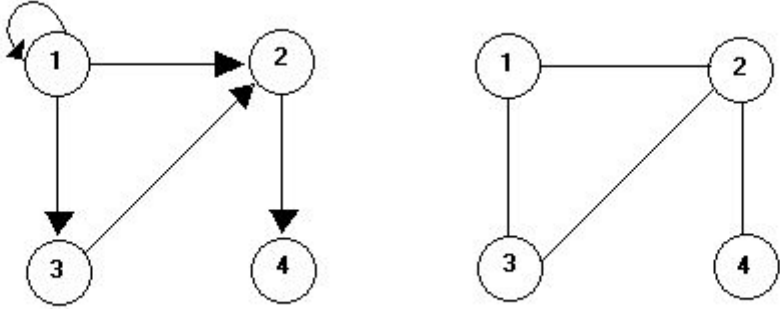


TAD de Grafos	
	
OPERACIONES:	
Creación del grafo	crearGrafo(grafo) → Grafo (Constructora)
Inserción de vértices	agregarVertice(grafo, referenciaVertice) → Grafo (Modificadora)
Eliminación de vértices	eliminarVertice(grafo, referenciaVertice) → --- (Destructor)
Inserción de aristas	agregarArista(grafo, verticeUno, verticeDos) → Grafo (Modificadora)
Eliminación de aristas	eliminarArista(grafo, arista) → --- (Destructor)
Adyacencia	esAdyacente(Grafo, VerticeUno, VerticeDos) → boolean (Consultora)

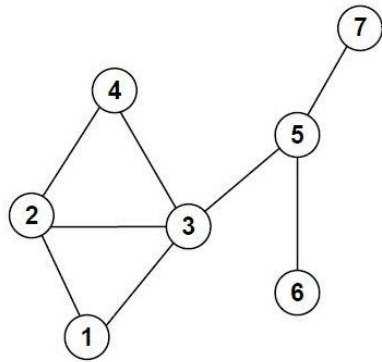
Para este TAD se usaron las siguientes referencias para los métodos reales en nuestro código de lenguaje Java:

- crearGrafo para el método Graph()
- agregarVertice para el método addVertex()
- eliminarVertice para el método removeVertex()
- agregarArista para el método addEdge()
- eliminarArista para el método removeEdge()
- esAdyacente para el método isAdjacent()

crearGrafo(): “Crea un nuevo grafo”

{ pre: }

{post: Grafo=

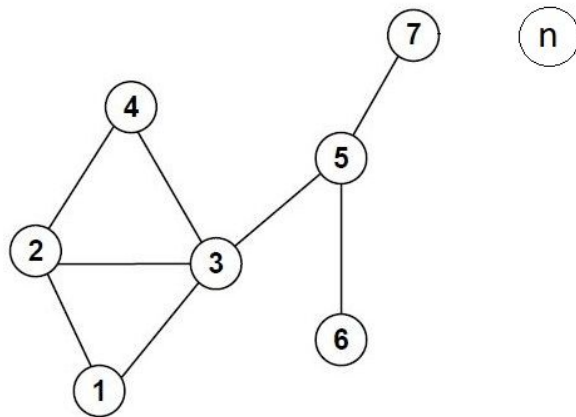


}

agregarVertice(Grafo, referenciaVertice): “Se agrega un vértice al grafo anteriormente creado”

{pre: El grafo debe estar creado, n }

{post: Grafo=

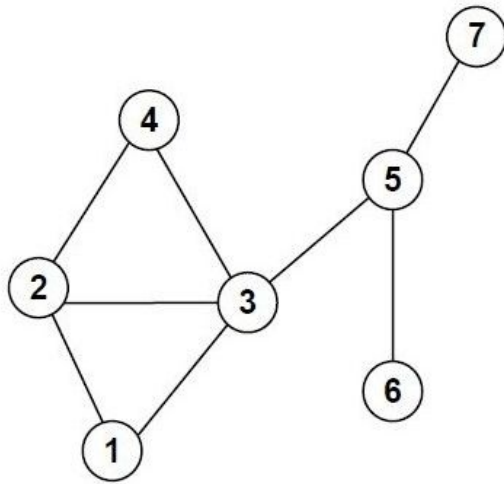


}

eliminarVertice(Grafo, referenciaVertice): "Se elimina un vértice del grafo"

{pre: la referencia del vértice a eliminar se encuentra dentro del grafo(n)

{post:

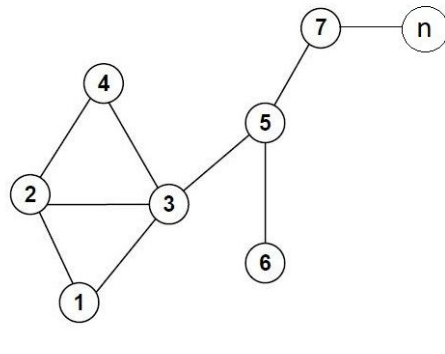


}

agregarArista(Grafo, verticeUno, verticeDos): "Se agrega una arista entre dos vértices"

{pre: El grafo debe de estar creado, 7,n}

{post:

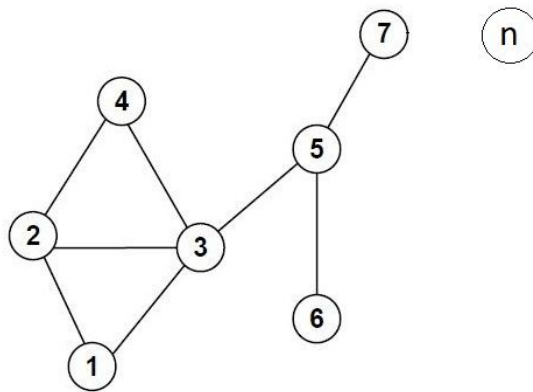


}

eliminarArista(Grafo, arista):"Se elimina una arista"

{pre: la arista a eliminar debe existir dentro del grafo}

{post:



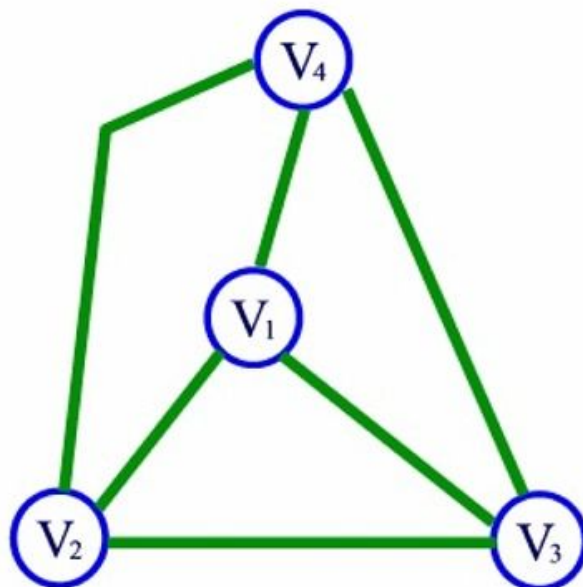
}

esAdyacente():"Permite saber si un grafo es adyacente"

{pre: El árbol debe de estar creado }

{post: booleano, que es True si el grafo es adyacente y False en caso contrario}

TAD de Vértice



OPERACIONES:

Dar Valor

darValor()→Cadena de Texto
(Consultora)

Dar Arista

darArista()→Arista (Consultora)

Para este TAD se usaron las siguientes referencias para los métodos reales en nuestro código de lenguaje Java:

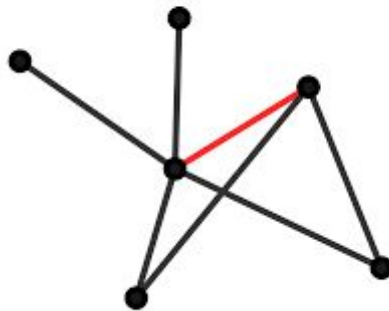
- darValor para el método getValue()
- darArista para el método getEdge()

darValor(): "Da el valor del vertice"
{pre: Grafo debe estar creado, y el vértice debe estar presente dentro del grafo}
{post: Valor del vértice, el cual es una cadena de texto}

darArista(): "Da la arista perteneciente a un vertice"
{pre: Grafo debe estar creado, el vértice y la arista deben estar presentes dentro del grafo}
{post: Arista }

TAD de Arista

Arista



OPERACIONES:

Es Dirigible	esDirigible() \rightarrow Boolean(Consultora)
Dar Valor	darValor() \rightarrow Cadena de Texto(Consultora)
Dar Costo	darCosto() \rightarrow Double(Consultora)
Dar Origen	darOrigen() \rightarrow Vertice(Consultora)
Dar Destino	darDestino() \rightarrow Vertice(Consultora)

Para este TAD se usaron las siguientes referencias para los métodos reales en nuestro código de lenguaje Java:

- esDirigible para el método isDirected()
- darValor para el método getValue()
- darCosto para el método getCost()
- darOrigen para el método getOrigin()
- darDestino para el método getDestination()

esDirigible(): "Nos permite saber si la arista seleccionada es dirigible o no"
--

{pre: Grafo debe estar creado, y la arista que se desea saber si es dirigible debe estar presente dentro del grafo}

{post: booleano que retorna True si la arista es dirigible y False en caso contrario}

darValor(): "Da el valor de la arista"
--

{pre: Grafo debe estar creado, y la arista debe estar presente dentro del grafo}
--

{post: Valor de la arista, el cual es una cadena de texto}
--

darCosto(): "Da el costo de la arista"
--

{pre: Grafo debe estar creado, y la arista a la cual se le desea calcular el costo debe estar presente dentro del grafo}

{post: Costo de la arista, el cual es un valor de tipo double}

darOrigen(): "Da el vértice de origen de la arista"

{pre: Grafo debe estar creado, y la arista debe estar presente dentro del grafo}

{post: Vertice, el cual es el origen de dicha arista}

darDestino(): "Da el vértice de destino de la arista"

{pre: Grafo debe estar creado, y la arista debe estar presente dentro del grafo}

{post: Vértice que hace referencia al destino de la arista}