TAD of Graph - Integrative task two.

TAD <Graph>
Abstract object: A graph is a data structure consisting of a set of nodes or vertices, connected to each other by edges or arcs. Each vertex can have a label or value associated with it, and each edge can have additional attributes such as weight or direction. The graph can be directed, where the edges have a specified direction, or undirected, where the edges do not have a specified direction.

{
 inv:
 	
 • There must be no duplicate vértices. Each vertex in the graph must be unique, that is, there cannot be two vertices with the same id or label.
 	
 • There must be no duplicate edges. Each edge in the graph must be unique, which means that no two edges can connect exactly the same vertices.
 	
 • All vertices and edges of the graph must be connected. This means that any vertex in the graph must be connected through edges to other vertices.
 	
 }

Primitive operations:
 	
 • addVertex: (E element) → void
  - Modifier operation: Because it modifies the structure of the graph to add a new vertex.
 	
 • addEdge: (E source, E destination, double weight) → void
  - Modifier operation: Because it modifies the structure of the graph to add a new edge.
 	
 • deleteVertex: (E element) → void
  - Modifier operation: Because it modifies the structure of the graph to delete a vertex.
 	
 • searchVertex: (E element) → Vertex<E>
  - Parser operation: because it only looks for the vertex in the graph and returns it.
 	
 • searchEdge: (E source, E destination) → Double
  - Parser operation: because it only looks for the edge in the graph and returns the weight of the Edge.

- deleteEdge: (E source, E destination) → void
  - Modifier operation: Because it modifies the structure of the graph to delete an edge.