ADT Graph

g = <<v1, e1>,<v2, e2>,<vn, en>>

{inv: <g ≠ Ø>,<∀ei,vn,vm((ei, vn) ∧ (ei, vm))⇒vn∈ei ∧ vm∈ei>}

Primitive Operations:

• addVertex	GraphXT	->Graph
• addConnection	Graph	->Graph
• addValue	GraphXv	->Graph
• search	GraphXv	->T
• BFS	Graph	->Graph
• dijkstraPath	Graph	->pathGraph
• checkConexivity	Graph	->Graph
• floydWarshall	Graph	->pathGraph
• prim	Graph	->pathGraph
• kruskal	Graph	->pathGraph
• containerOf	Graph	.->T
• clear	Graph	->null
• searchVertex	GraphXT	->T

addVertex(Graph, I id, T toAdd) ---> void

"Inserts a vertex to the graph"

{ pre: Graph = Elements ∧ k ∉ Elements }

{ pos: Graph = Elements ∧ k ∈ Elements }

addConnection(Graph, I pointer, I pointed, int weight) ---> void
"Creates an edge in the graph"

{ pre: Graph = Elements ∧ k ∉ Elements }

{ pos: Graph = Elements ∧ k ∈ Elements }

addValue(Graph, I pointer, I pointed, String direction, int weight) --> void

"Adds weight to the vertex"

{ pre: Graph = ∧ k ∉ Elements }

{ pos: Graph = Elements ∧ k ∈ Elements }

search(Graph, I id) --> T

"Search a vertex"

{ pos:Graph = T}

BFS (Graph, I s) --> void

"Verify if the board is conected"

{ pre: Graph ≠ null }

{ pos: ∀ vertex that has a path to s, color=2(black) }

DFS (Graph, I s) --> void

"Verify if the board is conected"

{ pre: Graph ≠ null }

{ pos: ∀ vertex that has a path to s, color=2(black) }

dijsktraPath(Graph,I startID, I endID) --> Stack

"Draws the path of the enemy to the player"

{ pre: Graph ≠ null }

{ pos: pathGraph}

checkConexivity(Graph, I start) --> Boolean

"Checks the connection between the edge and the vertex"

{ pre: false, edge ≠ null, vertex ≠ null }

{ pos:true}

floydWarshall(Graph) --> Hashmap

"Draws the path of the enemy to chase the player"

{ pre: Graph ≠ null }

{ pos: pathGraph}

prim(Graph, I rs) --> int

"Finds the minimun coater tree"

kruskal(Graph, I rs) --> TreeSet

"Finds the minimun coater tree"

containerOf(Graph, T value) --> Object

"Returns the vertex"

{ pre: Graph ≠ null }

{ pos:returns vertex=T}

clear(Graph, T value) --> void

"clears the current scenario"

{ pre: Graph ≠ null }

{ pos:Graph = null}

searchVertex(Graph, I id) --> Object

"Search a vertex"

{ pos:Graph = T}