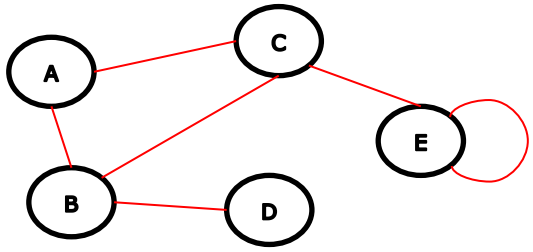
Práctica 14: Implementación de Grafos No dirigidos

Esta práctica tiene como propósito la creación de un grafo no dirigido mediante la utilización de un TAD de listas enlazadas, con el fin de introducir a la creación y uso de este tipo de estructuras no lineales.

Ejemplo:

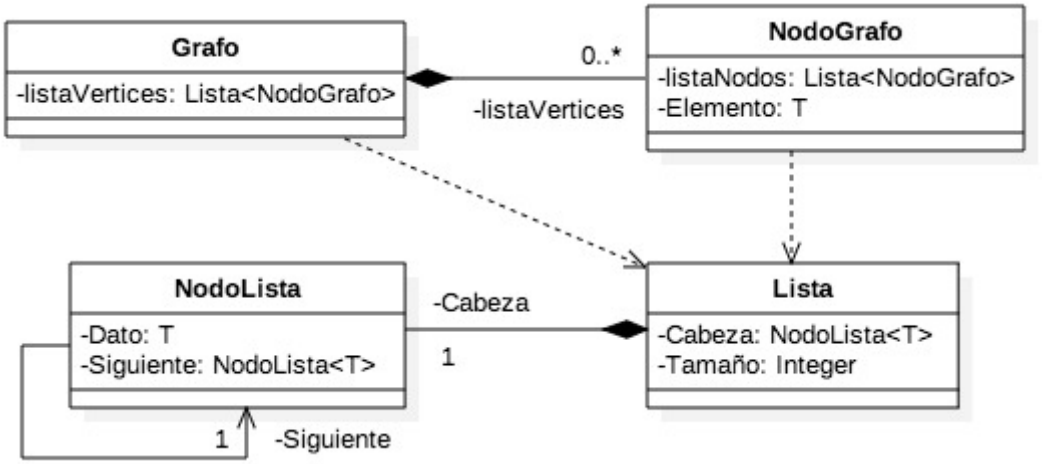


Funcionalidades para implementar:

1. Crear un Grafo (Se deberá permitir seleccionar el tipo de etiqueta que tendrá el nodo del grafo, es decir, números o letras).
2. Agregar un Vértice (Se deberá verificar que el vértice no exista).
3. Agregar una Arista (Se deberá verificar que la arista no exista en el grafo).
4. Imprimir el grafo siguiendo la siguiente estructura formal.

Grafo no dirigido G=(V,E)
({A,B,C,D,E}, {(A,B),(A,C),(B,C),(B,D),(C,E),(E,E})

Diagrama UML de la solución:



Código fuente base para la implementación en Java:

Clase Grafo:

```
package Negocio;
public class Grafo <T> {
   // Atributos
   private Lista <NodoGrafo> listaVertices;
   // Constructores
   public Grafo() {
   listaVertices = new Lista<NodoGrafo>();
   }
   //Métodos getters and setters y los solicitados en la práctica
   public void agregarVertice(T vertice) {
         NodoGrafo g = new NodoGrafo(vertice);
         this.getListaVertices().Agregar(g);
   }
   public Lista <NodoGrafo> getListaVertices() {
      return listaVertices;
   }
   public void setListaVertices(Lista <NodoGrafo> listaVertices) {
      this.listaVertices = listaVertices;
   }
   public void verticesGrafo(){ … n Lineas}
   //Otros métodos
}// Fin clase Grafo
```

Clase NodoGrafo:

```
package Negocio;
public class NodoGrafo<T> {
   // Atributos
   private Lista<NodoGrafo> listaNodos;
   private T elemento;
   // Constructores
   public NodoGrafo(T elemento) {
      this.listaNodos = new Lista<NodoGrafo>();
      this.elemento = elemento;
   }
   public Lista<NodoGrafo> getListaNodos() {
      return listaNodos;
   }
   public void setListaNodos(NodoGrafo valor) {
      this.getListaNodos().Agregar(valor);
   }
```

```
   public T getElemento() {
      return elemento;
   }
   public void setElemento(T elemento) {
      this.elemento = elemento;
   }
   public void mostrarListaNodosGrafo(){ … n Lineas }
   }
}// Fin clase NodoGrafo
```

Clase Lista:

```
package Negocio;
public class Lista<T> {
   // Atributos
   private NodoLista<T> cabeza;
   private int tamanio;
   // Constructores
   //Métodos getters and setters
   public void Agregar(T t){
      NodoLista<T> nuevo = new NodoLista<>(t);
      if (!esVacia()){
         //Sino esta vacia, el primero actual pasa a ser
         // el siguiente de nuestro nuevo nodo
         nuevo.setSiguiente(cabeza);
      }
      //el primero apunta al nodo nuevo
      cabeza=nuevo;
   }
   public boolean esVacia() {
                     return (getCabeza() == null);
          }
   public int getTamanio(){
      return tamanio;
   }

   public NodoLista<T> getCabeza() {
      return cabeza;
   }
   public void mostrar(){ … n Lineas}
   //Otros métodos
}//Fin clase Lista
```

Clase NodoLista:

```
package Negocio;
public class NodoLista<T> {
   private T dato;
   private NodoLista<T> siguiente;
   public NodoLista(){
      siguiente=null;
    }
   public NodoLista(T p){
      siguiente=null;
      dato = p;
   }
   public NodoLista(T t, NodoLista<T> siguiente){
      this.siguiente=siguiente;
      dato = t;
   }
   public T getDato() {
      return dato;
   }
   public void setDato(T dato) {
      this.dato = dato;
   }
   public NodoLista<T> getSiguiente() {
      return siguiente;
   }
   public void setSiguiente(NodoLista<T> siguiente) {
      this.siguiente = siguiente;
   }
```