Simulation templates for Processing

AUTHOR
Version 0.99
14.03.2022

Table of Contents

Table of contents

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

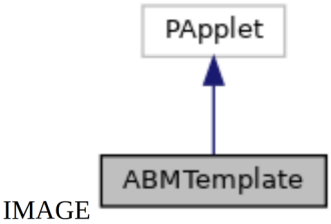
File List

Here is a list of all files with brief descriptions:

Class Documentation

ABMTemplate Class Reference

Inheritance diagram for ABMTemplate:



IMAGE

Classes

class **Agent**

Agent *is a one of two central class of each ABM model.*

class **PairOfInt**

Simple version of Pair containing a pair of integers.

class **World**

The main class of simulation.

Public Member Functions

void **setup** ()

Function ***setup()*** *is called only once, at the beginning of run At least* ***setup()*** *or* ***draw()*** *must be present in animation program.*

void **draw** ()

Function ***draw()*** *is called many times, to the end of run or noLoop() call.*

void **writeStatusLine** ()

Function designed to fill the status bar with simulation statistics.

void **initializeAgents** (**Agent**[][] agents)

Agents need to be initialised & they need logic of change ABM: BASIC INITIALISATION & EVERY STEP CHANGE.

void **initializeAgents** (**Agent**[] agents)

Initialization of agents (1D version)

void **dummyChangeAgents** (**Agent**[][] agents)

Random changes of agents for testing the visualization (2D version)

void **dummyChangeAgents** (**Agent**[] agents)

Random changes of agents for testing the visualization (1D version)

void **changeAgents** (**Agent**[][] agents)

Your agents change over time (2D version)

void **changeAgents** (**Agent**[] agents)

Your agents change over time (1D version)

void **initializeModel** (**World** world)

More alaborated functionalities are defined as stand-alone functions, not as methods because of not enought flexible syntax of Processing.

void **visualizeModel** (**World** world)

Draws a representation of the simulation world.

void **dummyChange** (**World** world)

*Dummy changes for testing of whole class **World**.*

void **modelStep** (**World** world)

Full model step. Change agents and other components if present.

void **initializeStats** ()

It prepares a unique statistics file name, opens the file and enters the header line.

void **doStatistics** (**World** world)

The function calculates all world statistics after the simulation step.

void **doStatisticsOnAgents** (**Agent**[] agents)

***Agent** statistics.*

void **doStatisticsOnAgents** (**Agent**[][] agents)

***Agent** statistics.*

void **visualizeAgents** (**Agent**[][] agents)

***World** full of agents need method of visualisation on screen/window.*

void **visualizeAgents** (**Agent**[] agents)

Visualization of agents. One-dimensional version.

void **keyPressed** ()

Model-specific event handler.

void **exit** ()

Everything that needs to be done when the application is terminated.

void **mouseClicked** ()

This function is automatically run by Processing when any mouse button is pressed.

PairOfInt findCell (**Agent**[][] agents)

Convert mouse coordinates to cell coordinates The parameter is only for checking type and SIZES Works as long as the agents visualization starts at point 0,0.

void **initVideoExport** (processing.core.PApplet parent, String Name, int Frames)
Make the beginning of the movie file!

void **FirstVideoFrame** ()
Initial second sequence for title and copyright.

void **NextVideoFrame** ()
Each subsequent frame of the movie.

void **CloseVideo** ()
This is what we call when we want to close the movie file.

void **settings** ()

Static Public Member Functions

static void **main** (String[] passedArgs)

Package Attributes

String **modelName** ="ABMTemplate"
Template for AGENT BASE MODEL utilized 1D or 2D discrete geometry.

int **side** =75
side of "world" main table

float **density** =0.75f
initial density of agents

World TheWorld =new **World**(**side**)
*Main table will be initialised inside **setup()***

int **cwidth** =15
requested size of cell

int **STATUSHEIGH** =40
height of status bar

int **STEPSperVIS** =1
how many model steps beetwen visualisations

int **FRAMEFREQ** =10
how many model steps per second

boolean **WITH_VIDEO** =false
Make a movie?

boolean **simulationRun** =false
Start/stop flag.

int **StepCounter** =0

World *is a one of two central class of each ABM model.*

PrintWriter **outstat**

Simulation have to collect and write down statistics from every step.

float **meanDummy** =0

average value for the dummy field

int **liveCount** =0

number of living agents

int **searchedX** =-1

Supports agent search on a mouse click, and possible inspection.

int **searchedY** =-1

The vertical coordinate of the mouse cursor.

boolean **Clicked** =false

Was there a click too?

int **selectedX** =-1

Converted into "world" indices, the agent's horizontal coordinate.

int **selectedY** =-1

Converted into "world" indices, the agent's vertical coordinate.

Agent selected =null

VideoExport **videoExport**

Tool for made video from simulation.

String **copyrightNote** ="(c) W.Borkowski @ ISS University of Warsaw"

< Copyright of your movie

Change it to your copyright.

Static Package Attributes

static int **videoFramesFreq** =0

How many frames per second for the movie. It doesn't have to be the same as in frameRate!

static boolean **videoExportEnabled** =false

Has film making been initiated?

Detailed Description

Definition at line 17 of file ABMTemplate.java.

Member Function Documentation

void ABMTemplate.changeAgents (Agent[] *agents*)

Your agents change over time (1D version)

Definition at line 207 of file ABMTemplate.java.

void ABMTemplate.changeAgents (Agent *agents*[][])

Your agents change over time (2D version)

Definition at line 192 of file ABMTemplate.java.

void ABMTemplate.CloseVideo ()

This is what we call when we want to close the movie file.

This function adds an ending second sequence with an author's note NOTE: there should be some "force screen update", but not found :-(So, if you x-click the window while drawing, the last frame will probably be incomplete

Definition at line 672 of file ABMTemplate.java.

void ABMTemplate.doStatistics (World *world*)

The function calculates all world statistics after the simulation step.

... statistics of other things

Definition at line 308 of file ABMTemplate.java.

void ABMTemplate.doStatisticsOnAgents (Agent[] *agents*)

Agent statistics.

One-dimensional version File outstat should be closed in **exit()** --> see Exit.pde

Definition at line 316 of file ABMTemplate.java.

void ABMTemplate.doStatisticsOnAgents (Agent *agents*[][])

Agent statistics.

Two-dimensional version File outstat should be closed in **exit()** --> see Exit.pde

Definition at line 341 of file ABMTemplate.java.

void ABMTemplate.draw ()

Function **draw()** is called many times, to the end of run or noLoop() call.

At least **setup()** or **draw()** must be present in animation program

Definition at line 79 of file ABMTemplate.java.

void ABMTemplate.dummyChange (World *world*)

Dummy changes for testing of whole class **World**.

Definition at line 265 of file ABMTemplate.java.

void ABMTemplate.dummyChangeAgents (Agent[] *agents*)

Random changes of agents for testing the visualization (1D version)

Definition at line 177 of file ABMTemplate.java.

void ABMTemplate.dummyChangeAgents (Agent *agents*[][])

Random changes of agents for testing the visualization (2D version)

Definition at line 164 of file ABMTemplate.java.

void ABMTemplate.exit ()

Everything that needs to be done when the application is terminated.

It is called whenever a window is closed.

Definition at line 484 of file ABMTemplate.java.

PairOfInt ABMTemplate.findCell (Agent *agents*[][])

Convert mouse coordinates to cell coordinates The parameter is only for checking type and SIZES Works as long as the agents visualization starts at point 0,0.

Definition at line 563 of file ABMTemplate.java.

void ABMTemplate.FirstVideoFrame ()

Initial second sequence for title and copyright.

Definition at line 641 of file ABMTemplate.java.

void ABMTemplate.initializeAgents (Agent[] *agents*)

Initialization of agents (1D version)

Definition at line 153 of file ABMTemplate.java.

void ABMTemplate.initializeAgents (Agent *agents*[][])

Agents need to be initialised & they need logic of change ABM: BASIC INITIALISATION & EVERY STEP CHANGE.

Initialization of agents (2D version)

Definition at line 141 of file ABMTemplate.java.

void ABMTemplate.initializeModel (World *world*)

More alaborated functionalities are defined as stand-alone functions, not as methods because of not enought flexible syntax of Processing.

Prepares the **World** class for the first step of the simulation

Definition at line 251 of file ABMTemplate.java.

void ABMTemplate.initializeStats ()

It prepares a unique statistics file name, opens the file and enters the header line.

Definition at line 295 of file ABMTemplate.java.

void ABMTemplate.initVideoExport (processing.core.PApplet *parent*, String *Name*, int *Frames*)

Make the beginning of the movie file!

Definition at line 629 of file ABMTemplate.java.

void ABMTemplate.keyPressed ()

Model-specific event handler.

Of course, the creator of a specific application has to match actions. Automatically run by Processing when any key on the keyboard is pressed. Inside, you can use the variables 'key' and 'keyCode'.

Definition at line 437 of file ABMTemplate.java.

static void ABMTemplate.main (String[] *passedArgs*)`[static]`

Definition at line 691 of file ABMTemplate.java.

void ABMTemplate.modelStep (World *world*)

Full model step. Change agents and other components if present.

Definition at line 272 of file ABMTemplate.java.

void ABMTemplate.mouseClicked ()

This function is automatically run by Processing when any mouse button is pressed.

Inside, you can use the variables 'mouseX' and 'mouseY'.

Definition at line 538 of file ABMTemplate.java.

void ABMTemplate.NextVideoFrame ()

Each subsequent frame of the movie.

Definition at line 655 of file ABMTemplate.java.

void ABMTemplate.settings ()

Definition at line 690 of file ABMTemplate.java.

void ABMTemplate.setup ()

Function **setup()** is called only once, at the beginning of run At least **setup()** or **draw()** must be present in animation program.

Definition at line 41 of file ABMTemplate.java.

void ABMTemplate.visualizeAgents (Agent[] *agents*)

Visualization of agents. One-dimensional version.

Definition at line 399 of file ABMTemplate.java.

void ABMTemplate.visualizeAgents (Agent *agents*[][])

World full of agents need method of visualisation on screen/window.

Visualization of agents. Two-dimensional version

Definition at line 375 of file ABMTemplate.java.

void ABMTemplate.visualizeModel (World *world*)

Draws a representation of the simulation world.

Definition at line 258 of file ABMTemplate.java.

void ABMTemplate.writeStatusLine ()

Function designed to fill the status bar with simulation statistics.

Definition at line 99 of file ABMTemplate.java.

Member Data Documentation

boolean ABMTemplate.Clicked =false[package]

Was there a click too?

Definition at line 515 of file ABMTemplate.java.

String ABMTemplate.copyrightNote ="(c) W.Borkowski @ ISS University of Warsaw"[package]

< Copyright of your movie

Change it to your copyright.

Best in **setup()** function.

Definition at line 625 of file ABMTemplate.java.

int ABMTemplate.cwidth =15[package]

requested size of cell
Definition at line 31 of file ABMTemplate.java.

float ABMTemplate.density =0.75f[package]

initial density of agents
Definition at line 26 of file ABMTemplate.java.

int ABMTemplate.FRAMEFREQ =10[package]

how many model steps per second
Definition at line 34 of file ABMTemplate.java.

int ABMTemplate.liveCount =0[package]

number of living agents
Definition at line 305 of file ABMTemplate.java.

float ABMTemplate.meanDummy =0[package]

average value for the dummy field
Definition at line 304 of file ABMTemplate.java.

String ABMTemplate.modelName ="ABMTemplate"[package]

Template for AGENT BASE MODEL utilized 1D or 2D discrete geometry.
Name of the model is used for log files
Definition at line 24 of file ABMTemplate.java.

PrintWriter ABMTemplate.outstat[package]

Simulation have to collect and write down statistics from every step.
Handle to the text file with the record of model statistics
Definition at line 291 of file ABMTemplate.java.

int ABMTemplate.searchedX =-1[package]

Supports agent search on a mouse click, and possible inspection.
The horizontal coordinate of the mouse cursor
Definition at line 513 of file ABMTemplate.java.

int ABMTemplate.searchedY =-1[package]

The vertical coordinate of the mouse cursor.
Definition at line 514 of file ABMTemplate.java.

Agent ABMTemplate.selected =null[package]

Definition at line 520 of file ABMTemplate.java.

int ABMTemplate.selectedX =-1[package]

Converted into "world" indices, the agent's horizontal coordinate.
Definition at line 518 of file ABMTemplate.java.

int ABMTemplate.selectedY =-1[package]

Converted into "world" indices, the agent's vertical coordinate.
Definition at line 519 of file ABMTemplate.java.

int ABMTemplate.side =75[package]

side of "world" main table
Definition at line 25 of file ABMTemplate.java.

boolean ABMTemplate.simulationRun =false[package]

Start/stop flag.
Definition at line 37 of file ABMTemplate.java.

int ABMTemplate.STATUSHEIGH =40[package]

height of status bar
Definition at line 32 of file ABMTemplate.java.

int ABMTemplate.StepCounter =0[package]

World is a one of two central class of each ABM model.
Global variable for caunting real simulation steps. Value may differ from frameCount.
Definition at line 227 of file ABMTemplate.java.

int ABMTemplate.STEPSperVIS =1[package]

how many model steps beetwen visualisations
Definition at line 33 of file ABMTemplate.java.

World ABMTemplate.TheWorld =new World(side)[package]

Main table will be initialised inside **setup()**

Definition at line 28 of file ABMTemplate.java.

VideoExport ABMTemplate.videoExport[package]

Tool for made video from simulation.

--> `http://funprogramming.org/VideoExport-for-Processing/examples/basic/basic.pde` Apart from the "hamoid" library, you also need to install the ffmpeg program to make it work !!! USAGE/UÅ»YCIE: This initVideoExport function call must be in **setup()** for the Video module to work: initVideoExport(this,FileName,Frames)); // The VideoExport class must have access to // the Processing application object // It's best to run at the end of the **setup()**. // NOTE !!!: The window must be EVEN sizes We call Next Video Frame for each frame of the movie, most often in the draw () function: **NextVideoFrame()**;//Video frame

... and at the end of the video we call CloseVideo: **CloseVideo()**;// Ideally in exit () CLASS object from additional library - must be installed

Definition at line 613 of file ABMTemplate.java.

boolean ABMTemplate.videoExportEnabled =false[static], [package]

Has film making been initiated?

Definition at line 619 of file ABMTemplate.java.

int ABMTemplate.videoFramesFreq =0[static], [package]

How many frames per second for the movie. It doesn't have to be the same as in frameRate!

Definition at line 616 of file ABMTemplate.java.

boolean ABMTemplate.WITH_VIDEO =false[package]

Make a movie?

Definition at line 35 of file ABMTemplate.java.

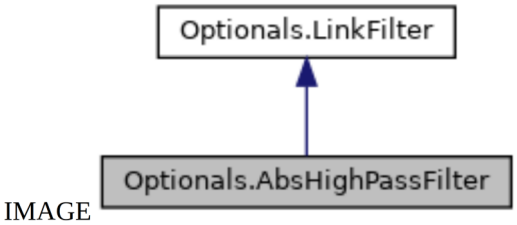
The documentation for this class was generated from the following file:

src.java/**ABMTemplate.java**

Optionals.AbsHighPassFilter Class Reference

Absolute High Pass Filter.

Inheritance diagram for Optionals.AbsHighPassFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

AbsHighPassFilter (float tres)

Package Attributes

float **treshold**

Detailed Description

Absolute High Pass Filter.

highPassFilter filtering links with higher absolute value of weight

Definition at line 1709 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.AbsHighPassFilter.AbsHighPassFilter (float *tres*)[package]

Definition at line 1712 of file Optionals.java.

Member Function Documentation

boolean Optionals.AbsHighPassFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1713 of file Optionals.java.

Member Data Documentation

float Optionals.AbsHighPassFilter.treshold [package]

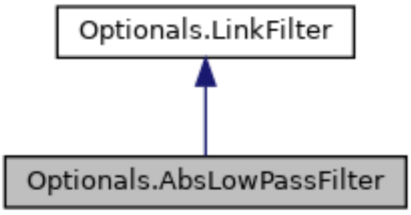
Definition at line 1711 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.AbsLowPassFilter Class Reference

Absolute Low Pass Filter.

Inheritance diagram for Optionals.AbsLowPassFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

AbsLowPassFilter (float tres)

Package Attributes

float **treshold**

Detailed Description

Absolute Low Pass Filter.

lowPassFilter filtering links with lower absolute value of weight

Definition at line 1700 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.AbsLowPassFilter.AbsLowPassFilter (float *tres*) [package]

Definition at line 1703 of file Optionals.java.

Member Function Documentation

boolean Optionals.AbsLowPassFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1704 of file Optionals.java.

Member Data Documentation

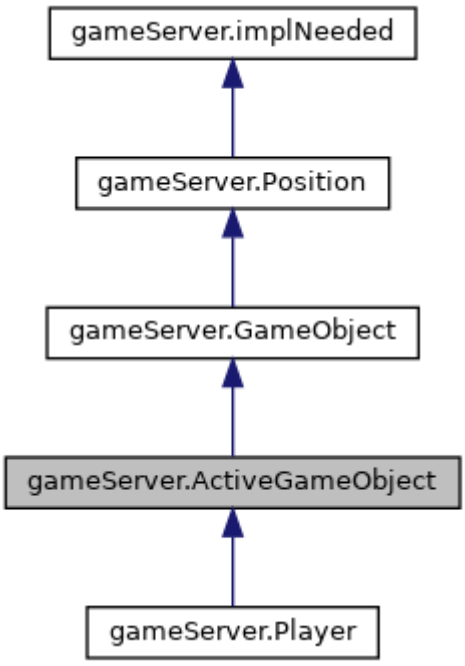
float Optionals.AbsLowPassFilter.treshold [package]

Definition at line 1702 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

gameServer.ActiveGameObject Class Reference

Inheritance diagram for gameServer.ActiveGameObject:



IMAGE

Public Member Functions

boolean **setState** (String field, String val)

Interface for changing the state of the game object over the network (mostly)

String **sayState** ()

The function creates a message block from those object state elements that have change flags (for network streaming)

Package Functions

ActiveGameObject (String iniName, float iniX, float iniY, float iniZ, float iniRadius)

constructor

Package Attributes

float **activeRadius** =1
GameObject interactionObject =null

Detailed Description

Definition at line 380 of file gameServer.java.

Constructor & Destructor Documentation

gameServer.ActiveGameObject.ActiveGameObject (String *iniName*, float *iniX*, float *iniY*, float *iniZ*, float *iniRadius*) [package]

constructor

Definition at line 386 of file gameServer.java.

Member Function Documentation

String gameServer.ActiveGameObject.sayState ()

The function creates a message block from those object state elements that have change flags (for network streaming)

Returns

: Ready to send list of all changes made on object (based on flags)

Reimplemented from **gameServer.GameObject** (*p.75*).

Reimplemented in **gameServer.Player** (*p.177*).

Definition at line 405 of file gameServer.java.

boolean gameServer.ActiveGameObject.setState (String *field*, String *val*)

Interface for changing the state of the game object over the network (mostly)

Reimplemented from **gameServer.GameObject** (*p.76*).

Reimplemented in **gameServer.Player** (*p.177*).

Definition at line 392 of file gameServer.java.

Member Data Documentation

float gameServer.ActiveGameObject.activeRadius =1 [package]

Definition at line 382 of file gameServer.java.

GameObject gameServer.ActiveGameObject.interactionObject =null [package]

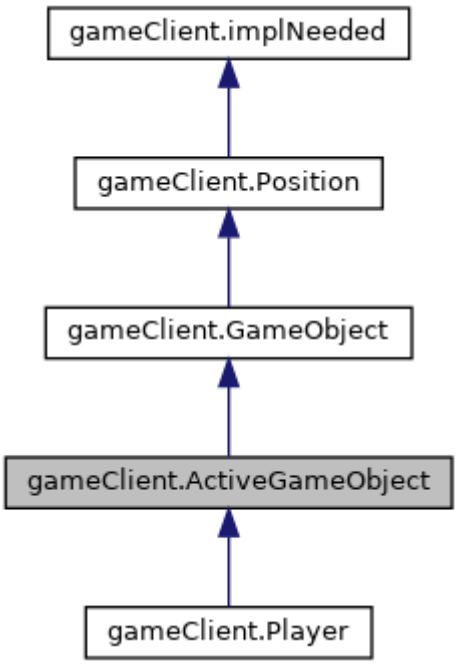
Definition at line 383 of file gameServer.java.

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

src.java/**gameServer.java**

gameClient.ActiveGameObject Class Reference

Inheritance diagram for gameClient.ActiveGameObject:



IMAGE

Public Member Functions

boolean **setState** (String field, String val)

Interface for changing the state of the game object over the network (mostly)

String **sayState** ()

The function creates a message block from those object state elements that have change flags (for network streaming)

Package Functions

ActiveGameObject (String iniName, float iniX, float iniY, float iniZ, float iniRadius)

constructor

Package Attributes

float **activeRadius** =1

Radius for active interaction with others objects.

GameObject interactionObject =null

Only one in a time.

Detailed Description

Definition at line 662 of file gameClient.java.

Constructor & Destructor Documentation

gameClient.ActiveGameObject.ActiveGameObject (String *iniName*, float *iniX*, float *iniY*, float *iniZ*, float *iniRadius*) [package]

constructor

Definition at line 668 of file gameClient.java.

Member Function Documentation

String gameClient.ActiveGameObject.sayState ()

The function creates a message block from those object state elements that have change flags (for network streaming)

Returns

: Ready to send list of all changes made on object (based on flags)

Reimplemented from **gameClient.GameObject** (*p.79*).

Reimplemented in **gameClient.Player** (*p.174*).

Definition at line 687 of file gameClient.java.

boolean gameClient.ActiveGameObject.setState (String *field*, String *val*)

Interface for changing the state of the game object over the network (mostly)

Reimplemented from **gameClient.GameObject** (*p.79*).

Reimplemented in **gameClient.Player** (*p.174*).

Definition at line 674 of file gameClient.java.

Member Data Documentation

float gameClient.ActiveGameObject.activeRadius =1 [package]

Radius for active interaction with others objects.

Definition at line 664 of file gameClient.java.

GameObject gameClient.ActiveGameObject.interactionObject =null [package]

Only one in a time.

Definition at line 665 of file gameClient.java.

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

src.java/**gameClient.java**

ABMTemplate.Agent Class Reference

Agent is a one of two central class of each ABM model.

Package Functions

Agent ()

*Constructor of the **Agent**.*

Package Attributes

float **dummy**

Detailed Description

Agent is a one of two central class of each ABM model.

Agent class

Definition at line 118 of file ABMTemplate.java.

Constructor & Destructor Documentation

ABMTemplate.Agent.Agent ()[package]

Constructor of the **Agent**.

Definition at line 124 of file ABMTemplate.java.

Member Data Documentation

float ABMTemplate.Agent.dummy[package]

Definition at line 120 of file ABMTemplate.java.

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

src.java/**ABMTemplate.java**

Optionals.Agent Class Reference

Dummy class of **Agent**.

Package Attributes

float **A**

Detailed Description

Dummy class of **Agent**.

Definition at line 40 of file Optionals.java.

Member Data Documentation

float Optionals.Agent.A [package]

Definition at line 42 of file Optionals.java.

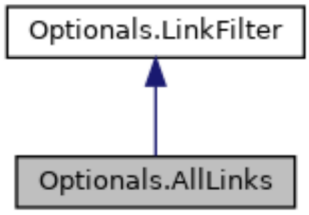
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.AllLinks Class Reference

Different filters of links and other link tools for a (social) network.

Inheritance diagram for Optionals.AllLinks:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Detailed Description

Different filters of links and other link tools for a (social) network.

Available filters: **AllLinks**, **AndFilter**, **OrFilter**, **TypeFilter**, **LowPassFilter**, **HighPassFilter**, **AbsLowPassFilter**, **AbsHighPassFilter** **TypeAndAbsHighPassFilter** - special type for efficient visualisation Simplest link filtering class which accepts all links

Definition at line 1627 of file Optionals.java.

Member Function Documentation

boolean Optionals.AllLinks.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1629 of file Optionals.java.

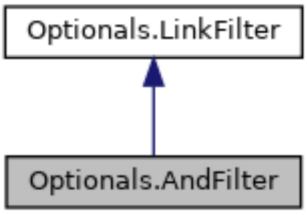
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.AndFilter Class Reference

AND two filters assembly class.

Inheritance diagram for Optionals.AndFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

AndFilter (**LinkFilter** aa, **LinkFilter** bb)

Package Attributes

LinkFilter a
LinkFilter b

Detailed Description

AND two filters assembly class.

A class for logically joining two filters with the AND operator.

Definition at line 1647 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.AndFilter.AndFilter (LinkFilter *aa*, LinkFilter *bb*) [package]

Definition at line 1651 of file Optionals.java.

Member Function Documentation

boolean Optionals.AndFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1652 of file Optionals.java.

Member Data Documentation

LinkFilter Optionals.AndFilter.a [package]

Definition at line 1649 of file Optionals.java.

LinkFilter Optionals.AndFilter.b [package]

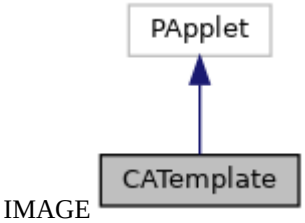
Definition at line 1650 of file Optionals.java.

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

src.java/**Optionals.java**

CATemplate Class Reference

Inheritance diagram for CATemplate:



IMAGE

Classes

class **PairOfInt**

Simple version of Pair containing a pair of integers.

class **World**

The main class of simulation.

Public Member Functions

void **setup** ()

*Function **setup()** is called only once, at the beginning of run At least **setup()** or **draw()** must be present in animation program.*

void **draw** ()

*Function **draw()** is called many times, to the end of run or noLoop() call.*

void **writeStatusLine** ()

Function designed to fill the status bar with simulation statistics.

void **initializeModel** (**World** world)

More alaborated functionalities are defined as stand-alone functions, not as methods because of not enought flexible syntax of Processing.

void **visualizeModel** (**World** world)

Draws a representation of the simulation world.

void **exampleChange** (**World** world)

Example changes of cells for testing the visualization.

void **modelStep** (**World** world)

Full model step. Change cells and other components if present.

void **initializeCells** (int[][] cells)

Cell is a one of two central types (typicaly char or int) of each CA model Cells need to be initialised & they need rules of change

void **initializeCells** (int[] cells)

Initialization of cells (1D version)

void **synchChangeCellsModulo** (int[][] cells, int[][] newcells)
Your cells change over time (SYNCHRONIC 2D version) (Example "modulo 5" model)

void **synchChangeCellsModulo** (int[] cells, int[] newcells)
Your cells change over time (SYNCHRONIC 1D version) (Example "modulo 5" model)

void **asyncChangeCellsModulo** (int[][] cells)
Your cells change over time (ASYNCHRONIC 2D version) (Example "modulo 5" model)

void **asyncChangeCellsModulo** (int[] cells)
Your cells change over time (ASYNCHRONIC 1D version) (Example "modulo 5" model)

void **initializeStats** ()
It prepares a unique statistics file name, opens the file and enters the header line.

void **doStatistics** (**World** world)
The function calculates all world statistics after the simulation step.

void **doStatisticsOnCells** (int[] cells)
Cell statistics.

void **doStatisticsOnCells** (int[][] cells)
Cell statistics.

void **visualizeCells** (int[][] cells)
***World** full of cells need method of visualisation on screen/window.*

void **visualizeCells** (int[] cells)
Visualization of cells. One-dimensional version.

void **keyPressed** ()
Model-specific event handler.

void **exit** ()
Everything that needs to be done when the application is terminated.

void **mouseClicked** ()
This function is automatically run by Processing when any mouse button is pressed.

PairOfInt findCell (int[][] cells)
Convert mouse coordinates to cell coordinates The parameter is only for checking type and SIZES Works as long as the cell visualization starts at point 0,0.

void **initVideoExport** (processing.core.PApplet parent, String Name, int Frames)
Make the beginning of the movie file!

void **FirstVideoFrame** ()
Initial second sequence for title and copyright.

void **NextVideoFrame** ()
Each subsequent frame of the movie.

void **CloseVideo** ()
This is what we call when we want to close the movie file.

void **settings** ()

Static Public Member Functions

static void **main** (String[] passedArgs)

Package Attributes

String **modelName** ="CATemplate"
Template for CA MODEL utilized 1D or 2D discrete geometry.

int **side** =201
side of "world" main table

float **density** =0.0000f
initial density of live cells

boolean **synchronicMode** =true
if false, then Monte Carlo mode is used

World TheWorld =new **World**(**side**)
*Main table will be initialised inside **setup()***

int **cwidth** =3
requested size of cell

int **STATUSHEIGH** =40
height of status bar

int **STEPSperVIS** =1
how many model steps beetwen visualisations

int **FRAMEFREQ** = 50
how many model steps per second

boolean **WITH_VIDEO** =false
Need the application make a movie?

boolean **simulationRun** =true
Start/stop flag.

int **StepCounter** =0

World *is a one of two central class of each CA model.*

PrintWriter **outstat**

Simulation have to collect and write down statistics from every step.

float **meanDummy** =0

the average of the non-zero cell values

int **liveCount** =0

number of non-zero cells

int **searchedX** =-1

The horizontal coordinate of the mouse cursor.

int **searchedY** =-1

The vertical coordinate of the mouse cursor.

boolean **Clicked** =false

Was there a click too?

int **selectedX** =-1

Converted into "world" indices, the agent's horizontal coordinate.

int **selectedY** =-1

Converted into "world" indices, the agent's vertical coordinate.

VideoExport **videoExport**

Tool for made video from simulation.

String **copyrightNote** ="(c) W.Borkowski @ ISS University of Warsaw"

< Copyright of your movie

Change it to your copyright.

Static Package Attributes

static int **videoFramesFreq** =0

How many frames per second for the movie. It doesn't have to be the same as in frameRate!

static boolean **videoExportEnabled** =false

Has film making been initiated?

Detailed Description

Definition at line 17 of file CATemplate.java.

Member Function Documentation

void CATemplate.asyncChangeCellsModulo (int[] *cells*)

Your cells change over time (ASYNCHRONIC 1D version) (Example "modulo 5" model)

Definition at line 294 of file CATemplate.java.

void CATemplate.asyncChangeCellsModulo (int *cells*[][])

Your cells change over time (ASYNCHRONIC 2D version) (Example "modulo 5" model)

Definition at line 272 of file CATemplate.java.

void CATemplate.CloseVideo ()

This is what we call when we want to close the movie file.

This function adds an ending second sequence with an author's note NOTE: there should be some "force screen update", but not found :-(So, if you x-click the window while drawing, the last frame will probably be incomplete

Definition at line 702 of file CATemplate.java.

void CATemplate.doStatistics (World *world*)

The function calculates all world statistics after the simulation step.

... statistics of other things

Definition at line 331 of file CATemplate.java.

void CATemplate.doStatisticsOnCells (int[] *cells*)

Cell statistics.

One-dimensional version outstat file should be closed in **exit()** --> see Exit.pde

Definition at line 339 of file CATemplate.java.

void CATemplate.doStatisticsOnCells (int *cells*[][])

Cell statistics.

Two-dimensional version outstat file should be closed in **exit()** --> see Exit.pde

Definition at line 374 of file CATemplate.java.

void CATemplate.draw ()

Function **draw()** is called many times, to the end of run or noLoop() call.

At least **setup()** or **draw()** must be present in animation program

Definition at line 81 of file CATemplate.java.

void CATemplate.exampleChange (World *world*)

Example changes of cells for testing the visualization.

Definition at line 172 of file CATemplate.java.

void CATemplate.exit ()

Everything that needs to be done when the application is terminated.

It is called whenever a window is closed.

Definition at line 520 of file CATemplate.java.

PairOfInt CATemplate.findCell (int *cells*[][])

Convert mouse coordinates to cell coordinates The parameter is only for checking type and SIZES Works as long as the cell visualization starts at point 0,0.

Definition at line 593 of file CATemplate.java.

void CATemplate.FirstVideoFrame ()

Initial second sequence for title and copyright.

Definition at line 671 of file CATemplate.java.

void CATemplate.initializeCells (int[] *cells*)

Initialization of cells (1D version)

Definition at line 219 of file CATemplate.java.

void CATemplate.initializeCells (int *cells*[][])

Cell is a one of two central types (typicaly char or int) of each CA model Cells need to be initialised & they need rules of change

Initialization of cells (2D version)

Definition at line 205 of file CATemplate.java.

void CATemplate.initializeModel (World *world*)

More alaborated functionalities are defined as stand-alone functions, not as methods because of not enought flexible syntax of Processing.

Prepares the **World** class for the first step of the simulation

Definition at line 158 of file CATemplate.java.

void CATemplate.initializeStats ()

It prepares a unique statistics file name, opens the file and enters the header line.

Definition at line 320 of file CATemplate.java.

void CATemplate.initVideoExport (processing.core.PApplet *parent*, String *Name*, int *Frames*)

Make the beginning of the movie file!

Definition at line 659 of file CATemplate.java.

void CATemplate.keyPressed ()

Model-specific event handler.

Of course, the creator of a specific application has to match actions. Automatically run by Processing when any key on the keyboard is pressed. Inside, you can use the variables 'key' and 'keyCode'.

Definition at line 474 of file CATemplate.java.

static void CATemplate.main (String[] *passedArgs*)`[static]`

Definition at line 721 of file CATemplate.java.

void CATemplate.modelStep (World *world*)

Full model step. Change cells and other components if present.

Definition at line 186 of file CATemplate.java.

void CATemplate.mouseClicked ()

This function is automatically run by Processing when any mouse button is pressed.

Inside, you can use the variables 'mouseX' and 'mouseY'.

Definition at line 573 of file CATemplate.java.

void CATemplate.NextVideoFrame ()

Each subsequent frame of the movie.

Definition at line 685 of file CATemplate.java.

void CATemplate.settings ()

Definition at line 720 of file CATemplate.java.

void CATemplate.setup ()

Function **setup()** is called only once, at the beginning of run At least **setup()** or **draw()** must be present in animation program.

Definition at line 42 of file CATemplate.java.

void CATemplate.synchChangeCellsModulo (int[] *cells*, int[] *newcells*)

Your cells change over time (SYNCHRONIC 1D version) (Example "modulo 5" model)

Definition at line 257 of file CATemplate.java.

void CATemplate.synchChangeCellsModulo (int *cells*[][], int *newcells*[][])

Your cells change over time (SYNCHRONIC 2D version) (Example "modulo 5" model)

Definition at line 236 of file CATemplate.java.

void CATemplate.visualizeCells (int[] *cells*)

Visualization of cells. One-dimensional version.

Definition at line 440 of file CATemplate.java.

void CATemplate.visualizeCells (int *cells*[][])

World full of cells need method of visualisation on screen/window.

Visualization of cells. Two-dimensional version

Definition at line 417 of file CATemplate.java.

void CATemplate.visualizeModel (World *world*)

Draws a representation of the simulation world.

Definition at line 165 of file CATemplate.java.

void CATemplate.writeStatusLine ()

Function designed to fill the status bar with simulation statistics.

Definition at line 101 of file CATemplate.java.

Member Data Documentation

boolean CATemplate.Clicked =false[package]

Was there a click too?

Definition at line 550 of file CATemplate.java.

String CATemplate.copyrightNote ="(c) W.Borkowski @ ISS University of Warsaw"[package]

< Copyright of your movie
Change it to your copyright.
Best in **setup()** function.
Definition at line 655 of file CATemplate.java.

int CATemplate.cwidth =3[package]

requested size of cell
Definition at line 32 of file CATemplate.java.

float CATemplate.density =0.0000f[package]

initial density of live cells
Definition at line 26 of file CATemplate.java.

int CATemplate.FRAMEFREQ = 50[package]

how many model steps per second
Definition at line 35 of file CATemplate.java.

int CATemplate.liveCount =0[package]

number of non-zero cells
Definition at line 328 of file CATemplate.java.

float CATemplate.meanDummy =0[package]

the average of the non-zero cell values
Definition at line 327 of file CATemplate.java.

String CATemplate.modelName ="CATemplate"[package]

Template for CA MODEL utilized 1D or 2D discrete geometry.
Name of the model is used for log files
Definition at line 24 of file CATemplate.java.

PrintWriter CATemplate.outstat[package]

Simulation have to collect and write down statistics from every step.
Handle to the text file with the record of model statistics
Definition at line 316 of file CATemplate.java.

int CATemplate.searchedX =-1[package]

The horizontal coordinate of the mouse cursor.

Definition at line 548 of file CATemplate.java.

int CATemplate.searchedY =-1[package]

The vertical coordinate of the mouse cursor.
Definition at line 549 of file CATemplate.java.

int CATemplate.selectedX =-1[package]

Converted into "world" indices, the agent's horizontal coordinate.
Definition at line 553 of file CATemplate.java.

int CATemplate.selectedY =-1[package]

Converted into "world" indices, the agent's vertical coordinate.
Definition at line 554 of file CATemplate.java.

int CATemplate.side =201[package]

side of "world" main table
Definition at line 25 of file CATemplate.java.

boolean CATemplate.simulationRun =true[package]

Start/stop flag.
Definition at line 38 of file CATemplate.java.

int CATemplate.STATUSHEIGH =40[package]

height of status bar
Definition at line 33 of file CATemplate.java.

int CATemplate.StepCounter =0[package]

World is a one of two central class of each CA model.
Global variable for caunting real simulation steps. Value may differ from frameCount.
Definition at line 119 of file CATemplate.java.

int CATemplate.STEPSperVIS =1[package]

how many model steps beetwen visualisations
Definition at line 34 of file CATemplate.java.

boolean CATemplate.synchronicMode =true[package]

if false, then Monte Carlo mode is used

Definition at line 27 of file CATemplate.java.

World CATemplate.TheWorld =new World(side)[package]

Main table will be initialised inside **setup()**

Definition at line 29 of file CATemplate.java.

VideoExport CATemplate.videoExport[package]

Tool for made video from simulation.

--> `http://funprogramming.org/VideoExport-for-Processing/examples/basic/basic.pde` Apart from the "hamoid" library, you also need to install the ffmpeg program to make it work !!! USAGE/UÅ»YCIE: This initVideoExport function call must be in **setup()** for the Video module to work: initVideoExport(this,FileName,Frames)); // The VideoExport class must have access to // the Processing application object // It's best to run at the end of the **setup()**. // NOTE !!!: The window must be EVEN sizes We call Next Video Frame for each frame of the movie, most often in the draw () function: **NextVideoFrame()**;//Video frame

... and at the end of the video we call CloseVideo: **CloseVideo()**;// Ideally in exit () CLASS object from additional library - must be installed

Definition at line 643 of file CATemplate.java.

boolean CATemplate.videoExportEnabled =false[static], [package]

Has film making been initiated?

Definition at line 649 of file CATemplate.java.

int CATemplate.videoFramesFreq =0[static], [package]

How many frames per second for the movie. It doesn't have to be the same as in frameRate!

Definition at line 646 of file CATemplate.java.

boolean CATemplate.WITH_VIDEO =false[package]

Need the application make a movie?

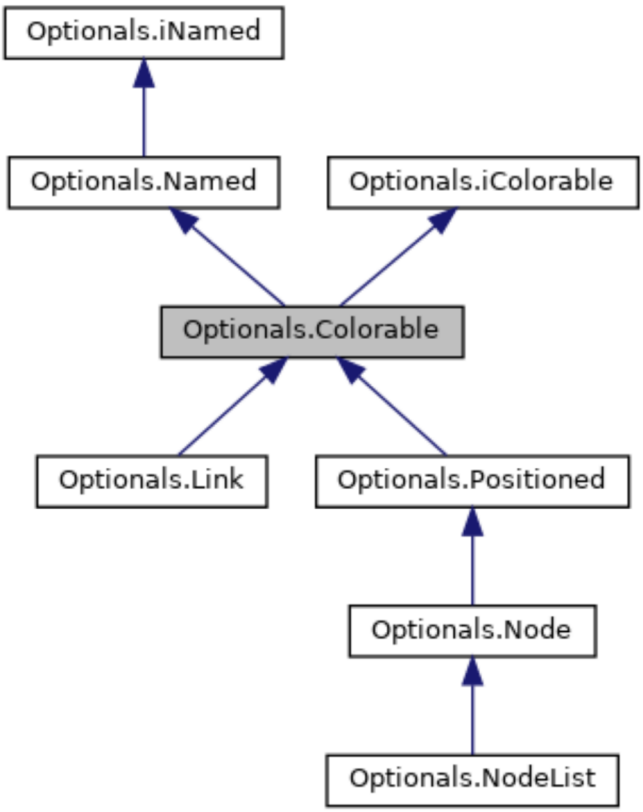
Definition at line 36 of file CATemplate.java.

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

src.java/**CATemplate.java**

Optionals.Colorable Class Reference

Inheritance diagram for Optionals.Colorable:



IMAGE

Public Member Functions

void **setFill** (float modifier)

INFO: For visualisation.

void **setStroke** (float modifier)

Detailed Description

Definition at line 1810 of file Optionals.java.

Member Function Documentation

void Optionals.Colorable.setFill (float *modifier*)

INFO: For visualisation.

Implements **Optionals.iColorable** (*p.101*).

Definition at line 1812 of file Optionals.java.

void Optionals.Colorable.setStroke (float *modifier*)

Implements **Optionals.iColorable** (*p.101*).

Reimplemented in **Optionals.Link** (*p.118*).

Definition at line 1813 of file Optionals.java.

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

src.java/**Optionals.java**

Optionals.describable Interface Reference

Generally useable interfaces:

Public Member Functions

String **getDescription** ()

Detailed Description

Generally useable interfaces:

Any object which have description as (potentially) long, multi line string @ABSOLETE!

Definition at line 1538 of file Optionals.java.

Member Function Documentation

String Optionals.describable.getDescription ()

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

src.java/**Optionals.java**

Optionals.DummyBool Class Reference

A class for taking an object from a simple logic variable (true-false).

Package Attributes

boolean **val** =false

Detailed Description

A class for taking an object from a simple logic variable (true-false).

Needed for common configuration values or to pass to functions by reference.

Definition at line 587 of file Optionals.java.

Member Data Documentation

boolean Optionals.DummyBool.val =false [package]

Definition at line 588 of file Optionals.java.

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

src.java/**Optionals.java**

Optionals.DummyDouble Class Reference

A class for taking an object from a simple variable of type double.

Package Attributes

double **val** =0

Detailed Description

A class for taking an object from a simple variable of type double.

Needed for common configuration values or to pass to functions by reference.

Definition at line 597 of file Optionals.java.

Member Data Documentation

double Optionals.DummyDouble.val =0[package]

Definition at line 598 of file Optionals.java.

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

src.java/**Optionals.java**

Optionals.DummyFloat Class Reference

A class for taking an object from a simple variable of type float.

Package Attributes

float **val** =0

Detailed Description

A class for taking an object from a simple variable of type float.

Needed for common configuration values or to pass to functions by reference.

Definition at line 592 of file Optionals.java.

Member Data Documentation

float Optionals.DummyFloat.val =0 [package]

Definition at line 593 of file Optionals.java.

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

src.java/**Optionals.java**

Optionals.DummyInt Class Reference

Classes for taking an object from a simple variable of type int, boolean, float & double.

Package Attributes

int **val** =0

Detailed Description

Classes for taking an object from a simple variable of type int, boolean, float & double.

Useful when you need a REFERENCE to a value. In Processing as hell :-) I can't find how to pass something other than an object by reference However, the existing Integer or Float classes are not suitable for this because they are "final". They will behave like constants. And sometimes such an opportunity is needed! A class for taking an object from a simple variable of type int. Needed for common configuration values or to pass to functions by reference.

Definition at line 582 of file Optionals.java.

Member Data Documentation

int Optionals.DummyInt.val =0 [package]

Definition at line 583 of file Optionals.java.

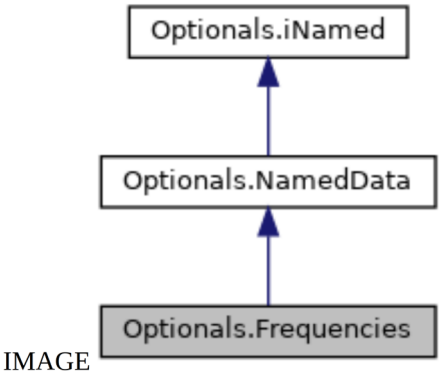
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.Frequencies Class Reference

This class represens a named histogram of frequencies.

Inheritance diagram for Optionals.Frequencies:



IMAGE

Public Member Functions

int **numOfElements** ()

In this case, the items are histogram buckets.

void **reset** ()

Ready to start collecting data again.

void **addValue** (float value)

Adding the real value updates the corresponding bucket.

Package Functions

Frequencies (int numberOfBuckets, float lowerBound, float upperBound, String Name)

Constructor needs more than a name.

Package Attributes

float **sizeOfbucket** =0
float **lowerb** =+Float.MAX_VALUE
float **upperb** =-Float.MAX_VALUE
int **outsideLow** =0
int **outsideHig** =0
int **inside** =0
int **higherBucket** =0
int **higherBucketIndex** =-1

Private Attributes

int[] **buckets** =null

Detailed Description

This class represens a named histogram of frequencies.

Definition at line 210 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.Frequencies.Frequencies (int *numberOfBuckets*, float *lowerBound*, float *upperBound*, String *Name*)[package]

Constructor needs more than a name.

Definition at line 223 of file Optionals.java.

Member Function Documentation

void Optionals.Frequencies.addValue (float *value*)

Adding the real value updates the corresponding bucket.

Definition at line 248 of file Optionals.java.

int Optionals.Frequencies.numOfElements ()

In this case, the items are histogram buckets.

Definition at line 233 of file Optionals.java.

void Optionals.Frequencies.reset ()

Ready to start collecting data again.

Definition at line 236 of file Optionals.java.

Member Data Documentation

int [] Optionals.Frequencies.buckets =null[private]

Definition at line 212 of file Optionals.java.

int Optionals.Frequencies.higherBucket =0[package]

Definition at line 219 of file Optionals.java.

int Optionals.Frequencies.higherBucketIndex =-1[package]

Definition at line 220 of file Optionals.java.

int Optionals.Frequencies.inside =0[package]

Definition at line 218 of file Optionals.java.

float Optionals.Frequencies.lowerb =+Float.MAX_VALUE [package]

Definition at line 214 of file Optionals.java.

int Optionals.Frequencies.outsideHig =0 [package]

Definition at line 217 of file Optionals.java.

int Optionals.Frequencies.outsideLow =0 [package]

Definition at line 216 of file Optionals.java.

float Optionals.Frequencies.sizeOfbucket =0 [package]

Definition at line 213 of file Optionals.java.

float Optionals.Frequencies.upperb =-Float.MAX_VALUE [package]

Definition at line 215 of file Optionals.java.

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

src.java/**Optionals.java**

Optionals.Function2D Interface Reference

A function of two values in the form of a class - a functor.

Public Member Functions

float **calculate** (float X, float Y)
float **getMin** ()
float **getMax** ()

Detailed Description

A function of two values in the form of a class - a functor.

Definition at line 1560 of file Optionals.java.

Member Function Documentation

float Optionals.Function2D.calculate (float *X*, float *Y*)

float Optionals.Function2D.getMax ()

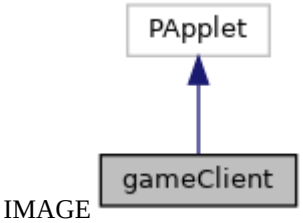
float Optionals.Function2D.getMin ()

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

src.java/**Optionals.java**

gameClient Class Reference

Inheritance diagram for gameClient:



IMAGE

Classes

class **ActiveGameObject**

class **GameObject**

Representation of simple game object.

class **implNeeded**

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

class **Opcs**

Protocol dictionary ("opcodes" etc.)

class **Player**

Representation of generic player.

class **Position**

Representation of 3D position in the game world However, the value of Z is not always used.

Public Member Functions

void **setup** ()

Startup of a game client. Still not connected after that.

void **draw** ()

A function that is triggered many times per second, carrying out the main tasks of the client.

void **drawStartUpInfo** ()

Intro view placeholder ;-)

void **whenConnectedToServer** ()

Initial courtesy exchange with the server.

void **drawTryConnect** ()

Attempting to connect and initial communication.

void **loadSettings** ()

Loads the player's name from the file "player.txt" But may do more...

void **implementObjectMenagement** (String[] cmd)

Simple object type management.

GameObject makeGameObject (String name, float X, float Y, float Z, float R)
Very simple placeholder for use types dictionary.

void **visualisationChanged** (**GameObject**[] table, String name, String vtype)
Handle changes in visualisation type of any game object.

void **colorChanged** (**GameObject**[] table, String name, String hexColor)
Handle changes in colors of any game object.

void **positionChanged** (**GameObject**[] table, String name, float[] inpos)
Handle changes in position of any game object.

void **stateChanged** (**GameObject**[] table, String objectName, String field, String val)
Handle changes in states of agents/objects.

void **beginInteraction** (**GameObject**[] table, String[] objectAndActionsNames)
Handling for getting the avatar in contact with the game object.

void **finishInteraction** (**GameObject**[] table, String objectName)
Handling for getting out of the avatar's contact with the game object.

void **interpretMessage** (String msg)
Handling interpretation of messages from server.

void **clientGameDraw** ()
This function performs communication with server & visualisation of the game world.

void **keyPressed** ()
***gameClient** - keyboard input & other asynchronic events*

void **disconnectEvent** (Client client)
ClientEvent message is generated when a client disconnects.

void **serverEvent** (Server server, Client client)
Event handler called on server when a client connects to server.

int **localiseByName** (**GameObject**[] table, String name)
Determines the index of the object with the specified proper name in an array of objects or players.

void **removeObject** (**GameObject**[] table, String name)
It removes object reffered by name from the table.

int **findCollision** (**GameObject**[] table, int indexOfMoved, int startIndex, boolean withZ)
It finds the index of the first collided object 'indexOfMoved' is the index of the object for which we check for collisions.

String **makeAllTypeInfo** (**GameObject**[] table)
Prepares information about the types and names of all objects on the game board.

void **visualise2D** (float startX, float startY, float width, float height)
Simplest flat map visualisation of game board.

boolean **playerMove** (String dir, **Player** player)
Moves allowed for the player.

void **playerAction** (String action, **Player** player)
The actions of agents and players in the game are defined by names in the protocol, thanks to which their set is expandable.

void **performAction** (**ActiveGameObject** subject, String action, **GameObject** object)
Actions placeholder.

String **sayHELLO** (String myName)
It composes server-client handshake.

String **decodeHELLO** (String msgHello)
It decodes handshake.

String **sayOptCode** (char optCode)
It composes one OPC info.

String **sayOptAndInf** (char opCode, String inf)
*It composes simple string info - **Opcs.SPC** inside 'inf' is allowed.*

String **decodeOptAndInf** (String msg)
Decode one string message.

String **sayOptAndInfos** (char opCode, String objName, String info)
Compose one string info - SPC inside info is NOT allowed.

String **sayOptAndInfos** (char opCode, String objName, String info1, String info2)
Compose many(=2) string info - SPC inside infos is NOT allowed.

String **sayOptAndInfos** (char opCode, String objName, String info1, String info2, String info3)
Compose many(=3) string info - SPC inside infos is NOT allowed.

String **decodeInfos** (String msgInfos, String[] infos)
It decodes 1-9 infos message.

String **sayTouch** (String nameOfTouched, float distance, String actionDef)
It constructs touch message with only one possible action.

String **sayTouch** (String nameOfTouched, float distance, String action1, String action2)

It constructs touch message with two possible actions.

String **sayTouch** (String nameOfTouched, float distance, String[] actions)

It constructs touch message with many possible actions.

float **decodeTouch** (String msg, String[] infos)

It decodes touch message.

String **sayPosition** (char EUCorPOL, String objName, float coord)

It composes message about object position (1 dimension) E1 OName Data @ - Euclidean position float(X) P1 OName Data @ - Polar position float(Alfa +-180)

String **sayPosition** (char EUCorPOL, String objName, float coord1, float coord2)

It composes message about object position (2 dimensions)

*E2 OName Data*2 @ - Euclidean position float(X) float(Y) P2 OName Data*2 @ - Polar position float(Alfa +-180) float(DISTANCE) OName == object identification or name of player or 'Y'.*

String **sayPosition** (char EUCorPOL, String objName, float coord1, float coord2, float coord3)

*It composes message about object position (3 dimensions) E3 OName Data*3 @ - Euclidean position float(X) float(Y) float(H) P3 OName Data*3 @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) OName == object identification or name of player or 'Y'.*

String **sayPosition** (char EUCorPOL, String objName, float[] coordinates)

*It composes message about object position (1-9 dimensions) En OName Data*n @ - Euclidean position float(X) float(Y) float(H) "class name of object or name of player" Pn OName Data*n @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) "class name of object or name of player" OName == object identification or name of player or 'Y'.*

String **decodePosition** (String msgPosition, float[] coordinates)

It decodes 1-9 dimensional positioning message.

String **sayObjectType** (String type, String objectName)

For objects types management - type of object.

String **sayObjectRemove** (String objectName)

For objects types management - object removing from the game world.

String[] **decodeObjectMng** (String msg)

It decodes message of objects types management - decoding.

void **settings** ()

Static Public Member Functions

static void **main** (String[] passedArgs)

Package Functions

void **clientEvent** (Client client)

ClientEvent message is generated when the server sends data to an existing client.

Package Attributes

int **DEBUG** =1

*Client for **gameServer - setup()** & **draw()** SOURCE FILE.*

int **VIEWMESG** =0
int **INTRO_FRAMES** =3
int **DEF_FRAME_RATE** =60
String **playerName** =""
Client **myClient** =null
StringDict **inventory** =new StringDict()

***gameClient** - more communication logic*

float[] **inparr1** =new float[1]
float[] **inparr2** =new float[2]
float[] **inparr3** =new float[3]
String[] **instr1** =new String[1]
String[] **instr2** =new String[2]
String[] **instr3** =new String[3]
int **initialSizeOfMainArray** =30

Game classes and its basic behaviours.

float **initialMaxX** =100

Initial horizontal size of game "board".

float **initialMaxY** =100

Initial vertical size of game "board".

int **indexOfMe** =-1

Index of object visualising the client or the server supervisor.

String[] **plants** = {"_","O","...\nI","_\\|/_\nI ","|/",":","â˜˙ï¸"}

plants...

String[] **avatars** ={".","^v^" ,"o^o","@","&","ðŸ˜ƒ","ðŸ˜Ⴉ"}

peoples...

final int **VISSWITH** = unbinary("000000001")

object is invisible (but in info level name is visible)

final int **MOVED_MSK** = unbinary("000000010")

object was moved (0x1)

final int **VISUAL_MSK** = unbinary("000000100")

object changed its type of view

final int **COLOR_MSK** = unbinary("000001000")
object changed its colors

final int **HPOINT_MSK** = unbinary("000010000")
object changed its hp state (most frequently changed state)

final int **SCORE_MSK** = unbinary("000100000")
object changed its score (for players it is most frequently changed state)

final int **PASRAD_MSK** = unbinary("001000000")
object changed its passive radius (ex. grow);

final int **ACTRAD_MSK** = unbinary("010000000")
object changed its radius of activity (ex. go to sleep);

final int **STATE_MSK** = **HPOINT_MSK** | **SCORE_MSK** | **PASRAD_MSK** | **ACTRAD_MSK**
object changed its states

final int **TOUCH_MSK** = unbinary("1000000000000000")
To visualize the interaction between background objects.

final int **ALL_CHNG_MSK** = **MOVED_MSK** | **VISUAL_MSK** | **COLOR_MSK** | **STATE_MSK**
All initial changes.

int **INFO_LEVEL** =1 | **SCORE_MSK**
Visualisation with information about objects (name & score by default)

boolean **VIS_MIN_MAX** =true
Visualisation with min/max value.

boolean **KEEP_ASPECT** =true
Visualisation with proportional aspect ratio.

GameObject[] **gameWorld** =null
MAIN ARRAY OF GameObjects.

String **serverIP** ="127.0.0.1"
Declaration common for client and server (op.codes and coding/decoding functions)

int **servPORT** =5205
Teoretically it could be any above 1024.

Detailed Description

Definition at line 18 of file gameClient.java.

Member Function Documentation

void gameClient.beginInteraction (GameObject[] *table*, String[] *objectAndActionsNames*)

Handling for getting the avatar in contact with the game object.

Definition at line 268 of file gameClient.java.

void gameClient.clientEvent (Client *client*) [package]

ClientEvent message is generated when the server sends data to an existing client.

This is alternative, asynchronous way to read messages from the server.

Definition at line 497 of file gameClient.java.

void gameClient.clientGameDraw ()

This function performs communication with server & visualisation of the game world.

Definition at line 391 of file gameClient.java.

void gameClient.colorChanged (GameObject[] *table*, String *name*, String *hexColor*)

Handle changes in colors of any game object.

Definition at line 211 of file gameClient.java.

String gameClient.decodeHELLO (String *msgHello*)

It decodes handshake.

Returns

: Name of client or name of game implemented on server

Definition at line 1015 of file gameClient.java.

String gameClient.decodeInfos (String *msgInfos*, String[] *infos*)

It decodes 1-9 infos message.

Dimension of the array must be proper

Returns

: object name, and fill the infos

Definition at line 1084 of file gameClient.java.

String [] gameClient.decodeObjectMng (String *msg*)

It decodes message of objects types management - decoding.

Returns

: array of strings with "del" action and objectName or "new" action, type name and object name. Other actions are possible in the future.

Definition at line 1266 of file gameClient.java.

String gameClient.decodeOptAndInf (String *msg*)

Decode one string message.

Returns

: All characters between a message header (OpCode+SPC) and a final pair (SPC+EOR)

Definition at line 1041 of file gameClient.java.

String gameClient.decodePosition (String *msgPosition*, float[] *coordinates*)

It decodes 1-9 dimensional positioning message.

Dimension of the array must be proper

Returns

: name of object and also fill coordinates.

Definition at line 1223 of file gameClient.java.

float gameClient.decodeTouch (String *msg*, String[] *infos*)

It decodes touch message.

Returns

: distance The infos will be filled with name of touched object and up to 9 possible actions (or more - NOT TESTED!)

Definition at line 1142 of file gameClient.java.

void gameClient.disconnectEvent (Client *client*)

ClientEvent message is generated when a client disconnects.

Definition at line 479 of file gameClient.java.

void gameClient.draw ()

A function that is triggered many times per second, carrying out the main tasks of the client.

First it shows the intro for the first few frames.
Then it is trying to establish connection with server
Finally it realising communication with server & visualisation When connection with server broke, it go back to establishing connection.

Definition at line 57 of file gameClient.java.

void gameClient.drawStartUpInfo ()

Intro view placeholder ;-)

Definition at line 78 of file gameClient.java.

void gameClient.drawTryConnect ()

Attempting to connect and initial communication.

Definition at line 124 of file gameClient.java.

int gameClient.findCollision (GameObject[] *table*, int *indexOfMoved*, int *startIndex*, boolean *withZ*)

It finds the index of the first collided object 'indexOfMoved' is the index of the object for which we check for collisions.

The first time 'startIndex' should be 0, but thanks to this parameter you can continue searching for more collisions. Whem withZ parameter is false, only 2D distance is calculated.

Returns

: index or -1 if nothing collidend with object reffered by indexOfMoved

Definition at line 775 of file gameClient.java.

void gameClient.finishInteraction (GameObject[] *table*, String *objectName*)

Handling for getting out of the avatar's contact with the game object.

Definition at line 293 of file gameClient.java.

void gameClient.implementObjectMenagement (String[] *cmd*)

Simple object type management.

Definition at line 169 of file gameClient.java.

void gameClient.interpretMessage (String *msg*)

Handling interpretation of messages from server.

Definition at line 317 of file gameClient.java.

void gameClient.keyPressed ()

gameClient - keyboard input & other asynchronic events

Keyboard events - mostly control of the avatar

Definition at line 428 of file gameClient.java.

void gameClient.loadSettings ()

Loads the player's name from the file "player.txt" But may do more...

Definition at line 147 of file gameClient.java.

int gameClient.localiseByName (GameObject[] *table*, String *name*)

Determines the index of the object with the specified proper name in an array of objects or players.

Simple implementation for now, but you can change into dictionary or something after that.

Returns

: index or -1 if not found.

Definition at line 749 of file gameClient.java.

static void gameClient.main (String[] *passedArgs*)`[static]`

Definition at line 1291 of file gameClient.java.

String gameClient.makeAllTypeInfo (GameObject[] *table*)

Prepares information about the types and names of all objects on the game board.

Mainly needed when a new client connects.

Returns

: Ready to send list of type infos messages for whole table

Definition at line 807 of file gameClient.java.

GameObject gameClient.makeGameObject (String *name*, float *X*, float *Y*, float *Z*, float *R*)

Very simple placeholder for use types dictionary.

Definition at line 184 of file gameClient.java.

void gameClient.performAction (ActiveGameObject *subject*, String *action*, GameObject *object*)

Actions placeholder.

Definition at line 934 of file gameClient.java.

void gameClient.playerAction (String *action*, Player *player*)

The actions of agents and players in the game are defined by names in the protocol, thanks to which their set is expandable.

Definition at line 922 of file gameClient.java.

boolean gameClient.playerMove (String *dir*, Player *player*)

Moves allowed for the player.

Intended to be used on the server side.

Returns

: false if dir string contains unknow command, otherwise true

Definition at line 904 of file gameClient.java.

void gameClient.positionChanged (GameObject[] *table*, String *name*, float[] *inpos*)

Handle changes in position of any game object.

Definition at line 226 of file gameClient.java.

void gameClient.removeObject (GameObject[] *table*, String *name*)

It removes object reffered by name from the table.

The object may remains somowhere else, so no any destruction will be performed.

Definition at line 763 of file gameClient.java.

String gameClient.sayHELLO (String *myName*)

It composes server-client handshake.

Returns

message PREPARED to send.

Definition at line 1007 of file gameClient.java.

String gameClient.sayObjectRemove (String *objectName*)

For objects types management - object removing from the game world.

Returns

: message PREPARED to send

Definition at line 1255 of file gameClient.java.

String gameClient.sayObjectType (String *type*, String *objectName*)

For objects types management - type of object.

Returns

: message PREPARED to send

Definition at line 1245 of file gameClient.java.

String gameClient.sayOptAndInf (char *opCode*, String *inf*)

It composes simple string info - **Opcs.SPC** inside 'inf' is allowed.

Returns

: message PREPARED to send.

Definition at line 1034 of file gameClient.java.

String gameClient.sayOptAndInfos (char *opCode*, String *objName*, String *info*)

Compose one string info - SPC inside info is NOT allowed.

Returns

: message PREPARED to send.

Definition at line 1051 of file gameClient.java.

String gameClient.sayOptAndInfos (char *opCode*, String *objName*, String *info1*, String *info2*)

Compose many(=2) string info - SPC inside infos is NOT allowed.

Returns

: message PREPARED to send.

Definition at line 1061 of file gameClient.java.

String gameClient.sayOptAndInfos (char *opCode*, String *objName*, String *info1*, String *info2*, String *info3*)

Compose many(=3) string info - SPC inside infos is NOT allowed.

Returns

: message PREPARED to send.

Definition at line 1072 of file gameClient.java.

String gameClient.sayOptCode (char *optCode*)

It composes one OPC info.

For which, when recieved, only charAt(0) is important.

Returns

: message PREPARED to send.

Definition at line 1027 of file gameClient.java.

String gameClient.sayPosition (char *EUCorPOL*, String *objName*, float *coord*)

It composes message about object position (1 dimension) E1 OName Data @ - Euclidean position float(X) P1 OName Data @ - Polar position float(Alfa +-180)

Returns

: message PREPARED to send

Definition at line 1165 of file gameClient.java.

String gameClient.sayPosition (char *EUCorPOL*, String *objName*, float *coord1*, float *coord2*)

It composes message about object position (2 dimensions)

E2 OName Data*2 @ - Euclidean position float(X) float(Y) P2 OName Data*2 @ - Polar position float(Alfa +-180) float(DISTANCE) OName == object identification or name of player or 'Y'.

Returns

: message PREPARED to send

Definition at line 1178 of file gameClient.java.

String gameClient.sayPosition (char *EUCorPOL*, String *objName*, float *coord1*, float *coord2*, float *coord3*)

It composes message about object position (3 dimensions) E3 OName Data*3 @ - Euclidean position float(X) float(Y) float(H) P3 OName Data*3 @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) OName == object identification or name of player or 'Y'.

Returns

: message PREPARED to send

Definition at line 1192 of file gameClient.java.

String gameClient.sayPosition (char *EUCorPOL*, String *objName*, float[] *coordinates*)

It composes message about object position (1-9 dimensions) En OName Data*n @ - Euclidean position float(X) float(Y) float(H) "class name of object or name of player" Pn OName Data*n @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) "class name of object or name of player" OName == object identification or name of player or 'Y'.

Returns

: message PREPARED to send

Definition at line 1207 of file gameClient.java.

String gameClient.sayTouch (String *nameOfTouched*, float *distance*, String *action1*, String *action2*)

It constructs touch message with two possible actions.

Returns

: message PREPARED to send

Definition at line 1112 of file gameClient.java.

String gameClient.sayTouch (String *nameOfTouched*, float *distance*, String *actionDef*)

It constructs touch message with only one possible action.

Returns

: message PREPARED to send.

Definition at line 1101 of file gameClient.java.

String gameClient.sayTouch (String *nameOfTouched*, float *distance*, String[] *actions*)

It constructs touch message with many possible actions.

Returns

: message PREPARED to send

Definition at line 1124 of file gameClient.java.

void gameClient.serverEvent (Server *server*, Client *client*)

Event handler called on server when a client connects to server.

Definition at line 488 of file gameClient.java.

void gameClient.settings ()

Definition at line 1290 of file gameClient.java.

void gameClient.setup ()

Startup of a game client. Still not connected after that.

Definition at line 38 of file gameClient.java.

void gameClient.stateChanged (GameObject[] *table*, String *objectName*, String *field*, String *val*)

Handle changes in states of agents/objects.

Definition at line 247 of file gameClient.java.

void gameClient.visualisationChanged (GameObject[] *table*, String *name*, String *vtype*)

Handle changes in visualisation type of any game object.

Definition at line 198 of file gameClient.java.

void gameClient.visualise2D (float *startX*, float *startY*, float *width*, float *height*)

Simplest flat map visualisation of game board.

Definition at line 818 of file gameClient.java.

void gameClient.whenConnectedToServer ()

Initial courtesy exchange with the server.

Definition at line 88 of file gameClient.java.

Member Data Documentation

final int gameClient.ACTRAD_MSK = unbinary("010000000") [package]

object changed its radius of activity (ex. go to sleep);

Definition at line 530 of file gameClient.java.

final int gameClient.ALL_CHNG_MSK = MOVED_MSK | VISUAL_MSK | COLOR_MSK | STATE_MSK [package]

All initial changes.

Definition at line 536 of file gameClient.java.

String [] gameClient.avatars ={".","^v^" ,"o^o","@","&","ðŸ˜ƒ","ðŸ˜Ⓔ"} [package]

peoples...

Definition at line 520 of file gameClient.java.

final int gameClient.COLOR_MSK = unbinary("000001000") [package]

object changed its colors

Definition at line 526 of file gameClient.java.

int gameClient.DEBUG =1 [package]

Client for **gameServer** - **setup()** & **draw()** SOURCE FILE.

Losely based on: --> `https://forum.processing.org/one/topic/how-do-i-send-data-to-only-one-client-using-the-network-library.html`

Definition at line 28 of file gameClient.java.

int gameClient.DEF_FRAME_RATE =60[package]

Definition at line 31 of file gameClient.java.

GameObject [] gameClient.gameWorld =null[package]

MAIN ARRAY OF GameObjects.
Definition at line 952 of file gameClient.java.

final int gameClient.HPOINT_MSK = unbinary("000010000")[package]

object changed its hp state (most frequently changed state)
Definition at line 527 of file gameClient.java.

int gameClient.indexOfMe =-1[package]

Index of object visualising the client or the server supervisor.
Definition at line 516 of file gameClient.java.

int gameClient.INFO_LEVEL =1 | SCORE_MSK[package]

Visualisation with information about objects (name & score by default)
Definition at line 539 of file gameClient.java.

float gameClient.initialMaxX =100[package]

Initial horizontal size of game "board".
Definition at line 514 of file gameClient.java.

float gameClient.initialMaxY =100[package]

Initial vertical size of game "board".
Definition at line 515 of file gameClient.java.

int gameClient.initialSizeOfMainArray =30[package]

Game classes and its basic behaviours.
Initial number of @GameObjects in @gameWorld
Definition at line 513 of file gameClient.java.

float [] gameClient.inparr1 =new float[1][package]

Definition at line 309 of file gameClient.java.

float [] gameClient.inparr2 =new float[2][package]

Definition at line 310 of file gameClient.java.

float [] gameClient.inparr3 =new float[3][package]

Definition at line 311 of file gameClient.java.

String [] gameClient.instr1 =new String[1][package]

Definition at line 312 of file gameClient.java.

String [] gameClient.instr2 =new String[2][package]

Definition at line 313 of file gameClient.java.

String [] gameClient.instr3 =new String[3][package]

Definition at line 314 of file gameClient.java.

int gameClient.INTRO_FRAMES =3[package]

Definition at line 30 of file gameClient.java.

StringDict gameClient.inventory =new StringDict()[package]

gameClient - more communication logic
Definition at line 166 of file gameClient.java.

boolean gameClient.KEEP_ASPECT =true[package]

Visualisation with proportional aspect ratio.
Definition at line 541 of file gameClient.java.

final int gameClient.MOVED_MSK = unbinary("000000010")[package]

object was moved (0x1)
Definition at line 524 of file gameClient.java.

Client gameClient.myClient =null[package]

Definition at line 35 of file gameClient.java.

final int gameClient.PASRAD_MSK = unbinary("001000000")[package]

object changed its passive radius (ex. grow);

Definition at line 529 of file gameClient.java.

String [] gameClient.plants = {"_","O","...\nl","_\\|/_\nl ","|/",":","â˜˙¸˙"}[package]

plants...
Definition at line 519 of file gameClient.java.

String gameClient.playerName =""[package]

Definition at line 33 of file gameClient.java.

final int gameClient.SCORE_MSK = unbinary("000100000")[package]

object changed its score (for players it is most frequently changed state)
Definition at line 528 of file gameClient.java.

String gameClient.serverIP ="127.0.0.1"[package]

Declaration common for client and server (op.codes and coding/decoding functions)
localhost
Definition at line 965 of file gameClient.java.

int gameClient.servPORT =5205[package]

Teoretically it could be any above 1024.
Definition at line 966 of file gameClient.java.

final int gameClient.STATE_MSK = HPOINT_MSK | SCORE_MSK | PASRAD_MSK | ACTRAD_MSK[package]

object changed its states
Definition at line 531 of file gameClient.java.

final int gameClient.TOUCH_MSK = unbinary("1000000000000000")[package]

To visualize the interaction between background objects.
16bits
Definition at line 534 of file gameClient.java.

int gameClient.VIEWMESG =0[package]

Definition at line 29 of file gameClient.java.

boolean gameClient.VIS_MIN_MAX =true[package]

Visualisation with min/max value.

Definition at line 540 of file gameClient.java.

final int gameClient.VISSWITH = unbinary("000000001")[package]

object is invisible (but in info level name is visible)

Definition at line 523 of file gameClient.java.

final int gameClient.VISUAL_MSK = unbinary("000000100")[package]

object changed its type of view

Definition at line 525 of file gameClient.java.

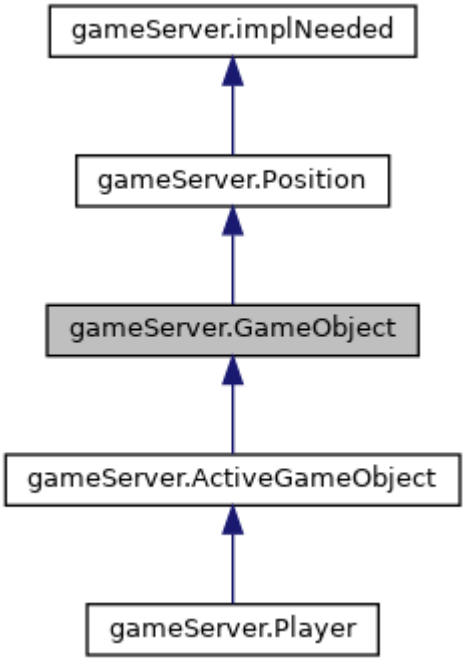
The documentation for this class was generated from the following file:

src.java/**gameClient.java**

gameServer.GameObject Class Reference

Representation of simple game object.

Inheritance diagram for gameServer.GameObject:



IMAGE

Public Member Functions

String[] **abilities** ()
String[] **possibilities** ()
boolean **setState** (String field, String val)

Interface for changing the state of the game object over the network (mostly)

String **sayState** ()

The function creates a message list (for network streaming) from those object state elements that have change flag sets.

String **info** (int level)

It can make string info about object.

Package Functions

GameObject (String iniName, float iniX, float iniY, float iniZ)

constructor

Package Attributes

String **name**
String **visual** ="?"
int **foreground** =0xff00ff00
float **hpoints** =1
float[] **distances** =null

float **passiveRadius** =1

Detailed Description

Representation of simple game object.

Definition at line 303 of file gameServer.java.

Constructor & Destructor Documentation

gameServer.GameObject.GameObject (String *iniName*, float *iniX*, float *iniY*, float *iniZ*) [package]

constructor

Definition at line 317 of file gameServer.java.

Member Function Documentation

String [] gameServer.GameObject.abilities ()

Returns

: List of actions that this object can performed

Definition at line 322 of file gameServer.java.

String gameServer.GameObject.info (int *level*)

It can make string info about object.

'level' is level of details, when 0 means "name only".

Reimplemented in **gameServer.Player** (*p.177*).

Definition at line 367 of file gameServer.java.

String [] gameServer.GameObject.possibilities ()

Returns

: List of actions that can be performed on this object

Definition at line 325 of file gameServer.java.

String gameServer.GameObject.sayState ()

The function creates a message list (for network streaming) from those object state elements that have change flag sets.

Returns

: Ready to send list of all changes made on object (based on flags)

Reimplemented in **gameServer.Player** (*p.177*), and **gameServer.ActiveGameObject** (*p.25*).

Definition at line 349 of file gameServer.java.

boolean gameServer.GameObject.setState (String *field*, String *val*)

Interface for changing the state of the game object over the network (mostly)

Returns

: true if field is found

Reimplemented in **gameServer.Player** (*p.177*), and **gameServer.ActiveGameObject** (*p.25*).

Definition at line 330 of file gameServer.java.

Member Data Documentation

float [] gameServer.GameObject.distances =null [package]

Definition at line 311 of file gameServer.java.

int gameServer.GameObject.foreground =0xff00ff00 [package]

Definition at line 307 of file gameServer.java.

float gameServer.GameObject.hpoints =1 [package]

Definition at line 309 of file gameServer.java.

String gameServer.GameObject.name [package]

Definition at line 305 of file gameServer.java.

float gameServer.GameObject.passiveRadius =1 [package]

Definition at line 314 of file gameServer.java.

String gameServer.GameObject.visual ="?" [package]

Definition at line 306 of file gameServer.java.

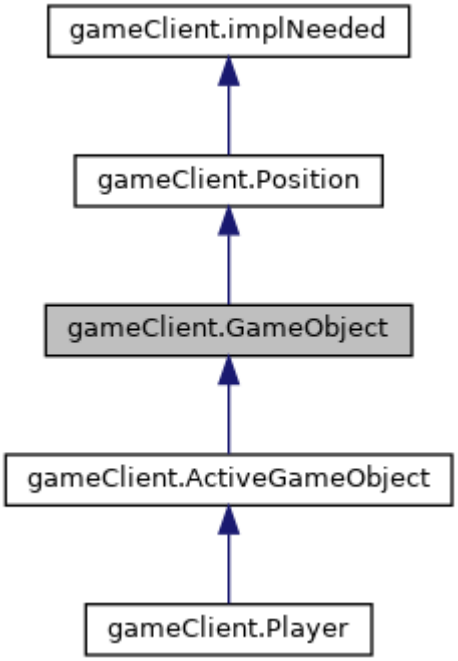
The documentation for this class was generated from the following file:

src.java/**gameServer.java**

gameClient.GameObject Class Reference

Representation of simple game object.

Inheritance diagram for gameClient.GameObject:



IMAGE

Public Member Functions

String[] **abilities** ()
String[] **possibilities** ()
boolean **setState** (String field, String val)

Interface for changing the state of the game object over the network (mostly)

String **sayState** ()

The function creates a message list (for network streaming) from those object state elements that have change flag sets.

String **info** (int level)

It can make string info about object.

Package Functions

GameObject (String iniName, float iniX, float iniY, float iniZ)

constructor

Package Attributes

String **name**

Each object has an individual identifier necessary for communication. Better short.

String **visual** ="?"

Text representation of the visualization. The unicode character or the name of an external file.

int **foreground** =0xff00ff00
float **hpoints** =1
Health points.

float[] **distances** =null
Array of distances to other objects.

float **passiveRadius** =1
Radius of passive interaction.

Detailed Description

Representation of simple game object.

Definition at line 585 of file gameClient.java.

Constructor & Destructor Documentation

gameClient.GameObject.GameObject (String *iniName*, float *iniX*, float *iniY*, float *iniZ*) [package]

constructor

Definition at line 599 of file gameClient.java.

Member Function Documentation

String [] gameClient.GameObject.abilities ()

Returns

: List of actions that this object can performed

Definition at line 604 of file gameClient.java.

String gameClient.GameObject.info (int *level*)

It can make string info about object.

'level' is level of details, when 0 means "name only".

Reimplemented in **gameClient.Player** (*p.174*).

Definition at line 649 of file gameClient.java.

String [] gameClient.GameObject.possibilities ()

Returns

: List of actions that can be performed on this object

Definition at line 607 of file gameClient.java.

String gameClient.GameObject.sayState ()

The function creates a message list (for network streaming) from those object state elements that have change flag sets.

Returns

: Ready to send list of all changes made on object (based on flags)

Reimplemented in **gameClient.Player** (*p.174*), and **gameClient.ActiveGameObject** (*p.27*).

Definition at line 631 of file gameClient.java.

boolean gameClient.GameObject.setState (String *field*, String *val*)

Interface for changing the state of the game object over the network (mostly)

Returns

: true if field is found

Reimplemented in **gameClient.Player** (*p.174*), and **gameClient.ActiveGameObject** (*p.27*).

Definition at line 612 of file gameClient.java.

Member Data Documentation

float [] gameClient.GameObject.distances =null [package]

Array of distances to other objects.

Not always in use!

Definition at line 593 of file gameClient.java.

int gameClient.GameObject.foreground =0xff00ff00 [package]

Definition at line 589 of file gameClient.java.

float gameClient.GameObject.hpoints =1 [package]

Health points.

Definition at line 591 of file gameClient.java.

String gameClient.GameObject.name [package]

Each object has an individual identifier necessary for communication. Better short.

Definition at line 587 of file gameClient.java.

float gameClient.GameObject.passiveRadius =1[package]

Radius of passive interaction.

Definition at line 596 of file gameClient.java.

String gameClient.GameObject.visual ="?"[package]

Text representation of the visualization. The unicode character or the name of an external file.

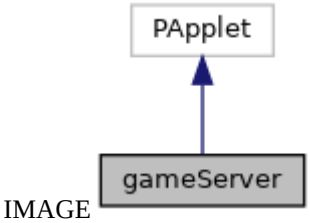
Definition at line 588 of file gameClient.java.

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

src.java/**gameClient.java**

gameServer Class Reference

Inheritance diagram for gameServer:



IMAGE

Classes

class **ActiveGameObject**

class **GameObject**

Representation of simple game object.

class **implNeeded**

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

class **Opcs**

Protocol dictionary ("opcodes" etc.)

class **Player**

Representation of generic player.

class **Position**

Representation of 3D position in the game world However, the value of Z is not always used.

Public Member Functions

void **setup** ()

Startup of a game server.

void **draw** ()

This function is called many times per second.

void **serverWaitingDraw** ()

Waiting view placeholder ;-)

void **confirmClient** (Client newClient, **Player** player)

Confirm client registration and send correct current name.

void **whenClientConnected** (Client newClient, String playerName)

This is stuff that should be done,

when new client was connected.

void **keyPressed** ()

***gameServer** (dummy) keyboard input & other asynchronic events*

void **serverEvent** (Server me, Client newClient)

Event handler called when a client connects to server.

void **disconnectEvent** (Client someClient)
ClientEvent message is generated when a client disconnects from server.

int **localiseByName** (**GameObject**[] table, String name)
Determines the index of the object with the specified proper name in an array of objects or players.

void **removeObject** (**GameObject**[] table, String name)
It removes object reffered by name from the table.

int **findCollision** (**GameObject**[] table, int indexOfMoved, int startIndex, boolean withZ)
It finds the index of the first collided object 'indexOfMoved' is the index of the object for which we check for collisions.

String **makeAllTypeInfo** (**GameObject**[] table)
Prepares information about the types and names of all objects on the game board.

void **visualise2D** (float startX, float startY, float width, float height)
Simplest flat map visualisation of game board.

boolean **playerMove** (String dir, **Player** player)
Moves allowed for the player.

void **playerAction** (String action, **Player** player)
The actions of agents and players in the game are defined by names in the protocol, thanks to which their set is expandable.

void **performAction** (**ActiveGameObject** subject, String action, **GameObject** object)
Actions placeholder.

String **sayHELLO** (String myName)
It composes server-client handshake.

String **decodeHELLO** (String msgHello)
It decodes handshake.

String **sayOptCode** (char optCode)
It composes one OPC info.

String **sayOptAndInf** (char opCode, String inf)
*It composes simple string info - **Opcs.SPC** inside 'inf' is allowed.*

String **decodeOptAndInf** (String msg)
Decode one string message.

String **sayOptAndInfos** (char opCode, String objName, String info)

Compose one string info - SPC inside info is NOT allowed.

String **sayOptAndInfos** (char opCode, String objName, String info1, String info2)
Compose many(=2) string info - SPC inside infos is NOT allowed.

String **sayOptAndInfos** (char opCode, String objName, String info1, String info2, String info3)
Compose many(=3) string info - SPC inside infos is NOT allowed.

String **decodeInfos** (String msgInfos, String[] infos)
It decodes 1-9 infos message.

String **sayTouch** (String nameOfTouched, float distance, String actionDef)
It constructs touch message with only one possible action.

String **sayTouch** (String nameOfTouched, float distance, String action1, String action2)
It constructs touch message with two possible actions.

String **sayTouch** (String nameOfTouched, float distance, String[] actions)
It constructs touch message with many possible actions.

float **decodeTouch** (String msg, String[] infos)
It decodes touch message.

String **sayPosition** (char EUCorPOL, String objName, float coord)
It composes message about object position (1 dimension) E1 OName Data @ - Euclidean position float(X) P1 OName Data @ - Polar position float(Alfa +-180)

String **sayPosition** (char EUCorPOL, String objName, float coord1, float coord2)
It composes message about object position (2 dimensions)

*E2 OName Data*2 @ - Euclidean position float(X) float(Y) P2 OName Data*2 @ - Polar position float(Alfa +-180) float(DISTANCE) OName == object identification or name of player or 'Y'.*

String **sayPosition** (char EUCorPOL, String objName, float coord1, float coord2, float coord3)
*It composes message about object position (3 dimensions) E3 OName Data*3 @ - Euclidean position float(X) float(Y) float(H) P3 OName Data*3 @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) OName == object identification or name of player or 'Y'.*

String **sayPosition** (char EUCorPOL, String objName, float[] coordinates)
*It composes message about object position (1-9 dimensions) En OName Data*n @ - Euclidean position float(X) float(Y) float(H) "class name of object or name of player" Pn OName Data*n @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) "class name of object or name of player" OName == object identification or name of player or 'Y'.*

String **decodePosition** (String msgPosition, float[] coordinates)
It decodes 1-9 dimensional positioning message.

String **sayObjectType** (String type, String objectName)
For objects types management - type of object.

String **sayObjectRemove** (String objectName)
For objects types management - object removing from the game world.

String[] **decodeObjectMng** (String msg)
It decodes message of objects types management - decoding.

void **sendWholeUpdate** ()
This function sends a full game board update to all clients.

void **sendUpdateOfChangedAgents** ()
This function sends all recent changes to all clients.

void **readMessagesFromPlayers** ()
It reads pending messages from all players.

void **initialiseGame** ()
initialisation of game world

void **stepOfGameMechanics** ()
Do in game world all, what is independent of players actions.

void **checkCollisions** ()
Checks for collisions and sends information about them to clients In the example game, only the players move, so only they can cause collisions.

void **serverGameDraw** ()
Server real jobs during game:

void **interpretMessage** (String msg, **Player** player)
This function interprets message from a particular player.

void **settings** ()

Static Public Member Functions

static void **main** (String[] passedArgs)

Package Functions

void **clientEvent** (Client client)
ClientEvent handler is called when the server recives data from an existing client.

Package Attributes

int **DEBUG** =0
*Server for gameClients - **setup()** & **draw()** SOURCE FILE.*

Server **mainServer**
Object representing server's TCP/IP COMMUNICATION.

Player[] **players** = new **Player**[0]
The array of clients (players)

int **initialSizeOfMainArray** =30
Game classes and its basic behaviours.

float **initialMaxX** =100
Initial horizontal size of game "board".

float **initialMaxY** =100
Initial vertical size of game "board".

int **indexOfMe** =-1
Index of object visualising the client or the server supervisor.

String[] **plants** = {"_","O","...\nI","_\\|/_\nI ","|/",":","â˜˙ï¸"}
plants...

String[] **avatars** ={".","^v^" ,"o^o","@","&","ðŸ˜ƒ","ðŸ˜⊙"}
peoples...

final int **VISSWITH** = unbinary("000000001")
object is invisible (but in info level name is visible)

final int **MOVED_MSK** = unbinary("000000010")
object was moved (0x1)

final int **VISUAL_MSK** = unbinary("000000100")
object changed its type of view

final int **COLOR_MSK** = unbinary("000001000")
object changed its colors

final int **HPOINT_MSK** = unbinary("000010000")
object changed its hp state (most frequently changed state)

final int **SCORE_MSK** = unbinary("000100000")
object changed its score (for players it is most frequently changed state)

final int **PASRAD_MSK** = unbinary("001000000")
object changed its passive radius (ex. grow);

final int **ACTRAD_MSK** = unbinary("010000000")

object changed its radius of activity (ex. go to sleep);

final int **STATE_MSK** = **HPOINT_MSK** | **SCORE_MSK** | **PASRAD_MSK** | **ACTRAD_MSK**
object changed its states

final int **TOUCH_MSK** = unbinary("1000000000000000")
To visualize the interaction between background objects.

final int **ALL_CHNG_MSK** = **MOVED_MSK** | **VISUAL_MSK** | **COLOR_MSK** | **STATE_MSK**
All initial changes.

int **INFO_LEVEL** =1 | **SCORE_MSK**
Visualisation with information about objects (name & score by default)

boolean **VIS_MIN_MAX** =true
Visualisation with min/max value.

boolean **KEEP_ASPECT** =true
Visualisation with proportional aspect ratio.

GameObject[] **gameWorld** =null
MAIN ARRAY OF GameObjects.

String **serverIP** ="127.0.0.1"
Declaration common for client and server (op.codes and coding/decoding functions)

int **servPORT** =5205
Teoretically it could be any above 1024.

int **Xmargin** =0
***gameServer** - more communication & game logic*

boolean **wholeUpdateRequested** =false
Information about a client requesting information about the entire scene.

Detailed Description

Definition at line 19 of file gameServer.java.

Member Function Documentation

void gameServer.checkCollisions ()

Checks for collisions and sends information about them to clients In the example game, only the players move, so only they can cause collisions.

Definition at line 1120 of file gameServer.java.

void gameServer.clientEvent (Client *client*) [package]

ClientEvent handler is called when the server recives data from an existing client.

This is alternative, asynchronous way to read messages from clients.

Definition at line 215 of file gameServer.java.

void gameServer.confirmClient (Client *newClient*, Player *player*)

Confirm client registration and send correct current name.

Definition at line 85 of file gameServer.java.

String gameServer.decodeHELLO (String *msgHello*)

It decodes handshake.

Returns

: Name of client or name of game implemented on server

Definition at line 733 of file gameServer.java.

String gameServer.decodeInfos (String *msgInfos*, String[] *infos*)

It decodes 1-9 infos message.

Dimension of the array must be proper

Returns

: object name, and fill the infos

Definition at line 802 of file gameServer.java.

String [] gameServer.decodeObjectMng (String *msg*)

It decodes message of objects types management - decoding.

Returns

: array of strings with "del" action and objectName or "new" action, type name and object name. Other actions are possible in the future.

Definition at line 984 of file gameServer.java.

String gameServer.decodeOptAndInf (String *msg*)

Decode one string message.

Returns

: All characters between a message header (OpCode+SPC) and a final pair (SPC+EOR)

Definition at line 759 of file gameServer.java.

String gameServer.decodePosition (String *msgPosition*, float[] *coordinates*)

It decodes 1-9 dimensional positioning message.

Dimension of the array must be proper

Returns

: name of object and also fill coordinates.

Definition at line 941 of file gameServer.java.

float gameServer.decodeTouch (String *msg*, String[] *infos*)

It decodes touch message.

Returns

: distance The infos will be filled with name of touched object and up to 9 possible actions (or more - NOT TESTED!)

Definition at line 860 of file gameServer.java.

void gameServer.disconnectEvent (Client *someClient*)

ClientEvent message is generated when a client disconnects from server.

Definition at line 191 of file gameServer.java.

void gameServer.draw ()

This function is called many times per second.

Real work of server depends of current state of clients connections.

Definition at line 62 of file gameServer.java.

int gameServer.findCollision (GameObject[] *table*, int *indexOfMoved*, int *startIndex*, boolean *withZ*)

It finds the index of the first collided object 'indexOfMoved' is the index of the object for which we check for collisions.

The first time 'startIndex' should be 0, but thanks to this parameter you can continue searching for more collisions. Whem withZ parameter is false, only 2D distance is calculated.

Returns

: index or -1 if nothing collidend with object reffered by indexOfMoved

Definition at line 493 of file gameServer.java.

void gameServer.initialiseGame ()

initialisation of game world

Definition at line 1087 of file gameServer.java.

void gameServer.interpretMessage (String *msg*, Player *player*)

This function interprets message from a particular player.

Definition at line 1198 of file gameServer.java.

void gameServer.keyPressed ()

gameServer (dummy) keyboard input & other asynchronic events

Keyboard handler for the server. In most cases not useable. However, it protects the server against accidental stopping with the ESC key

Definition at line 158 of file gameServer.java.

int gameServer.localiseByName (GameObject[] *table*, String *name*)

Determines the index of the object with the specified proper name in an array of objects or players.

Simple implementation for now, but you can change into dictionary or something after that.

Returns

: index or -1 if not found.

Definition at line 467 of file gameServer.java.

static void gameServer.main (String[] *passedArgs*)`[static]`

Definition at line 1229 of file gameServer.java.

String gameServer.makeAllTypeInfo (GameObject[] *table*)

Prepares information about the types and names of all objects on the game board.

Mainly needed when a new client connects.

Returns

: Ready to send list of type infos messages for whole table

Definition at line 525 of file gameServer.java.

void gameServer.performAction (ActiveGameObject *subject*, String *action*, GameObject *object*)

Actions placeholder.

Definition at line 652 of file gameServer.java.

void gameServer.playerAction (String *action*, Player *player*)

The actions of agents and players in the game are defined by names in the protocol, thanks to which their set is expandable.

Definition at line 640 of file gameServer.java.

boolean gameServer.playerMove (String *dir*, Player *player*)

Moves allowed for the player.

Intended to be used on the server side.

Returns

: false if dir string contains unknow command, otherwise true

Definition at line 622 of file gameServer.java.

void gameServer.readMessagesFromPlayers ()

It reads pending messages from all players.

Definition at line 1062 of file gameServer.java.

void gameServer.removeObject (GameObject[] *table*, String *name*)

It removes object reffered by name from the table.

The object may remains somowhere else, so no any destruction will be performed.

Definition at line 481 of file gameServer.java.

String gameServer.sayHELLO (String *myName*)

It composes server-client handshake.

Returns

message PREPARED to send.

Definition at line 725 of file gameServer.java.

String gameServer.sayObjectRemove (String *objectName*)

For objects types management - object removing from the game world.

Returns

: message PREPARED to send

Definition at line 973 of file gameServer.java.

String gameServer.sayObjectType (String *type*, String *objectName*)

For objects types management - type of object.

Returns

: message PREPARED to send

Definition at line 963 of file gameServer.java.

String gameServer.sayOptAndInf (char *opCode*, String *inf*)

It composes simple string info - **Opcs.SPC** inside 'inf' is allowed.

Returns

: message PREPARED to send.

Definition at line 752 of file gameServer.java.

String gameServer.sayOptAndInfos (char *opCode*, String *objName*, String *info*)

Compose one string info - SPC inside info is NOT allowed.

Returns

: message PREPARED to send.

Definition at line 769 of file gameServer.java.

String gameServer.sayOptAndInfos (char *opCode*, String *objName*, String *info1*, String *info2*)

Compose many(=2) string info - SPC inside infos is NOT allowed.

Returns

: message PREPARED to send.

Definition at line 779 of file gameServer.java.

String gameServer.sayOptAndInfos (char *opCode*, String *objName*, String *info1*, String *info2*, String *info3*)

Compose many(=3) string info - SPC inside infos is NOT allowed.

Returns

: message PREPARED to send.

Definition at line 790 of file gameServer.java.

String gameServer.sayOptCode (char *optCode*)

It composes one OPC info.

For which, when recieved, only charAt(0) is important.

Returns

: message PREPARED to send.

Definition at line 745 of file gameServer.java.

String gameServer.sayPosition (char *EUCorPOL*, String *objName*, float *coord*)

It composes message about object position (1 dimension) E1 OName Data @ - Euclidean position float(X) P1 OName Data @ - Polar position float(Alfa +-180)

Returns

: message PREPARED to send

Definition at line 883 of file gameServer.java.

String gameServer.sayPosition (char *EUCorPOL*, String *objName*, float *coord1*, float *coord2*)

It composes message about object position (2 dimensions)

E2 OName Data*2 @ - Euclidean position float(X) float(Y) P2 OName Data*2 @ - Polar position float(Alfa +-180) float(DISTANCE) OName == object identification or name of player or 'Y'.

Returns

: message PREPARED to send

Definition at line 896 of file gameServer.java.

String gameServer.sayPosition (char *EUCorPOL*, String *objName*, float *coord1*, float *coord2*, float *coord3*)

It composes message about object position (3 dimensions) E3 OName Data*3 @ - Euclidean position float(X) float(Y) float(H) P3 OName Data*3 @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) OName == object identification or name of player or 'Y'.

Returns

: message PREPARED to send

Definition at line 910 of file gameServer.java.

String gameServer.sayPosition (char *EUCorPOL*, String *objName*, float[] *coordinates*)

It composes message about object position (1-9 dimensions) En OName Data*n @ - Euclidean position float(X) float(Y) float(H) "class name of object or name of player" Pn OName Data*n @ - Polar position float(Alfa +-180) float(DISTANCE) float(Beta +-180) "class name of object or name of player" OName == object identification or name of player or 'Y'.

Returns

: message PREPARED to send

Definition at line 925 of file gameServer.java.

String gameServer.sayTouch (String *nameOfTouched*, float *distance*, String *action1*, String *action2*)

It constructs touch message with two possible actions.

Returns

: message PREPARED to send

Definition at line 830 of file gameServer.java.

String gameServer.sayTouch (String *nameOfTouched*, float *distance*, String *actionDef*)

It constructs touch message with only one possible action.

Returns

: message PREPARED to send.

Definition at line 819 of file gameServer.java.

String gameServer.sayTouch (String *nameOfTouched*, float *distance*, String[] *actions*)

It constructs touch message with many possible actions.

Returns

: message PREPARED to send

Definition at line 842 of file gameServer.java.

void gameServer.sendUpdateOfChangedAgents ()

This function sends all recent changes to all clients.

Definition at line 1039 of file gameServer.java.

void gameServer.sendWholeUpdate ()

This function sends a full game board update to all clients.

Definition at line 1016 of file gameServer.java.

void gameServer.serverEvent (Server *me*, Client *newClient*)

Event handler called when a client connects to server.

Definition at line 169 of file gameServer.java.

void gameServer.serverGameDraw ()

Server real jobs during game:

Displays how many clients have connected to the server
Visualises the current state of the game
Does what players decided to do (if doable)
If any client requested update, send whole update to clients!
If not, send to clients only last changed things
Definition at line 1173 of file gameServer.java.

void gameServer.serverWaitingDraw ()

Waiting view placeholder ;-)
Definition at line 74 of file gameServer.java.

void gameServer.settings ()

Definition at line 1228 of file gameServer.java.

void gameServer.setup ()

Startup of a game server.
It initialises window (if required) and TCP/IP port. It exits, if is not able to do that.
Definition at line 41 of file gameServer.java.

void gameServer.stepOfGameMechanics ()

Do in game world all, what is independent of players actions.
Definition at line 1102 of file gameServer.java.

void gameServer.visualise2D (float *startX*, float *startY*, float *width*, float *height*)

Simplest flat map visualisation of game board.
Definition at line 536 of file gameServer.java.

void gameServer.whenClientConnected (Client *newClient*, String *playerName*)

This is stuff that should be done,
when new client was connected.
Definition at line 110 of file gameServer.java.

Member Data Documentation

final int gameServer.ACTRAD_MSK = unbinary("010000000")[package]

object changed its radius of activity (ex. go to sleep);

Definition at line 248 of file gameServer.java.

final int gameServer.ALL_CHNG_MSK = MOVED_MSK | VISUAL_MSK | COLOR_MSK | STATE_MSK [package]

All initial changes.

Definition at line 254 of file gameServer.java.

String [] gameServer.avatars ={".","^v^" ,"o^o","@","&","ðŸ˜ƒ","ðŸ˜Ž"} [package]

peoples...

Definition at line 238 of file gameServer.java.

final int gameServer.COLOR_MSK = unbinary("000001000") [package]

object changed its colors

Definition at line 244 of file gameServer.java.

int gameServer.DEBUG =0 [package]

Server for gameClients - **setup()** & **draw()** SOURCE FILE.

Losely base on example code for a server with multiple clients communicating to only one at a time. (see: `https://forum.processing.org/one/topic/how-do-i-send-data-to-only-one-client-using-the-network-library.html`) Level of debug logging

Definition at line 32 of file gameServer.java.

GameObject [] gameServer.gameWorld =null [package]

MAIN ARRAY OF GameObjects.

Definition at line 670 of file gameServer.java.

final int gameServer.HPOINT_MSK = unbinary("000010000") [package]

object changed its hp state (most frequently changed state)

Definition at line 245 of file gameServer.java.

int gameServer.indexOfMe =-1 [package]

Index of object visualising the client or the server supervisor.

Definition at line 234 of file gameServer.java.

int gameServer.INFO_LEVEL =1 | SCORE_MSK [package]

Visualisation with information about objects (name & score by default)

Definition at line 257 of file gameServer.java.

float gameServer.initialMaxX =100 [package]

Initial horizontal size of game "board".
Definition at line 232 of file gameServer.java.

float gameServer.initialMaxY =100 [package]

Initial vertical size of game "board".
Definition at line 233 of file gameServer.java.

int gameServer.initialSizeOfMainArray =30 [package]

Game classes and its basic behaviours.
Initial number of @GameObjects in @gameWorld
Definition at line 231 of file gameServer.java.

boolean gameServer.KEEP_ASPECT =true [package]

Visualisation with proportional aspect ratio.
Definition at line 259 of file gameServer.java.

Server gameServer.mainServer [package]

Object representing server's TCP/IP COMMUNICATION.
Definition at line 34 of file gameServer.java.

final int gameServer.MOVED_MSK = unbinary("000000010") [package]

object was moved (0x1)
Definition at line 242 of file gameServer.java.

final int gameServer.PASRAD_MSK = unbinary("001000000") [package]

object changed its passive radius (ex. grow);
Definition at line 247 of file gameServer.java.

String [] gameServer.plants = {"_","O","...\nI","_\\|/_\nI ","|/",":","â˜¨"} [package]

plants...
Definition at line 237 of file gameServer.java.

Player [] gameServer.players = new Player[0] [package]

The array of clients (players)

Definition at line 36 of file gameServer.java.

final int gameServer.SCORE_MSK = unbinary("000100000")[package]

object changed its score (for players it is most frequently changed state)

Definition at line 246 of file gameServer.java.

String gameServer.serverIP ="127.0.0.1"[package]

Declaration common for client and server (op.codes and coding/decoding functions)

localhost

Definition at line 683 of file gameServer.java.

int gameServer.servPORT =5205[package]

Teoretically it could be any above 1024.

Definition at line 684 of file gameServer.java.

final int gameServer.STATE_MSK = HPOINT_MSK | SCORE_MSK | PASRAD_MSK | ACTRAD_MSK[package]

object changed its states

Definition at line 249 of file gameServer.java.

final int gameServer.TOUCH_MSK = unbinary("1000000000000000")[package]

To visualize the interaction between background objects.

16bits

Definition at line 252 of file gameServer.java.

boolean gameServer.VIS_MIN_MAX =true[package]

Visualisation with min/max value.

Definition at line 258 of file gameServer.java.

final int gameServer.VISSWITH = unbinary("000000001")[package]

object is invisible (but in info level name is visible)

Definition at line 241 of file gameServer.java.

final int gameServer.VISUAL_MSK = unbinary("000000100")[package]

object changed its type of view

Definition at line 243 of file gameServer.java.

boolean gameServer.wholeUpdateRequested =false[package]

Information about a client requesting information about the entire scene.
In such a case, it is sent to all clients (in this template of the server)
Definition at line 1012 of file gameServer.java.

int gameServer.Xmargin =0[package]

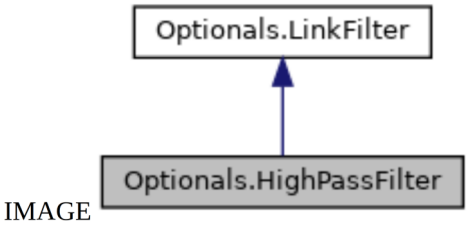
gameServer - more communication & game logic
Left margin of server screen (status column)
Definition at line 1011 of file gameServer.java.

The documentation for this class was generated from the following file:
src.java/**gameServer.java**

Optionals.HighPassFilter Class Reference

High Pass Filter.

Inheritance diagram for Optionals.HighPassFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

HighPassFilter (float tres)

Package Attributes

float **treshold**

Detailed Description

High Pass Filter.

Class which filters links with higher weights

Definition at line 1691 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.HighPassFilter.HighPassFilter (float *tres*)[package]

Definition at line 1694 of file Optionals.java.

Member Function Documentation

boolean Optionals.HighPassFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1695 of file Optionals.java.

Member Data Documentation

float Optionals.HighPassFilter.treshold [package]

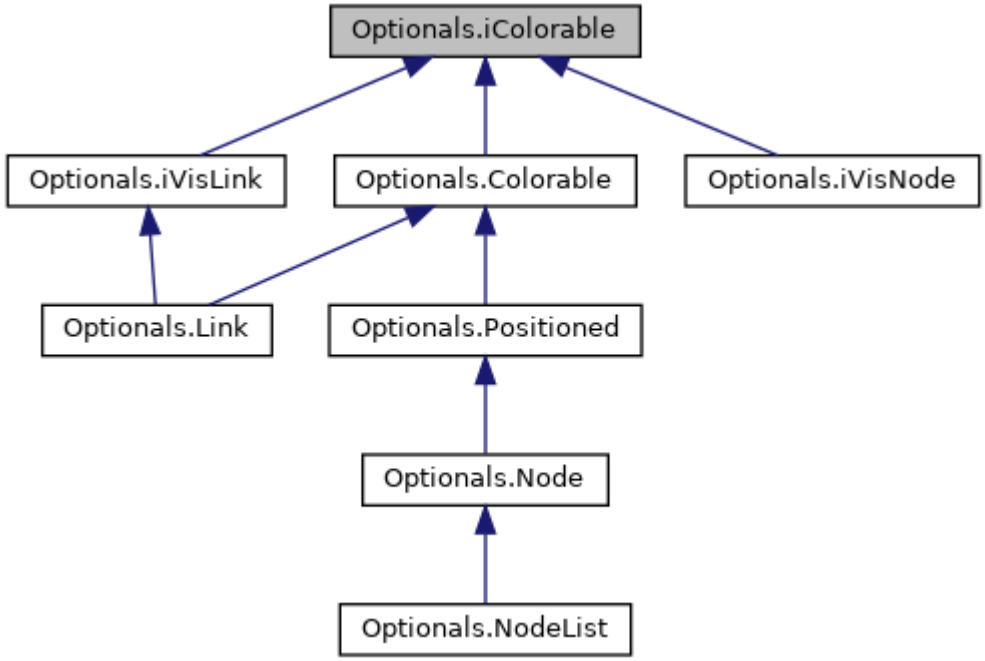
Definition at line 1693 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.iColorable Interface Reference

VISUALISATION INTERFACES:

Inheritance diagram for Optionals.iColorable:



IMAGE

Public Member Functions

void **setFill** (float intensity)
void **setStroke** (float intensity)

Detailed Description

VISUALISATION INTERFACES:

Forcing setFill & setStroke methods for visualisation

Definition at line 1572 of file Optionals.java.

Member Function Documentation

void Optionals.iColorable.setFill (float *intensity*)

Implemented in **Optionals.Colorable** (*p.44*).

void Optionals.iColorable.setStroke (float *intensity*)

Implemented in **Optionals.Colorable** (*p.44*), and **Optionals.Link** (*p.118*).

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

src.java/**Optionals.java**

Optionals.iDescribable Interface Reference

Any object which have description as (potentially) long, multi line string.

Public Member Functions

String **Description** ()

Detailed Description

Any object which have description as (potentially) long, multi line string.

Definition at line 1548 of file Optionals.java.

Member Function Documentation

String Optionals.iDescribable.Description ()

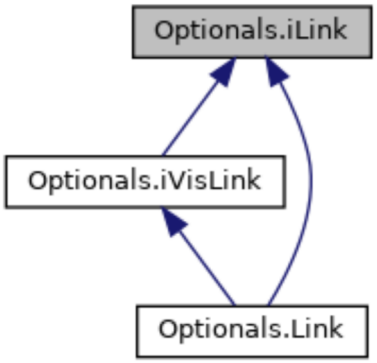
The documentation for this interface was generated from the following file:

src.java/**Optionals.java**

Optionals.iLink Interface Reference

Network connection/link interface Is **iLink** interface really needed?

Inheritance diagram for Optionals.iLink:



IMAGE

Public Member Functions

float **getWeight** ()

Detailed Description

Network connection/link interface Is **iLink** interface really needed?

Definition at line 1589 of file Optionals.java.

Member Function Documentation

float Optionals.iLink.getWeight ()

Implemented in **Optionals.Link** (*p.118*).

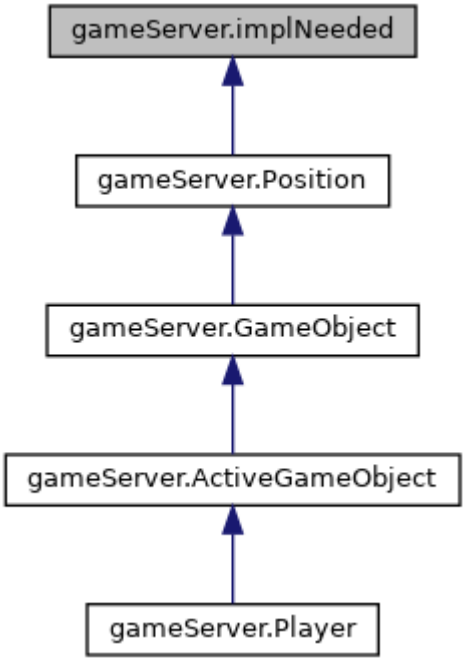
The documentation for this interface was generated from the following file:

src.java/**Optionals.java**

gameServer.implNeeded Class Reference

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

Inheritance diagram for gameServer.implNeeded:



IMAGE

Public Member Functions

String **myClassName** ()

Package Attributes

int **flags** =0

Detailed Description

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

Definition at line 264 of file gameServer.java.

Member Function Documentation

String gameServer.implNeeded.myClassName ()

Definition at line 268 of file gameServer.java.

Member Data Documentation

int gameServer.implNeeded.flags =0[package]

Definition at line 266 of file gameServer.java.

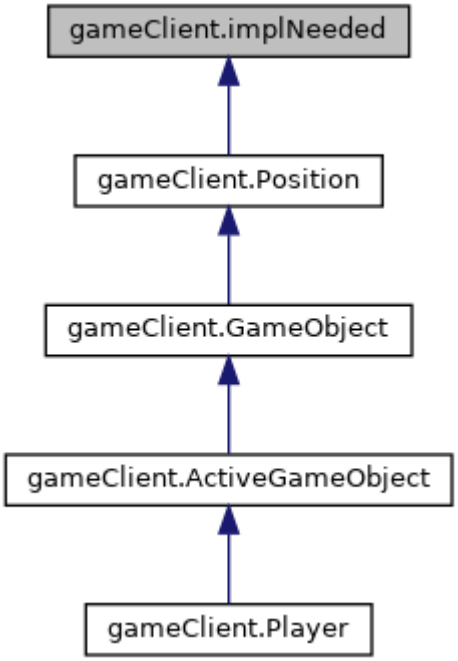
The documentation for this class was generated from the following file:

src.java/**gameServer.java**

gameClient.implNeeded Class Reference

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

Inheritance diagram for gameClient.implNeeded:



IMAGE

Public Member Functions

String **myClassName** ()

Package Attributes

int **flags** =0

**_MSK alloved here*

Detailed Description

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

Definition at line 546 of file gameClient.java.

Member Function Documentation

String gameClient.implNeeded.myClassName ()

Definition at line 550 of file gameClient.java.

Member Data Documentation

int gameClient.implNeeded.flags =0[package]

*_MSK alloved here

Definition at line 548 of file gameClient.java.

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

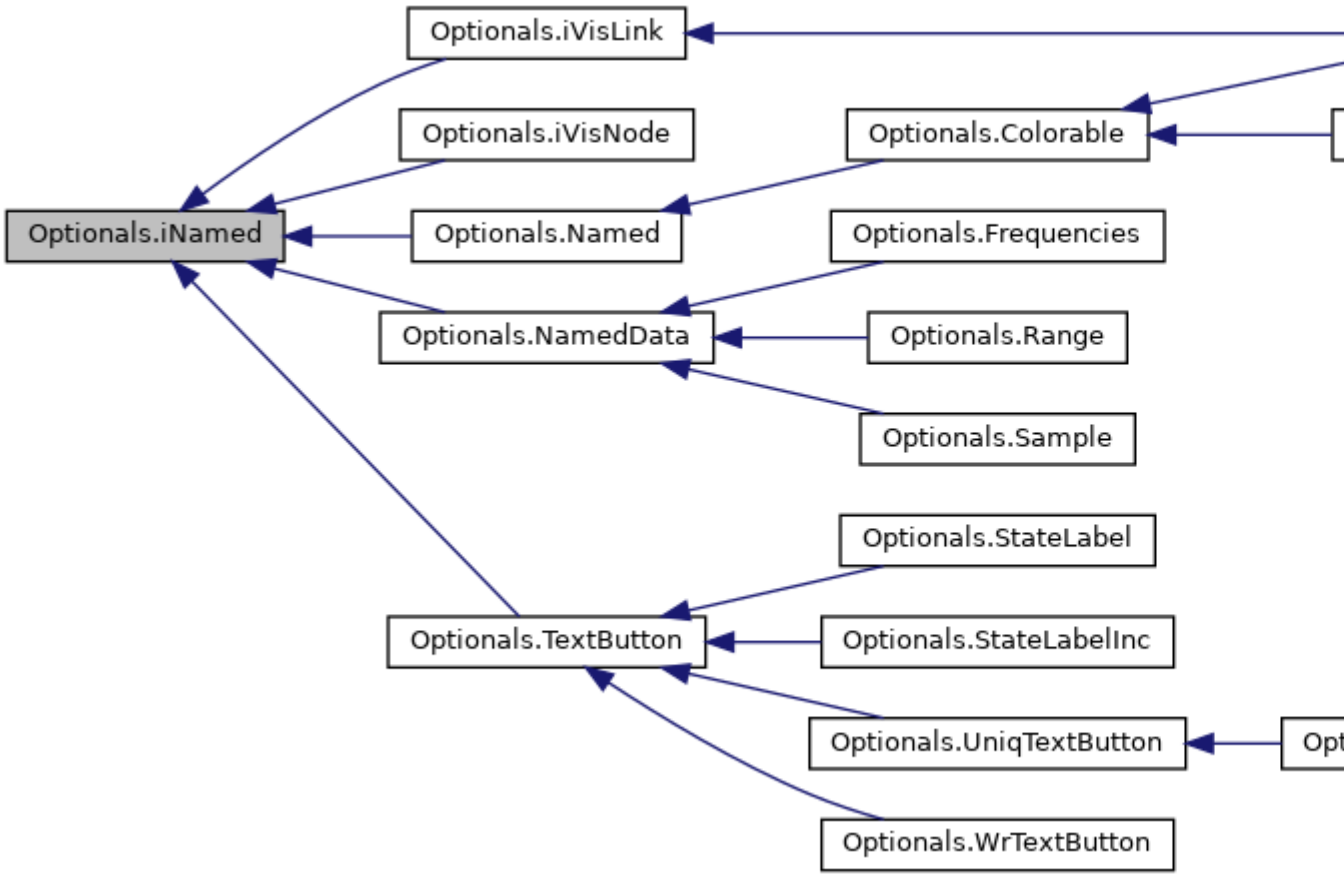
src.java/**gameClient.java**

Optionals.iNamed Interface Reference

Forcing name available as String (planty of usage)

Inheritance diagram for Optionals.iNamed:

IMAGE



Public Member Functions

String **name** ()

Detailed Description

Forcing name available as String (planty of usage)

Definition at line 1543 of file Optionals.java.

Member Function Documentation

String Optionals.iNamed.name ()

Implemented in **Optionals.Link** (*p.118*), **Optionals.Named** (*p.124*), **Optionals.TextButton** (*p.202*), and **Optionals.NamedData** (*p.126*).

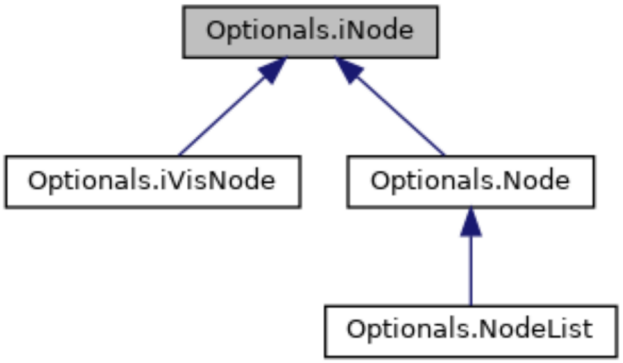
The documentation for this interface was generated from the following file:

src.java/**Optionals.java**

Optionals.iNode Interface Reference

Network node interface "Conn" below is a shortage from Connection.

Inheritance diagram for Optionals.iNode:



IMAGE

Public Member Functions

int **addConn** (**Link** l)
int **delConn** (**Link** l)
int **numOfConn** ()
Link getConn (int i)
Link getConn (**Node** n)
Link getConn (String k)
Link[] **getConns** (**LinkFilter** f)

Detailed Description

Network node interface "Conn" below is a shortage from Connection.

Definition at line 1595 of file Optionals.java.

Member Function Documentation

int Optionals.iNode.addConn (Link *l*)

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.128*).

int Optionals.iNode.delConn (Link *l*)

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.129*).

Link Optionals.iNode.getConn (int *i*)

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.129*).

Link Optionals.iNode.getConn (Node *n*)

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.129*).

Link Optionals.iNode.getConn (String *k*)

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.129*).

Link [] Optionals.iNode.getConns (LinkFilter *f*)

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.129*).

int Optionals.iNode.numOfConn ()

Implemented in **Optionals.NodeList** (*p.131*), and **Optionals.Node** (*p.129*).

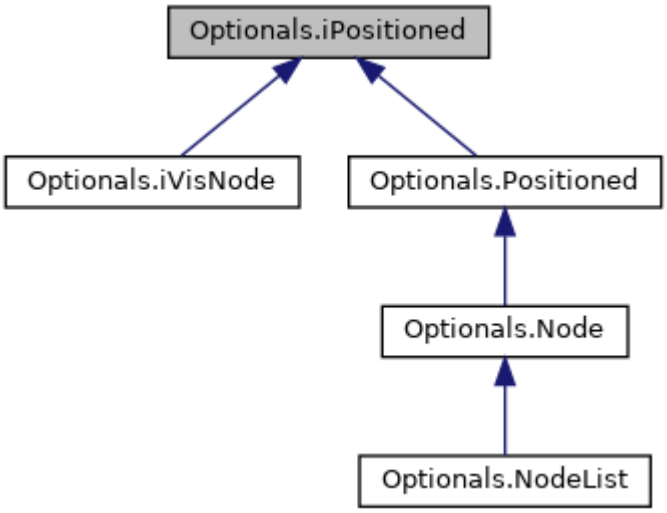
The documentation for this interface was generated from the following file:

src.java/**Optionals.java**

Optionals.iPositioned Interface Reference

Forcing **posX()** & **posY()** & **posZ()** methods for visualisation and mapping

Inheritance diagram for Optionals.iPositioned:



IMAGE

Public Member Functions

float **posX** ()
float **posY** ()
float **posZ** ()

Detailed Description

Forcing **posX()** & **posY()** & **posZ()** methods for visualisation and mapping

Definition at line 1578 of file Optionals.java.

Member Function Documentation

float Optionals.iPositioned.posX ()

Implemented in **Optionals.Positioned** (*p.184*).

float Optionals.iPositioned.posY ()

Implemented in **Optionals.Positioned** (*p.185*).

float Optionals.iPositioned.posZ ()

Implemented in **Optionals.Positioned** (*p.185*).

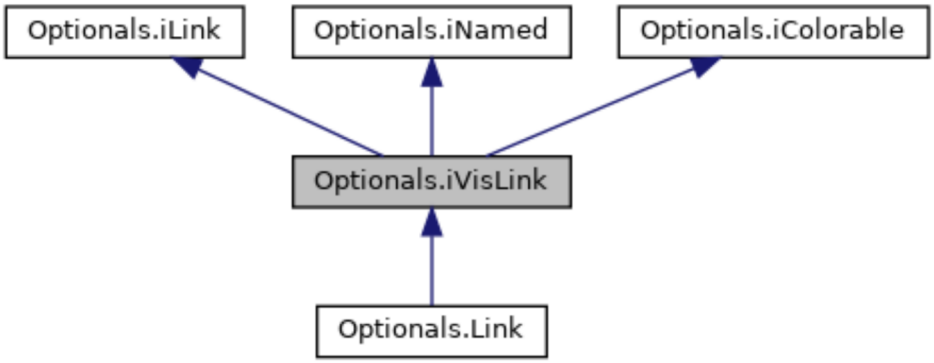
The documentation for this interface was generated from the following file:

src.java/**Optionals.java**

Optionals.iVisLink Interface Reference

Visualisable network connection.

Inheritance diagram for Optionals.iVisLink:



IMAGE

Additional Inherited Members

Detailed Description

Visualisable network connection.

Definition at line 1611 of file Optionals.java.

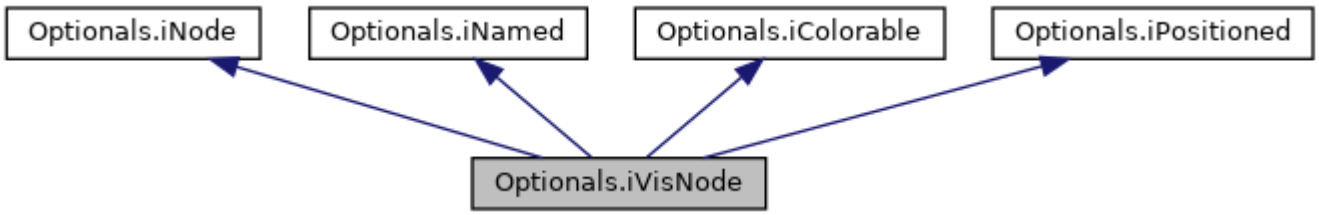
The documentation for this interface was generated from the following file:
src.java/**Optionals.java**

Optionals.iVisNode Interface Reference

Visualisable network node.

Inheritance diagram for Optionals.iVisNode:

IMAGE



Additional Inherited Members

Detailed Description

Visualisable network node.

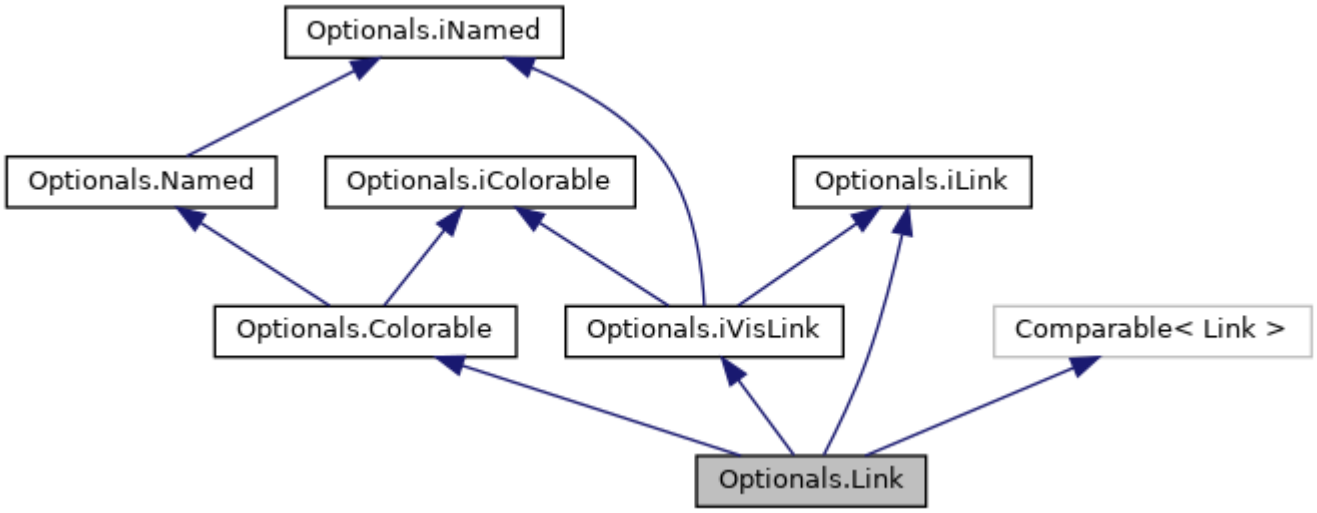
Definition at line 1607 of file Optionals.java.

The documentation for this interface was generated from the following file:
src.java/**Optionals.java**

Optionals.Link Class Reference

Inheritance diagram for Optionals.Link:

IMAGE



Public Member Functions

String **fullInfo** (String fieldSeparator)
int **compareTo** (**Link** o)
String **name** ()
float **getWeight** ()
void **setStroke** (float Intensity)

Package Functions

Link (**Node** targ, float we, int ty)

Package Attributes

Node target

INFO: This class is available for user modifications.

float **weight**
int **ltype**

Detailed Description

Definition at line 1838 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.Link.Link (Node *targ*, float *we*, int *ty*) [package]

Definition at line 1846 of file Optionals.java.

Member Function Documentation

int Optionals.Link.compareTo (Link *o*)

Definition at line 1854 of file Optionals.java.

String Optionals.Link.fullInfo (String *fieldSeparator*)

Definition at line 1848 of file Optionals.java.

float Optionals.Link.getWeight ()

Implements **Optionals.iLink** (*p.104*).
Definition at line 1865 of file Optionals.java.

String Optionals.Link.name ()

Implements **Optionals.iNamed** (*p.109*).
Definition at line 1863 of file Optionals.java.

void Optionals.Link.setStroke (float *Intensity*)

Reimplemented from **Optionals.Colorable** (*p.44*).
Definition at line 1867 of file Optionals.java.

Member Data Documentation

int Optionals.Link.ltype[package]

Definition at line 1842 of file Optionals.java.

Node Optionals.Link.target[package]

INFO: This class is available for user modifications.
Definition at line 1840 of file Optionals.java.

float Optionals.Link.weight[package]

Definition at line 1841 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.LinkFactory Class Reference

Public Member Functions

Link makeLink (**Node** Source, **Node** Target)

INFO:

Link makeSelfLink (**Node** Self)

Detailed Description

Definition at line 1797 of file Optionals.java.

Member Function Documentation

Link Optionals.LinkFactory.makeLink (Node *Source*, Node *Target*)

INFO:

Definition at line 1799 of file Optionals.java.

Link Optionals.LinkFactory.makeSelfLink (Node *Self*)

Definition at line 1801 of file Optionals.java.

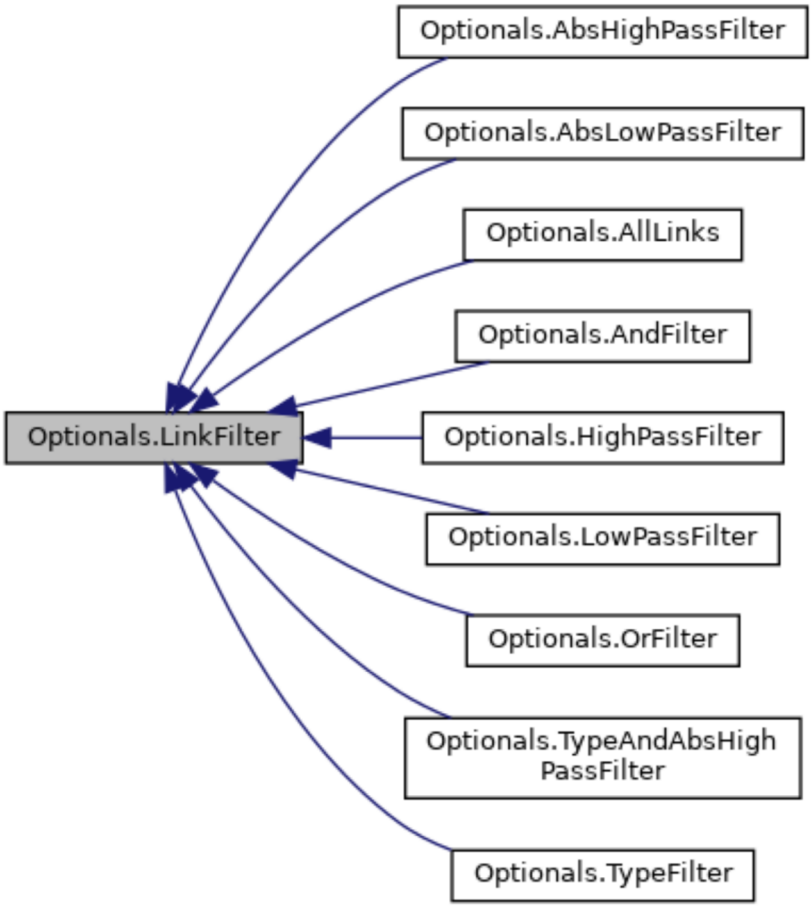
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.LinkFilter Class Reference

DEBUG level for network. Visible autside this file!

Inheritance diagram for Optionals.LinkFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Detailed Description

DEBUG level for network. Visible autside this file!

Definition at line 1791 of file Optionals.java.

Member Function Documentation

boolean Optionals.LinkFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented in **Optionals.AbsHighPassFilter** (*p.20*), **Optionals.AbsLowPassFilter** (*p.22*), **Optionals.HighPassFilter** (*p.99*), **Optionals.LowPassFilter** (*p.122*), **Optionals.TypeFilter**

(*p.206*), **Optionals.OrFilter** (*p.166*), **Optionals.AndFilter** (*p.31*), **Optionals.TypeAndAbsHighPassFilter** (*p.204*), and **Optionals.AllLinks** (*p.30*).

Definition at line 1793 of file Optionals.java.

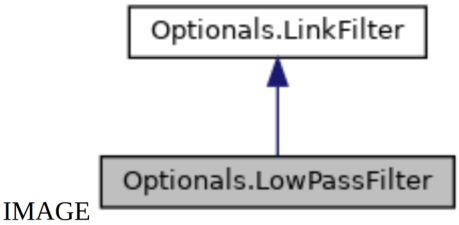
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.LowPassFilter Class Reference

Low Pass Filter.

Inheritance diagram for Optionals.LowPassFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

LowPassFilter (float tres)

Package Attributes

float **treshold**

Detailed Description

Low Pass Filter.

Class which filters links with lower weights

Definition at line 1682 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.LowPassFilter.LowPassFilter (float *tres*) [package]

Definition at line 1685 of file Optionals.java.

Member Function Documentation

boolean Optionals.LowPassFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1686 of file Optionals.java.

Member Data Documentation

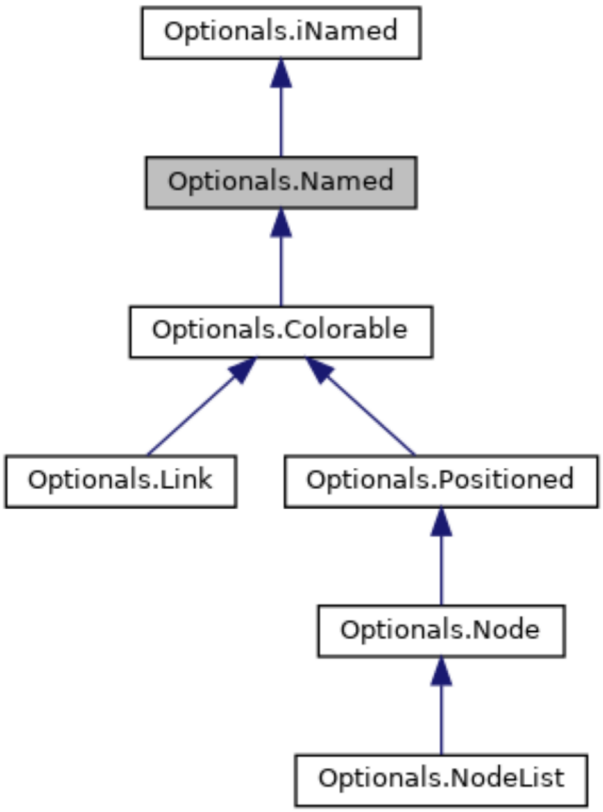
float Optionals.LowPassFilter.treshold [package]

Definition at line 1684 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.Named Class Reference

Inheritance diagram for Optionals.Named:



IMAGE

Public Member Functions

String **name** ()

*INFO: Forcing **name()** method for visualisation and mapping*

Detailed Description

Definition at line 1805 of file Optionals.java.

Member Function Documentation

String Optionals.Named.name ()

INFO: Forcing **name()** method for visualisation and mapping

Implements **Optionals.iNamed** (*p.109*).

Reimplemented in **Optionals.Link** (*p.118*).

Definition at line 1807 of file Optionals.java.

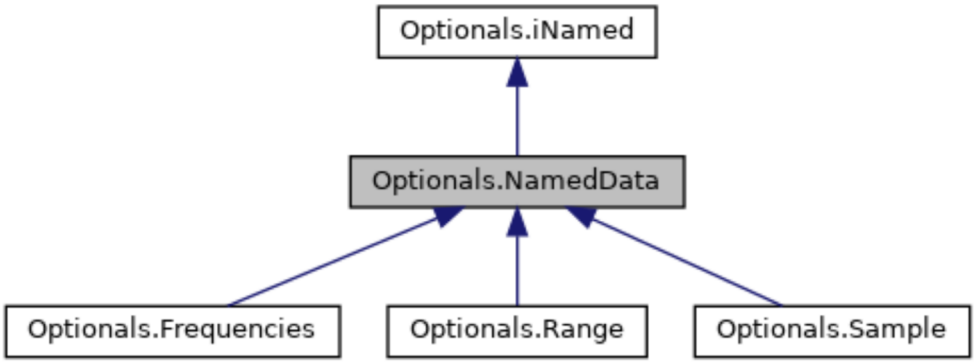
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.NamedData Class Reference

Functions & classes for chart making.

Inheritance diagram for Optionals.NamedData:



IMAGE

Public Member Functions

String **name** ()

Package Functions

NamedData (String Name)

Package Attributes

String **myName**

Detailed Description

Functions & classes for chart making.

A class that implements only the interface having a proper object name

Definition at line 82 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.NamedData.NamedData (String *Name*) [package]

Definition at line 85 of file Optionals.java.

Member Function Documentation

String Optionals.NamedData.name ()

Implements **Optionals.iNamed** (*p.109*).

Definition at line 86 of file Optionals.java.

Member Data Documentation

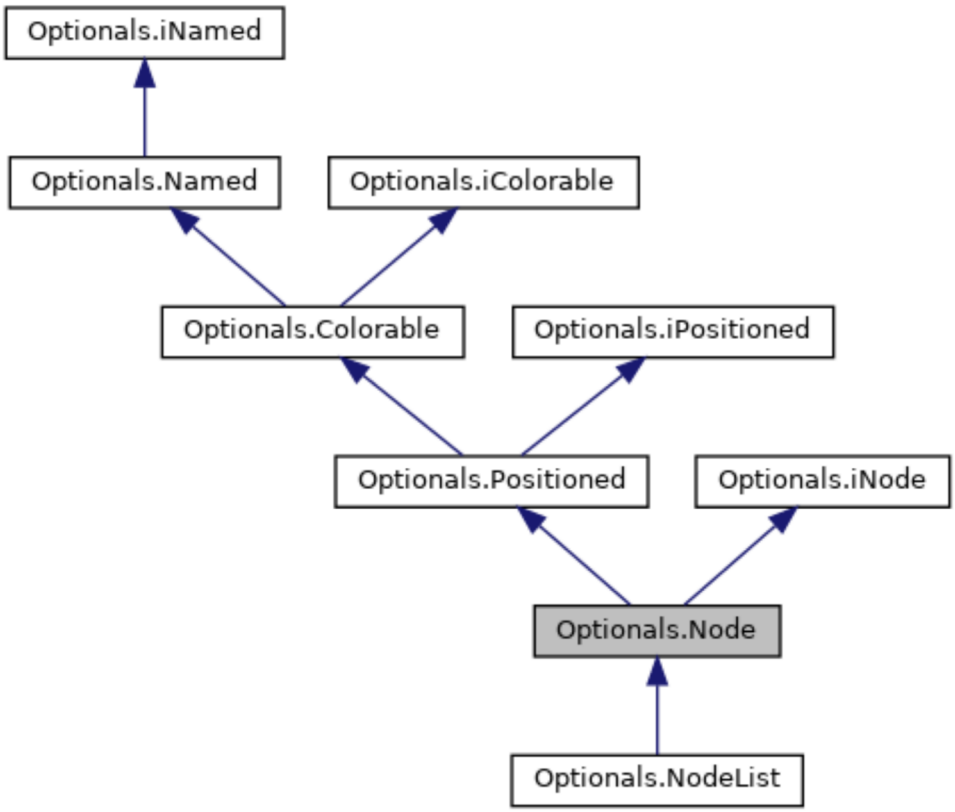
String Optionals.NamedData.myName[package]

Definition at line 84 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.Node Class Reference

Inheritance diagram for Optionals.Node:



IMAGE

Public Member Functions

int **addConn** (**Link** l)

INFO:

int **delConn** (**Link** l)
int **numOfConn** ()
Link getConn (int i)
Link getConn (**Node** n)
Link getConn (String k)
Link[] **getConns** (**LinkFilter** f)

Detailed Description

Definition at line 1823 of file Optionals.java.

Member Function Documentation

int Optionals.Node.addConn (Link *l*)

INFO:

Implements **Optionals.iNode** (*p.111*).

Reimplemented in **Optionals.NodeList** (*p.131*).

Definition at line 1825 of file Optionals.java.

int Optionals.Node.delConn (Link *l*)

Implements **Optionals.iNode** (*p.111*).
Reimplemented in **Optionals.NodeList** (*p.131*).
Definition at line 1826 of file Optionals.java.

Link Optionals.Node.getConn (int *i*)

Implements **Optionals.iNode** (*p.111*).
Reimplemented in **Optionals.NodeList** (*p.131*).
Definition at line 1828 of file Optionals.java.

Link Optionals.Node.getConn (Node *n*)

Implements **Optionals.iNode** (*p.111*).
Reimplemented in **Optionals.NodeList** (*p.131*).
Definition at line 1829 of file Optionals.java.

Link Optionals.Node.getConn (String *k*)

Implements **Optionals.iNode** (*p.112*).
Reimplemented in **Optionals.NodeList** (*p.131*).
Definition at line 1830 of file Optionals.java.

Link [] Optionals.Node.getConns (LinkFilter *f*)

Implements **Optionals.iNode** (*p.112*).
Reimplemented in **Optionals.NodeList** (*p.131*).
Definition at line 1831 of file Optionals.java.

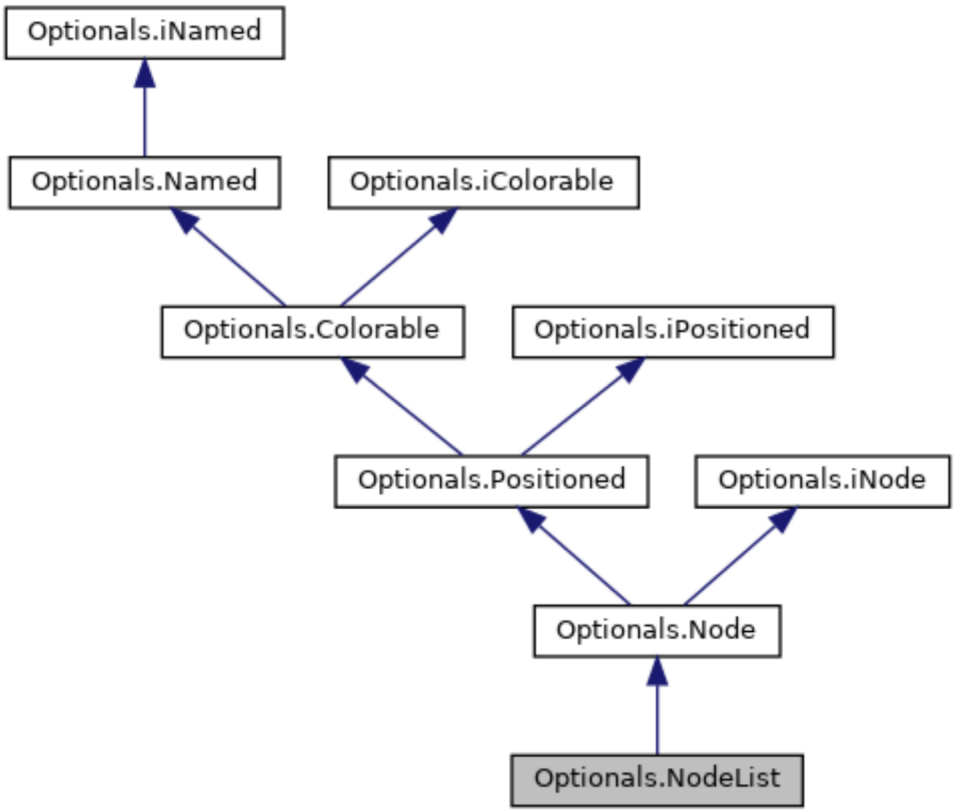
int Optionals.Node.numOfConn ()

Implements **Optionals.iNode** (*p.112*).
Reimplemented in **Optionals.NodeList** (*p.131*).
Definition at line 1827 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.NodeList Class Reference

Inheritance diagram for Optionals.NodeList:



IMAGE

Public Member Functions

int **numOfConn** ()
int **addConn** (**Link** l)
INFO:

int **delConn** (**Link** l)
Link getConn (int i)
Link getConn (**Node** n)
Link getConn (String k)
Link[] **getConns** (**LinkFilter** f)

Package Functions

NodeList ()

Package Attributes

ArrayList< **Link** > **connections**
INFO: ***Node*** *implementation based on list.*

Detailed Description

Definition at line 2317 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.NodeList.NodeList () [package]

Definition at line 2321 of file Optionals.java.

Member Function Documentation

int Optionals.NodeList.addConn (Link *l*)

INFO:
Reimplemented from **Optionals.Node** (*p.128*).
Definition at line 2328 of file Optionals.java.

int Optionals.NodeList.delConn (Link *l*)

Reimplemented from **Optionals.Node** (*p.129*).
Definition at line 2349 of file Optionals.java.

Link Optionals.NodeList.getConn (int *i*)

Reimplemented from **Optionals.Node** (*p.129*).
Definition at line 2357 of file Optionals.java.

Link Optionals.NodeList.getConn (Node *n*)

Reimplemented from **Optionals.Node** (*p.129*).
Definition at line 2363 of file Optionals.java.

Link Optionals.NodeList.getConn (String *k*)

Reimplemented from **Optionals.Node** (*p.129*).
Definition at line 2374 of file Optionals.java.

Link [] Optionals.NodeList.getConns (LinkFilter *f*)

Reimplemented from **Optionals.Node** (*p.129*).
Definition at line 2385 of file Optionals.java.

int Optionals.NodeList.numOfConn ()

Reimplemented from **Optionals.Node** (*p.129*).
Definition at line 2326 of file Optionals.java.

Member Data Documentation

ArrayList<Link> Optionals.NodeList.connections [package]

INFO: **Node** implementation based on list.

Definition at line 2319 of file Optionals.java.

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

src.java/**Optionals.java**

gameServer.Opcs Class Reference

Protocol dictionary ("opcodes" etc.)

Static Package Attributes

static final String **name** ="sampleGame"
ASCI IDENTIFIER OF PROTOCOL.

static final String **sYOU** ="Y"
REPLACER OF CORESPONDENT NAME as a ready to use String.

static final char **EOR** =0x03
End of record (EOR). EOL is not used, because of it use inside data starings.

static final char **SPC** ='\t'
Field separator.

static final char **ERR** ='e'
Error message for partner.

static final char **HEL** ='H'
Hello message (client-server handshake)

static final char **IAM** ='I'
I am "name of server/name of client".

static final char **YOU** ='Y'
Redefining player name if not suitable.

static final char **GET** ='G'
Get global resource by name TODO.

static final char **BIN** ='B'
Binary hunk of resources (name.type\tsize\tthen data) TODO Data hunk is recived exactly "as is"!

static final char **TXT** ='X'
Text hunk of resources (name.type\tsize\tthen data) TODO Text may be recoded on the reciver side if needed!

static final char **OBJ** ='O'
Objects managment: "On typename objectName" or "Od objectName".

static final char **UPD** ='U'
Request for update about a whole scene.

static final char **VIS** ='V'
Visualisation info for a particular object.

static final char **COL** ='C'
Colors of a particular object.

static final char **STA** ='S'
Named state attribute of a particular object (ex.: objname\thp\tval, objname\tsc\tval etc.)

static final char **EUC** ='E'
Euclidean position of an object.

static final char **POL** ='P'
Polar position of an object.

static final char **TCH** ='T'
Active "Touch" with other object (info about name & possible actions)

static final char **DTC** ='D'
Detouch with any of previously touched object (name provided)

static final char **NAV** ='N'
Navigation of the avatar (wsad and arrows in the template)

static final char **ACT** ='A'
'defo'(-ult) or user defined actions of the avatar

Detailed Description

Protocol dictionary ("opcodes" etc.)

Definition at line 687 of file gameServer.java.

Member Data Documentation

final char gameServer.Opcs.ACT ='A'[static], [package]

'defo'(-ult) or user defined actions of the avatar

Definition at line 718 of file gameServer.java.

final char gameServer.Opcs.BIN ='B'[static], [package]

Binary hunk of resources (name.type\tsize\tthen data) TODO Data hunk is recived exactly "as is"!

Definition at line 701 of file gameServer.java.

final char gameServer.Opcs.COL ='C'[static], [package]

Colors of a particular object.
Definition at line 709 of file gameServer.java.

final char gameServer.Opcs.DTC ='D'[static], [package]

Detouch with any of previously touched object (name provided)
Definition at line 715 of file gameServer.java.

final char gameServer.Opcs.EOR =0x03[static], [package]

End of record (EOR). EOL is not used, because of it use inside data starings.
Definition at line 692 of file gameServer.java.

final char gameServer.Opcs.ERR ='e'[static], [package]

Error message for partner.
Definition at line 695 of file gameServer.java.

final char gameServer.Opcs.EUC ='E'[static], [package]

Euclidean position of an object.
Definition at line 711 of file gameServer.java.

final char gameServer.Opcs.GET ='G'[static], [package]

Get global resource by name TODO.
Definition at line 700 of file gameServer.java.

final char gameServer.Opcs.HEL ='H'[static], [package]

Hello message (client-server handshake)
Definition at line 696 of file gameServer.java.

final char gameServer.Opcs.IAM ='I'[static], [package]

I am "name of server/name of client".
Definition at line 697 of file gameServer.java.

final String gameServer.Opcs.name ="sampleGame"[static], [package]

ASCI IDENTIFIER OF PROTOCOL.
Definition at line 688 of file gameServer.java.

final char gameServer.Opcs.NAV ='N'[static], [package]

Navigation of the avatar (wsad and arrows in the template)

Definition at line 717 of file gameServer.java.

final char gameServer.Opcs.OBJ ='O'[static], [package]

Objects managment: "On typename objectName" or "Od objectName".

Definition at line 705 of file gameServer.java.

final char gameServer.Opcs.POL ='P'[static], [package]

Polar position of an object.

Definition at line 712 of file gameServer.java.

final char gameServer.Opcs.SPC ='\t'[static], [package]

Field separator.

Definition at line 693 of file gameServer.java.

final char gameServer.Opcs.STA ='S'[static], [package]

Named state attribute of a particular object (ex.: objname\thp\tval, objname\tsc\tval etc.)

Definition at line 710 of file gameServer.java.

final String gameServer.Opcs.sYOU ="Y"[static], [package]

REPLACER OF CORESPONDENT NAME as a ready to use String.

Character.toString(YOU);<-not for static

Definition at line 689 of file gameServer.java.

final char gameServer.Opcs.TCH ='T'[static], [package]

Active "Touch" with other object (info about name & possible actions)

Definition at line 714 of file gameServer.java.

final char gameServer.Opcs.TXT ='X'[static], [package]

Text hunk of resources (name.type\tsize\tthen data) TODO Text may be recoded on the reciver side if needed!

Definition at line 703 of file gameServer.java.

final char gameServer.Opcs.UPD ='U'[static], [package]

Request for update about a whole scene.

Definition at line 707 of file gameServer.java.

final char gameServer.Opcs.VIS ='V'[static], [package]

Visualisation info for a particular object.

Definition at line 708 of file gameServer.java.

final char gameServer.Opcs.YOU ='Y'[static], [package]

Redefining player name if not suitable.

Definition at line 698 of file gameServer.java.

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

src.java/**gameServer.java**

gameClient.Opcs Class Reference

Protocol dictionary ("opcodes" etc.)

Static Package Attributes

static final String **name** ="sampleGame"
static final String **sYOU** ="Y"
REPLACER OF CORESPONDENT NAME as a ready to use String.

static final char **EOR** =0x03
End of record (EOR). EOL is not used, because of it use inside data starings.

static final char **SPC** ='\t'
Field separator.

static final char **ERR** ='e'
Error message for partner.

static final char **HEL** ='H'
Hello message (client-server handshake)

static final char **IAM** ='I'
I am "name of server/name of client".

static final char **YOU** ='Y'
Redefining player name if not suitable.

static final char **GET** ='G'
Get global resource by name TODO.

static final char **BIN** ='B'
Binary hunk of resources (name.type\tsize\tthen data) TODO Data hunk is recived exactly "as is"!

static final char **TXT** ='X'
Text hunk of resources (name.type\tsize\tthen data) TODO Text may be recoded on the reciver side if needed!

static final char **OBJ** ='O'
Objects managment: "On typename objectName" or "Od objectName".

static final char **UPD** ='U'
Request for update about a whole scene.

static final char **VIS** ='V'
Visualisation info for a particular object.

static final char **COL** ='C'
Colors of a particular object.

static final char **STA** ='S'
Named state attribute of a particular object (ex.: objname\thp\tval, objname\tsc\tval etc.)

static final char **EUC** ='E'
Euclidean position of an object.

static final char **POL** ='P'
Polar position of an object.

static final char **TCH** ='T'
Active "Touch" with other object (info about name & possible actions)

static final char **DTC** ='D'
Detouch with any of previously touched object (name provided)

static final char **NAV** ='N'
Navigation of the avatar (wsad and arrows in the template)

static final char **ACT** ='A'
'defo'(-ult) or user defined actions of the avatar

Detailed Description

Protocol dictionary ("opcodes" etc.)

Definition at line 969 of file gameClient.java.

Member Data Documentation

final char gameClient.Opcs.ACT ='A'[static], [package]

'defo'(-ult) or user defined actions of the avatar

Definition at line 1000 of file gameClient.java.

final char gameClient.Opcs.BIN ='B'[static], [package]

Binary hunk of resources (name.type\tsize\tthen data) TODO Data hunk is recived exactly "as is"!

Definition at line 983 of file gameClient.java.

final char gameClient.Opcs.COL ='C'[static], [package]

Colors of a particular object.
Definition at line 991 of file gameClient.java.

final char gameClient.Opcs.DTC ='D'[static], [package]

Detouch with any of previously touched object (name provided)
Definition at line 997 of file gameClient.java.

final char gameClient.Opcs.EOR =0x03[static], [package]

End of record (EOR). EOL is not used, because of it use inside data starings.
Definition at line 974 of file gameClient.java.

final char gameClient.Opcs.ERR ='e'[static], [package]

Error message for partner.
Definition at line 977 of file gameClient.java.

final char gameClient.Opcs.EUC ='E'[static], [package]

Euclidean position of an object.
Definition at line 993 of file gameClient.java.

final char gameClient.Opcs.GET ='G'[static], [package]

Get global resource by name TODO.
Definition at line 982 of file gameClient.java.

final char gameClient.Opcs.HEL ='H'[static], [package]

Hello message (client-server handshake)
Definition at line 978 of file gameClient.java.

final char gameClient.Opcs.IAM ='I'[static], [package]

I am "name of server/name of client".
Definition at line 979 of file gameClient.java.

final String gameClient.Opcs.name ="sampleGame"[static], [package]

Definition at line 970 of file gameClient.java.

final char gameClient.Opcs.NAV ='N'[static], [package]

Navigation of the avatar (wsad and arrows in the template)

Definition at line 999 of file gameClient.java.

final char gameClient.Opcs.OBJ ='O'[static], [package]

Objects managment: "On typename objectName" or "Od objectName".

Definition at line 987 of file gameClient.java.

final char gameClient.Opcs.POL ='P'[static], [package]

Polar position of an object.

Definition at line 994 of file gameClient.java.

final char gameClient.Opcs.SPC ='\t'[static], [package]

Field separator.

Definition at line 975 of file gameClient.java.

final char gameClient.Opcs.STA ='S'[static], [package]

Named state attribute of a particular object (ex.: objname\thp\tval, objname\tsc\tval etc.)

Definition at line 992 of file gameClient.java.

final String gameClient.Opcs.sYOU ="Y"[static], [package]

REPLACER OF CORESPONDENT NAME as a ready to use String.

Character.toString(YOU);<-not for static

Definition at line 971 of file gameClient.java.

final char gameClient.Opcs.TCH ='T'[static], [package]

Active "Touch" with other object (info about name & possible actions)

Definition at line 996 of file gameClient.java.

final char gameClient.Opcs.TXT ='X'[static], [package]

Text hunk of resources (name.type\tsize\tthen data) TODO Text may be recoded on the reciver side if needed!

Definition at line 985 of file gameClient.java.

final char gameClient.Opcs.UPD ='U'[static], [package]

Request for update about a whole scene.

Definition at line 989 of file gameClient.java.

final char gameClient.Opcs.VIS ='V'`[static]`, `[package]`

Visualisation info for a particular object.

Definition at line 990 of file gameClient.java.

final char gameClient.Opcs.YOU ='Y'`[static]`, `[package]`

Redefining player name if not suitable.

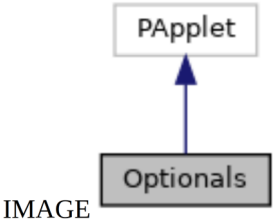
Definition at line 980 of file gameClient.java.

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

src.java/**gameClient.java**

Optionals Class Reference

Inheritance diagram for Optionals:



IMAGE

Classes

class **AbsHighPassFilter**
Absolute High Pass Filter.

class **AbsLowPassFilter**
Absolute Low Pass Filter.

class **Agent**
*Dummy class of **Agent**.*

class **AllLinks**
Different filters of links and other link tools for a (social) network.

class **AndFilter**
AND two filters assembly class.

class **Colorable**

interface **describable**
Generally useable interfaces:

class **DummyBool**
A class for taking an object from a simple logic variable (true-false).

class **DummyDouble**
A class for taking an object from a simple variable of type double.

class **DummyFloat**
A class for taking an object from a simple variable of type float.

class **DummyInt**
Classes for taking an object from a simple variable of type int, boolean, float & double.

class **Frequencies**
This class represens a named histogram of frequencies.

interface **Function2D**
A function of two values in the form of a class - a functor.

class **HighPassFilter**
High Pass Filter.

interface **iColorable**
VISUALISATION INTERFACES:

interface **iDescribable**
Any object which have description as (potentially) long, multi line string.

interface **iLink**
*Network connection/link interface Is **iLink** interface really needed?*

interface **iNamed**
Forcing name available as String (planty of usage)

interface **iNode**
Network node interface "Conn" below is a shortage from Connection.

interface **iPositioned**
*Forcing **posX()** & **posY()** & **posZ()** methods for visualisation and mapping*

interface **iVisLink**
Visualisable network connection.

interface **iVisNode**
Visualisable network node.

class **Link**
class **LinkFactory**
class **LinkFilter**
DEBUG level for network. Visible autside this file!

class **LowPassFilter**
Low Pass Filter.

class **Named**
class **NamedData**
Functions & classes for chart making.

class **Node**
class **NodeList**
class **OrFilter**
OR two filters assembly class.

class **Pair**
COMMON TEMPLATES, INTERFACES AND ABSTRACT CLASSES.

class **PanelOfTextButtons**
A class of a panel that contains many buttons.

class **pointxy**
class **Positioned**
class **Range**
Class of a NAMED range of real (float) numbers.

class **RectArea**
Rectangular screen area class as the basis for various active areas.

class **Sample**
*This class represents a NAMED series of real (float) numbers Should it also be a descendant of the **Range**? ...*

class **settings_bar3d**
class **StateLabel**
*A pseudo-button class that displays the state, not the name, Also ignores **flip_state()** and that changes to state through **set_state()** are "protected".*

class **StateLabelInc**
A button class that increments a state label, possibly undoing the operation of the opposite pair.

class **TextButton**
Rectangular button with text content.

class **TypeAndAbsHighPassFilter**
Special type of filter for efficient visualisation.

class **TypeFilter**
Type of link filter.

class **UniqTextButton**
Unique button.

class **WrTextButton**
A button that remembers the column to which its unique marker is to be saved.

class **WrUniqTextButton**

UniqButton additionally remembers the column to which it is to save its unique marker.

Public Member Functions

void **setup** ()

Dummy setup - additional gr. primitives are tested here:

void **draw** ()

Console only applet! - TESTING TODO.

void **viewAxis** (int startX, int startY, int width, int height)

Visualizes the axes of the coordinate system.

void **viewFrame** (float startX, float startY, int width, int height)

Visualizes a box around the area.

void **viewTicsV** (int startX, int startY, int width, int height, float space)

Draws tics along the vertical axis.

void **viewTicsH** (float startX, float startY, float width, float height, float space)

Draws tics along the horizontal axis.

void **viewScaleV** (**Range** MinMax, int startX, int startY, int width, int height)

Visualizes the limits of the vertical scale NOTE: We're not drawing dashes here yet (tics)

void **viewAsPoints** (**Sample** data, int startD, float startX, float startY, int width, int height, boolean logaritm, **Range** commMinMax, boolean connect)

Visualization of data series as a series of points or a continuous line.

float **viewAsColumns** (**Frequencies** hist, float startX, float startY, int width, int height, boolean logaritm)

Bar visualization of a histogram or something similar.

int **swithbit** (int sou, int pos)

Bit tools.

int **countbits** (int u)

Integer bit counting function.

float **distance** (float X1, float X2, float Y1, float Y2)

Different ways to calculate Euclid distances in 2D (flat and torus)

float **distanceEucl** (float X1, float X2, float Y1, float Y2)

2D Euclidean distance on float numbers Often needed in simulation programs Version compatible with int and double versions

double **distanceEucl** (double X1, double X2, double Y1, double Y2)

2D Euclidean distance on double numbers Sometimes needed in simulation programs Version compatible with int and float versions

float **distanceTorus** (float X1, float X2, float Y1, float Y2, float Xdd, float Ydd)

Euclidean like distance on torus (float numbers) Sometimes needed in simulation programs Version compatible with int and double versions.

double **distanceTorus** (double X1, double X2, double Y1, double Y2, double Xdd, double Ydd)

Euclidean like distance on torus (double numbers) Sometimes needed in simulation programs Version compatible with int and float versions.

double **distanceTorusInt** (int X1, int X2, int Y1, int Y2, int Xdd, int Ydd)

Euclidean like distance on torus (int numbers) Sometimes needed in simulation programs Version compatible with float and double versions.

void **appendTextToFile** (String filename, String text)

Tools for CSV files.

void **createFile** (File f)

Creates a new file including all subfolders in the path.

boolean **obsolete_sketchFullScreen** ()

Old version of full screen Processing application Now the method sketchFullScreen is final in PApplet & cannot be overriden!

float **log10** (float x)

Calculates the base-10 logarithm of a number.

float **log2** (float x)

Calculates the base-2 logarithm of a number.

double **log2** (double x)

Calculates the base-2 logarithm of a number with double precision.

double **log10** (double x)

Calculates the base-10 logarithm of a number with double precision.

int **sign** (int val)

Function for determining the sign of a integer number.

int **sign** (float val)

Function for determining the sign of a float number.

int **sign** (double val)

Function for determining the sign of a double number.

float **upToTresh** (float val, float incr, float tresh)

Function for increasing no more than up to a certain threshold value.

int **whichIsMax** (float v0, float v1, float v2)
Function to find which of the three values is the largest?

int **sqr** (int a)
Functions for easy and READABLE in squaring expressions.

float **sqr** (float a)
A square of an float number.

double **sqr** (double a)
A square of an double number.

void **setupMenu** ()
A function that constructs an example menu.

float **randomGaussPareto** (int Dist)
Functions that improve the use of pseudo-random numbers.

double **RandomXorShift** ()
Function which generates xorshift random value.

void **mousePressed** ()
Mouse click support for buttons.

void **mouseReleased** ()
Mouse button relase support.

void **view_all_areas** ()
View all interface elements.

float **meanArithmetic** (float data[], int offset, int limit)
Various simple statistics for one-dimensional arrays.

double **meanArithmetic** (double data[], int offset, int limit)
Arithmetic mean of the "double" precision data.

double **correlation** (float data1[], float data2[], int offset1, int offset2, int limit)
Pearson's correlation.

double **meanCorrelations** (double data[], int offset, int limit)
Mean of the correlation using Z Unfortunately, the = 1 and = -1 correlations are not transformable, so we cheat a bit.

double **entropyFromHist** (int[] histogram)
Informational entropy from the histogram.

int[] **makeHistogramOfA** (**Agent**[][] Ags, int N, double Min, double Max, **DummyInt** Counter, **DummyDouble** CMin, **DummyDouble** CMax)

A template of making a histogram from an example agent with "A" field It would be difficult to generalize to any field.

int[] **makeHistogramOfA** (**Agent**[] Ags, int N, double Min, double Max, **DummyInt** Counter, **DummyDouble** CMin, **DummyDouble** CMax)

Version for a two-dimensional array of agents.

void **makeRingNet** (**Node**[] nodes, **LinkFactory** linkfac, int neighborhood)
void **makeTorusNet** (**Node**[] nodes, **LinkFactory** links, int neighborhood)
void **makeTorusNet** (**Node**[][] nodes, **LinkFactory** linkfac, int neighborhood)
void **rewireLinksRandomly** (**Node**[] nodes, float probability, boolean reciprocal)
void **rewireLinksRandomly** (**Node**[][] nodes, float probability, boolean reciprocal)
void **makeSmWorldNet** (**Node**[] nodes, **LinkFactory** links, int neighborhood, float probability, boolean reciprocal)
void **makeSmWorldNet** (**Node**[][] nodes, **LinkFactory** links, int neighborhood, float probability, boolean reciprocal)
void **makeImSmWorldNet** (**Node**[][] nodes, **LinkFactory** links, int neighborhood, float probability, boolean reciprocal)
void **makeImSmWorldNet** (**Node**[] nodes, **LinkFactory** links, int neighborhood, float probability, boolean reciprocal)
boolean **inCluster** (**Node**[] cluster, **Node** what)
void **makeScaleFree** (**Node**[] nodes, **LinkFactory** linkfac, int sizeOfFirstCluster, int numberOfNewLinkPerAgent, boolean reciprocal)
void **makeFullNet** (**Node**[] nodes, **LinkFactory** linkfac)
void **makeFullNet** (**Node**[][] nodes, **LinkFactory** linkfac)
void **makeRandomNet** (**Node**[] nodes, **LinkFactory** linkfac, float probability, boolean reciprocal)
void **makeOrphansAdoption** (**Node**[] nodes, **LinkFactory** linkfac, boolean reciprocal)
void **makeRandomNet** (**Node**[][] nodes, **LinkFactory** linkfac, float probability, boolean reciprocal)
void **mouseMoved** ()

Examples for handling mouse events.

void **initVideoExport** (processing.core.PApplet parent, String Name, int Frames)

The beginning of the movie file.

void **FirstVideoFrame** ()

Initial second sequence for title and copyright.

void **NextVideoFrame** ()

Each subsequent frame of the movie.

void **CloseVideo** ()

This is what we call when we want to close the movie file.

void **checkCommandLine** ()

Template handling of program call parameters, if available.

void **dashedLine** (float x0, float y0, float x1, float y1, float[] spacing)

Function for drawing dashed lines.

void **dottedLine** (float x1, float y1, float x2, float y2, float steps)
void **dottedline** (int x1, int y1, int x2, int y2, float dens)
void **dashedline** (float x0, float y0, float x1, float y1, float dens)
void **surround** (int x1, int y1, int x2, int y2)
void **cross** (float x, float y, float cross_width)
void **cross** (int x, int y, int cross_width)
void **baldhead** (int x, int y, int r, float direction)
void **regularpoly** (float x, float y, float radius, int npoints)
void **polygon** (**pointxy**[] lst, int N)
Pair< **pointxy**, **pointxy** > **nearestPoints** (final **pointxy**[] listA, final **pointxy**[] listB)
void **bar3dRGB** (float x, float y, float h, int R, int G, int B, int Shad)
void **dashedline** (float x0, float y0, float x1, float y1, float[] spacing)
void **arrow** (float x1, float y1, float x2, float y2)
void **arrow_d** (int x1, int y1, int x2, int y2, float size, float theta)
void **visualiseLinks1D** (**iVisNode**[] nodes, **LinkFilter** filter, float defX, float defY, float cellside, boolean intMode)

number od=f links visualised last time

void **visualiseLinks2D** (**iVisNode**[] nodes, **LinkFilter** filter, float defX, float defY, float cellside, boolean intMode)
void **visualiseLinks** (**iVisNode**[][] nodes, **LinkFilter** filter, float defX, float defY, float cellside, boolean intMode)
void **settings** ()

Static Public Member Functions

static void **main** (String[] passedArgs)

Package Attributes

final int **LINK_INTENSITY** =2

mandatory globals

final float **MAX_LINK_WEIGHT** =1.0f
final int **MASKBITS** =0xffffffff

Redefine, when smaller width is required.

int **FRAMEFREQ** =10

simulation speed

final float **FLOAT_MAX** =MAX_FLOAT

Some of my older programs show the constant FLOAT_MAX, while MAX_FLOAT is currently available.

MenuBar **myMenu**

Template of the function that allows to construct the window menu in the setup.

final double **denominator** =(double)9223372036854775807L

denominator for xorshift randomizer (why double?)

ArrayList< **RectArea** > **allAreas** = new ArrayList<**RectArea**>()

@Function RandomPareto().

ArrayList< **TextButton** > **allButtons** = new ArrayList<**TextButton**>()

Button list.

int **iniTxButtonSize** =16
The initial size of the button.

int **iniTxButtonCornerRadius** =6
The default rounding of the corners of the buttons.

final float **INF_NOT_EXIST** =Float.MAX_VALUE
MATH INTERFACES:

final **AllLinks allLinks** =new **AllLinks**()
Such type of filter is used very frequently.

int **debug_level** =1
VideoExport **videoExport**
Tool for made video from simulation.

String **copyrightNote** ="(c) W.Borkowski @ ISS University of Warsaw"
*Change it to your copyright. Best in **setup()**.*

settings_bar3d bar3dsett =new **settings_bar3d**()
pointxy bar3dromb [] ={new **pointxy**(),new **pointxy**(),new **pointxy**(),new **pointxy**(),new **pointxy**(),new **pointxy**()}
float **def_arrow_size** =15
float **def_arrow_theta** =PI/6.0f+PI
float **XSPREAD** =0.01f
int **linkCounter** =0
how far is target point of link of type 1, from center of the cell

Static Package Attributes

static long **xl** =123456789L
XOR SHIFT random number generator with flat distribution Apart from the function, it also needs a variable for storing the grain and a constant for storing the denominator.

static int **videoFramesFreq** =0
How many frames per second for the movie. It doesn't have to be the same as in frameRate!

static boolean **videoExportEnabled** =false
Has film making been initiated?

Detailed Description

Definition at line 26 of file Optionals.java.

Member Function Documentation

void Optionals.appendTextToFile (String *filename*, String *text*)

Tools for CSV files.

See: `https://stackoverflow.com/questions/17010222/how-do-i-append-text-to-a-csv-txt-file-in-processing` Appends text to the end of a text file located in the data directory, creates the file if it does not exist. Can be used for big files with lots of rows, existing lines will not be rewritten

Definition at line 617 of file Optionals.java.

void Optionals.arrow (float *x1*, float *y1*, float *x2*, float *y2*)

Definition at line 2991 of file Optionals.java.

void Optionals.arrow_d (int *x1*, int *y1*, int *x2*, int *y2*, float *size*, float *theta*)

Definition at line 2996 of file Optionals.java.

void Optionals.baldhead (int *x*, int *y*, int *r*, float *direction*)

Definition at line 2804 of file Optionals.java.

void Optionals.bar3dRGB (float *x*, float *y*, float *h*, int *R*, int *G*, int *B*, int *Shad*)

Definition at line 2893 of file Optionals.java.

void Optionals.checkCommandLine ()

Template handling of program call parameters, if available.

Parsing command line if available

Definition at line 2620 of file Optionals.java.

void Optionals.CloseVideo ()

This is what we call when we want to close the movie file.

This function adds an ending second sequence with an author's note NOTE: there should be some "force screen update", but not found If you x-click the window while drawing, it is the last frame will probably be incomplete

Definition at line 2598 of file Optionals.java.

double Optionals.correlation (float *data1*[], float *data2*[], int *offset1*, int *offset2*, int *limit*)

Pearson's correlation.

https://pl.wikipedia.org/wiki/Wsp%C3%B3%C5%82czynnik_korelacji_Pearsona

Definition at line 1284 of file Optionals.java.

int Optionals.countbits (int *u*)

Integer bit counting function.

Definition at line 441 of file Optionals.java.

void Optionals.createFile (File *f*)

Creates a new file including all subfolders in the path.

Definition at line 634 of file Optionals.java.

void Optionals.cross (float *x*, float *y*, float *cross_width*)

Definition at line 2792 of file Optionals.java.

void Optionals.cross (int *x*, int *y*, int *cross_width*)

Definition at line 2798 of file Optionals.java.

void Optionals.dashedline (float *x0*, float *y0*, float *x1*, float *y1*, float *dens*)

Definition at line 2779 of file Optionals.java.

void Optionals.dashedLine (float *x0*, float *y0*, float *x1*, float *y1*, float *spacing*[])

Function for drawing dashed lines.

NOTE: uses the Processing specific function lerp() Draw a dashed line with given set of dashes and gap lengths.

Parameters

x0	starting x-coordinate of line.
y0	starting y-coordinate of line.
x1	ending x-coordinate of line.
y1	ending y-coordinate of line.
spacing	is an array giving lengths of dashes and gaps in pixels; an array with values {5, 3, 9, 4} will draw a line with a 5-pixel dash, 3-pixel gap, 9-pixel dash, and 4-pixel gap. if the array has an odd number of entries, the values are recycled, so an array of {5, 3, 2} will draw a line with a 5-pixel dash, 3-pixel gap, 2-pixel dash, 5-pixel gap, 3-pixel dash, and 2-pixel gap, then repeat.

See:
https://processing.org/discourse/beta/num_1202486379.html

Definition at line 2704 of file Optionals.java.

void Optionals.dashedline (float *x0*, float *y0*, float *x1*, float *y1*, float *spacing*[])

Definition at line 2946 of file Optionals.java.

float Optionals.distance (float *X1*, float *X2*, float *Y1*, float *Y2*)

Different ways to calculate Euclid distances in 2D (flat and torus)

Default Euclidean distance on float numbers Often needed in simulation programs Actually the same as dist already shipped in Processing 3.xx

Definition at line 464 of file Optionals.java.

double Optionals.distanceEucl (double *X1*, double *X2*, double *Y1*, double *Y2*)

2D Euclidean distance on double numbers Sometimes needed in simulation programs Version compatible with int and float versions

Definition at line 490 of file Optionals.java.

float Optionals.distanceEucl (float *X1*, float *X2*, float *Y1*, float *Y2*)

2D Euclidean distance on float numbers Often needed in simulation programs Version compatible with int and double versions

Definition at line 477 of file Optionals.java.

double Optionals.distanceTorus (double *X1*, double *X2*, double *Y1*, double *Y2*, double *Xdd*, double *Ydd*)

Euclidean like distance on torus (double numbers) Sometimes needed in simulation programs Version compatible with int and float versions.

Parameters

Xdd	&
Ydd	are the horizontal and vertical perimeter of the torus

Definition at line 521 of file Optionals.java.

float Optionals.distanceTorus (float *X1*, float *X2*, float *Y1*, float *Y2*, float *Xdd*, float *Ydd*)

Euclidean like distance on torus (float numbers) Sometimes needed in simulation programs Version compatible with int and double versions.

Parameters

Xdd	&
Ydd	are the horizontal and vertical perimeter of the torus

Definition at line 505 of file Optionals.java.

double Optionals.distanceTorusInt (int *X1*, int *X2*, int *Y1*, int *Y2*, int *Xdd*, int *Ydd*)

Euclidean like distance on torus (int numbers) Sometimes needed in simulation programs Version compatible with float and double versions.

Parameters

Xdd	&
Ydd	are the horizontal and vertical perimeter of the torus

Definition at line 537 of file Optionals.java.

void Optionals.dottedLine (float *x1*, float *y1*, float *x2*, float *y2*, float *steps*)

Definition at line 2755 of file Optionals.java.

void Optionals.dottedline (int *x1*, int *y1*, int *x2*, int *y2*, float *dens*)

Definition at line 2769 of file Optionals.java.

void Optionals.draw ()

Console only applet! - TESTING TODO.

Console only **draw()** This functins set window visibility to false, and can do anything but drawing :-D

Definition at line 66 of file Optionals.java.

double Optionals.entropyFromHist (int[] *histogram*)

Informational entropy from the histogram.

Definition at line 1369 of file Optionals.java.

void Optionals.FirstVideoFrame ()

Initial second sequence for title and copyright.

Definition at line 2568 of file Optionals.java.

boolean Optionals.inCluster (Node[] *cluster*, Node *what*)

Definition at line 2060 of file Optionals.java.

void Optionals.initVideoExport (processing.core.PApplet *parent*, String *Name*, int *Frames*)

The beginning of the movie file.

Definition at line 2556 of file Optionals.java.

double Optionals.log10 (double *x*)

Calculates the base-10 logarithm of a number with double precision.

Definition at line 682 of file Optionals.java.

float Optionals.log10 (float *x*)

Calculates the base-10 logarithm of a number.

Definition at line 664 of file Optionals.java.

double Optionals.log2 (double *x*)

Calculates the base-2 logarithm of a number with double precision.

Definition at line 676 of file Optionals.java.

float Optionals.log2 (float *x*)

Calculates the base-2 logarithm of a number.

Definition at line 670 of file Optionals.java.

static void Optionals.main (String[] *passedArgs*)`[static]`

Definition at line 3185 of file Optionals.java.

void Optionals.makeFullNet (Node[] *nodes*, LinkFactory *linkfac*)

Full connected network 1D

Definition at line 2138 of file Optionals.java.

void Optionals.makeFullNet (Node *nodes*[][], LinkFactory *linkfac*)

Full connected network 2D

Definition at line 2156 of file Optionals.java.

int [] Optionals.makeHistogramOfA (Agent[] *Ags*, int *N*, double *Min*, double *Max*, DummyInt *Counter*, DummyDouble *CMin*, DummyDouble *CMax*)

Version for a two-dimensional array of agents.

Definition at line 1455 of file Optionals.java.

int [] Optionals.makeHistogramOfA (Agent *Ags*[][], int *N*, double *Min*, double *Max*, DummyInt *Counter*, DummyDouble *CMin*, DummyDouble *CMax*)

A template of making a histogram from an example agent with "A" field It would be difficult to generalize to any field.

Easier you can just rename the field as needed. Version for a two-dimensional array of agents

Parameters

	Ags	Two-dimensional "world" of agents - a two-dimensional array
	N	Number of buckets in the histogram
	Min	Possibility to give the minimum known from other calculations
	Max	Possibility to give the maximum known from other calculations
out	*Counter*	[out] How many values counted in this statistic

out	*CMin*	[out] MIN calculated - for reference
out	*CMax*	[out] MAX calculated - for reference

Definition at line 1401 of file Optionals.java.

void Optionals.makeImSmWorldNet (Node[] *nodes*, LinkFactory *links*, int *neighborhood*, float *probability*, boolean *reciprocal*)

Improved Small World 1D

Definition at line 2055 of file Optionals.java.

void Optionals.makeImSmWorldNet (Node *nodes*[][], LinkFactory *links*, int *neighborhood*, float *probability*, boolean *reciprocal*)

Improved Small World 2D

Definition at line 2050 of file Optionals.java.

void Optionals.makeOrphansAdoption (Node[] *nodes*, LinkFactory *linkfac*, boolean *reciprocal*)

Connect all orphaned nodes with at least one link

Definition at line 2225 of file Optionals.java.

void Optionals.makeRandomNet (Node[] *nodes*, LinkFactory *linkfac*, float *probability*, boolean *reciprocal*)

Randomly connected network 1D

Definition at line 2180 of file Optionals.java.

void Optionals.makeRandomNet (Node *nodes*[][], LinkFactory *linkfac*, float *probability*, boolean *reciprocal*)

Randomly connected network 2D

Definition at line 2268 of file Optionals.java.

void Optionals.makeRingNet (Node[] *nodes*, LinkFactory *linkfac*, int *neighborhood*)

Ring network

Definition at line 1886 of file Optionals.java.

void Optionals.makeScaleFree (Node[] *nodes*, LinkFactory *linkfac*, int *sizeOfFirstCluster*, int *numberOfNewLinkPerAgent*, boolean *reciprocal*)

Scale Free 1D

Definition at line 2072 of file Optionals.java.

void Optionals.makeSmWorldNet (Node[] *nodes*, LinkFactory *links*, int *neighborhood*, float *probability*, boolean *reciprocal*)

Classic Small World 1D

Definition at line 2040 of file Optionals.java.

void Optionals.makeSmWorldNet (Node *nodes*[][], LinkFactory *links*, int *neighborhood*, float *probability*, boolean *reciprocal*)

Classic Small World 2D

Definition at line 2045 of file Optionals.java.

void Optionals.makeTorusNet (Node[] *nodes*, LinkFactory *links*, int *neighborhood*)

Torus lattice 1D - It is alias for Ring net only

Definition at line 1919 of file Optionals.java.

void Optionals.makeTorusNet (Node *nodes*[][], LinkFactory *linkfac*, int *neighborhood*)

Torus lattice 2D

Definition at line 1923 of file Optionals.java.

double Optionals.meanArithmetic (double *data*[], int *offset*, int *limit*)

Arithmetic mean of the "double" precision data.

See: `https://en.wikipedia.org/wiki/Arithmetic_mean`

Definition at line 1267 of file Optionals.java.

float Optionals.meanArithmetic (float *data*[], int *offset*, int *limit*)

Various simple statistics for one-dimensional arrays.

EN: Arithmetic mean of the float data See: `https://en.wikipedia.org/wiki/Arithmetic_mean`

Definition at line 1250 of file Optionals.java.

double Optionals.meanCorrelations (double *data*[], int *offset*, int *limit*)

Mean of the correlation using Z Unfortunately, the = 1 and = -1 correlations are not transformable, so we cheat a bit.

Definition at line 1343 of file Optionals.java.

void Optionals.mouseMoved ()

Examples for handling mouse events.

Mouse movement support. It shouldn't be too time consuming. see: //https://processing.org/reference/mouseMoved_.html

Definition at line 2487 of file Optionals.java.

void Optionals.mousePressed ()

Mouse click support for buttons.

If the program may also respond to clicks differently, this must be taken into account here.

Definition at line 890 of file Optionals.java.

void Optionals.mouseReleased ()

Mouse button relase support.

If the program may also respond to clicks differently, this must be taken into account here.

Definition at line 904 of file Optionals.java.

Pair<pointxy,pointxy> Optionals.nearestPoints (final pointxy[] *listA*, final pointxy[] *listB*)

Definition at line 2859 of file Optionals.java.

void Optionals.NextVideoFrame ()

Each subsequent frame of the movie.

Definition at line 2582 of file Optionals.java.

boolean Optionals.obsolete_sketchFullScreen ()

Old version of full screen Processing application Now the method sketchFullScreen is final in PApplet & cannot be overriden!

Definition at line 651 of file Optionals.java.

void Optionals.polygon (pointxy[] *lst*, int *N*)

Definition at line 2849 of file Optionals.java.

float Optionals.randomGaussPareto (int *Dist*)

Functions that improve the use of pseudo-random numbers.

Function generates pseudo random number with non-flat distribution. When `Dist` is negative, it is Pareto-like, when is positive, it is Gaussian-like

Definition at line 814 of file Optionals.java.

double Optionals.RandomXorShift ()

Function which generates xorshift random value.

Definition at line 842 of file Optionals.java.

void Optionals.regularpoly (float *x*, float *y*, float *radius*, int *npoints*)

Definition at line 2820 of file Optionals.java.

void Optionals.rewireLinksRandomly (Node[] *nodes*, float *probability*, boolean *reciprocal*)

Rewire some connection for Small World 1D

Definition at line 1960 of file Optionals.java.

void Optionals.rewireLinksRandomly (Node *nodes*[][], float *probability*, boolean *reciprocal*)

Rewire some connection for Small World 2D

Definition at line 1999 of file Optionals.java.

void Optionals.settings ()

Definition at line 3184 of file Optionals.java.

void Optionals.setup ()

Dummy setup - additional gr. primitives are tested here:

Definition at line 46 of file Optionals.java.

void Optionals.setupMenu ()

A function that constructs an example menu.

Processig does not see the height of MenuBar added to Window!

Definition at line 781 of file Optionals.java.

int Optionals.sign (double *val*)

Function for determining the sign of a double number.

Definition at line 713 of file Optionals.java.

int Optionals.sign (float *val*)

Function for determining the sign of a float number.

Definition at line 705 of file Optionals.java.

int Optionals.sign (int *val*)

Function for determining the sign of a integer number.

Definition at line 697 of file Optionals.java.

double Optionals.sqr (double *a*)

A square of an double number.

Definition at line 757 of file Optionals.java.

float Optionals.sqr (float *a*)

A square of an float number.

Definition at line 751 of file Optionals.java.

int Optionals.sqr (int *a*)

Functions for easy and READABLE in squaring expressions.

A square of an int number

Definition at line 745 of file Optionals.java.

void Optionals.surround (int *x1*, int *y1*, int *x2*, int *y2*)

Definition at line 2784 of file Optionals.java.

int Optionals.swithbit (int *sou*, int *pos*)

Bit tools.

Function for mutating integer bits

Definition at line 426 of file Optionals.java.

float Optionals.upToTresh (float *val*, float *incr*, float *tresh*)

Function for increasing no more than up to a certain threshold value.

Definition at line 721 of file Optionals.java.

void Optionals.view_all_areas ()

View all interface elements.

Should be called in **draw()** or in an event handlers.

Definition at line 918 of file Optionals.java.

float Optionals.viewAsColumns (Frequencies *hist*, float *startX*, float *startY*, int *width*, int *height*, boolean *logaritm*)

Bar visualization of a histogram or something similar.

Parameters

hist	Data source. The object containing the data to be visualized
startX	The horizontal starting point of the display area
startY	The vertical starting point of the display area
width	The width of the display area
height	The height of the display area
logaritm	Should the data be transformed by logarith?

Definition at line 388 of file Optionals.java.

void Optionals.viewAsPoints (Sample *data*, int *startD*, float *startX*, float *startY*, int *width*, int *height*, boolean *logaritm*, Range *commMinMax*, boolean *connect*)

Visualization of data series as a series of points or a continuous line.

Parameters

data	Data source. The object containing the data to be visualized
startD	Data starting point, or end-to-end number if negative
startX	The horizontal starting point of the display area
startY	The vertical starting point of the display area
width	The width of the display area
height	Height of the display area
logaritm	Should the data be transformed by logarith?
commMinMax	Optionally common **Range** for multiple series or null
connect	Should data points be combined into a single line?

Definition at line 314 of file Optionals.java.

void Optionals.viewAxis (int *startX*, int *startY*, int *width*, int *height*)

Visualizes the axes of the coordinate system.

Definition at line 270 of file Optionals.java.

void Optionals.viewFrame (float *startX*, float *startY*, int *width*, int *height*)

Visualizes a box around the area.

Definition at line 280 of file Optionals.java.

void Optionals.viewScaleV (Range *MinMax*, int *startX*, int *startY*, int *width*, int *height*)

Visualizes the limits of the vertical scale NOTE: We're not drawing dashes here yet (tics)

Definition at line 304 of file Optionals.java.

void Optionals.viewTicsH (float *startX*, float *startY*, float *width*, float *height*, float *space*)

Draws tics along the horizontal axis.

Definition at line 296 of file Optionals.java.

void Optionals.viewTicsV (int *startX*, int *startY*, int *width*, int *height*, float *space*)

Draws tics along the vertical axis.

Definition at line 289 of file Optionals.java.

void Optionals.visualiseLinks (iVisNode *nodes*[][], LinkFilter *filter*, float *defX*, float *defY*, float *cellside*, boolean *intMode*)

Definition at line 3134 of file Optionals.java.

void Optionals.visualiseLinks1D (iVisNode[] *nodes*, LinkFilter *filter*, float *defX*, float *defY*, float *cellside*, boolean *intMode*)

number od=f links visualised last time

Definition at line 3047 of file Optionals.java.

void Optionals.visualiseLinks2D (iVisNode[] *nodes*, LinkFilter *filter*, float *defX*, float *defY*, float *cellside*, boolean *intMode*)

Definition at line 3087 of file Optionals.java.

int Optionals.whichIsMax (float *v0*, float *v1*, float *v2*)

Function to find which of the three values is the largest?

Definition at line 728 of file Optionals.java.

Member Data Documentation

ArrayList<RectArea> Optionals.allAreas = new ArrayList<RectArea>() [package]

@Function RandomPareto().

It generates pareto distribution from flat distribution See: `https://math.stackexchange.com/questions/1777367/how-to-generate-a-random-number-from-a-pareto-distribution` Not tested!!! TODO! "Active rectangles" - proprietary application interface system in Processing @Author Wojciech Borkowski List of areas to be displayed

Definition at line 882 of file Optionals.java.

ArrayList<TextButton> Optionals.allButtons = new ArrayList<TextButton>() [package]

Button list.

Definition at line 883 of file Optionals.java.

final AllLinks Optionals.allLinks =new AllLinks() [package]

Such type of filter is used very frequently.

Definition at line 1632 of file Optionals.java.

pointxy Optionals.bar3dromb[] ={new pointxy(),new pointxy(),new pointxy(),new pointxy(),new pointxy(),new pointxy()}[package]

Definition at line 2891 of file Optionals.java.

settings_bar3d Optionals.bar3dsett =new settings_bar3d()[package]

Definition at line 2889 of file Optionals.java.

String Optionals.copyrightNote ="(c) W.Borkowski @ ISS University of Warsaw"[package]

Change it to your copyright. Best in **setup()**.

Definition at line 2552 of file Optionals.java.

int Optionals.debug_level =1[package]

Definition at line 1786 of file Optionals.java.

float Optionals.def_arrow_size =15[package]

Definition at line 2988 of file Optionals.java.

float Optionals.def_arrow_theta =PI/6.0f+PI[package]

Definition at line 2989 of file Optionals.java.

final double Optionals.denominator =(double)9223372036854775807L[package]

denominator for xorshift randomizer (why double?)

Definition at line 838 of file Optionals.java.

final float Optionals.FLOAT_MAX =MAX_FLOAT[package]

Some of my older programs show the constant FLOAT_MAX, while MAX_FLOAT is currently available.

Definition at line 694 of file Optionals.java.

int Optionals.FRAMEFREQ =10[package]

simulation speed

Definition at line 36 of file Optionals.java.

final float Optionals.INF_NOT_EXIST =Float.MAX_VALUE[package]

MATH INTERFACES:

Missing value marker

Definition at line 1557 of file Optionals.java.

int Optionals.iniTxButtonCornerRadius =6[package]

The default rounding of the corners of the buttons.

Definition at line 886 of file Optionals.java.

int Optionals.iniTxButtonSize =16[package]

The initial size of the button.

Definition at line 885 of file Optionals.java.

final int Optionals.LINK_INTENSITY =2[package]

mandatory globals

Definition at line 33 of file Optionals.java.

int Optionals.linkCounter =0[package]

how far is target point of link of type 1, from center of the cell

Definition at line 3033 of file Optionals.java.

final int Optionals.MASKBITS =0xffffffff[package]

Redefine, when smaller width is required.

Definition at line 35 of file Optionals.java.

final float Optionals.MAX_LINK_WEIGHT =1.0f[package]

Definition at line 34 of file Optionals.java.

MenuBar Optionals.myMenu[package]

Template of the function that allows to construct the window menu in the setup.

Unfortunately, this breaks the calculation of the built-in variable height in Processing! Handle to menu.

Definition at line 777 of file Optionals.java.

VideoExport Optionals.videoExport[package]

Tool for made video from simulation.

See:
`http://funprogramming.org/VideoExport-for-Processing/example`

`s/basic/basic.pde` USAGE: Apart from the "hamoid" library, you also need to install the ffmpeg program to make it work !!!

Here we import the necessary library containing the VideoExport class USAGE This initVideoExport function call must be in **setup()** for the Video module to work: initVideoExport(this,FileName,Frames)); // The VideoExport class must have access to // the Processing application object // It's best to run at the end of the **setup()**. // NOTE !!!: The window must be EVEN sizes We call Next Video Frame for each frame of the movie, most often in the draw () function: **NextVideoFrame()**;//Video frame

... and at the end of the video we call CloseVideo: **CloseVideo()**; // Ideally in exit () Obiekt KLASY z dodatkowej biblioteki - trzeba zainstalowaÄ‡

Definition at line 2546 of file Optionals.java.

boolean Optionals.videoExportEnabled =false`[static]`, `[package]`

Has film making been initiated?

Definition at line 2550 of file Optionals.java.

int Optionals.videoFramesFreq =0`[static]`, `[package]`

How many frames per second for the movie. It doesn't have to be the same as in frameRate!

Definition at line 2548 of file Optionals.java.

long Optionals.xl =123456789L`[static]`, `[package]`

XOR SHIFT random number generator with flat distribution Apart from the function, it also needs a variable for storing the grain and a constant for storing the denominator.

See:
`http://www.javamex.com/tutorials/random_numbers/xorshift.shtml#.WT6NEzekKXI` seed for xorshift randomizer

Definition at line 837 of file Optionals.java.

float Optionals.XSPREAD =0.01f`[package]`

Definition at line 3032 of file Optionals.java.

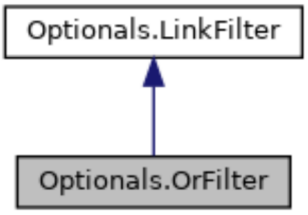
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.OrFilter Class Reference

OR two filters assembly class.

Inheritance diagram for Optionals.OrFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

OrFilter (**LinkFilter** aa, **LinkFilter** bb)

Package Attributes

LinkFilter a
LinkFilter b

Detailed Description

OR two filters assembly class.

A class for logically joining two filters with the OR operator.

Definition at line 1660 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.OrFilter.OrFilter (LinkFilter *aa*, LinkFilter *bb*) [package]

Definition at line 1664 of file Optionals.java.

Member Function Documentation

boolean Optionals.OrFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1665 of file Optionals.java.

Member Data Documentation

LinkFilter Optionals.OrFilter.a [package]

Definition at line 1662 of file Optionals.java.

LinkFilter Optionals.OrFilter.b [package]

Definition at line 1663 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.Pair< A, B > Class Template Reference

COMMON TEMPLATES, INTERFACES AND ABSTRACT CLASSES.

Public Member Functions

Pair (A **a**, B **b**)

Public Attributes

final A **a**
final B **b**

Detailed Description

COMMON TEMPLATES, INTERFACES AND ABSTRACT CLASSES.

Templates: Simple version of **Pair** template useable for returning a pair of values

Definition at line 1520 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.Pair< A, B >.Pair (A *a*, B *b*)

Definition at line 1524 of file Optionals.java.

Member Data Documentation

final A Optionals.Pair< A, B >.a

Definition at line 1521 of file Optionals.java.

final B Optionals.Pair< A, B >.b

Definition at line 1522 of file Optionals.java.

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

src.java/**Optionals.java**

ABMTemplate.PairOfInt Class Reference

Simple version of Pair containing a pair of integers.

Public Member Functions

PairOfInt (int **a**, int **b**)

Public Attributes

final int **a**
final int **b**

Detailed Description

Simple version of Pair containing a pair of integers.

Definition at line 523 of file ABMTemplate.java.

Constructor & Destructor Documentation

ABMTemplate.PairOfInt.PairOfInt (int *a*, int *b*)

Definition at line 528 of file ABMTemplate.java.

Member Data Documentation

final int ABMTemplate.PairOfInt.a

Definition at line 525 of file ABMTemplate.java.

final int ABMTemplate.PairOfInt.b

Definition at line 526 of file ABMTemplate.java.

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

src.java/**ABMTemplate.java**

CATemplate.PairOfInt Class Reference

Simple version of Pair containing a pair of integers.

Public Member Functions

PairOfInt (int **a**, int **b**)

Public Attributes

final int **a**
final int **b**

Detailed Description

Simple version of Pair containing a pair of integers.

Definition at line 558 of file CATemplate.java.

Constructor & Destructor Documentation

CATemplate.PairOfInt.PairOfInt (int *a*, int *b*)

Definition at line 563 of file CATemplate.java.

Member Data Documentation

final int CATemplate.PairOfInt.a

Definition at line 560 of file CATemplate.java.

final int CATemplate.PairOfInt.b

Definition at line 561 of file CATemplate.java.

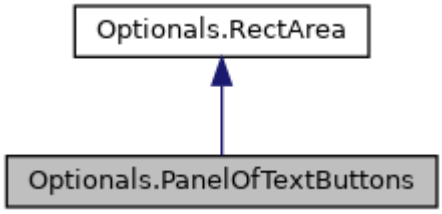
The documentation for this class was generated from the following file:

src.java/**CATemplate.java**

Optionals.PanelOfTextButtons Class Reference

A class of a panel that contains many buttons.

Inheritance diagram for Optionals.PanelOfTextButtons:



IMAGE

Public Member Functions

void **view** ()

Area display function.

void **add** (**TextButton** but)

Filling the panel with buttons.

Package Functions

PanelOfTextButtons (float iX1, float iY1, float iX2, float iY2)

Constructor.

Package Attributes

ArrayList< **TextButton** > **list**

Detailed Description

A class of a panel that contains many buttons.

Definition at line 962 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.PanelOfTextButtons.PanelOfTextButtons (float *iX1*, float *iY1*, float *iX2*, float *iY2*) [package]

Constructor.

Requires data on the corners of the area.

Definition at line 968 of file Optionals.java.

Member Function Documentation

void Optionals.PanelOfTextButtons.add (TextButton *but*)

Filling the panel with buttons.

Definition at line 985 of file Optionals.java.

void Optionals.PanelOfTextButtons.view ()

Area display function.

Reimplemented from **Optionals.RectArea** (*p.189*).

Definition at line 975 of file Optionals.java.

Member Data Documentation

ArrayList<TextButton> Optionals.PanelOfTextButtons.list [package]

Definition at line 964 of file Optionals.java.

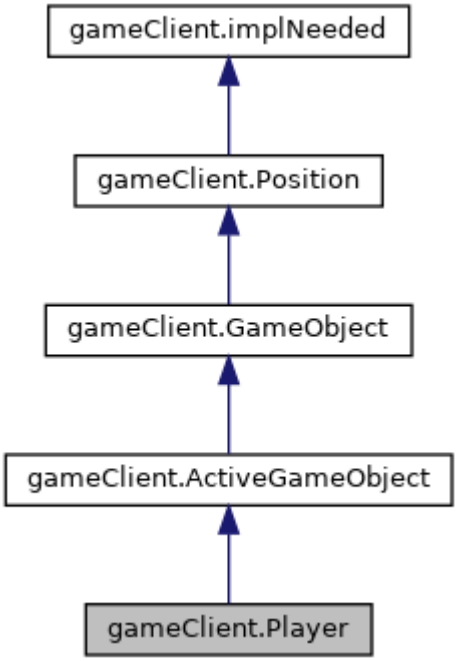
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

gameClient.Player Class Reference

Representation of generic player.

Inheritance diagram for gameClient.Player:



IMAGE

Public Member Functions

boolean **setState** (String field, String val)

Interface for changing the state of the game object over the network (mostly)

String **sayState** ()

The function creates a message block from those object state elements that have change flags (for network streaming)

String **info** (int level)

It can make string info about object.

Package Functions

Player (Client iniClient, String iniName, float iniX, float iniY, float iniZ, float iniRadius)

constructor

Package Attributes

float **score** =0

Result.

Client **netLink**

Network connection to client application.

int **indexInGameWorld** =-1

Detailed Description

Representation of generic player.

Definition at line 697 of file gameClient.java.

Constructor & Destructor Documentation

gameClient.Player.Player (Client *iniClient*, String *iniName*, float *iniX*, float *iniY*, float *iniZ*, float *iniRadius*) [package]

constructor

Definition at line 704 of file gameClient.java.

Member Function Documentation

String gameClient.Player.info (int *level*)

It can make string info about object.

'level' is level of details, when 0 means "name only".

Reimplemented from **gameClient.GameObject** (*p.78*).

Definition at line 734 of file gameClient.java.

String gameClient.Player.sayState ()

The function creates a message block from those object state elements that have change flags (for network streaming)

Returns

: Ready to send list of all changes made on object (based on flags)

Reimplemented from **gameClient.ActiveGameObject** (*p.27*).

Definition at line 724 of file gameClient.java.

boolean gameClient.Player.setState (String *field*, String *val*)

Interface for changing the state of the game object over the network (mostly)

Returns

: true if field is found

Reimplemented from **gameClient.ActiveGameObject** (*p.27*).

Definition at line 711 of file gameClient.java.

Member Data Documentation

int gameClient.Player.indexInGameWorld =-1[package]

Definition at line 701 of file gameClient.java.

Client gameClient.Player.netLink[package]

Network connection to client application.

Definition at line 700 of file gameClient.java.

float gameClient.Player.score =0[package]

Result.

Definition at line 699 of file gameClient.java.

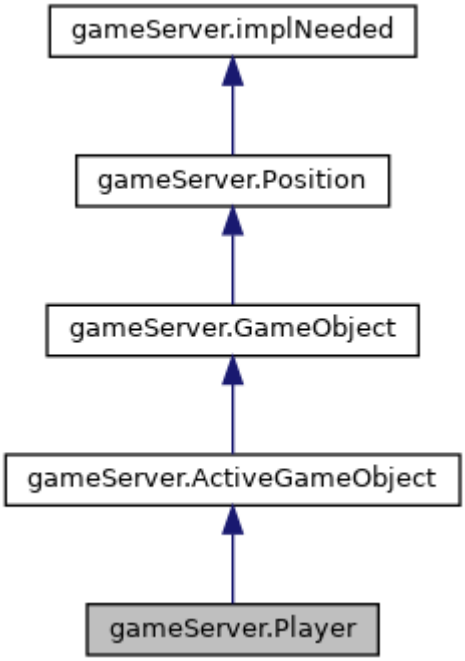
The documentation for this class was generated from the following file:

src.java/**gameClient.java**

gameServer.Player Class Reference

Representation of generic player.

Inheritance diagram for gameServer.Player:



IMAGE

Public Member Functions

boolean **setState** (String field, String val)

Interface for changing the state of the game object over the network (mostly)

String **sayState** ()

The function creates a message block from those object state elements that have change flags (for network streaming)

String **info** (int level)

It can make string info about object.

Package Functions

Player (Client iniClient, String iniName, float iniX, float iniY, float iniZ, float iniRadius)

constructor

Package Attributes

float **score** =0
Client **netLink**
int **indexInGameWorld** =-1

Detailed Description

Representation of generic player.

Definition at line 415 of file gameServer.java.

Constructor & Destructor Documentation

gameServer.Player.Player (Client *iniClient*, String *iniName*, float *iniX*, float *iniY*, float *iniZ*, float *iniRadius*) [package]

constructor

Definition at line 422 of file gameServer.java.

Member Function Documentation

String gameServer.Player.info (int *level*)

It can make string info about object.

'level' is level of details, when 0 means "name only".

Reimplemented from **gameServer.GameObject** (*p.75*).

Definition at line 452 of file gameServer.java.

String gameServer.Player.sayState ()

The function creates a message block from those object state elements that have change flags (for network streaming)

Returns

: Ready to send list of all changes made on object (based on flags)

Reimplemented from **gameServer.ActiveGameObject** (*p.25*).

Definition at line 442 of file gameServer.java.

boolean gameServer.Player.setState (String *field*, String *val*)

Interface for changing the state of the game object over the network (mostly)

Returns

: true if field is found

Reimplemented from **gameServer.ActiveGameObject** (*p.25*).

Definition at line 429 of file gameServer.java.

Member Data Documentation

int gameServer.Player.indexInGameWorld =-1 [package]

Definition at line 419 of file gameServer.java.

Client gameServer.Player.netLink [package]

Definition at line 418 of file gameServer.java.

float gameServer.Player.score =0 [package]

Definition at line 417 of file gameServer.java.

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

src.java/**gameServer.java**

Optionals.pointxy Class Reference

Package Functions

pointxy ()
pointxy (float ix, float iy)

Package Attributes

float **x**
float **y**

Detailed Description

Definition at line 2835 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.pointxy.pointxy ()[package]

Definition at line 2839 of file Optionals.java.

Optionals.pointxy.pointxy (float *ix*, float *iy*)[package]

Definition at line 2843 of file Optionals.java.

Member Data Documentation

float Optionals.pointxy.x[package]

Definition at line 2837 of file Optionals.java.

float Optionals.pointxy.y[package]

Definition at line 2838 of file Optionals.java.

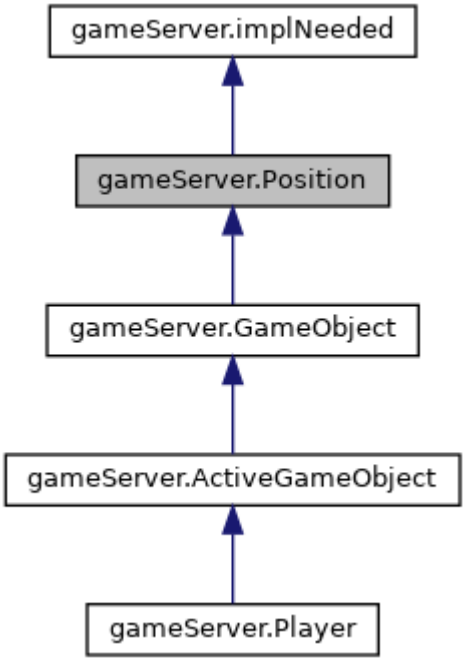
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

gameServer.Position Class Reference

Representation of 3D position in the game world However, the value of Z is not always used.

Inheritance diagram for gameServer.Position:



IMAGE

Public Member Functions

float **distance2D** (**Position** toWhat)

2D distance calculation

float **distance3D** (**Position** toWhat)

3D distance calculation

Package Functions

Position (float iniX, float iniY, float iniZ)

constructor

Package Attributes

float **X**
float **Y**
float **Z**

Detailed Description

Representation of 3D position in the game world However, the value of Z is not always used.

Definition at line 279 of file gameServer.java.

Constructor & Destructor Documentation

gameServer.Position.Position (float *iniX*, float *iniY*, float *iniZ*) [package]

constructor

Definition at line 285 of file gameServer.java.

Member Function Documentation

float gameServer.Position.distance2D (Position *toWhat*)

2D distance calculation

Definition at line 290 of file gameServer.java.

float gameServer.Position.distance3D (Position *toWhat*)

3D distance calculation

Definition at line 296 of file gameServer.java.

Member Data Documentation

float gameServer.Position.X [package]

Definition at line 281 of file gameServer.java.

float gameServer.Position.Y [package]

Definition at line 281 of file gameServer.java.

float gameServer.Position.Z [package]

Definition at line 282 of file gameServer.java.

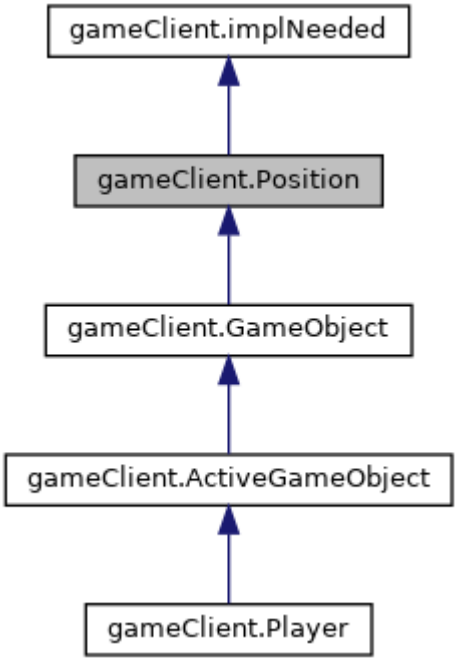
The documentation for this class was generated from the following file:

src.java/**gameServer.java**

gameClient.Position Class Reference

Representation of 3D position in the game world However, the value of Z is not always used.

Inheritance diagram for gameClient.Position:



IMAGE

Public Member Functions

float **distance2D** (**Position** toWhat)

2D distance calculation

float **distance3D** (**Position** toWhat)

3D distance calculation

Package Functions

Position (float iniX, float iniY, float iniZ)

constructor

Package Attributes

float **X**

float **Y**

2D coordinates

float **Z**

Even on a 2D board, objects can pass each other without collision.

Detailed Description

Representation of 3D position in the game world However, the value of Z is not always used.
Definition at line 561 of file gameClient.java.

Constructor & Destructor Documentation

gameClient.Position.Position (float *iniX*, float *iniY*, float *iniZ*) [package]

constructor
Definition at line 567 of file gameClient.java.

Member Function Documentation

float gameClient.Position.distance2D (Position *toWhat*)

2D distance calculation
Definition at line 572 of file gameClient.java.

float gameClient.Position.distance3D (Position *toWhat*)

3D distance calculation
Definition at line 578 of file gameClient.java.

Member Data Documentation

float gameClient.Position.X [package]

Definition at line 563 of file gameClient.java.

float gameClient.Position.Y [package]

2D coordinates
Definition at line 563 of file gameClient.java.

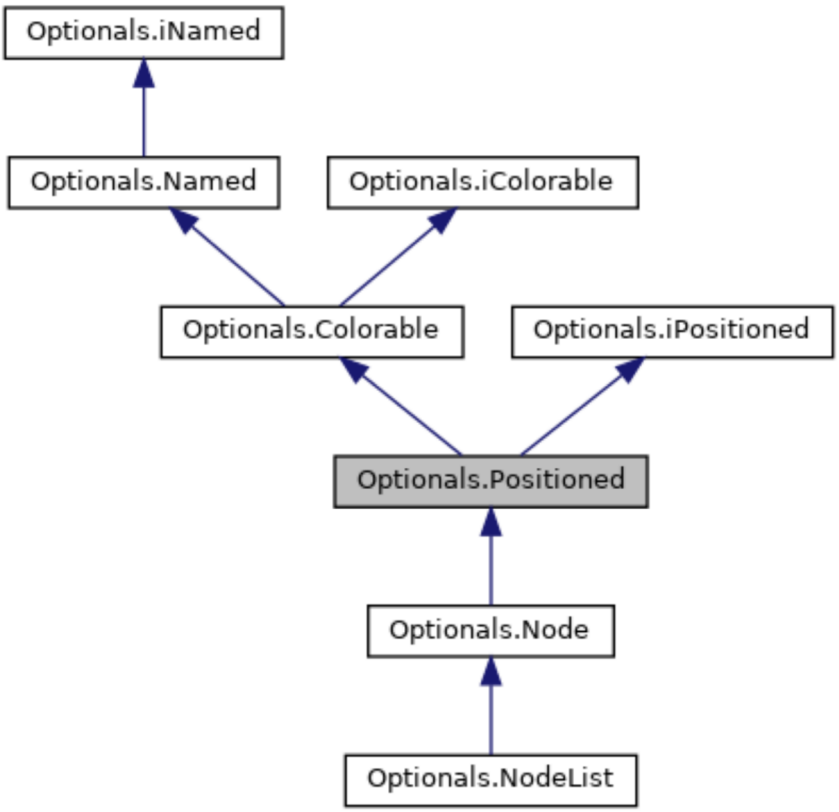
float gameClient.Position.Z [package]

Even on a 2D board, objects can pass each other without collision.
Definition at line 564 of file gameClient.java.

The documentation for this class was generated from the following file:
src.java/**gameClient.java**

Optionals.Positioned Class Reference

Inheritance diagram for Optionals.Positioned:



IMAGE

Public Member Functions

float **posX** ()

*INFO: Forcing **posX()** & **posY()** & **posZ()** methods for visualisation and mapping*

float **posY** ()

float **posZ** ()

Detailed Description

Definition at line 1816 of file Optionals.java.

Member Function Documentation

float Optionals.Positioned.posX ()

INFO: Forcing **posX()** & **posY()** & **posZ()** methods for visualisation and mapping

Implements **Optionals.iPositioned** (*p.113*).

Definition at line 1818 of file Optionals.java.

float Optionals.Positioned.posY ()

Implements **Optionals.iPositioned** (*p.113*).
Definition at line 1819 of file Optionals.java.

float Optionals.Positioned.posZ ()

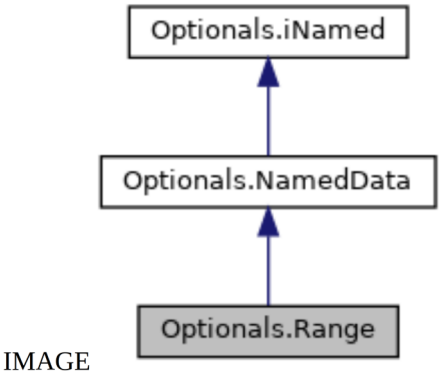
Implements **Optionals.iPositioned** (*p.113*).
Definition at line 1820 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.Range Class Reference

Class of a NAMED range of real (float) numbers.

Inheritance diagram for Optionals.Range:



IMAGE

Public Member Functions

void **addValue** (float value)

Adding a value to a range can make it wider.

Package Functions

Range (String Name)

Constructor need only a name.

Package Attributes

float **min** =+Float.MAX_VALUE

Current minimal value.

float **max** =-Float.MAX_VALUE

Current maximal value.

Detailed Description

Class of a NAMED range of real (float) numbers.

Definition at line 90 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.Range.Range (String *Name*)[package]

Constructor need only a name.

Definition at line 96 of file Optionals.java.

Member Function Documentation

void Optionals.Range.addValue (float *value*)

Adding a value to a range can make it wider.

Definition at line 99 of file Optionals.java.

Member Data Documentation

float Optionals.Range.max =-Float.MAX_VALUE [package]

Current maximal value.

Definition at line 93 of file Optionals.java.

float Optionals.Range.min =+Float.MAX_VALUE [package]

Current minimal value.

Definition at line 92 of file Optionals.java.

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

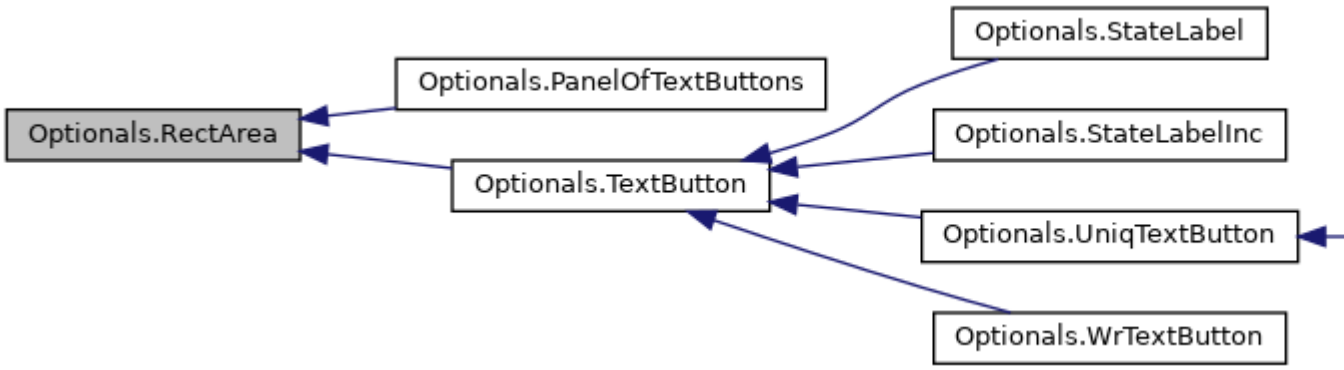
src.java/**Optionals.java**

Optionals.RectArea Class Reference

Rectangular screen area class as the basis for various active areas.

Inheritance diagram for Optionals.RectArea:

IMAGE



Public Member Functions

void **view** ()

Area display function.

boolean **hitted** (int x, int y)

The function of checking if you click on an area.

Package Functions

RectArea (float iX1, float iY1, float iX2, float iY2)

Constructor.

Package Attributes

int **x1**
int **y1**
int **x2**
int **y2**

Corners of the area.

int **back**

Colour of background.

Detailed Description

Rectangular screen area class as the basis for various active areas.

Definition at line 927 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.RectArea.RectArea (float *iX1*, float *iY1*, float *iX2*, float *iY2*) [package]

Constructor.

Requires data on the corners of the area.

Definition at line 934 of file Optionals.java.

Member Function Documentation

boolean Optionals.RectArea.hitted (int *x*, int *y*)

The function of checking if you click on an area.

Definition at line 954 of file Optionals.java.

void Optionals.RectArea.view ()

Area display function.

Reimplemented in **Optionals.StateLabel** (*p.197*), **Optionals.TextButton** (*p.202*), and **Optionals.PanelOfTextButtons** (*p.172*).

Definition at line 945 of file Optionals.java.

Member Data Documentation

int Optionals.RectArea.back [package]

Colour of background.

Definition at line 930 of file Optionals.java.

int Optionals.RectArea.x1 [package]

Definition at line 929 of file Optionals.java.

int Optionals.RectArea.x2 [package]

Definition at line 929 of file Optionals.java.

int Optionals.RectArea.y1 [package]

Definition at line 929 of file Optionals.java.

int Optionals.RectArea.y2 [package]

Corners of the area.

Definition at line 929 of file Optionals.java.

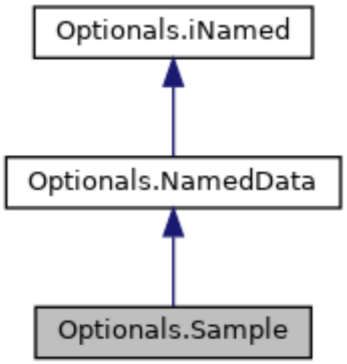
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.Sample Class Reference

This class represents a NAMED series of real (float) numbers Should it also be a descendant of the **Range**? ...

Inheritance diagram for Optionals.Sample:



IMAGE

Public Member Functions

void **addValue** (float value)

Adding values to a series immediately updates the base stats.

int **numOfElements** ()

Number of recorded values Together with empty entries equal to INF_NOT_EXIST.

void **reset** ()

Ready to start collecting data again.

float **getMin** ()

Secured reading of the minimum.

float **getMax** ()

Secured reading of the maximum.

float **getMean** ()

Secured reading of the the mean.

float **getStdDev** ()

Secured reading of the standard deviation.

Package Functions

Sample (String Name)

Constructor need only a name.

Package Attributes

FloatList **data** =null

int **count** =0
How much data has been entered (not counting INF_NOT_EXIST)

float **min** =+Float.MAX_VALUE
Current minimal value.

int **whmin** =-1
Position of the current minimal value.

float **max** =-Float.MAX_VALUE
Current maximal value.

int **whmax** =-1
Position of the current maximal value.

double **sum** =0
The current sum of values.

Detailed Description

This class represents a NAMED series of real (float) numbers Should it also be a descendant of the **Range**? ...

Or at least implements the same interface? TODO?

Definition at line 116 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.Sample.Sample (String *Name*)[package]

Constructor need only a name.

Definition at line 129 of file Optionals.java.

Member Function Documentation

void Optionals.Sample.addValue (float *value*)

Adding values to a series immediately updates the base stats.

Definition at line 132 of file Optionals.java.

float Optionals.Sample.getMax ()

Secured reading of the maximum.

Definition at line 177 of file Optionals.java.

float Optionals.Sample.getMean ()

Secured reading of the the mean.
Definition at line 184 of file Optionals.java.

float Optionals.Sample.getMin ()

Secured reading of the minimum.
Definition at line 170 of file Optionals.java.

float Optionals.Sample.getStdDev ()

Secured reading of the standard deviation.
Definition at line 191 of file Optionals.java.

int Optionals.Sample.numOfElements ()

Number of recorded values Together with empty entries equal to INF_NOT_EXIST.
Definition at line 155 of file Optionals.java.

void Optionals.Sample.reset ()

Ready to start collecting data again.
Definition at line 158 of file Optionals.java.

Member Data Documentation

int Optionals.Sample.count =0 [package]

How much data has been entered (not counting INF_NOT_EXIST)
Definition at line 121 of file Optionals.java.

FloatList Optionals.Sample.data =null [package]

Definition at line 118 of file Optionals.java.

float Optionals.Sample.max =-Float.MAX_VALUE [package]

Current maximal value.
Definition at line 124 of file Optionals.java.

float Optionals.Sample.min =+Float.MAX_VALUE [package]

Current minimal value.
Definition at line 122 of file Optionals.java.

double Optionals.Sample.sum =0[package]

The current sum of values.

Definition at line 126 of file Optionals.java.

int Optionals.Sample.whmax =-1[package]

Position of the current maximal value.

Definition at line 125 of file Optionals.java.

int Optionals.Sample.whmin =-1[package]

Position of the current minimal value.

Definition at line 123 of file Optionals.java.

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

src.java/**Optionals.java**

Optionals.settings_bar3d Class Reference

Package Attributes

int **a** =10
int **b** =6
int **c** =6
int **wire** =color(255,255,255)
int **back** =color(0,0,0)

Detailed Description

Definition at line 2880 of file Optionals.java.

Member Data Documentation

int Optionals.settings_bar3d.a =10 [package]

Definition at line 2882 of file Optionals.java.

int Optionals.settings_bar3d.b =6 [package]

Definition at line 2883 of file Optionals.java.

int Optionals.settings_bar3d.back =color(0,0,0) [package]

Definition at line 2886 of file Optionals.java.

int Optionals.settings_bar3d.c =6 [package]

Definition at line 2884 of file Optionals.java.

int Optionals.settings_bar3d.wire =color(255,255,255) [package]

Definition at line 2885 of file Optionals.java.

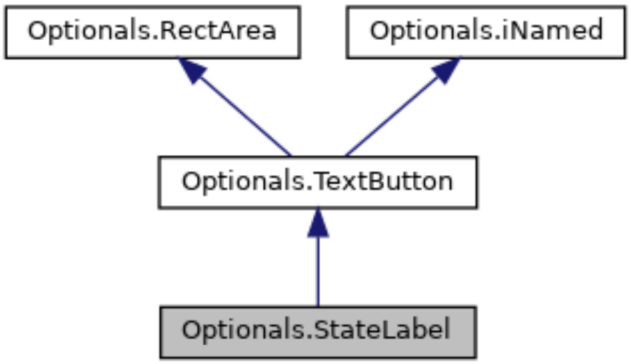
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.StateLabel Class Reference

A pseudo-button class that displays the state, not the name, Also ignores **flip_state()** and that changes to state through **set_state()** are "protected".

Inheritance diagram for Optionals.StateLabel:



IMAGE

Public Member Functions

void **view** ()
Area display function.

void **allow** ()
A function that allows you to change the state.

void **flip_state** (boolean visual)
Specific for the class.

void **set_state** (int new_state, boolean visual)
Specific for the class. It uses allowChng field and then clears it.

Package Functions

StateLabel (int iState, String iTitle, float iX1, float iY1, float iX2, float iY2)
Constructor.

Private Attributes

boolean **allowChng**
*Normally using **set_state()** in this class does not change anything.*

Additional Inherited Members

Detailed Description

A pseudo-button class that displays the state, not the name, Also ignores **flip_state()** and that changes to state through **set_state()** are "protected".

Definition at line 1069 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.StateLabel.StateLabel (int *iState*, String *iTitle*, float *iX1*, float *iY1*, float *iX2*, float *iY2*)[package]

Constructor.

Requires data on the corners of the area, text "title" and initial state.

Definition at line 1079 of file Optionals.java.

Member Function Documentation

void Optionals.StateLabel.allow ()

A function that allows you to change the state.

Definition at line 1103 of file Optionals.java.

void Optionals.StateLabel.flip_state (boolean *visual*)

Specific for the class.

It does not change the state by flip, at most it repeats the display - although it is probably useless

Reimplemented from **Optionals.TextButton** (*p.202*).

Definition at line 1111 of file Optionals.java.

void Optionals.StateLabel.set_state (int *new_state*, boolean *visual*)

Specific for the class. It uses allowChng field and then clears it.

Reimplemented from **Optionals.TextButton** (*p.202*).

Definition at line 1118 of file Optionals.java.

void Optionals.StateLabel.view ()

Area display function.

Reimplemented from **Optionals.TextButton** (*p.202*).

Definition at line 1086 of file Optionals.java.

Member Data Documentation

boolean Optionals.StateLabel.allowChng[private]

Normally using **set_state()** in this class does not change anything.

You have to set this field, which clears always after changing, so only code working on objects of this class can do it, but code working on base class can't :-)

Definition at line 1075 of file Optionals.java.

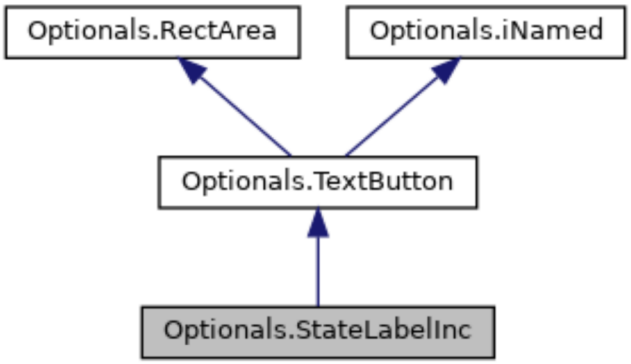
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.StateLabelInc Class Reference

A button class that increments a state label, possibly undoing the operation of the opposite pair.

Inheritance diagram for Optionals.StateLabelInc:



IMAGE

Public Member Functions

void **flip_state** (boolean visual)

Class-specific overlay of the inherited method.

void **decrement** (boolean visual)

The method that undoes state increments, i.e. decreases the "counter".

Package Functions

StateLabelInc (String iTitle, float iX1, float iY1, float iX2, float iY2, **StateLabel** iTarget, **StateLabelInc** iOpponent)

Constructor.

Package Attributes

StateLabel target
StateLabelInc opponent

Additional Inherited Members

Detailed Description

A button class that increments a state label, possibly undoing the operation of the opposite pair.

Definition at line 1137 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.StateLabelInc.StateLabelInc (String *iTitle*, float *iX1*, float *iY1*, float *iX2*, float *iY2*, StateLabel *iTarget*, StateLabelInc *iOpponent*) [package]

Constructor.

Requires data on the corners of the area, text "title" and connected areas.

Definition at line 1144 of file Optionals.java.

Member Function Documentation

void Optionals.StateLabelInc.decrement (boolean *visual*)

The method that undoes state increments, i.e. decreases the "counter".

Definition at line 1166 of file Optionals.java.

void Optionals.StateLabelInc.flip_state (boolean *visual*)

Class-specific overlay of the inherited method.

It increases or decreses the state.

Reimplemented from **Optionals.TextButton** (*p.202*).

Definition at line 1155 of file Optionals.java.

Member Data Documentation

StateLabelInc Optionals.StateLabelInc.opponent [package]

Definition at line 1140 of file Optionals.java.

StateLabel Optionals.StateLabelInc.target [package]

Definition at line 1139 of file Optionals.java.

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

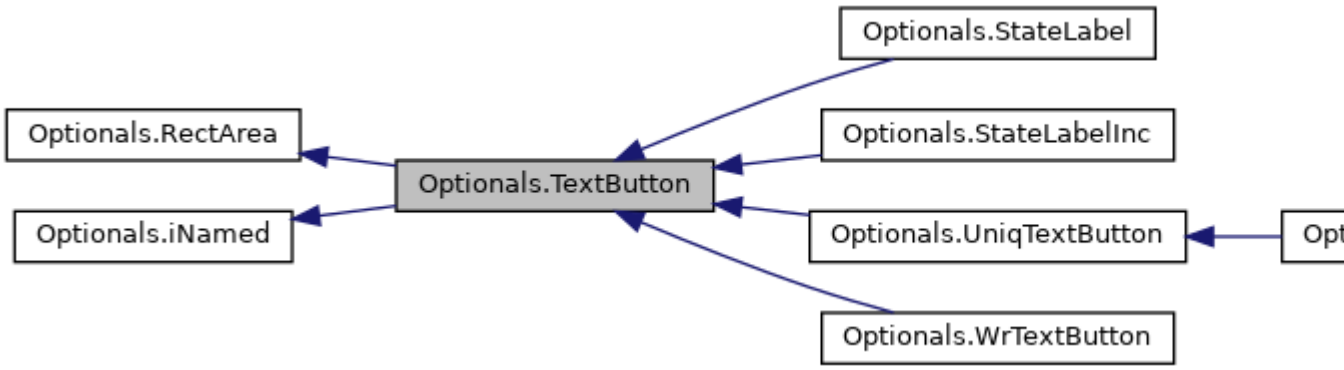
src.java/**Optionals.java**

Optionals.TextButton Class Reference

Rectangular button with text content.

Inheritance diagram for Optionals.TextButton:

IMAGE



Public Member Functions

String **name** ()

*Implements the **iNamed** interface requirement.*

void **view** ()

Area display function.

void **flip_state** (boolean visual)

Change to the opposite state (0 to 1, other to 0) and possibly visualize.

void **set_state** (int new_state, boolean visual)

It changes the state to 0 or 1 and optionally visualizes.

Protected Attributes

int **state**

Package Functions

TextButton (String iTitle, float iX1, float iY1, float iX2, float iY2)

Constructor.

Package Attributes

int **txt**
int **strok**
int **strokW**
int **txtSiz**
int **corner** =**iniTxButtonCornerRadius**
String **title**

Detailed Description

Rectangular button with text content.

Definition at line 997 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.TextButton.TextButton (String *iTitle*, float *iX1*, float *iY1*, float *iX2*, float *iY2*) [package]

Constructor.

Requires data on the corners of the area and text content ("title")

Definition at line 1009 of file Optionals.java.

Member Function Documentation

void Optionals.TextButton.flip_state (boolean *visual*)

Change to the opposite state (0 to 1, other to 0) and possibly visualize.

Reimplemented in **Optionals.UniqTextButton** (*p.209*), **Optionals.StateLabelInc** (*p.200*), and **Optionals.StateLabel** (*p.197*).

Definition at line 1045 of file Optionals.java.

String Optionals.TextButton.name ()

Implements the **iNamed** interface requirement.

Implements **Optionals.iNamed** (*p.109*).

Definition at line 1025 of file Optionals.java.

void Optionals.TextButton.set_state (int *new_state*, boolean *visual*)

It changes the state to 0 or 1 and optionally visualizes.

Reimplemented in **Optionals.StateLabel** (*p.197*).

Definition at line 1054 of file Optionals.java.

void Optionals.TextButton.view ()

Area display function.

Reimplemented from **Optionals.RectArea** (*p.189*).

Reimplemented in **Optionals.StateLabel** (*p.197*).

Definition at line 1028 of file Optionals.java.

Member Data Documentation

int Optionals.TextButton.corner =iniTxButtonCornerRadius[package]

Definition at line 1002 of file Optionals.java.

int Optionals.TextButton.state[protected]

Definition at line 1005 of file Optionals.java.

int Optionals.TextButton.strok[package]

Definition at line 999 of file Optionals.java.

int Optionals.TextButton.strokW[package]

Definition at line 1000 of file Optionals.java.

String Optionals.TextButton.title[package]

Definition at line 1004 of file Optionals.java.

int Optionals.TextButton.txt[package]

Definition at line 999 of file Optionals.java.

int Optionals.TextButton.txtSiz[package]

Definition at line 1001 of file Optionals.java.

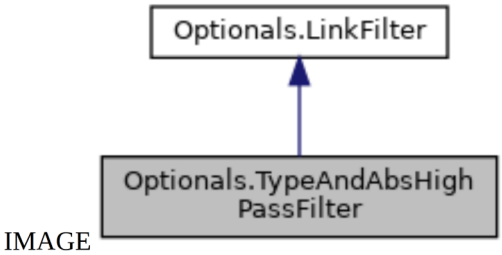
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.TypeAndAbsHighPassFilter Class Reference

Special type of filter for efficient visualisation.

Inheritance diagram for Optionals.TypeAndAbsHighPassFilter:



IMAGE

Public Member Functions

TypeAndAbsHighPassFilter reset (int t, float tres)
boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

TypeAndAbsHighPassFilter ()
TypeAndAbsHighPassFilter (int t, float tres)

Package Attributes

int **ltype**
float **treshold**

Detailed Description

Special type of filter for efficient visualisation.

Definition at line 1635 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.TypeAndAbsHighPassFilter.TypeAndAbsHighPassFilter () [package]

Definition at line 1639 of file Optionals.java.

Optionals.TypeAndAbsHighPassFilter.TypeAndAbsHighPassFilter (int *t*, float *tres*) [package]

Definition at line 1640 of file Optionals.java.

Member Function Documentation

boolean Optionals.TypeAndAbsHighPassFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1642 of file Optionals.java.

TypeAndAbsHighPassFilter Optionals.TypeAndAbsHighPassFilter.reset (int *t*, float *tres*)

Definition at line 1641 of file Optionals.java.

Member Data Documentation

int Optionals.TypeAndAbsHighPassFilter.ltype [package]

Definition at line 1637 of file Optionals.java.

float Optionals.TypeAndAbsHighPassFilter.treshold [package]

Definition at line 1638 of file Optionals.java.

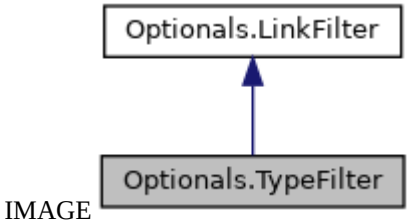
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.TypeFilter Class Reference

Type of link filter.

Inheritance diagram for Optionals.TypeFilter:



IMAGE

Public Member Functions

boolean **meetsTheAssumptions** (**Link** l)

INFO:

Package Functions

TypeFilter (int t)

Package Attributes

int **ltype**

Detailed Description

Type of link filter.

Class which filters links of specific "color"/"type"

Definition at line 1673 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.TypeFilter.TypeFilter (int *t*)[package]

Definition at line 1676 of file Optionals.java.

Member Function Documentation

boolean Optionals.TypeFilter.meetsTheAssumptions (Link *l*)

INFO:

Reimplemented from **Optionals.LinkFilter** (*p.120*).

Definition at line 1677 of file Optionals.java.

Member Data Documentation

int Optionals.TypeFilter.ltype [package]

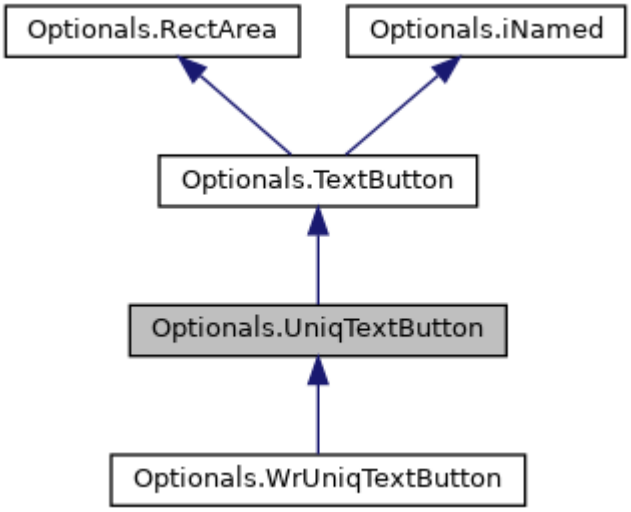
Definition at line 1675 of file Optionals.java.

The documentation for this class was generated from the following file:
src.java/**Optionals.java**

Optionals.UniqTextButton Class Reference

Unique button.

Inheritance diagram for Optionals.UniqTextButton:



IMAGE

Public Member Functions

void **flip_state** (boolean visual)

Normally the method changes the state to the opposite (0 to 1, other to 0) and possibly visualizes.

Package Functions

UniqTextButton (ArrayList< **TextButton** > iSibl, String iTitle, float iX1, float iY1, float iX2, float iY2)

Constructor.

Package Attributes

ArrayList< **TextButton** > **siblings**

List of mutually exclusive buttons.

Additional Inherited Members

Detailed Description

Unique button.

The class of the button, which when clicked, resets the state of all the others on the list

Definition at line 1176 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.UniqTextButton.UniqTextButton (ArrayList< TextButton > *iSibl*, String *iTitle*, float *iX1*, float *iY1*, float *iX2*, float *iY2*) [package]

Constructor.

Requires data on the corners of the area, text "title" and a list of mutual siblings.

Definition at line 1182 of file Optionals.java.

Member Function Documentation

void Optionals.UniqTextButton.flip_state (boolean *visual*)

Normally the method changes the state to the opposite (0 to 1, other to 0) and possibly visualizes.

However, if the button state changes to other than 0 then his companions (siblings) on the list must be reset.

Reimplemented from **Optionals.TextButton** (*p.202*).

Definition at line 1191 of file Optionals.java.

Member Data Documentation

ArrayList<TextButton> Optionals.UniqTextButton.siblings [package]

List of mutually exclusive buttons.

Definition at line 1178 of file Optionals.java.

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

src.java/**Optionals.java**

CATemplate.World Class Reference

The main class of simulation.

Public Member Functions

void **swap** ()
Swap the cell arrays in synchronic mode.

Package Functions

World (int **side**)
*Constructor of the **World**.*

Package Attributes

int **cells** [][]
int **newcells** [][]

Detailed Description

The main class of simulation.

Definition at line 124 of file CATemplate.java.

Constructor & Destructor Documentation

CATemplate.World.World (int *side*) [package]

Constructor of the **World**.

Definition at line 133 of file CATemplate.java.

Member Function Documentation

void CATemplate.World.swap ()

Swap the cell arrays in synchronic mode.

Definition at line 143 of file CATemplate.java.

Member Data Documentation

int CATemplate.World.cells[][] [package]

Definition at line 129 of file CATemplate.java.

int CATemplate.World.newcells[][] [package]

Definition at line 130 of file CATemplate.java.

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

src.java/**CATemplate.java**

ABMTemplate.World Class Reference

The main class of simulation.

Package Functions

World (int **side**)

*Constructor of the **World**.*

Package Attributes

Agent agents [][]

Detailed Description

The main class of simulation.

Definition at line 231 of file ABMTemplate.java.

Constructor & Destructor Documentation

ABMTemplate.World.World (int *side*) [package]

Constructor of the **World**.

Definition at line 238 of file ABMTemplate.java.

Member Data Documentation

Agent ABMTemplate.World.agents[][] [package]

Definition at line 235 of file ABMTemplate.java.

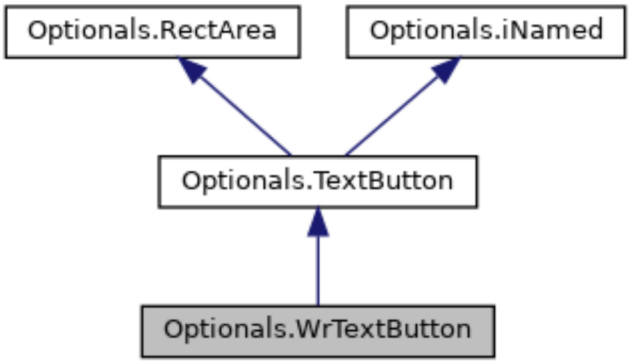
The documentation for this class was generated from the following file:

src.java/**ABMTemplate.java**

Optionals.WrTextButton Class Reference

A button that remembers the column to which its unique marker is to be saved.

Inheritance diagram for Optionals.WrTextButton:



IMAGE

Package Functions

WrTextButton (String iTitle, float iX1, float iY1, float iX2, float iY2, String iMarker, int iColumn)
Constructor.

Package Attributes

int **column**
String **marker**

Additional Inherited Members

Detailed Description

A button that remembers the column to which its unique marker is to be saved.

Definition at line 1207 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.WrTextButton.WrTextButton (String *iTitle*, float *iX1*, float *iY1*, float *iX2*, float *iY2*, String *iMarker*, int *iColumn*) [package]

Constructor.

Requires data on the corners of the area, text "title", text marker and specific output column.

Definition at line 1214 of file Optionals.java.

Member Data Documentation

int Optionals.WrTextButton.column [package]

Definition at line 1209 of file Optionals.java.

String Optionals.WrTextButton.marker [package]

Definition at line 1210 of file Optionals.java.

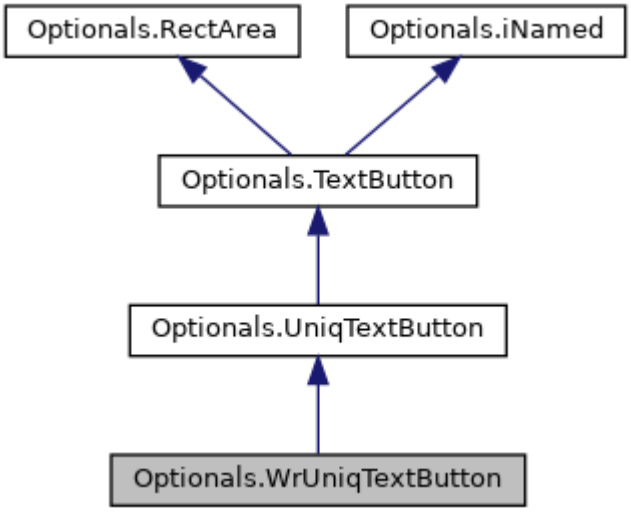
The documentation for this class was generated from the following file:

src.java/**Optionals.java**

Optionals.WrUniqTextButton Class Reference

UniqButton additionally remembers the column to which it is to save its unique marker.

Inheritance diagram for Optionals.WrUniqTextButton:



IMAGE

Package Functions

WrUniqTextButton (ArrayList< **TextButton** > iSibl, String iTitle, float iX1, float iY1, float iX2, float iY2, String iMarker, int iColumn)

Constructor.

Package Attributes

int **column**
String **marker**

Additional Inherited Members

Detailed Description

UniqButton additionally remembers the column to which it is to save its unique marker.

Definition at line 1223 of file Optionals.java.

Constructor & Destructor Documentation

Optionals.WrUniqTextButton.WrUniqTextButton (ArrayList< TextButton > *iSibl*, String *iTitle*, float *iX1*, float *iY1*, float *iX2*, float *iY2*, String *iMarker*, int *iColumn*) [package]

Constructor.

Requires data on the corners of the area, text "title", text marker and specific output column, and, of course, a list of mutual siblings.

Definition at line 1231 of file Optionals.java.

Member Data Documentation

int Optionals.WrUniqTextButton.column [package]

Definition at line 1225 of file Optionals.java.

String Optionals.WrUniqTextButton.marker [package]

Definition at line 1226 of file Optionals.java.

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

src.java/**Optionals.java**

File Documentation

src.java/ABMTemplate.java File Reference

Classes

class **ABMTemplate**
class **ABMTemplate.Agent**
__Agent__ is a one of two central class of each ABM model.
class **ABMTemplate.World**
The main class of simulation.
class **ABMTemplate.PairOfInt**
Simple version of Pair containing a pair of integers.

src.java/CATemplate.java File Reference

Classes

class **CATemplate**
class **CATemplate.World**
The main class of simulation.
class **CATemplate.PairOfInt**
Simple version of Pair containing a pair of integers.

src.java/gameClient.java File Reference

Classes

class **gameClient**

class **gameClient.implNeeded**

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

class **gameClient.Position**

Representation of 3D position in the game world However, the value of Z is not always used.

class **gameClient.GameObject**

Representation of simple game object.

class **gameClient.ActiveGameObject**

class **gameClient.Player**

Representation of generic player.

class **gameClient.Opcs**

Protocol dictionary ("opcodes" etc.)

src.java/gameServer.java File Reference

Classes

class **gameServer**

class **gameServer.implNeeded**

Server side implementation part of any game object needs modification flags, but client side are free to use this parts.

class **gameServer.Position**

Representation of 3D position in the game world However, the value of Z is not always used.

class **gameServer.GameObject**

Representation of simple game object.

class **gameServer.ActiveGameObject**

class **gameServer.Player**

Representation of generic player.

class **gameServer.Opcs**

Protocol dictionary ("opcodes" etc.)

src.java/Optionals.java File Reference

Classes

class **Optionals**

class **Optionals.Agent**

*Dummy class of **Agent**.*

class **Optionals.NamedData**

Functions & classes for chart making.

class **Optionals.Range**

Class of a NAMED range of real (float) numbers.

class **Optionals.Sample**

*This class represents a NAMED series of real (float) numbers Should it also be a descendant of the **Range**? ...*

class **Optionals.Frequencies**

This class represens a named histogram of frequencies.

class **Optionals.DummyInt**

Classes for taking an object from a simple variable of type int, boolean, float & double.

class **Optionals.DummyBool**

A class for taking an object from a simple logic variable (true-false).

class **Optionals.DummyFloat**

A class for taking an object from a simple variable of type float.

class **Optionals.DummyDouble**

A class for taking an object from a simple variable of type double.

class **Optionals.RectArea**

Rectangular screen area class as the basis for various active areas.

class **Optionals.PanelOfTextButtons**

A class of a panel that contains many buttons.

class **Optionals.TextButton**

Rectangular button with text content.

class **Optionals.StateLabel**

*A pseudo-button class that displays the state, not the name, Also ignores **flip_state()** and that changes to state through **set_state()** are "protected".*

class **Optionals.StateLabelInc**

A button class that increments a state label, possibly undoing the operation of the opposite pair.

class **Optionals.UniqTextButton**

Unique button.

class **Optionals.WrTextButton**

A button that remembers the column to which its unique marker is to be saved.

class **Optionals.WrUniqTextButton**

UniqButton additionally remembers the column to which it is to save its unique marker.

class **Optionals.Pair< A, B >**

COMMON TEMPLATES, INTERFACES AND ABSTRACT CLASSES.

interface **Optionals.describable**

Generally useable interfaces:

interface **Optionals.iNamed**

Forcing name available as String (planty of usage)

interface **Optionals.iDescribable**

Any object which have description as (potentially) long, multi line string.

interface **Optionals.Function2D**
A function of two values in the form of a class - a functor.

interface **Optionals.iColorable**
VISUALISATION INTERFACES:

interface **Optionals.iPositioned**
*Forcing **posX()** & **posY()** & **posZ()** methods for visualisation and mapping*

interface **Optionals.iLink**
*Network connection/link interface Is **iLink** interface really needed?*

interface **Optionals.iNode**
Network node interface "Conn" below is a shortage from Connection.

interface **Optionals.iVisNode**
Visualisable network node.

interface **Optionals.iVisLink**
Visualisable network connection.

class **Optionals.AllLinks**
Different filters of links and other link tools for a (social) network.

class **Optionals.TypeAndAbsHighPassFilter**
Special type of filter for efficient visualisation.

class **Optionals.AndFilter**
AND two filters assembly class.

class **Optionals.OrFilter**
OR two filters assembly class.

class **Optionals.TypeFilter**
Type of link filter.

class **Optionals.LowPassFilter**
Low Pass Filter.

class **Optionals.HighPassFilter**
High Pass Filter.

class **Optionals.AbsLowPassFilter**
Absolute Low Pass Filter.

class **Optionals.AbsHighPassFilter**
Absolute High Pass Filter.

class **Optionals.LinkFilter**
DEBUG level for network. Visible autside this file!

class **Optionals.LinkFactory**
class **Optionals.Named**
class **Optionals.Colorable**
class **Optionals.Positioned**
class **Optionals.Node**
class **Optionals.Link**
class **Optionals.NodeList**
class **Optionals.pointxy**
class **Optionals.settings_bar3d**

Index

INDEX