Scenario Configuration

Name	Class	Scenery
setupStage1	GraphsTest	An empty object of the MatrixGraph and another of the AdjacentGraphList, both graphs are undirected and allow loops.
setupStage2	GraphsTest	An empty object of the MatrixGraph and another of the AdjacentGraphList, both graphs are directed and allow loops.
setupStage3	GraphsTest	An empty object of the MatrixGraph and another of the AdjacentGraphList, both graphs are undirected and don't allow loops.

Test Case Design

Test objective: Verify that vertices are added correctly and that exceptions are generated correctly.				
Class	**Method**	**Scenery**	**Input values**	**Result**
MatrixGraph & AdjacentGraphList	addVertex	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”}	List of vertices: "A”: “B”, “C”
MatrixGraph & AdjacentGraphList	addVertex	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“ ”, “C”}	Exception indicating that there can´t be a vertex with a null key.

Test objective: Verify that edges are added correctly and that exceptions are

generated correctly.				
Class	**Method**	**Scenery**	**Input values**	**Result**
MatrixGraph & AdjacentGraphList	addEdge	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} vertex{“A”, “B”, 1} vertex{“A”, “C”, 1}	List of adjacent vertices from “A”: “B”, “C”
MatrixGraph & AdjacentGraphList	addEdge	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} vertex{“A”, “B”, 1} vertex{“A”, “C”, 1} vertex{“B”, “C”, 1} vertex{“A”, “C”, 1}	Exception indicating that the edge already exists.
MatrixGraph & AdjacentGraphList	addEdge	setupStage 3	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} vertex{“A”, “B”, 1} vertex{“A”, “C”, 1} vertex{“B”, “C”, 1} vertex{“B”, “B”, 1}	Exception indicating that the graph doesn't allow loops.

Test objective: Verify that the method BSF works correctly and shows the correct reachable nodes.				
Class	**Method**	**Scenery**	**Input values**	**Result**
MatrixGraph & AdjacentGraphList	bsf	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} node{“D”, “D”} node{“E”, “E”} Edge{“A”, ”B”, 1} Edge{“B”, ”C”, 1} Edge{“C”, ”D”, 1} Edge{“D”, ”E”, 1}	Nodes reachable from "A”: “B”, “C”, “D”, “E”
MatrixGraph & Adjac	bsf	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} node{“D”, “D”}	Nodes reachable from "B”: “ ”

entGraphList				
MatrixGraph & AdjacentGraphList	bsf	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} node{“D”, “D”} node{“E”, “E”} node{“F”, “F”} node{“G”, “G”} node{“H”, “H”} node{“I”, “I”} node{“J”, “J”} node{“K”, “K”} Edge{“A”, ”A”, 1} Edge{“A”, ”B”, 1} Edge{“A”, ”C”, 1} Edge{“B”, ”C”, 1} Edge{“B”, ”D”, 1} Edge{“C”, ”D”, 1} Edge{“D”, ”J”, 1} Edge{“J”, ”E”, 1} Edge{“J”, ”H”, 1}	Nodes reachable from "D”: “A”, “B”, “C”, “D”, “E”, “J”, “E”, “H”

Test objective: Verify that the Dijkstra method works correctly and shows the minimum path from a node to another.				
Class	**Method**	**Scenery**	**Input values**	**Result**
MatrixGraph & AdjacentGraphList	dijkstra	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} node{“D”, “D”} node{“E”, “E”} node{“F”, “F”} Edge{“A”, ”B”, 1} Edge{“A”, ”C”, 1} Edge{“B”, ”D”, 1} Edge{“B”, ”F”, 1} Edge{“C”, ”D”, 1} Edge{“C”, ”E”, 1} Edge{“D”, ”E”, 1} Edge{“D”, ”F”, 1} Edge{“E”, ”F”, 1}	Shortest path from “A” to “F”: A -> B -> F
MatrixGraph	dijkstra	setupStage 1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”}	Message indicating that

& AdjacentGraphList			node{“D”, “D”} node{“E”, “E”} node{“F”, “F”} Edge{“A”, ”B”, 1} Edge{“A”, ”C”, 1} Edge{“B”, ”D”, 1} Edge{“C”, ”D”, 1} Edge{“D”, ”A”, 1} Edge{“E”, ”F”, 1}	there is no path leading from A to F.
MatrixGraph & AdjacentGraphList	dijkstra	setupStage 1	node{“P”, “P”} node{“Q”, “Q”} node{“R”, “R”} node{“S”, “S”} node{“T”, “T”} Edge{“P”, ”Q”, 2} Edge{“P”, ”R”, 3} Edge{“Q”, ”S”, 4} Edge{“R”, ”S”, 1} Edge{“S”, ”T”, 5}	Shortest path with distance 9, from “P” to ”T”: P -> R -> S -> T

Test objective: Verify that the Prim method works correctly and shows the tree with the minimum paths of all the nodes of the network.				
Class	**Method**	**Scenery**	**Input values**	**Result**
MatrixGraph & AdjacentGraphList	Prim	setupStage1	node{“A”, “A”} node{“B”, “B”} node{“C”, “C”} node{“D”, “D”} Edge{“A”, ”B”, 5} Edge{“A”, ”C”, 3} Edge{“B”, ”C”, 2} Edge{“B”, ”D”, 4} Edge{“C”, ”D”, 1}	Minimum connection tree from “A”: A - C, C - D, C - B
MatrixGraph & AdjacentGraphList	Prim	setupStage2	node{“P”, “P”} node{“Q”, “Q”} node{“R”, “R”} node{“S”, “S”} node{“T”, “T”} Edge{“P”, ”R”, 3} Edge{“Q”, ”R”, 8} Edge{“S”, ”R”, 10} Edge{“T”, ”R”, 7}	Created Graph is directed, “Prim Method does not apply on Directed Graph” message must be displayed
MatrixGraph & AdjacentGr	Prim	setupStage3	node{"A", "A"} node{"B", "B"} node{"C", "C"} node{"D", "D"} node{"E", "E"} node{"F", "F"}	Minimum connection tree from “C”: C - G C - A

aphLi st			node{"G", "G"} node{"H", "H"} node{"I", "I"} node{"J", "J"} node{"K", "K"} node{"L", "L"} node{"M", "M"} node{"N", "N"} node{"O", "O"} Edge{"A", "B", 5} Edge{"A", "C", 2} Edge{"B", "D", 4} Edge{"B", "E", 3} Edge{"C", "F", 6} Edge{"C", "G", 1} Edge{"D", "H", 8} Edge{"D", "I", 9} Edge{"E", "J", 7} Edge{"E", "K", 6} Edge{"F", "L", 3} Edge{"G", "M", 5} Edge{"H", "N", 2} Edge{"I", "O", 4} Edge{"J", "K", 2}	A - B B - E B - D G - M E - K K - J C - F F - L D - H H - N D - I I - O