Diseño de Pruebas unitarias.

Especificación de Escenarios de prueba:

Nombre:	Clase:	Especificación:
stage1	AdjacencyMatrixGraph	Grafo **g** no dirigido, no ponderado y vacío es instanciado.
stage2	AdjacencyMatrixGraph	Grafo **g** no dirigido, no ponderado y conexo es instanciado y 5 vértices le son añadidos.
stage3	AdjacencyMatrixGraph	Grafo **g** ponderado, dirigido y con bordes de cualquier valor (evitando ciclos negativos). Tiene 5 vértices.
stage4	AdjacencyMatrixGraph	Grafo **g** ponderado, dirigido, conexo y con bordes positivos. Tiene 5 vértices.
stage5	AdjacencyMatrixGraph	Grafo **g** ponderado, no dirigido y con bordes de cualquier valor.
stage6	AdjacencyListGraph	Grafo **g** no dirigido, no ponderado y vacío es instanciado.
stage7	AdjacencyListGraph	Grafo **g** no dirigido, no ponderado y conexo es instanciado y 5 vértices le son añadidos.
stage8	AdjacencyListGraph	Grafo **g** ponderado, dirigido y con bordes de cualquier valor (evitando ciclos negativos). Tiene 5 vértices.
stage9	AdjacencyListGraph	Grafo **g** ponderado, dirigido, conexo y con bordes positivos. Tiene 5 vértices.
stage10	AdjacencyListGraph	Grafo **g** ponderado, no dirigido y con bordes de cualquier valor.

Diseño de pruebas:

Prueba No 1:	Objetivo: Probar las operaciones de agregar de la interfaz IGraph.			
Clase	**Método**	**Escenario**	**Entradas**	**Salidas**
IGraphTest	addVertex(V u)	stage1	u : V; u no se encuentra en el grafo.	true
IGraphTest	addVertex(V u)	stage2	u : V; u se encuentra en el grafo.	false
IGraphTest	addEdge(V u, V v)	stage2	u : V; v : V; g no ponderado; u no pertenece al grafo.	ElementNotFoundException
IGraphTest	addEdge(V u, V v)	stage2	u : V; v : V; g no ponderado; v no pertenece al grafo.	ElementNotFoundException
IGraphTest	addEdge(V u, V v)	stage2	u : V; v : V; g no ponderado; g no dirigido.	true
IGraphTest	addEdge(V u, V v)	stage5	u : V; v : V; g no ponderado; g dirigido.	true

IGraphTest	addEdge(V u, V v)	stage3	u : V; v : V; u.equals(v) (admite bucles).	true
IGraphTest	addEdge(V u, V v, double w)	stage3	u : V; v : V; w : double; u no pertenece al grafo.	ElementNotFoundException
IGraphTest	addEdge(V u, V v, double w)	stage3	u : V; v : V; w : double; v no pertenece al grafo.	ElementNotFoundException
IGraphTest	addEdge(V u, V v, double w)	stage2	u : V; v : V; w : double; el grafo no es ponderado.	WrongEdgeTypeException
IGraphTest	addEdge(V u, V v, double w)	stage3	u : V; v : V; w : double.	true
IGraphTest	addEdge(V u, V v, double w)	stage3	u : V; v : V; w : double; u.equals(v) (admite bucles).	true

Prueba No 2:	Objetivo: Probar las operaciones de eliminar de la interfaz IGraph.			
Clase	**Método**	**Escenario**	**Entradas**	**Salidas**
IGraphTest	removeVertex(V u)	stage1	u : V; u no se encuentra en el grafo.	ElementNotFoundException
IGraphTest	removeVertex(V u)	stage1	u : V; u se encuentra en el grafo.	true
IGraphTest	removeEdge(V u, V v)	Stage6?	u : V; v : V; g no ponderado; g dirigido.	false
IGraphTest	removeEdge(V u, V v)	stage2	u : V; v : V; u y v no comparten borde, g no dirigido.	false
IGraphTest	removeEdge(V u, V v)	stage2	u : V; v : V; u y v comparten borde, g no dirigido.	true
IGraphTest	removeEdge(V u, V v)	Stage6?	u : V; v : V; existe un borde de u a v, g dirigido.	true
IGraphTest	removeEdge(V u, V v)	stage2	u : V; v : V; u no pertenece al grafo.	ElementNotFoundException
IGraphTest	removeEdge(V u, V v)	stage2	u : V; v : V; v no pertenece al grafo.	ElementNotFoundException

Prueba No 3:	Objetivo: Probar las operaciones de consulta de la interfaz IGraph.			
Clase	**Método**	**Escenario**	**Entradas**	**Salidas**
IGraphTest	vertexAdjacent(V u)	stage4	u : V; u no se encuentra en el grafo.	ElementNotFoundException
IGraphTest	vertexAdjacent(V u)	stage4	u : V; u tiene al menos un vértice adyacente.	adjacentVertices : List<V>
IGraphTest	vertexAdjacent(V u)	stage4	u : V; u no tiene vértices adyacentes.	null
IGraphTest	areConnected(V u, V v)	stage3	u : V; v : V; u no se encuentra en el grafo.	ElementNotFoundException
IGraphTest	areConnected(V u, V v)	stage3	u : V; v : V; v no se encuentra en el grafo.	ElementNotFoundException
IGraphTest	areConnected(V u, V v)	stage3	u : V; v : V; u y v comparten al menos un borde.	true
IGraphTest	areConnected(V u, V v)	stage3	u : V; v : V; u y v no comparten ningún borde.	false
IGraphTest	weightMatrix()	stage1	void	adjacencyMatrix : double[][]
IGraphTest	isDirected()	stage3	void	true
IGraphTest	isDirected()	stage2	void	false
IGraphTest	getIndex(V u)	stage5	u : V; u no se encuentra en el grafo.	ElementNotFoundException
IGraphTest	getIndex(V u)	stage5	u : V; u se encuentra en el grafo.	index : int
IGraphTest	getVertexSize()	stage1	void	0
IGraphTest	getVertexSize()	stage2	void	5
IGraphTest	getVertices()	stage3	void	vertices : Map<V, Integer>
IGraphTest	getEdges()	stage2	void	edges : ArrayList<Edge>

Prueba No 4:	Objetivo: Probar los algoritmos sobre grafos de la clase GraphAlgorithms.			
Clase	**Método**	**Escenario**	**Entradas**	**Salidas**
IGraphTest	bfs(IGraph<V> g, V u)	stage3	g : IGraph; u : V; u no pertenece a g.	ElementNotFoundException
IGraphTest	bfs(IGraph<V> g, V u)	stage3	g : IGraph; u : V; u pertenece a g.	list : List<V>
IGraphTest	dfs(IGraph<V> g, V u)	stage3	g : IGraph; u : V; u no pertenece a g.	ElementNotFoundException

IGraphTest	dfs(IGraph<V> g, V u)	stage3	g : IGraph; u : V; u pertenece a g.	list : List<V>
IGraphTest	traversal(IGraph<V> g, V u, ICollection<V> ds)	stage3	g : IGraph; u : V; u no pertenece a g.	ElementNotFoundException
IGraphTest	traversal(IGraph<V> g, V u, ICollection<V> ds)	stage3	g : IGraph; u : V; u pertenece a g.	list : List<V>
IGraphTest	dijkstra(IGraph<V> g, V s)	stage4	g : IGraph; s : V; s no pertenece a g.	ElementNotFoundException
IGraphTest	dijkstra(IGraph<V> g, V s)	stage4	g : IGraph; s : V; s pertenece a g.	shortestPath : double[][]
IGraphTest	dijkstra(IGraph<V> g, V s)	stage3	g : IGraph; s : V.	WrongEdgeTypeException
IGraphTest	floydWarshall(IGraph<V> g)	stage2	g : IGraph.	WrongGraphTypeException
IGraphTest	floydWarshall(IGraph<V> g)	stage3	g : IGraph.	d : double[][]
IGraphTest	prim(IGraph<V> g, V s)	stage5	g : IGraph; s : V; s no pertenece a g.	ElementNotFoundException
IGraphTest	prim(IGraph<V> g, V s)	stage2	g : IGraph; s : V.	WrongGraphTypeException
IGraphTest	prim(IGraph<V> g, V s)	stage3	g : IGraph; s : V.	WrongGraphTypeException
IGraphTest	kruskal(IGraph<V> g)	stage5	g : IGraph.	WrongGraphTypeException