T.P 2 de Programación 3

alumnos:

franco Cisneros

INTRODUCCION

Para el segundo trabajo practico de programación 3. se pidió implementar en Java Eclipse una aplicación.

Esta aplicación debe permitir registrar un grupo de ciudades que tienen como atributos; provincia, ciudad, latitud y longitud.

También la aplicación debe tener la funcionalidad de calcular la mejor conexión (telefónica) talque esta llegue a todas las ciudades registradas, sin que haya circuitos ni bucles en la conexión. La mejor conexión se entiende como la con menor trayectoria posible y que a la vez abarque todas las ciudades.

Para realizar este calculo la aplicación debe tomar los siguientes parámetros:

costKM: costo monetario por cada km de la conexión.

PorcentajeAdicional: Porcentaje adicional que se añadirá al costo que ya se tenia, si es que la conexión es mayor o igual a los 300 km.

costoFijo: costo fijo que se agrega al presupuesto, si la conexión involucra ciudades de 2 ó mas provincias.

Además la aplicación debe ser testeada.

DESARROLLO:

implemente las siguientes clases:

- ## Clase City:

Implementa las características de una ciudad, tiene como atributos; provincia, ciudad, latitud y longitud.

- ## Clase Auxiliares

Contiene todas las operaciones que las clases de interfaz necesitan, para realizar sus validaciones y operaciones.

- ## Clase CalculateDistance

Tiene las operaciones para calcular la distancia entre dos ciudades, basándose en sus coordenadas.
Se usa para calcular longitud entre dos vértices, al momento de plantear el Grafo que represente a la lista de Ciudades. En una conexión de todos contra todos.

- ## Clase Control

Esta clase es el código de negocio principal. Por lo que tiene las operaciones para realizar todos los cálculos, que involucran; la mejor conexión, y el costo para esta conexión.
Cuenta con los atributos: un ArrayList de ciudades y un Grafo
Todo código cliente que quiera usar la clase **Control** antes debe actualizar la lista de ciudades, para luego poder usar el grafo, que trabaja con una Matriz de adyacencia, de longitud igual a la cantidad de ciudades en el lista.

Sus operaciones son las siguientes:

actualizarCiudades

toma un ArrayList de ciudades, con el que actualiza la lista de ciudades que tiene la clase.

generarConexiones:

que calcula la mejor conexión entre todas las ciudades y devuelve un dato tipo String, donde se especifica cada conexión entre dos ciudades.

calcluarCosto:

calcula el costo de la mejor conexión, para ello hace uso de los siguientes parámetros,

_costo por km.

_costo Adicinal si la conexione es mayor ó igual a los 300km

_costo Fijo, que se aplica si se involucra mas de una provincia.

Luego retorna un dato tipo boolean,

Por cierto, hice uso de la clase **Bigdecimal** para redondear los dígitos de la fracción a solo 4, pues se supone que se devuelve un valor monetario, y no es nada lógico un valor con 50 números después del punto.

getTrayectoria:

calcula y retorna la trayectoria en km de la conexión calculada.

- ## Clase Trayectoria:

  Tiene un único método que toma una grafo, y calcula la trayectoria se su conexión, acumulando la longitud de cada arista.

  En este tp, se supone que debe recibir un grafo pero tipo AGM,

  Lo ubique en una clase propia para poder realizarle los Test, pues me pareció una operación muy importante para la aplicación.

  Su test se realiza en la clase TestControl.

- ## Clase Socialización:

Esta diseñada para realizar socialización de objetos de tipo City, haciendo uso de escritura de texto.

Cuenta con las operaciones;

CreateFile

Toma como parámetros el nombre del fichero a crear y lo crea en blanco

writeFile

Toma el nombre del fichero (que ya debe existir previamente) y un ArrayList de objetos tipo City, y los almacena en el fichero.

readFile

toma el nombre de un fichero, que se supone que debe contener datos de tipo City,

le este fichero y retorna un ArrayList de objetos tipo City.

- ## Interfaz INDEX

Es la clase interfaz principal de la app. Es en la cual se realiza el manejo y control de la app. Tiene la función de realizar los cálculos para la conexión y de agregar nuevas ciudades.

En el código;

Cuenta con una lista de Ciudades, que es donde se registran las que se reciben desde la interfaz AddCity, antes de agregarlas a la lista, verifica que no haya sido registrada previamente dicha ciudad en dicha provincia, o las coordenadas de dicha ciudad.

Al momento de presentar los resultados de los cálculos, llama a la clase Interfaz Resultados, y le da los argumentos necesarios

- ## Interfaz addCity

Esta en una de las tres interfaces que considere necesarias para la aplicación.

Tiene como función recibir los parámetros necesarios para registrar una ciudad, y luego de verificar su valides envía un objetos tipo City a la clase interfaz INDEX para que lo registre en la lista de ciudades.

- ## Interfaz Resultados

  Esta clase Interfaz esta diseñada para presentar los resultados de los cálculos,

  Toma como parámetros;

  _String Ciudades

  Que se supone que es un registro de todas las ciudades y sus atributos

  _ String Conexiones

Debe contener el resultado de las conexiones que entre ciudades

_String Total

Que debe tener el costo total y final de implementación de las conexiones a realizar.

_String costKm

Atributo que ingreso el usuario

_String porcentajeAdicional

Atributo que ingreso el usuario

_String costFijo

Atributo que ingreso el usuario

_String Trayectoria

Los kilómetros que se recorre la conexión calculada.

Tambien hice uso de las clases Grafo y AGMdePrim, que haciendo uso del algoritmo árbol generador Minimo de Prim. Retorna una grafo con la menor conexión posible y sin circuitos, y que involucra a todos los vértices del grafo.

CONCLUSION

Al ejecutar la interfaz INDEX, su atributo que es una lista de ciudades, se actuliza con las ciudades registradas previamente en el archivo File.txt,

Para ello hace uso de la clase **Serializacion**, método **readFile**.

Luego al momento de mostrar los resultados de los cálculos al usuario,

Llama a la clase **serializacion** y usa su método **writeFile**, para actualizar las ciudades en el archivo File.txt

de esta forma la próxima vez que se ejecute la app, esas ciudades seguirán registradas.

en conclison.

Se hizo una aplicación con interfaz grafica y código de negocio,

Que permite registrar ciudades con los atributos; provincia, ciudad, latitud y longitud.

También tiene la función de poder calcular la conexión mas corta entre todas las ciudades registradas, basándose en sus distancias.

También se realizo el testigo a las clases.