Manual Tecnico

Proyecto Final Estructura de Datos

Índice

Archivos

ProyectoFinalEDD.java

main()

```java
public class ProyectoFinalEDD {

    public static void main(String[] args) {
        Controlador controlador = new Controlador();
        Inicio iniciar = new Inicio(controlador);
        controlador.setInicio(iniciar);
        iniciar.setLocationRelativeTo(null);
        iniciar.setVisible(true);
    }
}
```

Metodo main(), este método es el encargado de iniciar con el programa, ya que contiene el método main. Crea el objeto controlador, el cual es importante por que contiene la mayoria de la lógica del programa.

Controlador.java

constructor()

```java
public Controlador() {
    this.bibliotecas = new ListaEnlazadaDoble<>();
    this.libros = new ListaEnlazadaDoble<>();
    this.conexiones = new ListaEnlazadaDoble<>();
    this.erroresCargaArchivos = new ListaEnlazadaDoble<>();
    this.grafoBibliotecas = new GrafoBiblioteca(this);
    this.arbolAVL = new ArbolAVL();
    this.arbolB = new ArbolB(3);
    this.arbolBMas = new ArbolBMas(3);
    this.tablaHashLibros = new TablaHashLibros(10);
}
```

Este es el método controlador, el cual inicializa unas listas enlazadas las cuales son 4, las cuales almacenan: bibliotecas, libros, conexiones entre bibliotecas y errores generados cuando se cargan los archivos csv.

archivoCSV()

```java
/**
 * Funcion encargada de poder recolectar un archivo con extension .csv
 *
 * @return retorna el archivo seleccionado por el usuario
 * @throws ExceptionBibliotecaMagica lanza excepcion por algun error de
 * archivo
 */
private File archivoCSV() throws ExceptionBibliotecaMagica {
    this.fileChooser = new FileChooser();
    String pathSCV = this.fileChooser.seleccionarArchivoCSV();
    //Validar que se haya seleccionado un archivo
    if (pathSCV == null) {
        throw new ExceptionBibliotecaMagica("Ningun archivo seleccionado.");
    }
    File fileCSV = new File(pathSCV);

    // Validar extensión .csv
    if (!fileCSV.getName().toLowerCase().endsWith(".csv")) {
        throw new ExceptionBibliotecaMagica("El archivo seleccionado no tiene extensión .csv");
    }
    return fileCSV;
}
```

Este método se encarga de abrir un gestor de archivos el cual podrá seleccionar un archivo con extensión csv, retorna un objeto de tipo File.

cargarLibros()

```java
/**
 * Metodo que carga los libros desde un archivo con extension csv
 */
public void cargarLibros() {
   this.erroresCargaArchivos.eliminarLista();

   try {
       File archivo = this.archivoCSV();

       try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
           String linea;
           int numeroLinea = 0;

           while ((linea = br.readLine()) != null) {
               numeroLinea++;

               // Saltar encabezado o líneas vacías
               if (numeroLinea == 1 || linea.trim().isEmpty()) {
                   continue;
               }

               String[] datos = linea.split(",");
               if (datos.length < 9) {
                   System.out.println("⚠Línea ignorada (formato incorrecto): " + linea);
                   this.erroresCargaArchivos.agregarValorAlFinal("Línea [" + numeroLinea + "] ignorada (formato incorrecto): " + linea);
                   continue;
               }

               if (!this.ISBNValido(datos[1], datos[0])) {
                   this.erroresCargaArchivos.agregarValorAlFinal("Línea [" + numeroLinea + "] ignorada (ISBN incorrecto): " + linea);
                   continue;
               }

               try {
                   // Coincidir con el orden del encabezado del CSV
                   String titulo = datos[0].trim();
                   String isbn = datos[1].trim();
                   String genero = datos[2].trim();
                   int anio = Integer.parseInt(datos[3].trim());
                   String autor = datos[4].trim();
                   String estado = datos[5].trim();
                   String idOrigen = datos[6].trim();
                   String idDestino = datos[7].trim();
                   String prioridad = datos[8].trim();
```

Este método es el encargado de recolectar los datos que se encuentran en el archivo csv y almacenarlos en objetos de tipo libro.

almacenarLibrosCargados()

```
private void almacenarLibrosCargados() {
    NodoListaEnlazadaDoble<Libro> actualLibro = this.libros.getInicio();
    while (actualLibro != null) {
        NodoListaEnlazadaDoble<NodoGrafo> actualNodoGrafo = this.grafoBibliotecas.getNodosGrafo().getInicio();
        while (actualNodoGrafo != null) {
            //Condicional para verificar si la biblioteca origen del libro coincide con alguna de las del grafo
            if (actualLibro.getDato().getBibliotecaDestino().getId().equals(actualNodoGrafo.getDato().getBiblioteca().getId())) {
                actualNodoGrafo.getDato().getBiblioteca().getArbolAVL().insertar(actualLibro.getDato());
                actualNodoGrafo.getDato().getBiblioteca().getArbolB().insert(actualLibro.getDato());
                actualNodoGrafo.getDato().getBiblioteca().getArbolBMas().insertar(actualLibro.getDato());
                actualNodoGrafo.getDato().getBiblioteca().getTablaHash().insertar(actualLibro.getDato());
                actualNodoGrafo.getDato().getBiblioteca().getListaEnlazada().agregarValorAlFinal(actualLibro.getDato());
                break;
            }
            actualNodoGrafo = actualNodoGrafo.getSiguiente();
        }
        actualLibro = actualLibro.getSiguiente();
    }

}
```

Este método es el encargado de almacenar todos los libros en cada una de las estructuras que cada biblioteca contiene.

ISBNValido()

```
/**
 * Funcion que indica si el isbn es valido: si existe pero con el mismo
 * titulo es valido, si existe pero con otro titulo se rechaza
 *
 * @param isbn isbn a comparar con los ya registrados
 * @param titulo titulo del libro a comparar
 * @return retorna verdadero si el isbn es valido o falso si es invalido
 */
private boolean ISBNValido(String isbn, String titulo) {
    NodoListaEnlazadaDoble<Libro> actual = this.libros.getInicio();
    while (actual != null) {
        Libro existente = actual.getDato();
        if (existente.getISBN().equalsIgnoreCase(isbn)) {
            // ISBN ya existe, verificar título
            if (!existente.getTitulo().equalsIgnoreCase(titulo)) {
                // ISBN duplicado con otro título → error
                return false;
            }
        }
        actual = actual.getSiguiente();
    }
    return true; // No hay conflicto
}
```

Este método se encarga de validar un isbn durante la carga de los archivos con extensión csv.

cargarBibliotecas()

```java
public void cargarBibliotecas() throws ExceptionBibliotecaMagica {
    File archivo = archivoCSV();
    this.erroresCargaArchivos.eliminarLista();

    try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
        String linea;
        int numeroLinea = 0;

        while ((linea = br.readLine()) != null) {
            numeroLinea++;

            // Saltar encabezado o líneas vacías
            if (numeroLinea == 1 || linea.trim().isEmpty()) {
                continue;
            }

            String[] datos = linea.split(",");

            //Verica que la linea a analizar tenga la cantidad de campos correctos
            if (datos.length < 6) {
                System.out.println("⚠Línea ignorada (formato incorrecto): " + linea);
                this.erroresCargaArchivos.agregarValorAlFinal("Error de formato incorrecto en linea: [" + numeroLinea + "] " + linea);
                continue;
            }

            //Verifica que el id de la nueva biblioteca no se repita con otra ya ingresada
            if (!validoIdBiblioteca(datos[0])) {
                this.erroresCargaArchivos.agregarValorAlFinal("Error de ID Biblioteca en linea: [" + numeroLinea + "] " + linea);
                continue;
            }
```

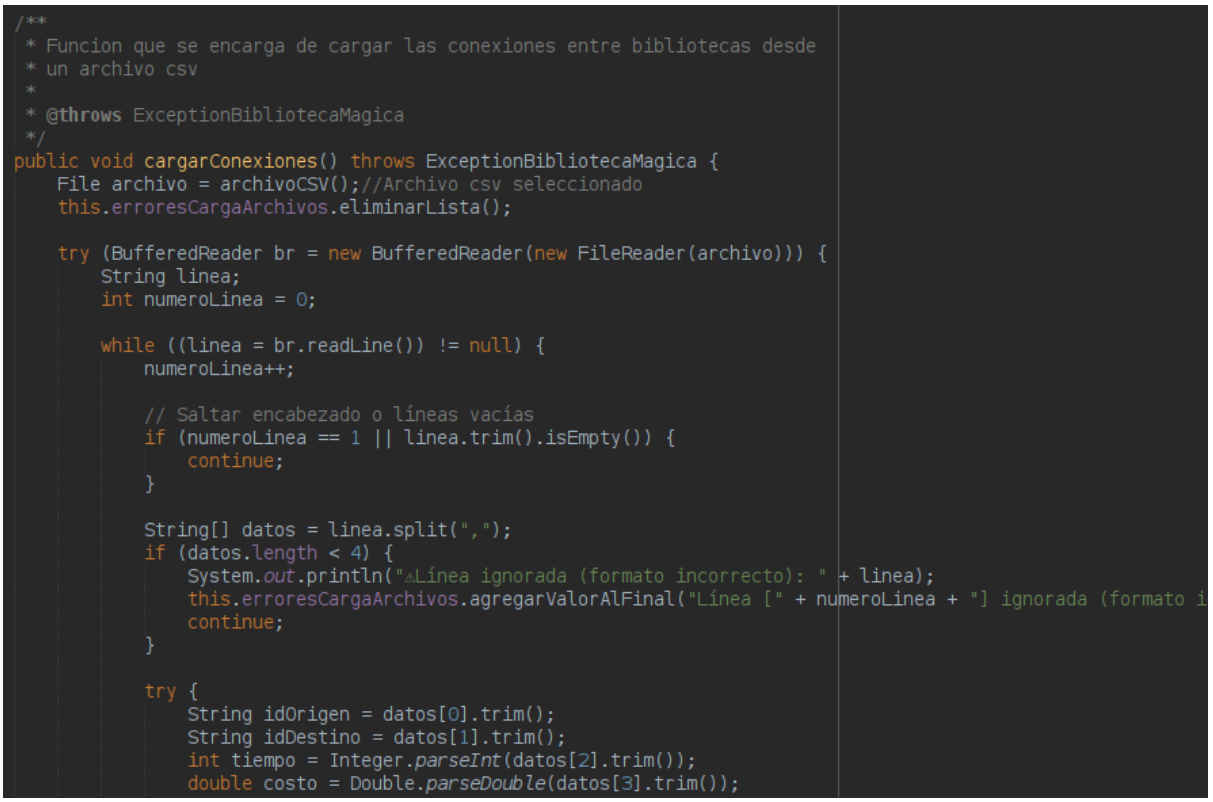
Este método es el encargado de recolectar los datos que se encuentran en el archivo csv y almacenarlos en objetos de tipo biblioteca.

validoIdBiblioteca(String idBiblioteca)

```java
/**
 * Funcion que valida si un id esta o no repetido dentro de la lista
 * enlazada de bibliotecas
 *
 * @param idBiblioteca el id de la biblioteca que se va a ingresar
 * @return retorna verdadero o falso si el id existe o no
 */
private boolean validoIdBiblioteca(String idBiblioteca) {
    NodoListaEnlazadaDoble<Biblioteca> actual = this.bibliotecas.getInicio();
    while (actual != null) {
        Biblioteca existente = actual.getDato();
        if (existente.getId().equalsIgnoreCase(idBiblioteca)) {
            return false;
        }
        actual = actual.getSiguiente();
    }
    return true;
}
```

Este método es el encargado de validar los id de los bibliotecas, retorna un valor booleano indicando si el id de la biblioteca es válido o no.

cargarConexiones()



```java
/**
 * Funcion que se encarga de cargar las conexiones entre bibliotecas desde
 * un archivo csv
 *
 * @throws ExceptionBibliotecaMagica
 */
public void cargarConexiones() throws ExceptionBibliotecaMagica {
    File archivo = archivoCSV();//Archivo csv seleccionado
    this.erroresCargaArchivos.eliminarLista();

    try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
        String linea;
        int numeroLinea = 0;

        while ((linea = br.readLine()) != null) {
            numeroLinea++;

            // Saltar encabezado o líneas vacías
            if (numeroLinea == 1 || linea.trim().isEmpty()) {
                continue;
            }

            String[] datos = linea.split(",");
            if (datos.length < 4) {
                System.out.println("⚠Línea ignorada (formato incorrecto): " + linea);
                this.erroresCargaArchivos.agregarValorAlFinal("Línea [" + numeroLinea + "] ignorada (formato i
                continue;
            }

            try {
                String idOrigen = datos[0].trim();
                String idDestino = datos[1].trim();
                int tiempo = Integer.parseInt(datos[2].trim());
                double costo = Double.parseDouble(datos[3].trim());
```

Este método es el encargado de cargar las conexiones que van a tener las bibliotecas con otras bibliotecas.

Metodos getters() y setters()

```java
public Inicio getInicio() {
    return inicio;
}

public void setInicio(Inicio inicio) {
    this.inicio = inicio;
}

public GrafoBiblioteca getGrafoBibliotecas() {
    return grafoBibliotecas;
}

public void setGrafoBibliotecas(GrafoBiblioteca grafoBibliotecas) {
    this.grafoBibliotecas = grafoBibliotecas;
}

public ListaEnlazadaDoble<Libro> getLibros() {
    return libros;
}

public void setLibros(ListaEnlazadaDoble<Libro> libros) {
    this.libros = libros;
}

public ListaEnlazadaDoble<Biblioteca> getBibliotecas() {
    return bibliotecas;
}

public void setBibliotecas(ListaEnlazadaDoble<Biblioteca> bibliotecas) {
    this.bibliotecas = bibliotecas;
}

public ListaEnlazadaDoble<Conexion> getConexiones() {
    return conexiones;
}

public void setConexiones(ListaEnlazadaDoble<Conexion> conexiones) {
    this.conexiones = conexiones;
}

public boolean isHayBibliotecas() {
    return hayBibliotecas;
}

public boolean isHayConexiones() {
    return hayConexiones;
}
```

Metodos getters y setters para poder acceder a los atributos de la clase.