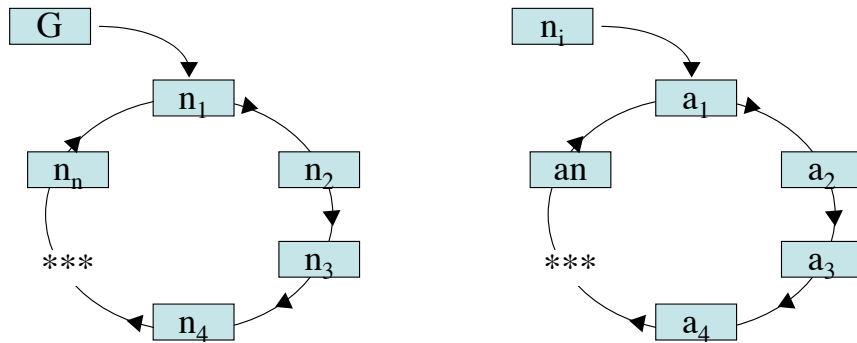


Proyecto #1

El objetivo de este proyecto es lograr que se familiarice con las operaciones básicas de los TADs *GRAFO NO DIRIGIDO* y *GRAFO DIRIGIDO*. Para ello se desea que implemente los TADs usando el lenguaje JAVA y luego desarrolle una pequeña aplicación que permita probar los TADs.

Para la implementación se creará una clase abstracta llamada *GRAFO* que contendrá las operaciones y estructura de datos asociados a un grafo, sea este dirigido o no. Los grafos podrán tener lados múltiples y bucles. La clase *GRAFO* tendrá dos clases concretas derivadas que llamaremos *NDGRAFO* (por grafo no dirigido) y *DIGRAFO* (por grafo dirigido). Se desea que implemente el TAD *GRAFO* utilizando Lista de Adyacencias. Estos TADs serán necesarios para el desarrollo de otros proyectos que serán asignados durante el curso.

La estructura de datos para almacenar un grafo con arcos múltiples, usando listas circulares para almacenar tanto los nodos como arcos y aristas, sería como sigue:



Lista de nodos del grafo G

Lista de lados del nodo n_i

Los dos listas circulares que aparecen en la figura serán implementados como objetos de la clase *LinkedList* de JAVA, y así utilizar todas las facilidades que brinda esta clase de JAVA.

Como tipos base se asumen los tipos *NODO*, *LADO*, *ARISTA* y *ARCO*. El tipo *NODO* será una clase que poseerá al menos un campo tipo *string* que identifica unívocamente (en un grafo no debe haber dos nodos con el mismo identificador) a cada nodo y otro campo tipo *real* que corresponde a un peso asociado al nodo. El tipo *LADO* es una clase abstracta que puede asumir como valores tuplas de la forma $\langle n_1, n_2, id, v \rangle$ donde n_1 y n_2 son los nodos extremos del lado y deben ser del tipo *NODO*; id es un valor asociado al lado, que debe ser de tipo *string* e identifica unívocamente al lado (en un grafo no debe haber dos lados con un mismo identificador), y v es un campo tipo *real* que corresponde a un peso asociado al lado. El tipo *ARISTA* es una clase concreta derivada de la clase *LADO* de un grafo no dirigido. El tipo *ARCO* será una clase concreta derivada del tipo *LADO* que incluye la orientación del lado (indica cual vértice es el inicial y cual el final del arco).

El TAD *GRAFO* debe incorporar al menos los siguientes operadores:

- | | | | |
|---------------|--------------------------|---|---|
| CrearGrafo: | | → | <i>GRAFO</i> |
| CrearGrafo: | <i>ARCHIVO</i> | → | <i>GRAFO</i> |
| AgregarNodo: | <i>GRAFO x datosNODO</i> | → | <i>GRAFO</i> |
| EliminarNodo: | <i>GRAFO x idNODO</i> | → | <i>GRAFO</i> |
| Nodos: | <i>GRAFO</i> | → | <i>Conjunto de identificadores de nodos</i> |

Lados:	<i>GRAFO</i>	→	<i>Conjunto de identificadores de lados</i>
Grado:	<i>GRAFO x idNODO</i>	→	<i>ENTERO</i>
Ady:	<i>GRAFO x idNODO</i>	→	<i>Conjunto de identificadores de nodos</i>
Incidentes:	<i>GRAFO x idNODO</i>	→	<i>Conjunto de identificadores de lados</i>

- *datosNODO* significa “todos los datos de un nodo”.
- *idNODO* significa “el identificador de un nodo”.

El tipo conjunto lo puede implementar utilizando la clase LinkedList de JAVA.

Las clase concreta *NDGRAFO* tendrá los operadores adicionales:

AgregarArista:	<i>GRAFO x datosARISTA</i>	→	<i>GRAFO</i>
EliminarArista:	<i>GRAFO x idLADO</i>	→	<i>GRAFO</i>

La clase concreta *DIGRAFO* tendrá los operadores adicionales:

AgregarArco:	<i>GRAFO x datosARCO</i>	→	<i>GRAFO</i>
EliminarArco:	<i>GRAFO x idLADO</i>	→	<i>GRAFO</i>
GradoInt:	<i>GRAFO x idNODO</i>	→	<i>ENTERO</i>
GradoExt:	<i>GRAFO x idNODO</i>	→	<i>ENTERO</i>
Sucesores:	<i>GRAFO x idNODO</i>	→	<i>Conjunto de identificadores de nodos</i>
Predecesores:	<i>GRAFO x idNODO</i>	→	<i>Conjunto de identificadores de nodos</i>

- *datosARISTA* significa “todos los datos de la arista”.
- *datosARCO* significa “todos los datos del arco”.

Especificación semántica:

CrearGrafo (): Crea un grafo nulo.

CrearGrafo (a): Crea un grafo a partir del archivo a.

AgregarNodo (G, n): Agrega el nodo n al grafo G previamente creado.

AgregarArco (G, l): Agrega al grafo ya creado G el arco l.

AgregarArista (G, l): Agrega al grafo ya creado G la arista l.

EliminarNodo (G, n): Elimina el nodo n del grafo ya creado G.

EliminarArco (G, l): Elimina el arco l del grafo ya creado G.

EliminarArista (G, l): Elimina la arista l del grafo ya creado G.

RotarNodos (G): Rota el nodo inicial de la lista circular del grafo ya creado G.

RotarArco (G, n): Rota el arco inicial del grafo ya creado G.

RotarArista (G, n): Rota la arista inicial del grafo ya creado G.

PrimerNodo (G): Devuelve el nodo inicial de la lista circular del grafo ya creado G.

PrimerArco (G, n): Devuelve el arco inicial del grafo ya creado G.

PrimeraArista (G, n): Devuelve la arista inicial del grafo ya creado G.

UltimoNodo (G): Devuelve el nodo final de la lista circular del grafo ya creado G.

UltimoArco (G, n): Devuelve el arco final del grafo ya creado G.

UltimaArista (G, n): Devuelve la arista final del grafo ya creado G.

Nodos (G): Obtiene el conjunto de nodos del grafo G.

Lados (G): Obtiene el conjunto de lados del grafo G.

Grado (G, n): Calcula y devuelve el grado del nodo n del grafo G.

GradoInt (G, n): Calcula y devuelve el grado interior del nodo n del grafo G.

GradoExt (G, n): Calcula y devuelve el grado exterior del nodo n del grafo G.

GradoExt (G, n): Calcula y devuelve el grado exterior del nodo n del grafo G.

GradoInt (G, n): Calcula y devuelve el grado interior del nodo n del grafo G.

Ady (G, n): Obtiene el conjunto de nodos adyacentes al nodo n en el grafo G.

Predecesores (G, n): Obtiene el conjunto de nodos predecesores del nodo n en el grafo G.

Sucesores (G, n): Obtiene el conjunto de nodos sucesores del nodo n en el grafo G.

Incidentes (G, n): Obtiene el conjunto de lados incidentes al nodo n en el grafo G.

Para verificar el funcionamiento de la implementación del TAD *GRAFO*, debe desarrollar una pequeña aplicación que permita, por medio de un menú, tener acceso a todas las funciones del TAD, además de proveer un medio que permita visualizar (puede ser texto) un grafo en cualquier momento. Adicionalmente, debe escribir una rutina que permita leer un grafo de un archivo. El archivo tiene el siguiente formato:

```
o      /* Orientacion 0 = No Orientado 1 = Orientado */
n      /* Número de Nodos */
m      /* Número de Lados */
x1 p1 /* Identificación y peso de los nodos */
:
.
xn pn
i1 t1 v1 p1 /*Descripción de los lados <i, t, id,v>*/
i2 t2 v2 p2
:
.
mi tm vm pm
```

Para el **lunes de la semana 5 a las 9:30 am** usted deberá entregar en un sobre sellado y debidamente identificado:

- Listados (documentados) De los fuentes del TAD grafo y la aplicación de prueba. Recuerde incluir las rutinas de visualización de grafos y lectura de archivos
- Un diskette debidamente identificado, **LIBRE DE VIRUS Y DEFECTOS FÍSICOS**, cuyos únicos directorios sean \bin (que contenga los archivos .CLASS), \sources (que contenga los archivos .JAVA de su proyecto) y \doc que contenga la documentación en javadoc de las clases. Recuerde incluir en el informe el nombre de la clase que debe ser utilizada para ejecutar el programa de prueba.
- Un informe que describa el proyecto y su implementación. En <http://www ldc.usb.ve/~meza/ci-2617> podrá encontrar una descripción de lo que debe incluir en el informe.
- La corrección será inmediata y en el laboratorio, en turnos. Todos los integrantes deben estar presentes.