KGraph Meta Model

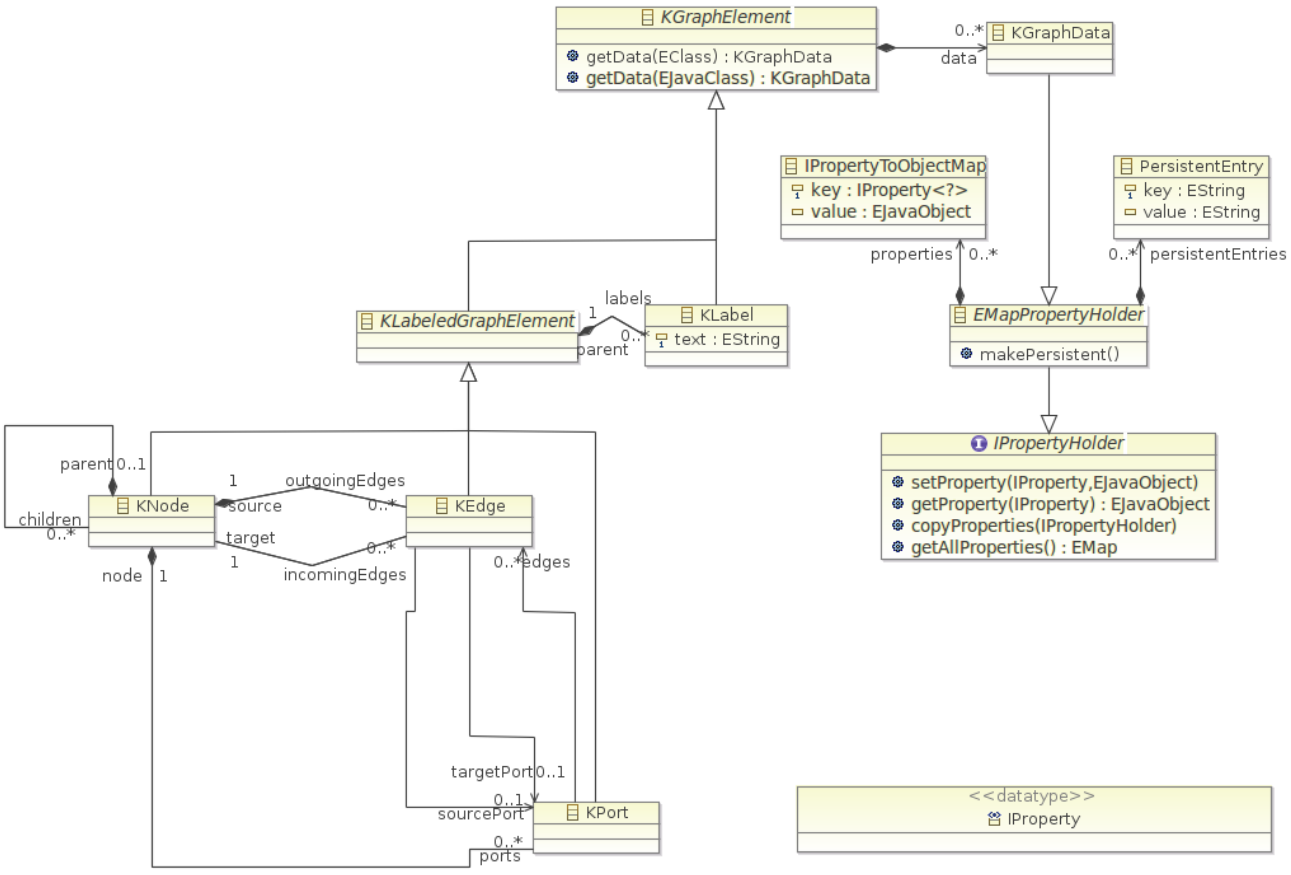
KGraph is the central data structure in KIELER for representation of graphs. It was generated using EMF and therefore inherits all capabilities of EMF models such as storage in XML format, transformation, and validation. The KGraph is used for several purposes:

- as intermediate data structure in the KIML project for exchanging layout information between graphical diagrams and graph layout algorithms,
- as a file format for graphs in the KWebS project, and
- as basic structure for the notational model of the KLighD project.

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

See also KLayoutData and KRendering.

The Meta Model



Each node in a KGraph may contain other nodes, thus representing a nested sub-graph. The graph itself is represented by a top-level node that has no parent node. Nodes may have incoming and outgoing edges, and each edge has references to its source node and its target node. Usage of ports is optional and can be applied to special cases such as data flow diagrams. If ports are used, they each contain a list of connected edges, and the edges have references to their corresponding source port and target port. Nodes, edges, and ports may have arbitrarily many labels, which are usually (but not necessarily) represented by text.

Each element of the graph may contain arbitrary additional data, which must implement the interface KGraphData. There are two important extending models that use this interface: KLayoutData, which is used to store layout data for the graph, and KRendering, which adds graphical information. Graph data inherits from the plain Java IPropertyHolder interface, which enables it to provide a key-value mapping. Keys are IProperty instances, which have an identifier string and a type, and values are arbitrary objects of the corresponding type. This mapping is transient and is thus not stored in the XML format. The makePersistent() operation can be used to translate the mapping into a serializable list of string pairs.

Editing KGraphs

KGraphs can be easily created and edited using our Textual KGraph (KGT) format.

API Example

```
KGraphFactory factory = KGraphFactory.eINSTANCE;
KNode myGraph = factory.createKNode();

// Setting the parent reference automatically adds a node to
// the parent's list of children.
KNode node1 = factory.createKNode();
node1.setParent(myGraph);
KNode node2 = factory.createKNode();
node2.setParent(myGraph);

// Setting the source and target references automatically adds
// an edge to the corresponding incoming and outgoing edges lists.
// In the XML structure edges are contained by their source node.
KEdge edge = factory.createKEdge();
edge.setSource(node1);
edge.setTarget(node2);

// Setting the parent reference automatically adds a label to
// the parent's list of labels.
KLabel label = factory.createKLabel();
label.setText("Hello, World!");
label.setParent(edge);

// Setting the node reference automatically adds a port to the
// node's list of ports.
KPort port1 = factory.createKPort();
port1.setNode(node1);
KPort port2 = factory.createKPort();
port2.setNode(node2);

// Setting the source port / target port reference automatically
// adds an edge to the port's list of edges.
edge.setSourcePort(port1);
edge.setTargetPort(port2);
```

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