TAD GRAFO	Primitive Operations:
object abstract: Grafo Grafo: {<ArrayList> = vertices <HashMap> = vertixes int time= 0 int white= 1 int grey = 2 int black = 3} invariant: un grafo se compone de vertices y aristas, es un conjunto de estas mismas. no vacio	addVertix: value T, key int ---> void addAdyacent: vertex int, father int --->void addEdge: keyFrom int,keyTo int,weight int --->void deleteVertix key int --->void deleteAllRef: key int --->void BFS: keyRoot int --->void DFS: --->void dfsVisit: --->void getHashSize: --->int proveConec: --->int Dijkstra: source int ---> String Floyd-Warshall: graph Grafo[][]--->String Prim: graph Grafo ---> String Kruskal: graph Grafo --->String

Kruskal{Grafo graph} ---> String

"find the minimum spanning tree from a graph "

{pre : Graph ≠ null}

{pos : minimum spanning tree}

addVertix{T value, int Key} ---> void

"Creates an especific Vertex and add it into the vertex ArrayList"

{pre : The vertex to add is not into the vertexes ArrayList"}

{pos : Vertex added}

deleteVertix{int Key} ---> void

"Deletes the Vertex with the specific key from the vertex ArrayList"

{pre : The vertex to delete is into the vertexes ArrayList"}

{pos : Vertex deleted}

addEdge{int KeyFrom, int keyTo,int peso } ---> void

"Add a certain edge"

{pre : "the vertexes connected by the edge exist at the vertexes ArrayList"}

{pos : true}

deleteAllRef{int Key} ---> void

"Deletes all vertexes"

{pre : none"}

{pos : Vertex ArrayList == null}

BFS{int KeyRoot} ---> void

"Verify connectivity from the root vertex to its neighbors"

{pre : Graph ≠ null"}

{pos : BF tree}

Dijkstra{int source} ---> String

"Returns the with less weight from the source to a certain vertex"

{pre : Graph ≠ null"}

{pos : path with less weight}

DFS{} ---> void

"Cover all the graph vertexes"

{pre : Graph ≠ null"}

{pos : DF forest}

gertHashSize{} ---> int

"Returns the vertexes Array Size"

floyd-Warshall{Grafo[][] graph} ---> String

"Find the shortest path between all the pairs of vertexes in a weighted graph"

{pre: Graph ≠ null}
{pos: shortest path between all the pairs of vertexes}

provConex{} ---> int

"Check if the graph is strongly connected"

{pre : edge ≠ null, vertex ≠ null"}

{pos : true}

addAdyacent{int vertex,int father} ---> void

"Add to the vertex father and adjacent vertex"

prim{Grafo graph} ---> String

"Find the minimum spanning tree from a graph "

{pre: Graph ≠ null
{pos: minimum spanning tree}