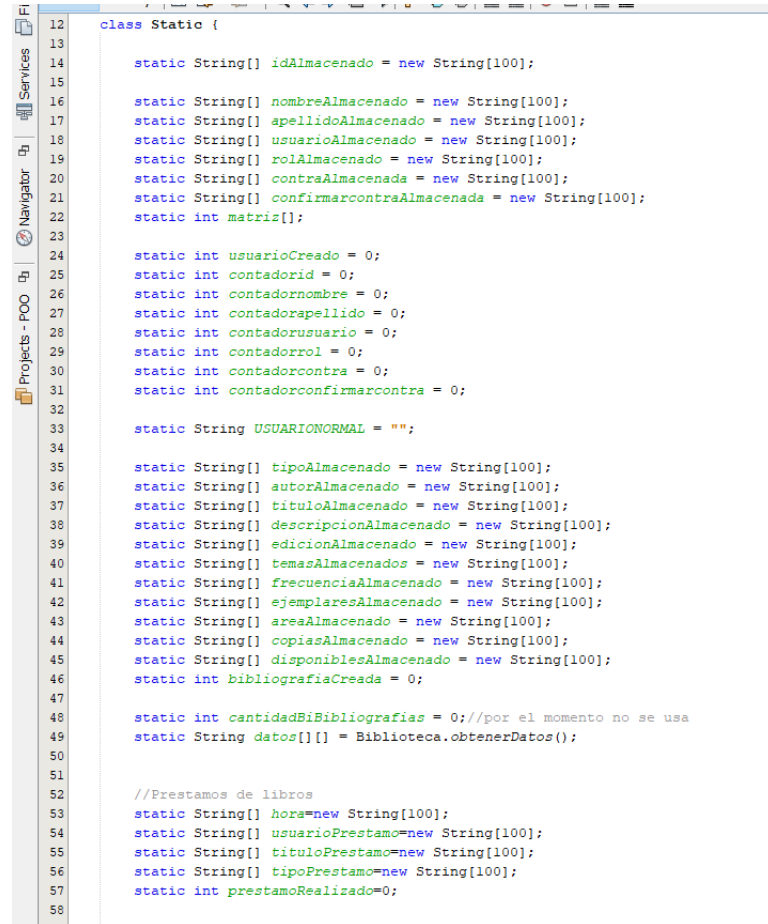


Manual Técnico

En este manual se presentarán distintas partes del código hechas para el funcionamiento del programa “Biblioteca Usac”.

Clase Static

The image shows a screenshot of an IDE with a sidebar on the left containing icons for 'Services', 'Navigator', and 'Projects - POO'. The main editor area displays the code for a class named 'Static'. The code is written in Java and includes various static variables and arrays. The line numbers 12 through 58 are visible on the left side of the code editor. The code defines several static String arrays for user data (id, nombre, apellido, usuario, rol, contra, confirmarcontra), static int variables for counters (usuarioCreado, contadorid, contadornombre, contadorsapellido, contadorausuario, contadorrol, contadorcontra, contadorconfirmarcontra), a static String constant for 'USUARIONORMAL', and several static String arrays for library data (tipoAlmacenado, autorAlmacenado, tituloAlmacenado, descripcionAlmacenado, edicionAlmacenado, temasAlmacenados, frecuenciaAlmacenado, ejemplaresAlmacenado, areasAlmacenado, copiasAlmacenado, disponiblesAlmacenado, bibliografiaCreada). It also includes a static int variable for 'cantidadBiBibliografias' and a static String array 'datos' which is assigned the value of 'Biblioteca.obtenerDatos()'. Finally, it defines static String arrays for loan data (hora, usuarioPrestamo, tituloPrestamo, tipoPrestamo) and a static int variable 'prestamoRealizado'.

Esta clase es utilizada para guardar arreglos de manera estática en el proyecto para utilizarse en distintos apartados como fueron la creación de usuarios, eliminar usuarios, modificar usuarios, mostrar usuarios, creación de bibliografías, etc.

Método para carga masiva

```
public void cargarBibliografias(String texto) {

    String[] lineasTexto = texto.split("\n");

    for (String linea : lineasTexto) {
        String[] atributos = linea.split(";");

        if (atributos.length == 11) {

            Static.tipoAlmacenado[Static.bibliografiaCreada]=atributos[0];
            Static.autorAlmacenado[Static.bibliografiaCreada]=atributos[1];
            Static.tituloAlmacenado[Static.bibliografiaCreada]=atributos[2];
            Static.descripcionAlmacenado[Static.bibliografiaCreada]=atributos[3];
            Static.edicionAlmacenado[Static.bibliografiaCreada]=atributos[4];
            // String str=atributos[5];

            // String temas = str.replace(" ", "");
            Static.temasAlmacenados[Static.bibliografiaCreada]=atributos[5];
            Static.frecuenciaAlmacenado[Static.bibliografiaCreada]=atributos[6];
            Static.ejemplaresAlmacenado[Static.bibliografiaCreada]=atributos[7];
            Static.areaAlmacenado[Static.bibliografiaCreada]=atributos[8];
            Static.copiasAlmacenado[Static.bibliografiaCreada]=atributos[9];
            Static.disponiblesAlmacenado[Static.bibliografiaCreada]=atributos[10];

            Static.bibliografiaCreada++;

        } else {
            System.out.println("Fila omitida, al no cumplir con la estructura de entrada");
        }
    }

    JOptionPane.showMessageDialog(null, "La carga ha sido completada", "Mensaje", JOptionPane.INFORMATION_MESSAGE); //Mensaje cuando finaliza la carga.

    UsuarioAdmin ventanaAdmin = new UsuarioAdmin();
    ventanaAdmin.setVisible(true);
    dispose();
    // setVisible(false); //Ocultar esta ventana(ventana de carga masiva)

}
```

Con este método se implemento el uso de “split” en las cadenas de texto, lo cual hacia que separaba distintos datos ingresados por el usuario en una área de texto para luego ser guardada en los atributos correspondientes y así permanecer almacenada en la biblioteca.

Paquete utilidades

```
package Utilidades;

/**
 *
 * @author Elvis
 */
public class MetodosSueitos {
    /**
     * Valida si una cadena es un numero entero
     *
     * @param texto String que contiene el valor a validar
     * @return True = es un numero entero
     */
    public static boolean validaNumeroEntero_Exp(String texto) {
        return texto.matches("^-?[0-9]+$");
    }

    /**
     * Valida si una cadena es un numero real (positivo o negativo)
     *
     * @param texto String que contiene el valor a validar
     * @return True = es un numero real
     */
    public static boolean validaNumeroReal_Exp(String texto) {
        return texto.matches("^-?[0-9]+([\\\\.][0-9]+)?$");
    }

    public static String [] ordenamiento(String[] words){
        for(int i = 0; i < words.length - 1; i++)
        {
            for(int j = i+1; j < words.length; j++)
            {
                if(words[i].compareTo(words[j]) > 0)//words[i] is greater than words[j]
                {
                    String temp = words[i];
                    words[i] = words[j];
                    words[j] = temp;
                }
            }
        }
        return words;
    }
}
```

En este paquete se almaceno una clase que contenía distintos métodos

Método para ordenar un arreglo de String de manera alfabética

```
public static String [] ordenamiento(String[] words){
    for(int i = 0; i < words.length - 1; i++)
    {
        for(int j = i+1; j < words.length; j++)
        {
            if(words[i].compareTo(words[j]) > 0)//words[i] is greater than words[j]
            {
                String temp = words[i];
                words[i] = words[j];
                words[j] = temp;
            }
        }
    }
    return words;
}
```

Este método es similar al método burbuja aplicado en enteros con la diferencia que se aplica en cadenas de texto, sirvió para ordenar de manera alfabética un arreglo que contenía todos los temas de las bibliografías.

Método para validar enteros

```
/*
 * Valida si una cadena es un numero entero
 *
 * @param texto String que contiene el valor a validar
 * @return True = es un numero entero
 */
public static boolean validaNumeroEntero_Exp(String texto) {
    return texto.matches("^-?[0-9]+$");
}
```

Este método se utilizó para verificar si una cadena de texto contenía un numero entero, se utilizó para validar si se ingresaban enteros en cajas de texto que pedían el ID de usuarios, la disponibilidad de bibliografías, entre otros.

Evento para ingresar como usuario Admin o usuario normal

```
public void actionPerformed(ActionEvent ae) {
    String nombre = usuario1.getText();
    String pass = contrasena1.getText();
    /*
    si contrasena ingresada es igual a un array de contrasenas que se tendran entonces
    luego otro if para saber si es usuario normal o admin
    */
    int pos = -1;
    for (int i = 0; i < 100; i++) {
        if (nombre.equals(Static.usuarioAlmacenado[i])) {
            pos = i;
        }
    }

    if (nombre.equals("admin") && pass.equals("password")) {
        UsuarioAdmin ventanaAdmin = new UsuarioAdmin();
        ventanaAdmin.setVisible(true);
        dispose();
    } else if (nombre.equals("admin") && !pass.equals("password")) {
        JOptionPane.showMessageDialog(ingresar,
            "Credenciales erroneas, por favor intente nuevamente",
            "Error",
            JOptionPane.ERROR_MESSAGE);
    } else if (pos != -1) {
        if (nombre.equals(Static.usuarioAlmacenado[pos]) && pass.equals(Static.contraAlmacenada[pos])) {
            Static.USUARIONORMAL= Static.usuarioAlmacenado[pos];
            UsuarioNormal ventanUserNormal = new UsuarioNormal();
            ventanUserNormal.setVisible(true);
            dispose();
        } else {
            JOptionPane.showMessageDialog(ingresar,
                "Credenciales erroneas, por favor intente nuevamente",
                "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(ingresar,
            "El usuario no existe, por favor contactese con el administrador",
            "Error",
            JOptionPane.ERROR_MESSAGE);
    }
};
ingresar.addActionListener(eventoIngresar);
```

En estas líneas de código se puede ver que se validaba si se ingresaban credenciales correctas y que estuvieran almacenadas en el sistema, de ser así dependiendo del tipo de usuario no redirigía a la ventana correspondiente.

Evento para poder buscar bibliografías por medio de los temas

```
@Override
public void actionPerformed(ActionEvent ae) {
    String filtro = buscarTma.getText();
    System.out.println(filtro);
    modelo = new DefaultTableModel();

    //modelo.addColumn("NO.");
    modelo.addColumn("Tipo");
    modelo.addColumn("Autor");
    modelo.addColumn("Titulo");
    modelo.addColumn("Descripcion");
    modelo.addColumn("Edicion");
    modelo.addColumn("Temas");
    modelo.addColumn("Frecuencia");
    modelo.addColumn("Ejemplares");
    modelo.addColumn("Area");
    modelo.addColumn("Copias");
    modelo.addColumn("Disponibles");
    // modelo.addColumn("Prestar");

    String[] matriz = new String[11];

    int cant = Static.bibliografiaCreada;
    for (int i = 0; i < Static.bibliografiaCreada; i++) {
        String a = Static.temasAlmacenados[i];
        String[] b = a.split(",");
        String[] x = new String[3];
        // x[0]=b[0];
        // x[1]=b[1];
        // x[2]=b[2];

        System.out.println(a);
        // for (int j = 0; j < x.length; j++) {
        if (Static.temasAlmacenados[i].toLowerCase().contains(filtro.toLowerCase())) {
            System.out.println("si se encontro");
            matriz[0] = Static.tipoAlmacenado[i];
            matriz[1] = Static.autorAlmacenado[i];
            matriz[2] = Static.tituloAlmacenado[i];
            matriz[3] = Static.descripcionAlmacenado[i];
            matriz[4] = Static.edicionAlmacenado[i];
            matriz[5] = Static.temasAlmacenados[i];
            matriz[6] = Static.frecuenciaAlmacenado[i];
            matriz[7] = Static.ejemplaresAlmacenado[i];
            matriz[8] = Static.areaAlmacenado[i];
            matriz[9] = Static.copiasAlmacenado[i];
            matriz[10] = Static.disponiblesAlmacenado[i];
        }
    }
}
```

```

105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150

String a = Static.temasAlmacenados[i];
String[] b = a.split(",");
String[] x = new String[3];
// x[0]=b[0];
// x[1]=b[1];
// x[2]=b[2];

System.out.println(a);
// for (int j = 0; j < x.length; j++) {
if (Static.temasAlmacenados[i].toLowerCase().contains(filtro.toLowerCase())) {
    System.out.println("si se encontro");
    matriz[0] = Static.tipoAlmacenado[i];
    matriz[1] = Static.autorAlmacenado[i];
    matriz[2] = Static.tituloAlmacenado[i];
    matriz[3] = Static.descripcionAlmacenado[i];
    matriz[4] = Static.edicionAlmacenado[i];
    matriz[5] = Static.temasAlmacenados[i];
    matriz[6] = Static.frecuenciaAlmacenado[i];
    matriz[7] = Static.ejemplaresAlmacenado[i];
    matriz[8] = Static.areaAlmacenado[i];
    matriz[9] = Static.copiasAlmacenado[i];
    matriz[10] = Static.disponiblesAlmacenado[i];
    modelo.addRow(matriz);
    // break;
}

// }
JTable tabla = new JTable(modelo);
//tabla.getColumnModel().getColumn(12);
// tabla.setCellEditor(tabla.getDefaultEditor(Boolean.class));

tabla.setBounds(20, 400, 1250, 300);
panel.add(tabla);

JScrollPane scroll = new JScrollPane(tabla, JScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED, JScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
scroll.setBounds(20, 400, 1250, 300);
panel.add(scroll);
}

};
btnBuscar.addActionListener(eventoFiltro);
}

```

Esto se hizo para poder mostrar una tabla con todas las bibliografías que tuvieran el tema ingresado y buscado por el usuario para así tener una mejor forma de ver que bibliografías se deseaban prestar.