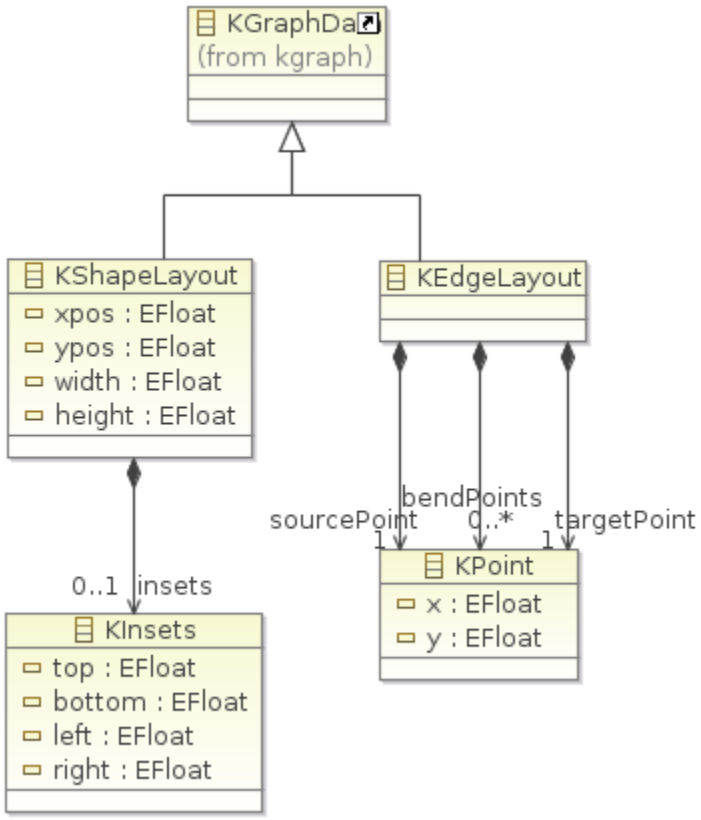
KLayoutData Meta Model



The layout data structure was generated with EMF and is used in KIML to attach layout options as well as layout information to each element of a KGraph s tructure. Layout options specify the behavior of layout algorithms for the specific graph instance, and layout information is written by layout algorithms to describe the position and size of each graph element. Many commonly used layout options are defined in the class LayoutOptions, while each layout algorithm may additionally define its own specific options.

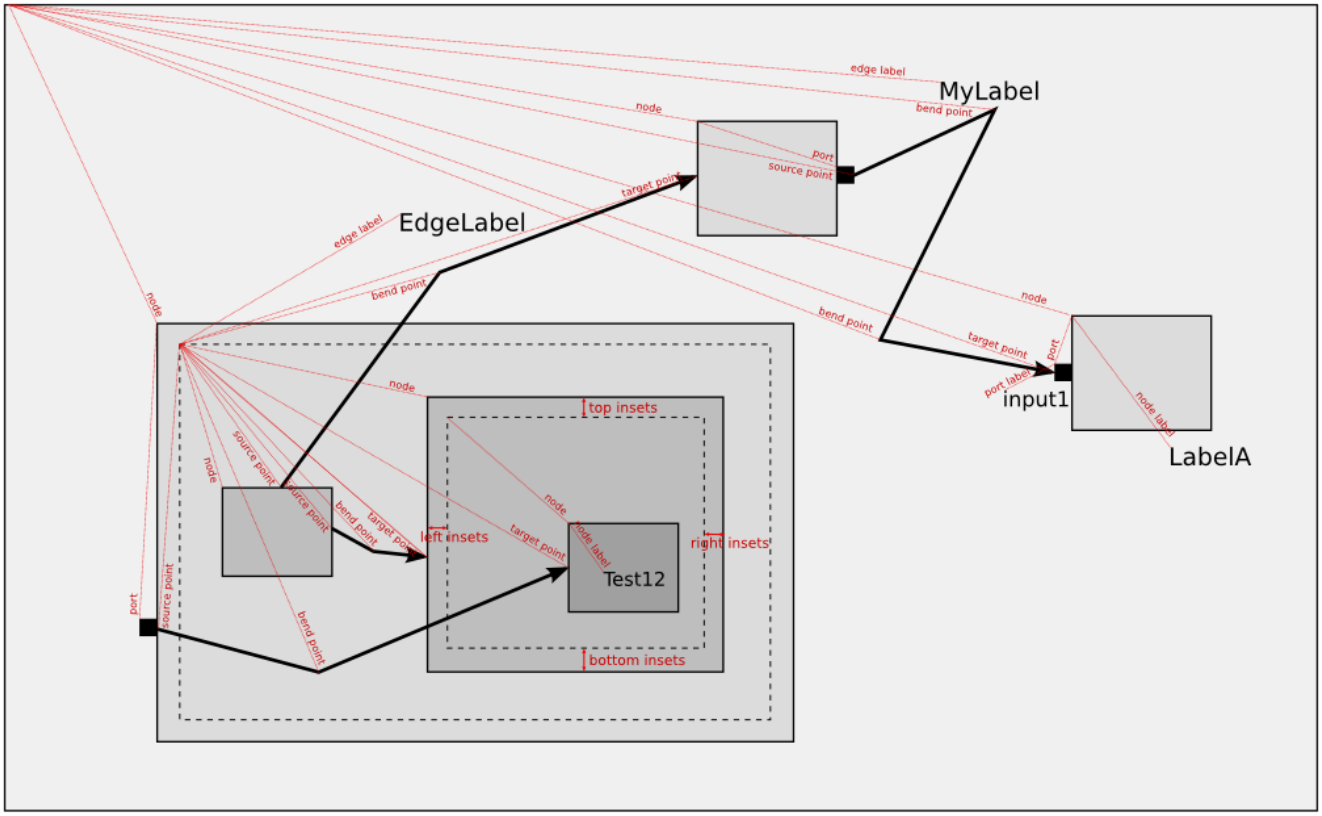
The KShapeLayout information is used for nodes, labels, and ports. It stores the position coordinates of the top left corner of the corresponding graph element as well as the size of its bounding box. Nodes may contain nested sub-graphs represented by their children reference; these child nodes are drawn inside the parent's bounding box respecting a certain border given by the KInsets object. The KEdgeLayout information is used for edges. It stores the position coordinates of the points where an edge touches its source node and its target node as well as the bend points.

The meta model and generated data structure is contained in the plugin `de.cau.cs.kieler.kiml`.

Graph elements have the following reference points for their position:

- Nodes: the parent node position plus its top / left insets (the *child area*)
- Node labels: the position of the containing node
- Ports: the position of the containing node
- Port labels: the position of the containing port
- Edge bend points and source / target points: the source node child area if the target node is directly or indirectly contained by the source node, the source node's parent child area otherwise
- Edge labels: same as bend points

Consider the following example, where graph element positions are visualized as red dotted vectors pointing to the corresponding reference points. See also the SVG version.



API Example

```
// Suppose there is a 'KNode node' and a 'KEdge edge',
// now edit layout data attributes of the elements.
KLayoutDataFactory layoutDataFactory = KLayoutDataFactory.eINSTANCE;
KShapeLayout nodeLayout = node.getData(KShapeLayout.class);
nodeLayout.setXPos(10);
nodeLayout.setYPos(10);
nodeLayout.setHeight(100);
nodeLayout.setWidth(100);

KEdgeLayout edgeLayout = edge.getData(KEdgeLayout.class);
KPoint sourcePoint = layoutDataFactory.createKPoint();
sourcePoint.setX(10);
sourcePoint.setY(10);
edgeLayout.setSourcePoint(sourcePoint);
KPoint targetPoint = layoutDataFactory.createKPoint();
targetPoint.setX(110);
targetPoint.setY(110);
edgeLayout.setTargetPoint(targetPoint);
KPoint bendPoint = layoutDataFactory.createKPoint();
bendPoint.setX(50);
bendPoint.setY(50);
edgeLayout.getBendpoints().add(bendPoint);

// Work with layout options using the IPropertyHolder interface.
edgeLayout.setProperty(LayoutOptions.EDGE_TYPE, EdgeType.GENERALIZATION);
nodeLayout.setProperty(LayoutOptions.PRIORITY, 5);
Boolean isInteractive = nodeLayout.getProperty(LayoutOptions.INTERACTIVE);
```

A larger example that uses the KGraph with layout data for automatic layout is found in standalone/LayoutTest/src/test/layout/Test.java.