TESTING DOCUMENTATION

AdjacencyList
Scenarios Set Up:

Name	Class	Scenario
setupScenary1	GraphAdjacencyList	New graph -> String Add nodes: U, V, X, Y, Z Add vértices: (U, X), (X, Y), (Y, Z), (Z, Y), (Y, V), (V, U)
SetupScenary2	GraphAdjacencyList	New graph -> Integer No elements added

Tests:

Test Objective: Verify if the graph is strongly connected				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
GraphAdjacencyList	strongConnected1()	setupScenary1	New graph	True (Graph is strongly connected).
GraphAdjacencyList	strongConnected2()	SetupScenary2	New graph with integers	True (Graph is strongly connected).

Test Objective: Check if BFS works properly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
GraphAdjacencyList	Bfs1()	setupScenary1	Graph: Add nodes: A, B, C, D. Add edges: (A, B), (A, C), (C, B), (B, D), (D, A).	True (BFS from vertex A covers all nodes).
GraphAdjacencyList	Bfs1()	SetupScenary2	Graph: Add vertices: 1, 2, 3, 4. Add edges: (1, 2), (1, 3), (3, 2), (2, 4).	True (BFS from vertex 1 covers all vertices).

Test Objective: Check if DFS works properly

Class	Method	Scenario	Entry Values	Result
GraphAdjacencyList	dfs1	setupScenary1	Graph: Add nodes: A, B, C, D. Add edges: (A, B), (A, C), (C, B), (B, D), (D, A).	1 (Number of connected components).
GraphAdjacencyList	Dfs2	setupScenary2	Graph: Add vertices: 1, 2, 3, 4. Add edges: (1, 2), (1, 3), (3, 2), (2, 4).	1 (Number of connected components).

Test Objective: Check if Dijkstra algorithm works properly

Class	Method	Scenario	Entry Values	Result
GraphAdjacencyList	dijkstra1	setupScenary1	Graph: Add nodes: A, B, C, D. Add edges: (A, B), (A, C), (C, B), (B, D), (D, A).	1 in the position (0,1) since “A” is connected to “B” Matrix with correct distances.
GraphAdjacencyList	Dijkstra2	setupScenary2	Graph: Add vertices: 1, 2, 3, 4. Add edges: (1, 2), (1, 3), (3, 2), (2, 4).	1 in the position (0,1) since 1 is connected to 2 Matrix with correct distances.

Test Objective: Check if floydWarshall algorithm works properly

Class	Method	Scenario	Entry Values	Result

GraphAdjacencyList	Floyd1()	setupScenary1	Graph: Add nodes: A, B, C, D. Add edges: (A, B), (A, C), (C, B), (B, D), (D, A).	1 in the position (0,2) since “A” is connected to “C”
GraphAdjacencyList	Floyd2()	setupScenary2	Graph: Add vertices: 1, 2, 3, 4. Add edges: (1, 2), (1, 3), (3, 2), (2, 4).	1 in the position (2,3) since 3 is connected to 4

Adjacency Matrix

Scenarios Set Up:

Name	Class	Scenario
setUp1	Adjacency Matrix	New adjacency matrix • No elements added

setUp2	Adjacency Matrix	graph.addVertex("V0");
		graph.addVertex("V1");
		graph.addVertex("V2");
		graph.addEdge("V0", "V1");

setUp3	Adjacency Matrix	graph.addVertex("V0");
		graph.addVertex("V1");
		graph.addVertex("V2");
		graph.addEdge("V0", "V1");
		graph.addEdge("V1", "V2");

Tests:

Test Objective: Check the correct insertion of edges

Class	Method	Scenario	Entry Values	Result
Adjacency Matrix	testAddEdge1	setUp1	"V0", "V1"	The matrix should not be null
Adjacency Matrix	testAddEdge2	setUp2	"V0", "V1"	The matrix should not be null
Adjacency Matrix	testAddEdge3	setUp3	"V1", "V2"	The matrix should not be null

Test Objective: Check if DFS method works properly

Class	Method	Scenario	Entry Values	Result

Adjacency Matrix	testDfs1()	setUp1	Empty graph	Throws NullPointerException (DFS not possible in an empty graph).
Adjacency Matrix	testDfs2()	setUp2	graph with vertices V0, V1, and an edge between V0 and V1.	Size of result is 2 (Two vertices visited in DFS).
Adjacency Matrix	testDfs3()	setUp3	graph with vertices V0, V1, V2, and edges between V0-V1 and V1-V2.	Size of result is 3 (Three vertices visited in DFS).

Test Objective: Check if Dijkstra’s algorithm works properly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
Adjacency Matrix	testDijkstraSingleNode()	setUp1	graph with a single node V0.	Distance from V0 to V0 is 0.
Adjacency Matrix	testDijkstra2()	setUp2	graph with vertices V0, V1, and an edge between V0 and V1.	Size of result is 3 (Distances calculated for three vertices).
Adjacency Matrix	testDijkstra3()	setUp3	graph with vertices V0, V1, V2, and edges between V0-V1 and V1-V2.	Size of result is 3 (Distances calculated for three vertices).