|  |  |
| --- | --- |
| 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) → --- (Destructora) |
| Inserción de aristas | agregarArista(grafo,verticeUno,verticeDos) →Grafo (Modificadora) |
| Eliminación de aristas | eliminarArista(grafo, arista) → --- (Destructora) |
| Adyacencia | esAdyacente(Grafo,VerticeUno,VerticeDos)→boolean(Consultora) |

|  |
| --- |
| 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) |

|  |
| --- |
| 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 | |
|  | |
| OPERACIONES: | |
| Es Dirigible | esDirigible()→Boolean(Consultora) |
| Dar Valor | darValor()→Cadena de Texto(Consultora) |
| Dar Costo | darCosto()→Double(Consultora) |
| Dar Origen | darOrigen()→Vertice(Consultora) |
| Dar Destino | darDestino()→Vertice(Consultora) |

|  |
| --- |
| 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} |