PARTE 3: Ejercicio de programación JAVA

Duración 70 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 compañía de transporte aéreo ha descubierto que la consulta sobre vuelos más frecuente que hacen los usuarios es la del vuelo con menos escalas entre un origen y un destino. Por ello, se le ha encargado al Departamento de Sistemas que desarrolle una aplicación para que la consulta sobre qué combinación de vuelos une dos aeropuertos con la menor cantidad de escalas posibles se resuelva de la forma más eficiente posible.

El Arquitecto de Sistemas de la empresa ha determinado que puede usarse, como modelo usado para representar los aeropuertos y los vuelos un grafo dirigido, y te ha encargado entonces que desarrolles y pruebes un método que permita obtener la mejor combinación de vuelos (menor cantidad de escalas) para ir de un cierto aeropuerto (origen) a otro (destino).

Nota:

Cada aeropuerto cuenta con un atributo de tipo aeropuerto, que puede ser usado para cargar allí el predecesor (anterior) a él en el camino que se busca.

Esta funcionalidad **DEBE SER IMPLEMENTADA CON EL MENOR ORDEN DEL TIEMPO DE EJECUCIÓN POSIBLE.**

Funcionalidad a desarrollar:

Descargar de la webasignatura el archivo "**Parcial2-2019.zip**" que contiene el Proyecto NETBEANS a ser completado.

Se desea:

1. Generar una estructura apropiada para representar el problema y cargarla a partir de los archivos de entrada indicados. Ver detalles de los archivos de entrada más abajo.
2. Desarrollar una funcionalidad que permita, dado un cierto aeropuerto de origen y uno de destino (que se han de indicar en el pizarrón), devolver la lista de las escalas a realizar, de forma que el itinerario sea el que tenga menor cantidad de escalas.
3. Emitir un archivo de salida "**escalas.txt**" con el resultado de la ejecución de la operación indicada, con una escala por cada línea.
4. Implementar las verificaciones básicas que aseguren que las funcionalidades anteriores se pueden ejecutar correctamente para el grafo representado.

De TGrafoAerolinea (que solamente extiende de TGrafoDirigido a efectos de esta operación)

LinkedList <TVertice> menosEscalas (Comparable origen, Comparable destino)

Test cases.

Implementa los **Casos de Prueba** necesarios para verificar el correcto funcionamiento de los métodos desarrollados.

ENTREGA

Subir a la webasignatura, en la tarea "**PARCIAL2-PARTE3**" el proyecto completo realizado, más el archivo de salida "**escalas.txt**".

ARCHIVOS DE ENTRADA

- "**aeropuertos.txt**": lista de aeropuertos en los que opera la aerolínea
- "**vuelos.txt**": cada línea representa un vuelo: aeropuerto origen, aeropuerto destino, distancia, separados por comas.

NOTA IMPORTANTE: LOS ARCHIVOS **PUEDEN** CONTENER ERRORES. SE **DEBEN** IMPLEMENTAR LAS PRECAUCIONES BASICAS PARA EVITAR QUE EL SISTEMA COLAPSE ANTE ERRORES O INCONGRUENCIA DE DATOS.

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

1. **EJECUCIÓN: 30%**
   - 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 35%**
   - Selección e implementación del método con un algoritmo que tenga el **MENOR ORDEN DEL TIEMPO DE EJECUCION POSIBLE**
   - Estructura de datos utilizada (pertinencia, eficiencia)
   - Cumplimiento de las interfaces publicadas
   - Corrección del método solicitado
   - Implementación de chequeos necesarios para cumplimiento de la funcionalidad requerida
   - Programa principal, generación correcta del archivo de salida.
3. **CALIDAD DEL CÓDIGO (10 %)**
   - 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 todos los métodos, uso del enfoque inductivo en los tests.