Criterio C: Desarrollo

Programación orientada a objetos

Tratando de alcanzar el primer criterio de logro que se refiere a la creación de elementos “Libro” bajo los atributos “ID”, “Ubicación”, “Título”, “Lector”, “Autor” y “Tipo”, la utilización de objetos es la base para el funcionamiento del programa, pues se debe crear un objeto “Libro” que sea capaz de contener los datos Para ello se creó la clase “Libro”:

```
public class Libro {

    private int id;
    private String titulo;
    private String autor;
    private String ubicacion;
    private String lector;
    private String tipo;

    public Libro(int id, String titulo, String autor, String ubicacion, String lector, String tipo)
        this.id = id;
        this.titulo = titulo;
        this.autor = autor;
        this.ubicacion = ubicacion;
        this.lector = lector;
        this.tipo = tipo;
    }

    public int getId() {
        return id;
    }
```

A partir de los distintos atributos de cada objeto y sus respectivos métodos setters y getters, se pueden llevar a cabo algoritmos de búsqueda, modificación, etc. A continuación, se muestra la creación de objetos.

```
Libro libro, aux = new Libro(0,null,null,null,null,null);
```

A su vez, a través del programa es necesario visualizar cada uno de los atributos de cada objeto “Libro”, como se muestra a continuación.

En las siguientes técnicas, se abordará más a fondo la aplicación de programación orientada a objetos.

Lista Encadenada

Para lograr la realización de registros manipulables (Criterio de logro 1) y lo referente al criterio de logro 3, las listas encadenadas utilizadas en el código tienen el propósito de contener todos los objetos tipo “Libro” que contenga el inventario.

Para la utilización de Listas Encadenadas, primeramente, se debe de importar la siguiente librería e inicializar una lista:

```
import java.util.ArrayList;

public class VerInventario extends javax.swing.JFrame {

        private ArrayList<Libro> ListaLibros;
```

A continuación, se observa un fragmento del método MostrarTabla()

```java
public void MostrarTabla(){
    try{

    ListaLibros = new ArrayList<Libro>();

    File Archivo_txt = new File("Libros.txt");
    FileReader Fr;
    BufferedReader Br;
    Libro libro, aux = new Libro(0,null,null,null,null,null);

        Fr = new FileReader(Archivo_txt);
        Br = new BufferedReader(Fr);
        libro = aux.obtenerLibro(Br);

        while(libro != null){
            ListaLibros.add(libro);
            libro = aux.obtenerLibro(Br);
        }
    Br.close();
    Fr.close();
```

Dentro del método, se crea la Lista Encadenada “ListaLibros” que lee los datos del archivo de texto y los almacena a través de la utilización de un ciclo while en conjunto con un BufferedReader, un FileReader. Con esto se logra que la “ListaLibros” contenga cada uno de los libros del inventario

Ahora bien, para añadir un libro nuevo a la ListaLibros que no es perteneciente al archivo de texto, se debe utilizar el siguiente código:

```
String titulo = titulo_txt.getText();
String autor = autor_txt.getText();
String ubicacion = ubicacion_txt.getText();

String lector = null;
if(niño_boton.isSelected()) {
    lector = "Niño";
}
if(joven_boton.isSelected()) {
    lector = "Joven";
}
if(adulto_boton.isSelected()) {
    lector = "Adulto";
}
String tipo = null;
if(libro_boton.isSelected()) {
    tipo = "Libro";
}
if(academico_boton.isSelected()) {
    tipo = "Académico";
}
if(revista_boton.isSelected()) {
    tipo = "Revista";
}
if(otro_boton.isSelected()) {
    tipo = otro_txt.getText();

}
```

En este código se toman los valores de cada uno de los atributos pertenecientes al objeto y posteriormente se emplea las siguientes líneas para añadirlo a la ListaLibros:

```
if(titulo != null && autor != null && ubicacion != null && lector != null && tipo != null){
    ListaLibros.add(new Libro(id,titulo,autor,ubicacion,lector,tipo));
```

Ahora bien, cuando la “ListaLibros” está completa se crea un nuevo objeto temporal “x”, que tomará los valores de cada libro en la “ListaLibros” y los insertará en la matriz “Mat[][]” **[1].** A partir de la matriz creada, se utiliza la función “setModel” que tomará los datos de la matriz y los añadirá en la JTable_Libros **[2]** Esto permitirá la visualización de cada objeto junto con sus atributos pertenecientes a la lista encadenada

```
public void MostrarTabla(){
    try{

    ListaLibros = new ArrayList<Libro>();

    File Archivo_txt = new File("Libros.txt");
    FileReader Fr;
    BufferedReader Br;
    Libro libro, aux = new Libro(0,null,null,null,null,null);

        Fr = new FileReader(Archivo_txt);
        Br = new BufferedReader(Fr);
        libro = aux.obtenerLibro(Br);

        while(libro != null){
            ListaLibros.add(libro);
            libro = aux.obtenerLibro(Br);
        }
    Br.close();
    Fr.close();


String Mat[][] = new String[ListaLibros.size()][6];
Libro x;
for(int i = 0; i < ListaLibros.size(); i++){
    x = ListaLibros.get(i);
    Mat[i][0] = Integer.toString(x.getId());
    Mat[i][1] = x.getTitulo();
    Mat[i][2] = x.getAutor();
    Mat[i][3] = x.getUbicacion();
    Mat[i][4] = x.getLector();
    Mat[i][5] = x.getTipo();
}

JTable_Libros.setModel(new javax.swing.table.DefaultTableModel(
        Mat,
    new String [] {
        "ID", "Título", "Autor", "Ubicación", "Lector", "Tipo"
    }

));
```

[1]

[2]

Finalmente, se crea la JTable dentro de la siguiente pantalla.

Posteriormente, para eliminar un objeto de la lista, se debe seleccionar un libro en la JTable_Libros, a partir de esto con la función “remove” se eliminará directamente el objeto perteneciente a la fila y se renovará la matriz sin el objeto.

```
NúmeroDeFila = JTable_Libros.getSelectedRow();
ListaLibros.remove(NúmeroDeFila);
String Mat[][] = new String[ListaLibros.size()][6];
```

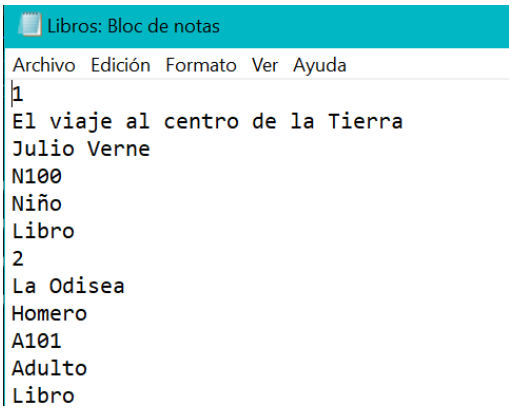
Cabe destacar que, cuando se renueva la matriz, los datos pertenecientes a los mismo son actualizados en el archivo de texto y toda la información es guardada como se explicará en el siguiente apartado.

Manejo de archivos

El manejo de archivos es sustancial para el programa (siguiendo los criterios de logro 1 y 3) ya que permite el almacenamiento de datos fuera de la aplicación, pues de otra forma se borraría la información al cerrar el programa.

Archivos de texto

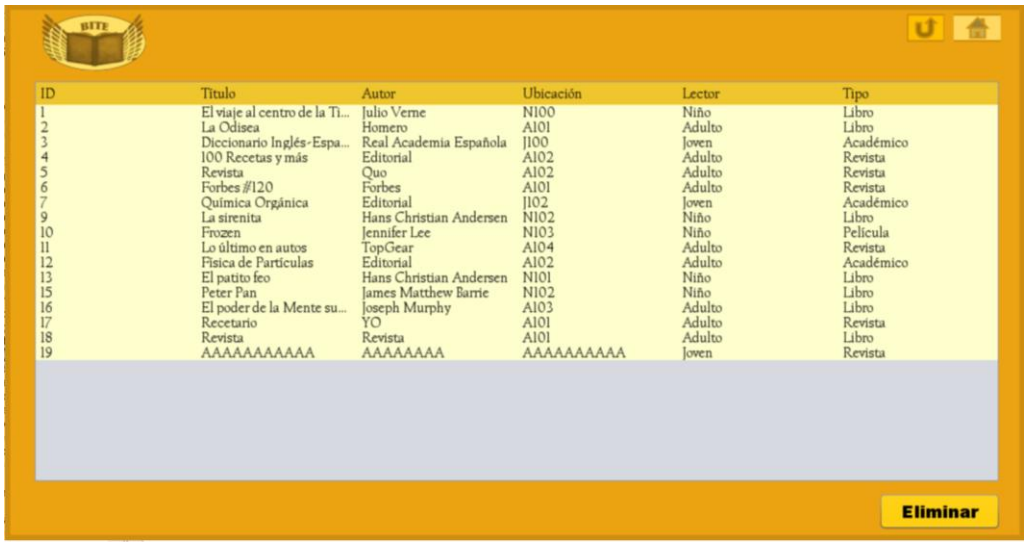
Debido a que cada objeto contiene seis atributos (folio, título, autor, ubicación, lector, y tipo), cada uno de estos representa una línea del archivo de texto, desembocando en el siguiente modelo:



A partir de este modelo surgen los métodos que vuelven posible la lectura y manipulación de datos dentro del archivo, siendo uno de los principales el método antes mencionado “MostrarTabla()”

```
public void MostrarTabla(){
    try{

    ListaLibros = new ArrayList<Libro>();

    File Archivo_txt = new File("Libros.txt");
    FileReader Fr;
    BufferedReader Br;
    Libro libro, aux = new Libro(0,null,null,null,null,null);

        Fr = new FileReader(Archivo_txt);
        Br = new BufferedReader(Fr);
        libro = aux.obtenerLibro(Br);

        while(libro != null){
            ListaLibros.add(libro);
            libro = aux.obtenerLibro(Br);
        }
    Br.close();
    Fr.close();
```

El método anterior crea la “ListaLibros” a partir de los objetos almacenados en el archivo de texto; “Libros.txt”. Para esto se utiliza FileReader y BufferedReader al igual que objetos temporales y un ciclo while que se repetirá obtener todos los objetos. A continuación se muestra la pantalla después de que se ejecuta el método MostrarTabla().



Cabe mencionar que se llama al método “obtenerLibro” que es aquel que permite leer línea por línea y determinar el valor de cada atributo dependiendo de la línea del objeto

```
public Libro obtenerLibro(BufferedReader Br) {

    int folio;
    String tit,aut,ubi,lec,tip;

    try{

        folio = Integer.parseInt(Br.readLine());
        tit = Br.readLine();
        aut = Br.readLine();
        ubi = Br.readLine();
        lec = Br.readLine();
        tip = Br.readLine();

        return new Libro(folio,tit,aut,ubi,lec,tip);

    }catch (Exception e){

    }return null;
}
```

En caso de añadir nuevos objetos en el archivo de texto, se debe utilizar PrintWriter, esto se realiza a través del método guardarLibros() que se muestra a continuación:

```
public void guardarLibros(){
    File file = new File("Libros.txt");
    PrintWriter Pw;

    try{

        Libro libro;
        Pw = new PrintWriter(file, "utf-8");
        for(int i = 0; i < ListaLibros.size(); i++){
            libro = ListaLibros.get(i);
            libro.guardarLibro(Pw);
        }
        Pw.close();
    }catch (Exception e){

}
}
```

Como se puede observar se vuelven a imprimir todos los objetos Libro con sus respectivos atributos de la ListaLibros en el archivo de texto y se guardan a través del siguiente método

```
void guardarLibro(PrintWriter Pw) {
    Pw.println(id);
    Pw.println(titulo);
    Pw.println(autor);
    Pw.println(ubicacion);
    Pw.println(lector);
    Pw.println(tipo);
}
```

En caso de que no exista el archivo de texto, se crea uno a partir de lo siguiente.

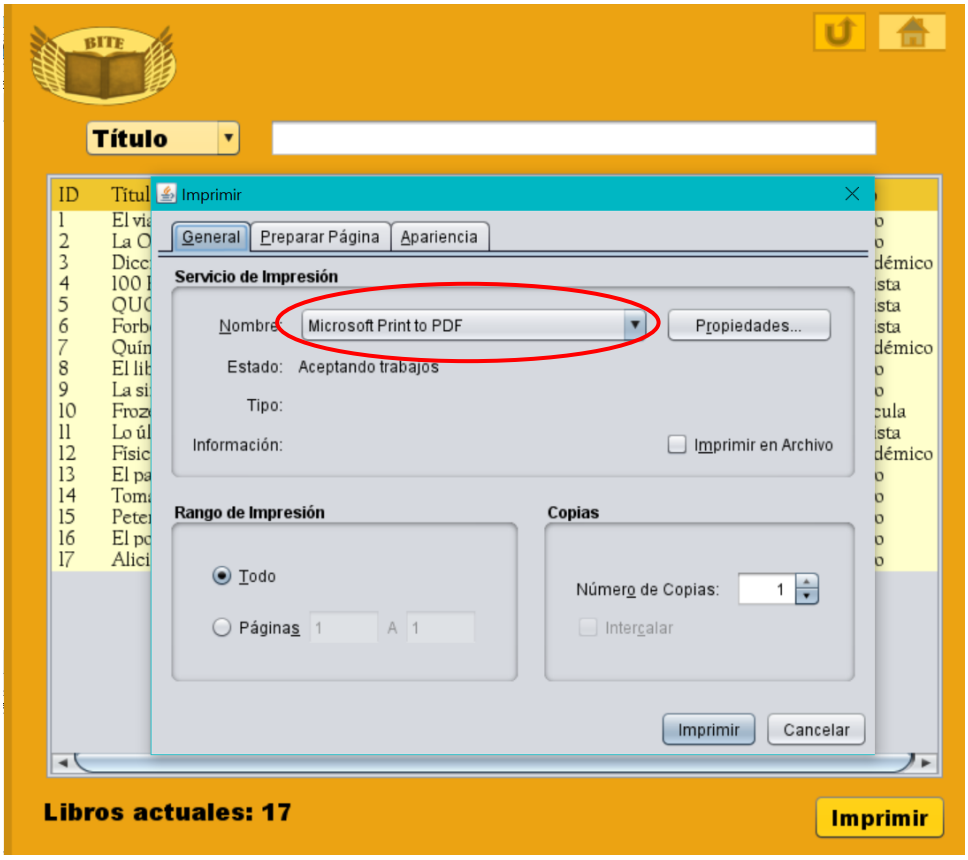
```
try{

File Archivo_txt = new File("Libros.txt");
if(!Archivo_txt.exists()){
        Archivo_txt.createNewFile();
}
```

Archivo PDF:

Para cumplir con el criterio de logro 3 (impresión del inventario), se crea un archivo PDF cuya única función es al momento de exportar los datos y tiene como propósito la impresión de los mismos. Como se puede observar en el siguiente código, se exporta la JTable_Libros dentro de la clase "VerInventario.java"

```
private void imprimir_btnActionPerformed(java.awt.event.ActionEvent evt) {

    MessageFormat titulo = new MessageFormat("Libros existente en bodega");
    try{
        JTable_Libros.print(JTable.PrintMode.NORMAL, titulo, null, true, null, true);
    }catch(java.awt.print.PrinterException e){
        System.err.format("No se puede imprimir", e.getMessage());
    }
```

Al seleccionar dicho botón, surge el menú de impresión que exporta los datos a un archivo PDF



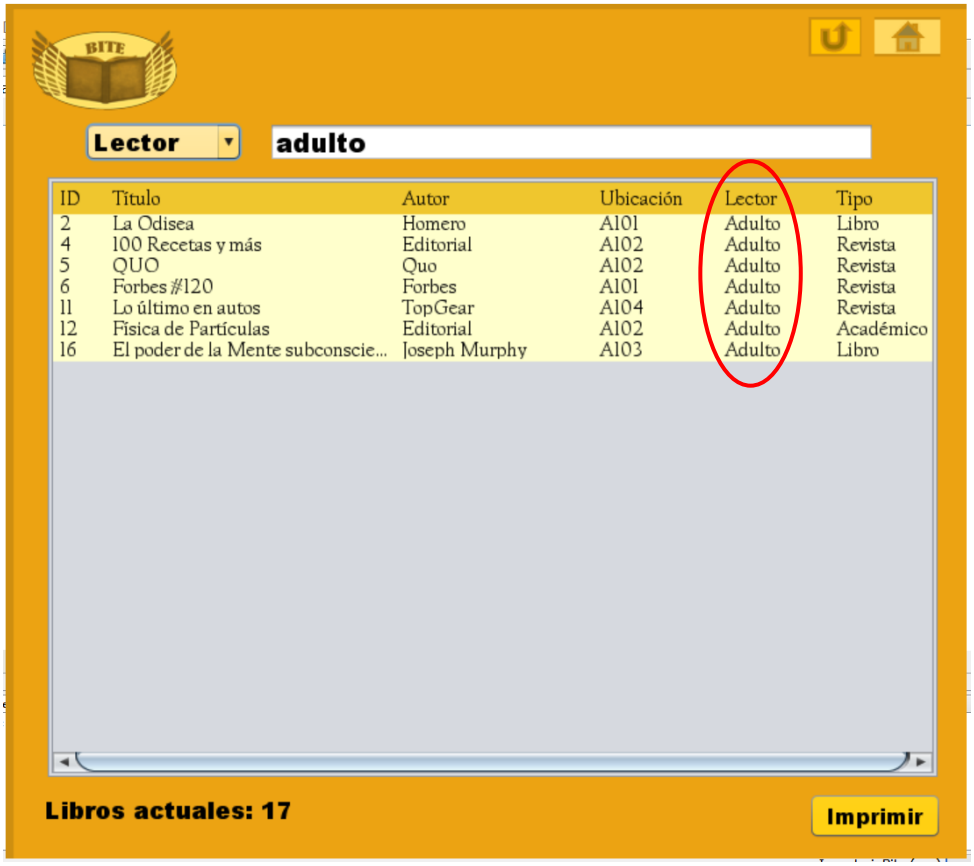
Exportándose de la siguiente manera:

Libros existente en bodega

ID	Título	Autor	Ubicación	Lector	Tipo
1	El viaje al centro de la Tierra	Julio Verne	N100	Niño	Libro
2	La Odisea	Homero	A101	Adulto	Libro
3	DiccionarioInglés-Español	Real Academia Espa...	J100	Joven	Académico
4	100 Recetas y más	Editorial	A102	Adulto	Revista
5	QUO	Quo	A102	Adulto	Revista
6	Forbes #120	Forbes	A101	Adulto	Revista
7	Química Orgánica	Editorial	J102	Joven	Académico
8	El libro de la selva	Rudyard Kipling	N101	Niño	Libro
9	La sirenita	HansChristianAnde...	N102	Niño	Libro
10	Frozen	Jennifer Lee	N103	Niño	Película
11	Lo último en autos	TopGear	A104	Adulto	Revista
12	Física de Partículas	Editorial	A102	Adulto	Académico
13	El patito feo	HansChristianAnde...	N101	Niño	Libro
14	Tomás y su amigos	Wilbert Audry	N101	Niño	Libro
15	Peter Pan	JamesMatthewBarrie	N102	Niño	Libro
16	El poder de la Mente subconscie...	Joseph Murphy	A103	Adulto	Libro
17	Alicia en el país de las maravillas	LewisCarroll	J102	Joven	Libro

Método de búsqueda

La búsqueda dentro de la aplicación (criterio de logro 3) consiste en filtrar los libros dependiendo del atributo solicitado. En el siguiente ejemplo se observa que el atributo solicitado es “adulto” dentro de la variable “lector”.



Para esto, primero se importaron las siguientes librerías, e igualmente se inicializó un TableRowSorter.

```
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import javax.swing.RowFilter;
import javax.swing.table.TableRowSorter;

public class VerInventario extends javax.swing.JFrame {

        private ArrayList<Libro> ListaLibros;
        private TableRowSorter Filtro;
```

Partiendo de esto, se utiliza la función “addKeyListener”, “keyReleased” y se obtiene la columna a la que pertenece el criterio solicitado:

```
filtrar_txt.addKeyListener(new KeyAdapter(){
    public void keyReleased(final KeyEvent e){


        String opcion = filtrar_box.getSelectedItem().toString();
        int i = 0;

        if(opcion.equals("Título")){
            i = 1;
        }
        if(opcion.equals("Autor")){
            i = 2;
        }
        if(opcion.equals("Ubicación")){
            i = 3;
        }
        if(opcion.equals("Lector")){
            i = 4;
        }
        if(opcion.equals("Tipo")){
            i = 5;
        }
        FiltrarLibros(i);
    }
```

Una vez que se obtiene el número de la columna, se llama al método “FiltrarLibros” que buscará todas las coincidencias del atributo introducido dependiendo de la columna a la que se está referenciando.

```
public void FiltrarLibros(int i){
    int ColumnaDeFiltro = i;

    if(ColumnaDeFiltro == 0){
      Filtro.setRowFilter(RowFilter.regexFilter(filtrar_txt.getText(), ColumnaDeFiltro));
    }else{
      Filtro.setRowFilter(RowFilter.regexFilter("(?i)" +filtrar_txt.getText(), ColumnaDeFiltro));
    }
}
```

Para finalizar se utiliza “TableRowSorter” para lograr que el modelo de la tabla contenga los libros que coincidan con el atributo solicitado

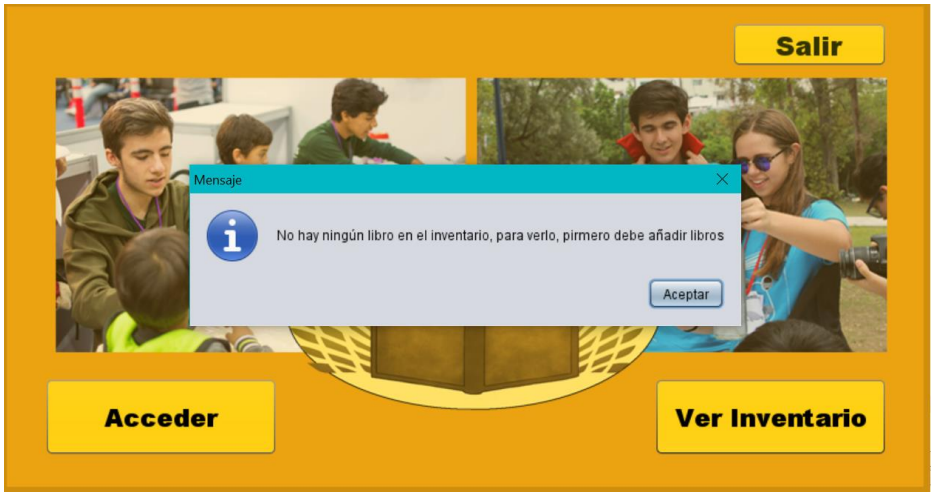
```
    Filtro = new TableRowSorter(JTable_Libros.getModel());
    JTable_Libros.setRowSorter(Filtro);
```

Validación

La utilización de excepciones es fundamental para el funcionamiento del programa (criterio de logro 1 y 3). Por ejemplo, se requiere en algunas pantallas que exista “Libros.txt”, por ello se utiliza el código:

```
private void VerInventario_btnActionPerformed(java.awt.event.ActionEvent evt) {

    try {

    File Archivo_txt = new File("Libros.txt");
    PrintWriter Pw;
    if(!Archivo_txt.exists()){
            JOptionPane.showMessageDialog(null, "No hay ningún libro en el inventario, para verlo, pirmero debe añadir libros");
    }else{

        VerInventario abrir;
        abrir = new VerInventario();
        abrir.setVisible(true);
        dispose();

    }
    } catch (IOException ex) {
        Logger.getLogger(BibliotecaDigital.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Se puede observar que, si no existe el archivo, en lugar de abrirse la ventana, solamente aparecerá un mensaje en pantalla.



Otro caso sucede cuando se requiere modificar u eliminar un libro de la JTable_Libros ya que primeramente se debe seleccionar una fila

```
try{

String TituloDelLIbroAModificar = JTable_Libros.getValueAt(JTab

    if(!titulo_txt.getText().equals("") && !autor_txt.getText()

    int opcion = JOptionPane.showConfirmDialog(this, "¿Seguro q
    if(opcion == JOptionPane.YES_OPTION){


String titulo = titulo_txt.getText(),
        autor = autor_txt.getText(),
        ubicacion = ubicacion_txt.getText(),
        lector = lector_txt.getText(),
        tipo = tipo_txt.getText();
Libro libro = null;

            int NúmeroDeFila = JTable_Libros.getSelectedRow();
            libro = ListaLibros.get(NúmeroDeFila);

            libro.setTitulo(titulo);
            libro.setAutor(autor);
            libro.setUbicacion(ubicacion);
            libro.setLector(lector);
            libro.setTipo(tipo);
```

```
}catch (Exception e){
    JOptionPane.showMessageDialog(null, "Verifique que haya seleccionado un libro a modificar");
}
```

De la misma manera, se utilizan las excepciones para asignar el valor de 1 al "ID" del primer libro añadido al archivo "Libros.txt" ya que no existe otro libro del cual éste pueda tomar referencia

```
int id = 0;
try{
id = Integer.parseInt(JTable_Libros.getModel().getValueAt(JTable_Libros.getRowCount()-1, 0).toString()) + 1;
}catch(Exception e){
    id = 1;
}
```

Número de palabras: 1,027 palabras

Referencias Bibliográficas:

- InformativaC (2018, Abril 17) Agregar, Modificar, Eliminar registro en Java Parte 2 [Archivo de video]. Recuperado de: https://www.youtube.com/watch?v=5ZxC9cnpAbk
- Informatica Jch (2016, Marzo 8) Java POO: Guardar Datos en txt y Mostrarlo en jTable [Archivo de video]. Recuperado de: https://www.youtube.com/watch?v=AH4lRZ5EvNI
- ProgrammingKnowledge (2012, Mayo 14) Java prog#26.How to print JTable in Java netbeans [Archivo de video]. Recuperado de: https://www.youtube.com/watch?v=yhqu-NG-AzY
- Shadown (2015, Noviembre 30) FILTRO JTABLE [Archivo de video] Recuperado de: https://www.youtube.com/watch?v=7CDV9IJMifw