Table Of Content

Package controller

Class Summary

CloseListener

Shows an "Are you sure?"

DifficultyAction

Takes care of showing a New Game-dialog to the user and start a new game based on their difficultychoice.

DifficultySelectionAction

Starts a new game based on the user's selection

HelpAction

Takes care of finding and showing help to the user.

NumberAction

Changes the number on the board based on a user's selection.

SudokuApplet

The main initiation point for the Applet-version of our game.

SudokuGame

The main initiation point for the Application-version of our game.

---

controller

Class CloseListener

```
java.lang.Object
    |
    +--controller.CloseListener
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.awt.event.WindowListener

< Constructors > < Methods >

---

public class **CloseListener**
extends java.lang.Object
implements java.awt.event.ActionListener, java.awt.event.WindowListener

Shows an "Are you sure?" dialog before exiting.

Constructors

CloseListener

```
public  CloseListener()
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent arg0)
```

Checks whether or not the user is sure before closing the window.

---

windowActivated

```
public void windowActivated(java.awt.event.WindowEvent e)
```

Doesn't need to do anything special.

---

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent e)
```

Doesn't need to do anything special.

---

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent e)
```

Checks whether or not the user is sure before closing the window.

---

windowDeactivated

```
public void windowDeactivated(java.awt.event.WindowEvent e)
```

Doesn't need to do anything special.

---

windowDeiconified

```
public void windowDeiconified(java.awt.event.WindowEvent e)
```

Doesn't need to do anything special.

windowIconified

```
public void windowIconified(java.awt.event.WindowEvent e)
```

Doesn't need to do anything special.

windowOpened

```
public void windowOpened(java.awt.event.WindowEvent e)
```

Doesn't need to do anything special.

controller

Class DifficultyAction

```
java.lang.Object
    |
    +--javax.swing.AbstractAction
        |
        +--controller.DifficultyAction
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Cloneable, javax.swing.Action

< Constructors > < Methods >

public class **DifficultyAction**
extends javax.swing.AbstractAction

Takes care of showing a New Game-dialog to the user and start a new game based on their difficultychoice.

Constructors

DifficultyAction

```
public  DifficultyAction(MainInterface component,
                         Game game,
                         java.lang.Boolean firstScreen)
```

Creates a DifficultyAction associated with the component and based on the game. The firstScreen-flag determines if there should be an "Are you sure?"-dialog before showing the difficultyselection.

Parameters:

component - The MainInterface to associate the action with
game - The game to reset if the user wants to start a new game
firstScreen - Enables the "Are you sure"-dialog if firstScreen is false

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

---

controller

Class DifficultySelectionAction

```
java.lang.Object
    |
    +--javax.swing.AbstractAction
        |
        +--controller.DifficultySelectionAction
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Cloneable, javax.swing.Action

---

< Constructors > < Methods >

---

public class **DifficultySelectionAction**
extends javax.swing.AbstractAction

Starts a new game based on the user's selection

Constructors

DifficultySelectionAction

```
public  DifficultySelectionAction(Game game,
                                  java.awt.Component frame)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent arg0)
```

Starts a new game based on the ActionCommand sent when the user clicked one of the difficultybuttons.

---

controller

Class HelpAction

```
java.lang.Object
    |
    +--javax.swing.AbstractAction
        |
        +--controller.HelpAction
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Cloneable, javax.swing.Action

---

< Constructors > < Methods >

---

public class **HelpAction**
extends javax.swing.AbstractAction

Takes care of finding and showing help to the user.

Constructors

HelpAction

```
public  HelpAction(MainInterface frame,
                   Game game)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

If there are mistakes on the board, these gets marked. Otherwise it finds a solvable field (if any) and marks it on the board.

controller

Class NumberAction

```
java.lang.Object
    |
    +--javax.swing.AbstractAction
        |
        +--controller.NumberAction
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Cloneable, javax.swing.Action

< Constructors > < Methods >

public class **NumberAction**
extends javax.swing.AbstractAction

Changes the number on the board based on a user's selection.

Constructors

NumberAction

```
public  NumberAction(MainInterface main,
                     int fieldId,
                     java.awt.Component frame)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Shows a dialog and changes the numbervalue and the sheep's text based on the user's selection.

controller

Class SudokuApplet

```
java.lang.Object
   |
   +--java.awt.Component
         |
         +--java.awt.Container
               |
               +--java.awt.Panel
                     |
                     +--java.applet.Applet
                           |
                           +--javax.swing.JApplet
                                 |
                                 +--controller.SudokuApplet
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer

< Constructors > < Methods >

public class **SudokuApplet**
extends javax.swing.JApplet

The main initiation point for the Applet-version of our game.

Constructors

SudokuApplet

```
public  SudokuApplet()
```

Methods

init

```
public void init()
```

Gets run when the applet gets loaded into the browser.

Overrides:

init in class java.applet.Applet

controller

Class SudokuGame

```
java.lang.Object
    |
    +--controller.SudokuGame
```

< Constructors > < Methods >

---

public class **SudokuGame**
extends java.lang.Object

The main initiation point for the Application-version of our game.

Constructors

SudokuGame

```
public  SudokuGame()
```

Methods

main

```
public static void main(java.lang.String[] args)
```

Package model

Interface Summary

GameSettings

Interface to retrieve GameSettings

Class Summary

Board

Board.java is able to generate sudoku boards, get values from fields and set values in fields.

EasySettings

The easy gamesettings for a 9x9 Sudoku.

Game

Contains the gameboard and the solution.

General9x9Settings

Contains the general settings for a 9x9 sudoku

Generator

The sudoku generator.

HardSettings

The hard gamesettings for a 9x9 Sudoku.

Helper

Helper.java is able to find a field thats solveable, find fields that are incorrectly filled in and how many mistakes present on the board.

NormalSettings

The normal gamesettings for a 9x9 Sudoku.

Solver

Solver class used to solve a Sudoku-puzzle.

Statistics

The statistics of the current game.

SudokuMath

Performs various mathematical operations on SudokuBoards.

model

Class Board

```
java.lang.Object
    |
    +--java.util.Observable
        |
        +--model.Board
```

< Constructors > < Methods >

public class **Board**
extends java.util.Observable

Board.java is able to generate sudoku boards, get values from fields and set values in fields. It can compare two boards to check if the sudoku is solved.

Constructors

Board

```
public  Board()
```

Creates a board with all fields filled in.

Board

```
public  Board(int[] boardArray)
```

Board

```
public  Board(int[] boardArray,
              GameSettings settings)
```

Board

```
public  Board(Board board)
```

Board

```
public  Board(GameSettings settings)
```

Methods

getSettings

```
public GameSettings getSettings()
```

Returns:

the settings

---

getValue

```
public int getValue(int fieldId)
             throws java.lang.IllegalArgumentException
```

Gets the value from board[] at position a

Parameters:

fieldId -

Returns:

The value at the fieldId-position.

Throws:

java.lang.IllegalArgumentException -

---

isEqualTo

```
public boolean isEqualTo(Board compareBoard)
```

Compares current board to supplied board to determine if the sudoku is correctly solved.

Parameters:

compareBoard - The board to compare to.

Returns:

True or false depending on whether or not the boards are equal.

---

print

```
public void print()
```

reset

```
public void reset(GameSettings settings,
                  Game game)
```

setValue

```
public void setValue(int fieldId,
                     int value)
            throws java.lang.IllegalArgumentException
```

Sets a value b into board[] at position a

Parameters:

fieldId - The fieldnumber whose value is to be set
value - The value to set

Throws:

java.lang.IllegalArgumentException -

shuffle

```
public void shuffle()
```

Shuffles rows, columns, quadrantcolums and quadrantrows of the board.

switchColumns

```
public void switchColumns(int first,
                          int second)
```

Switches the two supplied columns in the board.

Parameters:

first - The first column to be switched.
second - The second column to be switched.

switchQuadrantColumns

```
public void switchQuadrantColumns(int first,
                                  int second)
```

Switches the two supplied quadrantcolumns in the board.

Parameters:

first - The first quadrantcolumn to be switched.
second - The second quadrantcolumn to be switched.

---

switchQuadrantRows

```
public void switchQuadrantRows(int first,
                               int second)
```

Switches the two supplied quadrantrows in the board.

Parameters:

first - The first quadrantrow to be switched.
second - The second quadrantrow to be switched.

---

switchRows

```
public void switchRows(int first,
                       int second)
```

Switches the two supplied rows in the board.

Parameters:

first - The first row to be switched.
second - The second row to be switched.

---

toArray

```
public int[] toArray()
```

model

Class EasySettings

```
java.lang.Object
  |
  +--General9x9Settings
        |
        +--model.EasySettings
```

All Implemented Interfaces:

GameSettings

< Fields > < Constructors > < Methods >

public class **EasySettings**
extends General9x9Settings
implements GameSettings

The easy gamesettings for a 9x9 Sudoku.

Fields

IDENTIFIER

```
public static final int IDENTIFIER
```

Constructors

EasySettings

```
public  EasySettings()
```

Methods

getDifficulty

```
public java.lang.String getDifficulty()
```

returns the difficulty

Returns:

The difficulty

getNumbersToRemove

```
public int getNumbersToRemove()
```

returns the numbers to remove

Returns:

The numbers to remove

Overrides:

getNumbersToRemove in class General9x9Settings

---

model

Class Game

```
java.lang.Object
   |
   +--model.Game
```

< Constructors > < Methods >

---

public class **Game**
extends java.lang.Object

Contains the gameboard and the solution. When a new game is created Game will create a new board.

Constructors

Game

```
public  Game()
```

Stores the solved board in solutionBoard. Makes the generator remove numbers until the desired difficulty is reached and then stores the new board in currentBoard

Methods

getCurrentBoard

```
public Board getCurrentBoard()
```

returns the current board

Returns:

The current board

---

getSolutionBoard

`public Board getSolutionBoard()`

returns the solved board

Returns:

The solved board

---

getStatistics

`public Statistics getStatistics()`

returns the statistics

Returns:

The statistics

---

reset

`public void reset(GameSettings settings)`

creates a new statistics element based on the settings.

Parameters:

settings -

---

setSolutionBoard

`public void setSolutionBoard(Board solutionBoard)`

stores the solved board

Parameters:

solutionBoard -

---

model

Interface GameSettings

< Methods >

---

public interface **GameSettings**

Interface to retrieve GameSettings

Methods

getBoardDimensions

```
public int getBoardDimensions()
```

getBoardLength

```
public int getBoardLength()
```

getDifficulty

```
public java.lang.String getDifficulty()
```

getNumbersToRemove

```
public int getNumbersToRemove()
```

getQuadrantDimensions

```
public int getQuadrantDimensions()
```

getStdBoardArray

```
public int[] getStdBoardArray()
```

getValidValues

```
public int[] getValidValues()
```

model

Class General9x9Settings

```
java.lang.Object
   |
   +--model.General9x9Settings
```

All Implemented Interfaces:
GameSettings

Direct Known Subclasses:
EasySettings, HardSettings, NormalSettings

< Constructors > < Methods >

---

public abstract class **General9x9Settings**
extends java.lang.Object
implements GameSettings

Contains the general settings for a 9x9 sudoku

Constructors

General9x9Settings

`public  General9x9Settings()`

Methods

getBoardDimensions

`public int getBoardDimensions()`

returns the board dimension

Returns:
The board dimension

---

getBoardLength

`public int getBoardLength()`

returns the board length

Returns:
The board length

---

getNumbersToRemove

```
public abstract int getNumbersToRemove()
```

returns the number of fields to remove

Returns:

The number of fields to remove

---

getQuadrantDimensions

```
public int getQuadrantDimensions()
```

returns the quadrant dimension

Returns:

The quadrant dimension

---

getStdBoardArray

```
public int[] getStdBoardArray()
```

returns the standard board array for the current settings

Returns:

The standard board array for the current settings

---

getValidValues

```
public int[] getValidValues()
```

returns the valid values

Returns:

The valid values

---

model

Class Generator

```
java.lang.Object
  |
  +--model.Generator
```

< Constructors > < Methods >

---

public abstract class **Generator**
extends java.lang.Object

The sudoku generator.

Constructors

Generator

```
public  Generator()
```

Methods

generate

```
public static void generate(Board board)
```

---

generate

```
public static void generate(Board board,
                            GameSettings settings)
```

Generates a playable gameboard removing one field at a time untill the desired difficulty is achieved.

Parameters:

board - The board to remove fields from.
settings - The board settings (board size)

---

model

Class HardSettings

```
java.lang.Object
    |
    +--General9x9Settings
        |
        +--model.HardSettings
```

All Implemented Interfaces:
GameSettings

< Fields > < Constructors > < Methods >

---

public class **HardSettings**
extends General9x9Settings

implements GameSettings

The hard gamesettings for a 9x9 Sudoku.

Fields

IDENTIFIER

`public static final int` **`IDENTIFIER`**

Constructors

HardSettings

`public` **`HardSettings`**`()`

Methods

getDifficulty

`public java.lang.String` **`getDifficulty`**`()`

returns the difficulty

Returns:

The difficulty

---

getNumbersToRemove

`public int` **`getNumbersToRemove`**`()`

returns the numbers to remove

Returns:

The numbers to remove

Overrides:

getNumbersToRemove in class General9x9Settings

model

Class Helper

```
java.lang.Object
    |
    +--model.Helper
```

< Constructors > < Methods >

public abstract class **Helper**
extends java.lang.Object

Helper.java is able to find a field thats solveable, find fields that are incorrectly filled in and how many mistakes present on the board.

Constructors

Helper

```
public  Helper()
```

Methods

findSolveable

```
public static int findSolveable(Board board,
                                GameSettings settings)
                 throws java.lang.NoSuchFieldException
```

Finds a field on the current board which is solveable and returns that field id.

Parameters:

board - The board from which the help is needed.
settings - The game settings from which the board is created under.

Returns:

A fieldId that is solveable and suggested to the player.

Throws:

java.lang.NoSuchFieldException -

findSolveable

```
public static int findSolveable(Game game)
                         throws java.lang.NoSuchFieldException
```

Finds a field on the current board which is solveable and returns that fieldId. Calls findSolveable(game.getCurrentBoard(), game.getCurrentBoard().getSettings());

Parameters:

game - The current game.

Returns:

A fieldId that is solveable and suggested to the player.

Throws:

java.lang.NoSuchFieldException -

---

getAmountOfMistakes

```
public static int getAmountOfMistakes(Game game)
```

This function scans the board for mistakes and returns how many mistakes there currently is.

Parameters:

game - The current game that's being played.

Returns:

The number of mistakes currently on the board.

---

getFieldsWithMistakes

```
public static int[] getFieldsWithMistakes(Game game)
```

Gets an int-array with the fieldIds containing mistakes. Calls getFieldsWithMistakes (game, getAmountOfMistakes(game));

Parameters:

game - The current game.

Returns:

An int[] containing the fieldIds with mistakes.

---

getFieldsWithMistakes

```
public static int[] getFieldsWithMistakes(Game game,
                                          int amountOfMistakes)
```

Gets an int-array with the fieldIds containing mistakes.

Parameters:

game - The current game.
amountOfMistakes - Used to the size of the result array.

Returns:

An int[] containing the fieldIds with mistakes.

---

model

Class NormalSettings

```
java.lang.Object
    |
    +--General9x9Settings
        |
        +--model.NormalSettings
```

All Implemented Interfaces:
GameSettings

< Fields > < Constructors > < Methods >

---

public class **NormalSettings**
extends General9x9Settings
implements GameSettings

The normal gamesettings for a 9x9 Sudoku.

Fields

IDENTIFIER

```
public static final int IDENTIFIER
```

Constructors

NormalSettings

```
public  NormalSettings()
```

Methods

getDifficulty

```
public java.lang.String getDifficulty()
```

returns the difficulty

Returns:

The difficulty

---

getNumbersToRemove

```
public int getNumbersToRemove()
```

returns the numbers to remove

Returns:

The numbers to remove

Overrides:

getNumbersToRemove in class General9x9Settings

---

model

Class Solver

```
java.lang.Object
    |
    +--model.Solver
```

< Constructors > < Methods >

---

public abstract class **Solver**
extends java.lang.Object

Solver class used to solve a Sudoku-puzzle.

Constructors

Solver

```
public  Solver()
```

Methods

solveField

```
public static int solveField(int fieldNum,
                             Board board)
```

In calling the solver, first solverLevelOne is used, then if this is unable to produce a unique result, call solverLevelTwo.

Parameters:

fieldNum - the integer value the specific field has on the board
board - the sudoku board used

Returns:

the result of trying to solve the field, 0 if not solveable.

---

model

Class Statistics

```
java.lang.Object
   |
   +--model.Statistics
```

< Constructors > < Methods >

---

public class **Statistics**
extends java.lang.Object

The statistics of the current game. Keeps track of the amount of Hints and Mistakes made.

Constructors

Statistics

```
public  Statistics()
```

Creates a new statistics type for the game. The amount of hints used and mistakes made from start is, naturally, 0.

Methods

getElapsedTime

```
public java.lang.String getElapsedTime()
```

returns the time elapsed durring play

Returns:

The time elapsed durring play

---

getHints

```
public int getHints()
```

Returns the amount of hints used.

Returns:

The amount of hints.

---

getMistakes

```
public int getMistakes()
```

Returns the amount of mistakes used.

Returns:

The amount of mistakes

---

increaseHints

```
public void increaseHints()
```

Increases the amount of hints used by 1.

---

increaseMistakesBy

```
public void increaseMistakesBy(int amount)
```

Increases the amount of mistakes used by 1.

---

setStopTime

```
public void setStopTime()
```

Sets the time at which the Sudoku was solved.

model

Class SudokuMath

```
java.lang.Object
    |
    +--model.SudokuMath
```

< Constructors > < Methods >

public abstract class **SudokuMath**
extends java.lang.Object

Performs various mathematical operations on SudokuBoards. Everything is 0-index'ed, and works on boards of all sizes. Fx. a row from a 3x3x9 Sudoku has the numbers 0 to 8.

Constructors

SudokuMath

```
public  SudokuMath()
```

Methods

getColumnFromPos

```
public static int[] getColumnFromPos(int position,
                                     Board board)
```

Gets the contents of the column based on the position in the board. This is done by first calculating the columnNumber, adding the value this position contains to an array, and then continuously adding boardDim to the columnNumber, adding that value to the array until the array has boardDimension values in it.

Parameters:

position - The position to get the column from.
board - The board to get the column from.

Returns:

An int-array containing the column.

getColumnNumber

```
public static int getColumnNumber(int position,
                                  GameSettings settings)
```

Converts a position to a columnnumber, by calculating (position % boardDimension).

Parameters:

position - The position on the board.

Returns:

The number of the column.

---

getQuadrantFromPos

```
public static int[] getQuadrantFromPos(int position,
                                       Board board)
```

Gets the contents of the quadrant based on the position in the board. It calculates the fieldId of the top left corner of the quadrant, and then adds the fieldIds based on the size of the board.

Parameters:

position - The position to get the quadrant from.
board - The board to get the cquadrant from.

Returns:

An int-array containing the quadrant.

---

getQuadrantNumber

```
public static int getQuadrantNumber(int position,
                                    GameSettings settings)
```

Converts a position to a quadrantnumber, by calculating ((rowNumber / quadrantDim) * quadrantDim + columnNumber / quadrantDim).

Parameters:

position - The position on the board.

Returns:

The number of the quadrant.

---

getRowFromPos

```
public static int[] getRowFromPos(int position,
                                  Board board)
```

Gets the contents of the row based on the position in the board. This is done by substracting the columnNumber from the position value, and adding values to an int-array until it has reached the length of the boardDimension.

Parameters:

position - The position to get the row from.
board - The board to get the row from.

Returns:

An int-array containing the row.

---

getRowNumber

```
public static int getRowNumber(int position,
                               GameSettings settings)
```

Converts a position to a rownumber, by calculating (position / boardDimension).

Parameters:

position - The position on the board.

Returns:

The number of the row.

Package tests

Class Summary

TestFindSolveable

The JUnit TestCase Class for Helper.findSolveable().

TestSudokuMathGetFromPos

The JUnit TestCase Class for the getXFromPos methods used in SudokuMath.

TestSudokuMathGetNumber

The JUnit TestCase Class for the getXNumber methods used in SudokuMath.

tests

Class TestFindSolveable

```
java.lang.Object
   |
   +--junit.framework.Assert
         |
         +--junit.framework.TestCase
               |
               +--tests.TestFindSolveable
```

All Implemented Interfaces:

junit.framework.Test

< Constructors > < Methods >

public class **TestFindSolveable**
extends junit.framework.TestCase

The JUnit TestCase Class for Helper.findSolveable().

Constructors

TestFindSolveable

```
public  TestFindSolveable()
```

Methods

testFindSolveable01

`public void testFindSolveable01()`

First test of findSolveable(): Tests that the Helper returns an NoSuchFieldException when no solveable fieldId can be found.

---

testFindSolveable02

`public void testFindSolveable02()`

Second test of findSolveable(): This test looks at the opposite of testFindSolveable01, namely when there is a field that can be solved, and that the correct fieldId is returned, plus no Exception is thrown.

---

tests

Class TestSudokuMathGetFromPos

```
java.lang.Object
    |
    +--junit.framework.Assert
        |
        +--junit.framework.TestCase
            |
            +--tests.TestSudokuMathGetFromPos
```

All Implemented Interfaces:
junit.framework.Test

< Constructors > < Methods >

public class **TestSudokuMathGetFromPos**
extends junit.framework.TestCase

The JUnit TestCase Class for the getXFromPos methods used in SudokuMath.

Constructors

TestSudokuMathGetFromPos

`public  TestSudokuMathGetFromPos()`

Methods

testGetColumnFromPos

```
public void testGetColumnFromPos()
```

This tests the second of the getFromPos-methods: getColumnFromPos().

---

testGetQuadrantFromPos

```
public void testGetQuadrantFromPos()
```

This tests the third of the getFromPos-methods: getQuadrantFromPos().

---

testGetRowFromPos

```
public void testGetRowFromPos()
```

This tests the first of the getFromPos-methods: getRowFromPos().

---

tests

Class TestSudokuMathGetNumber

```
java.lang.Object
    |
    +--junit.framework.Assert
        |
        +--junit.framework.TestCase
            |
            +--tests.TestSudokuMathGetNumber
```

All Implemented Interfaces:
junit.framework.Test

< Constructors > < Methods >

---

public class **TestSudokuMathGetNumber**
extends junit.framework.TestCase

The JUnit TestCase Class for the getXNumber methods used in SudokuMath.

Constructors

TestSudokuMathGetNumber

```
public  TestSudokuMathGetNumber()
```

Methods

testGetColumnNumber

`public void testGetColumnNumber()`

This tests the second of the get-methods: getColumnNumber().

---

testGetQuadrantNumber

`public void testGetQuadrantNumber()`

This tests the third of the get-methods: getQuadrantNumber().

---

testGetRowNumber

`public void testGetRowNumber()`

This tests the first of the get-methods: getRowNumber().

Package view

Interface Summary

MainInterface

Interface for the two different kinds of windows our program containts.

Class Summary

Background

Handles the drawing of the backgroundimage.

Board

The graphical representation of our Sudokuboard.

CongratulationScreen

A congratulationscreen used to congratulate the user when they have completed a Sudokupuzzle.

DifficultySelection

A screen used to let the user select between the different difficulties.

Header

The headerimage (the "title").

IngameControls

The ingamecontrols (Help- and New Game-button)

MainApplet

Our program in Applet-form.

MainWindow

NumberDialog

PlaceCenter

Class containg a function to center a component.

SheepSpeak

The SheepSpeak, which is our main method to communicate with the users.

Statistics

SudokuButton

Creates buttons based on images.

SudokuMenu

A menubar containing the proper menuitems.

ViewSettings

Abstract class containing the various settings used in the GUI.

---

view

Class Background

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--javax.swing.JComponent
                |
                +--javax.swing.JPanel
                    |
                    +--view.Background
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

---

< Constructors > < Methods >

---

public class **Background**
extends javax.swing.JPanel

Handles the drawing of the backgroundimage.

Constructors

Background

```
public  Background(java.lang.String imageFile)
```

Creates a new backgroundimage from the supplied image.

Parameters:

imageFile - The image to load

Background

```
public  Background(java.lang.String imageFile,
                   java.awt.Dimension dim)
```

Creates a new backgroundimage from the supplied image with the specified dimension.

Parameters:

imageFile - The image to load
dim - The dimension of the image

Methods

paint

```
public void paint(java.awt.Graphics g)
```

Overrides the extended JPanel's paint-method so the image actually gets drawed.

Overrides:

paint in class javax.swing.JComponent

---

view

Class Board

```
java.lang.Object
   |
   +--java.awt.Component
         |
         +--java.awt.Container
               |
               +--javax.swing.JComponent
                     |
                     +--javax.swing.JPanel
                           |
                           +--view.Board
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, java.util.Observer, javax.accessibility.Accessible

---

< Constructors > < Methods >

---

public class **Board**
extends javax.swing.JPanel
implements java.util.Observer

The graphical representation of our Sudokuboard.

Constructors

Board

```
public  Board(MainInterface main)
```

Creates a new Board where the dimensions are extracted from the MainInterface-object.

Parameters:

main - The object containing the board to base the view on.

Board

```
public  Board(MainInterface main,
              java.awt.Dimension dimension)
```

Creates a new Board with a specified dimension.

Parameters:

main - The object containing the board to base the view on.
dimension - The dimension the board should get.

Methods

clearHintNotices

```
public void clearHintNotices()
```

Removes all hintnotices from the board.

---

clearNotice

```
public void clearNotice(int fieldId)
```

Removes a single notice from the board.

Parameters:

fieldId - The Id of the field whose backgroundcolor should be reset.

---

clearNotices

```
public void clearNotices()
```

Removes all colornotices from the board.

---

getViewBoardDimensions

```
public java.awt.Dimension getViewBoardDimensions()
```

Calculates the visual width of the current gameboard.

Returns:

The width of the board.

---

getViewBoardDimensions

```
public static java.awt.Dimension getViewBoardDimensions(Board board)
```

Calculates the visual width of the supplied gameboard.

Parameters:

board - The board to calculate the width of

Returns:

The width of the board.

---

setNotice

```
public void setNotice(int fieldId,
                      java.awt.Color color)
```

Sets a single notice on the board.

Parameters:

fieldId - The Id of the field whose backgroundcolor should be set.
color - The color the field should get.

---

setNotices

```
public void setNotices(int[] fieldIds,
                       java.awt.Color color)
```

Set notices on multiple fields.

Parameters:

fieldIds - An int-array containing the fieldIds of all the field who should have their backgroundcolor set.
color - The color to set.

---

setValue

```
public void setValue(int fieldId,
                     int value)
```

Change the value of a button / field.

Parameters:

fieldId - The fieldId of the button / field whose value should be changed.
value - The value to change to.

---

update

```
public void update(java.util.Observable arg0,
                   java.lang.Object arg1)
```

When a new game gets created, this updates the board with the new values

view

Class CongratulationScreen

```
java.lang.Object
    |
    +--view.CongratulationScreen
```

< Constructors > < Methods >

public class **CongratulationScreen**
extends java.lang.Object

A congratulationscreen used to congratulate the user when they have completed a Sudokupuzzle. Also shows the statistics.

Constructors

CongratulationScreen

```
public  CongratulationScreen()
```

Methods

show

```
public void show(MainInterface frame,
                 Game game)
```

Shows a congratulationscreen

Parameters:

frame - The MainInterface to show the congratulationscreen on
game - The game to get the statistics from

view

Class DifficultySelection

```
java.lang.Object
    |
    +--view.DifficultySelection
```

< Constructors > < Methods >

public class **DifficultySelection**
extends java.lang.Object

A screen used to let the user select between the different difficulties.

Constructors

DifficultySelection

```
public  DifficultySelection(Game game)
```

Methods

show

```
public void show(MainInterface frame)
```

Shows the screen.

Parameters:

frame - The frame to show the screen on.

view

Class Header

```
java.lang.Object
   |
   +--java.awt.Component
         |
         +--java.awt.Container
               |
               +--javax.swing.JComponent
                     |
                     +--javax.swing.JPanel
                           |
                           +--view.Header
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

< Constructors > < Methods >

public class **Header**
extends javax.swing.JPanel

The headerimage (the "title").

Constructors

Header

```
public  Header()
```

Creates the header based on the headerimage.

Methods

paint

```
public void paint(java.awt.Graphics g)
```

Overrides the paint-method to make sure that the image actually gets painted.

Overrides:

paint in class javax.swing.JComponent

view

Class IngameControls

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--javax.swing.JComponent
                |
                +--javax.swing.JPanel
                    |
                    +--view.IngameControls
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

---

< Constructors > < Methods >

---

public class **IngameControls**
extends javax.swing.JPanel

The ingamecontrols (Help- and New Game-button)

Constructors

IngameControls

```
public  IngameControls(java.awt.Component frame,
                       DifficultyAction difficultyAction,
                       HelpAction helpAction)
```

Creates the controls.

Parameters:

frame - The frame the screen should be added to.
difficultyAction - The DifficultyAction showing the New Game-screen.
helpAction - The HelpAction handling requests for help.

Methods

getDifficultyAction

```
public DifficultyAction getDifficultyAction()
```

Gets the DifficultyAction associated with the controls.

Returns:

The DifficultyAction

getHelpAction

```
public HelpAction getHelpAction()
```

Gets the HelpAction associated with the controls.

Returns:

The HelpAction

view

Class MainApplet

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--javax.swing.JApplet
                                |
                                +--view.MainApplet
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, MainInterface

< Constructors > < Methods >

public class **MainApplet**
extends javax.swing.JApplet
implements MainInterface

Our program in Applet-form.

Constructors

MainApplet

```
public  MainApplet()
```

Methods

add

```
public java.awt.Component add(java.awt.Component component,
                              int zindex,
                              int x,
                              int y)
```

Adds a component to the window.

Parameters:

component - The component to add.
zindex - The Z-Index of the position.
x - The X-coordinate.
y - The Y-coordinate.

Returns:

The added component.

---

createBackgroundPanel

```
public void createBackgroundPanel(java.lang.String backgroundImage)
```

Creates a new background with the supplied image.

Parameters:

backgroundImage - The image to use as a background.

---

createBoard

```
public void createBoard()
```

Creates a new board. setGame() must have been called before.

---

createHeader

```
public void createHeader()
```

---

createIngameControls

```
public void createIngameControls(DifficultyAction difficultyAction,
                                 HelpAction helpAction)
```

Creates the ingamecontrols.

Parameters:

difficultyAction - The action to perform when the "New game" button is pressed.
helpAction - The action to perform when the "Help" button is pressed.

createSheepSpeak

`public void createSheepSpeak()`

Create the SheepSpeak-object.

getBackgroundPanel

`public Background getBackgroundPanel()`

Gets the background.

Returns:

The background.

getBoard

`public Board getBoard()`

Gets the graphical representation of the board.

Returns:

The board.

getBoardDimension

`public java.awt.Dimension getBoardDimension()`

Gets the current dimensions of the board.

Returns:

The dimensions of the board.

getControls

`public IngameControls getControls()`

getGame

```
public Game getGame()
```

Gets the current instance of the game used.

Returns:

The game.

---

getSheepSpeak

```
public SheepSpeak getSheepSpeak()
```

Gets the current SheepSpeak - the box in which the wise words of the sheep are.

Returns:

The SheepSpeak-object.

---

hideElements

```
public void hideElements()
```

---

setGame

```
public void setGame(Game game)
```

Sets the current game instance. Must be called before createBoard().

Parameters:

game - The game to set.

---

setMenu

```
public void setMenu()
```

---

setup

```
public void setup()
```

Performs some standard operations on the window.

---

showElements

```
public void showElements()
```

---

view

Interface MainInterface

< Methods >

public interface **MainInterface**

Interface for the two different kinds of windows our program containts.

Methods

add

```
public java.awt.Component add(java.awt.Component component,
                              int zindex,
                              int x,
                              int y)
```

Adds a component to the window.

Parameters:

component - The component to add.
zindex - The Z-Index of the position.
x - The X-coordinate.
y - The Y-coordinate.

Returns:

The added component.

---

createBackgroundPanel

```
public void createBackgroundPanel(java.lang.String backgroundImage)
```

Creates a new background with the supplied image.

Parameters:

backgroundImage - The image to use as a background.

---

createBoard

```
public void createBoard()
```

Creates a new board. setGame() must have been called before.

---

createHeader

```
public void createHeader()
```

Creates and adds the header to the frame.

---

createIngameControls

```
public void createIngameControls(DifficultyAction difficultyAction,
                                 HelpAction helpAction)
```

Creates the ingamecontrols.

Parameters:

difficultyAction - The action to perform when the "New game" button is pressed.
helpAction - The action to perform when the "Help" button is pressed.

---

createSheepSpeak

```
public void createSheepSpeak()
```

Creates and adds the SheepSpeak-object.

---

getBackgroundPanel

```
public Background getBackgroundPanel()
```

Gets the background contained in the frame.

Returns:

The background.

---

getBoard

`public Board getBoard()`

Gets the graphical representation of the board.

Returns:

The board.

---

getBoardDimension

`public java.awt.Dimension getBoardDimension()`

Gets the current dimensions of the board.

Returns:

The dimensions of the board.

---

getControls

`public IngameControls getControls()`

Gets the IngameControls contained in the frame.

Returns:

The IngameControls

---

getGame

`public Game getGame()`

Gets the current instance of the game used.

Returns:

The game.

---

getSheepSpeak

`public SheepSpeak getSheepSpeak()`

Gets the current SheepSpeak - the box in which the wise words of the sheep are.

Returns:

The SheepSpeak-object.

---

hideElements

```
public void hideElements()
```

Hides all interfaceelements.

---

setGame

```
public void setGame(Game game)
```

Sets the current game instance. Must be called before createBoard().

Parameters:

game - The game to set.

---

setGlassPane

```
public void setGlassPane(java.awt.Component glassPane)
```

Set the glasspane of the frame to the specified glasspane.

Parameters:

glassPane - The glassPane to set as glasspane.

---

setMenu

```
public void setMenu()
```

Creates and adds the menu to the frame.

---

setup

```
public void setup()
```

Performs some standard operations on the window.

---

showElements

```
public void showElements()
```

Shows all interfaceelements.

---

view

Class MainWindow

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--java.awt.Window
                |
                +--java.awt.Frame
                    |
                    +--javax.swing.JFrame
                        |
                        +--view.MainWindow
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, MainInterface

< Constructors > < Methods >

public class **MainWindow**
extends javax.swing.JFrame
implements MainInterface

Author:

Julian

Constructors

MainWindow

```
public  MainWindow()
```

Methods

add

```
public java.awt.Component add(java.awt.Component component,
                              int zindex,
                              int x,
                              int y)
```

Adds a component to the window.

Parameters:

component - The component to add.
zindex - The Z-Index of the position.
x - The X-coordinate.
y - The Y-coordinate.

Returns:

The added component.

---

createBackgroundPanel

```
public void createBackgroundPanel(java.lang.String backgroundImage)
```

Creates a new background with the supplied image.

Parameters:

backgroundImage - The image to use as a background.

---

createBoard

```
public void createBoard()
```

Creates a new board. setGame() must have been called before.

---

createHeader

```
public void createHeader()
```

---

createIngameControls

```
public void createIngameControls(DifficultyAction difficultyAction,
                                 HelpAction helpAction)
```

Creates the ingamecontrols.

Parameters:

difficultyAction - The action to perform when the "New game" button is pressed.
helpAction - The action to perform when the "Help" button is pressed.

createSheepSpeak

```
public void createSheepSpeak()
```

Create the SheepSpeak-object.

getBackgroundPanel

```
public Background getBackgroundPanel()
```

Gets the background.

Returns:

The background.

getBoard

```
public Board getBoard()
```

Gets the graphical representation of the board.

Returns:

The board.

getBoardDimension

```
public java.awt.Dimension getBoardDimension()
```

Gets the current dimensions of the board.

Returns:

The dimensions of the board.

getControls

```
public IngameControls getControls()
```

getGame

```
public Game getGame()
```

Gets the current instance of the game used.

Returns:

The game.

---

getSheepSpeak

```
public SheepSpeak getSheepSpeak()
```

Gets the current SheepSpeak - the box in which the wise words of the sheep are.

Returns:

The SheepSpeak-object.

---

hideElements

```
public void hideElements()
```

---

setGame

```
public void setGame(Game game)
```

Sets the current game instance. Must be called before createBoard().

Parameters:

game - The game to set.

---

setMenu

```
public void setMenu()
```

---

setup

```
public void setup()
```

Performs some standard operations on the window.

---

showElements

```
public void showElements()
```

view

Class NumberDialog

```
java.lang.Object
   |
   +--java.awt.Component
         |
         +--java.awt.Container
               |
               +--java.awt.Window
                     |
                     +--java.awt.Dialog
                           |
                           +--javax.swing.JDialog
                                 |
                                 +--view.NumberDialog
```

All Implemented Interfaces:
java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

< Constructors > < Methods >

public class **NumberDialog**
extends javax.swing.JDialog

Author:
Julian

Constructors

NumberDialog

```
public  NumberDialog(MainInterface main)
```

Creates a numberdialog with using the settings of the supplied MainInterface and also using it as parent.

Parameters:
main -

Methods

getValue

```
public int getValue()
```

---

view

Class PlaceCenter

```
java.lang.Object
   |
   +--view.PlaceCenter
```

< Constructors > < Methods >

public abstract class **PlaceCenter**
extends java.lang.Object

Class containg a function to center a component.

Constructors

PlaceCenter

```
public  PlaceCenter()
```

Methods

placeCenter

```
public static void placeCenter(java.awt.Component component)
```

Places the `component` on the center of the screen.

Parameters:

component - The component to center.

view

Class SheepSpeak

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--view.SheepSpeak
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

---

< Constructors > < Methods >

---

public class **SheepSpeak**
extends javax.swing.JPanel

The SheepSpeak, which is our main method to communicate with the users.

Constructors

SheepSpeak

```
public  SheepSpeak()
```

Methods

paint

```
public void paint(java.awt.Graphics g)
```

Overrides:

paint in class javax.swing.JComponent

---

resetText

```
public void resetText()
```

---

setText

```
public void setText(java.lang.String text)
```

---

view

Class Statistics

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--view.Statistics
```

All Implemented Interfaces:
java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

---

< Constructors >

---

public class **Statistics**
extends javax.swing.JPanel

Author:
Julian

Constructors

Statistics

```
public  Statistics(Game game)
```

Creates the statisticspanel based on the supplied game.

Parameters:
game - The game to base the statistics on.

---

view

Class SudokuButton

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.AbstractButton
                          |
                          +--javax.swing.JButton
                                |
                                +--view.SudokuButton
```

All Implemented Interfaces:

java.awt.ItemSelectable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.SwingConstants

< Constructors > < Methods >

public class **SudokuButton**
extends javax.swing.JButton

Creates buttons based on images.

Constructors

SudokuButton

`public  SudokuButton(java.lang.String imageFile)`

Creates a button based on the supplied image.

Parameters:

imageFile - The image to use as a background.

Methods

paint

`public void paint(java.awt.Graphics g)`

Overrides the paint-method to make sure that the image gets drawed.

Overrides:

paint in class javax.swing.JComponent

view

Class SudokuMenu

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--javax.swing.JComponent
                |
                +--javax.swing.JMenuBar
                    |
                    +--view.SudokuMenu
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.MenuElement

---

< Constructors >

---

public class **SudokuMenu**
extends javax.swing.JMenuBar

A menubar containing the proper menuitems.

Constructors

SudokuMenu

`public  SudokuMenu(MainInterface main)`

Create the menubar and add it to the supplied MainInterface

Parameters:

main - The MainInterface which should get the menubar.

---

view

Class ViewSettings

```
java.lang.Object
    |
    +--view.ViewSettings
```

< Constructors > < Methods >

---

public abstract class **ViewSettings**
extends java.lang.Object

Abstract class containing the various settings used in the GUI.

Constructors

ViewSettings

```
public  ViewSettings()
```

Methods

getAlternateHeight

```
public static int getAlternateHeight()
```

Returns:

The alternate height used for the congratulationscreen and difficultyselection.

---

getBoardSpacing

```
public static int getBoardSpacing()
```

Returns:

The spacing between the quadrants on the board.

---

getButtonBackground

```
public static java.awt.Color getButtonBackground()
```

Returns:

The backgroundcolor of the buttons

---

getButtonDimension

```
public static java.awt.Dimension getButtonDimension()
```

Returns:

The size of the buttons as a Dimension.

---

getButtonSize

```
public static int getButtonSize()
```

Returns:

The size of the buttons on the board.

---

getHintColor

```
public static java.awt.Color getHintColor()
```

Returns:

The color used for showing hints on the board.

---

getMainHeight

```
public static int getMainHeight()
```

Returns:

The height of the mainwindow.

---

getMainWidth

```
public static int getMainWidth()
```

Returns:

The width of the mainwindow.

---

getWrongNumberColor

```
public static java.awt.Color getWrongNumberColor()
```

Returns:

The color used for wrong numbers on the board.

INDEX