Comprehension Questions

Co	Comprehension Questions						
	Questions on the Qt system	Diagrams Shown ¹					
1.	What is the class stereotype assigned to <i>QSqlDatabase</i> ?	qt-D1 qt-D2					
2.	Which classes are dependent on <i>QPainter</i> ?	qt-G1 qt-G2					
3.	Is this statement true or false? The control class <i>QDragManager</i> is associated with the entity class <i>QPixmap</i> via the attribute <i>pm_cursor</i> .	qt-I1 qt-I2					
4.	Which class is involved in a composition relationship with <i>QAction</i> and is part of the <i>QEvent</i> hierarchy?	qt-E1 qt-E2					
5.	Qt's model/view architecture makes it possible to display the same data in different ways without changing the underlying data structure, delivering flexibility and scalability to large data sets.	qt-MV1 qt-MV2					
	Which concrete class (or classes) implement the view component in the model/view architecture? Select all that apply.						
6.	Select the class responsible for setting the central widget of a window.	qt-MW1 qt-MW2					
7.	Which abstract class is responsible for managing layout for the graphical user interface?	qt-L1 qt-L2					
8.	A view renders the contents of a model. You need to use a model with QListView. Which class would you use to represent the model?	qt-MV1 qt-MV2					
9.	SVG is an XML-based file format and language for describing 2 dimensional graphics.	qt-G1 qt-G2					
	Which class controls the drawing of SVG files to a paint device?						
10.	Which class provides an action that can be inserted into widgets?	qt-W1 qt-W2					
11.	Which concrete layout related classes and widget related classes are used in the Dialog box below?	qt-L1 qt-L2					

-

 $^{^{1}}$ One of the two diagrams were shown for each question. The first one is in multi-cluster layout, the second is in orthogonal layout

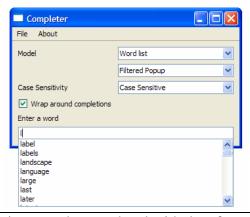
Select all layout related concrete classes and widget related concrete classes that apply. Basic Layouts File Horizontal layout Button 1 Button 3 Grid layout This widget takes up about two thirds of Line 1: Line 2: This widget takes up all the remaining space in the top-level layout. ОК Cancel 12. Which input event class stores information qt-E1 about which key on the keyboard was pressed? qt-E2 13. Ot allows data obtained from models to be qt-D1 related to specific widgets. Which class is qt-D2 responsible for setting up this data relation between SQL models and widgets? 14. Consider the Analog clock QT application qt-MW1 shown below. The main program is shown qt-MW2 below. #include < QApplication > #include "analogclock.h" int main(int argc, char *argv[]) { QApplication app(argc, argv); AnalogClock clock; clock.show(); return app.exec(); ∠-M Analog Clock We would like to add a 'second' hand to the clock. Which method does this change impact?

15. You need to add an auto completion facility to QT that provides completions to words entered into a line textbox or combobox widget. The completions are based on an item model. See the figure below for an example of the completer in action. You decide that the best way to do this is to create a new class *QCompleter* that provides this functionality.

qt-W1 qt-W2

Which widget classes need to change (i.e., add methods or variables) if *QCompleter* is added to QT?

Which class should *QCompleter* inherit from?

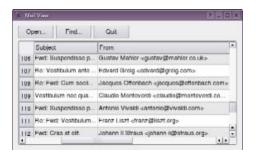


qt-MV1 qt-MV2

16. A view may be populated with data from any model such as a table or directory model. One such example of a view is shown below.

Notice how the rows have an alternating background color to them. In order to implement this functionality, you need to add a method with the following signature.

void setAlternatingRowColors(bool enable)



Which class would you add this method to?

17. You are given the following bug description:	qt-W1
Bug Title: Change foreground color of checkboxes inside tables Bug Description: When a checkbox is displayed in a table and the row is highlighted, the checkbox rectangle should be drawn with the same color as the text in the cell.	qt-W2
Which class (or classes) will most likely need to be changed in order to fix this bug?	
18. You need to extend Qt's functionality to include an image plugin to read and write images using an new image format you invented.	qt-I1 qt-I2
Which class do you need to subclass for this plugin?	
19. Syntax Highlighting is the use of appropriate fonts and colors to highlight different elements in a document. You need to create a new class CodeHighlighter that adds syntax highlighting to the <i>QTextEdit</i> widget. <i>QTextEdit</i> is used to edit and display both plain and rich text.	qt-W1 qt-W2
a) Which abstract class would you derive <i>CodeHighlighter</i> from?	
b) Which method would <i>CodeHighlighter</i> need to reimplement from the abstract class?	
c) You will need to instantiate <i>CodeHighlighter</i> with <i>QTextEdit</i> 's underlying <i>QTextDocument</i> as the parent. Which method in <i>QTextEdit</i> returns a text document?	
20. OpenGL is a standard API for rendering 3D graphics. To use OpenGL-enabled widgets in a Qt application, which class do you need to subclass?	qt-G1 qt-G2
21. You need to improve the design of the module shown in the following diagram and remove some redundancy.	qt-I1 qt-I2
Which method will most probably need to be defined in <i>QIODevice</i> from it's subclasses? (Note that the method can still be reimplemented in <i>QIODevice</i> subclasses as	

needed.) Should this field be private, public or protected? 22. The class *QHBox* provides horizontal geometry qt-L1 management for its child widgets. All the qt-L2 horizontal box's child widgets are placed alongside each other and sized accordingly. See the following example of *QHBox* in action. Button 1 Button 2 Now consider the following code fragments. Code Fragment 1 OHBox *hbox = new OHBox; OPushButton *child1 = new PushButton(hbox); QPushButton *child2 = new PushButton(hbox); Code Fragment 2 OWidget *hbox = new OWidget; OPushButton *child1 = new OPushButton; QPushButton *child2 = new OPushButton; QHBoxLayout *layout = new QHBoxLayout; layout->addWidget(child1); layout->addWidget(child2); hbox->setLayout(layout); a) True or False: Code Fragment 1 and Code Fragment 2 are equivalent. In other words, when executed, they produce the same result. b) True or False: *QHBox* can be removed without causing any code to break. Base your answer on information given in the diagram. c) True or False: *QHBoxLayout* can be removed without causing any code to break. Base vour answer on information given in the diagram.

	Questions on the wxWidgets System	Diagrams Shown ²
1.	What is the class stereotype assigned to	wx-D1
	wxDbTableInf?	wx -D2
2.	Which classes are dependent on wxRect?	wx-G1
		wx-G2
3.	Is this statement true or false?	wx-I1
	The boundary class wxDC is associated with	wx-I2
	the entity class $wxBitmap$ via the attribute $m_selectedBitmap$.	
4.	Which class is involved in an aggregation	wx-E1
	relationship with <i>wxObject</i> and is also associated with <i>wxHashTable</i> ?	wx-E2
5.	wxWidget's document/view architecture lets	wx-DV1
	you model your application primarily in terms	wx-DV2
	of documents, which store data and views,	
	which display and manipulate the data.	
	Which concrete class (or classes) implement	
	the <i>document</i> component in the document/view	
	architecture?	
6.	Splash screen is a term used to describe an	wx-MW1
	image that appears while a computer program	wx-MW2
	is loading. Splash screens sometimes do not	
	cover the entire screen, but only a rectangle near the center.	
	Select the class responsible for setting the	
	image/bitmap for the splash screen.	
	mage, ormap for the spacen sereem	
7.	Which abstract boundary class is responsible	wx-L1
	for managing layout for the graphical user	wx-L2
	interface?	
8.	Which class models the relationship between a	wx-DV1
	document class and a view class?	wx-DV2
9.	SVG is an XML-based file format and	wx-G1
	language for describing 2 dimensional graphics.	wx-G2
	Which class acts as an input handler for the	
	SVG file format?	
10.	Which class provides a control that models	wx-W1
	hierarchical data?	wx-W2

 $^{^{2}}$ One of the two diagrams were shown for each question. The first one is in multi-cluster layout, the second is in orthogonal layout

11. Which concrete layout related classes and	wx-L1
widget related classes are used in the Dialog	wx-L2
box below?	
con celow.	
Select all layout related concrete classes and	
widget related concrete classes that apply.	
GotoClass	
Class Name	
Matching Classes	
☐ Case Sensitive ☐ Nested Classes ☐ Non-Project Classes	
Ok Close	
12. Which command event class is used to give an	wx-E1
application the chance to update various user	wx-E2
interface elements?	
13. Which class lets you view information from a	wx-D1
database table in a grid?	wx-D2
database table in a grid:	WA DZ
14. Consider the Regions wxWidgets application	wx-MW1
	wx-MW2
shown below. The orange filled squares in the	W X-IVI W Z
first row and third column of the 'Regions'	
frame window need to be changed to blue.	
7771' 1	
Which method does this change impact?	

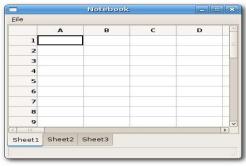
Regions	
15. You need to add undo/rado functionality to	wx-W1
15. You need to add undo/redo functionality to commands which are actions usually performed	wx-w1 wx-W2
by selecting a control such as a menu item,	WX-WZ
pressing a button or any other means provided	
by the application to change the data or view.	
Which classes need to change (i.e., add	
methods or attributes) if undo/redo is to be	
implemented?	DVI
16. You need to add a method to the	wx-DV1
document/view component that returns a reference to the list of documents. The	wx-DV2
signature of the method to be added is	
signature of the method to be added is	
<pre>wxList& GetDocuments()</pre>	
Which class would you add this method to?	
17. You are given the following bug description:	wx-W1
	wx-W2
Bug Title: SetForegroundColour doesn't update the	
control.	
Bug Description: wxStaticText::SetForegroundColour does not	
update the control to change the colour. The	
following code does not work after the control has	
been created.	
bool set_to_red;	
wxStaticText *t;	
<pre>t->SetForegroundColour(set_to_red ? wxColour("RED") : wxColour("GREEN"));</pre>	
Which class (or classes) will most likely need to be	
changed in order to fix this bug?	
18. You need to extend wxWidget's functionality	wx-I1

to include an image handler to read and write images using an new image format you invented. Which class do you need to subclass to support this new format?	wx-I2
19. wxSpellChecker is a generic spell-check component that can use different spell checking engines. You need to add engine interface support for the GNU 'Aspell' engine. A new class called <i>AspellInterface</i> needs to be created. a) Which class would you derive <i>AspellInterface</i> from?	wx-W1 wx-W2
b) Which method would need to be reimplemented in <i>AspellInterface</i> from the base class?	
c) You also need to add a dialog interface to your spellchecker similar to the one shown below. A new class <i>MySpellingDialog</i> needs to be created. Which class would you derive <i>MySpellingDialog</i> from?	
Spell Checker	
Not in Dictionary: tesst	
Change to: test	
Suggestions: Ignore Ignore All Iset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset Itset I	
Example showing spellchecker in action	
20. OpenGL is a standard API for rendering 3D graphics. Which class do you need to subclass to create an OpenGL enabled window?	wx-G1 wx-G2
21. You need to improve the design of the module shown in the following diagram and remove some redundancy.	wx-I1 wx-I2
Which attribute (field) will most probably need to be defined in <i>wxDC</i> from it's subclasses?	
Should this field be private, public or	

protected?

23. Sizers are used to define layout of controls in dialogs. wxNotebookSizer is a specialized sizer to make sizers work in connection with using notebooks. This sizer is different from any other sizer as you must not add any children to it - instead, it queries the notebook class itself. The only thing this sizer does is to determine the size of the biggest page of the notebook and report an adjusted minimal size. An example of a notebook control is shown below.

wx-L1 wx-L2



Now consider the following code fragments.

Code Fragment 1
f=wxFrame()

mainsizer=wxBoxSizer(wxVERTICAL)
f.SetSizer(mainsizer)

notebook=wxNotebook()
notesizer =
wxNotebookSizer(notebook)
mainsizer.Add(notesizer)

Code Fragment 2
f=wxFrame()
mainsizer=wxBoxSizer(wxVERTICAL)
f.SetSizer(mainsizer)

notebook=wxNotebook()
notesizer = wxBoxSizer(notebook)
mainsizer.Add(notesizer)

- a) True or False: Code Fragment 1 and Code Fragment 2 are equivalent. In other words, when executed, they produce the same result.
- b) True or False: wxNoteBookSizer can be removed without causing any code to break. Base your answer on information given in the diagram.
- c) True or False: *wxBoxSizer* can be removed without causing any code to break. Base your answer on information given in the diagram.

Preference RatingsFor preference ratings, the following template was used for each question for the 6 experts and for two of the questions in qt and wxWidgets for the main study.

Comprehension Rating	Very Useful	Somewhat Useful	Don't know	Somewhat Useless	Completely Useless
<system>- <module>1</module></system>					
<system>- <module>2</module></system>					
<system>- <module>3</module></system>					

Aesthetic Rating	Very Appealing	Somewhat Appealing	Don't know	Somewhat Unappealing	Very Unappealing
<system>- <module>1</module></system>					
<system>- <module>2</module></system>					
<system>- <module>3</module></system>					

Comments on your comprehension and aesthetic rating:

Best and Worst Aesthetics Ranking-qt

For each Diagram triplet, enter the diagram ID with the best and worst aesthetics. For your convenience, each triplet is shown on one page in thumbnail form and is also available in full size.

Diagrams (IDs)	Diagram with the Best Aesthetics	Diagram with the Worst Aesthetics	Comments
qt-D1 qt-D2 qt-D3			
qt-E1 qt-E2 qt-E3			
qt-MV1 qt-MV2 qt-MV3			
qt-G1 qt-G2 qt-G3			
qt-I1 qt-I2 qt-I3			
qt-L1 qt-L2 qt-L3			
qt-MW1 qt-MW2 qt-MW3			
qt-W1 qt-W2 qt-W3			

Comments about this questionnaire:

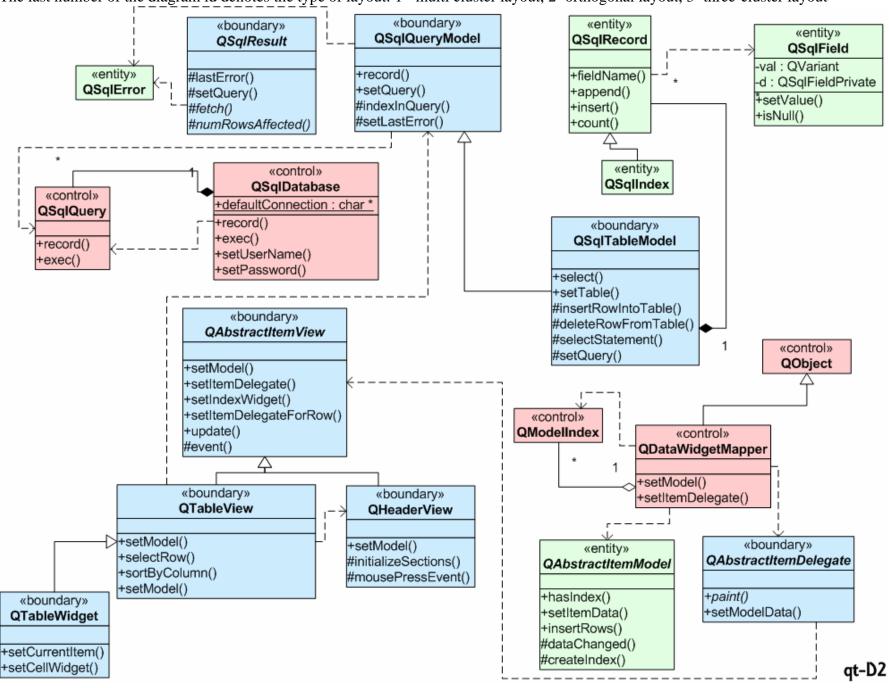
Best and Worst Aesthetics Ranking -wxWidgets

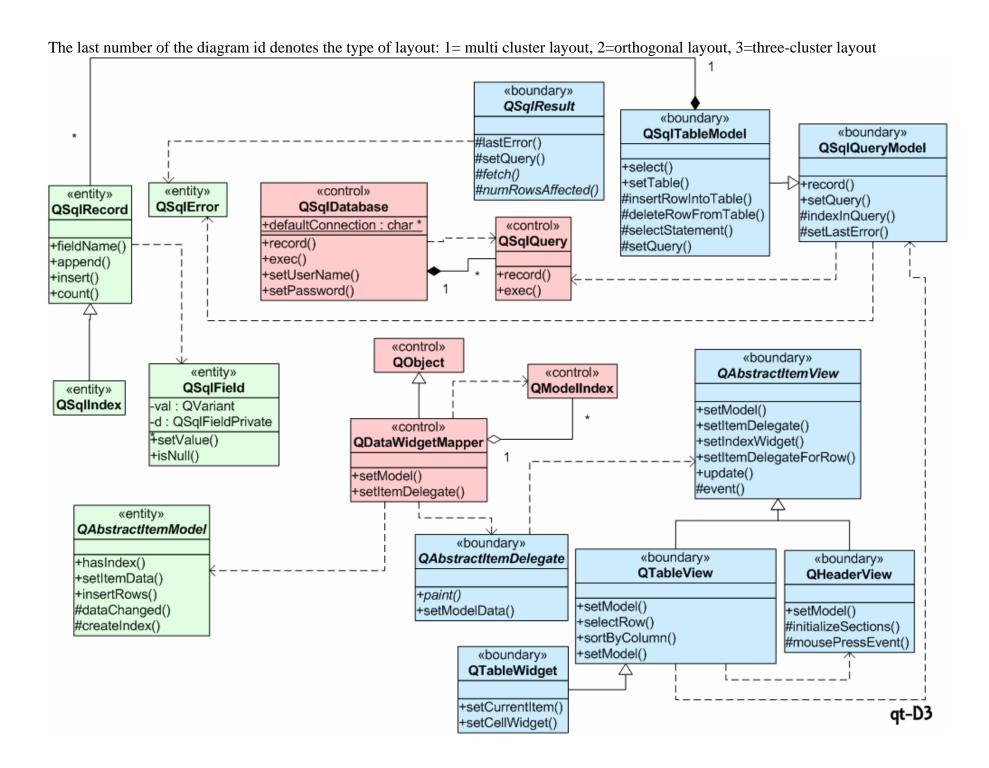
For each Diagram triplet, enter the diagram ID with the best and worst aesthetics. For your convenience, each triplet is shown on one page in thumbnail form and is also available in full size.

Diagrams (IDs)	Diagram with the Best Aesthetics	Diagram with the Worst Aesthetics	Comments
wx-D1 wx-D2 wx-D3			
wx-E1 wx-E2 wx-E3			
wx-DV1 wx-DV2 wx-DV3			
wx-G1 wx-G2 wx-G3			
wx-I1 wx-I2 wx-I3			
wx-L1 wx-L2 wx-L3			
wx-MW1 wx-MW2 wx-MW3			
wx-W1 wx-W2 wx-W3			

Comments about this questionnaire:

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» «boundary» «entity» **QSalResult** QSqlQueryModel **QSalRecord** «boundary» QSqlTableModel #lastError() +record() +fieldName() «entity» #setQuery() +setQuery() +append() **QSqlError** +select() #indexInQuery() #fetch() +insert() +setTable() #numRowsAffected() #setLastError() +count() #insertRowIntoTable() #deleteRowFromTable() #selectStatement() «entity» «control» #setQuery() **QSqlIndex** «control» **QSqIDatabase QSqlQuery** +defaultConnection : char * «entity» +record() **QSqlField** +record() +exec() -val : QVariant +exec() +setUserName() -d : QSqlFieldPrivate +setPassword() *setValue() «boundary» +isNull() **QAbstractItemView** +setModel() +setItemDelegate() «control» +setIndexWidget() **QObject** +setItemDelegateForRow() +update() #event() «control» «control» 4 **QModelIndex** QDataWidgetMapper «boundary» «boundary» +setModel() **QTableView** QHeaderView +setItemDelegate() +setModel() +setModel() +selectRow() #initializeSections() «entity» «boundary» +sortByColumn() #mousePressEvent() **QAbstractItemModel QAbstractItemDelegate** +setModel() +hasIndex() +paint() «boundary» +setItemData() +setModelData() **QTableWidget** +insertRows() #dataChanged() #createIndex() +setCurrentItem() +setCellWidget() qt-D1 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout





The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «control» **QObject QEvent** #timerEvent() +accept() #childEvent() +ignore() +event() «boundary» «control» «control» «control» «control» «control» **QTimer** QEventLoop QAction **QActionEvent QPaintEvent** QMoveEvent -inter : int -act : QAction * #p : QPoint #m_rect : QRect -bef : QAction * #oldp : QPoint #m_region : QRegion +timerld() +setIconText(in text) +processEvents() #m erased : bool +setInterval() +setToolTip(in tip) +action(): QAction * +exec() +pos() +start() +setStatusTip(in statusTip) +before(): QAction ' +oldPos() +exit() +rect() +setMenu(in menu : QMenu*) +stop() +isRunning() +region() #timeout() +wakeUp() +isToggleAction(): bool +setErased() +quit() «boundary» QWidget «boundary» «control» «control» #mouseMoveEvent() QToolBarltem QKeyMapper QInputEvent +event() +action : QAction * #modState +show() +customWidget : bool +sendKeyEvent() +modifiers() +paintEvent() «control» «control» «control» «control» «control» QKeyEvent **QContextMenuEvent** QWheelEvent QTabletEvent QMouseEvent #k : int #p : QPoint #mouseState #mouseState #gp : QPoint +key() +pos() : QPoint +pos() : QPoint +delta(): int +text() +x(): int +x(): int +x(): int +pos(): QPoint +isAutoRepeat() +y(): int +y(): int +y(): int +ascii() +pos() : QPoint +reason(): Reason «entity» **QPoint** -xp:int -yp:int +manhattanLength(): int qt-E1 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» **QObject** «boundary» #timerEvent() «boundary» QWidget #childEvent() **QToolBarItem** +action : QAction * +event() #mouseMoveEvent() +customWidget : bool +event() +show() +paintEvent() «control» «boundary» «control» «control» QKeyMapper **QTimer** QEventLoop QAction -inter : int «control» +timerId() +setIconText(in text) +sendKeyEvent() +processEvents() **QEvent** +setInterval() +exec() +setToolTip(in tip) +start() +exit() +setStatusTip(in statusTip) +stop() +isRunning() +setMenu(in menu : QMenu*) +accept() #timeout() +wakeUp() +isToggleAction(): bool +ignore() +quit() «control» «control» «control» «control» QInputEvent **QActionEvent QMoveEvent QPaintEvent** #p : QPoint act : QAction * #modState #m rect : QRect -bef : QAction * #oldp : QPoint #m region : QRegion +modifiers() #m erased:bool +pos() +action(): QAction * +rect() +before() : QAction * +oldPos() +region() +setErased() «control» «control» «control» «control» «control» **QContextMenuEvent** QKeyEvent QWheelEvent QTabletEvent **QMouseEvent** #p : QPoint #k:int #mouseState #mouseState #gp : QPoint +pos(): QPoint +key() +delta(): int +pos(): QPoint +x(): int +pos(): QPoint +x(): int +x(): int +text() +isAutoRepeat() +y(): int +y() : int +y(): int +pos(): QPoint +ascii() +reason(): Reason «entity» **QPoint** xp:int -yp : int

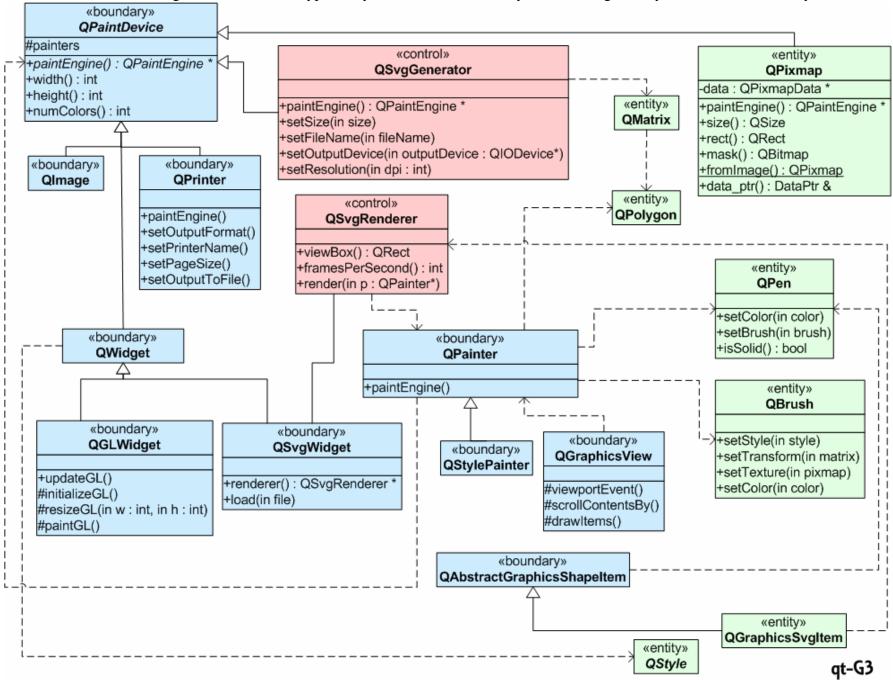
+manhattanLength(): int

qt-E2

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» **QEvent** «control» «boundary» +accept() **QObject** QWidget +ignore() #timerEvent() #mouseMoveEvent() #childEvent() +event() +event() +show() «control» «control» «control» +paintEvent() **QActionEvent** QMoveEvent **QPaintEvent** #p : QPoint #m rect : QRect -act : QAction * #m_region : QRegion #oldp : QPoint -bef : QAction * #m erased : bool +action() : QAction * +pos() «boundary» «control» «control» «control» +before(): QAction * +oldPos() +rect() **QTimer** QKeyMapper QEventLoop **QAction** +region() -inter : int +setErased() +timerld() +sendKeyEvent() +processEvents() +setIconText(in text) +setInterval() +exec() +setToolTip(in tip) +start() +exit() +setStatusTip(in statusTip) +stop() +isRunning() +setMenu(in menu : QMenu*) +wakeUp() +isToggleAction(): bool «control» #timeout() QInputEvent +quit() #modState «boundary» +modifiers() **QToolBarItem** +action : QAction * +customWidget : bool «control» «control» «control» «control» «control» QKeyEvent **QContextMenuEvent** QMouseEvent QWheelEvent QTabletEvent #p : QPoint #k:int #mouseState #mouseState #gp : QPoint +pos(): QPoint +key() +delta(): int +pos() : QPoint +text() +pos(): QPoint +x(): int +x(): int +x(): int +isAutoRepeat() +y(): int +y(): int +y(): int +ascii() +pos(): QPoint +reason(): Reason «entity» **QPoint** -xp:int -yp:int +manhattanLength(): int

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» **QPaintDevice** #painters +paintEngine() : QPaintEngine +width(): int «entity» +height(): int QBrush +numColors(): int +setStyle(in style) +setTransform(in matrix) «boundary» «entity» «boundary» +setTexture(in pixmap) **QImage QPrinter QPixmap** +setColor(in color) -data : QPixmapData * «entity» QPolygon +paintEngine(): QPaintEngine * +paintEngine() +setOutputFormat() +size(): QSize «boundary» +setPrinterName() +rect(): QRect **QGraphicsView** +mask(): QBitmap +setPageSize() «boundary» +fromImage(): QPixmap +setOutputToFile() **QPainter** #viewportEvent() +data_ptr() : DataPtr & #scrollContentsBy() +paintEngine() #drawltems() «entity» **QMatrix** «boundary» «control» QStylePainter QSvgGenerator «entity» «entity» QStyle **QPen** +paintEngine(): QPaintEngine * +setSize(in size) +setColor(in color) +setFileName(in fileName) «boundary» +setBrush(in brush) +setOutputDevice(in outputDevice : QIODevice*) **QWidget** +isSolid(): bool +setResolution(in dpi : int) «boundary» «boundary» QAbstractGraphicsShapeltem QGLWidget «control» «boundary» +paintEngine() : QPaintEngine * QSvgRenderer QSvgWidget «entity» +updateGL() QGraphicsSvgltem #initializeGL() +viewBox(): QRect +renderer(): QSvgRenderer * #resizeGL(in w : int, in h : int) +framesPerSecond(): int +load(in file) #paintGL() +render(in p : QPainter*) qt-G1 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» **QPaintDevice** #painters +paintEngine() : QPaintEngine +width(): int +height(): int +numColors(): int «entity» **QMatrix** «boundary» «entity» «control» «boundary» **QPixmap** Qlmage QSvgGenerator **QPrinter** -data : QPixmapData * «entity» **QPolygon** +paintEngine(): QPaintEngine * +paintEngine(): QPaintEngine * +paintEngine() +size(): QSize +setSize(in size) +setOutputFormat() +rect(): QRect +setFileName(in fileName) +setPrinterName() +mask(): QBitmap +setOutputDevice(in outputDevice : QIODevice*) +setPageSize() +fromImage(): QPixmap +setResolution(in dpi : int) +setOutputToFile() +data_ptr(): DataPtr & «control» «boundary» QSvgRenderer **QPainter** «boundary» +viewBox(): QRect **QGraphicsView** +paintEngine() +framesPerSecond(): int |-+render(in p : QPainter*) #viewportEvent() «entity» **QBrush** #scrollContentsBy() «boundary» #drawltems() «boundary» **QStylePainter** +setStyle(in style) **QWidget** +setTransform(in matrix) «entity» +setTexture(in pixmap) **QStyle** «boundary» +setColor(in color) QAbstractGraphicsShapeItem «boundary» «boundary» **QGLWidget QSvgWidget** «entity» **QPen** «entity» +paintEngine(): QPaintEngine * +renderer(): QSvgRenderer * QGraphicsSvgltem +setColor(in color) +updateGL() +load(in file) +setBrush(in brush) #initializeGL() +isSolid(): bool #resizeGL(in w : int, in h : int) qt-G2 #paintGL()

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout



The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» «control» **QObject QPaintDevice** QlmagelOHandler «boundary» Δ QlmagelOPlugin #painters +paintEngine() : QPaintEngine 1 +setDevice(in device : QIODevice*) «control» «control» +width(): int +setFormat(in format) +capabilities() **QFtp** QHttp +canRead(): bool +height(): int +create() +numColors(): int +read(in image : QImage*) : bool +keys() +write(in image) : bool «boundary» **QlmageWriter** «boundary» «boundary» **QIODevice QImage** +setFormat(in format) +setDevice(in device : QIODevice*) +isReadable(): bool +setFileName(in fileName) +copy() +isWritable(): bool +convertToFormat() +open(in mode : OpenMode) : bool +write(in image) : bool +width() +close() «boundary» +height() +seek(in pos) : bool QlmageReader +size() +readBlock(in data : char*, in maxlen) : long +load() +writeBlock(in data : const char*, in len) : long +setFormat(in format) +save() +setDevice(in device : QIODevice*) +setFileName(in fileName) «boundary» +read() : QImage «entity» «control» «boundary» **QAbstractSocket QBuffer QFile QProcess** «boundary» +setBuffer() **QClipboard** +setFileName() +connectToHost() +setReadChannel() +setData() +readBufferSize() +setEnvironment() +rename() +open() +copy() +execute() #readData() +setText() +close() +open() +terminate() +setImage() +seek() +close() +kill() +setPixmap() «boundary» #readData() +size() #readData() **QTcpSocket** +pos() «control» +seek() «entity» QDragManager #readData() #readData() **QPixmap** +object : QDrag * #readLineData() -data : QPixmapData * +eventLoop : QEventLoop * +size(): QSize +dropData : QDropData +rect(): QRect -pm_cursor : QPixmap * +mask(): QBitmap -currentDropTarget : QWidget * +fromImage(): QPixmap +source() qt-I1 +data ptr(): DataPtr & +dragCursor()

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» «control» **QObject QPaintDevice QimagelOHandler** #painters +paintEngine() : QPaintEngine * +setDevice(in device : QIODevice*) «boundary» +width(): int +setFormat(in format) QlmagelOPlugin «control» «control» +height(): int +canRead() : bool QHttp **QFtp** +numColors(): int +read(in image : QImage*) : bool +capabilities() +write(in image): bool +create() +keys() «boundary» **QImage** «boundary» «boundary» **QlmageReader QIODevice** +copy() «boundary» +convertToFormat() QlmageWriter +setFormat(in format) +isReadable(): bool +width() +setDevice(in device : QIODevice*) +height() +isWritable(): bool +setFileName(in fileName) +size() +open(in mode : OpenMode) : bool +setFormat(in format) +read(): QImage +load() +close() +setDevice(in device : QIODevice*) +seek(in pos): bool +save() +setFileName(in fileName) +readBlock(in data : char*, in maxlen) : long +write(in image) : bool +writeBlock(in data : const char*, in len) : long «boundary» «entity» **QClipboard QPixmap** -data : QPixmapData * +setText() +size(): QSize +setImage() +rect() : QRect +mask(): QBitmap +setPixmap() «entity» «boundary» «control» «boundary» +fromImage(): QPixmap QBuffer **QAbstractSocket QFile QProcess** +data_ptr() : DataPtr & +setBuffer() +setFileName() +setReadChannel() +connectToHost() «control» +setData() +readBufferSize() +setEnvironment() +rename() QDragManager +open() +execute() #readData() +copy() +object : QDrag * +close() +terminate() +open() +eventLoop : QEventLoop * +seek() +kill() +close() +dropData : QDropData * #readData() #readData() +size() -pm cursor : QPixmap * +pos()

«boundary»

QTcpSocket

#readData()

+seek()

#readData()

#readLineData()

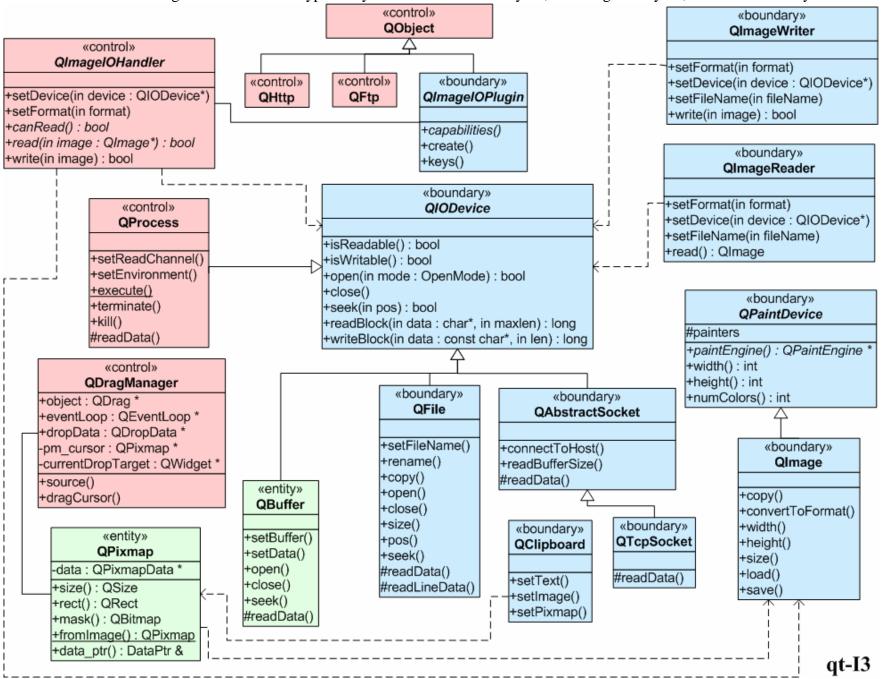
-currentDropTarget : QWidget *

qt-I2

+source()

+dragCursor()

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout



The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» «control» QLayoutItem QObject #align +sizeHint() : QSize +isWidgetType(): bool «boundary» «boundary» +minimumSize(): QSize +moveToThread(in thread : QThread*) **QWidget** QLayout +maximumSize() : QSize +setParent(in parent : QObject*) +widget() : QWidget * +event(in e : QEvent*) : bool +pos(): QPoint +setAlignment(in a) +mainWidget() : QWidget * +repaint() +addWidget(in w : QWidget*) #widgetEvent(in e : QEvent*) +event() «entity» +show() **QSize** +setAlignment() +setLayout(in layout : QLayout) +setSpacing(in spacing : int) «boundary» «boundary» «boundary» «boundary» QMainWindowLayout QBoxLayout QStackedLayout QGridLayout pluggingWidget : QWidget * +addWidget() +addSpacing() +setSpacing() +setStatusBar() +addWidget() +insertWidget() +columnCount() -setCentralWidget() +findWidget() +currentWidget() +rowCount() +addToolBar() #insertItem() +setCurrentWidget() +setColSpacing() restoreState() +setRowSpacing() «boundary» «boundary» **QFrame** QAbstractButton «boundary» «boundary» QVBoxLayout QHBoxLayout +setFrameStyle() «boundary» +setLineWidth() «boundary» Q3Wizard #drawFrame() **QPushButton** +setDefault() «boundary» «boundary» «boundary» **QAbstractScrollArea** QLabel QHBox +setAlignment(in align) #paintEvent() +setSpacing() +setStretchFactor() setText(in text) «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» **QVBox QTextEdit** QLineEdit **QMenuBar** QMenu **QGroupBox** QDialogButtonBox +setDocument() +displayText() +addAction() +addAction() +setTitle(in title) +setOrientation() +setDocumentTitle() +setMaxLength() +addMenu() +popup() +addButton() +setAlignment() +addSeparator() +exec() +setStandardButtons() #enterEvent() +isUndoAvailable() #paintEvent() +setCenterButtons() +insertItem()

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» «boundary» **QObject** QLayout QLayoutItem #align +isWidgetType(): bool +mainWidget(): QWidget * +sizeHint() : QSize +moveToThread(in thread : QThread*) +addWidget(in w : QWidget*) +minimumSize(): QSize +setParent(in parent : QObject*) #widgetEvent(in e : QEvent*) +maximumSize(): QSize +event(in e : QEvent*) : bool +setAlignment() «entity» +widget() : QWidget * +setSpacing(in spacing : int) **QSize** +setAlignment(in a) «boundary» «boundary» «boundary» «boundary» «boundary» **QFrame** QMainWindowLayout QBoxLayout QStackedLayout QGridLayout +pluggingWidget : QWidget * +setFrameStyle() +addSpacing() +addWidget() +setSpacing() +setStatusBar() +setLineWidth() +addWidget() +insertWidget() +columnCount() +setCentralWidget() #drawFrame() +findWidget() +currentWidget() +rowCount() +addToolBar() Д +setColSpacing() #insertItem() +setCurrentWidget() +restoreState() +setRowSpacing() «boundary» «boundary» QHBox QLabel «boundary» QWidget «boundary» «boundarv» «boundary» +setSpacing() +setAlignment(in align) QVBoxLayout Q3Wizard QHBoxLayout +setStretchFactor() +setText(in text) +pos(): QPoint +repaint() +event() «boundary» +show() **QVBox** +setLayout(in layout : QLayout) «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» **QAbstractButton** QLineEdit QMenuBar QMenu QGroupBox QDialogButtonBox **QAbstractScrollArea** +displayText() +addAction() +addAction() +setTitle(in title) +setOrientation() #paintEvent() +setMaxLength() +addMenu() +popup() +addButton() «boundary» +setAlignment() +addSeparator() +setStandardButtons() +exec() **QPushButton** +isUndoAvailable() #paintEvent() #enterEvent() +setCenterButtons() «boundary»

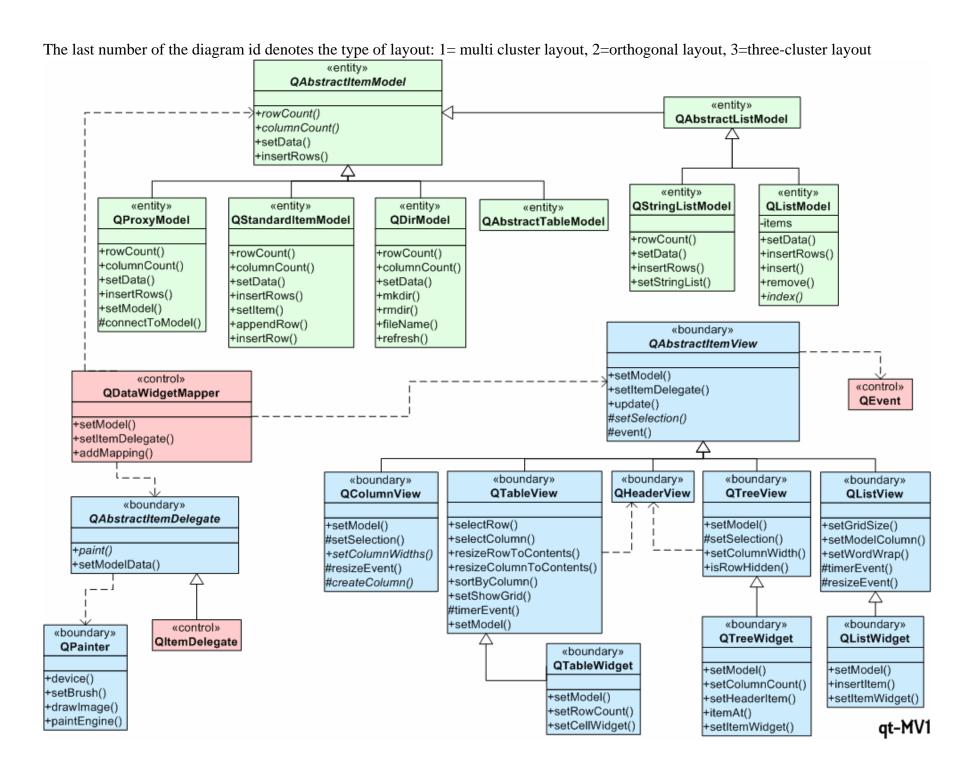
+insertItem()

+setDefault()

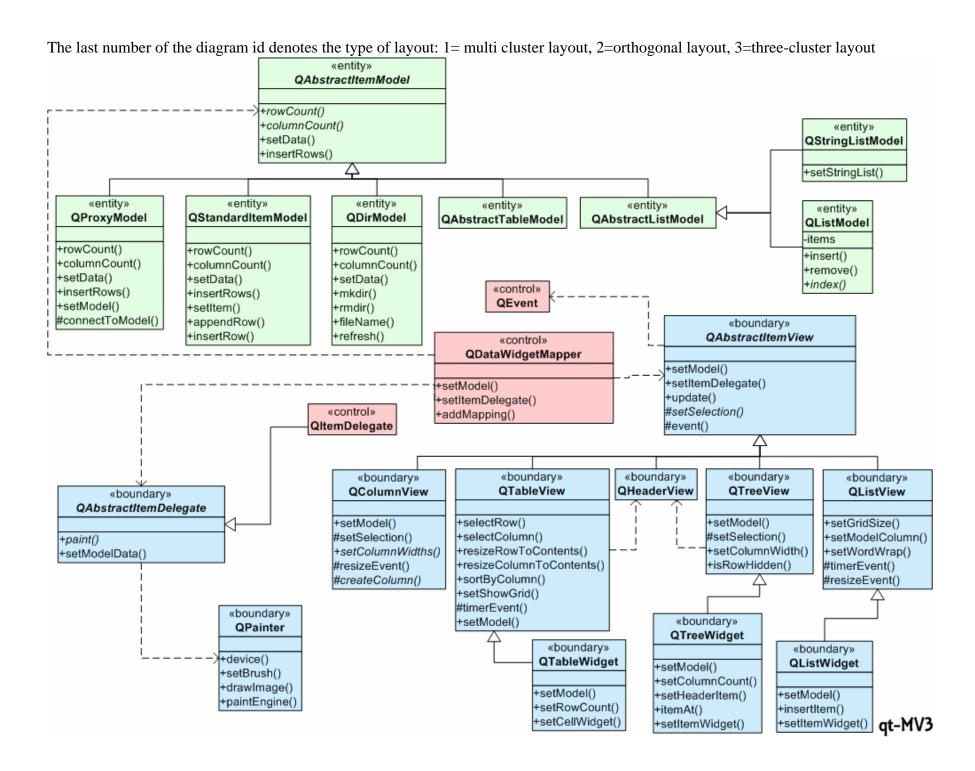
QTextEdit

qt-L2

+setDocument() +setDocumentTitle() The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» «control» QLayoutItem QObject #align +sizeHint() : QSize «boundary» +isWidgetType(): bool +minimumSize(): QSize QLayout +moveToThread(in thread : QThread*) +maximumSize(): QSize +setParent(in parent : QObject*) +widget() : QWidget * +event(in e : QEvent*) : bool +mainWidget() : QWidget * +setAlignment(in a) +addWidget(in w : QWidget*) #widgetEvent(in e : QEvent*) +setAlignment() «entity» +setSpacing(in spacing : int) **QSize** «boundary» QWidget «boundary» «boundary» «boundary» «boundary» +pos(): QPoint QMainWindowLayout QBoxLayout QStackedLayout QGridLayout +repaint() +pluggingWidget : QWidget * +event() +addSpacing() +addWidget() +setSpacing() +setStatusBar() +show() «boundary» +addWidget() +insertWidget() +columnCount() +setCentralWidget() +setLayout(in layout : QLayout) Q3Wizard +findWidget() +currentWidget() +rowCount() +addToolBar() #insertItem() +setCurrentWidget() +setColSpacing() +restoreState() +setRowSpacing() «boundary» «boundary» «boundary» QAbstractButton QVBoxLayout QHBoxLayout «boundary» «boundary» **QFrame** QPushButton +setFrameStyle() +setDefault() +setLineWidth() #drawFrame() «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» QLineEdit **QMenuBar** QMenu QDialogButtonBox **QGroupBox** QLabel QHBox QAbstractScrollArea +displayText() +addAction() +addAction() +setTitle(in title) +setOrientation() +setAlignment(in align) +setSpacing() #paintEvent() +setMaxLength() +addMenu() +popup() +addButton() +setText(in text) +setStretchFactor() +setAlignment() +addSeparator() 4 +exec() +setStandardButtons() +isUndoAvailable() #paintEvent() #enterEvent() +setCenterButtons() «boundary» +insertItem() **QTextEdit** «boundary» +setDocument() qt-L3 **QVBox** +setDocumentTitle()



The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «entity» **QAbstractItemModel** «control» QDataWidgetMapper +rowCount() +columnCount() +setModel() +setData() +setItemDelegate() +insertRows() +addMapping() «entity» «entity» «entity» «entity» «entity» QStandardItemModel QAbstractListModel QAbstractTableModel «boundary» QProxyModel **QDirModel** QAbstractItemDelegate Δ +rowCount() +rowCount() +rowCount() +paint() +columnCount() +columnCount() «entity» «entity» +columnCount() +setModelData() +setData() QStringListModel QListModel +setData() +setData() +insertRows() +insertRows() +mkdir() -items +setModel() +setItem() +rmdir() +rowCount() +setData() #connectToModel() +fileName() +appendRow() «boundary» +setData() +insertRows() +insertRow() +refresh() **QPainter** +insert() +insertRows() «control» +setStringList() +remove() **QItemDelegate** «boundary» +index() +device() **QAbstractItemView** +setBrush() +drawlmage() +setModel() +paintEngine() +setItemDelegate() «control» +update() **QEvent** #setSelection() #event() «boundary» «boundary» «boundary» «boundary» «boundary» **QColumnView** QHeaderView **QListView QTableView QTreeView** +setModel() +setGridSize() +selectRow() +setModel() #setSelection() +setModelColumn() +selectColumn() #setSelection() +setColumnWidths() +setWordWrap() +resizeRowToContents() +setColumnWidth() «boundary» **QTreeWidget** #resizeEvent() #timerEvent() +resizeColumnToContents() +isRowHidden() #createColumn() #resizeEvent() +sortByColumn() «boundary» «boundary» +setShowGrid() +setModel() QListWidget **QTableWidget** #timerEvent() +setColumnCount() +setModel() +setHeaderItem() +setModel() +setModel() +itemAt() +setRowCount() +insertItem() +setItemWidget() +setItemWidget() +setCellWidget() qt-MV2



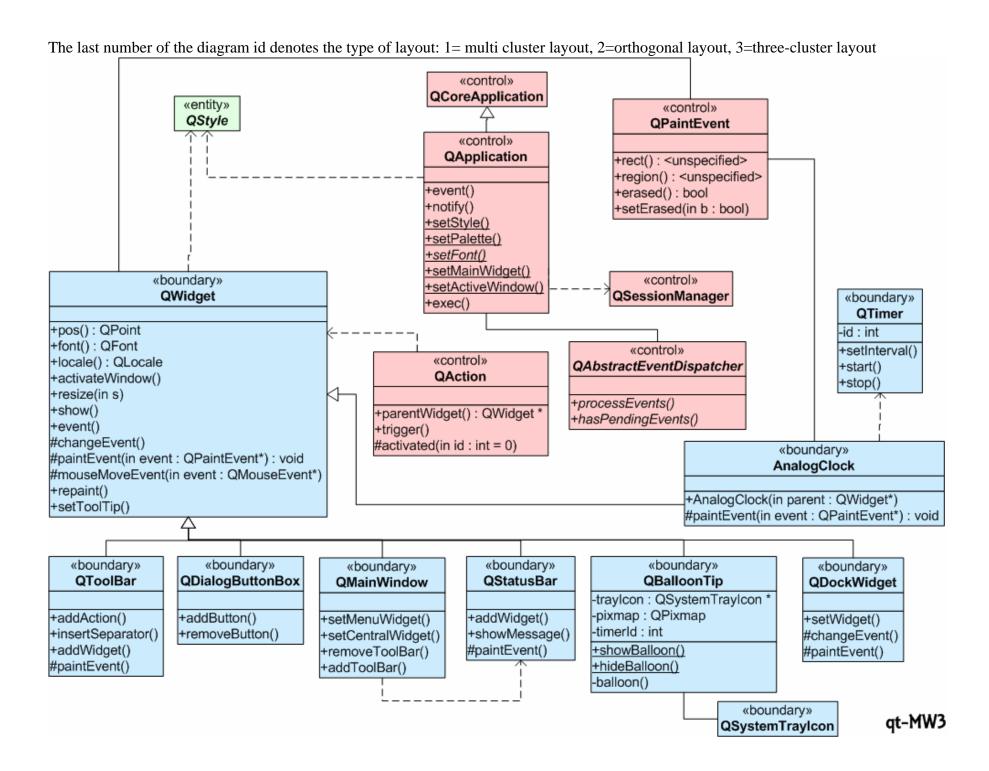
The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «entity» «control» **QAction** QStyle QSessionManager +parentWidget(): QWidget * «control» +trigger() **QCoreApplication** «boundary» #activated(in id : int = 0) QWidget «control» +pos(): QPoint **QApplication** +font(): QFont +locale(): QLocale +event() «control» +activateWindow() +notify() **QPaintEvent** +resize(in s) +setStyle() «boundary» +show() +setPalette() +rect(): <unspecified> **QTimer** +event() +setFont() +region(): <unspecified> -id : int #changeEvent() +setMainWidget() +erased(): bool +setInterval() #paintEvent(in event : QPaintEvent*) : void +setActiveWindow() +setErased(in b : bool) +start() #mouseMoveEvent(in event : QMouseEvent*) +exec() +stop() +repaint() +setToolTip() «control» «boundary» QAbstractEventDispatcher AnalogClock +processEvents() +AnalogClock(in parent : QWidget*) +hasPendingEvents() #paintEvent(in event : QPaintEvent*) : void «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» QDockWidget **QDialogButtonBox QMainWindow QToolBar QStatusBar** QBalloonTip -traylcon : QSystemTraylcon * -pixmap : QPixmap +setWidget() +addAction() +addButton() +setMenuWidget() +addWidget() -timerId: int #changeEvent() +insertSeparator() +removeButton() +setCentralWidget() +showMessage() #paintEvent() #paintEvent() +addWidget() +removeToolBar() +showBalloon() #paintEvent() +addToolBar() +hideBalloon()

-balloon()

«boundary»

QSystemTraylcon

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» «control» QCoreApplication QWidget **QAction** +pos(): QPoint +parentWidget(): QWidget * «control» +font(): QFont **QSessionManager** +trigger() «control» +locale(): QLocale #activated(in id : int = 0) **QApplication** +activateWindow() «entity» +resize(in s) **QStyle** +event() +show() +notify() +event() +setStyle() «control» #changeEvent() +setPalette() **QAbstractEventDispatcher** #paintEvent(in event : QPaintEvent*) : void +setFont() #mouseMoveEvent(in event : QMouseEvent*) +setMainWidget() +processEvents() +repaint() +setActiveWindow() +hasPendingEvents() +setToolTip() +exec() «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» QDialogButtonBox **QToolBar** QMainWindow **QStatusBar** QBalloonTip **QDockWidget** -traylcon : QSystemTraylcon * +addAction() +addButton() +setMenuWidget() +addWidget() +setWidget() -pixmap : QPixmap +insertSeparator() +setCentralWidget() +removeButton() +showMessage() -timerId : int #changeEvent() +addWidget() +removeToolBar() #paintEvent() #paintEvent() +showBalloon() #paintEvent() +addToolBar() +hideBalloon() -balloon() «boundary» QSystemTraylcon «control» «boundary» **QPaintEvent** «boundary» AnalogClock **QTimer** +rect(): <unspecified> -id : int +AnalogClock(in parent : QWidget*) +region(): <unspecified> +setInterval() #paintEvent(in event : QPaintEvent*) : void +erased(): bool +start() +setErased(in b : bool) +stop()

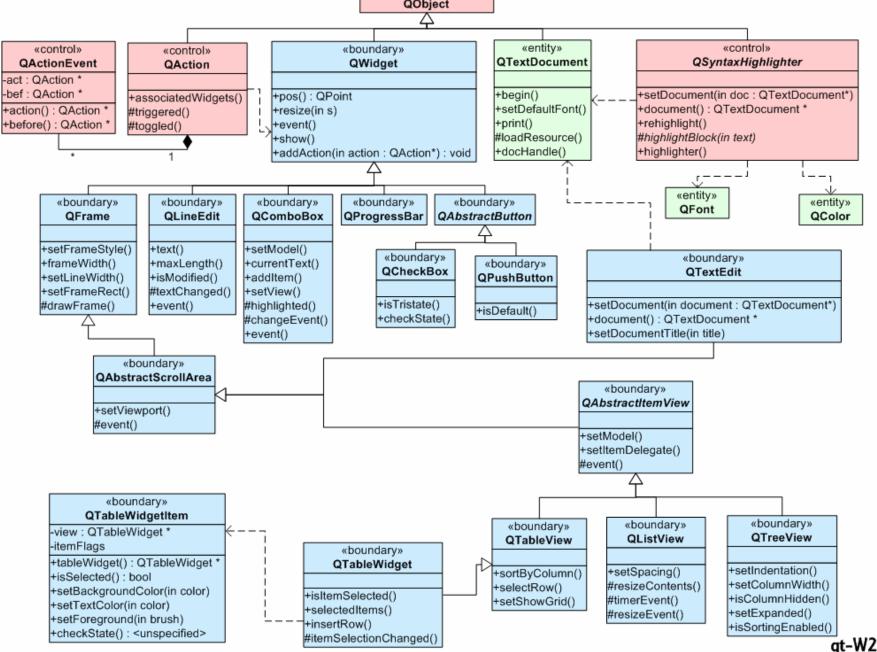


The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «entity» QObject QTextDocument +begin() «boundary» «control» «control» setDefaultFont() QWidget **QAction QActionEvent** +print() «control» #loadResource() act : QAction * QSyntaxHighlighter -bef : QAction * +docHandle() +pos() : QPoint +associatedWidgets() #triggered() +resize(in s) +action() : QAction * +setDocument(in doc : QTextDocument*) +before(): QAction 1 +event() #toggled() +document() : QTextDocument * +show() +rehighlight() +addAction(in action : QAction*) : void «entity» #highlightBlock(in text) **QColor** +highlighter() «entity» «boundary» «boundary» «boundary» «boundary» «boundary» QFont **QFrame** QLineEdit **QComboBox** QProgressBar QAbstractButton «boundary» +setFrameStyle() +text() +setModel() **QTextEdit** «boundary» «boundary» +frameWidth() +maxLength() +currentText() **QCheckBox QPushButton** +setLineWidth() +isModified() +addItem() +setDocument(in document : QTextDocument*) +setFrameRect() #textChanged() +setView() +document() : QTextDocument * +isTristate() +isDefault() #drawFrame() event() #highlighted() +setDocumentTitle(in title) +checkState() #changeEvent() +event() «boundary» QAbstractScrollArea 4 6 1 +setViewport() «boundary» #event() **QAbstractItemView** +setModel() +setItemDelegate() #event() «boundary» QTableWidgetItem -view : QTableWidget * «boundary» «boundary» «boundary» -itemFlags «boundary» **QTreeView** QListView **QTableView** +tableWidget() : QTableWidget * QTableWidget +isSelected(): bool +setIndentation() +setSpacing() +sortByColumn() +setBackgroundColor(in color) +isItemSelected() +setColumnWidth() #resizeContents() +selectRow() +setTextColor(in color) +selectedItems() #timerEvent() +setShowGrid() +isColumnHidden() +setForeground(in brush) +insertRow() +setExpanded() #resizeEvent() +checkState(): <unspecified>

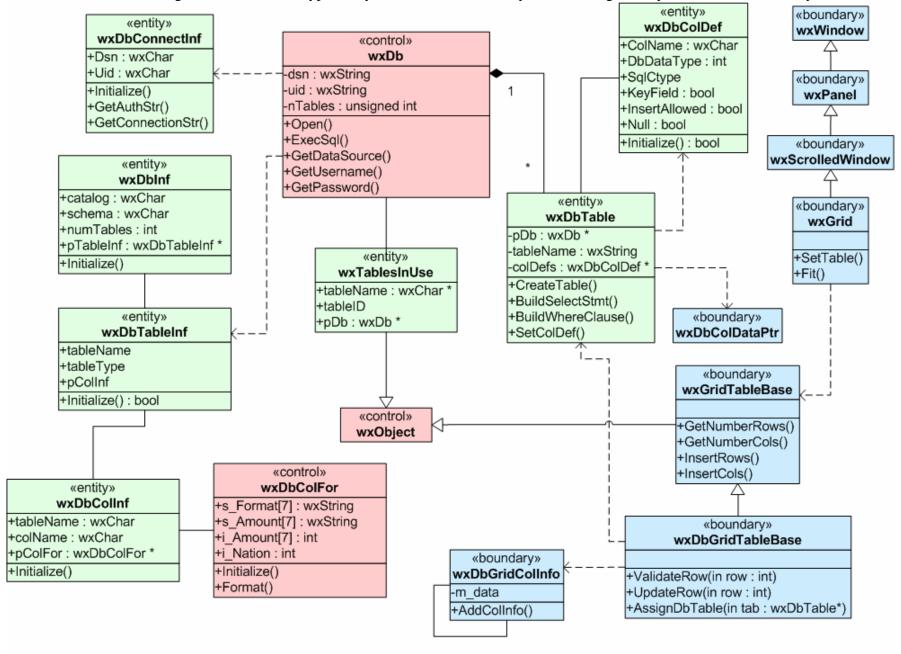
+isSortingEnabled()

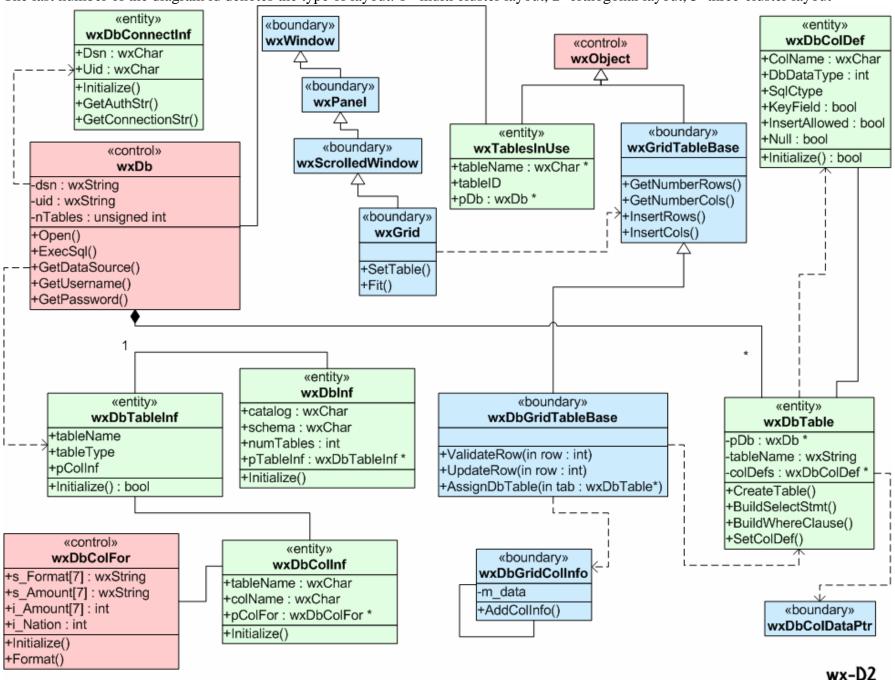
#itemSelectionChanged()

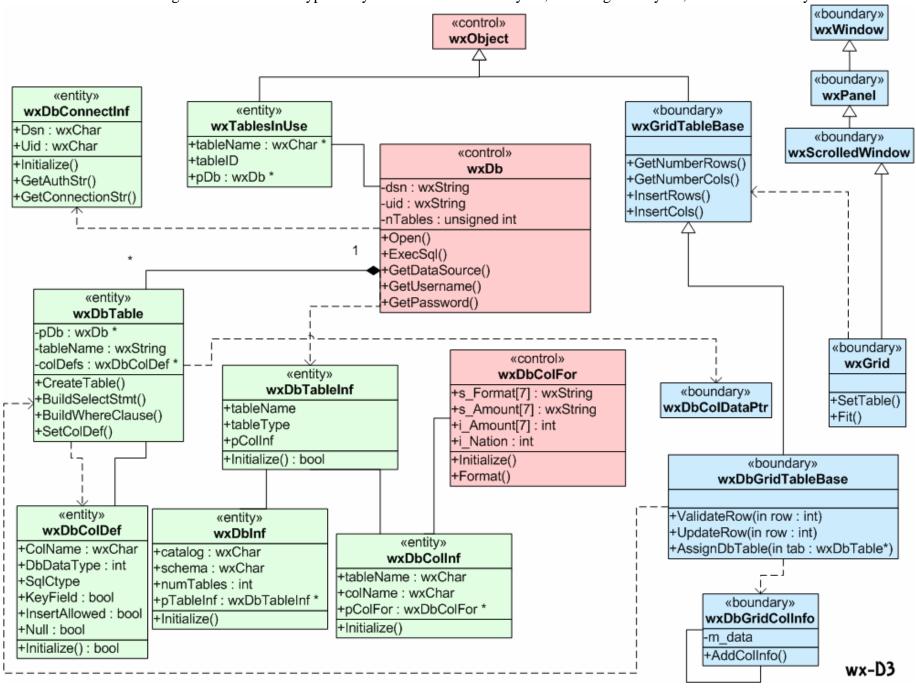
The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» QObject Д «boundary» «entity» «control» «control» «control» QTextDocument QSyntaxHighlighter QActionEvent **QAction** QWidget -act : QAction * -bef : QAction * +pos(): QPoint +begin() +setDocument(in doc : QTextDocument*) +associatedWidgets() +setDefaultFont() +document() : QTextDocument * +action() : QAction ' +resize(in s) #triggered() +print() +rehighlight() +before() : QAction * +event() #toggled() #loadResource() #highlightBlock(in text) +show() +docHandle() +highlighter() +addAction(in action : QAction*) : void «entity» «boundary» «boundary» «boundary» «boundary» «boundary» «entity» QFont QAbstractButton **QFrame** QLineEdit **QComboBox** QProgressBar **QColor** +setFrameStyle() +text() +setModel() «boundary» «boundary» +frameWidth() +maxLength() +currentText() «boundary» **QCheckBox QTextEdit** +setLineWidth() +isModified() +addItem() **QPushButton** +setFrameRect() #textChanged() +setView() +isTristate() +setDocument(in document : QTextDocument*) #drawFrame() +event() #highlighted() +isDefault() #changeEvent() +checkState() +document() : QTextDocument * +event() +setDocumentTitle(in title) «boundary» **QAbstractScrollArea** «boundary» **QAbstractItemView** +setViewport() #event() +setModel() +setItemDelegate() #event() «boundary»



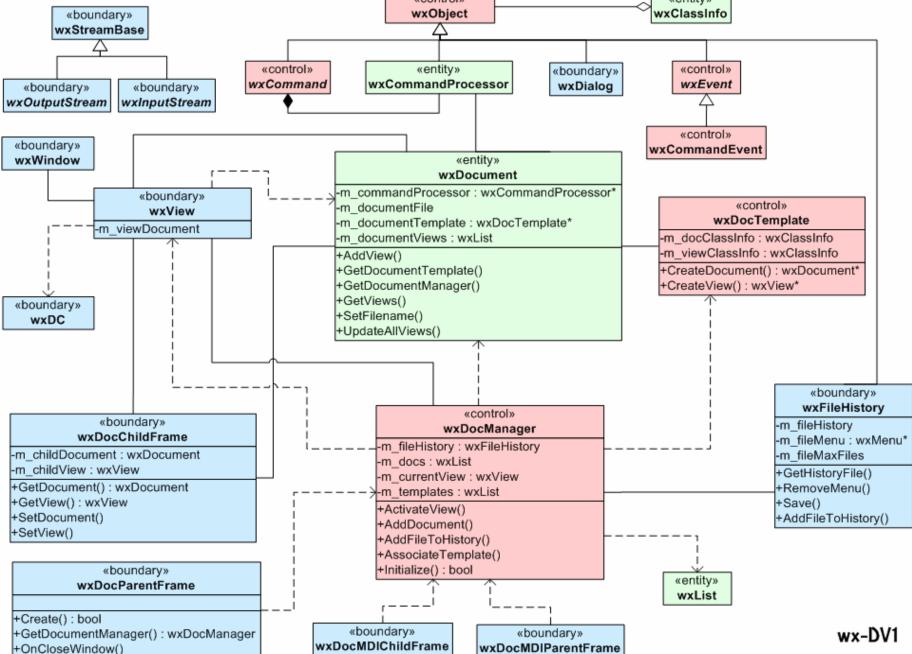
The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» **QObject** «control» «control» «control» **QAction** QActionEvent QSyntaxHighlighter «boundary» -act : QAction * QWidget -bef : QAction * +setDocument(in doc : QTextDocument*) +associatedWidgets() +document(): QTextDocument * #triggered() +action() : QAction * +pos(): QPoint #toggled() +before(): QAction * +rehighlight() +resize(in s) #highlightBlock(in text) +event() +highlighter() +show() «entity» +addAction(in action : QAction*) : void **QTextDocument** +begin() «boundary» «boundary» «boundary» «boundary» «boundary» +setDefaultFont() **QFrame** QLineEdit **QComboBox** QProgressBar QAbstractButton +print() #loadResource() «entity» +docHandle() +setFrameStyle() +text() +setModel() «boundary» **QColor** «boundary» +frameWidth() +currentText() +maxLength() **QCheckBox QPushButton** +setLineWidth() +isModified() +addItem() #textChanged() +setFrameRect() +setView() «entity» +isTristate() #drawFrame() +event() #highlighted() +isDefault() QFont +checkState() #changeEvent() +event() «boundary» QAbstractScrollArea 4 6 1 «boundary» «boundary» QAbstractItemView +setViewport() **QTextEdit** #event() +setModel() +setDocument(in document : QTextDocument*) +setItemDelegate() +document() : QTextDocument * #event() +setDocumentTitle(in title) «boundary» QTableWidgetItem «boundary» «boundary» «boundary» «boundary» **QTreeView** QListView **QTableView** -view : QTableWidget * **QTableWidget** -itemFlags +tableWidget() : QTableWidget * +setSpacing() +sortByColumn() +setIndentation() +isItemSelected() +isSelected(): bool #resizeContents() +setColumnWidth() +selectRow() +selectedItems() +setBackgroundColor(in color) +isColumnHidden() #timerEvent() +setShowGrid() +insertRow() +setTextColor(in color) +setExpanded() #resizeEvent() #itemSelectionChanged() +setForeground(in brush) +isSortingEnabled() at-W3 +checkState(): <unspecified>

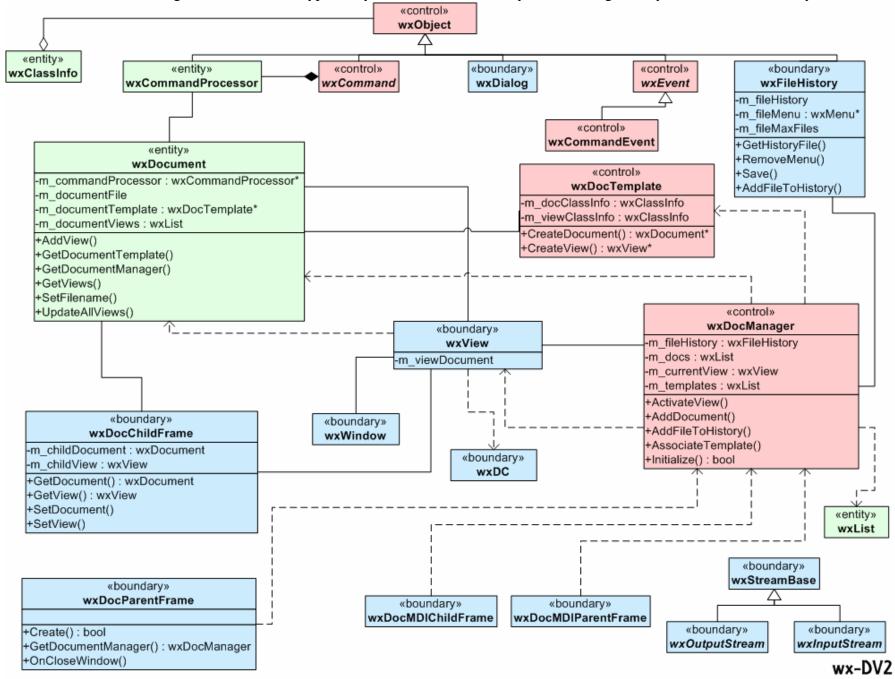




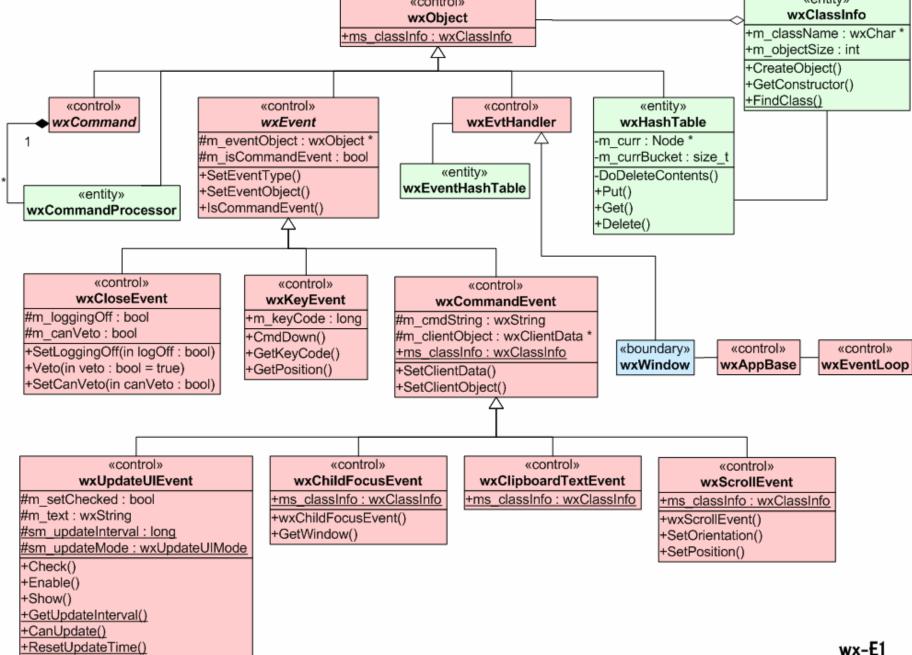


The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «entity» «boundary» wxObject wxClassInfo wxStreamBase «control» «entity» «control» «boundary» wxCommandProcessor wxCommand wxEvent «boundary» «boundary» wxDialog wxOutputStream wxInputStream «control» «boundary» wxCommandEvent «entity» wxWindow wxDocument -m_commandProcessor : wxCommandProcessor* «boundary» «control» -m documentFile wxView wxDocTemplate -m_documentTemplate : wxDocTemplate* m viewDocument -m documentViews : wxList -m docClassInfo : wxClassInfo -m viewClassInfo : wxClassInfo +AddView() +GetDocumentTemplate() +CreateDocument(): wxDocument* +GetDocumentManager() +CreateView(): wxView* +GetViews() «boundary» +SetFilename() wxDC +UpdateAllViews() «boundary» wxFileHistory «control» «boundary» -m fileHistory wxDocManager wxDocChildFrame -m fileMenu: wxMenu* -m fileHistory : wxFileHistory -m childDocument : wxDocument -m_fileMaxFiles -m_docs: wxList -m childView : wxView +GetHistoryFile() -m_currentView:wxView +GetDocument(): wxDocument +RemoveMenu() -m_templates : wxList +GetView(): wxView +Save() +ActivateView() +SetDocument() +AddFileToHistory() +AddDocument() +SetView() +AddFileToHistory() +AssociateTemplate() «boundary» +Initialize(): bool «entity» wxDocParentFrame





The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» «control» «entity» wxFileHistory wxClassInfo wxObject -m fileHistory Δ -m fileMenu: wxMenu* -m fileMaxFiles «boundary» «control» «control» «entity» +GetHistoryFile() wxEvent wxDialog wxCommand wxCommandProcessor +RemoveMenu() +Save() +AddFileToHistory() «control» wxCommandEvent «boundary» wxWindow «entity» «boundary» «control» wxDocument wxView wxDocManager -m commandProcessor : wxCommandProcessor* m fileHistory: wxFileHistory m viewDocument -m documentFile -m_docs : wxList -m_documentTemplate : wxDocTemplate* -m currentView : wxView «boundary» -m documentViews : wxList -m templates : wxList wxDC +AddView() +ActivateView() +GetDocumentTemplate() «boundary» +AddDocument() +GetDocumentManager() wxDocChildFrame +AddFileToHistory() +GetViews() +AssociateTemplate() -m childDocument : wxDocument +SetFilename() +Initialize(): bool -m childView : wxView +UpdateAllViews() +GetDocument(): wxDocument +GetView(): wxView +SetDocument() «entity» +SetView() wxList «boundary» «control» wxDocParentFrame wxDocTemplate -m_docClassInfo : wxClassInfo +Create(): bool «boundary» -m_viewClassInfo : wxClassInfo +GetDocumentManager(): wxDocManager wxDocMDIChildFrame +CreateDocument(): wxDocument* +OnCloseWindow() +CreateView(): wxView* «boundary» «boundary» wxDocMDIParentFrame wxStreamBase «boundary» «boundary» wxOutputStream wxInputStream wx-DV3 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «entity» «control» wxClassInfo wxObject +m className: wxChar * +ms classInfo : wxClassInfo +m objectSize: int +CreateObject() +GetConstructor() +FindClass() «control» «control» «control» «entity» wxCommand wxEvent wxEvtHandler wxHashTable #m eventObject : wxObject * -m curr : Node * 1 #m isCommandEvent : bool -m currBucket: size t «entity» +SetEventType() -DoDeleteContents() wxEventHashTable +SetEventObject() +Put() «entity» +IsCommandEvent() +Get() wxCommandProcessor +Delete() «control» «control» «control» wxCloseEvent wxKeyEvent wxCommandEvent #m loggingOff : bool +m keyCode : long #m cmdString: wxString #m canVeto: bool #m clientObject : wxClientData * +CmdDown() «boundary» «control» «control» +SetLoggingOff(in logOff : bool) +ms classInfo : wxClassInfo +GetKeyCode() wxWindow wxEventLoop wxAppBase +Veto(in veto : bool = true) +GetPosition() +SetClientData() +SetCanVeto(in canVeto : bool) +SetClientObject() «control» «control» «control» «control» wxUpdateUIEvent wxChildFocusEvent wxClipboardTextEvent wxScrollEvent #m setChecked : bool +ms classInfo : wxClassInfo +ms classInfo : wxClassInfo +ms classInfo : wxClassInfo #m text : wxString +wxChildFocusEvent() +wxScrollEvent() #sm_updateInterval:long +GetWindow() +SetOrientation() #sm_updateMode : wxUpdateUIMode +SetPosition()



The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» wxObject +ms classInfo : wxClassInfo «entity» «entity» «control» «control» «control» wxHashTable wxCommandProcessor wxCommand wxEvent wxEvtHandler #m eventObject : wxObject * -m curr : Node * #m isCommandEvent : bool -m currBucket:size t 1 -DoDeleteContents() +SetEventType() +Put() +SetEventObject() «entity» +Get() +IsCommandEvent() wxEventHashTable +Delete() «boundary» «control» «control» «control» «control» wxKeyEvent wxWindow wxCloseEvent wxCommandEvent wxEventLoop #m loggingOff : bool #m cmdString: wxString +m keyCode : long #m clientObject : wxClientData * #m canVeto: bool +CmdDown() +ms classInfo : wxClassInfo +SetLoggingOff(in logOff : bool) +GetKeyCode() «control» +Veto(in veto : bool = true) +SetClientData() +GetPosition() wxAppBase +SetCanVeto(in canVeto : bool) +SetClientObject() «control» «control» «control» «control» wxClipboardTextEvent wxUpdateUIEvent wxScrollEvent wxChildFocusEvent +ms classInfo : wxClassInfo #m setChecked : bool +ms classInfo : wxClassInfo +ms classInfo : wxClassInfo #m text : wxString +wxChildFocusEvent() +wxScrollEvent() #sm_updateInterval:long +GetWindow() +SetOrientation() «entity» #sm_updateMode : wxUpdateUIMode +SetPosition() wxClassInfo +Check() +m className : wxChar * +Enable() +m objectSize : int +Show() +GetUpdateInterval() +CreateObject() +GetConstructor() +CanUpdate() +ResetUpdateTime() +FindClass() wx-E2 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «control» «control» wxAppBase wxEventLoop wxObject +ms classInfo : wxClassInfo «entity» «control» «control» «control» «entity» wxCommand wxHashTable wxEvtHandler wxEvent wxCommandProcessor #m eventObject : wxObject * -m curr : Node * #m isCommandEvent : bool -m currBucket: size t +SetEventType() -DoDeleteContents() +SetEventObject() +Put() «boundary» +lsCommandEvent() +Get() wxWindow +Delete() «entity» wxEventHashTable «control» «entity» «control» «control» wxClassInfo wxCommandEvent wxCloseEvent wxKeyEvent +m className : wxChar ' #m cmdString: wxString #m loggingOff: bool +m keyCode : long #m clientObject : wxClientData * +m_objectSize : int #m canVeto: bool +CmdDown() +ms classInfo : wxClassInfo +SetLoggingOff(in logOff : bool) +CreateObject() +GetKeyCode() +GetConstructor() +Veto(in veto : bool = true) +SetClientData() +GetPosition() +SetClientObject() +FindClass() +SetCanVeto(in canVeto : bool) «control» «control» «control» «control» wxUpdateUIEvent wxChildFocusEvent wxClipboardTextEvent wxScrollEvent +ms classInfo : wxClassInfo #m setChecked : bool +ms_classInfo : wxClassInfo +ms classInfo : wxClassInfo #m text : wxString +wxChildFocusEvent() +wxScrollEvent() #sm_updateInterval:long +GetWindow() +SetOrientation() #sm_updateMode : wxUpdateUIMode +SetPosition() +Check() +Enable() +Show() +GetUpdateInterval() +CanUpdate() +ResetUpdateTime() wx-E3 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «boundary» «control» wxObject wxDCBase +ms classInfo : wxClassInfo «boundary» «control» «control» «entity» wxDC wxlOHandler wxGLContext wxEvtHandler #m canvas : wxWindow * #m glContext #m selectedBitmap : wxBitmap +ms classInfo : wxClassInfo +SetPen() «boundary» +SetCurrent() +SetBrush() «control» wxWindow +SetBackground() wxlOHandlerSVGIn #m_hWnd #GetDeviceSize() -m doc +Create() #LoadSvg() +Update() +Load() +CanLoad() «boundary» «boundary» «boundary» wxWindowDC wxScreenDC wxPrinterDC «boundary» wxGLCanvas #m_glContext : wxGLContext * #m palette : wxPalette «boundary» +Create() wxRegion +SetColour() «control» +ms classInfo : wxClassInfo +GetContext() wxRenderer +Clear() +GetPalette() +IsEmpty() +DrawBackground() #DoGetBox() +DrawBorder() +DrawFrame() +GetCheckBitmapSize() «entity» wxRect «entity» +x:int wxSize «entity» +y:int wxPoint +x : int «entity» «entity» +width : int +y:int +x:int wxBitmap wxPen +height: int +y:int +GetWidth() +ms_classInfo : wxClassInfo +ms classInfo : wxClassInfo +SetWidth() +GetHeight() +SetColour() +Create() +SetHeight() +SetStyle() +SaveFile() SetSize() +GetColour() +GetBitmapData() +GetStipple() wx-G1

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» wxObject +ms_classInfo : wxClassInfo «entity» «control» «boundary» «control» wxDCBase wxGLContext wxIOHandler wxEvtHandler #m glContext +ms classInfo : wxClassInfo «boundary» +SetCurrent() «control» wxDC «boundary» wxlOHandlerSVGIn wxWindow #m canvas : wxWindow * -m doc #m_selectedBitmap : wxBitmap #m hWnd #LoadSvg() +SetPen() +Create() +Load() +SetBrush() +Update() +CanLoad() +SetBackground() #GetDeviceSize() «boundary» «boundary» «boundary» «entity» wxPrinterDC wxWindowDC wxScreenDC «boundary» wxRect wxGLCanvas +x:int #m_glContext : wxGLContext * +y:int #m_palette : wxPalette +width: int +Create() «entity» «control» +height : int +SetColour() wxSize wxRenderer +SetWidth() +GetContext() +x:int +SetHeight() +GetPalette() +y:int +DrawBackground() +SetSize() +GetWidth() +DrawBorder() +GetHeight() +DrawFrame() +GetCheckBitmapSize() «entity» «boundary» «entity» wxPen wxRegion wxPoint +ms_classInfo : wxClassInfo +ms classInfo: wxClassInfo +x:int «entity» +SetColour() +Clear() +y:int wxBitmap +SetStyle() +IsEmpty() +ms classInfo : wxClassInfo +GetColour() #DoGetBox() +Create() +GetStipple() +SaveFile() +GetBitmapData() wx-G2 The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» wxObject +ms classInfo : wxClassInfo «entity» «boundary» «control» «control» wxGLContext wxDCBase wxEvtHandler wxIOHandler #m glContext +ms classInfo : wxClassInfo «boundary» +SetCurrent() «control» wxDC wxIOHandlerSVGIn «boundary» #m canvas : wxWindow * wxWindow -m doc #m selectedBitmap : wxBitmap #m_hWnd #LoadSvg() «entity» +SetPen() +Load() +Create() wxRect +SetBrush() +CanLoad() +Update() +x:int +SetBackground() +y:int #GetDeviceSize() +width : int «control» wxRenderer +height : int +SetWidth() «boundary» «boundary» +SetHeight() +DrawBackground() «boundary» wxPrinterDC +SetSize() +DrawBorder() wxWindowDC wxScreenDC +DrawFrame() +GetCheckBitmapSize() «boundary» wxGLCanvas «entity» «entity» #m_glContext : wxGLContext * wxSize wxPoint #m palette : wxPalette +x:int +x:int +Create() +y:int +y:int +SetColour() +GetWidth() +GetContext() «entity» +GetHeight() «entity» +GetPalette() wxBitmap wxPen +ms classInfo: wxClassInfo +ms classInfo: wxClassInfo «boundary» +Create() +SetColour() wxRegion +SaveFile() +SetStyle() +ms classInfo : wxClassInfo +GetBitmapData() +GetColour() +Clear() +GetStipple() +IsEmpty() #DoGetBox()

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» wxObject +ms classInfo : wxClassInfo «boundary» «control» wxlmageHandler wxDCBase #m pen:wxPen #m brush : wxBrush +LoadFile() #m backgroundBrush : wxBrush +SaveFile() «boundary» «entity» +GetName() wxlmage wxBitmap +ms classInfo : wxClassInfo +Create() «boundary» +Create() +LoadFile() «boundary» +LoadFile() wxDC +SaveFile() wxStreamBase +SaveFile() #m canvas : wxWindow * +Size() +GetBitmapData() #m selectedBitmap : wxBitmap +AddHandler() +SetPen() +RemoveHandler() «boundary» «boundary» +SetBrush() wxInputStream wxOutputStream +SetBackground() «boundary» «boundary» «boundary» «boundary» «boundary» wxPrinterDC wxMemoryDC wxScreenDC wxMirrorDC wxWindowDC -m_dc : wxMirrorDC & -m oldPen -m oldPen -m oldPen +GetPaperRect(): wxRect -m mirror : bool #DoGetSize() #DoGetSize() #InitDC() -m oldPen #Init() #DoGetSize() +SetBrush() +SetBackground() «boundary» #GetX() «boundary» wxBufferedDC #GetY() wxClientDC -m dc: wxDC * -m oldPen -m buffer : wxBitmap * #DoGetSize() -m style : int #InitDC() -m oldPen +Init() +SetStyle() «boundary» «boundary»

wxBufferedPaintDC

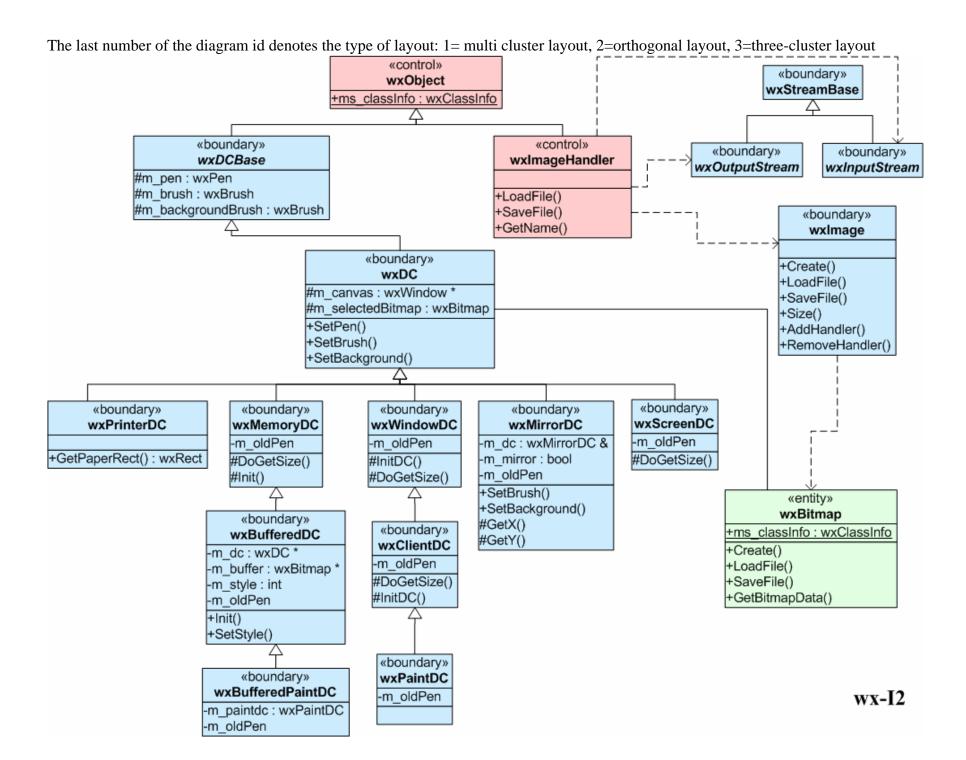
-m paintdc : wxPaintDC

-m oldPen

wxPaintDC

-m_oldPen

wx-I1



The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» wxObject +ms classInfo : wxClassInfo «boundary» «control» wxDCBase wxlmageHandler #m pen:wxPen #m brush : wxBrush +LoadFile() #m backgroundBrush : wxBrush +SaveFile() «boundary» +GetName() wxDC «boundary» #m canvas : wxWindow * wxStreamBase #m selectedBitmap : wxBitmap +SetPen() +SetBrush() «boundary» +SetBackground() «boundary» wxInputStream wxOutputStream «boundary» «boundary» «boundary» «boundary» «boundary» wxScreenDC wxWindowDC wxPrinterDC wxMirrorDC wxMemoryDC -m dc: wxMirrorDC & -m oldPen -m oldPen -m oldPen -m mirror : bool +GetPaperRect(): wxRect #DoGetSize() #DoGetSize() #InitDC() -m oldPen #Init() #DoGetSize() +SetBrush() +SetBackground() «boundary» #GetX() «boundary» wxBufferedDC #GetY() wxClientDC -m dc: wxDC * -m oldPen -m buffer : wxBitmap * «boundary» #DoGetSize() -m style : int wxlmage #InitDC() -m oldPen +Init() «entity» +Create() +SetStyle() +LoadFile() wxBitmap «boundary» +SaveFile() +ms classInfo : wxClassInfo «boundary» wxPaintDC

wxBufferedPaintDC

-m paintdc: wxPaintDC

-m oldPen

-m oldPen

wx-I3

+Size()

+AddHandler()

+RemoveHandler()

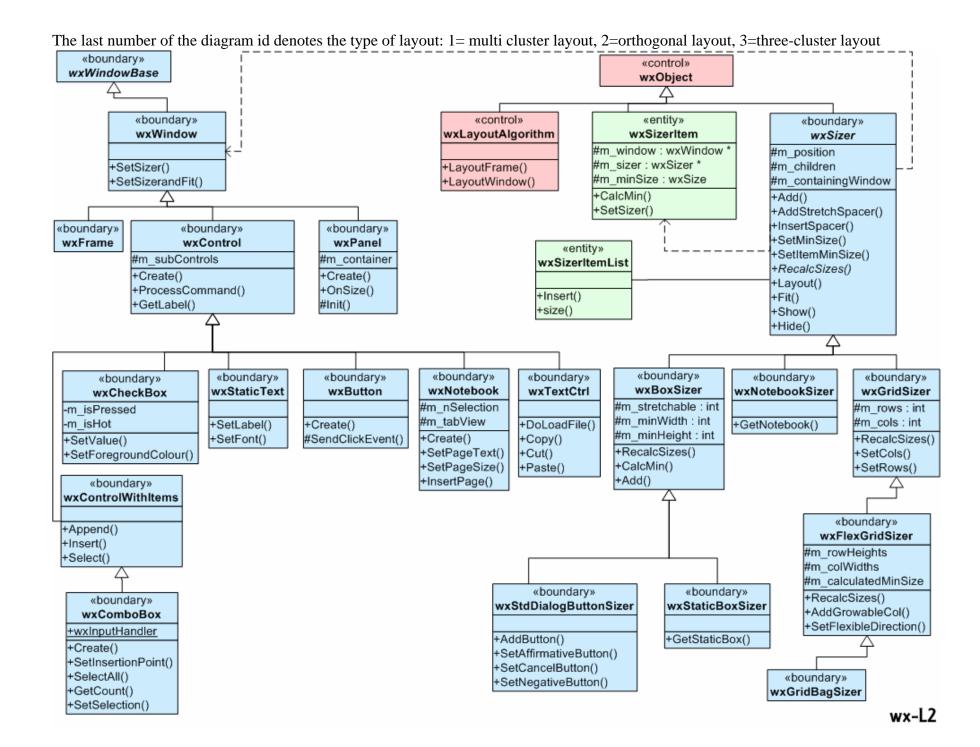
+Create()

+LoadFile()

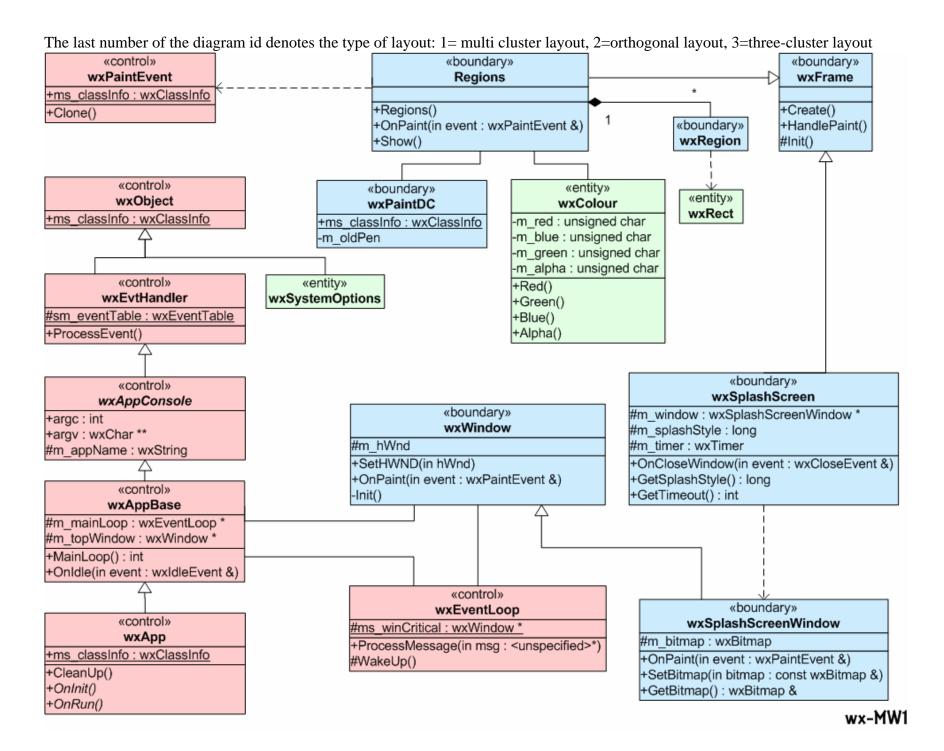
+SaveFile()

+GetBitmapData()

The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» wxObject wxWindowBase Δ «boundary» «boundary» «entity» «control» wxSizer wxSizerItem wxWindow wxLayoutAlgorithm #m window : wxWindow * #m position #m sizer : wxSizer * +SetSizer() #m children +LayoutFrame() #m_containingWindow #m minSize : wxSize +SetSizerandFit() +LayoutWindow() +CalcMin() +Add() +SetSizer() +AddStretchSpacer() +InsertSpacer() «boundary» «boundary» «boundary» «entity» +SetMinSize() wxFrame wxControl wxPanel wxSizerItemList +SetItemMinSize() #m subControls #m container +RecalcSizes() Create() +Create() +Layout() +Insert() +ProcessCommand() +OnSize() +Fit() +size() +GetLabel() #Init() +Show() +Hide() «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» wxGridSizer wxBoxSizer wxNotebookSizer wxNotebook wxStaticText wxButton wxTextCtrl wxCheckBox #m rows:int #m stretchable : int #m nSelection -m isPressed #m cols:int #m minWidth:int #m_tabView +Create() -m isHot +GetNotebook() +SetLabel() +DoLoadFile() #m minHeight:int +RecalcSizes() +Create() +SetFont() #SendClickEvent() +Copy() +SetValue() +RecalcSizes() +SetCols() +SetPageText() +Cut() +SetForegroundColour() +SetRows() +CalcMin() +SetPageSize() +Paste() +Add() +InsertPage() «boundary» wxControlWithItems «boundary» wxFlexGridSizer +Append() #m rowHeights «boundary» «boundary» +Insert() #m colWidths wxStdDialogButtonSizer wxStaticBoxSizer +Select() #m_calculatedMinSize +RecalcSizes() +AddButton() +GetStaticBox() +AddGrowableCol() «boundary» +SetAffirmativeButton() +SetFlexibleDirection() wxComboBox +SetCancelButton() +wxInputHandler +SetNegativeButton() +Create() +SetInsertionPoint() «boundary» +SelectAll() wx-L1 wxGridBagSizer +GetCount() +SetSelection()

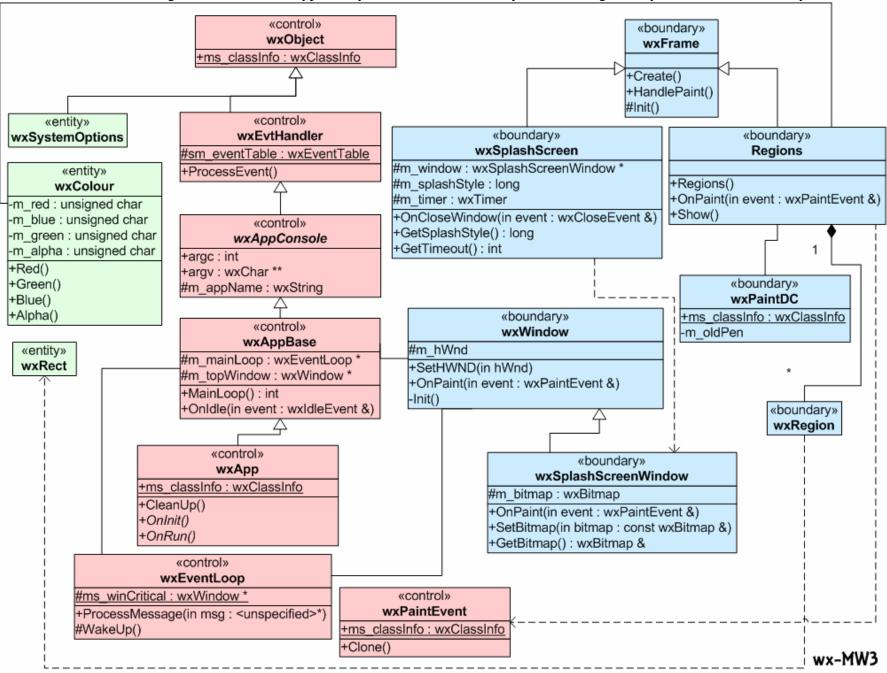


The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» wxObject wxWindowBase Д «entity» «control» «boundary» «boundary» wxSizerItem wxSizer wxLayoutAlgorithm wxWindow #m window : wxWindow * #m position #m_sizer:wxSizer* +LayoutFrame() #m children +SetSizer() #m minSize: wxSize #m containingWindow +LayoutWindow() +SetSizerandFit() +CalcMin() +Add() Д +AddStretchSpacer() +SetSizer() +InsertSpacer() «boundary» «boundary» «boundary» +SetMinSize() wxFrame wxControl wxPanel «entity» +SetItemMinSize() wxSizerItemList #m_subControls #m container +RecalcSizes() +Create() Create() +Layout() +ProcessCommand() OnSize() +Insert() +Fit() +GetLabel() #Init() +size() +Show() +Hide() «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» wxStaticText wxTextCtrl «boundary» «boundary» wxNotebook wxButton wxCheckBox wxGridSizer wxBoxSizer wxNotebookSizer #m nSelection -m isPressed #m rows:int #m stretchable : int #m tabView +Create() +DoLoadFile() -m_isHot +SetLabel() +Copy() #m minWidth:int #m_cols : int +GetNotebook() +Create() +SetFont() #SendClickEvent() +SetValue() #m minHeight:int +Cut() +RecalcSizes() +SetPageText() +SetForegroundColour() +Paste() +RecalcSizes() +SetPageSize() +SetCols() «boundary» +SetRows() +InsertPage() +CalcMin() wxControlWithItems +Add() Д +Append() «boundary» +Insert() wxFlexGridSizer +Select() #m rowHeights «boundary» «boundary» #m colWidths wxStdDialogButtonSizer wxStaticBoxSizer #m calculatedMinSize «boundary» wxComboBox +RecalcSizes() +AddButton() +GetStaticBox() +AddGrowableCol() +wxInputHandler +SetAffirmativeButton() +SetFlexibleDirection() +SetCancelButton() +Create() +SetNegativeButton() +SetInsertionPoint() +SelectAll() «boundary» +GetCount() wx-L3 wxGridBagSizer +SetSelection()

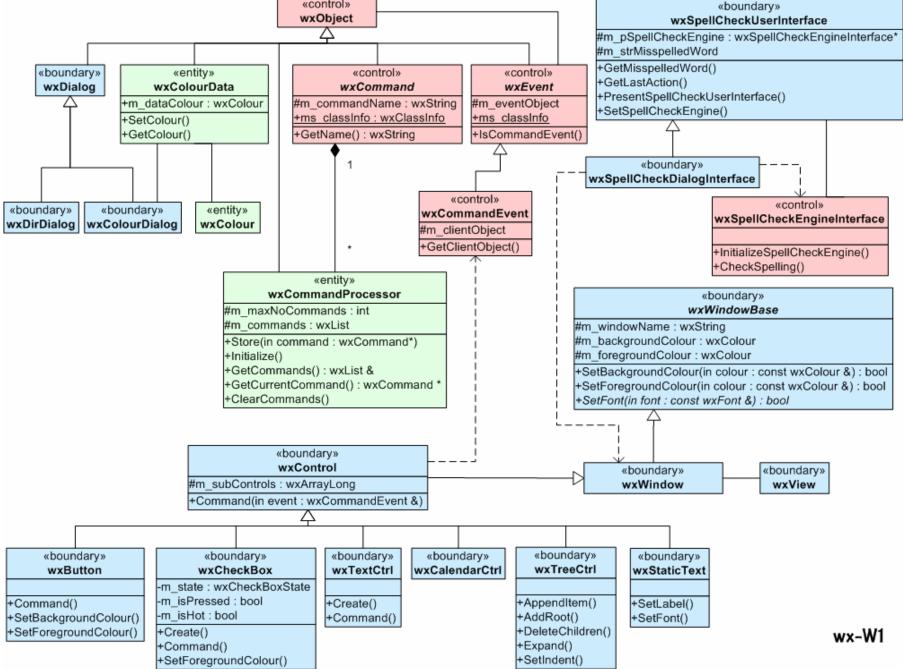


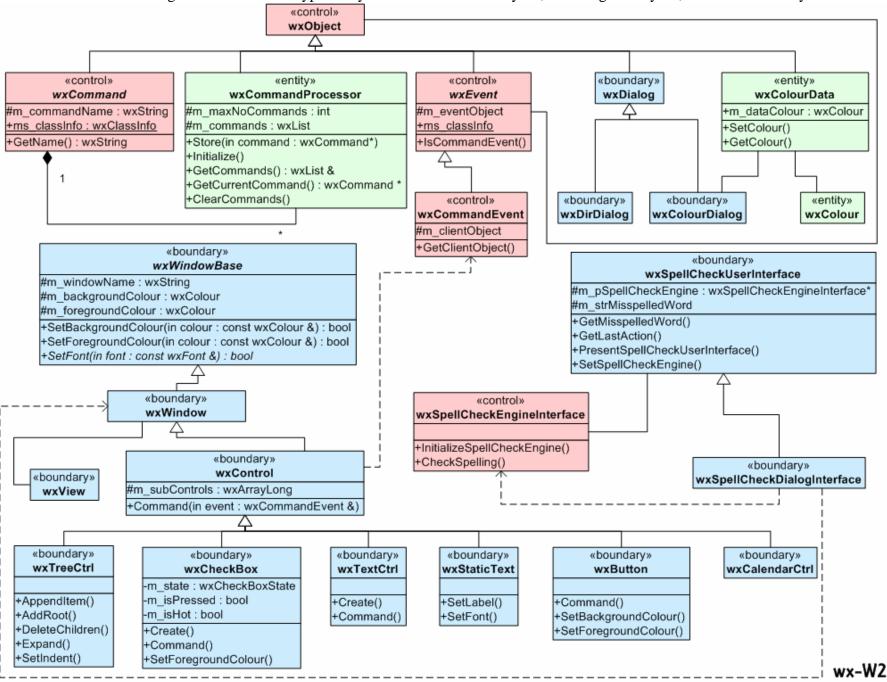
The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» «boundary» wxObject wxSplashScreen wxFrame +ms classInfo : wxClassInfo #m_window:wxSplashScreenWindow* #m splashStyle : long Ą +Create() #m_timer : wxTimer +HandlePaint() +OnCloseWindow(in event : wxCloseEvent &) «control» #Init() «entity» +GetSplashStyle(): long wxEvtHandler wxSystemOptions +GetTimeout(): int #sm_eventTable : wxEventTable +ProcessEvent() «entity» «boundary» «entity» wxColour wxRegion wxRect «control» -m red: unsigned char wxAppConsole -m_blue : unsigned char +argc : int -m green : unsigned char +argv : wxChar ** -m alpha: unsigned char «boundary» #m_appName: wxString +Red() Regions +Green() +Blue() +Regions() «control» +Alpha() +OnPaint(in event : wxPaintEvent &) wxAppBase +Show() #m_mainLoop : wxEventLoop * #m_topWindow : wxWindow * «boundary» wxPaintDC +MainLoop(): int +Onldle(in event : wxldleEvent &) +ms classInfo : wxClassInfo -m oldPen «boundary» «control» wxWindow qqAxw «control» wxPaintEvent +ms classInfo : wxClassInfo #m hWnd +ms classInfo : wxClassInfo +SetHWND(in hWnd) +CleanUp() +OnPaint(in event : wxPaintEvent &) +OnInit() +Clone() +OnRun() -Init() «control» «boundary» wxEventLoop wxSplashScreenWindow #ms_winCritical: wxWindow * #m bitmap : wxBitmap +ProcessMessage(in msg : <unspecified>*) +OnPaint(in event : wxPaintEvent &) #WakeUp() +SetBitmap(in bitmap : const wxBitmap &) +GetBitmap(): wxBitmap &

wx-MW2



The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» «boundary» wxObject wxSpellCheckUserInterface





The last number of the diagram id denotes the type of layout: 1= multi cluster layout, 2=orthogonal layout, 3=three-cluster layout «control» wxObject Δ «control» «control» «entity» «entity» wxEvent wxCommand wxCommandProcessor wxColourData #m eventObject #m commandName: wxString #m maxNoCommands : int +m dataColour : wxColour +ms classInfo +ms_classInfo : wxClassInfo #m commands : wxList +SetColour() +IsCommandEvent() +GetName(): wxString +Store(in command : wxCommand*) +GetColour() +Initialize() Д +GetCommands(): wxList & 1 «entity» -GetCurrentCommand(): wxCommand * «control» wxColour +ClearCommands() «control» wxSpellCheckEngineInterface wxCommandEvent #m clientObject +InitializeSpellCheckEngine() +GetClientObject() «boundary» -CheckSpelling() wxWindowBase «boundary» #m windowName : wxString wxDialog #m_backgroundColour : wxColour «boundary» #m foregroundColour : wxColour wxView +SetBackgroundColour(in colour : const wxColour &) : bool +SetForegroundColour(in colour : const wxColour &) : bool «boundary» «boundary» +SetFont(in font : const wxFont &) : bool wxDirDialog wxColourDialog «boundary» wxWindow «boundary» wxSpellCheckUserInterface «boundary» #m pSpellCheckEngine: wxSpellCheckEngineInterface* wxSpellCheckDialogInterface #m strMisspelledWord «boundary» +GetMisspelledWord() wxControl +GetLastAction() #m_subControls : wxArrayLong PresentSpellCheckUserInterface() +Command(in event : wxCommandEvent &) SetSpellCheckEngine() «boundary» «boundary» «boundary» «boundary» «boundary» «boundary» wxTreeCtrl wxCheckBox wxTextCtrl wxStaticText wxButton wxCalendarCtrl -m state: wxCheckBoxState -m_isPressed : bool +AppendItem() +Create() +SetLabel() +Command() -m isHot : bool +AddRoot() +SetBackgroundColour() +Command() +SetFont() +DeleteChildren() +Create() +SetForegroundColour() +Expand() +Command() +SetIndent() +SetForegroundColour() wx-W3