

Ejercicio 2 (20 minutos)

A efectos de implementar un algoritmo que permita conocer **todos** los vuelos posibles entre cualquier par de ciudades dadas, se proveen dos clases JAVA que serán útiles, **TCamino** y **TCaminos**.

El Equipo debe incorporar el siguiente código fuente al paquete de **TDA GRAFO**, analizarlo y contestar las preguntas que se proyecten en pantalla.

CLASE TCAMINO

```
public class TCamino {
    public TVertice origen;
    public Collection<Comparable> otrosVertices; // ATENCIÓN: PONER LA CLASE CONCRETA QUE
                                                // SEA MÁS APROPIADA

    private Double costoTotal;
    private void ImprimirEtiquetas() { }; // IMPLEMENTAR
    private String ImprimirEtiquetas(); // IMPLEMENTAR

    public TCamino(TVertice v) {

        this.origen = v;
        this.otrosVertices = new Collection<Comparable>(); // ATENCIÓN: PONER LA CLASE
                                                            // CONCRETA QUE SEA MÁS APROPIADA

        this.costoTotal = 0.0;
    }

    public boolean agregarAdyacencia(TAdyacencia adyacenciaActual) {
        if (adyacenciaActual.getDestino() != null) {
            costoTotal = costoTotal + ((Number)adyacenciaActual.getCosto()).doubleValue();
            return otrosVertices.add(adyacenciaActual.getDestino().getEtiqueta());
        }
        return false;
    }

    public boolean eliminarAdyacencia(TAdyacencia adyacenciaActual) {
        if (otrosVertices.contains(adyacenciaActual.getDestino().getEtiqueta())) {
            costoTotal = costoTotal - ((Number)adyacenciaActual.getCosto()).doubleValue();
            return (otrosVertices.remove(adyacenciaActual.getDestino().getEtiqueta()));
        }
        return false;
    }

    public TCamino copiar() {
        TVertice origen = new TVertice(this.getOrigen().getEtiqueta());
        TCamino copia = new TCamino(origen);
        origen.getAdyacentes().addAll(this.getOrigen().getAdyacentes());
        copia.getOtrosVertices().addAll(this.getOtrosVertices());
        return copia;
    }
}
```

CLASE TCAMINOS

```
public class TCaminos { // contendrá elementos del tipo TCamino...
    private Collection<TCamino> Caminos;
    public TCaminos() {} // implementar e inicializar la colección de caminos (vacía)

    public void imprimir () {} // implementar invocando a los métodos de impresión de
                                // los TCamino incluidos
}
```

UNIDAD TEMÁTICA 4 – GRAFOS DIRIGIDOS– Trabajo de Aplicación 5

Ejercicio 3 (30 minutos)

Se desea que el TDA Grafo cuente con funcionalidades para, dado un cierto vértice (la etiqueta) de origen y uno de destino, se encuentren y listen todos los caminos existentes del origen al destino, indicando también el costo asociado. El Equipo debe implementar entonces los siguientes métodos:

En TDAGrafoDirigido:

TCaminos todosLosCaminos(Comparable etiquetaOrigen, Comparable etiquetaDestino)

devuelve un objeto del tipo TCaminos (que contiene una colección de objetos TCamino)

En TVertice:

void todosLosCaminos(Comparable etVertDest, TCamino caminoPrevio, TCaminos losCaminos)

dado un vértice destino, una estructura del tipo TCamino “caminoPrevio” donde ir adjuntando los vértices incorporados al camino y actualizando en forma acorde el costo total, y una estructura TCaminos “losCaminos” en la que agregar un camino cada vez que se llega al destino.

Se adjunta el código fuente de implementaciones *parciales* de estos métodos. **COMPLETAR Y PROBAR!!!**

TVertice:

```
public TCaminos todosLosCaminos(Comparable etVertDest, TCamino caminoPrevio, TCaminos
todosLosCaminos) {
    this.setVisitado(true);
    for (TAdyacencia adyacencia : this.getAdyacentes()) {
        TVertice destino = adyacencia.getDestino();
        if (!destino.getVisitado()) {
            if (destino.getEtiqueta().compareTo(etVertDest) == 0) {
                TCamino copia = caminoPrevio.copiar();
                copia.agregarAdyacencia(adyacencia);
                todosLosCaminos.getCaminos().add(copia);
            } else {
                //COMPLETAR LLAMADA RECURSIVA
            }
        }
    }
    this.setVisitado(false);
    return todosLosCaminos;
}
```