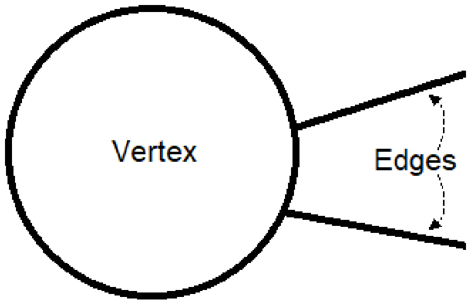
The ADT Graph

ADT Vertex		
Representation:  Vertex = {value = <Object>, edgeList = <List>}		
{inv: value ≠ NIL, edgeList.size ≥ 0 }		
Primitive Operations:		
createVertex	Value	➔ Vertex
addEdge	Vertex x Edge	➔ Vertex
removeEdge	Vertex x Edge	➔ Vertex
getValue	Vertex	➔ Value
getEdges	Vertex	➔ List
isAdjacent	Vertex x Vertex	➔ Boolean

createVertex(val)
“Creates a new Vertex, with its given value.”
{pre: TRUE}
{post: vertex={val, edgeList} }

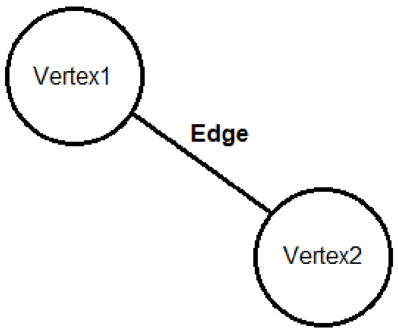
addEdge(vert, edg)
“Connects this vertex to a new edge.”
{pre: vert ≠ NIL, edg ∈ Edge, (edg.vertex1 = NIL ∧ edg.vertex2 ≠ NIL) ∨ (edg.vertex1 ≠ NIL ∧ edg.vertex2 = NIL)}
{post: edg ∈ vert.edgeList}

removeEdge(vert, edg)
“Disconnects this vertex from an edge.”
{pre: vert ≠ NIL, edg ∈ vert.edgeList, edg.vertex1 = vert v edg.vertex2 = vert}
{post: edg ∉ vert.edgeList}

getValue (vert)
“Returns the value of this Vertex”
{pre: vert ≠ NIL}
{post: <value>}

getEdges (vert)
“Returns all of the edges this vertex is connected to.”
{pre: vert ≠ NIL}
{post: <edgeList>}

isAdjacent(vert1, vert2)
“Determines whether a pair of vertexes are adjacent or not.”
{pre: vert1 ≠ NIL, vert1.edgeList.size > 0, vert2 ≠ NIL, vert2.edgeList.size > 0}
{post: FALSE if (edg.vert1 = vert2 or edg.vert2 = vert2) and edg ∈ vert1.edgeList; TRUE otherwise}

ADT Edge			
Representation:  Edge = {Vertex1 = <Vertex>, Vertex2 = <Vertex>, Weight = <Integer>, Directed = <Boolean>}			
{inv: Vertex1 ≠ NIL, Vertex2 ≠ NIL, Weight ≥ 0 }			
Primitive Operations:			
createEdge	Vertex x Vertex x Integer x Boolean	➔	Edge
isWeighted	Edge	➔	Boolean
getWeight	Edge	➔	Integer
isDirected	Edge	➔	Boolean
getVertex1	Edge	➔	Vertex
getVertex2	Edge	➔	Vertex

createEdge(v1,v2, w, d)
"Creates a new Edge and connects two vertexes to it. Also determines its weight and if its either directed or not."
{pre: TRUE}
{post: edge={v1, v2, w, d}

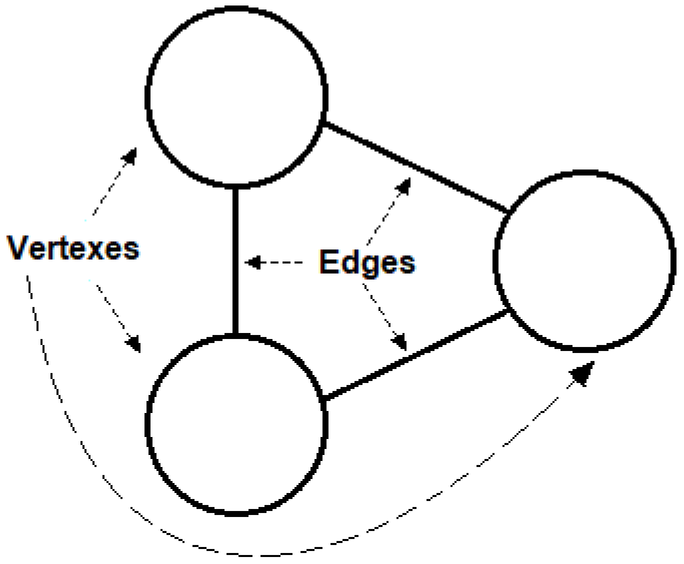
isWeighted(ed)
"Determines whether an edge is weighted or not."
{pre: ed ≠ NIL}
{post: TRUE if ed.Weight >0; FALSE otherwise}

getWeight (ed)
"Determines the weight of this edge."
{pre: ed ≠ NIL}
{post: <Weight>}

isDirected(ed)
"Determines whether an edge is directed or not, in which case it'll be directed from ed.Vertex1 to ed.Vertex2"
{pre: ed ≠ NIL}
{post: <Directed>}

getVertex1(ed)
“Returns the first vertex this edge is connected to.”
{pre: ed ≠ NIL}
{post: <Vertex1>}

getVertex2(ed)
“Returns the second vertex this edge is connected to.”
{pre: ed ≠ NIL}
{post: <Vertex2>}

ADT Graph		
Representation:  Graph = {V, E}, where V is a set of Vertexes and E is a set of Edges		
{inv: V.size ≥ 0, E.size ≥ 0}		
Primitive Operations:		
createGraph		➔ Graph
isWeighted	Graph	➔ Boolean
isDirected	Graph	➔ Boolean
isRelated	Graph	➔ Boolean
addVertex	Graph x Vertex	➔ Graph
addEdge	Graph x Edge	➔ Graph
removeVertex	Graph x Vertex	➔ Graph
removeEdge	Graph x Edge	➔ Graph
getNumberOfEdges	Graph	➔ Integer
getNumberOfVertexes	Graph	➔ Integer
areConnected	Graph x Vertex x Vertex	➔ Boolean
getWeightMatrix	Graph	➔ $A = \{a_{ij}\}$
getDirectionMatrix	Graph	➔ $A = \{a_{ij}\}$
DFS	Graph	➔ List<Vertex>
BFS	Graph	➔ List<Vertex>
Dijkstra	Graph	➔ List<Edge>
Floyd-Warshall	Graph	➔ $A = \{a_{ij}\}$
Prim	Graph	➔ Graph

createGraph()
"Creates a new Graph and initializes its components."
{pre: TRUE}
{post: graph={V, E}, V = {}, E = {} }

isWeighted(gr)
“Determines whether a Graph is directed or not.”
{pre: TRUE}
{post: TRUE if }