.Game, 301
  - test.driver.BotDriver, 70
  - test.driver.ConfigurationDriver, 172
  - test.driver.GameDriver, 355
- getCurrentBoard
  - test.driver.BoardDriver, 53
- getDifficulty

- domain.Bot, 63
- domain.Difficulty, 213
- test.driver.BotDriver, 69
- test.driver.EasyDifficultyDriver, 271
- test.driver.HardDifficultyDriver, 395
- test.driver.MediumDifficultyDriver, 442
- getEntries
  - domain.Ranking, 525
  - test.unitary.RankingJUnit, 543
- getFirst
  - test.driver.PairDriver, 480
  - util.Pair< F, S >, 476
- getGame
  - domain.DomainCtrl, 242
  - domain.GameCtrl, 342
  - repository.GameRepository, 360
  - repository.GameRepositoryCtrl, 366
- getID
  - domain.Player, 484
  - test.driver.BotDriver, 70
  - test.driver.UserDriver, 603
- getInitialBoard
  - domain.ConfigurationCtrl, 168
  - domain.DomainCtrl, 239
- getIsDeleted
  - domain.Player, 484
  - test.driver.BotDriver, 70
  - test.driver.UserDriver, 603
- getLines
  - repository.FixtureRepository, 289
- getMaxDepth
  - domain.Difficulty, 215
  - test.driver.EasyDifficultyDriver, 272
  - test.driver.HardDifficultyDriver, 396
  - test.driver.MediumDifficultyDriver, 443
- getName
  - domain.Configuration, 156
  - domain.Game, 295
  - domain.Player, 484
  - domain.Ranking, 524
  - test.driver.BotDriver, 69
  - test.driver.ConfigurationDriver, 172
  - test.driver.GameDriver, 351
  - test.driver.UserDriver, 602
  - test.unitary.RankingJUnit, 543
- getNextBestPosition
  - test.driver.EasyDifficultyDriver, 273
  - test.driver.HardDifficultyDriver, 397
  - test.driver.MediumDifficultyDriver, 444
- getNumPieces
  - domain.BoardCtrl, 47
  - domain.DomainCtrl, 248
- getPassword
  - domain.User, 589
  - test.driver.UserDriver, 603
- getPiecesPlayer1
  - domain.Board, 29
- getPiecesPlayer2
  - domain.Board, 29
- getPiecesPlayers
  - test.driver.BoardDriver, 53
- getPieceType
  - domain.Difficulty, 215
  - test.driver.EasyDifficultyDriver, 272
  - test.driver.HardDifficultyDriver, 396
  - test.driver.MediumDifficultyDriver, 443
- getPlayer
  - domain.DomainCtrl, 230
- getPlayer1ID
  - domain.Game, 296
  - test.driver.GameDriver, 352
- getPlayer2ID
  - domain.Game, 297
  - test.driver.GameDriver, 352
- getPlayerID
  - domain.Entry, 279
  - test.unitary.EntryJUnit, 283
- getPlayingBoard
  - domain.DomainCtrl, 242
  - domain.GameCtrl, 342
- getRanking
  - domain.DomainCtrl, 253
  - domain.RankingCtrl, 537
- getRecord
  - domain.Ranking, 522
  - test.unitary.RankingJUnit, 541
- getSecond
  - test.driver.PairDriver, 480
  - util.Pair< F, S >, 476
- getSelectedPosition
  - domain.HardDifficulty.TreeNode, 579
- getState
  - domain.Game, 299
  - test.driver.GameDriver, 354
- getTurn
  - domain.Game, 298
  - test.driver.GameDriver, 353
- getType
  - test.driver.BotDriver, 70
  - test.driver.UserDriver, 603
- getUser
  - domain.DomainCtrl, 229
  - domain.PlayerCtrl, 492
- getValue
  - domain.Entry, 280
  - test.unitary.EntryJUnit, 283
- getWinnerID
  - domain.Game, 300
  - test.driver.GameDriver, 354
- getWinRatio
  - domain.HardDifficulty.TreeNode, 580
- goToGame
  - view.PlayView, 513, 517
- goToGameButton
  - view.PlayView, 517
- goToMenu

- view.ConsultInitialBoardView, 196, 198
- view.GameBoardView, 313, 321
- HardDifficulty
  - domain.HardDifficulty, 389
- HardDifficulty.java, 649
- hardDifficulty.java, 648
- HardDifficultyDriver
  - test.driver.HardDifficultyDriver, 393
- HardDifficultyDriver.java, 649
- hash
  - domain.PlayerCtrl, 496
- id
  - domain.Player, 486
- IN\_PROGRESS
  - domain.Game.GameState, 382
- IncorrectCredentialsException
  - domain.Exceptions.IncorrectCredentialsException, 400
- INCREMENTAL
  - domain.Ranking.RankingType, 550
- InexistingConfigurationException
  - domain.Exceptions.InexistingConfigurationException, 401
- InexistingPlayerException
  - domain.Exceptions.InexistingPlayerException, 402
- info
  - view.PlayView, 516
- InitialBoardView
  - view.InitialBoardView, 406
- InitialBoardView.java, 650
- initialize
  - view.BotsConsultView, 74
  - view.BotsCreateView, 85
  - view.BotsModifyView, 96
  - view.BotsView, 109
  - view.ConfigConsultView, 118
  - view.ConfigCreateView, 130
  - view.ConfigModifyView, 141
  - view.ConfigView, 186
  - view.ConsultInitialBoardView, 196
  - view.GameBoardView, 313
  - view.GamesCreateView, 371
  - view.GamesView, 383
  - view.InitialBoardView, 406
  - view.ModifyInitialBoardView, 451
  - view.PlayView, 509
  - view.RankingConsultView, 528
  - view.RankingView, 552
  - view.RecordConsultView, 560
  - view.UserDeleteView, 593
  - view.UserModifyView, 607
  - view.UserView, 616
- input
  - test.driver.Driver, 260
- inputBool
  - test.driver.Driver, 261
- inputInt
  - test.driver.Driver, 261
- InvalidBoardException
  - domain.Exceptions.InvalidBoardException, 424
- InvalidConfigurationException
  - domain.Exceptions.InvalidConfigurationException, 425
- InvalidDifficultyException
  - domain.Exceptions.InvalidDifficultyException, 425
- InvalidNameException
  - domain.Exceptions.InvalidNameException, 426
- InvalidPasswordException
  - domain.Exceptions.InvalidPasswordException, 427
- InvalidPlayersException
  - domain.Exceptions.InvalidPlayersException, 427
- InvalidPositionException
  - domain.Exceptions.InvalidPositionException, 428
- InvalidRulesException
  - domain.Exceptions.InvalidRulesException, 429
- inversePieceType
  - domain.Difficulty, 216
  - domain.DomainCtrl, 250
- inversePlayer
  - domain.Board, 37
- isDeleted
  - domain.Player, 486
- isLeaf
  - domain.HardDifficulty.TreeNode, 579
- isSpectating
  - view.GameBoardView, 338
- isValid
  - domain.Board, 30
  - domain.BoardCtrl, 48
- isValidBoard
  - domain.DomainCtrl, 249
- isVsBot
  - view.GameBoardView, 338
- list
  - domain.ConfigurationCtrl, 168
  - domain.GameCtrl, 343
  - repository.Repository, 568
- listBoardFixtures
  - test.driver.EasyDifficultyDriver, 274
  - test.driver.HardDifficultyDriver, 398
  - test.driver.MediumDifficultyDriver, 445
- listBots
  - domain.DomainCtrl, 231
  - domain.PlayerCtrl, 493
  - repository.PlayerRepository, 502
  - repository.PlayerRepositoryCtrl, 506
- listConfigurations
  - domain.DomainCtrl, 239
  - repository.ConfigurationRepository, 179
  - repository.ConfigurationRepositoryCtrl, 183
- listFiles
  - repository.FixtureRepository, 288
- listGames
  - domain.DomainCtrl, 243
  - repository.GameRepository, 363

- repository.GameRepositoryCtrl, 368
- listRankings
  - domain.DomainCtrl, 253
  - domain.RankingCtrl, 537
  - repository.RankingRepository, 546
  - repository.RankingRepositoryCtrl, 549
- listRecords
  - domain.DomainCtrl, 254
  - domain.RankingCtrl, 538
- listUsers
  - domain.DomainCtrl, 231
  - domain.PlayerCtrl, 493
  - repository.PlayerRepository, 501
  - repository.PlayerRepositoryCtrl, 506
- loadBoard
  - test.driver.BoardDriver, 52
  - test.driver.EasyDifficultyDriver, 273
  - test.driver.HardDifficultyDriver, 397
  - test.driver.MediumDifficultyDriver, 443
- login
  - view.LoginView, 431
  - view.SignUpView, 574, 575
- login
  - domain.DomainCtrl, 229
  - domain.PlayerCtrl, 491
- loginResult
  - view.LoginView, 432
- LoginView
  - view.LoginView, 430
- LoginView.java, 650
- logout
  - view.BotsConsultView, 78, 82
  - view.BotsCreateView, 89, 93
  - view.BotsModifyView, 101, 106
  - view.BotsView, 112, 115
  - view.ConfigConsultView, 122, 128
  - view.ConfigCreateView, 134, 139
  - view.ConfigModifyView, 147, 152
  - view.ConfigView, 189, 192
  - view.GamesCreateView, 376, 380
  - view.GamesView, 386, 388
  - view.PlayView, 513, 518
  - view.RankingConsultView, 531, 534
  - view.RankingView, 555, 558
  - view.RecordConsultView, 562
  - view.UserDeleteView, 596, 599
  - view.UserModifyView, 610, 614
  - view.UserView, 619, 622
- logout
  - domain.DomainCtrl, 226
- main
  - cmd.driver.board, 24
  - cmd.driver.bot, 65
  - cmd.driver.configuration, 162
  - cmd.driver.easyDifficulty, 262
  - cmd.driver.game, 308
  - cmd.driver.hardDifficulty, 392
  - cmd.driver.mediumDifficulty, 433
  - cmd.driver.pair, 473
  - cmd.driver.user, 587
  - cmd.othello, 472
  - cmd.unitary.entry, 276
  - cmd.unitary.ranking, 519
  - view.ViewCtrl, 624
- mainMenu
  - test.driver.BotDriver, 67
  - test.driver.ConfigurationDriver, 171
  - test.driver.EasyDifficultyDriver, 270
  - test.driver.GameDriver, 350
  - test.driver.HardDifficultyDriver, 394
  - test.driver.MediumDifficultyDriver, 440
  - test.driver.PairDriver, 478
  - test.driver.UserDriver, 601
- maxDepth
  - domain.Difficulty, 218
- MediumDifficulty
  - domain.MediumDifficulty, 434
- MediumDifficulty.java, 651
- mediumDifficulty.java, 651
- MediumDifficultyDriver
  - test.driver.MediumDifficultyDriver, 440
- MediumDifficultyDriver.java, 652
- menu
  - test.driver.Driver, 258
- minimax
  - domain.EasyDifficulty, 266
  - domain.MediumDifficulty, 436
- modify
  - domain.ConfigurationCtrl, 164
- modifyBot
  - domain.DomainCtrl, 233
  - domain.PlayerCtrl, 494
  - view.BotsConsultView, 77, 80
  - view.BotsCreateView, 87, 91
  - view.BotsModifyView, 99, 104
  - view.BotsView, 111, 114
- modifyBotButton
  - view.BotsConsultView, 80
  - view.BotsCreateView, 91
  - view.BotsModifyView, 104
  - view.BotsView, 114
- modifyBotConfirm
  - view.BotsModifyView, 100, 105
- modifyBotConfirmButton
  - view.BotsModifyView, 106
- modifyBotResult
  - view.BotsModifyView, 105
- modifyConfig
  - view.ConfigConsultView, 119, 125
  - view.ConfigCreateView, 132, 136
  - view.ConfigModifyView, 144, 149
  - view.ConfigView, 188, 191
- modifyConfigButton
  - view.ConfigConsultView, 125
  - view.ConfigCreateView, 136
  - view.ConfigModifyView, 149

- view.ConfigView, 191
- modifyConfigConfirm
  - view.ConfigModifyView, 145, 151
- modifyConfigConfirmButton
  - view.ConfigModifyView, 151
- modifyConfigResult
  - view.ConfigModifyView, 152
- modifyConfiguration
  - domain.DomainCtrl, 236
- modifyInitialBoard
  - domain.DomainCtrl, 237
  - view.ConfigModifyView, 145, 150
- modifyInitialBoardButton
  - view.ConfigModifyView, 151
- ModifyInitialBoardView
  - view.ModifyInitialBoardView, 450
- ModifyInitialBoardView.java, 652
- modifyUser
  - domain.DomainCtrl, 232
  - domain.PlayerCtrl, 493
  - view.UserDeleteView, 595, 598
  - view.UserModifyView, 609, 612
  - view.UserView, 618, 621
- modifyUserButton
  - view.UserDeleteView, 598
  - view.UserModifyView, 612
  - view.UserView, 621
- modifyUserConfirm
  - view.UserModifyView, 609, 614
- modifyUserConfirmButton
  - view.UserModifyView, 614
- modifyUserResult
  - view.UserModifyView, 613
- MULTIPLE
  - domain.Ranking.RankingType, 550
- name
  - domain.Configuration, 160
  - domain.Game, 305
  - domain.Player, 486
  - domain.Ranking, 525
  - view.BotsConsultView, 81
  - view.ConfigConsultView, 126
- nameCurrentBoard
  - test.driver.BoardDriver, 60
  - test.driver.EasyDifficultyDriver, 276
  - test.driver.HardDifficultyDriver, 400
  - test.driver.MediumDifficultyDriver, 446
- nbotname
  - view.BotsCreateView, 92
  - view.BotsModifyView, 105
- nconfname
  - view.ConfigCreateView, 137
- newWindow
  - view.ViewCtrl, 624
- nextTurn
  - domain.DomainCtrl, 251
  - domain.Game, 305
  - domain.GameCtrl, 347
  - test.driver.GameDriver, 358
- NOT\_STARTED
  - domain.Game.GameState, 382
- NotCreatorException
  - domain.Exceptions.NotCreatorException, 469
- NotPlayerException
  - domain.Exceptions.NotPlayerException, 469
- NotPlayerPieceException
  - domain.Exceptions.NotPlayerPieceException, 470
- NotPlayerTurnException
  - domain.Exceptions.NotPlayerTurnException, 471
- NotStartedGameException
  - domain.Exceptions.NotStartedGameException, 471
- npassword
  - view.SignUpView, 575
  - view.UserModifyView, 613
- nusername
  - view.SignUpView, 575
  - view.UserModifyView, 613
- nVisits
  - domain.HardDifficulty.TreeNode, 584
- onChangeAssistedMode
  - view.GameBoardView, 320
- onChangeBotChooser
  - view.BotsConsultView, 75
  - view.BotsModifyView, 96
- onChangeConfigChooser
  - view.ConfigConsultView, 120
  - view.ConfigModifyView, 142
- onChangeGameChooser
  - view.PlayView, 511
- onChangeRankingChooser
  - view.RankingConsultView, 530
- othello.java, 653
- Pair
  - util.Pair< F, S >, 474
- Pair.java, 653
- pair.java, 654
- PairDriver
  - test.driver.PairDriver, 478
- PairDriver.java, 654
- password
  - domain.User, 590
  - view.LoginView, 431
  - view.UserDeleteView, 598
- path
  - repository.Repository, 571
- pause
  - test.driver.Driver, 259
- piecesPlayer1
  - domain.Board, 44
- piecesPlayer2
  - domain.Board, 44
- pieceType
  - domain.Difficulty, 219
  - domain.HardDifficulty.TreeNode, 585



- pl1Bot
  - view.GamesCreateView, 378
- pl1User
  - view.GamesCreateView, 378
- pl2Bot
  - view.GamesCreateView, 379
- pl2User
  - view.GamesCreateView, 379
- place
  - domain.Difficulty, 217
  - domain.EasyDifficulty, 267
  - domain.HardDifficulty, 390
  - domain.MediumDifficulty, 438
- placeBlackPieces
  - view.InitialBoardView, 423
  - view.ModifyInitialBoardView, 467
- placePiece
  - domain.Board, 32
  - domain.BoardCtrl, 45
  - domain.DomainCtrl, 249
- placePieceBoard
  - test.driver.BoardDriver, 53
- placePieceConfig
  - domain.Board, 34
  - domain.BoardCtrl, 46
  - domain.DomainCtrl, 247
- placePieceInitialBoard
  - test.driver.BoardDriver, 55
- placeWhitePieces
  - view.InitialBoardView, 423
  - view.ModifyInitialBoardView, 467
- play
  - domain.DomainCtrl, 244
  - domain.Game, 302
  - domain.GameCtrl, 344
  - domain.HardDifficulty.TreeNode, 581
  - test.driver.GameDriver, 356
  - view.BotsConsultView, 76, 79
  - view.BotsCreateView, 86, 90
  - view.BotsModifyView, 98, 103
  - view.BotsView, 110, 113
  - view.ConfigConsultView, 122, 124
  - view.ConfigCreateView, 132, 135
  - view.ConfigModifyView, 144, 148
  - view.ConfigView, 188, 190
  - view.GamesCreateView, 373, 377
  - view.GamesView, 385, 387
  - view.PlayView, 515
  - view.RankingConsultView, 529, 532
  - view.RankingView, 554, 556
  - view.RecordConsultView, 561, 564
  - view.UserDeleteView, 594, 597
  - view.UserModifyView, 608, 612
  - view.UserView, 618, 620
- playAgain
  - test.driver.BoardDriver, 59
- PLAYER
  - repository.Repository.RepositoryType, 572
- Player.java, 655
- PLAYER1
  - domain.Board.PieceType, 483
- player1
  - view.GameBoardView, 336
  - view.PlayView, 515
- player1ID
  - domain.Game, 305
- player1Pieces
  - view.GameBoardView, 336
- player1Turn
  - view.GameBoardView, 336
- player1Type
  - view.GameBoardView, 336
- PLAYER2
  - domain.Board.PieceType, 483
- player2
  - view.GameBoardView, 335
  - view.PlayView, 515
- player2ID
  - domain.Game, 306
- player2Pieces
  - view.GameBoardView, 335
- player2Turn
  - view.GameBoardView, 335
- player2Type
  - view.GameBoardView, 335
- playerColumn
  - view.RankingConsultView, 535
- PlayerCtrl
  - domain.PlayerCtrl, 488
- playerCtrl
  - domain.DomainCtrl, 255
- PlayerCtrl.java, 655
- playerID
  - domain.Entry, 281
- PlayerRepository
  - repository.PlayerRepository, 499
- PlayerRepository.java, 656
- PlayerRepositoryCtrl
  - repository.PlayerRepositoryCtrl, 503
- playerRepositoryCtrl
  - domain.GameCtrl, 348
- PlayerRepositoryCtrl.java, 656
- players
  - view.GameBoardView, 337
- playGame
  - view.PlayView, 517
- playGameButton
  - view.PlayView, 517
- playResult
  - view.PlayView, 517
- PlayView
  - view.PlayView, 509
- PlayView.java, 657
- printBoard
  - test.driver.EasyDifficultyDriver, 274
  - test.driver.HardDifficultyDriver, 398

- test.driver.MediumDifficultyDriver, 445
- printCurrentBoard
  - test.driver.BoardDriver, 57
  - test.driver.EasyDifficultyDriver, 273
  - test.driver.HardDifficultyDriver, 397
  - test.driver.MediumDifficultyDriver, 444
- quitPieces
  - view.InitialBoardView, 423
  - view.ModifyInitialBoardView, 468
- random
  - domain.HardDifficulty, 391
- randomInt
  - test.unitary.RankingJUnit, 541
- RANKING
  - repository.Repository.RepositoryType, 572
- Ranking
  - domain.Ranking, 520, 521
  - test.unitary.RankingJUnit, 540
- ranking
  - view.BotsConsultView, 76, 79
  - view.BotsCreateView, 86, 90
  - view.BotsModifyView, 98, 103
  - view.BotsView, 110, 113
  - view.ConfigConsultView, 121, 124
  - view.ConfigCreateView, 131, 135
  - view.ConfigModifyView, 143, 148
  - view.ConfigView, 187, 190
  - view.GamesCreateView, 372, 377
  - view.GamesView, 385, 387
  - view.PlayView, 512, 515
  - view.RankingConsultView, 532
  - view.RankingView, 556
  - view.RecordConsultView, 564
  - view.UserDeleteView, 594, 597
  - view.UserModifyView, 608, 612
  - view.UserView, 617, 620
- Ranking.java, 657
- ranking.java, 657
- rankingChooser
  - view.RankingConsultView, 534
- rankingColumn
  - view.RecordConsultView, 565
- RankingConsultView
  - view.RankingConsultView, 528
- RankingConsultView.java, 658
- RankingCtrl
  - domain.RankingCtrl, 536
- rankingCtrl
  - domain.DomainCtrl, 256
- RankingCtrl.java, 658
- rankingInfo
  - view.RankingConsultView, 534
- RankingJUnit.java, 659
- RankingRepository
  - repository.RankingRepository, 545
- RankingRepository.java, 659
- RankingRepositoryCtrl
  - repository.RankingRepositoryCtrl, 548
- RankingRepositoryCtrl.java, 660
- RankingView
  - view.RankingView, 552
- RankingView.java, 660
- RecordConsultView
  - view.RecordConsultView, 559
- RecordConsultView.java, 661
- remove
  - repository.Repository, 570
- removePiece
  - domain.Board, 32
  - domain.BoardCtrl, 46
  - domain.DomainCtrl, 247
- removePieceBoard
  - test.driver.BoardDriver, 54
- removePlayer
  - domain.Ranking, 523
- render
  - view.ConsultInitialBoardView, 196
  - view.GameBoardView, 318
  - view.InitialBoardView, 407
  - view.ModifyInitialBoardView, 452
- renderResult
  - view.GameBoardView, 317
- renderState
  - view.GameBoardView, 316
- Repository
  - repository.Repository, 567
- repository, 19
  - repository.ConfigurationRepositoryCtrl, 183
  - repository.GameRepositoryCtrl, 369
  - repository.PlayerRepositoryCtrl, 507
  - repository.RankingRepositoryCtrl, 550
- repository.ConfigurationRepository, 175
  - ConfigurationRepository, 176
  - delete, 177
  - getBoard, 178
  - getConfiguration, 178
  - listConfigurations, 179
  - save, 177
- repository.ConfigurationRepositoryCtrl, 180
  - ConfigurationRepositoryCtrl, 180
  - delete, 181
  - getBoard, 182
  - getConfiguration, 182
  - listConfigurations, 183
  - repository, 183
  - save, 181
- repository.FixtureRepository, 287
  - boardFileToJSON, 290
  - FixtureRepository, 288
  - getLines, 289
  - listFiles, 288
- repository.GameRepository, 359
  - existsGameByConfigurationName, 361
  - existsGameByPlayerID, 362
  - GameRepository, 359



- getBoard, 361
- getGame, 360
- listGames, 363
- save, 360
- repository.GameRepositoryCtrl, 364
  - existsGameByConfigurationName, 367
  - existsGameByPlayerID, 367
  - GameRepositoryCtrl, 365
  - getBoard, 366
  - getGame, 366
  - listGames, 368
  - repository, 369
  - save, 365
- Repository.java, 661
- repository.PlayerRepository, 498
  - delete, 500
  - get, 500
  - getByName, 501
  - listBots, 502
  - listUsers, 501
  - PlayerRepository, 499
  - save, 499
- repository.PlayerRepositoryCtrl, 503
  - delete, 504
  - get, 505
  - getByName, 506
  - listBots, 506
  - listUsers, 506
  - PlayerRepositoryCtrl, 503
  - repository, 507
  - save, 504
- repository.RankingRepository, 544
  - get, 546
  - listRankings, 546
  - RankingRepository, 545
  - save, 545
- repository.RankingRepositoryCtrl, 547
  - get, 549
  - listRankings, 549
  - RankingRepositoryCtrl, 548
  - repository, 550
  - save, 548
- repository.Repository, 566
  - createOrUpdate, 569
  - databasesPath, 571
  - fixturesPath, 571
  - get, 568
  - list, 568
  - path, 571
  - remove, 570
  - Repository, 567
  - type, 571
- repository.Repository.RepositoryType, 572
  - CONFIGURATION, 572
  - FIXTURE, 573
  - GAME, 572
  - PLAYER, 572
  - RANKING, 572
- repositoryCtrl
  - domain.ConfigurationCtrl, 169
  - domain.GameCtrl, 348
  - domain.PlayerCtrl, 497
  - domain.RankingCtrl, 539
- resetPairs
  - test.driver.PairDriver, 479
- rollOut
  - domain.HardDifficulty.TreeNode, 583
- rootType
  - domain.HardDifficulty.TreeNode, 585
- rpassword
  - view.SignUpView, 576
  - view.UserModifyView, 613
- save
  - domain.ConfigurationCtrl, 166
  - domain.GameCtrl, 344
  - repository.ConfigurationRepository, 177
  - repository.ConfigurationRepositoryCtrl, 181
  - repository.GameRepository, 360
  - repository.GameRepositoryCtrl, 365
  - repository.PlayerRepository, 499
  - repository.PlayerRepositoryCtrl, 504
  - repository.RankingRepository, 545
  - repository.RankingRepositoryCtrl, 548
  - view.GameBoardView, 314, 333
  - view.InitialBoardView, 407, 422
  - view.ModifyInitialBoardView, 451, 467
- saveBot
  - domain.PlayerCtrl, 490
- saveButton
  - view.GameBoardView, 334
  - view.InitialBoardView, 422
  - view.ModifyInitialBoardView, 467
- saveGame
  - domain.DomainCtrl, 241
- saveInitialBoardResult
  - view.InitialBoardView, 423
  - view.ModifyInitialBoardView, 468
- saveUser
  - domain.PlayerCtrl, 490
- second
  - util.Pair< F, S >, 477
- select
  - domain.HardDifficulty.TreeNode, 581
- selectedPosition
  - domain.HardDifficulty.TreeNode, 585
- selectGame
  - domain.DomainCtrl, 243
- serialize
  - domain.Board, 28
  - domain.Bot, 63
  - domain.Configuration, 155
  - domain.Entry, 279
  - domain.Game, 295
  - domain.Ranking, 522
  - domain.User, 589
  - test.driver.BotDriver, 69

- test.driver.ConfigurationDriver, 174
- test.driver.GameDriver, 355
- test.driver.UserDriver, 602
- test.unitary.EntryJUnit, 283
- test.unitary.RankingJUnit, 541
- serializeBoard
  - test.driver.BoardDriver, 52
- setCanEatDiagonally
  - domain.Configuration, 159
  - test.driver.ConfigurationDriver, 174
- setCanEatHorizontally
  - domain.Configuration, 157
  - test.driver.ConfigurationDriver, 173
- setCanEatVertically
  - domain.Configuration, 158
  - test.driver.ConfigurationDriver, 173
- setConfigurationName
  - domain.Game, 298
  - test.driver.GameDriver, 353
- setDifficulty
  - domain.Bot, 64
  - test.driver.BotDriver, 71
- setIsDeleted
  - domain.Player, 485
  - test.driver.BotDriver, 71
  - test.driver.UserDriver, 604
- setMaxDepth
  - domain.Difficulty, 216
  - test.driver.EasyDifficultyDriver, 272
  - test.driver.HardDifficultyDriver, 396
  - test.driver.MediumDifficultyDriver, 443
- setName
  - domain.Configuration, 156
  - domain.Game, 296
  - domain.Player, 485
  - test.driver.BotDriver, 71
  - test.driver.ConfigurationDriver, 172
  - test.driver.GameDriver, 352
  - test.driver.UserDriver, 604
- setPassword
  - domain.User, 590
  - test.driver.UserDriver, 604
- setPlayerID
  - domain.Entry, 280
  - test.unitary.EntryJUnit, 283
- setState
  - domain.Game, 300
  - test.driver.GameDriver, 354
- setTurn
  - domain.Game, 299
  - test.driver.GameDriver, 353
- setValue
  - domain.Entry, 281
  - test.unitary.EntryJUnit, 284
- showLevel
  - view.BotsCreateView, 88
  - view.BotsModifyView, 99
- signIn
  - view.LogInView, 430, 432
- signInButton
  - view.LogInView, 432
- signUp
  - view.LogInView, 430, 431
  - view.SignUpView, 574, 575
- signUp2
  - view.SignUpView, 576
- signInButton
  - view.SignUpView, 576
- signInResult
  - view.SignUpView, 576
- SignUpView
  - view.SignUpView, 574
- SignUpView.java, 662
- stage
  - view.ViewCtrl, 625
- start
  - test.driver.BoardDriver, 50
  - test.driver.BotDriver, 67
  - test.driver.ConfigurationDriver, 170
  - test.driver.EasyDifficultyDriver, 269
  - test.driver.GameDriver, 350
  - test.driver.HardDifficultyDriver, 394
  - test.driver.MediumDifficultyDriver, 440
  - test.driver.PairDriver, 478
  - test.driver.UserDriver, 601
  - view.ViewCtrl, 623
- state
  - domain.Game, 306
  - view.PlayView, 516
- stringToPieceType
  - domain.DomainCtrl, 245
- surrender
  - domain.DomainCtrl, 244
  - domain.Game, 302
  - domain.GameCtrl, 345
  - test.driver.GameDriver, 356
  - view.GameBoardView, 314, 334
- surrenderButton
  - view.GameBoardView, 334
- surroundingPieces
  - domain.Board, 34
- table
  - view.RankingConsultView, 535
  - view.RecordConsultView, 565
- test, 19
- test.driver, 20
- test.driver.BoardDriver, 49
  - addPositions, 56
  - BoardDriver, 50
  - correctBoolean, 60
  - correctNumber, 58
  - correctPieceType, 59
  - createBoard, 51
  - currentBoard, 60
  - defaultBoard, 51
  - deserialize, 57

- getCurrentBoard, 53
- getPiecesPlayers, 53
- loadBoard, 52
- nameCurrentBoard, 60
- placePieceBoard, 53
- placePieceInitialBoard, 55
- playAgain, 59
- printCurrentBoard, 57
- removePieceBoard, 54
- serializeBoard, 52
- start, 50
- transcribeToCharacters, 58
- validBoard, 55
- test.driver.BotDriver, 66
  - BotDriver, 67
  - createBot, 68
  - currentBot, 72
  - deserialize, 69
  - getCreatorID, 70
  - getDifficulty, 69
  - getID, 70
  - getIsDeleted, 70
  - getName, 69
  - getType, 70
  - mainMenu, 67
  - serialize, 69
  - setDifficulty, 71
  - setIsDeleted, 71
  - setName, 71
  - start, 67
- test.driver.ConfigurationDriver, 169
  - ConfigurationDriver, 170
  - create, 171
  - currentConfiguration, 175
  - deserialize, 175
  - getCanEatDiagonally, 174
  - getCanEatHorizontally, 173
  - getCanEatVertically, 173
  - getCreatorID, 172
  - getName, 172
  - mainMenu, 171
  - serialize, 174
  - setCanEatDiagonally, 174
  - setCanEatHorizontally, 173
  - setCanEatVertically, 173
  - setName, 172
  - start, 170
- test.driver.Driver, 258
  - clear, 259
  - input, 260
  - inputBool, 261
  - inputInt, 261
  - menu, 258
  - pause, 259
- test.driver.EasyDifficultyDriver, 268
  - create, 270
  - currentBoard, 275
  - currentEasyDifficulty, 275
  - EasyDifficultyDriver, 269
  - fixtureRepository, 276
  - getCanEatDiagonally, 272
  - getCanEatHorizontally, 271
  - getCanEatVertically, 271
  - getDifficulty, 271
  - getMaxDepth, 272
  - getNextBestPosition, 273
  - getPieceType, 272
  - listBoardFixtures, 274
  - loadBoard, 273
  - mainMenu, 270
  - nameCurrentBoard, 276
  - printBoard, 274
  - printCurrentBoard, 273
  - setMaxDepth, 272
  - start, 269
  - transcribeToCharacters, 275
- test.driver.GameDriver, 348
  - checkPlaceRights, 357
  - create, 351
  - currentGame, 358
  - deserialize, 355
  - finish, 357
  - GameDriver, 349
  - getConfigurationName, 352
  - getCreatedAt, 355
  - getCreatorID, 355
  - getName, 351
  - getPlayer1ID, 352
  - getPlayer2ID, 352
  - getState, 354
  - getTurn, 353
  - getWinnerID, 354
  - mainMenu, 350
  - nextTurn, 358
  - play, 356
  - serialize, 355
  - setConfigurationName, 353
  - setName, 352
  - setState, 354
  - setTurn, 353
  - start, 350
  - surrender, 356
- test.driver.HardDifficultyDriver, 392
  - create, 394
  - currentBoard, 399
  - currentHardDifficulty, 399
  - fixtureRepository, 400
  - getCanEatDiagonally, 396
  - getCanEatHorizontally, 395
  - getCanEatVertically, 395
  - getDifficulty, 395
  - getMaxDepth, 396
  - getNextBestPosition, 397
  - getPieceType, 396
  - HardDifficultyDriver, 393
  - listBoardFixtures, 398

- loadBoard, 397
- mainMenu, 394
- nameCurrentBoard, 400
- printBoard, 398
- printCurrentBoard, 397
- setMaxDepth, 396
- start, 394
- transcribeToCharacters, 399
- test.driver.MediumDifficultyDriver, 439
  - create, 441
  - currentBoard, 446
  - currentMediumDifficulty, 446
  - fixtureRepository, 446
  - getCanEatDiagonally, 442
  - getCanEatHorizontally, 442
  - getCanEatVertically, 442
  - getDifficulty, 442
  - getMaxDepth, 443
  - getNextBestPosition, 444
  - getPieceType, 443
  - listBoardFixtures, 445
  - loadBoard, 443
  - mainMenu, 440
  - MediumDifficultyDriver, 440
  - nameCurrentBoard, 446
  - printBoard, 445
  - printCurrentBoard, 444
  - setMaxDepth, 443
  - start, 440
  - transcribeToCharacters, 445
- test.driver.PairDriver, 477
  - comparePair, 481
  - createIntPair, 480
  - createStrIntPair, 480
  - createStrPair, 479
  - currentIntPair, 482
  - currentPair, 479
  - currentStrIntPair, 482
  - currentStrPair, 481
  - getFirst, 480
  - getSecond, 480
  - mainMenu, 478
  - PairDriver, 478
  - resetPairs, 479
  - start, 478
- test.driver.UserDriver, 600
  - createUser, 602
  - currentUser, 605
  - deserialize, 602
  - getID, 603
  - getIsDeleted, 603
  - getName, 602
  - getPassword, 603
  - getType, 603
  - mainMenu, 601
  - serialize, 602
  - setIsDeleted, 604
  - setName, 604
  - setPassword, 604
  - start, 601
  - UserDriver, 600
- test.unitary, 20
- test.unitary.EntryJUnit, 282
  - deserialize, 282
  - Entry, 282
  - getPlayerID, 283
  - getValue, 283
  - serialize, 283
  - setPlayerID, 283
  - setValue, 284
- test.unitary.RankingJUnit, 540
  - addEntry, 542
  - deserialize, 541
  - getEntries, 543
  - getName, 543
  - getRecord, 541
  - randomInt, 541
  - Ranking, 540
  - serialize, 541
- tielcon
  - view.GameBoardView, 334
- timer
  - view.GameBoardView, 338
- toString
  - util.Pair< F, S >, 475
- totValue
  - domain.HardDifficulty.TreeNode, 584
- transcribeToCharacters
  - domain.Board, 36
  - test.driver.BoardDriver, 58
  - test.driver.EasyDifficultyDriver, 275
  - test.driver.HardDifficultyDriver, 399
  - test.driver.MediumDifficultyDriver, 445
- transcribeToPieceType
  - domain.Board, 36
- transform
  - view.GameBoardView, 315
  - view.InitialBoardView, 406
  - view.ModifyInitialBoardView, 451
- TreeNode
  - domain.HardDifficulty.TreeNode, 578
- turn
  - domain.Game, 306
- turnPieceType
  - view.GameBoardView, 338
- turnPlayerID
  - view.GameBoardView, 337
- turnPlayerIsBot
  - view.GameBoardView, 338
- type
  - repository.Repository, 571
- UNIQUE
  - domain.Ranking.RankingType, 550
- updateStats
  - domain.HardDifficulty.TreeNode, 583
- User

- domain.User, 588
- user
  - view.BotsConsultView, 74, 78
  - view.BotsCreateView, 85, 90
  - view.BotsModifyView, 96, 102
  - view.BotsView, 109, 112
  - view.ConfigConsultView, 118, 123
  - view.ConfigCreateView, 130, 135
  - view.ConfigModifyView, 142, 148
  - view.ConfigView, 186, 190
  - view.GameBoardView, 337
  - view.GamesCreateView, 371, 376
  - view.GamesView, 384, 386
  - view.PlayView, 510, 514
  - view.RankingConsultView, 528, 532
  - view.RankingView, 553, 556
  - view.RecordConsultView, 560, 563
  - view.UserDeleteView, 597
  - view.UserModifyView, 611
  - view.UserView, 619
- User.java, 662
- user.java, 663
- userChooser1
  - view.GamesCreateView, 373, 378
- userChooser2
  - view.GamesCreateView, 374, 379
- UserDeleteView
  - view.UserDeleteView, 592
- UserDeleteView.java, 663
- UserDriver
  - test.driver.UserDriver, 600
- UserDriver.java, 663
- userMap
  - view.GamesCreateView, 381
- UserModifyView
  - view.UserModifyView, 606
- UserModifyView.java, 664
- username
  - view.LogInView, 431
- UserView
  - view.UserView, 616
- UserView.java, 664
- util, 20
- util.Pair< F, S >, 473
  - equals, 475
  - first, 477
  - getFirst, 476
  - getSecond, 476
  - Pair, 474
  - second, 477
  - toString, 475
- validBoard
  - test.driver.BoardDriver, 55
- validPositions
  - domain.Board, 30
  - domain.BoardCtrl, 48
  - domain.DomainCtrl, 248
- value
  - domain.Entry, 281
- valueColumn
  - view.RankingConsultView, 535
  - view.RecordConsultView, 566
- view, 21
- view.BotsConsultView, 72
  - botChooser, 81
  - botMap, 82
  - bots, 79
  - BotsConsultView, 74
  - config, 75, 79
  - consultBot, 77, 80
  - consultBotButton, 81
  - consultBotResult, 82
  - consultConfigResult, 81
  - createBot, 77, 80
  - createBotButton, 80
  - creator, 82
  - currentUserName, 82
  - difficultyNumber, 81
  - games, 76, 79
  - initialize, 74
  - logOut, 78, 82
  - modifyBot, 77, 80
  - modifyBotButton, 80
  - name, 81
  - onChangeBotChooser, 75
  - play, 76, 79
  - ranking, 76, 79
  - user, 74, 78
- view.BotsCreateView, 83
  - bots, 90
  - BotsCreateView, 85
  - config, 85, 90
  - consultBot, 87, 91
  - consultBotButton, 92
  - createBot, 87, 91
  - createBotButton, 91
  - createBotConfirm, 88, 93
  - createBotConfirmButton, 93
  - createBotResult, 92
  - currentUserName, 93
  - difficultyLevel, 92
  - difficultyNumber, 92
  - games, 86, 90
  - initialize, 85
  - logOut, 89, 93
  - modifyBot, 87, 91
  - modifyBotButton, 91
  - nbotname, 92
  - play, 86, 90
  - ranking, 86, 90
  - showLevel, 88
  - user, 85, 90
- view.BotsModifyView, 94
  - botChooser, 104
  - botMap, 107
  - bots, 102

- BotsModifyView, 95
- config, 97, 102
- consultBot, 99, 104
- consultBotButton, 104
- createBot, 98, 103
- createBotButton, 103
- currentUserName, 106
- deleteBot, 101, 106
- deleteBotButton, 106
- difficultyLevel, 105
- difficultyNumber, 105
- games, 97, 103
- initialize, 96
- logOut, 101, 106
- modifyBot, 99, 104
- modifyBotButton, 104
- modifyBotConfirm, 100, 105
- modifyBotConfirmButton, 106
- modifyBotResult, 105
- nbotname, 105
- onChangeBotChooser, 96
- play, 98, 103
- ranking, 98, 103
- showLevel, 99
- user, 96, 102
- view.BotsView, 107
  - bots, 113
  - BotsView, 108
  - config, 109, 113
  - consultBot, 111, 114
  - consultBotButton, 115
  - createBot, 111, 114
  - createBotButton, 114
  - currentUserName, 115
  - games, 110, 113
  - initialize, 109
  - logOut, 112, 115
  - modifyBot, 111, 114
  - modifyBotButton, 114
  - play, 110, 113
  - ranking, 110, 113
  - user, 109, 112
- view.ConfigConsultView, 116
  - bots, 120, 123
  - ced, 127
  - ceh, 126
  - cev, 126
  - checkInitialBoard, 122, 127
  - checkInitialBoardButton, 127
  - config, 123
  - configChooser, 126
  - ConfigConsultView, 118
  - consultConfig, 119, 125
  - consultConfigButton, 125
  - consultConfigConfirm, 125
  - consultConfigConfirmButton, 126
  - consultConfigResult, 127
  - createConfig, 119, 124
  - createConfigButton, 124
  - creator, 127
  - currentUserName, 128
  - games, 121, 124
  - initialize, 118
  - logOut, 122, 128
  - modifyConfig, 119, 125
  - modifyConfigButton, 125
  - name, 126
  - onChangeConfigChooser, 120
  - play, 122, 124
  - ranking, 121, 124
  - user, 118, 123
- view.ConfigCreateView, 128
  - bots, 131, 135
  - canEatDiagonally, 138
  - canEatHorizontally, 137
  - canEatVertically, 138
  - config, 135
  - ConfigCreateView, 130
  - consultConfig, 133, 136
  - consultConfigButton, 137
  - createConfig, 132, 136
  - createConfigButton, 136
  - createConfigConfirm, 133, 138
  - createConfigConfirmButton, 138
  - createConfigResult, 138
  - createInitialBoard, 133, 137
  - createInitialBoardButton, 137
  - currentUserName, 139
  - games, 131, 135
  - initialize, 130
  - logOut, 134, 139
  - modifyConfig, 132, 136
  - modifyConfigButton, 136
  - nconfname, 137
  - play, 132, 135
  - ranking, 131, 135
  - user, 130, 135
- view.ConfigModifyView, 139
  - bots, 142, 148
  - canEatDiagonally, 150
  - canEatHorizontally, 150
  - canEatVertically, 150
  - config, 148
  - configChooser, 152
  - ConfigModifyView, 141
  - consultConfig, 145, 149
  - consultConfigButton, 150
  - createConfig, 144, 149
  - createConfigButton, 149
  - currentUserName, 152
  - deleteConfig, 146, 151
  - deleteConfigButton, 151
  - games, 143, 148
  - initialize, 141
  - logOut, 147, 152
  - modifyConfig, 144, 149

- modifyConfigButton, 149
- modifyConfigConfirm, 145, 151
- modifyConfigConfirmButton, 151
- modifyConfigResult, 152
- modifyInitialBoard, 145, 150
- modifyInitialBoardButton, 151
- onChangeConfigChooser, 142
- play, 144, 148
- ranking, 143, 148
- user, 142, 148
- view.ConfigView, 185
  - bots, 187, 190
  - config, 190
  - ConfigView, 186
  - consultConfig, 189, 191
  - consultConfigButton, 192
  - createConfig, 188, 191
  - createConfigButton, 191
  - currentUserName, 192
  - games, 187, 190
  - initialize, 186
  - logOut, 189, 192
  - modifyConfig, 188, 191
  - modifyConfigButton, 191
  - play, 188, 190
  - ranking, 187, 190
  - user, 186, 190
- view.ConsultInitialBoardView, 192
  - board, 211
  - ConsultInitialBoardView, 196
  - drawPiece, 197
  - f1c1, 198
  - f1c2, 198
  - f1c3, 199
  - f1c4, 199
  - f1c5, 199
  - f1c6, 199
  - f1c7, 199
  - f1c8, 200
  - f2c1, 200
  - f2c2, 200
  - f2c3, 200
  - f2c4, 200
  - f2c5, 201
  - f2c6, 201
  - f2c7, 201
  - f2c8, 201
  - f3c1, 201
  - f3c2, 202
  - f3c3, 202
  - f3c4, 202
  - f3c5, 202
  - f3c6, 202
  - f3c7, 203
  - f3c8, 203
  - f4c1, 203
  - f4c2, 203
  - f4c3, 203
  - f4c4, 204
  - f4c5, 204
  - f4c6, 204
  - f4c7, 204
  - f4c8, 204
  - f5c1, 205
  - f5c2, 205
  - f5c3, 205
  - f5c4, 205
  - f5c5, 205
  - f5c6, 206
  - f5c7, 206
  - f5c8, 206
  - f6c1, 206
  - f6c2, 206
  - f6c3, 207
  - f6c4, 207
  - f6c5, 207
  - f6c6, 207
  - f6c7, 207
  - f6c8, 208
  - f7c1, 208
  - f7c2, 208
  - f7c3, 208
  - f7c4, 208
  - f7c5, 209
  - f7c6, 209
  - f7c7, 209
  - f7c8, 209
  - f8c1, 209
  - f8c2, 210
  - f8c3, 210
  - f8c4, 210
  - f8c5, 210
  - f8c6, 210
  - f8c7, 211
  - f8c8, 211
  - getCircle, 197
  - goToMenu, 196, 198
  - initialize, 196
  - render, 196
- view.GameBoardView, 308
  - assistedMode, 336
  - board, 337
  - drawPiece, 318
  - f1c1, 321
  - f1c2, 321
  - f1c3, 321
  - f1c4, 321
  - f1c5, 321
  - f1c6, 322
  - f1c7, 322
  - f1c8, 322
  - f2c1, 322
  - f2c2, 322
  - f2c3, 323
  - f2c4, 323
  - f2c5, 323



- f2c6, [323](#)
- f2c7, [323](#)
- f2c8, [324](#)
- f3c1, [324](#)
- f3c2, [324](#)
- f3c3, [324](#)
- f3c4, [324](#)
- f3c5, [325](#)
- f3c6, [325](#)
- f3c7, [325](#)
- f3c8, [325](#)
- f4c1, [325](#)
- f4c2, [326](#)
- f4c3, [326](#)
- f4c4, [326](#)
- f4c5, [326](#)
- f4c6, [326](#)
- f4c7, [327](#)
- f4c8, [327](#)
- f5c1, [327](#)
- f5c2, [327](#)
- f5c3, [327](#)
- f5c4, [328](#)
- f5c5, [328](#)
- f5c6, [328](#)
- f5c7, [328](#)
- f5c8, [328](#)
- f6c1, [329](#)
- f6c2, [329](#)
- f6c3, [329](#)
- f6c4, [329](#)
- f6c5, [329](#)
- f6c6, [330](#)
- f6c7, [330](#)
- f6c8, [330](#)
- f7c1, [330](#)
- f7c2, [330](#)
- f7c3, [331](#)
- f7c4, [331](#)
- f7c5, [331](#)
- f7c6, [331](#)
- f7c7, [331](#)
- f7c8, [332](#)
- f8c1, [332](#)
- f8c2, [332](#)
- f8c3, [332](#)
- f8c4, [332](#)
- f8c5, [333](#)
- f8c6, [333](#)
- f8c7, [333](#)
- f8c8, [333](#)
- game, [337](#)
- GameBoardView, [312](#)
- gameResult, [335](#)
- getCircle, [320](#)
- getClickedPos, [319](#)
- goToMenu, [313](#), [321](#)
- initialize, [313](#)
- isSpectating, [338](#)
- isVsBot, [338](#)
- onChangeAssistedMode, [320](#)
- player1, [336](#)
- player1Pieces, [336](#)
- player1Turn, [336](#)
- player1Type, [336](#)
- player2, [335](#)
- player2Pieces, [335](#)
- player2Turn, [335](#)
- player2Type, [335](#)
- players, [337](#)
- render, [318](#)
- renderResult, [317](#)
- renderState, [316](#)
- save, [314](#), [333](#)
- saveButton, [334](#)
- surrender, [314](#), [334](#)
- surrenderButton, [334](#)
- tielcon, [334](#)
- timer, [338](#)
- transform, [315](#)
- turnPieceType, [338](#)
- turnPlayerID, [337](#)
- turnPlayerIsBot, [338](#)
- user, [337](#)
- winIcon, [334](#)
- view.GamesCreateView, [369](#)
  - botChooser1, [374](#), [379](#)
  - botChooser2, [374](#), [379](#)
  - botMap, [381](#)
  - bots, [372](#), [376](#)
  - config, [372](#), [377](#)
  - configChooser, [378](#)
  - createGame, [373](#), [377](#)
  - createGameButton, [378](#)
  - createGameConfirm, [375](#), [380](#)
  - createGameConfirmButton, [380](#)
  - createGameResult, [380](#)
  - currentUserName, [380](#)
  - games, [377](#)
  - GamesCreateView, [371](#)
  - initialize, [371](#)
  - logOut, [376](#), [380](#)
  - pl1Bot, [378](#)
  - pl1User, [378](#)
  - pl2Bot, [379](#)
  - pl2User, [379](#)
  - play, [373](#), [377](#)
  - ranking, [372](#), [377](#)
  - user, [371](#), [376](#)
  - userChooser1, [373](#), [378](#)
  - userChooser2, [374](#), [379](#)
  - userMap, [381](#)
- view.GamesView, [382](#)
  - bots, [384](#), [387](#)
  - config, [384](#), [387](#)
  - createGame, [385](#), [388](#)



- createGameButton, [388](#)
- currentUserName, [388](#)
- games, [387](#)
- GamesView, [383](#)
- initialize, [383](#)
- logOut, [386](#), [388](#)
- play, [385](#), [387](#)
- ranking, [385](#), [387](#)
- user, [384](#), [386](#)
- view.InitialBoardView, [402](#)
  - board, [423](#)
  - drawPiece, [408](#)
  - f1c1, [409](#)
  - f1c2, [410](#)
  - f1c3, [410](#)
  - f1c4, [410](#)
  - f1c5, [410](#)
  - f1c6, [410](#)
  - f1c7, [411](#)
  - f1c8, [411](#)
  - f2c1, [411](#)
  - f2c2, [411](#)
  - f2c3, [411](#)
  - f2c4, [412](#)
  - f2c5, [412](#)
  - f2c6, [412](#)
  - f2c7, [412](#)
  - f2c8, [412](#)
  - f3c1, [413](#)
  - f3c2, [413](#)
  - f3c3, [413](#)
  - f3c4, [413](#)
  - f3c5, [413](#)
  - f3c6, [414](#)
  - f3c7, [414](#)
  - f3c8, [414](#)
  - f4c1, [414](#)
  - f4c2, [414](#)
  - f4c3, [415](#)
  - f4c4, [415](#)
  - f4c5, [415](#)
  - f4c6, [415](#)
  - f4c7, [415](#)
  - f4c8, [416](#)
  - f5c1, [416](#)
  - f5c2, [416](#)
  - f5c3, [416](#)
  - f5c4, [416](#)
  - f5c5, [417](#)
  - f5c6, [417](#)
  - f5c7, [417](#)
  - f5c8, [417](#)
  - f6c1, [417](#)
  - f6c2, [418](#)
  - f6c3, [418](#)
  - f6c4, [418](#)
  - f6c5, [418](#)
  - f6c6, [418](#)
  - f6c7, [419](#)
  - f6c8, [419](#)
  - f7c1, [419](#)
  - f7c2, [419](#)
  - f7c3, [419](#)
  - f7c4, [420](#)
  - f7c5, [420](#)
  - f7c6, [420](#)
  - f7c7, [420](#)
  - f7c8, [420](#)
  - f8c1, [421](#)
  - f8c2, [421](#)
  - f8c3, [421](#)
  - f8c4, [421](#)
  - f8c5, [421](#)
  - f8c6, [422](#)
  - f8c7, [422](#)
  - f8c8, [422](#)
  - getCircle, [409](#)
  - getClickedPos, [408](#)
  - InitialBoardView, [406](#)
  - initialize, [406](#)
  - placeBlackPieces, [423](#)
  - placeWhitePieces, [423](#)
  - quitPieces, [423](#)
  - render, [407](#)
  - save, [407](#), [422](#)
  - saveButton, [422](#)
  - saveInitialBoardResult, [423](#)
  - transform, [406](#)
- view.LogInView, [429](#)
  - logIn, [431](#)
  - logInResult, [432](#)
  - LogInView, [430](#)
  - password, [431](#)
  - signIn, [430](#), [432](#)
  - signInButton, [432](#)
  - signUp, [430](#), [431](#)
  - username, [431](#)
- view.ModifyInitialBoardView, [447](#)
  - board, [468](#)
  - drawPiece, [452](#)
  - f1c1, [454](#)
  - f1c2, [454](#)
  - f1c3, [454](#)
  - f1c4, [455](#)
  - f1c5, [455](#)
  - f1c6, [455](#)
  - f1c7, [455](#)
  - f1c8, [455](#)
  - f2c1, [456](#)
  - f2c2, [456](#)
  - f2c3, [456](#)
  - f2c4, [456](#)
  - f2c5, [456](#)
  - f2c6, [457](#)
  - f2c7, [457](#)
  - f2c8, [457](#)

- f3c1, [457](#)
- f3c2, [457](#)
- f3c3, [458](#)
- f3c4, [458](#)
- f3c5, [458](#)
- f3c6, [458](#)
- f3c7, [458](#)
- f3c8, [459](#)
- f4c1, [459](#)
- f4c2, [459](#)
- f4c3, [459](#)
- f4c4, [459](#)
- f4c5, [460](#)
- f4c6, [460](#)
- f4c7, [460](#)
- f4c8, [460](#)
- f5c1, [460](#)
- f5c2, [461](#)
- f5c3, [461](#)
- f5c4, [461](#)
- f5c5, [461](#)
- f5c6, [461](#)
- f5c7, [462](#)
- f5c8, [462](#)
- f6c1, [462](#)
- f6c2, [462](#)
- f6c3, [462](#)
- f6c4, [463](#)
- f6c5, [463](#)
- f6c6, [463](#)
- f6c7, [463](#)
- f6c8, [463](#)
- f7c1, [464](#)
- f7c2, [464](#)
- f7c3, [464](#)
- f7c4, [464](#)
- f7c5, [464](#)
- f7c6, [465](#)
- f7c7, [465](#)
- f7c8, [465](#)
- f8c1, [465](#)
- f8c2, [465](#)
- f8c3, [466](#)
- f8c4, [466](#)
- f8c5, [466](#)
- f8c6, [466](#)
- f8c7, [466](#)
- f8c8, [467](#)
- getCircle, [453](#)
- getClickedPos, [453](#)
- initialize, [451](#)
- ModifyInitialBoardView, [450](#)
- placeBlackPieces, [467](#)
- placeWhitePieces, [467](#)
- quitPieces, [468](#)
- render, [452](#)
- save, [451](#), [467](#)
- saveButton, [467](#)
- saveInitialBoardResult, [468](#)
- transform, [451](#)
- view.PlayView, [508](#)
  - bots, [510](#), [514](#)
  - config, [510](#), [514](#)
  - configuration, [516](#)
  - createdAt, [516](#)
  - creator, [516](#)
  - currentUserName, [518](#)
  - gameChooser, [518](#)
  - games, [511](#), [515](#)
  - goToGame, [513](#), [517](#)
  - goToGameButton, [517](#)
  - info, [516](#)
  - initialize, [509](#)
  - logOut, [513](#), [518](#)
  - onChangeGameChooser, [511](#)
  - play, [515](#)
  - player1, [515](#)
  - player2, [515](#)
  - playGame, [517](#)
  - playGameButton, [517](#)
  - playResult, [517](#)
  - PlayView, [509](#)
  - ranking, [512](#), [515](#)
  - state, [516](#)
  - user, [510](#), [514](#)
- view.RankingConsultView, [526](#)
  - bots, [528](#), [532](#)
  - config, [529](#), [532](#)
  - consultRankings, [530](#), [533](#)
  - consultRankingsButton, [533](#)
  - consultRankingsConfirm, [533](#)
  - consultRankingsConfirmButton, [533](#)
  - consultRecords, [531](#), [533](#)
  - consultRecordsButton, [534](#)
  - currentUserName, [534](#)
  - games, [529](#), [532](#)
  - initialize, [528](#)
  - logOut, [531](#), [534](#)
  - onChangeRankingChooser, [530](#)
  - play, [529](#), [532](#)
  - playerColumn, [535](#)
  - ranking, [532](#)
  - rankingChooser, [534](#)
  - RankingConsultView, [528](#)
  - rankingInfo, [534](#)
  - table, [535](#)
  - user, [528](#), [532](#)
  - valueColumn, [535](#)
- view.RankingView, [551](#)
  - bots, [553](#), [556](#)
  - config, [553](#), [556](#)
  - consultRanking, [557](#)
  - consultRankingButton, [557](#)
  - consultRankings, [554](#)
  - consultRecord, [557](#)
  - consultRecordButton, [557](#)

- consultRecords, 555
- currentUserName, 557
- games, 554, 556
- initialize, 552
- logOut, 555, 558
- play, 554, 556
- ranking, 556
- RankingView, 552
- user, 553, 556
- view.RecordConsultView, 558
  - bots, 560, 563
  - config, 561, 563
  - consultRanking, 564
  - consultRankingButton, 564
  - consultRankings, 562
  - consultRecord, 565
  - consultRecordButton, 565
  - consultRecords, 562
  - currentUserName, 565
  - games, 561, 564
  - initialize, 560
  - logOut, 562
  - play, 561, 564
  - ranking, 564
  - rankingColumn, 565
  - RecordConsultView, 559
  - table, 565
  - user, 560, 563
  - valueColumn, 566
- view.SignUpView, 573
  - logIn, 574, 575
  - npassword, 575
  - nusername, 575
  - rpassword, 576
  - signUp, 574, 575
  - signUp2, 576
  - signUpButton, 576
  - signUpResult, 576
  - SignUpView, 574
- view.UserDeleteView, 591
  - bots, 593, 597
  - config, 593, 597
  - currentUserName, 599
  - deleteUser, 595, 598
  - deleteUserButton, 598
  - deleteUserConfirm, 595, 599
  - deleteUserConfirmButton, 599
  - deleteUserResult, 599
  - games, 594, 597
  - initialize, 593
  - logOut, 596, 599
  - modifyUser, 595, 598
  - modifyUserButton, 598
  - password, 598
  - play, 594, 597
  - ranking, 594, 597
  - user, 597
  - UserDeleteView, 592
- view.UserModifyView, 605
  - bots, 607, 611
  - config, 607, 611
  - currentUserName, 614
  - deleteUser, 609, 612
  - deleteUserButton, 613
  - games, 608, 611
  - initialize, 607
  - logOut, 610, 614
  - modifyUser, 609, 612
  - modifyUserButton, 612
  - modifyUserConfirm, 609, 614
  - modifyUserConfirmButton, 614
  - modifyUserResult, 613
  - npassword, 613
  - nusername, 613
  - play, 608, 612
  - ranking, 608, 612
  - rpassword, 613
  - user, 611
  - UserModifyView, 606
- view.UserView, 615
  - bots, 616, 620
  - config, 617, 620
  - currentUserName, 621
  - deleteUser, 618, 621
  - deleteUserButton, 621
  - games, 617, 620
  - initialize, 616
  - logOut, 619, 622
  - modifyUser, 618, 621
  - modifyUserButton, 621
  - play, 618, 620
  - ranking, 617, 620
  - user, 619
  - UserView, 616
- view.ViewCtrl, 622
  - changeScene, 623
  - domainCtrl, 625
  - main, 624
  - newWindow, 624
  - stage, 625
  - start, 623
  - ViewCtrl, 623
- viewBoard
  - domain.DomainCtrl, 252
- viewConfiguration
  - domain.DomainCtrl, 240
- ViewCtrl
  - view.ViewCtrl, 623
- ViewCtrl.java, 665
- viewGame
  - domain.DomainCtrl, 245
- viewPlayers
  - domain.DomainCtrl, 235
- viewUser
  - domain.DomainCtrl, 234
- whereInsert

domain.Ranking, [523](#)  
winIcon  
    view.GameBoardView, [334](#)  
winnerID  
    domain.Game, [306](#)