**PRÁCTICA 32 | GRAFOS**

# Introducción

Una de las formas que había visto también de cómo implementar los grafos era a través de listas de adyacencia. Es grato ver que, en ésta, que se ha dicho es la última práctica, cerremos con eso para tener también el concepto y conocimiento de un grafo que utiliza memoria dinámica.

# Desarrollo

Lo primero fue decidir cuál de los dos algoritmos implementar para cumplir con los requisitos de la práctica. Decidí usar el **algoritmo de Prim**, ya que me pareció más fácil usarlo a comparación del de Kruskal.

Un problema que se presentó fue que se necesitaba antes de todo, que las aristas tuvieran la posibilidad de ser ponderadas con un valor arbitrario por el cliente de la clase. Con esto en mente pensé en varias posibilidades para realizarlo, y me terminé decidiendo por crear una lista extra como atributo de la clase, en donde fueran almacenadas las aristas que contiene el grafo junto con el costo que implica recorrer dicha unión entre vértices. Al hacer esto también tuve que modificar parte del código propuesto en los vídeos de la clase: las modificaciones se concentraron solamente en el método de *agregarArista,* donde además de añadir a la lista de aristas la unión que se realizaba en el método también cree una sobrecarga de la misma función con la posibilidad de agregar un peso específico, imitando la estructura de la clase **GrafoMatrizA** en este aspecto.

Algo más que se necesitaba antes de programar el algoritmo de Prim era asegurar que el grafo fuese conexo, de otro modo el árbol de costos mínimos no podría definir un camino hacia todos los vértices, así que también tuve que agregar algunos métodos para comprobarlo.

Después de atender a todas las dependencias anteriormente mencionadas ahora sí comencé a codificar el funcionamiento del algoritmo de Prim.

En general la práctica no fue muy complicada de realizar, una de las cosas que me suele tomar tiempo es pensar la forma en que deben estructurarse las cosas para que sean óptimas y ofrezcan facilidad de uso; en esta ocasión ese problema fue la manera de ponderar las aristas del grafo.

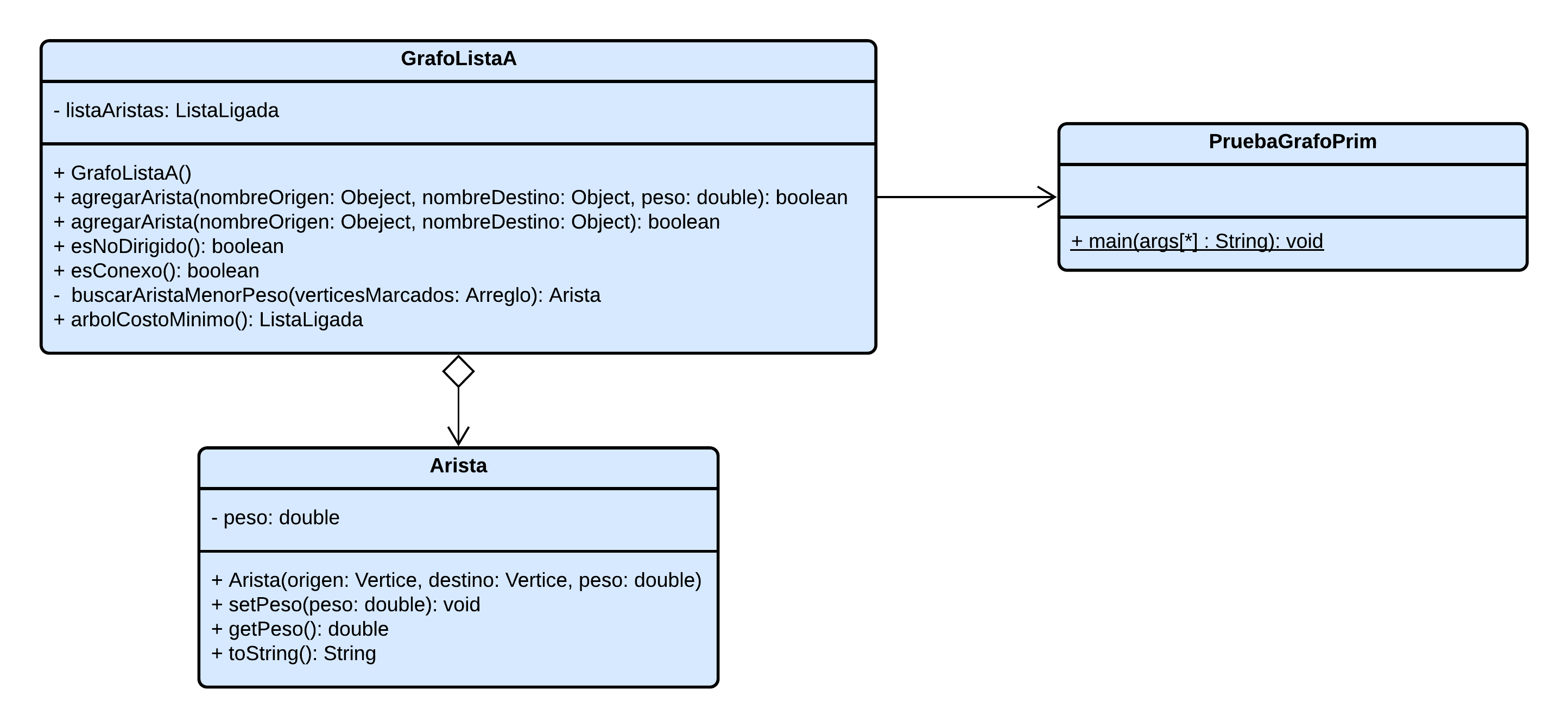
A continuación, se indican los archivos creados o modificados y su ubicación en el proyecto.

* ***GrafoListaA.java (paquete ednolineal).***
* ***Arista.java (paquete catalogos).***
* ***PruebaGrafoPrim.java (paquete interfaces).***

La clase anterior contiene las pruebas de esta práctica.

# Diagrama de clases

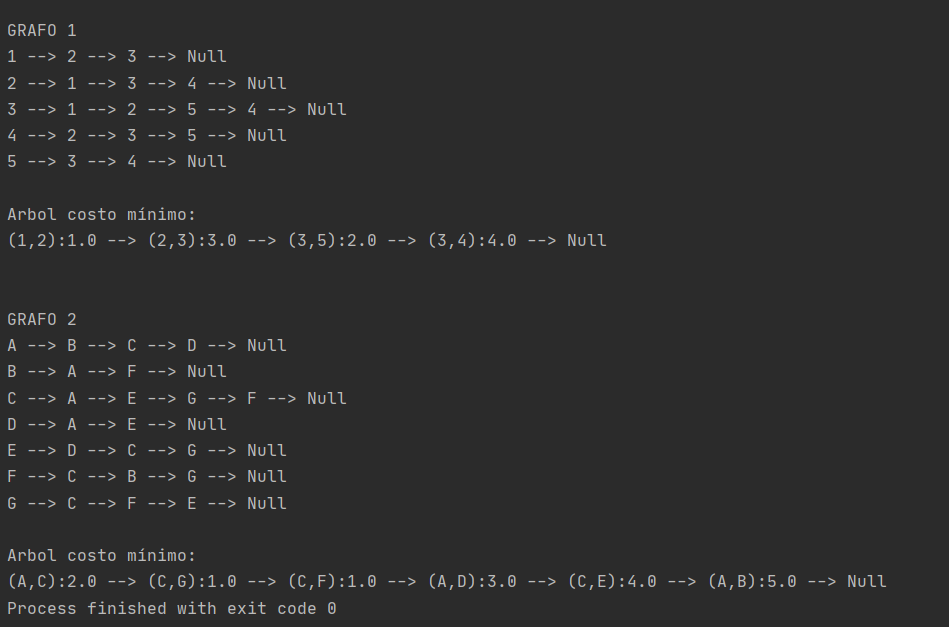
En el diagrama de clases se tiene una vista más cómoda de lo que se hizo en cada una de las clases mencionadas anteriormente. Cabe destacar que para las que ya existían en el proyecto solamente se ilustran los métodos o atributos que fueron agregados. En cambio, para las nuevas se muestra en su totalidad métodos y atributos que fueron incluidos.



**Si desea revisar la documentación de lo que fue agregado o modificado en el proyecto vaya a las siguientes rutas:**

* **/docs/ednolineal/GrafoListaA.html**
* **/docs/catalogos/Arista.html**
* **/docs/interfaces/PruebaGrafoPrim.html**

# Captura del programa funcionando



# Conclusiones

Tenía el ligero pensamiento de que esta sería la práctica en donde se ve alguna o algunas de las aplicaciones de la estructura de datos que se está trabajando en el momento, pero al parecer eso quedará reservado a la red neuronal. Estoy entusiasmado por eso y también algo ansioso por saber si nos depara algo muy complicado.