

LogicSim

Generated by Doxygen 1.15.0

1 DejaVu fonts v2.37	1
1.0.1 DejaVu License	1
2 jQuery v3.7.1	5
2.0.1 jQuery License	5
3 jQuery UI v1.14.1	7
3.0.1 jQuery UI License	7
4 Hierarchical Index	9
4.1 Class Hierarchy	9
5 Class Index	11
5.1 Class List	11
6 Class Documentation	13
6.1 com.logisim.domain.components.And Class Reference	13
6.1.1 Detailed Description	14
6.1.2 Constructor & Destructor Documentation	15
6.1.2.1 And()	15
6.1.3 Member Function Documentation	15
6.1.3.1 execute()	15
6.1.3.2 getOutput()	15
6.2 com.logisim.domain.components.Bulb Class Reference	16
6.2.1 Detailed Description	17
6.2.2 Constructor & Destructor Documentation	17
6.2.2.1 Bulb()	17
6.2.3 Member Function Documentation	18
6.2.3.1 execute()	18
6.2.3.2 getName()	18
6.2.3.3 isOn()	18
6.3 com.logisim.domain.Circuit Class Reference	18
6.3.1 Detailed Description	20
6.3.2 Member Function Documentation	20
6.3.2.1 addComponent()	20
6.3.2.2 addConnection() [1/2]	20
6.3.2.3 addConnection() [2/2]	21
6.3.2.4 analyze()	21
6.3.2.5 generateBooleanExpression()	21
6.3.2.6 getComponents()	22
6.3.2.7 getConnectors()	22
6.3.2.8 getId()	22
6.3.2.9 getName()	22
6.3.2.10 removeComponent()	22

6.3.2.11 setComponents()	23
6.3.2.12 setConnectors()	23
6.3.2.13 setId()	23
6.3.2.14 setName()	23
6.3.2.15 simulate()	24
6.3.2.16 toString()	24
6.4 com.logisim.data.CircuitDAO Class Reference	24
6.4.1 Detailed Description	25
6.4.2 Member Function Documentation	25
6.4.2.1 ConnectionRecord()	25
6.4.2.2 createCircuit()	25
6.4.2.3 deleteCircuit()	26
6.4.2.4 getCircuitsByProjectId()	26
6.4.2.5 loadComponents()	26
6.4.2.6 loadConnections()	27
6.4.2.7 saveCircuit()	27
6.4.2.8 saveComponents()	27
6.4.2.9 saveConnectors()	28
6.4.2.10 updateCircuit()	28
6.5 com.logisim.domain.components.Component Class Reference	29
6.5.1 Detailed Description	30
6.5.2 Constructor & Destructor Documentation	30
6.5.2.1 Component() [1/2]	30
6.5.2.2 Component() [2/2]	30
6.5.3 Member Function Documentation	31
6.5.3.1 execute()	31
6.5.3.2 getInputs()	31
6.5.3.3 getName()	31
6.5.3.4 getOutput()	31
6.5.3.5 getOutputs()	32
6.5.3.6 getPositionX()	32
6.5.3.7 getPositionY()	32
6.5.3.8 getUuid()	32
6.5.3.9 setInput()	32
6.5.3.10 setInputs()	33
6.5.3.11 setName()	33
6.5.3.12 setOutputs()	33
6.5.3.13 setPositionX()	33
6.5.3.14 setPositionY()	34
6.5.3.15 setUuid()	34
6.5.4 Member Data Documentation	34
6.5.4.1 inputs	34

6.5.4.2 outputs	34
6.5.4.3 uuid	35
6.6 com.logisim.ui.logic.ConnectionManager Class Reference	35
6.6.1 Detailed Description	36
6.6.2 Constructor & Destructor Documentation	36
6.6.2.1 ConnectionManager()	36
6.6.3 Member Function Documentation	36
6.6.3.1 cancelConnection()	36
6.6.3.2 createConnection()	36
6.6.3.3 handlePortClick()	37
6.6.3.4 onMouseMove()	37
6.6.3.5 setOnConnectionAdded()	37
6.6.4 Member Data Documentation	38
6.6.4.1 onConnectionAdded	38
6.6.4.2 selectedSourcePort	38
6.7 com.logisim.domain.Connector Class Reference	38
6.7.1 Detailed Description	39
6.7.2 Constructor & Destructor Documentation	39
6.7.2.1 Connector()	39
6.7.3 Member Function Documentation	39
6.7.3.1 getName()	39
6.7.3.2 getSink()	40
6.7.3.3 getSinkComp()	40
6.7.3.4 getSource()	40
6.7.3.5 getSourceComp()	40
6.7.3.6 process()	40
6.8 com.logisim.data.DatabaseManager Class Reference	41
6.8.1 Detailed Description	41
6.8.2 Constructor & Destructor Documentation	41
6.8.2.1 DatabaseManager()	41
6.8.3 Member Function Documentation	42
6.8.3.1 createTables()	42
6.8.3.2 getConnection()	42
6.8.3.3 getInstance()	42
6.8.4 Member Data Documentation	43
6.8.4.1 url	43
6.9 com.logisim.ui.components.GateFactory Class Reference	43
6.9.1 Detailed Description	44
6.9.2 Member Function Documentation	44
6.9.2.1 addPortsToGate()	44
6.9.2.2 configurePortEvents()	44
6.9.2.3 createGateWithHitBox()	45

6.9.2.4 createSubCircuitVisual()	45
6.9.2.5 makeDraggableandDeletable()	46
6.9.2.6 refreshComponentState()	46
6.9.2.7 setConnectionManager()	47
6.9.2.8 setupSwitchInteraction()	48
6.10 com.logisim.ui.controllers.GridController Class Reference	48
6.10.1 Detailed Description	49
6.10.2 Constructor & Destructor Documentation	49
6.10.2.1 GridController()	49
6.10.3 Member Function Documentation	49
6.10.3.1 drawGrid()	49
6.10.3.2 getGridSize()	49
6.10.3.3 snap()	49
6.11 com.logisim.MainApp Class Reference	50
6.12 com.logisim.ui.controllers.MainViewController Class Reference	50
6.12.1 Detailed Description	52
6.12.2 Member Function Documentation	52
6.12.2.1 findPort()	52
6.12.2.2 handleAnalyze()	53
6.12.2.3 handleBackToDashboard()	53
6.12.2.4 handleDeleteGate()	53
6.12.2.5 handleRun()	53
6.12.2.6 handleToggleSwitch()	53
6.12.2.7 initialize()	54
6.12.2.8 loadFullCircuitFromDB()	54
6.12.2.9 refreshSubCircuitSidebar()	54
6.12.2.10 setContext()	54
6.12.2.11 showAlert()	55
6.12.2.12 showAnalysisWindow()	55
6.12.2.13 spawnSubCircuit()	55
6.13 com.logisim.domain.components.Not Class Reference	56
6.13.1 Detailed Description	57
6.13.2 Constructor & Destructor Documentation	57
6.13.2.1 Not()	57
6.13.3 Member Function Documentation	58
6.13.3.1 execute()	58
6.13.3.2 getOutput()	58
6.13.3.3 setInput()	58
6.14 com.logisim.domain.components.Or Class Reference	58
6.14.1 Detailed Description	60
6.14.2 Constructor & Destructor Documentation	60
6.14.2.1 Or()	60

6.14.3 Member Function Documentation	60
6.14.3.1 execute()	60
6.14.3.2 getOutput()	61
6.15 com.logisim.ui.components.Port Class Reference	61
6.15.1 Detailed Description	62
6.15.2 Constructor & Destructor Documentation	62
6.15.2.1 Port()	62
6.15.3 Member Function Documentation	62
6.15.3.1 getConnectionState()	62
6.15.3.2 getIndex()	63
6.15.3.3 getParentGate()	63
6.15.3.4 isInput()	63
6.15.3.5 isSelected()	63
6.15.3.6 setConnectionState()	63
6.15.3.7 setSelected()	64
6.15.4 Member Data Documentation	64
6.15.4.1 isInput	64
6.16 com.logisim.domain.Project Class Reference	64
6.16.1 Detailed Description	65
6.16.2 Constructor & Destructor Documentation	66
6.16.2.1 Project() [1/3]	66
6.16.2.2 Project() [2/3]	66
6.16.2.3 Project() [3/3]	66
6.16.3 Member Function Documentation	66
6.16.3.1 export()	66
6.16.3.2 getCircuits()	67
6.16.3.3 getId()	67
6.16.3.4 getName()	67
6.16.3.5 getProjectdao()	67
6.16.3.6 load()	67
6.16.3.7 save()	68
6.16.3.8 setCircuits()	68
6.16.3.9 setId()	69
6.16.3.10 setName()	69
6.16.3.11 setProjectdao()	69
6.16.3.12 toString()	69
6.17 com.logisim.data.ProjectDAO Class Reference	70
6.17.1 Detailed Description	70
6.17.2 Member Function Documentation	70
6.17.2.1 deleteProject()	70
6.17.2.2 getAllProjects()	71
6.17.2.3 saveProject()	71

6.18 com.logisim.ui.controllers.ProjectDashboardController Class Reference	71
6.18.1 Detailed Description	72
6.18.2 Member Function Documentation	72
6.18.2.1 handleBack()	72
6.18.2.2 handleDeleteCircuit()	73
6.18.2.3 handleExport()	73
6.18.2.4 handleNewCircuit()	73
6.18.2.5 handleOpenCircuit()	73
6.18.2.6 initialize()	73
6.18.2.7 openMainEditor()	73
6.18.2.8 refreshList()	74
6.18.2.9 setProject()	74
6.18.2.10 styleDialog()	74
6.19 com.logisim.ui.logic.SafePoints Class Reference	74
6.19.1 Detailed Description	75
6.19.2 Member Function Documentation	75
6.19.2.1 getSafeSpawnPoint()	75
6.20 com.logisim.ui.controllers.StartScreenController Class Reference	75
6.20.1 Detailed Description	76
6.20.2 Member Function Documentation	76
6.20.2.1 handleDeleteProject()	76
6.20.2.2 handleLoadProject()	77
6.20.2.3 handleNewProject()	77
6.20.2.4 initialize()	77
6.20.2.5 loadDashboard()	77
6.20.2.6 showAlert()	77
6.20.2.7 styleDialog()	78
6.21 com.logisim.domain.components.SubCircuitComponent Class Reference	78
6.21.1 Detailed Description	80
6.21.2 Constructor & Destructor Documentation	80
6.21.2.1 SubCircuitComponent()	80
6.21.3 Member Function Documentation	81
6.21.3.1 execute()	81
6.21.3.2 getInnerCircuit()	81
6.21.3.3 getInternalBulbs()	81
6.21.3.4 getInternalSwitches()	81
6.21.3.5 getName()	82
6.21.3.6 getSourceCircuitId()	82
6.21.3.7 setInnerCircuit()	82
6.21.3.8 setInternalBulbs()	82
6.21.3.9 setInternalSwitches()	82
6.21.3.10 setSourceCircuitId()	83

6.22 com.logisim.domain.components.Switch Class Reference	83
6.22.1 Detailed Description	85
6.22.2 Constructor & Destructor Documentation	85
6.22.2.1 Switch()	85
6.22.3 Member Function Documentation	85
6.22.3.1 execute()	85
6.22.3.2 getName()	85
6.22.3.3 isOn()	86
6.22.3.4 setState()	86
6.22.3.5 toggle()	86
6.22.4 Member Data Documentation	86
6.22.4.1 isOn	86
6.23 com.logisim.ui.components.Wire Class Reference	87
6.23.1 Detailed Description	87
6.23.2 Constructor & Destructor Documentation	87
6.23.2.1 Wire()	87
6.23.3 Member Function Documentation	88
6.23.3.1 getSink()	88
6.23.3.2 getSource()	88
6.23.3.3 setupInteractions()	88
6.23.3.4 updateWire()	88
Index	89

Chapter 1

DejaVu fonts v2.37

1.0.1 DejaVu License

Fonts are (c) Bitstream (see below). DejaVu changes are in public domain.
Glyphs imported from Arev fonts are (c) Tavmjong Bah (see below)

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE

FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

TeX Gyre DJV Math

Fonts are (c) Bitstream (see below). DejaVu changes are in public domain.

Math extensions done by B. Jackowski, P. Strzelczyk and P. Pianowski (on behalf of TeX users groups) are in public domain.

Letters imported from Euler Fraktur from AMSfonts are (c) American Mathematical Society (see below).
 Bitstream Vera Fonts Copyright
 Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE. Except as contained in this notice, the names of GNOME, the GNOME Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the GNOME Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

AMSFonTS (v. 2.2) copyright

The PostScript Type 1 implementation of the AMSFonts produced by and previously distributed by Blue Sky Research and Y&Y, Inc. are now freely available for general use. This has been accomplished through the cooperation of a consortium of scientific publishers with Blue Sky Research and Y&Y. Members of this consortium include:

Elsevier Science IBM Corporation Society for Industrial and Applied
Mathematics (SIAM) Springer-Verlag American Mathematical Society (AMS)

In order to assure the authenticity of these fonts, copyright will be held by the American Mathematical Society. This is not meant to restrict in any way the legitimate use of the fonts, such as (but not limited to) electronic distribution of documents containing these fonts, inclusion of these fonts into other public domain or commercial font collections or computer applications, use of the outline data to create derivative fonts and/or faces, etc. However, the AMS does require that the AMS copyright notice be removed from any derivative versions of the fonts which have been altered in any way. In addition, to ensure the fidelity of TeX documents using Computer Modern fonts, Professor Donald Knuth, creator of the Computer Modern faces, has requested that any alterations which yield different font metrics be given a different name.

Chapter 2

jQuery v3.7.1

2.0.1 jQuery License

jQuery v 3.7.1
Copyright OpenJS Foundation and other contributors, <https://openjsf.org/>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 3

jQuery UI v1.14.1

3.0.1 jQuery UI License

Copyright OpenJS Foundation and other contributors, <https://openjsf.org/>

This software consists of voluntary contributions made by many individuals. For exact contribution history, see the revision history available at <https://github.com/jquery/jquery-ui>

The following license applies to all parts of this software except as documented below:

====

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

====

Copyright and related rights for sample code are waived via CC0. Sample code is defined as all source code contained within the demos directory.

CC0: <http://creativecommons.org/publicdomain/zero/1.0/>

====

All files located in the node_modules and external directories are externally maintained libraries used by this software which have their own licenses; we recommend you read them, as their terms may differ from the terms above.

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Application	
com.logisim.MainApp	50
Circle	
com.logisim.ui.components.Port	61
com.logisim.domain.Circuit	18
com.logisim.data.CircuitDAO	24
com.logisim.domain.components.Component	29
com.logisim.domain.components.And	13
com.logisim.domain.components.Bulb	16
com.logisim.domain.components.Not	56
com.logisim.domain.components.Or	58
com.logisim.domain.components.SubCircuitComponent	78
com.logisim.domain.components.Switch	83
com.logisim.ui.logic.ConnectionManager	35
com.logisim.domain.Connector	38
com.logisim.data.DatabaseManager	41
com.logisim.ui.components.GateFactory	43
com.logisim.ui.controllers.GridController	48
com.logisim.ui.controllers.MainViewController	50
Polyline	
com.logisim.ui.components.Wire	87
com.logisim.domain.Project	64
com.logisim.data.ProjectDAO	70
com.logisim.ui.controllers.ProjectDashboardController	71
com.logisim.ui.logic.SafePoints	74
com.logisim.ui.controllers.StartScreenController	75

Chapter 5

Class Index

5.1 Class List

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

com.logisim.domain.components.And	
Represents a logical AND gate component within the circuit simulation	13
com.logisim.domain.components.Bulb	
Represents a light bulb component in the circuit simulation	16
com.logisim.domain.Circuit	
Represents a digital logic circuit consisting of components and connections	18
com.logisim.data.CircuitDAO	
Data Access Object (DAO) responsible for handling database operations related to Circuit entities	24
com.logisim.domain.components.Component	
An abstract base class representing a generic component within a logic circuit	29
com.logisim.ui.logic.ConnectionManager	
Manages the interactive state for creating connections (wires) between components	35
com.logisim.domain.Connector	
Represents a wire connection between two components in the circuit	38
com.logisim.data.DatabaseManager	
Manages the SQLite database connection and schema initialization for the application	41
com.logisim.ui.components.GateFactory	
A factory class responsible for creating and configuring the visual representations of logic components (gates, switches, bulbs, etc.) on the UI canvas	43
com.logisim.ui.controllers.GridController	
Controller responsible for managing the background grid of the circuit editor	48
com.logisim.MainApp	50
com.logisim.ui.controllers.MainViewController	
The primary controller for the Circuit Editor view	50
com.logisim.domain.components.Not	
Represents a logical NOT gate (inverter) in the circuit simulation	56
com.logisim.domain.components.Or	
Represents a logical OR gate component within the circuit simulation	58
com.logisim.ui.components.Port	
Represents a specific connection point (pin) on a visual logic component	61
com.logisim.domain.Project	
Represents a project within the Logisim application	64
com.logisim.data.ProjectDAO	
Data Access Object (DAO) responsible for handling database operations related to Project entities	70

com.logisim.ui.controllers.ProjectDashboardController	
Controller class for the Project Dashboard view	71
com.logisim.ui.logic.SafePoints	
Utility class providing logic for determining safe spawn locations for UI components	74
com.logisim.ui.controllers.StartScreenController	
Controller class for the application's start screen	75
com.logisim.domain.components.SubCircuitComponent	
Represents a complex circuit encapsulated as a single component within another circuit	78
com.logisim.domain.components.Switch	
Represents a toggle switch component in the circuit simulation	83
com.logisim.ui.components.Wire	
Represents a visual wire connection between two Ports on the circuit canvas	87

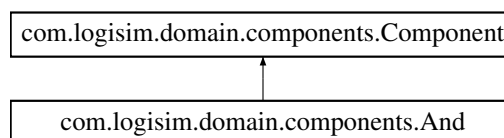
Chapter 6

Class Documentation

6.1 com.logisim.domain.components.And Class Reference

Represents a logical AND gate component within the circuit simulation.

Inheritance diagram for com.logisim.domain.components.And:



Public Member Functions

- [And](#) ()
Constructs a new AND gate component with default settings.
- boolean [getOutput](#) ()
Retrieves the current state of the AND gate's output pin.
- void [execute](#) ()
Executes the logic of the AND gate.

Public Member Functions inherited from [com.logisim.domain.components.Component](#)

- [Component](#) ()
Default constructor.
- [Component](#) (String [name](#))
Constructs a component with a specified name.
- void [setInput](#) (int index, boolean value)
Sets the state of a specific input pin.
- boolean [getOutput](#) (int index)
Retrieves the state of a specific output pin.
- String [getName](#) ()
Gets the name of the component.
- void [setName](#) (String [name](#))

- Sets the name of the component.*
- `boolean[] getInputs ()`
 - Retrieves the entire array of input pin states.*
- `void setInputs (boolean[] inputs)`
 - Sets the entire array of input pin states.*
- `boolean[] getOutputs ()`
 - Retrieves the entire array of output pin states.*
- `void setOutputs (boolean[] outputs)`
 - Sets the entire array of output pin states.*
- `double getPositionX ()`
 - Gets the X-coordinate of the component.*
- `double getPositionY ()`
 - Gets the Y-coordinate of the component.*
- `void setPositionX (double positionX)`
 - Sets the X-coordinate of the component.*
- `void setPositionY (double positionY)`
 - Sets the Y-coordinate of the component.*
- `String getUuid ()`
 - Gets the unique identifier (UUID) of the component.*
- `void setUuid (String uuid)`
 - Sets the unique identifier (UUID) of the component.*

Additional Inherited Members

Protected Attributes inherited from [com.logisim.domain.components.Component](#)

- String **name**
 - The name or type identifier of the component (e.g., "and", "or").*
- `boolean[] inputs`
 - An array representing the state of the component's input pins.*
- `boolean[] outputs`
 - An array representing the state of the component's output pins.*
- double **positionX**
 - The X-coordinate of the component's position in the visual interface.*
- double **positionY**
 - The Y-coordinate of the component's position in the visual interface.*

6.1.1 Detailed Description

Represents a logical AND gate component within the circuit simulation.

This component accepts two boolean inputs and produces a single boolean output. The output is true if and only if both inputs are true; otherwise, the output is false.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 And()

```
com.logisim.domain.components.And.And ()
```

Constructs a new AND gate component with default settings.

Initializes the component with:

- Name: "and"
- Input array size: 2
- Output array size: 1
- Default position: (100, 100)
- Initial state: All inputs and outputs set to false.

6.1.3 Member Function Documentation

6.1.3.1 execute()

```
void com.logisim.domain.components.And.execute ()
```

Executes the logic of the AND gate.

This method updates the internal output state based on the current values of the input pins. The output at index 0 is set to `true` only if both input index 0 and input index 1 are `true`.

Reimplemented from [com.logisim.domain.components.Component](#).

6.1.3.2 getOutput()

```
boolean com.logisim.domain.components.And.getOutput ()
```

Retrieves the current state of the AND gate's output pin.

This is a convenience method that delegates to the superclass's [Component#getOutput \(int\)](#) method with index 0.

Returns

`true` if the gate is outputting a high signal, `false` otherwise.

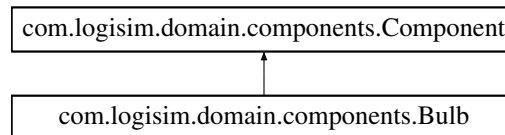
The documentation for this class was generated from the following file:

- `src/main/java/com/logisim/domain/components/And.java`

6.2 com.logisim.domain.components.Bulb Class Reference

Represents a light bulb component in the circuit simulation.

Inheritance diagram for com.logisim.domain.components.Bulb:



Public Member Functions

- [Bulb](#) ()
Constructs a new [Bulb](#) component.
- void [execute](#) ()
Updates the state of the bulb based on the current input.
- boolean [isOn](#) ()
Checks if the bulb is currently illuminated.
- String [getName](#) ()
Retrieves the unique type name of this component.

Public Member Functions inherited from [com.logisim.domain.components.Component](#)

- [Component](#) ()
Default constructor.
- [Component](#) (String [name](#))
Constructs a component with a specified name.
- void [setInput](#) (int index, boolean value)
Sets the state of a specific input pin.
- boolean [getOutput](#) (int index)
Retrieves the state of a specific output pin.
- void [setName](#) (String [name](#))
Sets the name of the component.
- boolean[] [getInputs](#) ()
Retrieves the entire array of input pin states.
- void [setInputs](#) (boolean[] [inputs](#))
Sets the entire array of input pin states.
- boolean[] [getOutputs](#) ()
Retrieves the entire array of output pin states.
- void [setOutputs](#) (boolean[] [outputs](#))
Sets the entire array of output pin states.
- double [getPositionX](#) ()
Gets the X-coordinate of the component.
- double [getPositionY](#) ()
Gets the Y-coordinate of the component.
- void [setPositionX](#) (double [positionX](#))
Sets the X-coordinate of the component.
- void [setPositionY](#) (double [positionY](#))
Sets the Y-coordinate of the component.
- String [getUuid](#) ()
Gets the unique identifier (UUID) of the component.
- void [setUuid](#) (String [uuid](#))
Sets the unique identifier (UUID) of the component.

Private Attributes

- boolean **isOn**
Internal state indicating whether the bulb is currently illuminated.

Additional Inherited Members

Protected Attributes inherited from [com.logisim.domain.components.Component](#)

- String **name**
The name or type identifier of the component (e.g., "and", "or").
- boolean[] **inputs**
An array representing the state of the component's input pins.
- boolean[] **outputs**
An array representing the state of the component's output pins.
- double **positionX**
The X-coordinate of the component's position in the visual interface.
- double **positionY**
The Y-coordinate of the component's position in the visual interface.

6.2.1 Detailed Description

Represents a light bulb component in the circuit simulation.

The [Bulb](#) serves as a visual output device. It accepts a single input signal. When the input signal is high (true), the bulb is considered "on" or illuminated. When the input signal is low (false), the bulb is "off".

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Bulb()

```
com.logisim.domain.components.Bulb.Bulb ()
```

Constructs a new [Bulb](#) component.

Initializes the component with:

- 1 Input pin.
- 0 Output pins.
- Initial state set to off (*false*).

6.2.3 Member Function Documentation

6.2.3.1 execute()

```
void com.logisim.domain.components.Bulb.execute ()
```

Updates the state of the bulb based on the current input.

This method reads the value from the first input pin (index 0) and updates the internal `isOn` state to match it.

Reimplemented from [com.logisim.domain.components.Component](#).

6.2.3.2 getName()

```
String com.logisim.domain.components.Bulb.getName ()
```

Retrieves the unique type name of this component.

Returns

The string literal "bulb".

Reimplemented from [com.logisim.domain.components.Component](#).

6.2.3.3 isOn()

```
boolean com.logisim.domain.components.Bulb.isOn ()
```

Checks if the bulb is currently illuminated.

Returns

`true` if the bulb is on (receiving a high signal), `false` otherwise.

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

- `src/main/java/com/logisim/domain/components/Bulb.java`

6.3 com.logisim.domain.Circuit Class Reference

Represents a digital logic circuit consisting of components and connections.

Public Member Functions

- **Circuit ()**
Constructs a new [Circuit](#) with a default name.
- void **addComponent** ([Component](#) comp) throws IllegalArgumentException
Adds a component to the circuit.
- void **removeComponent** ([Component](#) comp)
Removes a component from the circuit.
- void **addConnection** (int source, int sourceComp, int sink, int sinkComp)
Adds a connection between two components identified by their indices in the component list.
- void **addConnection** (int sourcePin, [Component](#) sourceComp, int sinkPin, [Component](#) sinkComp)
Adds a connection between two specific component instances.
- void **simulate** ()
Simulates the circuit logic for one cycle.
- boolean[][] **analyze** ()
Generates a truth table for the current circuit configuration.
- String **generateBooleanExpression** (boolean[][] truthTable, List< String > inputNames)
Generates a boolean algebraic expression (Sum of Products) based on the provided truth table.
- List< [Component](#) > **getComponents** ()
Retrieves the list of components in the circuit.
- String **getName** ()
Retrieves the name of the circuit.
- List< [Connector](#) > **getConnectors** ()
Retrieves the list of connectors (wires) in the circuit.
- long **getId** ()
Gets the unique identifier of the circuit.
- void **setId** (long id)
Sets the unique identifier of the circuit.
- void **setName** (String name)
Sets the name of the circuit.
- void **setComponents** (List< [Component](#) > components)
Replaces the current list of components with a new list.
- void **setConnectors** (List< [Connector](#) > connectors)
Replaces the current list of connectors with a new list.
- String **toString** ()
Returns the string representation of the circuit.

Private Attributes

- long **id**
The unique identifier for this circuit, typically used for database persistence.
- String **name**
The name of the circuit.
- List< [Component](#) > **components** = new ArrayList<>()
The list of logic components (gates, switches, bulbs, etc.) contained in this circuit.
- List< [Connector](#) > **connectors** = new ArrayList<>()
The list of connections (wires) linking the components together.

6.3.1 Detailed Description

Represents a digital logic circuit consisting of components and connections.

This class serves as the container for logic simulation. It manages a list of [Component](#) objects and [Connector](#) objects (wires). It provides functionality to modify the circuit structure, simulate the flow of signals through the components, and analyze the circuit's logic via truth tables and boolean expressions.

6.3.2 Member Function Documentation

6.3.2.1 addComponent()

```
void com.logisim.domain.Circuit.addComponent (
    Component comp) throws IllegalArgumentException
```

Adds a component to the circuit.

Parameters

<i>comp</i>	The Component to be added.
-------------	--

Exceptions

<i>IllegalArgumentException</i>	If the provided component is <code>null</code> or already exists in the circuit.
---------------------------------	--

6.3.2.2 addConnection() [1/2]

```
void com.logisim.domain.Circuit.addConnection (
    int source,
    int sourceComp,
    int sink,
    int sinkComp)
```

Adds a connection between two components identified by their indices in the component list.

Parameters

<i>source</i>	The index of the output pin on the source component.
<i>sourceComp</i>	The index of the source component in the internal components list.
<i>sink</i>	The index of the input pin on the sink (destination) component.
<i>sinkComp</i>	The index of the sink component in the internal components list.

Exceptions

<i>InvalidParameterException</i>	If the component indices are out of bounds or the pin indices are invalid for the respective components.
----------------------------------	--

6.3.2.3 addConnection() [2/2]

```
void com.logisim.domain.Circuit.addConnection (
    int sourcePin,
    Component sourceComp,
    int sinkPin,
    Component sinkComp)
```

Adds a connection between two specific component instances.

Parameters

<i>sourcePin</i>	The index of the output pin on the source component.
<i>sourceComp</i>	The source Component instance.
<i>sinkPin</i>	The index of the input pin on the sink component.
<i>sinkComp</i>	The sink Component instance.

Exceptions

<i>IllegalArgumentException</i>	If either component is null, or if the specified pin indices do not exist.
---------------------------------	--

6.3.2.4 analyze()

```
boolean[][] com.logisim.domain.Circuit.analyze ()
```

Generates a truth table for the current circuit configuration.

This method identifies all [com.logisim.domain.components.Switch](#) components as inputs and all [com.logisim.domain.components.Bulb](#) components as outputs. It iterates through all possible binary combinations of input states (2^n), simulates the circuit for each combination, and records the resulting output states.

Returns

A 2D boolean array representing the truth table. Rows represent each input combination. Columns [0 to $n-1$] represent input values. Columns [n to end] represent output values. Returns a 0x0 array if no inputs or outputs are found.

6.3.2.5 generateBooleanExpression()

```
String com.logisim.domain.Circuit.generateBooleanExpression (
    boolean truthTable[],
    List< String > inputNames)
```

Generates a boolean algebraic expression (Sum of Products) based on the provided truth table.

The method constructs a string expression representing the logic required to produce a 'true' output. It creates minterms for every row in the truth table where the output is true.

Parameters

<i>truthTable</i>	A 2D boolean array generated by <code>analyze()</code> .
<i>inputNames</i>	A list of names corresponding to the input columns in the truth table.

Returns

A String representing the boolean expression (e.g., "(A & B) + (!A & B)"). Returns "0" if the output is never true.

6.3.2.6 getComponents()

```
List< Component > com.logisim.domain.Circuit.getComponents ()
```

Retrieves the list of components in the circuit.

Returns

The list of `Component` objects.

6.3.2.7 getConnectors()

```
List< Connector > com.logisim.domain.Circuit.getConnectors ()
```

Retrieves the list of connectors (wires) in the circuit.

Returns

The list of `Connector` objects.

6.3.2.8 getId()

```
long com.logisim.domain.Circuit.getId ()
```

Gets the unique identifier of the circuit.

Returns

The circuit ID.

6.3.2.9 getName()

```
String com.logisim.domain.Circuit.getName ()
```

Retrieves the name of the circuit.

Returns

The circuit name.

6.3.2.10 removeComponent()

```
void com.logisim.domain.Circuit.removeComponent (
    Component comp)
```

Removes a component from the circuit.

This method also automatically removes any `Connectors` attached to the removed component to prevent dangling connections.

Parameters

<i>comp</i>	The Component to be removed.
-------------	--

6.3.2.11 setComponents()

```
void com.logisim.domain.Circuit.setComponents (  
    List< Component > components)
```

Replaces the current list of components with a new list.

Parameters

<i>components</i>	The new list of Component objects.
-------------------	--

6.3.2.12 setConnectors()

```
void com.logisim.domain.Circuit.setConnectors (  
    List< Connector > connectors)
```

Replaces the current list of connectors with a new list.

Parameters

<i>connectors</i>	The new list of Connector objects.
-------------------	--

6.3.2.13 setId()

```
void com.logisim.domain.Circuit.setId (  
    long id)
```

Sets the unique identifier of the circuit.

Parameters

<i>id</i>	The new circuit ID.
-----------	---------------------

6.3.2.14 setName()

```
void com.logisim.domain.Circuit.setName (  
    String name)
```

Sets the name of the circuit.

Parameters

<i>name</i>	The new circuit name.
-------------	-----------------------

6.3.2.15 simulate()

```
void com.logisim.domain.Circuit.simulate ()
```

Simulates the circuit logic for one cycle.

This method executes every component (calculating outputs from inputs) and then processes every connector (propagating outputs to inputs of connected components).

6.3.2.16 toString()

```
String com.logisim.domain.Circuit.toString ()
```

Returns the string representation of the circuit.

Returns

The name of the circuit.

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

- src/main/java/com/logisim/domain/Circuit.java

6.4 com.logisim.data.CircuitDAO Class Reference

Data Access Object (DAO) responsible for handling database operations related to [Circuit](#) entities.

Public Member Functions

- void [saveCircuit](#) ([Circuit](#) circuit, long projectId)
Saves a new circuit and its associated contents to the database.
- List< [Circuit](#) > [getCircuitsByProjectId](#) (long projectId)
Retrieves a list of circuits associated with a specific project.
- void [createCircuit](#) (long projectId, String name)
Creates a new empty circuit record in the database.
- List< [Component](#) > [loadComponents](#) (long circuitId)
Loads and reconstructs all components belonging to a specific circuit ID.
- record [ConnectionRecord](#) (String sourceUuid, int sourcePin, String sinkUuid, int sinkPin)
A record representing the raw data of a connection between two components.
- List< [ConnectionRecord](#) > [loadConnections](#) (long circuitId)
Retrieves the raw connection data for a specific circuit.
- void [updateCircuit](#) ([Circuit](#) circuit)
Updates an existing circuit in the database using a transactional approach.
- void [deleteCircuit](#) (long id)
Deletes a circuit from the database.

Private Member Functions

- void [saveComponents](#) ([Circuit](#) circuit, long circuitId, Connection conn) throws SQLException
Batch inserts the components of a circuit into the database.
- void [saveConnectors](#) ([Circuit](#) circuit, long circuitId, Connection conn) throws SQLException
Batch inserts the connectors (wires) of a circuit into the database.

6.4.1 Detailed Description

Data Access Object (DAO) responsible for handling database operations related to [Circuit](#) entities.

This class provides methods to create, retrieve, update, and delete circuits, as well as managing the persistence of their associated components and connectors.

6.4.2 Member Function Documentation

6.4.2.1 ConnectionRecord()

```
record com.logisim.data.CircuitDAO.ConnectionRecord (
    String sourceUuid,
    int sourcePin,
    String sinkUuid,
    int sinkPin)
```

A record representing the raw data of a connection between two components.

Parameters

<i>sourceUuid</i>	The UUID of the source component.
<i>sourcePin</i>	The pin index on the source component.
<i>sinkUuid</i>	The UUID of the destination (sink) component.
<i>sinkPin</i>	The pin index on the destination component.

6.4.2.2 createCircuit()

```
void com.logisim.data.CircuitDAO.createCircuit (
    long projectId,
    String name)
```

Creates a new empty circuit record in the database.

Parameters

<i>projectId</i>	The unique identifier of the project.
<i>name</i>	The name to be assigned to the new circuit.

6.4.2.3 deleteCircuit()

```
void com.logisim.data.CircuitDAO.deleteCircuit (
    long id)
```

Deletes a circuit from the database.

Due to database foreign key constraints (cascading deletes), removing the circuit usually removes associated components and connectors automatically.

Parameters

<i>id</i>	The unique identifier of the circuit to be deleted.
-----------	---

6.4.2.4 getCircuitsByProjectId()

```
List< Circuit > com.logisim.data.CircuitDAO.getCircuitsByProjectId (
    long projectId)
```

Retrieves a list of circuits associated with a specific project.

Note: This method retrieves the circuit metadata (ID and name) but does not automatically load the internal components or connectors.

Parameters

<i>projectId</i>	The unique identifier of the project.
------------------	---------------------------------------

Returns

A List of [Circuit](#) objects containing IDs and names.

6.4.2.5 loadComponents()

```
List< Component > com.logisim.data.CircuitDAO.loadComponents (
    long circuitId)
```

Loads and reconstructs all components belonging to a specific circuit ID.

This method instantiates specific component classes (e.g., [And](#), [Or](#), [Switch](#)) based on the 'type' column stored in the database. It also handles recursive loading for sub-circuits.

Parameters

<i>circuitId</i>	The unique identifier of the circuit to load components from.
------------------	---

Returns

A List of fully constructed [Component](#) objects.

6.4.2.6 loadConnections()

```
List< ConnectionRecord > com.logisim.data.CircuitDAO.loadConnections (
    long circuitId)
```

Retrieves the raw connection data for a specific circuit.

Parameters

<i>circuitId</i>	The unique identifier of the circuit.
------------------	---------------------------------------

Returns

A List of [ConnectionRecord](#) objects representing the connections.

6.4.2.7 saveCircuit()

```
void com.logisim.data.CircuitDAO.saveCircuit (
    Circuit circuit,
    long projectId)
```

Saves a new circuit and its associated contents to the database.

This method first inserts the circuit record. If successful, it retrieves the generated circuit ID and proceeds to save the circuit's components and connectors.

Parameters

<i>circuit</i>	The Circuit object containing the data to be saved.
<i>projectId</i>	The unique identifier of the project to which this circuit belongs.

6.4.2.8 saveComponents()

```
void com.logisim.data.CircuitDAO.saveComponents (
    Circuit circuit,
    long circuitId,
    Connection conn) throws SQLException [private]
```

Batch inserts the components of a circuit into the database.

Parameters

<i>circuit</i>	The circuit object containing the list of components.
<i>circuitId</i>	The database ID of the circuit these components belong to.

<i>conn</i>	The active database connection to be used for the operation.
-------------	--

Exceptions

<i>SQLException</i>	If a database access error occurs or the SQL execution fails.
---------------------	---

6.4.2.9 saveConnectors()

```
void com.logisim.data.CircuitDAO.saveConnectors (
    Circuit circuit,
    long circuitId,
    Connection conn) throws SQLException [private]
```

Batch inserts the connectors (wires) of a circuit into the database.

Parameters

<i>circuit</i>	The circuit object containing the list of connectors.
<i>circuitId</i>	The database ID of the circuit these connectors belong to.
<i>conn</i>	The active database connection to be used for the operation.

Exceptions

<i>SQLException</i>	If a database access error occurs or the SQL execution fails.
---------------------	---

6.4.2.10 updateCircuit()

```
void com.logisim.data.CircuitDAO.updateCircuit (
    Circuit circuit)
```

Updates an existing circuit in the database using a transactional approach.

This method performs the following steps:

1. Updates the circuit name.
2. Deletes all existing components associated with the circuit.
3. Deletes all existing connectors associated with the circuit.
4. Inserts the current state of components.
5. Inserts the current state of connectors.

Parameters

<i>circuit</i>	The Circuit object containing the updated data and ID.
----------------	--

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

- src/main/java/com/logisim/data/CircuitDAO.java

6.5 com.logisim.domain.components.Component Class Reference

An abstract base class representing a generic component within a logic circuit.

Inheritance diagram for com.logisim.domain.components.Component:



Public Member Functions

- [Component](#) ()
Default constructor.
- [Component](#) (String [name](#))
Constructs a component with a specified name.
- abstract void [execute](#) ()
Executes the logical operation of the component.
- void [setInput](#) (int index, boolean value)
Sets the state of a specific input pin.
- boolean [getOutput](#) (int index)
Retrieves the state of a specific output pin.
- String [getName](#) ()
Gets the name of the component.
- void [setName](#) (String [name](#))
Sets the name of the component.
- boolean[] [getInputs](#) ()
Retrieves the entire array of input pin states.
- void [setInputs](#) (boolean[] [inputs](#))
Sets the entire array of input pin states.
- boolean[] [getOutputs](#) ()
Retrieves the entire array of output pin states.
- void [setOutputs](#) (boolean[] [outputs](#))
Sets the entire array of output pin states.
- double [getPositionX](#) ()
Gets the X-coordinate of the component.
- double [getPositionY](#) ()
Gets the Y-coordinate of the component.
- void [setPositionX](#) (double [positionX](#))
Sets the X-coordinate of the component.
- void [setPositionY](#) (double [positionY](#))
Sets the Y-coordinate of the component.
- String [getUuid](#) ()
Gets the unique identifier (UUID) of the component.
- void [setUuid](#) (String [uuid](#))
Sets the unique identifier (UUID) of the component.

Protected Attributes

- String **name**
The name or type identifier of the component (e.g., "and", "or").
- boolean[] **inputs**
An array representing the state of the component's input pins.
- boolean[] **outputs**
An array representing the state of the component's output pins.
- double **positionX**
The X-coordinate of the component's position in the visual interface.
- double **positionY**
The Y-coordinate of the component's position in the visual interface.

Private Attributes

- String **uuid**
A unique identifier (UUID) assigned to this specific component instance.

6.5.1 Detailed Description

An abstract base class representing a generic component within a logic circuit.

This class provides the foundational structure for all specific logic gates and components (e.g., AND, OR, [Switch](#)). It manages common properties such as the component's unique identifier, its visual position on the canvas, and the state of its input and output pins.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Component() [1/2]

```
com.logisim.domain.components.Component.Component ()
```

Default constructor.

Initializes the component with a newly generated UUID.

6.5.2.2 Component() [2/2]

```
com.logisim.domain.components.Component.Component (
    String name)
```

Constructs a component with a specified name.

Initializes the component with the given name and a newly generated UUID.

Parameters

<i>name</i>	The name or type identifier for this component.
-------------	---

6.5.3 Member Function Documentation

6.5.3.1 execute()

```
abstract void com.logisim.domain.components.Component.execute () [abstract]
```

Executes the logical operation of the component.

Implementation classes must define this method to update the `outputs` array based on the current values in the `inputs` array.

Reimplemented in [com.logisim.domain.components.And](#), [com.logisim.domain.components.Bulb](#), [com.logisim.domain.components.NoOpComponent](#), [com.logisim.domain.components.Or](#), [com.logisim.domain.components.SubCircuitComponent](#), and [com.logisim.domain.components.Xor](#).

6.5.3.2 getInputs()

```
boolean[] com.logisim.domain.components.Component.getInputs ()
```

Retrieves the entire array of input pin states.

Returns

A boolean array representing all inputs.

6.5.3.3 getName()

```
String com.logisim.domain.components.Component.getName ()
```

Gets the name of the component.

Returns

The string identifier of the component.

Reimplemented in [com.logisim.domain.components.Bulb](#), [com.logisim.domain.components.SubCircuitComponent](#), and [com.logisim.domain.components.Switch](#).

6.5.3.4 getOutput()

```
boolean com.logisim.domain.components.Component.getOutput (  
    int index)
```

Retrieves the state of a specific output pin.

Parameters

<i>index</i>	The index of the output pin to read.
--------------	--------------------------------------

Returns

true if the pin is high, false if low.

6.5.3.5 getOutputs()

```
boolean[] com.logisim.domain.components.Component.getOutputs ()
```

Retrieves the entire array of output pin states.

Returns

A boolean array representing all outputs.

6.5.3.6 getPositionX()

```
double com.logisim.domain.components.Component.getPositionX ()
```

Gets the X-coordinate of the component.

Returns

The horizontal position.

6.5.3.7 getPositionY()

```
double com.logisim.domain.components.Component.getPositionY ()
```

Gets the Y-coordinate of the component.

Returns

The vertical position.

6.5.3.8 getUuid()

```
String com.logisim.domain.components.Component.getUuid ()
```

Gets the unique identifier (UUID) of the component.

Returns

The UUID string.

6.5.3.9 setInput()

```
void com.logisim.domain.components.Component.setInput (  
    int index,  
    boolean value)
```

Sets the state of a specific input pin.

Parameters

<i>index</i>	The index of the input pin to update.
<i>value</i>	The new boolean state (high/low) for the pin.

6.5.3.10 setInputs()

```
void com.logisim.domain.components.Component.setInputs (  
    boolean[] inputs)
```

Sets the entire array of input pin states.

Parameters

<i>inputs</i>	The boolean array to replace the current inputs.
---------------	--

6.5.3.11 setName()

```
void com.logisim.domain.components.Component.setName (  
    String name)
```

Sets the name of the component.

Parameters

<i>name</i>	The new name or type identifier.
-------------	----------------------------------

6.5.3.12 setOutputs()

```
void com.logisim.domain.components.Component.setOutputs (  
    boolean[] outputs)
```

Sets the entire array of output pin states.

Parameters

<i>outputs</i>	The boolean array to replace the current outputs.
----------------	---

6.5.3.13 setPositionX()

```
void com.logisim.domain.components.Component.setPositionX (  
    double positionX)
```

Sets the X-coordinate of the component.

Parameters

<i>positionX</i>	The new horizontal position.
------------------	------------------------------

6.5.3.14 setPositionY()

```
void com.logisim.domain.components.Component.setPositionY (
    double positionY)
```

Sets the Y-coordinate of the component.

Parameters

<i>positionY</i>	The new vertical position.
------------------	----------------------------

6.5.3.15 setUuid()

```
void com.logisim.domain.components.Component.setUuid (
    String uuid)
```

Sets the unique identifier (UUID) of the component.

This is typically used when reloading a component from the database to restore its original ID.

Parameters

<i>uuid</i>	The UUID string to assign.
-------------	----------------------------

6.5.4 Member Data Documentation

6.5.4.1 inputs

```
boolean [] com.logisim.domain.components.Component.inputs [protected]
```

An array representing the state of the component's input pins.

`true` represents a high signal (1), and `false` represents a low signal (0).

6.5.4.2 outputs

```
boolean [] com.logisim.domain.components.Component.outputs [protected]
```

An array representing the state of the component's output pins.

This is typically updated by the `execute()` method.

6.5.4.3 uuid

```
String com.logisim.domain.components.Component.uuid [private]
```

A unique identifier (UUID) assigned to this specific component instance.

Used for identifying the component during persistence and connection mapping.

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

- src/main/java/com/logisim/domain/components/Component.java

6.6 com.logisim.ui.logic.ConnectionManager Class Reference

Manages the interactive state for creating connections (wires) between components.

Public Member Functions

- [ConnectionManager](#) (Pane [canvasPane](#))
Constructs a new [ConnectionManager](#).
- void [setOnConnectionAdded](#) (Consumer< [Connector](#) > listener)
Sets the callback listener to be executed when a connection is successfully created.
- void [onMouseMove](#) (MouseEvent event)
Updates the temporary interaction line to follow the mouse cursor.
- void [cancelConnection](#) ()
Cancels the current connection attempt.
- void [handlePortClick](#) ([Port](#) clickedPort)
Handles click events on ports to initiate or finalize a connection.

Private Member Functions

- void [createConnection](#) ([Port](#) source, [Port](#) sink)
Finalizes the creation of a connection between two ports.

Private Attributes

- [Port](#) [selectedSourcePort](#) = null
The port selected as the starting point (Source/Output) of the connection.
- Pane [canvasPane](#)
The visual pane where wires and interaction lines are drawn.
- Consumer< [Connector](#) > [onConnectionAdded](#)
Callback listener that is triggered when a connection is successfully finalized.
- Line [interactionLine](#)
A temporary line used to visualize the wire being dragged by the user before the connection is finalized.

6.6.1 Detailed Description

Manages the interactive state for creating connections (wires) between components.

This class handles the two-step process of wiring: selecting a source output port and then selecting a destination input port. It provides visual feedback via a temporary interaction line that follows the mouse cursor and handles the validation logic to ensure connections are valid (e.g., Output to Input, no self-loops).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 ConnectionManager()

```
com.logisim.ui.logic.ConnectionManager.ConnectionManager (  
    Pane canvasPane)
```

Constructs a new [ConnectionManager](#).

Parameters

<i>canvasPane</i>	The <code>Pane</code> used as the drawing surface for the circuit.
-------------------	--

6.6.3 Member Function Documentation

6.6.3.1 cancelConnection()

```
void com.logisim.ui.logic.ConnectionManager.cancelConnection ()
```

Cancels the current connection attempt.

Resets the internal state by deselecting the source port and removing the temporary interaction line from the canvas.

6.6.3.2 createConnection()

```
void com.logisim.ui.logic.ConnectionManager.createConnection (  
    Port source,  
    Port sink) [private]
```

Finalizes the creation of a connection between two ports.

This method:

1. Creates the visual [Wire](#) and adds it to the canvas.
2. Creates the logical [Connector](#) object mapping component indices.
3. Triggers the `onConnectionAdded` callback to update the circuit model.

Parameters

<i>source</i>	The source (output) port.
<i>sink</i>	The sink (input) port.

6.6.3.3 handlePortClick()

```
void com.logisim.ui.logic.ConnectionManager.handlePortClick (  
    Port clickedPort)
```

Handles click events on ports to initiate or finalize a connection.

The logic flows as follows:

1. **No Selection Active:** If the clicked port is an OUTPUT, it is selected as the source, and a temporary line is drawn. Inputs are ignored.
2. **Source Selected:** If the clicked port is an INPUT, resides on a different gate, and is not already connected, the connection is finalized. Otherwise, the operation is canceled.

Parameters

<i>clickedPort</i>	The <code>Port</code> that was clicked by the user.
--------------------	---

6.6.3.4 onMouseMove()

```
void com.logisim.ui.logic.ConnectionManager.onMouseMove (  
    MouseEvent event)
```

Updates the temporary interaction line to follow the mouse cursor.

This method should be called when the mouse moves over the canvas. If a connection is in progress, the end of the line snaps to the current mouse coordinates.

Parameters

<i>event</i>	The <code>MouseEvent</code> containing the current cursor position.
--------------	---

6.6.3.5 setOnConnectionAdded()

```
void com.logisim.ui.logic.ConnectionManager.setOnConnectionAdded (  
    Consumer< Connector > listener)
```

Sets the callback listener to be executed when a connection is successfully created.

Parameters

<i>listener</i>	A Consumer that accepts a Connector object.
-----------------	---

6.6.4 Member Data Documentation

6.6.4.1 onConnectionAdded

```
Consumer<Connector> com.logisim.ui.logic.ConnectionManager.onConnectionAdded [private]
```

Callback listener that is triggered when a connection is successfully finalized.

This allows the logical circuit model to be updated.

6.6.4.2 selectedSourcePort

```
Port com.logisim.ui.logic.ConnectionManager.selectedSourcePort = null [private]
```

The port selected as the starting point (Source/Output) of the connection.

If `null`, no connection is currently being created.

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

- `src/main/java/com/logisim/ui/logic/ConnectionManager.java`

6.7 com.logisim.domain.Connector Class Reference

Represents a wire connection between two components in the circuit.

Public Member Functions

- [Connector](#) ([Component](#) sourceComp, [Component](#) sinkComp, int source, int sink)
Constructs a new [Connector](#) between two components.
- void [process](#) ()
Transmits the signal from the source to the sink.
- String [getName](#) ()
Retrieves the name of the connector.
- int [getSource](#) ()
Retrieves the index of the output pin on the source component.
- int [getSink](#) ()
Retrieves the index of the input pin on the sink component.
- [Component](#) [getSourceComp](#) ()
Retrieves the source component associated with this connection.
- [Component](#) [getSinkComp](#) ()
Retrieves the sink component associated with this connection.

Protected Attributes

- String **name**
- int **source**
- int **sink**
- [Component](#) **sourceComp**
- [Component](#) **sinkComp**

6.7.1 Detailed Description

Represents a wire connection between two components in the circuit.

A [Connector](#) facilitates the transmission of a logic signal (boolean state) from a specific output pin of a source component to a specific input pin of a sink component.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Connector()

```
com.logisim.domain.Connector.Connector (  
    Component sourceComp,  
    Component sinkComp,  
    int source,  
    int sink)
```

Constructs a new [Connector](#) between two components.

Parameters

<i>sourceComp</i>	The component generating the signal (source).
<i>sinkComp</i>	The component receiving the signal (sink).
<i>source</i>	The index of the output pin on the source component.
<i>sink</i>	The index of the input pin on the sink component.

6.7.3 Member Function Documentation

6.7.3.1 getName()

```
String com.logisim.domain.Connector.getName ()
```

Retrieves the name of the connector.

Returns

The name string (default "Connector").

6.7.3.2 `getSink()`

```
int com.logisim.domain.Connector.getSink ()
```

Retrieves the index of the input pin on the sink component.

Returns

The integer index of the sink pin.

6.7.3.3 `getSinkComp()`

```
Component com.logisim.domain.Connector.getSinkComp ()
```

Retrieves the sink component associated with this connection.

Returns

The [Component](#) instance acting as the sink.

6.7.3.4 `getSource()`

```
int com.logisim.domain.Connector.getSource ()
```

Retrieves the index of the output pin on the source component.

Returns

The integer index of the source pin.

6.7.3.5 `getSourceComp()`

```
Component com.logisim.domain.Connector.getSourceComp ()
```

Retrieves the source component associated with this connection.

Returns

The [Component](#) instance acting as the source.

6.7.3.6 `process()`

```
void com.logisim.domain.Connector.process ()
```

Transmits the signal from the source to the sink.

This method retrieves the current boolean value from the source component's output pin (specified by `source`) and applies it to the sink component's input pin (specified by `sink`).

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

- `src/main/java/com/logisim/domain/Connector.java`

6.8 com.logisim.data.DatabaseManager Class Reference

Manages the SQLite database connection and schema initialization for the application.

Public Member Functions

- Connection [getConnection](#) () throws SQLException
Establishes and returns a new connection to the SQLite database.

Static Public Member Functions

- static [DatabaseManager getInstance](#) ()
Retrieves the singleton instance of the [DatabaseManager](#).

Private Member Functions

- [DatabaseManager](#) ()
Private constructor to enforce the Singleton design pattern.
- void [createTables](#) ()
Initializes the database schema by creating required tables if they do not exist.

Static Private Attributes

- static final String [url](#)
The JDBC connection URL string pointing to the SQLite database file located in the user's home directory.
- static [DatabaseManager instance](#)
The single instance of the [DatabaseManager](#) class.

6.8.1 Detailed Description

Manages the SQLite database connection and schema initialization for the application.

This class implements the Singleton design pattern to ensure a centralized point of access for database operations. It handles the creation of the database file (stored in the user's home directory) and initializes the necessary tables if they do not already exist.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 DatabaseManager()

```
com.logisim.data.DatabaseManager.DatabaseManager () [private]
```

Private constructor to enforce the Singleton design pattern.

When instantiated, it automatically attempts to create the necessary database tables.

6.8.3 Member Function Documentation

6.8.3.1 createTables()

```
void com.logisim.data.DatabaseManager.createTables () [private]
```

Initializes the database schema by creating required tables if they do not exist.

The following tables are created:

- **projects**: Stores project metadata.
- **circuits**: Stores circuits linked to projects.
- **components**: Stores individual components within circuits.
- **connectors**: Stores wiring connections between components.

6.8.3.2 getConnection()

```
Connection com.logisim.data.DatabaseManager.getConnection () throws SQLException
```

Establishes and returns a new connection to the SQLite database.

Returns

A `Connection` object connected to the database.

Exceptions

<i>SQLException</i>	If a database access error occurs.
---------------------	------------------------------------

6.8.3.3 getInstance()

```
DatabaseManager com.logisim.data.DatabaseManager.getInstance () [static]
```

Retrieves the singleton instance of the `DatabaseManager`.

If the instance does not exist, it is created.

Returns

The single instance of `DatabaseManager`.

6.8.4 Member Data Documentation

6.8.4.1 url

```
final String com.logisim.data.DatabaseManager.url [static], [private]
```

Initial value:

```
=
    "jdbc:sqlite:" + System.getProperty("user.home") + "/logisim_data.db"
```

The JDBC connection URL string pointing to the SQLite database file located in the user's home directory.

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

- src/main/java/com/logisim/data/DatabaseManager.java

6.9 com.logisim.ui.components.GateFactory Class Reference

A factory class responsible for creating and configuring the visual representations of logic components (gates, switches, bulbs, etc.) on the UI canvas.

Classes

- class **Delta**
Helper class to store the offset during drag operations.

Static Public Member Functions

- static void [setConnectionManager](#) ([ConnectionManager](#) cm)
Sets the [ConnectionManager](#) used to handle port interactions.
- static [StackPane](#) [createGateWithHitBox](#) (String gateName, double x, double y, [Pane](#) canvasPane, [GridController](#) gridController, [Component](#) component, [Consumer](#)< [StackPane](#) > onDeleteAction, [Consumer](#)< [StackPane](#) > onToggleAction)
Creates a visual component ([StackPane](#)) representing a logic gate or device.
- static void [refreshComponentState](#) ([StackPane](#) visualGate)
Updates the visual state of a component ([Switch](#) or [Bulb](#)) based on its logical state.

Static Private Member Functions

- static [StackPane](#) [createSubCircuitVisual](#) (double x, double y, [com.logisim.domain.components.SubCircuitComponent](#) subComp, [Pane](#) canvasPane, [GridController](#) gridController, [java.util.function.Consumer](#)< [StackPane](#) > onDeleteAction)
Creates the specific visual representation for a [SubCircuitComponent](#).
- static void [setupSwitchInteraction](#) ([StackPane](#) stack, [ImageView](#) view, [Switch](#) switchLogic, [Consumer](#)< [StackPane](#) > onToggleAction)
Configures the mouse interaction for [Switch](#) components.
- static void [addPortsToGate](#) ([StackPane](#) stack)
Dynamically adds [Port](#) objects to a component's visual stack.
- static void [configurePortEvents](#) ([Port](#) port)
Binds mouse click events on ports to the [ConnectionManager](#).
- static void [makeDraggableandDeletable](#) ([StackPane](#) node, [Pane](#) canvasPane, [GridController](#) gridController, [Consumer](#)< [StackPane](#) > onDeleteAction)
Adds drag-and-drop functionality and a context menu for deletion to the component.

Static Private Attributes

- static [ConnectionManager](#) `connectionManager`
- static final double `GATE_VISUAL_SIZE` = 100.0

6.9.1 Detailed Description

A factory class responsible for creating and configuring the visual representations of logic components (gates, switches, bulbs, etc.) on the UI canvas.

This class handles:

- Loading the appropriate images for components.
- Creating dynamic visual structures for sub-circuits.
- Attaching input/output [Ports](#) to components.
- Setting up event handlers for dragging, clicking, and context menus.

6.9.2 Member Function Documentation

6.9.2.1 `addPortsToGate()`

```
void com.logisim.ui.components.GateFactory.addPortsToGate (  
    StackPane stack) [static], [private]
```

Dynamically adds [Port](#) objects to a component's visual stack.

This method calculates the physical positions of input pins (left side) and output pins (right side) based on the number of inputs/outputs defined in the component's logic.

Parameters

<i>stack</i>	The component's visual container.
--------------	-----------------------------------

6.9.2.2 `configurePortEvents()`

```
void com.logisim.ui.components.GateFactory.configurePortEvents (  
    Port port) [static], [private]
```

Binds mouse click events on ports to the [ConnectionManager](#).

Parameters

<i>port</i>	The port to configure.
-------------	------------------------

6.9.2.3 createGateWithHitBox()

```
StackPane com.logisim.ui.components.GateFactory.createGateWithHitBox (
    String gateName,
    double x,
    double y,
    Pane canvasPane,
    GridController gridController,
    Component component,
    Consumer< StackPane > onDeleteAction,
    Consumer< StackPane > onToggleAction) [static]
```

Creates a visual component (StackPane) representing a logic gate or device.

This is the main entry point for adding components to the UI. It determines the specific type of component (Standard Gate, [Switch](#), [Bulb](#), or [SubCircuit](#)), creates the visual elements, attaches ports, and configures user interactions (dragging, toggling, deleting).

Parameters

<i>gateName</i>	The string identifier for the gate type (used for image loading).
<i>x</i>	The initial X coordinate on the canvas.
<i>y</i>	The initial Y coordinate on the canvas.
<i>canvasPane</i>	The parent pane where this component will be added (used for drag bounds/calculations).
<i>gridController</i>	The controller used for snapping the component to the grid.
<i>component</i>	The underlying logical Component object.
<i>onDeleteAction</i>	A callback function to execute when the delete context menu item is clicked.
<i>onToggleAction</i>	A callback function to execute when a switch is toggled (can be null for non-switches).

Returns

A [StackPane](#) containing the visual elements of the component.

6.9.2.4 createSubCircuitVisual()

```
StackPane com.logisim.ui.components.GateFactory.createSubCircuitVisual (
    double x,
    double y,
    com.logisim.domain.components.SubCircuitComponent subComp,
    Pane canvasPane,
    GridController gridController,
    java.util.function.Consumer< StackPane > onDeleteAction) [static], [private]
```

Creates the specific visual representation for a [SubCircuitComponent](#).

Unlike standard gates which use static images, sub-circuits are drawn dynamically as a rectangle with text labels and a variable number of input/output pins.

Parameters

<i>x</i>	The initial X coordinate.
<i>y</i>	The initial Y coordinate.
<i>subComp</i>	The sub-circuit logical component.
<i>canvasPane</i>	The canvas pane.
<i>gridController</i>	The grid controller for snapping.
<i>onDeleteAction</i>	The callback for deletion.

Returns

A `StackPane` representing the sub-circuit.

6.9.2.5 makeDraggableandDeletable()

```
void com.logisim.ui.components.GateFactory.makeDraggableandDeletable (
    StackPane node,
    Pane canvasPane,
    GridController gridController,
    Consumer< StackPane > onDeleteAction) [static], [private]
```

Adds drag-and-drop functionality and a context menu for deletion to the component.

Dragging snaps the component to the grid defined by the `GridController`. The context menu allows removing the component via the `onDeleteAction`.

Parameters

<i>node</i>	The visual component to make interactive.
<i>canvasPane</i>	The pane containing the component.
<i>gridController</i>	The controller for grid snapping calculations.
<i>onDeleteAction</i>	The callback to execute on deletion.

6.9.2.6 refreshComponentState()

```
void com.logisim.ui.components.GateFactory.refreshComponentState (
    StackPane visualGate) [static]
```

Updates the visual state of a component (`Switch` or `Bulb`) based on its logical state.

This is typically called during or after circuit simulation to reflect signal changes (e.g., turning a bulb on or off).

Parameters

<i>visualGate</i>	The visual <code>StackPane</code> of the component.
-------------------	---

6.9.2.7 setConnectionManager()

```
void com.logisim.ui.components.GateFactory.setConnectionManager (  
    ConnectionManager cm) [static]
```

Sets the [ConnectionManager](#) used to handle port interactions.

Parameters

<i>cm</i>	The connection manager instance.
-----------	----------------------------------

6.9.2.8 setupSwitchInteraction()

```
void com.logisim.ui.components.GateFactory.setupSwitchInteraction (
    StackPane stack,
    ImageView view,
    Switch switchLogic,
    Consumer< StackPane > onToggleAction) [static], [private]
```

Configures the mouse interaction for [Switch](#) components.

Sets up a click handler that toggles the logical state of the switch, updates the visual image (on/off), and triggers the provided callback.

Parameters

<i>stack</i>	The visual container of the switch.
<i>view</i>	The ImageView to update.
<i>switchLogic</i>	The underlying switch logical component.
<i>onToggleAction</i>	The callback to execute after toggling.

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

- `src/main/java/com/logisim/ui/components/GateFactory.java`

6.10 com.logisim.ui.controllers.GridController Class Reference

Controller responsible for managing the background grid of the circuit editor.

Public Member Functions

- [GridController](#) (Canvas [gridCanvas](#), int [gridSize](#))
Constructs a new [GridController](#).
- void [drawGrid](#) ()
Renders the grid lines onto the canvas.
- double [snap](#) (double value)
Aligns a raw coordinate value to the nearest grid line.
- int [getGridSize](#) ()
Retrieves the configured grid size.

Private Attributes

- final Canvas **gridCanvas**
The canvas element where the grid lines are drawn.
- final int **gridSize**
The spacing between grid lines in pixels (cell size).

6.10.1 Detailed Description

Controller responsible for managing the background grid of the circuit editor.

This class handles the rendering of the visual grid lines on a JavaFX `Canvas`. It also provides utility methods for "snapping" coordinates to the nearest grid points, ensuring that components align neatly within the workspace.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `GridController()`

```
com.logisim.ui.controllers.GridController.GridController (
    Canvas gridCanvas,
    int gridSize)
```

Constructs a new [GridController](#).

Initializes the controller with a target canvas and specific grid size. It also sets up listeners on the canvas's width and height properties to automatically redraw the grid whenever the window is resized.

Parameters

<i>gridCanvas</i>	The JavaFX <code>Canvas</code> to draw the grid upon.
<i>gridSize</i>	The distance in pixels between grid lines.

6.10.3 Member Function Documentation

6.10.3.1 `drawGrid()`

```
void com.logisim.ui.controllers.GridController.drawGrid ()
```

Renders the grid lines onto the canvas.

This method first clears the entire canvas and then draws vertical and horizontal lines spaced by `gridSize`. The lines are drawn with a light gray color to serve as a subtle visual guide.

6.10.3.2 `getGridSize()`

```
int com.logisim.ui.controllers.GridController.getGridSize ()
```

Retrieves the configured grid size.

Returns

The size of the grid cells in pixels.

6.10.3.3 `snap()`

```
double com.logisim.ui.controllers.GridController.snap (
    double value)
```

Aligns a raw coordinate value to the nearest grid line.

This method rounds the provided value to the nearest multiple of `gridSize`. It is used to ensure components snap to the grid when dragged or placed.

Parameters

<i>value</i>	The raw coordinate value (X or Y).
--------------	------------------------------------

Returns

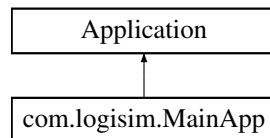
The coordinate value rounded to the nearest grid interval.

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

- `src/main/java/com/logisim/ui/controllers/GridController.java`

6.11 `com.logisim.MainApp` Class Reference

Inheritance diagram for `com.logisim.MainApp`:



Public Member Functions

- void **start** (Stage primaryStage) throws Exception

Static Public Member Functions

- static void **main** (String[] args)

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

- `src/main/java/com/logisim/MainApp.java`

6.12 `com.logisim.ui.controllers.MainViewController` Class Reference

The primary controller for the [Circuit](#) Editor view.

Public Member Functions

- [Circuit loadFullCircuitFromDB](#) (long id)
Loads a complete circuit model from the database to be used as a sub-circuit logic container.
- void [handleAnalyze](#) ()
Performs a combinatorial logic analysis of the circuit.
- void [initialize](#) ()
Initializes the controller.
- VBox [getSidebar](#) ()
- void [setSidebar](#) (VBox sidebar)
- Pane [getCanvasPane](#) ()
- void [setCanvasPane](#) (Pane canvasPane)
- ScrollPane [getCanvasScrollPane](#) ()
- void [setCanvasScrollPane](#) (ScrollPane canvasScrollPane)
- Button [getBtnAnd](#) ()
- void [setBtnAnd](#) (Button btnAnd)
- Button [getBtnOr](#) ()
- void [setBtnOr](#) (Button btnOr)
- Button [getBtnNot](#) ()
- void [setBtnNot](#) (Button btnNot)
- Canvas [getGridCanvas](#) ()
- void [setGridCanvas](#) (Canvas gridCanvas)
- [GridController](#) [getGridController](#) ()
- void [setGridController](#) ([GridController](#) gridController)
- [Project](#) [getCurrentProject](#) ()
- void [setCurrentProject](#) ([Project](#) currentProject)
- void [setContext](#) ([Project](#) project, [Circuit](#) circuit)
Initializes the view with a specific [Project](#) and [Circuit](#) context.

Static Public Member Functions

- static int [getGridsize](#) ()

Private Member Functions

- void [refreshSubCircuitSidebar](#) ()
Refreshes the sidebar with buttons to insert other circuits from the current project.
- void [spawnSubCircuit](#) ([Circuit](#) template)
Instantiates a sub-circuit component and places it on the canvas.
- void [handleBackToDashboard](#) ()
Navigates the user back to the [Project](#) Dashboard.
- void [handleSave](#) ()
Saves the current state of the circuit (layout, components, and connections) to the database.
- void [handleRun](#) ()
Triggers a simulation cycle for the current circuit.
- void [showAlert](#) (String title, String content)
Displays a generic information alert dialog.
- void [showAnalysisWindow](#) (boolean[][] rawData, List< String > headers, String expression)
Helper method to display the analysis results in a dedicated window.
- void [handleDeleteGate](#) (StackPane visualGate)
Handles the deletion of a component from the canvas and circuit logic.
- void [handleToggleSwitch](#) (StackPane visualGate)
Callback handler for toggling a switch component.
- [Port](#) [findPort](#) (StackPane gate, boolean isInput, int index)
Helper to find a specific [Port](#) inside a Gate's StackPane.

Private Attributes

- VBox **sidebar**
- Pane **canvasPane**
- ScrollPane **canvasScrollPane**
- Button **btnAnd**
- Button **btnOr**
- Button **btnNot**
- Button **btnSwitch**
- Button **btnBulb**
- Canvas **gridCanvas**
- VBox **subCircuitContainer**
- [GridController](#) **gridController**
- [Project](#) **currentProject**
- [Circuit](#) **currentCircuit**
- [CircuitDAO](#) **circuitDAO** = new [CircuitDAO](#)()

Static Private Attributes

- static final int **gridSize** = 20

6.12.1 Detailed Description

The primary controller for the [Circuit](#) Editor view.

This class manages the user interface interaction for creating, editing, and simulating digital logic circuits. It handles component placement, wiring, database persistence, and navigation within a project.

6.12.2 Member Function Documentation

6.12.2.1 findPort()

```
Port com.logisim.ui.controllers.MainViewController.findPort (
    StackPane gate,
    boolean isInput,
    int index) [private]
```

Helper to find a specific [Port](#) inside a Gate's StackPane.

Parameters

<i>gate</i>	The StackPane representing the gate
<i>isInput</i>	True if we want an input port, False for output
<i>index</i>	The index (e.g., 0 for top input, 1 for bottom input)

Returns

The [Port](#) object if found, otherwise null.

6.12.2.2 handleAnalyze()

```
void com.logisim.ui.controllers.MainViewController.handleAnalyze ()
```

Performs a combinatorial logic analysis of the circuit.

Identifies input switches and output bulbs, generates a truth table, and derives a boolean expression. Results are shown in a new window.

6.12.2.3 handleBackToDashboard()

```
void com.logisim.ui.controllers.MainViewController.handleBackToDashboard () [private]
```

Navigates the user back to the [Project](#) Dashboard.

This method loads the dashboard FXML and passes the current project context back to it.

6.12.2.4 handleDeleteGate()

```
void com.logisim.ui.controllers.MainViewController.handleDeleteGate (
    StackPane visualGate) [private]
```

Handles the deletion of a component from the canvas and circuit logic.

Also removes any wires attached to the deleted component.

Parameters

<i>visualGate</i>	The visual <code>StackPane</code> element to be removed.
-------------------	--

6.12.2.5 handleRun()

```
void com.logisim.ui.controllers.MainViewController.handleRun () [private]
```

Triggers a simulation cycle for the current circuit.

Updates the logical state of all components and refreshes the visual state (e.g., bulb images) to reflect the new logic values.

6.12.2.6 handleToggleSwitch()

```
void com.logisim.ui.controllers.MainViewController.handleToggleSwitch (
    StackPane visualGate) [private]
```

Callback handler for toggling a switch component.

Triggers a simulation run to propagate the new switch state through the circuit.

Parameters

<i>visualGate</i>	The visual element of the switch that was toggled.
-------------------	--

6.12.2.7 initialize()

```
void com.logisim.ui.controllers.MainViewController.initialize ()
```

Initializes the controller.

Sets up the grid controller, connection manager, event listeners for resizing, and assigns action handlers to the component toolbar buttons.

6.12.2.8 loadFullCircuitFromDB()

```
Circuit com.logisim.ui.controllers.MainViewController.loadFullCircuitFromDB (
    long id)
```

Loads a complete circuit model from the database to be used as a sub-circuit logic container.

Parameters

<i>id</i>	The database ID of the circuit to load.
-----------	---

Returns

A fully constructed [Circuit](#) object with components and logic connections.

6.12.2.9 refreshSubCircuitSidebar()

```
void com.logisim.ui.controllers.MainViewController.refreshSubCircuitSidebar () [private]
```

Refreshes the sidebar with buttons to insert other circuits from the current project.

This allows the user to use other existing circuits as "Sub-Circuits" within the current design. The method filters out the current circuit to prevent recursive inclusion.

6.12.2.10 setContext()

```
void com.logisim.ui.controllers.MainViewController.setContext (
    Project project,
    Circuit circuit)
```

Initializes the view with a specific [Project](#) and [Circuit](#) context.

This method loads the circuit's existing components and connections from the database, reconstructs their visual representations on the canvas, and refreshes the sidebar.

Parameters

<i>project</i>	The Project containing the circuit.
<i>circuit</i>	The Circuit to be edited.

6.12.2.11 showAlert()

```
void com.logisim.ui.controllers.MainViewController.showAlert (
    String title,
    String content) [private]
```

Displays a generic information alert dialog.

Parameters

<i>title</i>	The title of the dialog window.
<i>content</i>	The message content to display.

6.12.2.12 showAnalysisWindow()

```
void com.logisim.ui.controllers.MainViewController.showAnalysisWindow (
    boolean rawData[][],
    List< String > headers,
    String expression) [private]
```

Helper method to display the analysis results in a dedicated window.

Parameters

<i>rawData</i>	The 2D boolean array representing the truth table.
<i>headers</i>	The list of column headers (Input names and Output names).
<i>expression</i>	The derived boolean expression string.

6.12.2.13 spawnSubCircuit()

```
void com.logisim.ui.controllers.MainViewController.spawnSubCircuit (
    Circuit template) [private]
```

Instantiates a sub-circuit component and places it on the canvas.

Parameters

<i>template</i>	The Circuit metadata object representing the circuit to import.
-----------------	---

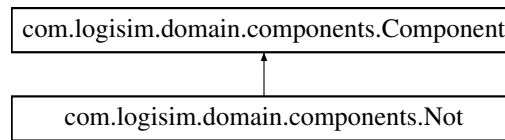
The documentation for this class was generated from the following file:

- src/main/java/com/logisim/ui/controllers/MainViewController.java

6.13 com.logisim.domain.components.Not Class Reference

Represents a logical NOT gate (inverter) in the circuit simulation.

Inheritance diagram for com.logisim.domain.components.Not:



Public Member Functions

- [Not](#) ()
Constructs a new NOT gate component with default settings.
- void [setInput](#) (boolean value)
Sets the state of the component's single input pin.
- boolean [getOutput](#) ()
Retrieves the state of the component's single output pin.
- void [execute](#) ()
Executes the logical operation of the NOT gate.

Public Member Functions inherited from [com.logisim.domain.components.Component](#)

- [Component](#) ()
Default constructor.
- [Component](#) (String name)
Constructs a component with a specified name.
- void [setInput](#) (int index, boolean value)
Sets the state of a specific input pin.
- boolean [getOutput](#) (int index)
Retrieves the state of a specific output pin.
- String [getName](#) ()
Gets the name of the component.
- void [setName](#) (String name)
Sets the name of the component.
- boolean[] [getInputs](#) ()
Retrieves the entire array of input pin states.
- void [setInputs](#) (boolean[] inputs)
Sets the entire array of input pin states.
- boolean[] [getOutputs](#) ()
Retrieves the entire array of output pin states.
- void [setOutputs](#) (boolean[] outputs)
Sets the entire array of output pin states.
- double [getPositionX](#) ()
Gets the X-coordinate of the component.
- double [getPositionY](#) ()
Gets the Y-coordinate of the component.

- void `setPositionX` (double `positionX`)
Sets the X-coordinate of the component.
- void `setPositionY` (double `positionY`)
Sets the Y-coordinate of the component.
- String `getUuid` ()
Gets the unique identifier (UUID) of the component.
- void `setUuid` (String `uuid`)
Sets the unique identifier (UUID) of the component.

Additional Inherited Members

Protected Attributes inherited from [com.logisim.domain.components.Component](#)

- String **name**
The name or type identifier of the component (e.g., "and", "or").
- boolean[] **inputs**
An array representing the state of the component's input pins.
- boolean[] **outputs**
An array representing the state of the component's output pins.
- double **positionX**
The X-coordinate of the component's position in the visual interface.
- double **positionY**
The Y-coordinate of the component's position in the visual interface.

6.13.1 Detailed Description

Represents a logical NOT gate (inverter) in the circuit simulation.

A NOT gate implements logical negation. It has a single input and a single output. The output is always the inverse of the input (i.e., if input is true, output is false; if input is false, output is true).

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Not()

```
com.logisim.domain.components.Not.Not ()
```

Constructs a new NOT gate component with default settings.

Initializes the component with:

- Name: "not"
- Input array size: 1
- Output array size: 1
- Default position: (100, 100)
- Initial state: Input set to false, resulting in Output set to true.

6.13.3 Member Function Documentation

6.13.3.1 `execute()`

```
void com.logisim.domain.components.Not.execute ()
```

Executes the logical operation of the NOT gate.

This method inverts the value found at input index 0 and stores the result at output index 0.

Reimplemented from [com.logisim.domain.components.Component](#).

6.13.3.2 `getOutput()`

```
boolean com.logisim.domain.components.Not.getOutput ()
```

Retrieves the state of the component's single output pin.

This is a convenience wrapper around [Component#getOutput\(int\)](#) targeting index 0.

Returns

`true` if the output is high, `false` otherwise.

6.13.3.3 `setInput()`

```
void com.logisim.domain.components.Not.setInput (
    boolean value)
```

Sets the state of the component's single input pin.

This is a convenience wrapper around [Component#setInput\(int, boolean\)](#) targeting index 0.

Parameters

<i>value</i>	The boolean value to apply to the input.
--------------	--

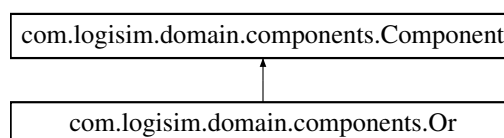
The documentation for this class was generated from the following file:

- `src/main/java/com/logisim/domain/components/Not.java`

6.14 `com.logisim.domain.components.Or` Class Reference

Represents a logical OR gate component within the circuit simulation.

Inheritance diagram for `com.logisim.domain.components.Or`:



Public Member Functions

- [Or](#) ()
Constructs a new OR gate component with default settings.
- boolean [getOutput](#) ()
Retrieves the current state of the OR gate's output pin.
- void [execute](#) ()
Executes the logical operation of the OR gate.

Public Member Functions inherited from [com.logisim.domain.components.Component](#)

- [Component](#) ()
Default constructor.
- [Component](#) (String [name](#))
Constructs a component with a specified name.
- void [setInput](#) (int [index](#), boolean [value](#))
Sets the state of a specific input pin.
- boolean [getOutput](#) (int [index](#))
Retrieves the state of a specific output pin.
- String [getName](#) ()
Gets the name of the component.
- void [setName](#) (String [name](#))
Sets the name of the component.
- boolean[] [getInputs](#) ()
Retrieves the entire array of input pin states.
- void [setInputs](#) (boolean[] [inputs](#))
Sets the entire array of input pin states.
- boolean[] [getOutputs](#) ()
Retrieves the entire array of output pin states.
- void [setOutputs](#) (boolean[] [outputs](#))
Sets the entire array of output pin states.
- double [getPositionX](#) ()
Gets the X-coordinate of the component.
- double [getPositionY](#) ()
Gets the Y-coordinate of the component.
- void [setPositionX](#) (double [positionX](#))
Sets the X-coordinate of the component.
- void [setPositionY](#) (double [positionY](#))
Sets the Y-coordinate of the component.
- String [getUuid](#) ()
Gets the unique identifier (UUID) of the component.
- void [setUuid](#) (String [uuid](#))
Sets the unique identifier (UUID) of the component.

Additional Inherited Members

Protected Attributes inherited from [com.logisim.domain.components.Component](#)

- String **name**
The name or type identifier of the component (e.g., "and", "or").
- boolean[] **inputs**
An array representing the state of the component's input pins.
- boolean[] **outputs**
An array representing the state of the component's output pins.
- double **positionX**
The X-coordinate of the component's position in the visual interface.
- double **positionY**
The Y-coordinate of the component's position in the visual interface.

6.14.1 Detailed Description

Represents a logical OR gate component within the circuit simulation.

This component accepts two boolean inputs and produces a single boolean output. The output is true if at least one of the inputs is true. It is false only if both inputs are false.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Or()

```
com.logisim.domain.components.Or.Or ()
```

Constructs a new OR gate component with default settings.

Initializes the component with:

- Name: "or"
- Input array size: 2
- Output array size: 1
- Default position: (100, 100)
- Initial state: All inputs and outputs set to false.

6.14.3 Member Function Documentation

6.14.3.1 execute()

```
void com.logisim.domain.components.Or.execute ()
```

Executes the logical operation of the OR gate.

This method updates the internal output state based on the current values of the input pins. The output at index 0 is set to `true` if either input index 0 or input index 1 (or both) are `true`.

Reimplemented from [com.logisim.domain.components.Component](#).

6.14.3.2 `getOutput()`

```
boolean com.logisim.domain.components.Or.getOutput ()
```

Retrieves the current state of the OR gate's output pin.

This is a convenience method that delegates to the superclass's `Component#getOutput (int)` method with index 0.

Returns

`true` if the gate is outputting a high signal, `false` otherwise.

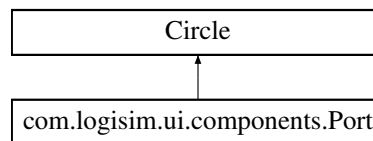
The documentation for this class was generated from the following file:

- `src/main/java/com/logisim/domain/components/Or.java`

6.15 com.logisim.ui.components.Port Class Reference

Represents a specific connection point (pin) on a visual logic component.

Inheritance diagram for `com.logisim.ui.components.Port`:



Public Member Functions

- `Port (boolean isInput, StackPane parentGate, int index)`
Constructs a new [Port](#) instance.
- `boolean isInput ()`
Checks if this port is an input.
- `boolean getConnectionState ()`
Retrieves the current connection state or signal value of the port.
- `void setConnectionState (boolean value)`
Sets the connection state or signal value for this port.
- `void setSelected (boolean value)`
Sets the selection status of the port.
- `StackPane getParentGate ()`
Retrieves the visual component that owns this port.
- `boolean isSelected ()`
Checks if the port is currently selected.
- `int getIndex ()`
Retrieves the logical index of this port.

Private Attributes

- boolean **isInput**
Indicates the direction of the port.
- boolean **selected**
Indicates whether this port is currently selected by the user (e.g., during the process of creating a connection).
- boolean **connectionState** = false
Represents the active state or signal value associated with this port.
- int **index**
The index of this port corresponding to the component's internal input or output arrays.
- StackPane **parentGate**
The visual container (StackPane) of the component to which this port belongs.

6.15.1 Detailed Description

Represents a specific connection point (pin) on a visual logic component.

A **Port** acts as an interface for creating wire connections between components. It is visually represented as a small circle. It maintains information about whether it is an input or output, its index within the component's logical structure, and its parent visual component.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Port()

```
com.logisim.ui.components.Port.Port (
    boolean isInput,
    StackPane parentGate,
    int index)
```

Constructs a new **Port** instance.

Initializes the port's visual style (transparent fill, blue stroke) and sets up mouse hover effects to provide visual feedback (turning yellow on hover).

Parameters

<i>isInput</i>	true if this is an input pin, false for output.
<i>parentGate</i>	The visual <code>StackPane</code> of the owning component.
<i>index</i>	The logical index of this pin on the component.

6.15.3 Member Function Documentation

6.15.3.1 getConnectionState()

```
boolean com.logisim.ui.components.Port.getConnectionState ()
```

Retrieves the current connection state or signal value of the port.

Returns

The boolean state of the connection.

6.15.3.2 getIndex()

```
int com.logisim.ui.components.Port.getIndex ()
```

Retrieves the logical index of this port.

This index corresponds to the `inputs []` or `outputs []` array in the underlying logic component.

Returns

The integer index.

6.15.3.3 getParentGate()

```
StackPane com.logisim.ui.components.Port.getParentGate ()
```

Retrieves the visual component that owns this port.

Returns

The parent `StackPane`.

6.15.3.4 isInput()

```
boolean com.logisim.ui.components.Port.isInput ()
```

Checks if this port is an input.

Returns

`true` if input, `false` if output.

6.15.3.5 isSelected()

```
boolean com.logisim.ui.components.Port.isSelected ()
```

Checks if the port is currently selected.

Returns

`true` if selected, `false` otherwise.

6.15.3.6 setConnectionState()

```
void com.logisim.ui.components.Port.setConnectionState (  
    boolean value)
```

Sets the connection state or signal value for this port.

Parameters

<i>value</i>	The new boolean state.
--------------	------------------------

6.15.3.7 setSelected()

```
void com.logisim.ui.components.Port.setSelected (
    boolean value)
```

Sets the selection status of the port.

Used by the connection manager to highlight ports during wiring.

Parameters

<i>value</i>	true to mark as selected, false otherwise.
--------------	--

6.15.4 Member Data Documentation

6.15.4.1 isInput

```
boolean com.logisim.ui.components.Port.isInput [private]
```

Indicates the direction of the port.

true if this is an input port; false if it is an output port.

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

- src/main/java/com/logisim/ui/components/Port.java

6.16 com.logisim.domain.Project Class Reference

Represents a project within the Logisim application.

Public Member Functions

- void `save` ()
Persists the current project state to the database.
- void `load` ()
Loads the project data from the database.
- void `export` ()
Exports the project's circuits to JPG image files.
- `Project` (String `name`)
Constructs a new `Project` with a specified name.
- `Project` (long `id`, String `name`)
Constructs a new `Project` with a specified ID and name.
- `Project` ()
Default constructor.
- String `getName` ()
Retrieves the name of the project.
- List< `Circuit` > `getCircuits` ()
Retrieves the list of circuits contained in this project.
- long `getId` ()
Retrieves the unique identifier of the project.
- void `setId` (long `id`)
Sets the unique identifier of the project.
- void `setName` (String `name`)
Sets the name of the project.
- void `setCircuits` (List< `Circuit` > `circuits`)
Sets the list of circuits for this project.
- `ProjectDAO` `getProjectdao` ()
Retrieves the Data Access Object associated with this project.
- void `setProjectdao` (`ProjectDAO` `projectdao`)
Sets the Data Access Object for this project.
- String `toString` ()
Returns a string representation of the project.

Private Attributes

- long `id`
The unique identifier for the project, typically assigned by the database.
- String `name`
The display name of the project.
- List< `Circuit` > `circuits` = new ArrayList<>()
The list of circuits associated with this project.
- `ProjectDAO` `projectdao` = new `ProjectDAO`()
The Data Access Object responsible for persisting project data.

6.16.1 Detailed Description

Represents a project within the Logisim application.

A `Project` serves as a container for multiple `Circuit` instances. It allows users to organize related circuits together. This class also handles high-level persistence operations by delegating to the `ProjectDAO`.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Project() [1/3]

```
com.logisim.domain.Project.Project (
    String name)
```

Constructs a new [Project](#) with a specified name.

Parameters

<i>name</i>	The name to assign to the project.
-------------	------------------------------------

6.16.2.2 Project() [2/3]

```
com.logisim.domain.Project.Project (
    long id,
    String name)
```

Constructs a new [Project](#) with a specified ID and name.

This constructor is typically used when reconstructing a project object from database records.

Parameters

<i>id</i>	The database ID of the project.
<i>name</i>	The name of the project.

6.16.2.3 Project() [3/3]

```
com.logisim.domain.Project.Project ()
```

Default constructor.

Initializes the project with the default name "Project".

6.16.3 Member Function Documentation

6.16.3.1 export()

```
void com.logisim.domain.Project.export ()
```

Exports the project's circuits to JPG image files.

This method renders the circuit components and wires onto a `BufferedImage`. It simulates the visual layout used in the UI (approx 100x100 components) and draws orthogonal lines for wires.

6.16.3.2 getCircuits()

```
List< Circuit > com.logisim.domain.Project.getCircuits ()
```

Retrieves the list of circuits contained in this project.

Returns

A list of [Circuit](#) objects.

6.16.3.3 getId()

```
long com.logisim.domain.Project.getId ()
```

Retrieves the unique identifier of the project.

Returns

The project ID.

6.16.3.4 getName()

```
String com.logisim.domain.Project.getName ()
```

Retrieves the name of the project.

Returns

The project name.

6.16.3.5 getProjectdao()

```
ProjectDAO com.logisim.domain.Project.getProjectdao ()
```

Retrieves the Data Access Object associated with this project.

Returns

The [ProjectDAO](#) instance.

6.16.3.6 load()

```
void com.logisim.domain.Project.load ()
```

Loads the project data from the database.

This method refreshes the list of circuits associated with this project ID. It uses [CircuitDAO](#) to fetch components AND connections, reconstructing the full logical graph of the circuit.

6.16.3.7 `save()`

```
void com.logisim.domain.Project.save ()
```

Persists the current project state to the database.

This method delegates the saving operation to the internal `ProjectDAO` instance.

6.16.3.8 `setCircuits()`

```
void com.logisim.domain.Project.setCircuits (  
    List< Circuit > circuits)
```

Sets the list of circuits for this project.

Parameters

<i>circuits</i>	The new list of Circuit objects.
-----------------	--

6.16.3.9 setId()

```
void com.logisim.domain.Project.setId (  
    long id)
```

Sets the unique identifier of the project.

Parameters

<i>id</i>	The new project ID.
-----------	---------------------

6.16.3.10 setName()

```
void com.logisim.domain.Project.setName (  
    String name)
```

Sets the name of the project.

Parameters

<i>name</i>	The new project name.
-------------	-----------------------

6.16.3.11 setProjectdao()

```
void com.logisim.domain.Project.setProjectdao (  
    ProjectDAO projectdao)
```

Sets the Data Access Object for this project.

Parameters

<i>projectdao</i>	The ProjectDAO instance to be used for persistence.
-------------------	---

6.16.3.12 toString()

```
String com.logisim.domain.Project.toString ()
```

Returns a string representation of the project.

Returns

The name of the project.

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

- src/main/java/com/logisim/domain/Project.java

6.17 com.logisim.data.ProjectDAO Class Reference

Data Access Object (DAO) responsible for handling database operations related to [Project](#) entities.

Public Member Functions

- void [saveProject](#) ([Project](#) project)
Saves a new project and all its associated circuits to the database.
- List< [Project](#) > [getAllProjects](#) ()
Retrieves all projects stored in the database.
- void [deleteProject](#) (long id)
Deletes a project from the database by its unique identifier.

Private Attributes

- final [CircuitDAO](#) **circuitDAO** = new [CircuitDAO](#)()
Helper DAO used to manage the persistence of circuits belonging to a project.

6.17.1 Detailed Description

Data Access Object (DAO) responsible for handling database operations related to [Project](#) entities.

This class manages the persistence of projects and delegates the saving of associated circuits to the [CircuitDAO](#).

6.17.2 Member Function Documentation

6.17.2.1 deleteProject()

```
void com.logisim.data.ProjectDAO.deleteProject (  
    long id)
```

Deletes a project from the database by its unique identifier.

Depending on the database schema configuration (specifically cascading deletes), this operation usually removes all associated circuits and components as well.

Parameters

<i>id</i>	The unique database identifier of the project to be deleted.
-----------	--

6.17.2.2 getAllProjects()

```
List< Project > com.logisim.data.ProjectDAO.getAllProjects ()
```

Retrieves all projects stored in the database.

The results are ordered by their creation timestamp in descending order (newest projects first).

Returns

A List of [Project](#) objects populated with IDs and names.

6.17.2.3 saveProject()

```
void com.logisim.data.ProjectDAO.saveProject (  
    Project project)
```

Saves a new project and all its associated circuits to the database.

This method performs an SQL insertion for the project to generate a [Project](#) ID. Once the ID is obtained, it iterates through the circuits contained within the project object and saves them using the [CircuitDAO](#).

Parameters

<i>project</i>	The Project object containing the data to be persisted.
----------------	---

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

- src/main/java/com/logisim/data/ProjectDAO.java

6.18 com.logisim.ui.controllers.ProjectDashboardController Class Reference

Controller class for the [Project](#) Dashboard view.

Public Member Functions

- void [initialize](#) ()
Initializes the controller class.
- void [setProject](#) ([Project](#) project)
Sets the project context for this dashboard.

Private Member Functions

- void [refreshList](#) ()
Refreshes the ListView with the latest circuits from the database.
- void [handleDeleteCircuit](#) ()
Handles the "Delete Circuit" button action.
- void [handleNewCircuit](#) ()
Handles the "New Circuit" button action.
- void [handleOpenCircuit](#) ()
Handles the "Open Circuit" button action.
- void [handleExport](#) ()
Handles the "Export Project" button action.
- void [handleBack](#) ()
Handles the "Back" button action.
- void [openMainEditor](#) ([Circuit](#) circuit)
Transitions the scene to the Main [Circuit](#) Editor.
- void [styleDialog](#) (DialogPane dialogPane)
Applies the application CSS styles to a dialog pane.

Private Attributes

- Label **lblProjectName**
- ListView< [Circuit](#) > **circuitList**
- Button **btnOpen**
- Button **btnDelete**
- Button **btnExport**
- [Project](#) **currentProject**
The currently active project being managed in this dashboard.
- final [CircuitDAO](#) **circuitDao** = new [CircuitDAO](#)()
Data Access Object used for performing database operations on circuits.

6.18.1 Detailed Description

Controller class for the [Project](#) Dashboard view.

This class manages the UI that appears after a project is selected or created. It displays a list of circuits contained within the current project and provides functionality to open existing circuits, create new ones, delete them, or export the entire project.

6.18.2 Member Function Documentation

6.18.2.1 [handleBack](#)()

```
void com.logisim.ui.controllers.ProjectDashboardController.handleBack () [private]
```

Handles the "Back" button action.

Navigates the user back to the initial Start View ([Project](#) Selection screen).

6.18.2.2 handleDeleteCircuit()

```
void com.logisim.ui.controllers.ProjectDashboardController.handleDeleteCircuit () [private]
```

Handles the "Delete Circuit" button action.

Prompts the user with a confirmation dialog. If confirmed, the selected circuit is removed from the database, and the list is refreshed.

6.18.2.3 handleExport()

```
void com.logisim.ui.controllers.ProjectDashboardController.handleExport () [private]
```

Handles the "Export Project" button action.

Reloads the project data to ensure the latest state is captured, then executes the project-wide export to JPG function. Displays a success message upon completion.

6.18.2.4 handleNewCircuit()

```
void com.logisim.ui.controllers.ProjectDashboardController.handleNewCircuit () [private]
```

Handles the "New Circuit" button action.

Displays a text input dialog to the user. If a valid name is provided, a new circuit is created in the database under the current project, and the list is refreshed.

6.18.2.5 handleOpenCircuit()

```
void com.logisim.ui.controllers.ProjectDashboardController.handleOpenCircuit () [private]
```

Handles the "Open Circuit" button action.

Retrieves the currently selected circuit and triggers the transition to the main editor view.

6.18.2.6 initialize()

```
void com.logisim.ui.controllers.ProjectDashboardController.initialize ()
```

Initializes the controller class.

This method is automatically called after the FXML file has been loaded. It sets up a listener on the `circuitList` to enable or disable the "Open" and "Delete" buttons based on whether a circuit is currently selected.

6.18.2.7 openMainEditor()

```
void com.logisim.ui.controllers.ProjectDashboardController.openMainEditor (  
    Circuit circuit) [private]
```

Transitions the scene to the Main [Circuit](#) Editor.

Loads the `MainView.fxml`, initializes the [MainViewController](#), and passes the current project and selected circuit context to it.

Parameters

<i>circuit</i>	The circuit to be opened in the editor.
----------------	---

6.18.2.8 refreshList()

```
void com.logisim.ui.controllers.ProjectDashboardController.refreshList () [private]
```

Refreshes the ListView with the latest circuits from the database.

It queries the [CircuitDAO](#) for all circuits associated with the `currentProject`'s ID and populates the `circuitList`.

6.18.2.9 setProject()

```
void com.logisim.ui.controllers.ProjectDashboardController.setProject (  
    Project project)
```

Sets the project context for this dashboard.

This method updates the UI to display the name of the provided project and refreshes the list of associated circuits.

Parameters

<i>project</i>	The Project object to display and manage.
----------------	---

6.18.2.10 styleDialog()

```
void com.logisim.ui.controllers.ProjectDashboardController.styleDialog (  
    DialogPane dialogPane) [private]
```

Applies the application CSS styles to a dialog pane.

Parameters

<i>dialogPane</i>	The dialog pane to style.
-------------------	---------------------------

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

- `src/main/java/com/logisim/ui/controllers/ProjectDashboardController.java`

6.19 com.logisim.ui.logic.SafePoints Class Reference

Utility class providing logic for determining safe spawn locations for UI components.

Static Public Member Functions

- static final [Point2D](#) [getSafeSpawnPoint](#) (ScrollPane canvasScrollPane, Pane canvasPane, int gridSize)
Calculates a spawn point in the center of the currently visible viewport area.

6.19.1 Detailed Description

Utility class providing logic for determining safe spawn locations for UI components.

This class helps prevent usability issues where new components might appear off-screen or in obscured areas by calculating coordinates relative to the user's current viewport.

6.19.2 Member Function Documentation

6.19.2.1 getSafeSpawnPoint()

```
final Point2D com.logisim.ui.logic.SafePoints.getSafeSpawnPoint (
    ScrollPane canvasScrollPane,
    Pane canvasPane,
    int gridSize) [static]
```

Calculates a spawn point in the center of the currently visible viewport area.

This method calculates the center coordinates of the `canvasScrollPane`'s viewport, taking into account the current scroll position. It then adjusts these coordinates to snap to the nearest grid lines defined by `gridSize`. This ensures that when a user adds a component, it appears right in front of them, even if they have scrolled away from the origin (0,0).

Parameters

<i>canvasScrollPane</i>	The scroll pane containing the canvas. Used to determine visible area and scroll offsets.
<i>canvasPane</i>	The actual content pane representing the circuit canvas. Used to determine total dimensions.
<i>gridSize</i>	The size of the grid cells for snapping calculations.

Returns

A `Point2D` representing the calculated X and Y coordinates for spawning a component.

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

- `src/main/java/com/logisim/ui/logic/SafePoints.java`

6.20 com.logisim.ui.controllers.StartScreenController Class Reference

Controller class for the application's start screen.

Public Member Functions

- void [initialize](#) ()
Initializes the controller class.

Private Member Functions

- void [handleNewProject](#) ()
Handles the "New Project" button action.
- void [handleLoadProject](#) ()
Handles the "Load Project" button action.
- void [handleDeleteProject](#) ()
Handles the "Delete Project" action.
- void [showAlert](#) (String title, String content)
Helper method to display a styled information alert.
- void [loadDashboard](#) ([Project](#) project)
Transitions the scene to the [Project](#) Dashboard.
- void [styleDialog](#) (javafx.scene.control.DialogPane dialogPane)
Applies the application CSS styles to a dialog pane.

Private Attributes

- Button **btnNewProject**
- Button **btnLoadProject**

6.20.1 Detailed Description

Controller class for the application's start screen.

This class handles the initial user interactions when the application launches. It provides functionality to create new projects, load existing projects from the database, or delete existing projects. It manages the transition from the start screen to the main project dashboard.

6.20.2 Member Function Documentation

6.20.2.1 [handleDeleteProject\(\)](#)

```
void com.logisim.ui.controllers.StartScreenController.handleDeleteProject () [private]
```

Handles the "Delete Project" action.

Retrieves existing projects and prompts the user to select one for deletion. If confirmed, the project (and its associated data) is removed from the database via [ProjectDAO#deleteProject\(long\)](#).

6.20.2.2 handleLoadProject()

```
void com.logisim.ui.controllers.StartScreenController.handleLoadProject () [private]
```

Handles the "Load Project" button action.

Retrieves a list of existing projects from the database using [ProjectDAO](#). If projects exist, a choice dialog is displayed. Upon selection, the application transitions to the dashboard view for the selected project. If no projects exist, an alert is shown.

6.20.2.3 handleNewProject()

```
void com.logisim.ui.controllers.StartScreenController.handleNewProject () [private]
```

Handles the "New Project" button action.

Opens a dialog prompting the user for a project name. If a valid name is provided, a new [Project](#) is created, saved to the database, and the application transitions to the dashboard view for the new project.

6.20.2.4 initialize()

```
void com.logisim.ui.controllers.StartScreenController.initialize ()
```

Initializes the controller class.

This method is automatically called after the FXML file has been loaded. It assigns specific CSS style classes to the primary buttons to ensure consistent UI theming.

6.20.2.5 loadDashboard()

```
void com.logisim.ui.controllers.StartScreenController.loadDashboard (  
    Project project) [private]
```

Transitions the scene to the [Project](#) Dashboard.

Loads the `project_dashboard.fxml`, initializes the [ProjectDashboardController](#) with the selected project context, applies the application stylesheet, and updates the stage.

Parameters

<code>project</code>	The Project context to pass to the dashboard.
----------------------	---

6.20.2.6 showAlert()

```
void com.logisim.ui.controllers.StartScreenController.showAlert (  
    String title,  
    String content) [private]
```

Helper method to display a styled information alert.

Parameters

<i>title</i>	The title of the alert window.
<i>content</i>	The message content to be displayed.

6.20.2.7 styleDialog()

```
void com.logisim.ui.controllers.StartScreenController.styleDialog (
    javafx.scene.control.DialogPane dialogPane) [private]
```

Applies the application CSS styles to a dialog pane.

This ensures that pop-up dialogs match the overall theme of the application.

Parameters

<i>dialogPane</i>	The <code>javafx.scene.control.DialogPane</code> to style.
-------------------	--

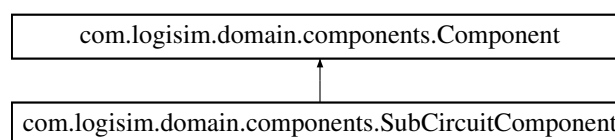
The documentation for this class was generated from the following file:

- `src/main/java/com/logisim/ui/controllers/StartScreenController.java`

6.21 com.logisim.domain.components.SubCircuitComponent Class Reference

Represents a complex circuit encapsulated as a single component within another circuit.

Inheritance diagram for `com.logisim.domain.components.SubCircuitComponent`:



Public Member Functions

- [SubCircuitComponent](#) ([Circuit](#) loadedCircuit)
Constructs a new [SubCircuitComponent](#) based on an existing circuit.
- void [execute](#) ()
Executes the logic of the encapsulated circuit.
- String [getName](#) ()
Retrieves the name of the component, which corresponds to the name of the inner circuit.
- long [getSourceCircuitId](#) ()
Gets the ID of the circuit used as the source for this component.
- [Circuit](#) [getInnerCircuit](#) ()
Retrieves the actual [Circuit](#) object being simulated internally.
- void [setInnerCircuit](#) ([Circuit](#) innerCircuit)

- Sets the inner circuit logic.*
 - void [setSourceCircuitId](#) (long [sourceCircuitId](#))
- Sets the source circuit ID.*
 - List< [Switch](#) > [getInternalSwitches](#) ()
- Retrieves the list of switches inside the inner circuit that act as inputs.*
 - void [setInternalSwitches](#) (List< [Switch](#) > [internalSwitches](#))
- Sets the list of internal switches.*
 - List< [Bulb](#) > [getInternalBulbs](#) ()
- Retrieves the list of bulbs inside the inner circuit that act as outputs.*
 - void [setInternalBulbs](#) (List< [Bulb](#) > [internalBulbs](#))
- Sets the list of internal bulbs.*

Public Member Functions inherited from [com.logisim.domain.components.Component](#)

- [Component](#) ()
 - Default constructor.*
- [Component](#) (String [name](#))
 - Constructs a component with a specified name.*
- void [setInput](#) (int index, boolean value)
 - Sets the state of a specific input pin.*
- boolean [getOutput](#) (int index)
 - Retrieves the state of a specific output pin.*
- void [setName](#) (String [name](#))
 - Sets the name of the component.*
- boolean[] [getInputs](#) ()
 - Retrieves the entire array of input pin states.*
- void [setInputs](#) (boolean[] [inputs](#))
 - Sets the entire array of input pin states.*
- boolean[] [getOutputs](#) ()
 - Retrieves the entire array of output pin states.*
- void [setOutputs](#) (boolean[] [outputs](#))
 - Sets the entire array of output pin states.*
- double [getPositionX](#) ()
 - Gets the X-coordinate of the component.*
- double [getPositionY](#) ()
 - Gets the Y-coordinate of the component.*
- void [setPositionX](#) (double [positionX](#))
 - Sets the X-coordinate of the component.*
- void [setPositionY](#) (double [positionY](#))
 - Sets the Y-coordinate of the component.*
- String [getUuid](#) ()
 - Gets the unique identifier (UUID) of the component.*
- void [setUuid](#) (String [uuid](#))
 - Sets the unique identifier (UUID) of the component.*

Private Attributes

- **Circuit** `innerCircuit`
The underlying circuit logic being wrapped by this component.
- long **sourceCircuitId**
The unique identifier of the source circuit in the database.
- List< **Switch** > **internalSwitches**
The list of switches within the inner circuit, serving as input interfaces.
- List< **Bulb** > **internalBulbs**
The list of bulbs within the inner circuit, serving as output interfaces.

Additional Inherited Members

Protected Attributes inherited from [com.logisim.domain.components.Component](#)

- String **name**
The name or type identifier of the component (e.g., "and", "or").
- boolean[] **inputs**
An array representing the state of the component's input pins.
- boolean[] **outputs**
An array representing the state of the component's output pins.
- double **positionX**
The X-coordinate of the component's position in the visual interface.
- double **positionY**
The Y-coordinate of the component's position in the visual interface.

6.21.1 Detailed Description

Represents a complex circuit encapsulated as a single component within another circuit.

This class allows for hierarchical circuit design by treating an entire [Circuit](#) as a "black box" component.

Mapping Logic:

- **Inputs:** [Switch](#) components inside the inner circuit act as input pins for this component.
- **Outputs:** [Bulb](#) components inside the inner circuit act as output pins for this component.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 `SubCircuitComponent()`

```
com.logisim.domain.components.SubCircuitComponent.SubCircuitComponent (
    Circuit loadedCircuit)
```

Constructs a new [SubCircuitComponent](#) based on an existing circuit.

This constructor scans the provided circuit for [Switch](#) and [Bulb](#) components. The size of the component's input array is determined by the number of switches, and the size of the output array is determined by the number of bulbs found.

Parameters

<i>loadedCircuit</i>	The fully loaded Circuit object to be encapsulated.
----------------------	---

6.21.3 Member Function Documentation

6.21.3.1 execute()

```
void com.logisim.domain.components.SubCircuitComponent.execute ()
```

Executes the logic of the encapsulated circuit.

This method performs three main steps:

1. Maps the values from this component's input pins to the internal switches of the inner circuit.
2. Simulates the inner circuit. It runs a loop proportional to the number of components to ensure signals propagate through the internal logic gates.
3. Maps the resulting states of the internal bulbs to this component's output pins.

Reimplemented from [com.logisim.domain.components.Component](#).

6.21.3.2 getInnerCircuit()

```
Circuit com.logisim.domain.components.SubCircuitComponent.getInnerCircuit ()
```

Retrieves the actual [Circuit](#) object being simulated internally.

Returns

The inner [Circuit](#) instance.

6.21.3.3 getInternalBulbs()

```
List< Bulb > com.logisim.domain.components.SubCircuitComponent.getInternalBulbs ()
```

Retrieves the list of bulbs inside the inner circuit that act as outputs.

Returns

A list of [Bulb](#) components.

6.21.3.4 getInternalSwitches()

```
List< Switch > com.logisim.domain.components.SubCircuitComponent.getInternalSwitches ()
```

Retrieves the list of switches inside the inner circuit that act as inputs.

Returns

A list of [Switch](#) components.

6.21.3.5 getName()

```
String com.logisim.domain.components.SubCircuitComponent.getName ()
```

Retrieves the name of the component, which corresponds to the name of the inner circuit.

Returns

The name of the encapsulated circuit.

Reimplemented from [com.logisim.domain.components.Component](#).

6.21.3.6 getSourceCircuitId()

```
long com.logisim.domain.components.SubCircuitComponent.getSourceCircuitId ()
```

Gets the ID of the circuit used as the source for this component.

Returns

The database ID of the inner circuit.

6.21.3.7 setInnerCircuit()

```
void com.logisim.domain.components.SubCircuitComponent.setInnerCircuit (  
    Circuit innerCircuit)
```

Sets the inner circuit logic.

Parameters

<i>innerCircuit</i>	The new Circuit to encapsulate.
---------------------	---

6.21.3.8 setInternalBulbs()

```
void com.logisim.domain.components.SubCircuitComponent.setInternalBulbs (  
    List< Bulb > internalBulbs)
```

Sets the list of internal bulbs.

Parameters

<i>internalBulbs</i>	The list of Bulb components.
----------------------	--

6.21.3.9 setInternalSwitches()

```
void com.logisim.domain.components.SubCircuitComponent.setInternalSwitches (  
    List< Switch > internalSwitches)
```

Sets the list of internal switches.

Parameters

<i>internalSwitches</i>	The list of Switch components.
-------------------------	--

6.21.3.10 setSourceCircuitId()

```
void com.logisim.domain.components.SubCircuitComponent.setSourceCircuitId (
    long sourceCircuitId)
```

Sets the source circuit ID.

Parameters

<i>sourceCircuitId</i>	The database ID of the circuit.
------------------------	---------------------------------

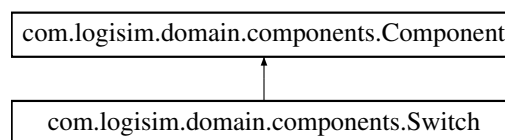
The documentation for this class was generated from the following file:

- src/main/java/com/logisim/domain/components/SubCircuitComponent.java

6.22 com.logisim.domain.components.Switch Class Reference

Represents a toggle switch component in the circuit simulation.

Inheritance diagram for com.logisim.domain.components.Switch:



Public Member Functions

- [Switch](#) ()
Constructs a new [Switch](#) component.
- boolean [isOn](#) ()
Checks if the switch is currently in the "On" state.
- void [execute](#) ()
Updates the component's output pin based on its internal state.
- void [toggle](#) ()
Toggles the current state of the switch.
- void [setState](#) (boolean state)
Explicitly sets the state of the switch.
- String [getName](#) ()
Retrieves the unique type name of this component.

Public Member Functions inherited from [com.logisim.domain.components.Component](#)

- [Component](#) ()
Default constructor.
- [Component](#) (String [name](#))
Constructs a component with a specified name.
- void [setInput](#) (int [index](#), boolean [value](#))
Sets the state of a specific input pin.
- boolean [getOutput](#) (int [index](#))
Retrieves the state of a specific output pin.
- void [setName](#) (String [name](#))
Sets the name of the component.
- boolean[] [getInputs](#) ()
Retrieves the entire array of input pin states.
- void [setInputs](#) (boolean[] [inputs](#))
Sets the entire array of input pin states.
- boolean[] [getOutputs](#) ()
Retrieves the entire array of output pin states.
- void [setOutputs](#) (boolean[] [outputs](#))
Sets the entire array of output pin states.
- double [getPositionX](#) ()
Gets the X-coordinate of the component.
- double [getPositionY](#) ()
Gets the Y-coordinate of the component.
- void [setPositionX](#) (double [positionX](#))
Sets the X-coordinate of the component.
- void [setPositionY](#) (double [positionY](#))
Sets the Y-coordinate of the component.
- String [getUuid](#) ()
Gets the unique identifier (UUID) of the component.
- void [setUuid](#) (String [uuid](#))
Sets the unique identifier (UUID) of the component.

Private Attributes

- boolean [isOn](#)
The internal state of the switch.

Additional Inherited Members

Protected Attributes inherited from [com.logisim.domain.components.Component](#)

- String **name**
The name or type identifier of the component (e.g., "and", "or").
- boolean[] [inputs](#)
An array representing the state of the component's input pins.
- boolean[] [outputs](#)
An array representing the state of the component's output pins.
- double **positionX**
The X-coordinate of the component's position in the visual interface.
- double **positionY**
The Y-coordinate of the component's position in the visual interface.

6.22.1 Detailed Description

Represents a toggle switch component in the circuit simulation.

A [Switch](#) acts as a primary input source for a circuit. It has no input pins but provides a single output pin. The output signal is determined by the internal state of the switch (On/High or Off/Low).

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Switch()

```
com.logisim.domain.components.Switch.Switch ()
```

Constructs a new [Switch](#) component.

Initializes the component with:

- Name: "switch"
- Input array size: 0 (Switches do not receive signals from other components)
- Output array size: 1
- Initial state: Off (`false`)

6.22.3 Member Function Documentation

6.22.3.1 execute()

```
void com.logisim.domain.components.Switch.execute ()
```

Updates the component's output pin based on its internal state.

If the internal state `isOn` is true, the output at index 0 becomes true. Otherwise, it becomes false.

Reimplemented from [com.logisim.domain.components.Component](#).

6.22.3.2 getName()

```
String com.logisim.domain.components.Switch.getName ()
```

Retrieves the unique type name of this component.

Returns

The string literal "switch".

Reimplemented from [com.logisim.domain.components.Component](#).

6.22.3.3 isOn()

```
boolean com.logisim.domain.components.Switch.isOn ()
```

Checks if the switch is currently in the "On" state.

Returns

`true` if the switch is on, `false` otherwise.

6.22.3.4 setState()

```
void com.logisim.domain.components.Switch.setState (
    boolean state)
```

Explicitly sets the state of the switch.

This method is useful for programmatically controlling the switch, such as when it is used as an input interface for a [SubCircuitComponent](#). Triggers `execute()` to update the output pin.

Parameters

<code>state</code>	The new state to apply (<code>true</code> for On, <code>false</code> for Off).
--------------------	---

6.22.3.5 toggle()

```
void com.logisim.domain.components.Switch.toggle ()
```

Toggles the current state of the switch.

If the switch is On, it turns Off. If it is Off, it turns On. After changing the state, `execute()` is called immediately to update the output pin.

6.22.4 Member Data Documentation

6.22.4.1 isOn

```
boolean com.logisim.domain.components.Switch.isOn [private]
```

The internal state of the switch.

`true` indicates the switch is closed (On), generating a high signal. `false` indicates the switch is open (Off), generating a low signal.

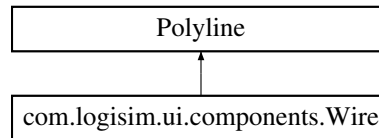
The documentation for this class was generated from the following file:

- `src/main/java/com/logisim/domain/components/Switch.java`

6.23 com.logisim.ui.components.Wire Class Reference

Represents a visual wire connection between two [Ports](#) on the circuit canvas.

Inheritance diagram for com.logisim.ui.components.Wire:



Public Member Functions

- [Wire](#) ([Port](#) source, [Port](#) sink)
Constructs a new [Wire](#) connecting a source port to a sink port.
- [Port getSource](#) ()
Retrieves the source port of this wire.
- [Port getSink](#) ()
Retrieves the sink port of this wire.

Private Member Functions

- void [updateWire](#) ()
Recalculates the geometric path of the wire based on the current positions of the ports.
- void [setupInteractions](#) ()
Configures mouse interactions for the wire.

Private Attributes

- final [Port](#) **source**
- final [Port](#) **sink**

6.23.1 Detailed Description

Represents a visual wire connection between two [Ports](#) on the circuit canvas.

This class extends [Polyline](#) to render a physical line representing the logic flow between components. It implements an orthogonal (Manhattan-style) routing algorithm to draw lines with right angles. The wire automatically updates its geometry whenever the connected components are moved.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Wire()

```
com.logisim.ui.components.Wire.Wire (
    Port source,
    Port sink)
```

Constructs a new [Wire](#) connecting a source port to a sink port.

This constructor initializes the wire's visual properties (color, width), attaches layout listeners to the parent components of the ports to handle movement, and sets up user interactions.

Parameters

<i>source</i>	The Port acting as the signal source.
<i>sink</i>	The Port acting as the signal sink.

6.23.3 Member Function Documentation

6.23.3.1 `getSink()`

```
Port com.logisim.ui.components.Wire.getSink ()
```

Retrieves the sink port of this wire.

Returns

The [Port](#) where the signal terminates.

6.23.3.2 `getSource()`

```
Port com.logisim.ui.components.Wire.getSource ()
```

Retrieves the source port of this wire.

Returns

The [Port](#) where the signal originates.

6.23.3.3 `setupInteractions()`

```
void com.logisim.ui.components.Wire.setupInteractions () [private]
```

Configures mouse interactions for the wire.

Adds handlers for:

- **Hover:** Increases stroke width to indicate focus.
- **Context Menu:** Provides options to change the wire color (Red, Blue, Green, etc.) or delete the wire.

6.23.3.4 `updateWire()`

```
void com.logisim.ui.components.Wire.updateWire () [private]
```

Recalculates the geometric path of the wire based on the current positions of the ports.

This method computes the absolute coordinates of the source and sink ports relative to their parent container. It then generates a 4-point path:

1. Start point (Source coordinates).
2. Midpoint 1 (Horizontal movement to the midpoint between X coordinates).
3. Midpoint 2 (Vertical movement to the target Y coordinate).
4. End point (Sink coordinates).

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

- `src/main/java/com/logisim/ui/components/Wire.java`

Index

- addComponent
 - com.logisim.domain.Circuit, [20](#)
- addConnection
 - com.logisim.domain.Circuit, [20](#), [21](#)
- addPortsToGate
 - com.logisim.ui.components.GateFactory, [44](#)
- analyze
 - com.logisim.domain.Circuit, [21](#)
- And
 - com.logisim.domain.components.And, [15](#)
- Bulb
 - com.logisim.domain.components.Bulb, [17](#)
- cancelConnection
 - com.logisim.ui.logic.ConnectionManager, [36](#)
- com.logisim.data.CircuitDAO, [24](#)
 - ConnectionRecord, [25](#)
 - createCircuit, [25](#)
 - deleteCircuit, [25](#)
 - getCircuitsByProjectId, [26](#)
 - loadComponents, [26](#)
 - loadConnections, [26](#)
 - saveCircuit, [27](#)
 - saveComponents, [27](#)
 - saveConnectors, [28](#)
 - updateCircuit, [28](#)
- com.logisim.data.DatabaseManager, [41](#)
 - createTables, [42](#)
 - DatabaseManager, [41](#)
 - getConnection, [42](#)
 - getInstance, [42](#)
 - url, [43](#)
- com.logisim.data.ProjectDAO, [70](#)
 - deleteProject, [70](#)
 - getAllProjects, [70](#)
 - saveProject, [71](#)
- com.logisim.domain.Circuit, [18](#)
 - addComponent, [20](#)
 - addConnection, [20](#), [21](#)
 - analyze, [21](#)
 - generateBooleanExpression, [21](#)
 - getComponents, [22](#)
 - getConnectors, [22](#)
 - getId, [22](#)
 - getName, [22](#)
 - removeComponent, [22](#)
 - setComponents, [23](#)
 - setConnectors, [23](#)
 - setId, [23](#)
 - setName, [23](#)
 - simulate, [24](#)
 - toString, [24](#)
- com.logisim.domain.components.And, [13](#)
 - And, [15](#)
 - execute, [15](#)
 - getOutput, [15](#)
- com.logisim.domain.components.Bulb, [16](#)
 - Bulb, [17](#)
 - execute, [18](#)
 - getName, [18](#)
 - isOn, [18](#)
- com.logisim.domain.components.Component, [29](#)
 - Component, [30](#)
 - execute, [31](#)
 - getInputs, [31](#)
 - getName, [31](#)
 - getOutput, [31](#)
 - getOutputs, [32](#)
 - getPositionX, [32](#)
 - getPositionY, [32](#)
 - getUuid, [32](#)
 - inputs, [34](#)
 - outputs, [34](#)
 - setInput, [32](#)
 - setInputs, [33](#)
 - setName, [33](#)
 - setOutputs, [33](#)
 - setPositionX, [33](#)
 - setPositionY, [34](#)
 - setUuid, [34](#)
 - uuid, [34](#)
- com.logisim.domain.components.Not, [56](#)
 - execute, [58](#)
 - getOutput, [58](#)
 - Not, [57](#)
 - setInput, [58](#)
- com.logisim.domain.components.Or, [58](#)
 - execute, [60](#)
 - getOutput, [60](#)
 - Or, [60](#)
- com.logisim.domain.components.SubCircuitComponent, [78](#)
 - execute, [81](#)
 - getInnerCircuit, [81](#)
 - getInternalBulbs, [81](#)
 - getInternalSwitches, [81](#)
 - getName, [81](#)
 - getSourceCircuitId, [82](#)

- setInnerCircuit, [82](#)
 - setInternalBulbs, [82](#)
 - setInternalSwitches, [82](#)
 - setSourceCircuitId, [83](#)
 - SubCircuitComponent, [80](#)
- com.logisim.domain.components.Switch, [83](#)
 - execute, [85](#)
 - getName, [85](#)
 - isOn, [85](#), [86](#)
 - setState, [86](#)
 - Switch, [85](#)
 - toggle, [86](#)
- com.logisim.domain.Connector, [38](#)
 - Connector, [39](#)
 - getName, [39](#)
 - getSink, [39](#)
 - getSinkComp, [40](#)
 - getSource, [40](#)
 - getSourceComp, [40](#)
 - process, [40](#)
- com.logisim.domain.Project, [64](#)
 - export, [66](#)
 - getCircuits, [66](#)
 - getId, [67](#)
 - getName, [67](#)
 - getProjectdao, [67](#)
 - load, [67](#)
 - Project, [66](#)
 - save, [67](#)
 - setCircuits, [68](#)
 - setId, [69](#)
 - setName, [69](#)
 - setProjectdao, [69](#)
 - toString, [69](#)
- com.logisim.MainApp, [50](#)
- com.logisim.ui.components.GateFactory, [43](#)
 - addPortsToGate, [44](#)
 - configurePortEvents, [44](#)
 - createGateWithHitBox, [44](#)
 - createSubCircuitVisual, [45](#)
 - makeDraggableandDeletable, [46](#)
 - refreshComponentState, [46](#)
 - setConnectionManager, [46](#)
 - setupSwitchInteraction, [48](#)
- com.logisim.ui.components.Port, [61](#)
 - getConnectionState, [62](#)
 - getIndex, [62](#)
 - getParentGate, [63](#)
 - isInput, [63](#), [64](#)
 - isSelected, [63](#)
 - Port, [62](#)
 - setConnectionState, [63](#)
 - setSelected, [64](#)
- com.logisim.ui.components.Wire, [87](#)
 - getSink, [88](#)
 - getSource, [88](#)
 - setupInteractions, [88](#)
 - updateWire, [88](#)
 - Wire, [87](#)
- com.logisim.ui.controllers.GridController, [48](#)
 - drawGrid, [49](#)
 - getGridSize, [49](#)
 - GridController, [49](#)
 - snap, [49](#)
- com.logisim.ui.controllers.MainViewController, [50](#)
 - findPort, [52](#)
 - handleAnalyze, [52](#)
 - handleBackToDashboard, [53](#)
 - handleDeleteGate, [53](#)
 - handleRun, [53](#)
 - handleToggleSwitch, [53](#)
 - initialize, [54](#)
 - loadFullCircuitFromDB, [54](#)
 - refreshSubCircuitSidebar, [54](#)
 - setContext, [54](#)
 - showAlert, [55](#)
 - showAnalysisWindow, [55](#)
 - spawnSubCircuit, [55](#)
- com.logisim.ui.controllers.ProjectDashboardController, [71](#)
 - handleBack, [72](#)
 - handleDeleteCircuit, [72](#)
 - handleExport, [73](#)
 - handleNewCircuit, [73](#)
 - handleOpenCircuit, [73](#)
 - initialize, [73](#)
 - openMainEditor, [73](#)
 - refreshList, [74](#)
 - setProject, [74](#)
 - styleDialog, [74](#)
- com.logisim.ui.controllers.StartScreenController, [75](#)
 - handleDeleteProject, [76](#)
 - handleLoadProject, [76](#)
 - handleNewProject, [77](#)
 - initialize, [77](#)
 - loadDashboard, [77](#)
 - showAlert, [77](#)
 - styleDialog, [78](#)
- com.logisim.ui.logic.ConnectionManager, [35](#)
 - cancelConnection, [36](#)
 - ConnectionManager, [36](#)
 - createConnection, [36](#)
 - handlePortClick, [37](#)
 - onConnectionAdded, [38](#)
 - onMouseMove, [37](#)
 - selectedSourcePort, [38](#)
 - setOnConnectionAdded, [37](#)
- com.logisim.ui.logic.SafePoints, [74](#)
 - getSafeSpawnPoint, [75](#)
- Component
 - com.logisim.domain.components.Component, [30](#)
- configurePortEvents
 - com.logisim.ui.components.GateFactory, [44](#)
- ConnectionManager
 - com.logisim.ui.logic.ConnectionManager, [36](#)
- ConnectionRecord

- com.logisim.data.CircuitDAO, 25
- Connector
 - com.logisim.domain.Connector, 39
- createCircuit
 - com.logisim.data.CircuitDAO, 25
- createConnection
 - com.logisim.ui.logic.ConnectionManager, 36
- createGateWithHitBox
 - com.logisim.ui.components.GateFactory, 44
- createSubCircuitVisual
 - com.logisim.ui.components.GateFactory, 45
- createTables
 - com.logisim.data.DatabaseManager, 42
- DatabaseManager
 - com.logisim.data.DatabaseManager, 41
- DejaVu fonts v2.37, 1
- deleteCircuit
 - com.logisim.data.CircuitDAO, 25
- deleteProject
 - com.logisim.data.ProjectDAO, 70
- drawGrid
 - com.logisim.ui.controllers.GridController, 49
- execute
 - com.logisim.domain.components.And, 15
 - com.logisim.domain.components.Bulb, 18
 - com.logisim.domain.components.Component, 31
 - com.logisim.domain.components.Not, 58
 - com.logisim.domain.components.Or, 60
 - com.logisim.domain.components.SubCircuitComponent, 81
 - com.logisim.domain.components.Switch, 85
- export
 - com.logisim.domain.Project, 66
- findPort
 - com.logisim.ui.controllers.MainViewController, 52
- generateBooleanExpression
 - com.logisim.domain.Circuit, 21
- getAllProjects
 - com.logisim.data.ProjectDAO, 70
- getCircuits
 - com.logisim.domain.Project, 66
- getCircuitsByProjectId
 - com.logisim.data.CircuitDAO, 26
- getComponents
 - com.logisim.domain.Circuit, 22
- getConnection
 - com.logisim.data.DatabaseManager, 42
- getConnectionState
 - com.logisim.ui.components.Port, 62
- getConnectors
 - com.logisim.domain.Circuit, 22
- getGridSize
 - com.logisim.ui.controllers.GridController, 49
- getId
 - com.logisim.domain.Circuit, 22
- com.logisim.domain.Project, 67
- getIndex
 - com.logisim.ui.components.Port, 62
- getInnerCircuit
 - com.logisim.domain.components.SubCircuitComponent, 81
- getInputs
 - com.logisim.domain.components.Component, 31
- getInstance
 - com.logisim.data.DatabaseManager, 42
- getInternalBulbs
 - com.logisim.domain.components.SubCircuitComponent, 81
- getInternalSwitches
 - com.logisim.domain.components.SubCircuitComponent, 81
- getName
 - com.logisim.domain.Circuit, 22
 - com.logisim.domain.components.Bulb, 18
 - com.logisim.domain.components.Component, 31
 - com.logisim.domain.components.SubCircuitComponent, 81
 - com.logisim.domain.components.Switch, 85
 - com.logisim.domain.Connector, 39
 - com.logisim.domain.Project, 67
- getOutput
 - com.logisim.domain.components.And, 15
 - com.logisim.domain.components.Component, 31
 - com.logisim.domain.components.Not, 58
 - com.logisim.domain.components.Or, 60
- getOutputs
 - com.logisim.domain.components.Component, 32
- getParentGate
 - com.logisim.ui.components.Port, 63
- getPositionX
 - com.logisim.domain.components.Component, 32
- getPositionY
 - com.logisim.domain.components.Component, 32
- getProjectdao
 - com.logisim.domain.Project, 67
- getSafeSpawnPoint
 - com.logisim.ui.logic.SafePoints, 75
- getSink
 - com.logisim.domain.Connector, 39
 - com.logisim.ui.components.Wire, 88
- getSinkComp
 - com.logisim.domain.Connector, 40
- getSource
 - com.logisim.domain.Connector, 40
 - com.logisim.ui.components.Wire, 88
- getSourceCircuitId
 - com.logisim.domain.components.SubCircuitComponent, 82
- getSourceComp
 - com.logisim.domain.Connector, 40
- getUuid
 - com.logisim.domain.components.Component, 32
- GridController

- com.logisim.ui.controllers.GridController, 49
- handleAnalyze
 - com.logisim.ui.controllers.MainViewController, 52
- handleBack
 - com.logisim.ui.controllers.ProjectDashboardController, 72
- handleBackToDashboard
 - com.logisim.ui.controllers.MainViewController, 53
- handleDeleteCircuit
 - com.logisim.ui.controllers.ProjectDashboardController, 72
- handleDeleteGate
 - com.logisim.ui.controllers.MainViewController, 53
- handleDeleteProject
 - com.logisim.ui.controllers.StartScreenController, 76
- handleExport
 - com.logisim.ui.controllers.ProjectDashboardController, 73
- handleLoadProject
 - com.logisim.ui.controllers.StartScreenController, 76
- handleNewCircuit
 - com.logisim.ui.controllers.ProjectDashboardController, 73
- handleNewProject
 - com.logisim.ui.controllers.StartScreenController, 77
- handleOpenCircuit
 - com.logisim.ui.controllers.ProjectDashboardController, 73
- handlePortClick
 - com.logisim.ui.logic.ConnectionManager, 37
- handleRun
 - com.logisim.ui.controllers.MainViewController, 53
- handleToggleSwitch
 - com.logisim.ui.controllers.MainViewController, 53
- initialize
 - com.logisim.ui.controllers.MainViewController, 54
 - com.logisim.ui.controllers.ProjectDashboardController, 73
 - com.logisim.ui.controllers.StartScreenController, 77
- inputs
 - com.logisim.domain.components.Component, 34
- isInput
 - com.logisim.ui.components.Port, 63, 64
- isOn
 - com.logisim.domain.components.Bulb, 18
 - com.logisim.domain.components.Switch, 85, 86
- isSelected
 - com.logisim.ui.components.Port, 63
- jQuery UI v1.14.1, 7
- jQuery v3.7.1, 5
- load
 - com.logisim.domain.Project, 67
 - loadComponents
 - com.logisim.data.CircuitDAO, 26
 - loadConnections
 - com.logisim.data.CircuitDAO, 26
 - loadDashboard
 - com.logisim.ui.controllers.StartScreenController, 77
 - loadFullCircuitFromDB
 - com.logisim.ui.controllers.MainViewController, 54
 - makeDraggableandDeletable
 - com.logisim.ui.components.GateFactory, 46
 - Not
 - com.logisim.domain.components.Not, 57
 - onConnectionAdded
 - com.logisim.ui.logic.ConnectionManager, 38
 - onMouseMove
 - com.logisim.ui.logic.ConnectionManager, 37
 - openMainEditor
 - com.logisim.ui.controllers.ProjectDashboardController, 73
 - Or
 - com.logisim.domain.components.Or, 60
 - outputs
 - com.logisim.domain.components.Component, 34
 - Port
 - com.logisim.ui.components.Port, 62
 - process
 - com.logisim.domain.Connector, 40
 - Project
 - com.logisim.domain.Project, 66
 - refreshComponentState
 - com.logisim.ui.components.GateFactory, 46
 - refreshList
 - com.logisim.ui.controllers.ProjectDashboardController, 74
 - refreshSubCircuitSidebar
 - com.logisim.ui.controllers.MainViewController, 54
 - removeComponent
 - com.logisim.domain.Circuit, 22
 - save
 - com.logisim.domain.Project, 67
 - saveCircuit
 - com.logisim.data.CircuitDAO, 27
 - saveComponents
 - com.logisim.data.CircuitDAO, 27
 - saveConnectors
 - com.logisim.data.CircuitDAO, 28
 - saveProject
 - com.logisim.data.ProjectDAO, 71
 - selectedSourcePort
 - com.logisim.ui.logic.ConnectionManager, 38
 - setCircuits
 - com.logisim.domain.Project, 68

- setComponents
 - com.logisim.domain.Circuit, 23
- setConnectionManager
 - com.logisim.ui.components.GateFactory, 46
- setConnectionState
 - com.logisim.ui.components.Port, 63
- setConnectors
 - com.logisim.domain.Circuit, 23
- setContext
 - com.logisim.ui.controllers.MainViewController, 54
- setId
 - com.logisim.domain.Circuit, 23
 - com.logisim.domain.Project, 69
- setInnerCircuit
 - com.logisim.domain.components.SubCircuitComponent, 82
- setInput
 - com.logisim.domain.components.Component, 32
 - com.logisim.domain.components.Not, 58
- setInputs
 - com.logisim.domain.components.Component, 33
- setInternalBulbs
 - com.logisim.domain.components.SubCircuitComponent, 82
- setInternalSwitches
 - com.logisim.domain.components.SubCircuitComponent, 82
- setName
 - com.logisim.domain.Circuit, 23
 - com.logisim.domain.components.Component, 33
 - com.logisim.domain.Project, 69
- setOnConnectionAdded
 - com.logisim.ui.logic.ConnectionManager, 37
- setOutputs
 - com.logisim.domain.components.Component, 33
- setPositionX
 - com.logisim.domain.components.Component, 33
- setPositionY
 - com.logisim.domain.components.Component, 34
- setProject
 - com.logisim.ui.controllers.ProjectDashboardController, 74
- setProjectdao
 - com.logisim.domain.Project, 69
- setSelected
 - com.logisim.ui.components.Port, 64
- setSourceCircuitId
 - com.logisim.domain.components.SubCircuitComponent, 83
- setState
 - com.logisim.domain.components.Switch, 86
- setupInteractions
 - com.logisim.ui.components.Wire, 88
- setupSwitchInteraction
 - com.logisim.ui.components.GateFactory, 48
- setUuid
 - com.logisim.domain.components.Component, 34
- showAlert
 - com.logisim.ui.controllers.MainViewController, 55
 - com.logisim.ui.controllers.StartScreenController, 77
- showAnalysisWindow
 - com.logisim.ui.controllers.MainViewController, 55
- simulate
 - com.logisim.domain.Circuit, 24
- snap
 - com.logisim.ui.controllers.GridController, 49
- spawnSubCircuit
 - com.logisim.ui.controllers.MainViewController, 55
- styleDialog
 - com.logisim.ui.controllers.ProjectDashboardController, 74
 - com.logisim.ui.controllers.StartScreenController, 78
- SubCircuitComponent
 - com.logisim.domain.components.SubCircuitComponent, 80
- Switch
 - com.logisim.domain.components.Switch, 85
- toggle
 - com.logisim.domain.components.Switch, 86
- toString
 - com.logisim.domain.Circuit, 24
 - com.logisim.domain.Project, 69
- updateCircuit
 - com.logisim.data.CircuitDAO, 28
- updateWire
 - com.logisim.ui.components.Wire, 88
- url
 - com.logisim.data.DatabaseManager, 43
- uuid
 - com.logisim.domain.components.Component, 34
- Wire
 - com.logisim.ui.components.Wire, 87