Graph class

Overview: A representation of a directed labeled multigraph. Object will contain a list of nodes and a list of edges mapping between nodes.

Abstraction Function: A graph is a list of nodes (represented as strings) and the connecting edges between them

Rep invariant: if node count == 0, edge count == 0

\* private data fields

\* ArrayList of edges

\* ArrayList of nodes (Strings)

\* constructor

Graph()

@modifies edges, nodes

@effects instantiates both lists (empty)

\* methods

\* getChildren(String parent)

@requires parent != null

@param parent node whose children we're finding

@return ArrayList of children Nodes

\* getParents(String child)

@requires child != null

@param child node whose parents we're finding

@return ArrayList of parent nodes (Strings)

\* listChildren(String parent)

@requires parent != null

@param parent node whose children we're finding

@return ListIterator<String> on an ArrayList of each child\_node(edge\_label) of parent in

alphabetical (lexicographical) order

\* listNodes()

@return ListIterator<String> on an ArrayList of each node in alphabetical (lexicographical) order

\* clear()

@modifies list of nodes and list of edges

@effects sets them both to empty them to empty

\* addNode(String node)

@requires node != null

@param node we are adding

@modifies list of nodes

@effects we add node to the list, if it's not

already inside the list

@return false if node was already in the list, true if node was not and we just added it

\* addEdge(String s, String r, String l)

@requires s!= null, r != null, and nodes contains both s and r

@param source node s, receiver node r, label l

@modifies list of edges

@effects add an edge to the list of edges, if it doesn't already contain that edge and both the nodes exist in the graph

@return true if the edge is added or if it already

exists, false if one of the nodes don't exist in the

graph, and thus, we couldn't add the edge

\* contains(String node)

@requires node != null

@param node; node were searching for

@return true of node exists in list of nodes; false if not

\* hasChild(String parent, String child)

@requires parent and child != null

@param parent, child; supposed parent and child whose relationship we're confirming

@return true if child param is a child of parent param, false otherwise

\* checkRep()

@throws a RuntimeException if the rep. invariant is violated

Edge class

Overview: A representation of an edge in a directed labeled multigraph. Contains a source, a receiver node, and a label for the edge.

Abstraction Function: An Edge maps from start to end and includes label, which contains a bit of info on the Edge

Rep Invariant: start and end nodes != null

\* private data fields

\* String start (source node)

\* Sting end (receiver node)

\* String label (contains some info about edge)

\* constructors

\* Edge(String s, String r, String l)

@requires s != null and r != null

@param source node, receiver node, and label

@modifies start, end, label

@effects start = s, end = r, label = l

\* methods

\* getStart()

@return start

\* getEnd()

@return end

\* getLabel()

@return label

\* equals(Edge edge)

@param edge that is being compared to this object

@return true if they hold all the same info, else false

\* checkRep()

@throws a RuntimeException if the rep. invariant is violated

SortbyChild Class

Overview: Defines how to compare Edge objects. Only guaranteed to work when sorting for listChildren().

\* method

\* compare(Edge a, Edge b)

@requires a and b != null

@param Edge objects a and b

@return -1 if a < b; 1 if a > b