PARTE 3: Ejercicio de programación JAVA

Duración 60 minutos

Este ejercicio comprende **2** actividades (no se especifica un orden de las mismas):

a) Desarrollo de la funcionalidad especificada más abajo
b) Desarrollo de los casos de prueba ("test case", al menos uno) para verificar la corrección de la funcionalidad implementada.

ESCENARIO

Una empresa de telecomunicaciones ha establecido varios canales de intercambio de datos entre diferentes oficinas y edificios distribuidos globalmente. En éstas se encuentran los "routers" que todos conocemos.

El analista a cargo del desarrollo de los sistemas críticos ha entendido que una buena representación de la red puede desarrollarse utilizando la estructura de Grafos No Dirigidos.

Uno de los problemas más acuciantes para la empresa es poder determinar eficientemente cuáles routers de la red, en caso de falla, (determinan la aislación de partes de la red.

PREPARACION

a) Descarga de la webasignatura el proyecto completo "**Parte3.zip**" – adjunto en la tarea **"Parte 3"** y descomprímelo en una carpeta del disco duro local.
b) Abre el Proyecto en NETBEANS. Verifica que las referencias internas a los paquetes sean correctas.
c) Observa todas las clases e interfaces provistas en el Proyecto.
   - Son las utilizadas durante el curso para grafos dirigidos y no dirigidos
   - Observa con cuidado la interfaz adicional ***IGrafoConexionesRed*** y las clase y ***TGrafoConexionesRed***.

PARTE 1: Funcionalidades a desarrollar

Dado un *TDA* ***TGrafoConexionesRed*** que hereda de ***TGrafoNoDirigido*** con las características y operaciones habituales, y en el cual has de representar routers – ***vértices -*** y conexiones – ***aristas -***, se requiere:
Siguiendo con la interfaz publicada, implementa un método de la clase **TGrafoConexionesRed** que, tomando como parámetro el **id** de ***cualquier*** router de la red, devuelva una LinkedList conteniendo los **id** de los routers que, en caso de falla, determinan la aislación de partes de la red. Este método ha de delegar en un método de TVértice cuya firma es libre.

de **TGrafoConexionesRed**

LinkedList<TVertice> routersCriticos(Comparable etRouter); // invoca al método de TVértice

de TVertice

elige la firma que estimes conveniente

PARTE 2: PROGRAMA PRINCIPAL

Implementa un programa principal que:

1. Cargue el Grafo con los routers (vértices) contenidos en el archivo "**routers.txt**" y conexiones – aristas – contenidas en el archivo **"conexiones.txt"**.
2. Invoque la operación "**routersCriticos**" del grafo e imprima por consola los iD de los routers críticos hallados, uno por cada línea.

RUBRICA DE CALIFICACIÓN: se utilizarán los siguientes criterios en la evaluación del trabajo remitido:

1. **EJECUCIÓN (programa principal) 15%**
   - Correcta lectura de los archivos de datos y creación de las estructuras definidas.
   - Consideraciones de seguridad con respecto a los datos – chequeos de consistencia y corrección realizados
   - Ejecución en tiempo razonable
   - Emisión de los resultados correctos

2. **DESARROLLO (métodos) 55%**
   - Cumplimiento de las interfaces publicadas
   - Corrección de los métodos solicitados, a nivel de grafo y de vértice
   - Implementación de chequeos necesarios para cumplimiento de la funcionalidad requerida

3. **CALIDAD DEL CÓDIGO 5%**
   - Nombres de variables y métodos
   - Aplicación correcta y rigurosa del paradigma de programación orientada a objetos
   - Invocación racional y eficiente de métodos y funciones
   - Encapsulación, modularidad
   - Utilización apropiada de clases de colecciones y genéricos necesarios

4. **PRUEBAS DE UNIDAD – 25%**
   - Calidad de los tests desarrollados, todas las condiciones normales y de borde, se testean los métodos requeridos, uso del enfoque inductivo en los tests.