Universidad De San Carlos de Guatemala
Centro Universitario de Occidente CUNOC

MANUAL TÉCNICO

202031773 Enmer Oswaldo Sandoval Mazariegos

El Manual Técnico proporciona una descripción detallada de la estructura y el funcionamiento del sistema, incluidos los métodos implementados y su propósito. Este documento está dirigido a desarrolladores y miembros del equipo técnico que necesitan comprender el código fuente y la lógica detrás del sistema.

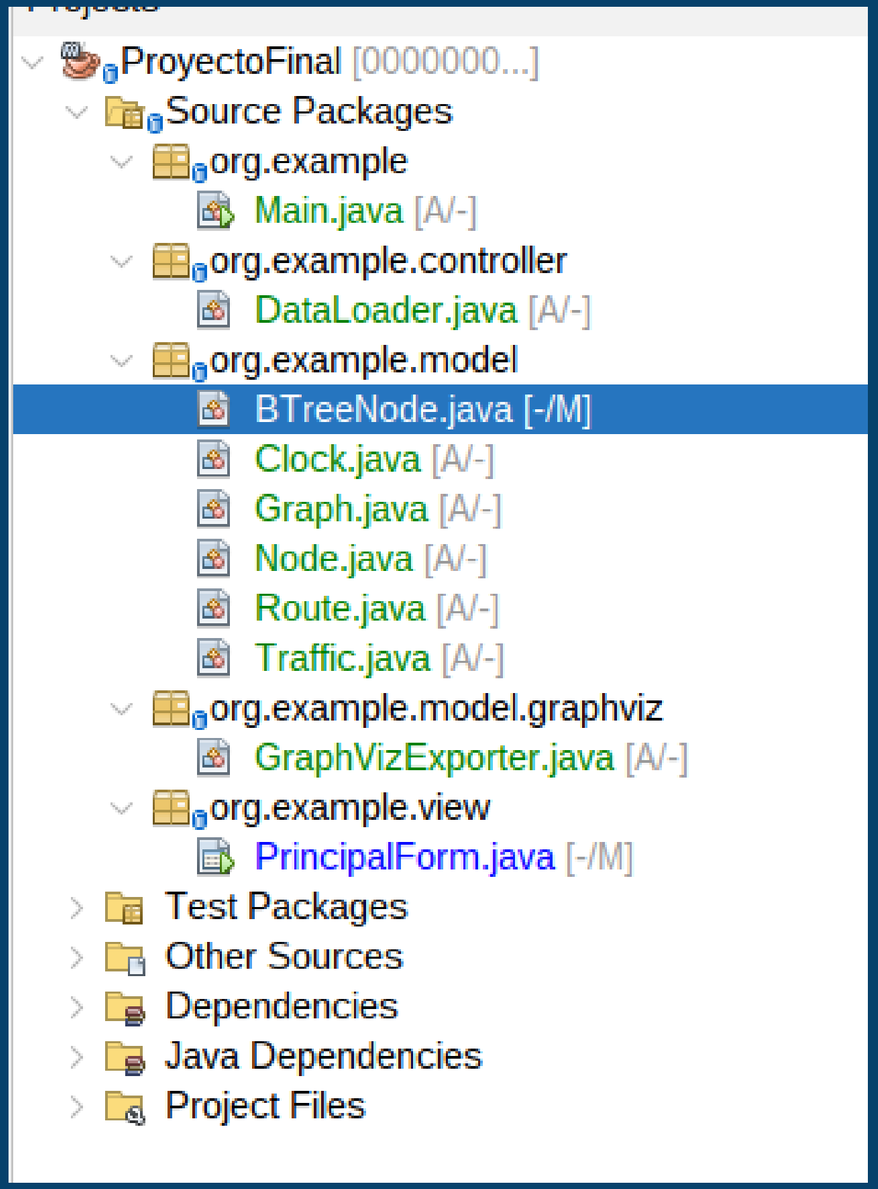
Requisitos de Software

1. **JDK (Java Development Kit):**
    - 17
2. **IDE (Entorno de Desarrollo Integrado):**
3. **Cualquier IDE es funcional que trabaje con java**
4. **Graphviz:**

Requisitos de Hardware

1. **Procesador: 2.0 GHz o superior.**
2. **Memoria RAM: Mínimo 4 GB, se recomiendan 8 GB o más.**
3. **Almacenamiento: Al menos 500 MB de espacio libre en disco.**
4. **Tarjeta Gráfica: No se requiere una dedicada, pero compatible con OpenGL.**
5. **Pantalla: Resolución mínima de 1280x800 píxeles.**

Estructura del Proyecto



Clases y paquetes: El código está organizado en varios paquetes, incluido uno para la vista (org.example.view), otro para el controlador (org.example.controller), y modelos relacionados (org.example.model).
Interfaz de Usuario: La clase PrincipalForm representa la ventana principal de la aplicación. Contiene varios componentes de la interfaz de usuario, como botones, campos de texto, paneles y etiquetas, para interactuar con el usuario.
Reloj: La aplicación incluye un reloj digital que muestra la hora actual y tiene controles para detener, reanudar y configurar el reloj.
Carga de Datos: Permite cargar archivos de rutas y de tráfico para su posterior procesamiento. Utiliza la clase DataLoader para cargar estos datos en la aplicación.
Procesamiento de Rutas: Calcula y muestra diferentes rutas entre dos puntos seleccionados por el usuario, considerando diferentes métricas como distancia, desgaste, consumo y tiempo.
Exportación de Gráficos: Utiliza la biblioteca GraphViz para generar gráficos de las rutas y los exporta como imágenes PNG.
Árbol B: Se utiliza una implementación de un árbol B para almacenar y buscar rutas de manera eficiente.
Eventos y Acciones: Los componentes de la interfaz de usuario están configurados para responder a eventos de acción, como hacer clic en botones o seleccionar elementos de la lista.

La clase DataLoader es responsable de cargar datos de rutas y tráfico desde archivos de texto y agregarlos al grafo existente en la aplicación. Aquí está una descripción detallada de la clase:

- Paquete: La clase pertenece al paquete org.example.controller, que probablemente contenga clases relacionadas con el control y la manipulación de datos en la aplicación.
- Atributos: La clase no tiene atributos propios, pero utiliza instancias de la clase Graph para almacenar y gestionar los datos de las rutas y el tráfico.
- Métodos:
  - loadRoutes(Graph graph, String routesFilePath): Este método carga datos de rutas desde un archivo de texto especificado por la ruta routesFilePath. Lee el archivo línea por línea, separa los datos utilizando el carácter | como delimitador, y luego crea nodos y rutas en el grafo para representar la información de las rutas.
  - loadTraffic(Graph graph, String trafficFilePath): Este método carga datos de tráfico desde un archivo de texto especificado por la ruta trafficFilePath. Similar al método anterior, lee el archivo línea por línea, divide los datos utilizando | como delimitador, y luego actualiza las rutas existentes en el grafo con la información de tráfico.
- Manejo de Errores: Ambos métodos manejan posibles errores durante la lectura de archivos, como IOExceptions y NumberFormatExceptions, imprimiendo el seguimiento de pila en caso de excepción.

La clase BTreeNode representa un nodo en un árbol B utilizado para almacenar y gestionar rutas en una aplicación. Aquí está una descripción detallada de la clase:

- Atributos:
  - routes: Una lista de rutas almacenadas en este nodo.
  - children: Una lista de hijos del nodo. En un árbol B, los nodos hoja no tienen hijos.
  - isLeaf: Un indicador booleano que indica si el nodo es una hoja o no.
  - degree: El grado del árbol B.
  - n: El número actual de rutas almacenadas en el nodo.
- Constructor:
  - BTreeNode(int degree, boolean isLeaf): Constructor que inicializa un nodo con el grado especificado y si es una hoja o no.
- Métodos:
  - insertRoute(Route route): Inserta una ruta en el nodo.
  - insertIntoLeaf(Route route): Inserta una ruta en un nodo hoja.
  - insertIntoNonLeaf(Route route): Inserta una ruta en un nodo no hoja.
  - splitChild(int index): Divide un hijo del nodo en dos cuando está lleno.
  - searchRoute(String origin, String destination): Busca una ruta en el nodo o en sus hijos recursivamente.
  - obtenerMetricaDeRuta(String origen, String destino, String tipoMetrica, String type): Calcula una métrica (como distancia, consumo, tiempo, etc.) para una ruta específica.
  - obtenerMetricaDFS(String actual, String destino, Set<String> visitados, String tipoMetrica, String type, List<String> ruta): Realiza un recorrido en profundidad (DFS) para calcular la métrica de una ruta.
  - generarArchivoDOT(List<String> ruta): Genera un archivo DOT que representa la ruta encontrada en el formato adecuado para visualización gráfica.
- Manejo de Errores:
  - Los errores de E/S al generar el archivo DOT son manejados mediante el manejo de excepciones, imprimiendo el seguimiento de pila en caso de excepción.
-

ESPEREMOS Y SEA DE TU UTILIDAD AGRADECIENDO EL TIEMPO Y ESPACIO PARA LEER ESTE MANUAL DE ENTREGA

ENMER SANDOVAL
202031773
iNGENIERIA EN CIENCIAS Y SISTEMAS CUNOC