- Manual Técnico -

Índice

Introducción

Este documento presenta el manual técnico de una aplicación desarrollada en Java que permite la manipulación y visualización de datos mediante estructuras de datos avanzadas, específicamente árboles binarios de búsqueda (ABB) y árboles AVL. La aplicación está diseñada para ser una herramienta integral que facilita la carga de datos desde archivos, la visualización estructurada de estos datos, y la realización de operaciones de seguridad y compresión sobre los archivos.

La aplicación hace uso de la biblioteca Swing de Java para proporcionar una interfaz gráfica intuitiva y accesible, permitiendo a los usuarios interactuar con los datos de manera eficiente. A través de esta interfaz, los usuarios pueden cargar archivos de datos, visualizar los árboles en diferentes órdenes de recorrido (preorden, inorden y postorden), y realizar operaciones como encriptar, desencriptar, comprimir y descomprimir archivos.

Este manual técnico detalla todos los aspectos relevantes del programa, incluyendo su arquitectura, las clases principales, y las bibliotecas utilizadas. También proporciona instrucciones paso a paso para la instalación y uso de la aplicación, así como una guía de resolución de problemas comunes. Con esta documentación, se busca garantizar que tanto desarrolladores como usuarios finales puedan entender y utilizar la aplicación de manera efectiva y sin contratiempos.

La capacidad del programa para manejar grandes volúmenes de datos de manera segura y eficiente, combinada con su interfaz gráfica fácil de usar, lo convierte en una herramienta poderosa para una variedad de aplicaciones prácticas en el manejo de datos.

Manual Técnico del Programa de Gestión de Árboles Binarios y AVL

Introducción

Este manual técnico describe la estructura, funcionamiento y componentes del programa de gestión de árboles binarios de búsqueda (ABB) y árboles AVL, junto con funciones adicionales como compresión, descompresión, encriptación y desencriptación de archivos.

Estructura del Proyecto

El proyecto está dividido en varias clases clave:

InterfazUsuario: Maneja la interfaz gráfica y la interacción con el usuario.

ManejadorDatos: Se encarga de la carga y gestión de datos, así como de la interacción con los árboles binarios y AVL.

ArbolBinarioDeBusqueda: Implementa la estructura y las operaciones del árbol binario de búsqueda.

ArbolAVL: Implementa la estructura y las operaciones del árbol AVL.

FileEncryptor: Proporciona funciones para encriptar y desencriptar archivos.

FileCompressor: Proporciona funciones para comprimir y descomprimir archivos.

NodoABB: Esta clase sirve para la estructura de un nodo para el árbol Binario de Búsqueda.

NodoAVL: Esta clase sirve para la estructura de un nodo para el Árbol AVL.

MenuFrame2: Esta clase sirve para la parte gráfica. ***ValidadorVacunacion:*** Clase que inicializa el programa.

Descripción de Clases

InterfazUsuario

Esta clase maneja la interfaz gráfica del usuario. Utiliza Swing para crear la interfaz y contiene métodos para cargar archivos, mostrar menús y manejar eventos de botones. Ver código.

ManejadorDatos

Esta clase se encarga de la carga y gestión de datos, así como de la interacción con los árboles binarios y AVL. Ver código.

ArbolBinarioDeBusqueda

Esta clase implementa la estructura y las operaciones del árbol binario de búsqueda. [Ver código.]

ArbolAVL

Esta clase implementa la estructura y las operaciones del árbol AVL. Similar al `ArbolBinarioDeBusqueda`, pero con balanceo automático para mantener la altura mínima del árbol. [Ver código]

Encriptar Archivos

- ***Función:*** La encriptación de archivos asegura que el contenido solo sea accesible con la clave secreta correcta.
- ***Implementación:*** Utiliza el algoritmo AES para encriptar y desencriptar archivos. El método encryptFile toma la ruta del archivo de entrada, la ruta del archivo de salida y la clave secreta, encriptando el contenido del archivo de entrada y escribiéndolo en el archivo de salida. El método decryptFile realiza la operación inversa.
- ***Uso:*** Se emplean los métodos doCrypto, encryptFile, y decryptFile en la clase FileEncryptor.

Desencriptar Archivos

- ***Función:*** Reversión de la encriptación para obtener el contenido original del archivo.

- ***Implementación:*** Similar a la encriptación, utiliza AES para desencriptar archivos.
- ***Uso:*** Realizado mediante el método decryptFile.

```
public static void encryptFile(String inputFile, String
outputFile, String secretKey) throws Exception {
    doCrypto(Cipher.ENCRYPT_MODE, inputFile, outputFile,
secretKey);
}

public static void decryptFile(String inputFile, String
outputFile, String secretKey) throws Exception {
    doCrypto(Cipher.DECRYPT_MODE, inputFile, outputFile,
secretKey);
```

Comprimir Archivos

- ***Función:*** Reducción del tamaño de los archivos para almacenamiento o transferencia eficientes.
- ***Implementación:*** Utiliza GZIP para comprimir y descomprimir archivos. El método comprimirArchivo toma la ruta del archivo de entrada y la ruta del archivo comprimido de salida, comprimiendo el contenido del archivo de entrada y escribiéndolo en el archivo de salida.
- ***Uso:*** Métodos comprimirArchivo y descomprimirArchivo en la clase FileCompressor.

```
public static void comprimirArchivo(String inputFile, String
outputFile) throws IOException {
    try (FileInputStream fis = new FileInputStream(inputFile);
         FileOutputStream fos = new
FileOutputStream(outputFile);
         GZIPOutputStream gzipOS = new GZIPOutputStream(fos))
{
        byte[] buffer = new byte[1024];
        int len;
        while ((len = fis.read(buffer)) != -1) {
            gzipOS.write(buffer, 0, len);
        }
    }
}}
```

Descomprimir Archivos

- ***Función:*** Reversión de la compresión para obtener el contenido original del archivo.
- **Implementación:** Utiliza GZIP para descomprimir archivos. El método descomprimirArchivo toma la ruta del archivo comprimido de entrada y la ruta del archivo de salida descomprimido, descomprimiendo el contenido del archivo de entrada y escribiéndolo en el archivo de salida.
- ***Uso:*** Realizado mediante el método descomprimirArchivo.

```
public static void descomprimirArchivo(String inputFile,
String outputFile) throws IOException {
    try (FileInputStream fis = new FileInputStream(inputFile);
         GZIPInputStream gzipIS = new GZIPInputStream(fis);
         FileOutputStream fos = new
FileOutputStream(outputFile)) {
        byte[] buffer = new byte[1024];
        int len;
        while ((len = gzipIS.read(buffer)) != -1) {
            fos.write(buffer, 0, len);
        }
    }
}
```

Librerías Utilizadas

Swing: Se utilizó para crear la interfaz gráfica del usuario. Proporciona componentes estándar como botones, campos de texto, áreas de texto, así como elementos para la visualización de árboles.

javax.swing.JOptionPane: Para mostrar cuadros de diálogo que informan al usuario sobre las operaciones realizadas, como los recorridos de los árboles.

javax.swing.tree.DefaultMutableTreeNode: Para representar los nodos de los árboles en la interfaz gráfica, permitiendo una visualización jerárquica de los datos.

java.awt: Utilizado para la creación y manipulación de elementos gráficos y eventos.

java.io: Proporciona clases esenciales para el manejo de entrada y salida, como la lectura y escritura de archivos.

java.security: Incluye clases para la encriptación y desencriptación de archivos.

javax.crypto: Ofrece clases y algoritmos necesarios para la criptografía avanzada, utilizada en las funcionalidades de encriptación y desencriptación.

java.util.logging: Utilizada para el manejo y registro de eventos y excepciones en el sistema.

Código completo

Ver código.

Conclusión

El programa desarrollado representa una herramienta robusta y versátil para la manipulación y visualización de datos mediante el uso de estructuras de datos avanzadas, como árboles binarios de búsqueda (ABB) y árboles AVL. Su capacidad para cargar datos desde archivos, visualizar árboles en diferentes órdenes de recorrido, y realizar operaciones de encriptación, desencriptación, compresión y descompresión de archivos, lo convierten en una solución integral para el manejo de grandes volúmenes de datos de manera segura y eficiente.

La interfaz gráfica, construida utilizando la biblioteca Swing de Java, facilita una interacción intuitiva y accesible para el usuario. Esto permite la visualización jerárquica de los árboles y la ejecución de diversas operaciones mediante un entorno amigable.

El manual técnico elaborado proporciona una guía exhaustiva sobre el funcionamiento interno del programa, detallando las estructuras de datos, los algoritmos implementados, y las bibliotecas utilizadas. Además, incluye instrucciones claras para la instalación, uso, y resolución de problemas, lo que asegura que tanto desarrolladores como usuarios finales puedan comprender y utilizar la aplicación con facilidad.