**Grafo TAD**

| **TAD** Grafo |
| --- |
| Grafo = {Vertex = <Vertex>, Edge = <Edge>} |
| {**inv:** Debe tener un vértice de partida y otro vértice de llegada} |
| **Primitive operations:**   | **Methods** | **Operation Type** | **Input** | **Output** | | --- | --- | --- | --- | | Grafo | Constructor |  | →Grafo | | createVertex | Constructor | T(start) |  | | createEdge | Modifier | Vertex X Vertex X weight |  | | adjacent | Analyzer | T(value) | →ArrayList<Vertex> | | getVertex | Analyzer | T(value) | → MyLinkedList<T> | | getEdge | Analyzer | weight | → Edge | | isEmpty | Analyzer |  | → Boolean | | deleteVertex | Modifier | T(value) | → Vertex | | deleteEdge | Modifier | weight |  | |

| **Grafo ():** |
| --- |
| \*Create an empty Grafo. \* |
| {pre:} |
| {post: Grafo was created} |

| **createVertex (value):** |
| --- |
| \*Create a new Vertex. \* |
| {pre: TRUE} |
| {post: Vertex created} |

| **createEdge (vertex1, vertex2):** |
| --- |
| \*Create an Edge between two vertices \* |
| {pre: Vertex1 must to exist}  {pre: Vertex2 must to exist} |
| {post: Edge between two vertices created} |

| **adjacent (vertex, grafo):** |
| --- |
| \*It’s a method to find the adjacent vertices of one vertex \* |
| {pre: Vertex must to exist}  {pre: It must exist an edge between the adjacent vertices and the original vertex} |
| {post: An ArrayList with the adjacent vertices was returned} |

| **getVertex (vertex, value):** |
| --- |
| \*This method allows to get a Vertex from the Grafo without deleting it. The vertex is obtained from the Grafo as long as it exits \* |
| {pre: Grafo must to be initialized}  {pre: vertex must to exist (It must be different from null because we need to compare somehow)}  {pre: vertex must to exist in the Grafo} |
| {post: Returns the Vertex to get} |

| **getEdge (edge, value):** |
| --- |
| \*This method allows to get an Edge from the Grafo without deleting it. The edge is obtained from the Grafo as long as it exits. \* |
| {pre: Grafo must to be initialized}  {pre: edge must to exist (It must be different from null because we need to compare somehow)}  {pre: edge must to exist in the Grafo} |
| {post: Return the Edge to get} |

| **isEmpty ():** |
| --- |
| \*Verify if the Grado has elements (Vertices and Edges) or not. \* |
| {pre: Grafo must to exist} |
| {post: True if it hasn’t elements == null            False otherwise} |

| **deleteVertex (vertex, value):** |
| --- |
| \*It’s a method to delete a vertex. \* |
| {pre: Grafo must to be initialized}  {pre: vertex must to exist (It must be different from null because we need to compare somehow)}  {pre: vertex must to exist in the Grafo} |
| {post: Vertex returned and removed} |

| **deleteEdge (vertex1, vertex2):** |
| --- |
| \*It’s a method to delete an edge between two vertices. \* |
| {pre: Grafo must to be initialized}  {pre: Vertex1 must to exist}  {pre: Vertex2 must to exist}  {pre: Vertex1 must to exist in the Grafo}  {pre: Vertex2 must to exist in the Grafo} |
| {post: Edge between two vertices returned and removed} |

<https://www6.uniovi.es/usr/cesar/Uned/EDA/Apuntes/TAD_apUM_07.pdf>

<http://informatica.utem.cl/~mcast/ESDATOS/GRAFOS/grafo.pdf>

<https://es.wikipedia.org/wiki/Grafo_(tipo_de_dato_abstracto)>

<https://www.fceia.unr.edu.ar/estruc/2005/graffund.htm>