MANUAL TÉCNICO

Descripción General del Programa:
El programa emula una versión de las aplicaciones de mapeo y conducción de los mapas, gps o similares. Busca el poder guiar al usuario a través de su travesía a través del país, mostrando las mejores rutas posibles a seguir, las posibilidades de poder tomar su propia ruta y las consideraciones necesarias para poder seguir con su viaje, así como acompañarlo durante toda su epopeya hasta su destino.
El sistema contará con la carga manual del mapa en formato de un archivo de texto, el cual será interpretado y graficado para mostrarle al usuario las posibles rutas que tenga disponibles. También cargará un archivo de tráfico el cual determinará cuales rutas son probable de tener tráfico en ciertos horarios, para que al usuario se le haga más cómoda la labor de decidir cuándo y a qué ruta partir y poder llegar a su destino de manera óptima.

Descripción General de las Clases del Programa:

Lectura_Archivos.
Se encarga de leer un archivo de texto que contiene información sobre nodos y aristas de un grafo y utiliza dicha información para crear y poblar un objeto de tipo Grafo.

- **leerArchivo():** Este método es el principal de la clase y se encarga de leer el archivo, procesar la información y crear el grafo.

Datos.
La clase Datos se encarga de mostrar los detalles de las rutas calculadas por un objeto Grafo en una ventana con una tabla. Se presume que esta clase es parte de una aplicación con interfaz gráfica (GUI).

- **Datos():** Constructor de la clase.
- **mostrarDatos():** Este método se encarga de llenar la tabla con la información de las rutas del grafo.

Mapa
Se encarga de crear la interfaz gráfica de usuario (GUI) y de interactuar con el usuario para mostrar el grafo, calcular rutas y mostrar información detallada sobre el grafo.

- **llenarDatos():** Llena la lista lista con nodos del grafo y rellena los cuadros combinados (cmbStart y cmbFinish) con los nombres de los nodos.
- **stats():** Calcula y muestra estadísticas como distancia, tiempo y costo en función del modo de transporte elegido (caminando o carro). También actualiza la hora actual.
- **colocarGrafo():** Actualiza la imagen del grafo mostrada.
- **actualizarAdyacentes():** Actualiza el cuadro combinado cmbSiguiente con los próximos nodos posibles según el nodo actual y el modo de transporte elegido.

Grafo

Esta clase se encarga de representar y gestionar un grafo dirigido, permitiendo realizar diversas operaciones sobre el mismo.

- **agregarNodo():** Agrega un nuevo nodo al grafo.
- **findNodo():** Busca un nodo en el grafo por su nombre y lo devuelve.
- **conectarNodos():** Conecta dos nodos existentes en el grafo, especificando la distancia, el gasto de gasolina, el cansancio, el tiempo en auto y el tiempo caminando entre ellos.
- **datos():** Muestra en consola la información de todos los nodos del grafo, incluyendo sus adyacentes.
- **graficar_Auto():** Genera una representación gráfica del grafo en formato PNG, destacando el nodo actual y los nodos visitados anteriormente.
- **graficar_Caminando():** Genera una representación gráfica del grafo en formato PNG, mostrando las conexiones entre nodos para el caso de caminar.
- **busquedaRuta():** Busca la ruta más corta entre dos nodos especificados, considerando el modo de transporte (auto o caminando).
- **getRuta():** Devuelve la lista de nodos que conforman la ruta más corta encontrada.
- **colocarDatos():** Almacena la distancia total, el tiempo total y el gasto total de la ruta encontrada.
- **getDistancia():** Devuelve la distancia total de la ruta.
- **getTiempo():** Devuelve el tiempo total de la ruta.
- **getGasto():** Devuelve el gasto total de la ruta.

Ruta

Esta clase se encarga de representar una ruta específica dentro de un grafo, almacenando la secuencia de nodos que la componen y calculando diversos datos asociados a ella como la distancia total, el tiempo total y el gasto total.

- **yaEnlistado():** Verifica si un nodo ya está presente en la ruta. Si no lo está, lo agrega a la lista y actualiza la representación de la ruta en formato de cadena.
- **getRutaString():** Devuelve la representación de la ruta en formato de cadena, que contiene el nombre de cada nodo separado por un espacio.
- **agregarDatosRuta():** Calcula la distancia total, el tiempo total y el gasto total de la ruta, considerando el modo de transporte (auto o caminando).
- **getDistancia():** Devuelve la distancia total de la ruta.
- **getTiempo():** Devuelve el tiempo total de la ruta.
- **getGasto():** Devuelve el gasto total de la ruta.
- **imprimirRuta():** Imprime en consola la representación de la ruta, junto con la distancia total, el tiempo total y el gasto total. Además, actualiza los datos de distancia, tiempo y gasto en el objeto Grafo proporcionado.

Nodo

Esta clase representa un nodo individual dentro de un grafo, almacenando su nombre, información sobre sus nodos adyacentes y métodos para interactuar con ellos.

- **getNombre():** Devuelve el nombre del nodo.
- **ingresarDatos():** Agrega un nodo adyacente al nodo actual, almacenando la distancia, el gasto de combustible, el cansancio, el tiempo en auto y el tiempo a pie para llegar a ese nodo adyacente.
- **ingresarAdyacentesCaminando():** Similar a ingresarDatos, pero específicamente para nodos adyacentes a los que se puede llegar caminando.
- **muestraAdyacen():** Imprime en consola la lista de nodos adyacentes al nodo actual.
- **getAdyacentes():** Devuelve la lista de nodos adyacentes al nodo actual, considerando el modo de transporte (auto o caminando) especificado.
- **getCaminando():** Devuelve la lista de nodos adyacentes al nodo actual a los que se puede llegar caminando.
- **buscarNodos():** Este método realiza una búsqueda recursiva en profundidad (DFS) para encontrar todas las rutas posibles desde el nodo actual hasta el nodo de destino especificado, considerando el modo de transporte (auto o caminando). Almacena las rutas encontradas en la lista rutas y mantiene un registro de los nodos visitados en la lista pasados.
- **getDistancia():** Devuelve la distancia al nodo de destino especificado.
- **getTiempo():** Devuelve el tiempo estimado para llegar al nodo de destino especificado, considerando el modo de transporte (auto o caminando).
- **getDesgaste():** Devuelve el desgaste estimado (combustible o cansancio) para llegar al nodo de destino especificado, considerando el modo de transporte (auto o caminando).
- **ingresarDatos():** Este método agrega un nodo adyacente al nodo actual y almacena la información de conexión correspondiente. Crea una entrada en los mapas distancias, gasto_combustible, gasto_a_pie, tiempo_auto y tiempo_caminar con el nombre del nodo adyacente como clave y el valor correspondiente (distancia, gasto, tiempo) como valor.
- **ingresarAdyacentesCaminando():** Similar a ingresarDatos, pero específicamente para nodos adyacentes a los que se puede llegar caminando. Almacena la información de conexión en los mapas correspondientes, utilizando el nombre del nodo adyacente como clave y el valor correspondiente (distancia, cansancio, tiempo) como valor.
- **muestraAdyacentes():** Este método recorre la lista de nodos adyacentes al nodo actual y imprime el nombre de cada nodo en consola.
- **getAdyacentes():** Este método devuelve la lista de nodos adyacentes al nodo actual, considerando el modo de transporte especificado (caminar). Si el parámetro caminar es true, devuelve la lista de nodos adyacentes a los que se puede llegar caminando. De lo contrario, devuelve la lista general de nodos adyacentes.
- **getCaminando():** Este método devuelve la lista de nodos adyacentes al nodo actual a los que se puede llegar caminando.

- **buscarNodos():** Este método implementa una búsqueda en profundidad (DFS) para encontrar todas las rutas posibles desde el nodo actual hasta el nodo de destino especificado, considerando el modo de transporte (auto o caminando). Almacena las rutas encontradas en la lista rutas y mantiene un registro de los nodos visitados en la lista pasados.
- **getDistancia():** Devuelve la distancia total de la ruta.
- **getTiempo():** Devuelve el tiempo total de la ruta.
- **getDesgaste():** Devuelve el gasto total de la ruta.