

```

LayeredEdge
-original: EKEdge
-reversed: boolean
-dummy: boolean
-bindPoints: ArrayList<Point>[]
-conflicted: boolean[]
<<Property>>-sources: LayeredGraphNode
<<Property>>-targets: LayeredGraphNode
<<Property>>-graph: LayeredGraphNode
+LayeredEdge(original: EKEdge, sources: ArrayList<LayeredGraphNode>, targets: ArrayList<LayeredGraphNode>, graph: LayeredGraphNode)
+isConflicted(layout: LayoutType): boolean
+setConflicted(conflicted: boolean, layout: LayoutType): void
+setStartPoint(x: int, y: int, layout: LayoutType): void
+addBindPoint(x: int, y: int, layout: LayoutType): void
+setEndPoint(x: int, y: int, layout: LayoutType): void
+getLinePoints(layout: LayoutType): ArrayList<Point>
+getOriginalEdge(): EKEdge
+remove(): void
+isCrossLayerEdge(): boolean
+replaceByDummyNodes(): void
+reverse(): void
+isDummyEdge(): boolean
+removeDummyNodes(): void
+setDummyEdge(): void
+setReverseEdge(): void
+isReverseEdge(): boolean
+calcEdgeCrossing(): ArrayList<LayeredGraphEdge>
+isConflicted(layout: LayoutType): boolean
+setConflicted(conflicted: boolean, layout: LayoutType): void
+setStartPoint(x: int, y: int, layout: LayoutType): void
+addBindPoint(x: int, y: int, layout: LayoutType): void
+setEndPoint(x: int, y: int, layout: LayoutType): void
+getLinePoints(layout: LayoutType): ArrayList<Point>
+toString(): String
  
```

io

The contents of the io package are displayed in another class diagram.

can be used for generating random graphs

```

LayeredGraphNode
<<Property>>-originalNode: EKNode
<<Property>>-name: String
<<Property>>-selected: LayoutType
<<Property>>-dummyNode: boolean
<<Property>>-layer: int
<<Property>>-containedLayers: ArrayList<ArrayList<LayeredGraphNode>>
<<Property>>-outgoingEdges: LayeredGraphEdge
<<Property>>-incomingEdges: LayeredGraphEdge
<<Property>>-sortedOutgoingEdges: LayeredGraphEdge
<<Property>>-sortedIncomingEdges: LayeredGraphEdge
<<Property>>-parent: LayeredGraphNode
<<Property>>-containedEdges: LayeredGraphEdge
<<Property>>-containedNodes: LayeredGraphNode
<<Property>>-sortedContainedNodes: LayeredGraphNode
<<Property>>-mouseOver: boolean
+setShift(shift: double, layout: LayoutType): void
+getShift(layout: LayoutType): double
+setSink(sink: LayeredGraphNode, layout: LayoutType): void
+getSink(layout: LayoutType): LayeredGraphNode
+isUndefined(layout: LayoutType): boolean
+setRoot(root: LayeredGraphNode, layout: LayoutType): void
+isRoot(layout: LayoutType): LayeredGraphNode
+setColor(color: Color, layout: LayoutType): void
+getColor(layout: LayoutType): Color
+isSelected(layout: LayoutType): boolean
+remove(): void
+parent(): LayeredGraphNode
+setX(x: double, def: boolean, layout: LayoutType): void
+getY(y: double, layout: LayoutType): void
+getX(layout: LayoutType): double
+getY(layout: LayoutType): double
+getWidth(layout: LayoutType): double
+getHeight(layout: LayoutType): double
+setWidth(w: double, layout: LayoutType): void
+setHeight(h: double, layout: LayoutType): void
+getNodeLayer(n: LayeredGraphNode): int
+isOrderedLayer(indizes: ArrayList<Double>, layerIndex: int): void
+isNodeLayer(n: LayeredGraphNode, index: int): void
+removeEdge(e: LayeredGraphEdge): void
+removeNodes(): LayeredGraphNode
+getOutgoingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+getSortedOutgoingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+getIncomingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+getSortedIncomingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+createNode(original: EKNode): LayeredGraphNode
+createEdge(original: EKEdge, sources: ArrayList<LayeredGraphNode>, targets: ArrayList<LayeredGraphNode>): LayeredGraphEdge
+createSimpleEdge(original: EKEdge, source: LayeredGraphNode, target: LayeredGraphNode): LayeredGraphEdge
+findEdgeFromOriginal(original: Object): LayeredGraphEdge
+findNodeFromOriginal(original: Object): LayeredGraphNode
+findNodeByName(name: String): LayeredGraphNode
+addNode(n: LayeredGraphNode): void
+addEdge(e: LayeredGraphEdge): void
+setAlign(algn: LayeredGraphNode, layout: LayoutType): void
+getAlign(layout: LayoutType): LayeredGraphNode
+unsetAlign(): void
+setShift(shift: double, layout: LayoutType): void
+getShift(layout: LayoutType): double
+setSink(sink: LayeredGraphNode, layout: LayoutType): void
+getSink(layout: LayoutType): LayeredGraphNode
+isUndefined(layout: LayoutType): boolean
+setAlign(algn: LayeredGraphNode, layout: LayoutType): void
+getAlign(layout: LayoutType): LayeredGraphNode
+setRoot(root: LayeredGraphNode, layout: LayoutType): void
+isRoot(layout: LayoutType): LayeredGraphNode
+setColor(color: Color, layout: LayoutType): void
+getColor(layout: LayoutType): Color
+getClassColor(layout: LayoutType): Color
+isSelected(layout: LayoutType): boolean
+setX(x: double, def: boolean, layout: LayoutType): void
+getY(y: double, layout: LayoutType): void
+getX(layout: LayoutType): double
+getY(layout: LayoutType): double
+getWidth(layout: LayoutType): double
+getHeight(layout: LayoutType): double
+setWidth(w: double, layout: LayoutType): void
+setHeight(h: double, layout: LayoutType): void
+findEdgeBetweenSource(s: LayeredGraphNode, target: LayeredGraphNode): LayeredGraphEdge
  
```

```

LayeredGraphEdge
<<Property>>-originalEdge: EKEdge
<<Property>>-crossLayerEdge: boolean
<<Property>>-reverseEdge: boolean
<<Property>>-dummyEdge: boolean
<<Property>>-sources: LayeredGraphNode
<<Property>>-targets: LayeredGraphNode
+replaceByDummyNodes(): void
+reverse(): void
+setStartPoint(x: int, y: int, layout: LayoutType): void
+setEndPoint(x: int, y: int, layout: LayoutType): void
+addBindPoint(x: int, y: int, layout: LayoutType): void
+getLinePoints(layout: LayoutType): ArrayList<Point>
+removeDummyNodes(): void
+setDummyEdge(): void
+setReverseEdge(): void
+calcEdgeCrossing(): ArrayList<LayeredGraphEdge>
+isConflicted(layout: LayoutType): boolean
+setConflicted(conflicted: boolean, layout: LayoutType): void
+setStartPoint(x: int, y: int, layout: LayoutType): void
+addBindPoint(x: int, y: int, layout: LayoutType): void
+setEndPoint(x: int, y: int, layout: LayoutType): void
+getLinePoints(layout: LayoutType): ArrayList<Point>
  
```

```

RandomGraphGenerator
+Subgraph: double
+Edge: double
+minLayer: int
+maxLayer: int
+minNodePerLayer: int
+maxNodePerLayer: int
+maxDepth: int
+MAX_TRIES: int = 100
+RandomGraphGenerator(subgraph: double, pEdge: double, minLayer: int, maxLayer: int, minNodePerLayer: int, maxNodePerLayer: int, maxDepth: int)
+createRandomNode(parent: LayeredGraphNode, depth: int): LayeredGraphNode
+createRandomNode(parent: LayeredGraphNode, depth: int, prove: boolean): LayeredGraphNode
+validateGraph(layout: LayeredGraphNode): boolean
  
```

```

LayeredNode
original: EKNode
dummy: boolean
<<Property>>-name: String
layers: ArrayList<ArrayList<LayeredGraphNode>>
-parent: LayeredGraphNode
-layers: LayoutInfo[]
-combined: CombinedLayoutInfo
-edges: LayeredGraphEdge
-nodes: LayeredGraphNode
<<Property>>-mouseOver: boolean
<<Property>>-attribute: LayeredGraphNode
+LayeredNode(original: EKNode, parent: LayeredGraphNode)
+setDummyNode(dummy: boolean): void
+isDummyNode(): boolean
+setShift(shift: double, layout: LayoutType): void
+getShift(layout: LayoutType): double
+setSink(sink: LayeredGraphNode, layout: LayoutType): void
+getSink(layout: LayoutType): LayeredGraphNode
+isUndefined(layout: LayoutType): boolean
+setRoot(root: LayeredGraphNode, layout: LayoutType): void
+isRoot(layout: LayoutType): LayeredGraphNode
+setSelected(layout: LayoutType): void
+isSelected(layout: LayoutType): boolean
+setColor(color: Color, layout: LayoutType): void
+getColor(layout: LayoutType): Color
+getOriginalNode(): EKNode
+remove(): void
+parent(): LayeredGraphNode
+setLayer(index: int): void
+getLayer(): int
+setX(x: double, def: boolean, layout: LayoutType): void
+setY(y: double, layout: LayoutType): void
+getX(layout: LayoutType): double
+getY(layout: LayoutType): double
+getWidth(layout: LayoutType): double
+getHeight(layout: LayoutType): double
+setWidth(w: double, layout: LayoutType): void
+removeEdge(e: LayeredGraphEdge): void
+removeNodes(): LayeredGraphNode
+setNodeLayer(n: LayeredGraphNode, index: int): void
+isOrderedLayer(indizes: ArrayList<Double>, layerIndex: int): void
+getOutgoingEdges(): ArrayList<LayeredGraphEdge>
+getSortedOutgoingEdges(): ArrayList<LayeredGraphEdge>
+getIncomingEdges(): ArrayList<LayeredGraphEdge>
+getSortedIncomingEdges(): ArrayList<LayeredGraphEdge>
+getContainedEdges(): ArrayList<LayeredGraphEdge>
+getContainedNodes(): ArrayList<LayeredGraphNode>
+getSortedContainedNodes(): ArrayList<LayeredGraphNode>
+getContainedLayers(): ArrayList<ArrayList<LayeredGraphNode>>
+getNodeLayer(n: LayeredGraphNode): int
+getOutgoingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+getIncomingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+getSortedOutgoingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+getSortedIncomingEdges(n: LayeredGraphNode): ArrayList<LayeredGraphEdge>
+createNode(original: EKNode): LayeredGraphNode
+createEdge(original: EKEdge, sources: ArrayList<LayeredGraphNode>, targets: ArrayList<LayeredGraphNode>): LayeredGraphEdge
+createSimpleEdge(original: EKEdge, source: LayeredGraphNode, target: LayeredGraphNode): LayeredGraphEdge
+findEdgeFromOriginal(original: Object): LayeredGraphEdge
+findNodeFromOriginal(original: Object): LayeredGraphNode
+findNodeByName(name: String): LayeredGraphNode
+addEdge(e: LayeredGraphEdge): void
+convertToLayeredGraph(n: EKNode): LayeredGraphNode
+setAlign(algn: LayeredGraphNode, layout: LayoutType): void
+getAlign(layout: LayoutType): LayeredGraphNode
+unsetAlign(): void
+setView(view: NodeView, layout: LayoutType): void
+getView(layout: LayoutType): NodeView
+setShift(shift: double, layout: LayoutType): void
+getShift(layout: LayoutType): double
+setSink(sink: LayeredGraphNode, layout: LayoutType): void
+getSink(layout: LayoutType): LayeredGraphNode
+isUndefined(layout: LayoutType): boolean
+setAlign(algn: LayeredGraphNode, layout: LayoutType): void
+getAlign(layout: LayoutType): LayeredGraphNode
+setRoot(root: LayeredGraphNode, layout: LayoutType): void
+isRoot(layout: LayoutType): LayeredGraphNode
+setSelected(layout: LayoutType): void
+isSelected(layout: LayoutType): boolean
+setColor(color: Color, layout: LayoutType): void
+getClassColor(layout: LayoutType): Color
+setX(x: double, def: boolean, layout: LayoutType): void
+setY(y: double, layout: LayoutType): void
+getX(layout: LayoutType): double
+getY(layout: LayoutType): double
+getWidth(layout: LayoutType): double
+getHeight(layout: LayoutType): double
+setWidth(w: double, layout: LayoutType): void
+setHeight(h: double, layout: LayoutType): void
+findEdgeBetweenSource(s: LayeredGraphNode, target: LayeredGraphNode): LayeredGraphEdge
+toString(): String
  
```

```

CombinedLayoutInfo
+x: double
+y: double
+w: double
+h: double
+color: Color
+selected: boolean
+view: NodeView
  
```

```

LayoutInfo
+x: double
+y: double
+w: double
+h: double
+color: Color
+selected: boolean
+xUnDef: boolean
+shift: double
+align: LayeredGraphNode
+root: LayeredGraphNode
+sink: LayeredGraphNode
+view: NodeView
  
```

```

LayeredGraphEdge
+sources: ArrayList<LayeredGraphNode>
+targets: ArrayList<LayeredGraphNode>
+sortedContainedNodes: ArrayList<LayeredGraphNode>
+sortedIncomingEdges: ArrayList<LayeredGraphEdge>
+sortedOutgoingEdges: ArrayList<LayeredGraphEdge>
+containedEdges: ArrayList<LayeredGraphEdge>
+containedNodes: ArrayList<LayeredGraphNode>
+mouseOver: boolean
  
```

