Configuración de los escenarios:

Nombre	Clase	Escenario
setupStage1	Graph	Un objeto de la clase grafo (de tipo String) con un único vértice de valor “A”.
setupStage2	Graph	Un objeto de la clase grafo con 3 vértices: - el primer vértice con valor “A” - el segundo con valor “B” - el tercero con valor “C”
setupStage3	Graph	Un objeto de la clase grafo con 3 vértices y una arista de peso 1: - el primer vértice con valor “A” - el segundo vértice con valor “B” - el tercero con valor “C” - la arista debe ir de “A” a “B”
setupStage4	Graph	Un objeto de la clase grafo con 4 vértices y 2 aristas: - el primer vértice con valor “A” - el segundo vértice con valor “B” - el tercer vértice con valor “C” - el cuarto vértice con valor “D” - la primer arista debe ir de “A” a “B” con peso 1 - la segunda arista debe ir de “A” a “D” con peso 3
setupStage5	Graph	Un objeto de la clase grafo (de tipo integer) con 4 vértices y 5 aristas: - el primer vértice con valor 1 - el segundo vértice con valor 2 - el tercer vértice con valor 3 - el cuarto vértice con valor 4 - la primer arista debe ir de 2 a 1 con peso 4 - la segunda arista debe ir de 2 a 3 con peso 3 - la tercera arista debe ir de 3 a 4 con peso 2 - la cuarta arista debe ir de 1 a 3 con peso -2 - la quita arista debe ir de 4 a 2 con peso -1
setupStage6	Graph	Se prueba con método de recorrido BFS iniciando por el vértice de valor “1”, si un objeto de la clase grafo (de tipo integer) con 4 vértices y 5 aristas es conexo o no lo es. El grafo es de la siguiente forma: - el primer vértice con valor 1 - el segundo vértice con valor 2

Nombre	Clase	Escenario
		- el tercer vértice con valor 3 - el cuarto vértice con valor 4 - la primer arista debe ir de 2 a 1 con peso 4 - la segunda arista debe ir de 2 a 3 con peso 3 - la tercera arista debe ir de 3 a 4 con peso 2 - la cuarta arista debe ir de 1 a 3 con peso -2 - la quita arista debe ir de 4 a 2 con peso -1
setupStage7	Graph	Se prueba con método de recorrido DFS si un objeto de la clase grafo (de tipo integer) con 4 vértices y 5 aristas es conexo o no lo es. El grafo es de la siguiente forma: - el primer vértice con valor 1 - el segundo vértice con valor 2 - el tercer vértice con valor 3 - el cuarto vértice con valor 4 - la primer arista debe ir de 2 a 1 con peso 4 - la segunda arista debe ir de 2 a 3 con peso 3 - la tercera arista debe ir de 3 a 4 con peso 2 - la cuarta arista debe ir de 1 a 3 con peso -2 - la quita arista debe ir de 4 a 2 con peso -1
setupStage8	Graph	Se hace una búsqueda del vértice con valor 1 dentro del grafo siguiente: - el primer vértice con valor 1 - el segundo vértice con valor 2 - el tercer vértice con valor 3 - el cuarto vértice con valor 4 - la primer arista debe ir de 2 a 1 con peso 4 - la segunda arista debe ir de 2 a 3 con peso 3 - la tercera arista debe ir de 3 a 4 con peso 2 - la cuarta arista debe ir de 1 a 3 con peso -2 - la quita arista debe ir de 4 a 2 con peso -1

Casos de Prueba:

Objetivo de la prueba: Verificar que los métodos Graph (método constructor), addVertex y newEdge funcionan correctamente				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Graph	Graph()	setupStage1.	value = “A”.	graph.vertices = {A} Objeto de la clase Graph con 1 vértice “A”
Graph	addVertex()	setupStage2	value = “B”.	graph.vertices = {A,B} Objeto de la clase Graph con 2 vértices “A”, “B”
Graph	addVertex()	setupStage2	value = “C”.	graph.vertices = {A,B,C} Objeto de la clase Graph con 3 vértices “A”, “B”, “C”
Graph	newEdge()	setupStage3	edge = “A”,”B”,1.	graph.vertices = {A,B,C} graph.edges = {[A,B,1]} Objeto de la clase Graph con 3 vértices “A”, “B”, “C” y una arista con peso 1 que va de “A” a “B”
Graph	newEdge()	setupStage4	edge = “A”,”D”,3.	graph.vertices = {A,B,C,D} graph.edges = {[A,B,1],[A,D,3]} Objeto de la clase Graph con 4 vértices “A”, “B”, “C”, “D” y dos aristas: la primera con peso 1 que va de “A” a “B”; la segunda con peso 3 que va de “A” a “D”
Graph	addVertex() (4 veces)	setupStage5	values: 1,2,3,4,5	graph.vertices = {1,2,3,4} Objeto de la clase graph con 4 vértices 1, 2, 3, 4.
Graph	newEdge() (5 veces)	setupStage5	values: (2,1,4);(2,3,3);(3,4,2);(1,3,-2);(4,2,-1)	graph.vertices = {1,2,3,4} graph.edges = {[2,1,4],[2,3,3],[3,4,2],[1,3,-2],[4,2,-1]}
Graph	BFS()	setupStage6	values: 1	True. El vértice es conexo. Vértice analizado: graph.vertices = {1,2,3,4} graph.edges = {[2,1,4],[2,3,3],[3,4,2],[1,3,-2],[4,2,-1]}
Graph	DFS()	setupStage7	values: null	True. El vértice es conexo. Vértice

Objetivo de la prueba: Verificar que los métodos Graph (método constructor), addVertex y newEdge funcionan correctamente				
				analizado: graph.vertices = {1,2,3,4} graph.edges = {[2,1,4],[2,3,3],[3,4,2],[1,3,-2],[4,2,-1]}
Graph	findVertex()	setupStage8	values: 1	El vértice se encuentra exitósamente