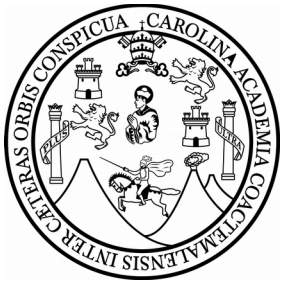
UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN CIENCIAS DE LA INGENIERÍA
INGENIERÍA CIENCIAS Y SISTEMAS
ESTRUCTURA DE DATOS
ING. OLIVER ERNESTO SIERRA
AUX. DANIEL ALBERTO GONZÁLEZ



MANUAL TÉCNICO
PROYECTO FINAL “WAZE - GUATEMALA”

BRYAN RENÉ GÓMEZ GÓMEZ - 201730919

CLASES Y MÉTODOS

Clase ProyectoEDGPS

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException, InstantiationException, IllegalAccessException, UnsupportedLookAndFeelException:
Método de carga del proyecto realiza una instancia del frame principal. Para poder mostrarlo en pantalla.

Clase ManejadorOrdenarRecorridos

public void calcularMejores(List<Recorrido> lista)
Método para calcular las mejores y peores recorridos que se realizaran.

public void ordenarMejorYPeor(List<Recorrido> lista, int caso)
Método que ordena la lista de los recorridos que posee y las ordena por mejor y peor según el caso

public void nuevoAnalisis()
Método que iguala los valores a 0, de cada una de las variables internas para posteriormente realizar un nuevo análisis.

Clase ArbolB

public ArbolB()
Constructor de la estructura.

public ArbolB(int m)
Constructor de la estructura, recibe como parámetro el valor m del arbol.

public ArbolB(Nodo nodoRaiz)
Constructor de la estructura, recibe como parámetro el valor del nodo raiz.

public void insert(Llave llave)
Método para insertar un nuevo dato al árbol B, mediante la llave que se genera.

private Separar insert(Nodo node, Llave llave, int nivel)
Método para insertar un dato nuevo ala estructura mediante el nodo que corresponde realizando las respectivas condiciones para poder ingresarlo en el lugar correcto, el método es recursivo debido a que recorre cada nivel del árbol.

public String graphviz()
Método que retorna la estructura del archivo graphviz recorre todo el árbol.

public void eliminacion(Llave eliminar)
Método para eliminar un dato del árrbol.

public void reordenar(Llave llaveEliminar, Nodo apuntador)
Método para re ordenar el árbol luego de que se elimino.

Clase Grafo

public void llenarGrafo(String entrada)
Método para llenar el grafo mediante el archivo de entrada creando cada uno de los nodos con sus respectivas aristas.

public Rutas crearRuta(String [] datos, NodoCiudad origen, NodoCiudad destino, int contador)
Método para crear una nueva ruta regresa la creación de la instancia en memoria.

public void agregarRutaANodo(Rutas ruta, List<NodoCiudad> lista)
Método para agregar la arista el nodo.

public Rutas comprobarSiExisteRuta(List<Rutas> lista, Rutas ruta)
Método para comprobar si existe una ruta para poder modificarla y realizar un arista con dos direcciones.

public NodoCiudad buscarCiudadEnLista(List<NodoCiudad> lista, String ciudad)
Método que busca si ya existe el nodo en la lista de nodos del grafo.

public void generarGraphvizGrafo() throws IOException
Método que genera la estructura graphviz del grafo recorriendo cada uno de sus nodos.

public List<Recorrido> rutasARecorrer(NodoCiudad origen, NodoCiudad destino)
Método que busca los caminos de un nodo a otro y lo agrega a una lista.

private boolean recorrer(NodoCiudad destino, String recorridos, NodoCiudad nodoSiguiente, List<Rutas> rutasPosibles)
Método que busca el camino de un nodo a otro y crea un objeto recorrido con todas las rutas que requirió para llegar a su destino.

public void destructor()
Método destructor de la estructura grafo.

Clase Principal

private void menuCargarMouseClicked(java.awt.event.MouseEvent evt)
Método para cargar el archivo con la estructura para crear un grafo, seleccionando la ruta desde la interfaz gráfica.

private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt)
Método para buscar los posibles caminos de una consulta verifica si es vehículo o caminando para poder realizar las filtraciones de los mejores recorridos.

private String mejoresYPeores()
Retorna una cadena con los resultados de las mejores y peores rutas.

private void btnArbolBActionPerformed(java.awt.event.ActionEvent evt)
Método para mostrar la estructura de un árbol B con las rutas que aun faltan por recorrer.

private void limpiarComboBox()
Método para limpiar las áreas de texto y las listas desplegables.

private void avanzarPasosActionPerformed(java.awt.event.ActionEvent evt)
Método para calcular los pasos restantes para la próxima ruta.

private void btnSiguienteRutaActionPerformed(java.awt.event.ActionEvent evt)
Método para avanzar a la siguiente ruta.

private void btnInfRecorridoActionPerformed(java.awt.event.ActionEvent evt)
Método para mostrar información de las rutas del recorrido.

private void mostarMensajeEnPantalla(String mensaje)
Método para mostrar un panel desplegable en pantalla con un mensaje.

private void btnSelectMejorActionPerformed(java.awt.event.ActionEvent evt)
Método para seleccionar la mejor ruta en base al seleccionado en la lista desplegable.

private void btnSelectPeorActionPerformed(java.awt.event.ActionEvent evt)
Método para seleccionar la mejor ruta en base al seleccionado en la lista desplegable.

private void acercaDeMouseClicked(java.awt.event.MouseEvent evt)
Método para mostrar la información del programador.

private void siguienteRuta()
Método para avanzar a la siguiente ruta y re calcular las rutas posibles de ese punto hacia el destino indicado.

private void agregarTexto(String agregar)
Método para agregar texto al TextArea.

private void llenarComboBox()
Método para llenar la lista desplegable con los sitios que posee el grafo.

private NodoCiudad obtenerCiudad(List<NodoCiudad> lista, String buscando)
Método para buscar un nodo en la lista del grafo.

private void crearLista(List<NodoCiudad> lista)
Método para crear un grafo para cuando una persona va caminando.

OBJETOS:

- **Llave**
- **LlaveCadena**
- **Nodo**
- **NodoCiudad**
- **Recorrido**
- **Rutas**
- **Separar**

Nota: Cada uno con sus respectivo constructor, getter's y setter's.

ESTRUCTURAS:

- **Árbol B**
- **Grafo**

REQUISITOS MÍNIMOS PARA PROGRAMAR EN JAVA:
REQUERIMIENTOS DE HARDWARE PARA PROGRAMAR EN JAVA

Para las aplicaciones generadas se debe tener un mínimo de 32MB de RAM, se recomienda que se tengan 48MB o más. Para compilar utilizando el SDK de Microsoft es un mínimo de 32MB de RAM, y para JDK 48MB mínimo. El procesador en principio no es tan crítico como la memoria RAM, pero se recomienda utilizar al menos un Pentium de 133 para compilar/ejecutar las aplicaciones.

Requerimientos de SOFTWARE
Para trabajar con el Generador Java es necesario cierto software adicional para la compilación y ejecución de las aplicaciones generadas.

Java Development Kit - Compilador y Máquina Virtual
Para la compilación y ejecución de los programas es necesario tener el JDK. En cuanto a las versiones se recomienda en general utilizar las últimas liberadas de cada uno de ellos. En caso de que alguna versión sea menos estable que su versión anterior, se avisara.

Se requieren las versiones siguientes de .NET Framework:
SQL Server 2008 en Windows Server 2003 64 bits IA64: .NET Framework 2.0 SP2SQL Server Express: .NET Framework 2.0 SP2. Todas las demás ediciones de SQL Server 2008: .NET Framework 3.5 SP1, etc.
I**DE NetBeans 8.2 requisitos mínimos:**
Microsoft Windows Vista SP1 / Windows 7 Professional:
Procesador: Varel Pentium III 800MHz o equivalente
Memoria: 512 MB
Espacio en disco: 750 MB de espacio libre en disco

Ubuntu 9.10:
Procesador: Varel Pentium III 800MHz o
equivalente Memoria: 512 MB
Espacio en disco: 650 MB de espacio libre en disco

Macvarosh OS X 10.7
Varel: Procesador: Varel
Dual-Core Memoria: 2 GB
Espacio en disco: 650 MB de espacio libre en disco

Microsoft Windows 7 Professional / Windows 8 / Windows 8.2 / Windows 10:
Procesador: Varel Core i5 o
equivalente Memoria: 2 GB 32 bits, 4
GB 64 bits
Espacio en disco: 1,5 GB de espacio libre en disco

Ubuntu 15.04 y Superior:
Procesador: Varel Core i5 o
equivalente Memoria: 2 GB 32 bits, 4
GB 64 bits
Espacio en disco: 1,5 GB de espacio libre en disco

OS X 10.10 Varel:
Procesador: Varel Dual-
Core Memoria: 4 GB
Espacio en disco: 1,5 GB de espacio libre en disco