JIDE Gantt Chart Developer Guide

Contents

Purpose of This Document

A Gantt chart is a graphical representation of the duration of tasks against the progression of time. It is a useful tool for planning and scheduling projects. A Gantt chart is helpful when monitoring a project's progress.

JIDE Gantt Chart is a pure Java Swing library that makes it possible to create a Gantt chart inside your Swing application. Just like many other components from JIDE, it provides all the basic features and a lot of APIs for you to further customize for your application.

Features

Here are the main features of *JIDE Gantt Chart*.

- Tree table view and GanttChart view side by side.
- Supports mouse interactive to change starting date, ending date and completion of the task entry

- GanttEntryRenderer support. User can write one's own renderer to render the Gantt entry and entry group on the GanttChart.
- GanttLabelRenderer support. User can write one's own renderer to render the label for the Gantt entry on the GanttChart.
- Keyboard navigation supported.
- Entry group supported.
- Multiple levels of periods on the scale area.
- Period format support on the scale area.
- Zoom in and zoom out.
- Entry relationship supported.
- Support both numeric and date as the period
- Context menu supported.

JIDE Gantt Chart

Here is what a Gantt chart looks like under Window L&F under Office 2007 style.

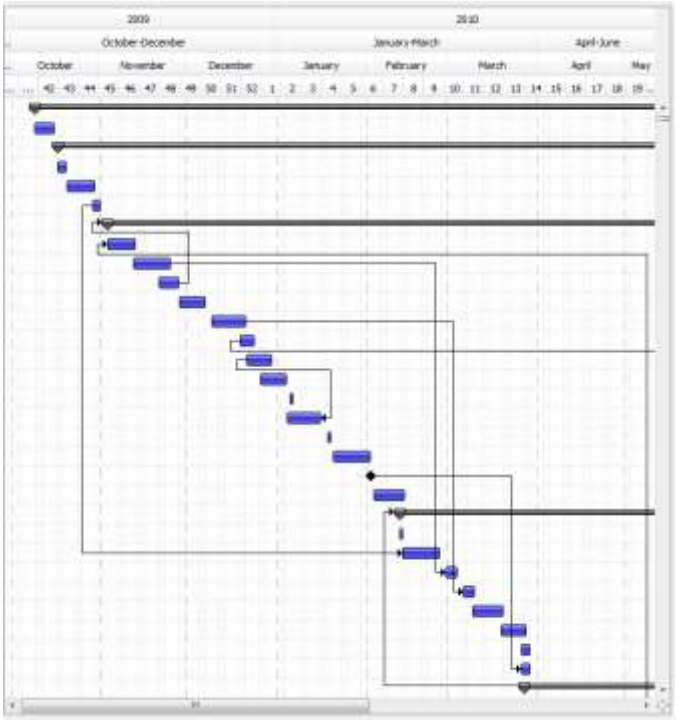


Figure 1 JIDE Gantt Chart

On the left hand side, there is *TreeTable* from *JIDE Grids*.

Overview of JIDE Gantt Chart APIs

GanttEntry

GanttEntry is an interface representing an entry in a Gantt chart. It has a name, a range (i.e. the starting date and the ending date) and a completion percentage. GanttEntry only has getters for those properties. *GanttEntry* extends *ExpandableRow* which means it can be used inside a *TreeTableModel*, a model defined in the JIDE Grids product.

MutableGanttEntry interface extends *GanttEntry* and adds setters for those properties.

DefaultGanttEntry implements the *MutableGanttEntry* and provides the getters/setters for all the properties.

Please note, there is no separate class for the Gantt entry group. If a *GanttEntry* has children, it automatically becomes an entry group and will be rendered as a group in the UI. If a *DefaultGanttEntry* is rendered as a group, it will update its range automatically to the union of the ranges of all its child entries.

If the range has the same start and end, it will be rendered as a milestone.

GanttModel

GanttModel contains a list of *GanttEntries*. Since an entry could have children entries, it forms a tree hierarchy. Here are the three main methods on *GanttModel* interface.

```
/**
 * Gets the total entry count.
 *
 * @return the total entry count.
 */
int getEntryCount();

/**
 * Gets the entry at the entry index.
 *
 * @param index the entry index.
 * @return Returns the GanttEntry at the index or null if the index is out of bounds.
 */
S getEntryAt(int index);

/**
 * Gets the index of the entry.
 *
 * @param entry the entry
 * @return the index.
 */
int getIndexOf(S entry);
```

Note the generic type *S* is defined as extending from *GanttEntry*. In the other word, users can define their own subclass of *GanttEntry* and use it inside *GanttModel*.

GanttModel also defined several sub-models. They are *TreeTableModel*, *ScaleModel* and *GanttEntryRelationModel*.

```
/**
```

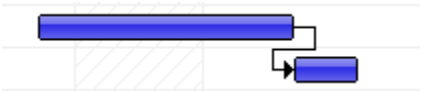
```
     * @return The TreeTableModel to be displayed in the TreeTable of the GanttChartPane.
     */
    ITreeTableModel<S> getTreeTableModel();

    /**
     * Gets the ScaleModel.
     *
     * @return the ScaleModel.
     */
    ScaleModel<T> getScaleModel();

    /**
     * Gets the model that defines the relationship among gantt entries.
     *
     * @return the GanttEntryRelationModel.
     */
    GanttEntryRelationModel<T, S> getGanttEntryRelationModel();
```

The *TreeTableModel* is the same model that is used by *TreeTable* in JIDE Grids product. Since *TreeTableModel* is a tree hierarchical data model that supports filtering and sorting, we decide to leverage it as Gantt entries are also represented in a hierarchical data model and also need filtering and sorting. Since we will use this *TreeTableModel* and display it in a *TreeTable* in the case of GanttChartPane, so using *TreeTableModel* seems like a good choice.

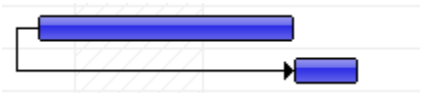
The *ScaleModel* provides information needed to display different scales in the *ScaleArea*. The model provides functions to map an instant in the base unit to a position on a long scale.

The *GanttEntryRelationModel* defines relationships between two entries. There are four types of relationships that are defined as constants on *GanttEntryRelation*. In order to explain the relation, we call the first entry the predecessor entry and the second entry the successor entry. Please note, the line that represents the relation always draws from the predecessor entry to the successor entry.

- ENTRY_RELATION_FINISH_TO_START: The successor entry cannot begin until the task that it depends (the predecessor entry) on is complete.

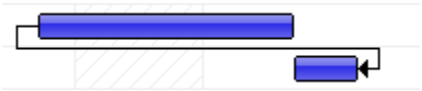


- ENTRY_RELATION_START_TO_START: The successor entry cannot begin until the task that it depends (the predecessor entry) has begun.



- ENTRY_RELATION_FINISH_TO_FINISH: The successor entry cannot be completed until the task that it depends on (the predecessor entry) is completed.

- ENTRY_RELATION_START_TO_FINISH: The successor entry cannot be completed until the task that it depends on (the predecessor entry) begins.



In JIDE Gantt Chart, we display the four relations of the entries but we didn't enforce them. You would still need to implement the logic in your code.

DefaultGanttModel

Although you can implement your own *GanttModel*, most likely you will stay with *DefaultGanttModel*, which is a default implement we provided for *GanttModel* interface. *DefaultGanttModel* allows you to define the period unit type and the *GanttEntry* type as generic. In the example code below, we created a *DefaultGanttModel* whose period unit type is Date and the *GanttEntry* is *DefaultGanttEntry*.

```
DefaultGanttModel<Date, DefaultGanttEntry<Date>> model = new DefaultGanttModel<Date,
DefaultGanttEntry<Date>>();
model.setScaleModel(new DateScaleModel());
```

Gantt Chart

Once you have the *GanttModel* created, creating a *GanttChart* is very simple.

```
GanttModel<T, S> ganttModel = …;
GanttChart<T, S> chart = new GanttChart<T, S>();
chart.setModel(ganttModel);
```

Markers

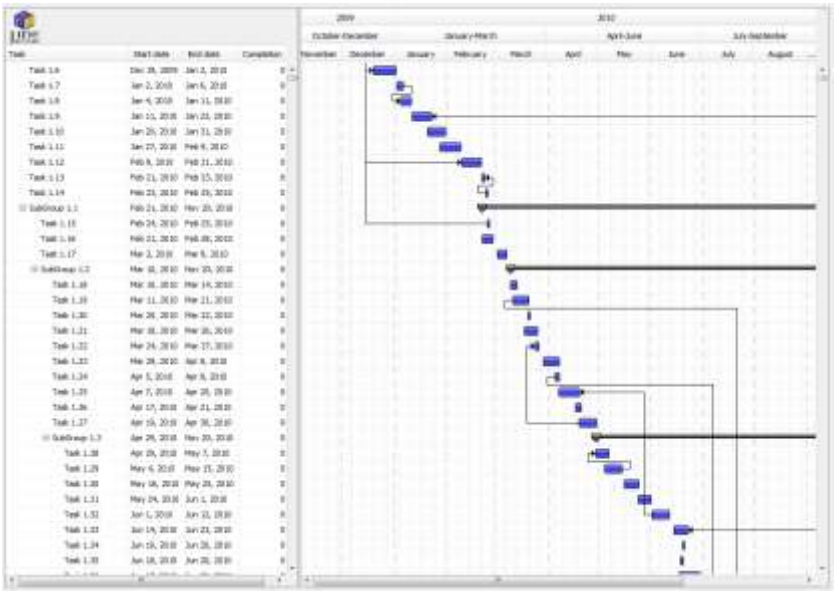
The *GanttChart* can be customized with background painter to mark certain range or periods, for example to mark the weekends on the chart. The *IntervalMarker* can be used to mark any interval or instant on the scale, while the *AbstractPeriodMarker* can be used to mark the periods.

See the *GanttChartDemo* for examples of *IntervalMarker* and *DatePeriodMarker*.

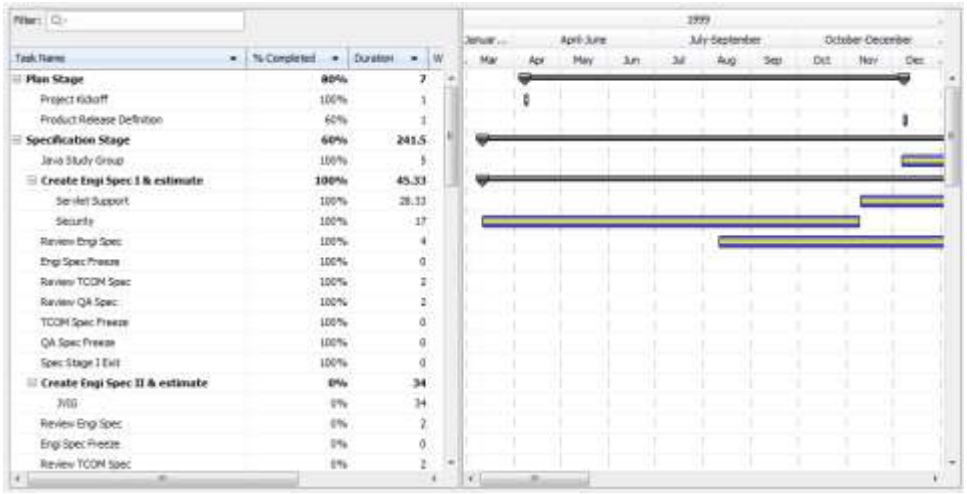
GanttChartPane

In addition to *GanttChart*, we also create a class called *GanttChartPane*. This is a pane that has both a *TreeTable* and a *GanttChart*.

```
GanttModel<T, S> ganttModel = …;
GanttChartPane ganttChartPane = new GanttChartPane<Integer, GanttEntry<Integer>>(ganttModel);
```



By default, the *TreeTable* has five columns as shown above. However you can customize it by creating your *TreeTableModel*. See below for a screenshot from *ProjectGanttChartDemo* which is one of the demos on Gantt chart.



ScaleModel

The *ScaleModel* is an interface to provide the *GanttChart* with information how to translate the *GanttEntry* range to positions on a long scale. It also defines the available periods and how they map to intervals on the long scale.

If you implement your own ScaleModel, it is important to follow the invariants specified in the Javadoc of the *ScaleModel* – especially that period intervals always ends at the same instant where the next period interval begins. But typically a *GanttChart* deals with time for which the *DateScaleModel* is provided to map *java.util.Date* to a millisecond long scale.

You can customize the available periods of by providing the appropriate *DatePeriod*s to the *DateScaleModel* constructor (always from the smallest to the largest). If the predefined periods do not include the period you want, you can specify your own *DatePeriod* as below. Note that if you want irregular periods, you'll need to create or extend your own custom *ScaleModel* which correctly implements *getPeriodStart/End* and *calculatePeriods* for that period.

```
// A fiscal year which starts in July instead of January
DatePeriod fiscalYear = new DatePeriod(Calendar.MONTH, 12, Calendar.JULY);
DateScaleModel scaleModel = new DateScaleModel(
        DatePeriod.DAY_OF_WEEK,
        DatePeriod.WEEK_OF_YEAR,
        DatePeriod.MONTH,
        fiscalYear);

// don't forget to add a PeriodConverter for your new period
scaleArea.setPeriodConverter(fiscalYear,
        new DatePeriodConverter("Fiscal Year",
                new SimpleDateFormat("'FY' yyyy"),
                new SimpleDateFormat("'Fiscal Year' yyyy")));
```

ScaleArea

The *ScaleArea* is the header of the Gantt chart and can show the scale in different granularities. It allows the user to pick from the available periods defined in the *ScaleModel* and zoom the *GanttChart* in and out. The default setup of the *ScaleArea* can be customized by setting the visible periods, the start and end instants and the preferred visible periods count and size to control the preferred width of the *GanttChart*.

*ScaleAreaPopupMenuCustomizer*s can be installed on the *ScaleArea* to provide a context sensitive popup menu depending on which period is clicked. By default the resize periods and the visible periods' popup menu customizers are installed.

For each period, a *PeriodConverter* can be set on the *ScaleArea* to control the text and tooltip of that period. *DateGanttChartPane* configures the *ScaleArea* with *PeriodConverter*s for the default *DatePeriod*s.

By default if the *GanttChart* is used in a *GanttChartPane*, we will renderer the period headers using the *TableHeader* of the *TreeTable* for a consistent look and feel. If you replace the table header of the *TreeTable*, special care might need to be taken so the *ScaleArea* renders correctly. See the *ProjectGanttChartDemo* for an example.

Mouse and Keyboard Support in GanttChart

Mouse

In addition to mouse click, press and drag to select the rows in *GanttChart*, *GanttChart* also supports mouse zoom and mouse editing. If you put the mouse over the entry, there are four possible actions depending on where the mouse is: move, resize upper range, resize lower range, and change completion. The cursor will change shape to tell you which action it will take when the mouse is pressed and dragged.

The mouse wheel will scroll up and down the *GanttChart* without any modification keys. When the CTRL is pressed and hold, mouse wheel will zoom in and out the *GanttChart*.

There is also a panning mode which you can set using *GanttChart#setViewMode*. Once setting to panning mode, mouse movement will pan the *GanttChart*. This is useful when the *GanttChart* is in view-only (not editable).

Keyboard

When *GanttChart* has focus, you can use keyboard to navigate the *GanttChart*. Here are the list of keystrokes that *GanttChart* supports.

Keyboard	Action
DOWN or KP_DOWN	selectNextRow
shift DOWN or shift KP_DOWN or ctrl shift DOWN or ctrl shift KP_DOWN	selectNextRowExtendSelection
ctrl DOWN or ctrl KP_DOWN	selectNextRowChangeLead
UP or KP_UP	selectPreviousRow
shift UP or shift KP_UP or ctrl shift UP or ctrl shift KP_UP	selectPreviousRowExtendSelection
ctrl UP or ctrl KP_UP	selectPreviousRowChangeLead
HOME	selectFirstRow
shift HOME or ctrl shift HOME	selectFirstRowExtendSelection
ctrl HOME	selectFirstRowChangeLead
END	selectLastRow
shift END or	selectLastRowExtendSelection

ctrl shift END	
ctrl END	selectLastRowChangeLead
PAGE_UP	scrollUp
shift PAGE_UP or ctrl shift PAGE_UP	scrollUpExtendSelection
ctrl PAGE_UP	scrollUpChangeLead
PAGE_DOWN	scrollDown
shift PAGE_DOWN or ctrl shift PAGE_DOWN	scrollDownExtendSelection
ctrl PAGE_DOWN	scrollDownChangeLead
TAB	selectNextColumnCell
shift TAB	selectPreviousColumnCell
ENTER	selectNextRowCell
shift ENTER	selectPreviousRowCell
ctrl A or ctrl SLASH	selectAll
ctrl BACK_SLASH	clearSelection
SPACE	addToSelection
ctrl SPACE	toggleAndAnchor
shift SPACE	extendTo
ctrl shift SPACE	moveSelectionTo
ctrl EQUALS	zoomInPeriods
ctrl MINUS	zoomOutPeriods

Internationalization Support

All Strings used in *JIDE Gantt Chart* are contained in one properties file called gantt.properties under com/jidesoft/gantt and scale.properties under com/jidesoft/scale. Some users contributed localized version of this file and we put those files inside jide-properties.jar. If you want to support languages other than those we provided, just extract this properties file, translated to the language you want, add the correct postfix and then jar it back into jide-properties jar. You are welcome to send the translated properties file back to us if you want to share it.