

Programación Orientada a eventos

Miniproyecto 2

Edwar Yamir Forero Blanco - 202259465

Santiago Anibal Carrillo Torres - 202259664

Juan Eduardo Calderon Jaramillo - 202259671

Tercer Semestre

Tecnología en desarrollo de software-2724

Universidad del valle

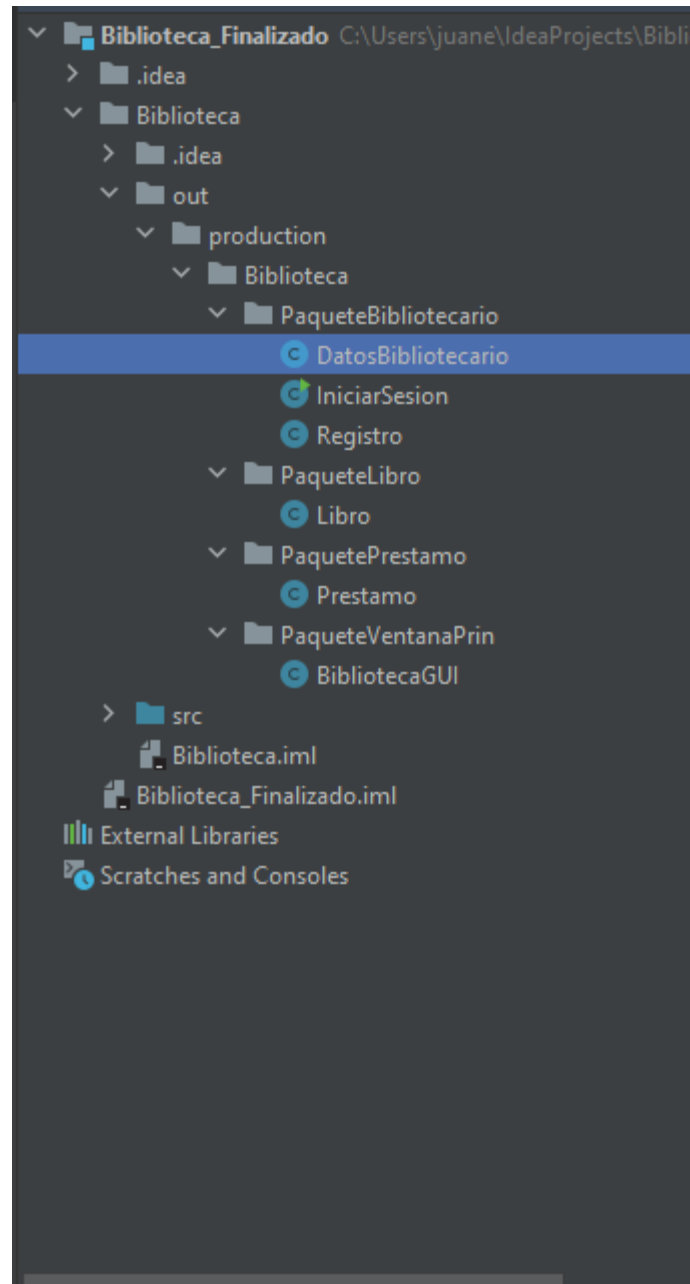
Tuluá-Valle

❖ Diagrama UML



❖ CLASES

Para llevar a cabo la construcción del proyecto se diseñaron 6 clases que se distribuyeron en 4 paquetes.



❖ PAQUETES BIBLIOTECARIO

→ DatosBibliotecario

En esta primera clase se crea para guardar el usuario y contraseña del bibliotecario jefe el cual está predefinido, pero también se guarda la de los bibliotecarios nuevos. Esta información es guardada en un hashmap. Se crean métodos que permiten recibir información y así mismo pasarla al hashmap para guardarla.

```
1  //.../
5
6  package PaqueteBibliotecario;
7
8  import ...
9
10
11 1 usage 1 inheritor
   public class DatosBibliotecario extends JFrame {
12      no usages
       private String nuevoNombre;
13      no usages
       private String nuevaContraseña;
14      no usages
       public static HashMap<String, String> agregarUsuario = new HashMap();
15
16      no usages
       public DatosBibliotecario() {
17      }
18
19      no usages
       public HashMap<String, String> BibliotecarioMaster() {
20         agregarUsuario.put("Jorge", "WF");
21         return agregarUsuario;
22     }
23
24     1 usage
       public void setNuevaSesion(String nom, String contra) {
25         this.nuevoNombre = nom;
26     }
27
28
29     5 usages
       public HashMap<String, String> nuevaSesion() {
30         agregarUsuario.put("Jorge", "WF");
31         agregarUsuario.put(this.nuevoNombre, this.nuevaContraseña);
32         return agregarUsuario;
33     }
34
35 }
```

◆ PAQUETES BIBLIOTECARIO

→ IniciarSesion

Esta clase es la que permite que el usuario jefe inicie sesión de principio se llama la clase datosBibliotecario, se crean y definen los JLabel, JTextField, JButton como también el panel, estos nos permiten darle estructura en cuanto a la visualización de la ventana.

```

1  @ /.../
2
3
4
5
6  package PaqueteBibliotecario;
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24  1 usage
25  public class IniciarSesion extends JFrame {
26      no usages
27      DatosBibliotecario datosBibliotecario = new DatosBibliotecario();
28      no usages
29      private Boolean[] usercorrecto = new Boolean[2];
30      no usages
31      JLabel lblTitulo = new JLabel("BIBLIOTECA U");
32      no usages
33      ImageIcon imagen = new ImageIcon("logo.png");
34      no usages
35      JLabel lblImagen = new JLabel();
36      no usages
37      public JTextField campoNombre;
38      no usages
39      public JPasswordField campoContrasena;
40      no usages
41      private JPanel panel;
42      no usages
43      public JButton botonConfirmar;
44      no usages

```

Luego se crea el constructor donde se define los parámetros del JFrame, también se crean otros métodos donde se define las características y posición de los objetos que van en la venta.

```

35 public void IniciarSesion() { this.login(); }
36
37
38
39 public void login() {
40     this.setTitle("Inicio de sesion: Biblioteca");
41     this.setSize(340, 460);
42     this.setLocationRelativeTo((Component)null);
43     this.Componentes();
44     this.setVisible(true);
45     this.setDefaultCloseOperation(3);
46     this.Accion();
47 }
48
49
50
51 public void Componentes() {
52     this.panel = new JPanel();
53     this.panel.setLayout((LayoutManager)null);
54     this.panel.setBackground(Color.WHITE);
55     this.getContentPane().add(this.panel);
56     this.CamposTexto();
57     this.ColocarTexto();
58     this.ColocarBoton();
59 }
60
61
62
63 public void CamposTexto() {
64     this.campoNombre = new JTextField();
65     this.campoNombre.setBounds(20, 235, 170, 20);
66 }
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

62         this.panel.add(this.campoNombre);
63         this.campoContrasena = new JPasswordField();
64         this.campoContrasena.setBounds(20, 295, 170, 20);
65         this.panel.add(this.campoContrasena);
66     }
67
68     1 usage
69     public void ColocarTexto() {
70         JLabel ingresaNombre = new JLabel();
71         ingresaNombre.setText("Nombre:");
72         ingresaNombre.setBounds(20, 220, 65, 10);
73         this.panel.add(ingresaNombre);
74         this.lblTitulo.setBounds(120, 10, 90, 25);
75         this.panel.add(this.lblTitulo);
76         this.lblImagen.setBounds(100, 50, 140, 160);
77         this.lblImagen.setIcon(new ImageIcon(this.Imagen.getImage().getScaledInstance(140, 160, 4)));
78         this.panel.add(this.lblImagen);
79         JLabel ingresaContrasena = new JLabel();
80         ingresaContrasena.setText("Contraseña:");
81         ingresaContrasena.setBounds(20, 280, 80, 10);
82         this.panel.add(ingresaContrasena);
83     }
84
85     1 usage
86     public void ColocarBoton() {
87         this.botonConfirmar = new JButton();
88         this.botonConfirmar.setText("Iniciar Sesión");
89         this.botonConfirmar.setBounds(100, 350, 120, 25);

```

En esta parte se crea un método en el cual se define las acciones del botón que permite iniciar sesión, este botón comprueba que la información ingresada sea correcta, de acuerdo con lo que hay en la clase dato bibliotecario, es decir que el login y contraseña sean las del bibliotecario jefe.

```

88         this.panel.add(this.botonConfirmar);
89     }
90
91     1 usage
92     public void Accion() {
93         ActionListener confirmarUsuario = actionPerformed(e) -> {
94             String nombre = IniciarSesion.this.campoNombre.getText();
95             String contrasena = IniciarSesion.this.campoContrasena.getText();
96             Iterator<String> iterator = IniciarSesion.this.datosBibliotecario.nuevaSesion().keySet().iterator();
97             boolean datosValidos = false;
98
99             while(iterator.hasNext()) {
100                 String nomJefe = String.valueOf(iterator.next());
101                 String contraJefe = String.valueOf(IniciarSesion.this.datosBibliotecario.nuevaSesion().get(nomJefe));
102                 if (nombre.equalsIgnoreCase(nomJefe) && contrasena.equalsIgnoreCase(contraJefe)) {
103                     datosValidos = true;
104                     break;
105                 }
106             }
107
108             if (datosValidos) {
109                 new BibliotecaGUI();
110                 IniciarSesion.this.dispose();
111             } else {
112                 JOptionPane.showMessageDialog((Component)null, "Por favor, verifique los datos");
113             }
114         };
115     }

```

Por último se llama al método main para dar inicio al programa.

```

117         this.botonConfirmar.addActionListener(confirmarUsuario);
118     }
119
120     no usages
121     public static void main(String[] args) { new IniciarSesion(); }
122 }
123
124

```

❖ PAQUETES BIBLIOTECARIO

→ Registro

En esta clase donde definimos todo el contenido y acciones dentro de la JFrame es muy similar a la clase Inicio de sesión. las diferencias se plantean mas adelante.

```
1  /.../
5
6  package PaqueteBibliotecario;
7
8  import ...
9
22
23  !usage
24  public class Registro extends DatosBibliotecario {
25      no usages
26      JLabel lblTitulo = new JLabel("BIBLIOTECA U");
27      no usages
28      ImageIcon imagen = new ImageIcon("logo.png");
29      no usages
30      JLabel lblImagen = new JLabel();
31      no usages
32      public JTextField campoNombre;
33      no usages
34      public JPasswordField campoContraseña;
35      no usages
36      private JPanel panel;
37
38      no usages
39
40      public Registro() {
41          this.setTitle("Registro de nuevo bibliotecario: Biblioteca");
42          this.setSize(340, 460);
43          this.setLocationRelativeTo((Component)null);
44          this.setVisible(true);
45          this.setDefaultCloseOperation(2);
46      }
47  }
```

```

37         this.Componentes();
38         this.CamposTexto();
39         this.ColocarTexto();
40         this.ColocarBoton();
41     }
42
43     ! usage
44     public void Componentes() {
45         this.panel = new JPanel();
46         this.panel.setLayout((LayoutManager)null);
47         this.panel.setBackground(Color.WHITE);
48         this.getContentPane().add(this.panel);
49     }
50
51     ! usage
52     public void CamposTexto() {
53         this.campoNombre = new JTextField();
54         this.campoNombre.setBounds(20, 235, 170, 20);
55         this.panel.add(this.campoNombre);
56         this.campoContrasena = new JPasswordField();
57         this.campoContrasena.setBounds(20, 295, 170, 20);
58         this.panel.add(this.campoContrasena);
59     }
60
61     ! usage
62     public void ColocarTexto() {
63         JLabel ingresaNombre = new JLabel();

```

Al igual que en la clase iniciar sesión esta clase registro permite iniciar sesión pero tiene una acción adicional y es que nos permite crear una nueva cuenta la cual es agregada al hashmap de la clase datosBibliotecario, dado lugar así a que en esta ventana podemos crear en iniciar sesión con todas las cuentas existentes y las que se creen nuevas. La primer acción listener del método ColocarBoton es la que no permite guardar una nueva cuenta.

```

61         ingresaNombre.setText("Nombre:");
62         ingresaNombre.setBounds(20, 220, 65, 10);
63         this.panel.add(ingresaNombre);
64         this.lblTitulo.setBounds(120, 10, 90, 25);
65         this.panel.add(this.lblTitulo);
66         this.lblImagen.setBounds(100, 50, 140, 160);
67         this.lblImagen.setIcon(new ImageIcon(this.imagen.getImage().getScaledInstance(140, 160, 4)));
68         this.panel.add(this.lblImagen);
69         JLabel ingresaContrasena = new JLabel();
70         ingresaContrasena.setText("Contraseña:");
71         ingresaContrasena.setBounds(20, 280, 80, 10);
72         this.panel.add(ingresaContrasena);
73     }
74
75     ! usage
76     public void ColocarBoton() {
77         JButton botonRegistrar = new JButton();
78         botonRegistrar.setText("Crear Cuenta");
79         botonRegistrar.setBounds(20, 350, 120, 25);
80         this.panel.add(botonRegistrar);
81
82         ActionListener confirmarUsuario = actionPerformed(e) -> {
83             if (Registro.this.campoNombre.getText() != "" && Registro.this.campoContrasena.getText() != "") {
84                 Registro.this.setNuevaSesion(Registro.this.campoNombre.getText(), Registro.this.campoContrasena.getText());
85                 Registro.this.campoContrasena.setText("");
86                 Registro.this.campoNombre.setText("");
87                 System.out.println("Registro");
88                 System.out.println(Registro.this.nuevaSesion());

```

La siguiente acción listener es la que permite iniciar sesión, donde comprueba que los datos ingresados sean correctos y correspondan a una cuenta existente.


```

88     } else {
89         JOptionPane.showMessageDialog((Component)null, "Digite información correcta en los campos");
90     }
91 }
92 };
93
94 botonRegistrar.addActionListener(confirmarUsuario);
95 JButton botonSesion = new JButton();
96 botonSesion.setText("Iniciar Sesion");
97 botonSesion.setBounds(180, 350, 120, 25);
98 this.panel.add(botonSesion);
99
100 ActionListener cuentaSecundaria = actionPerformed(e) -> {
101     String nombre = Registro.this.campoNombre.getText();
102     String contrasena = Registro.this.campoContrasena.getText();
103     boolean datosValidos = false;
104     Iterator<String> iterator = Registro.this.nuevaSesion().keySet().iterator();
105
106     while(iterator.hasNext()) {
107         String nomJefe = String.valueOf(iterator.next());
108         String contraJefe = String.valueOf(Registro.this.nuevaSesion().get(nomJefe));
109         if (nombre.equalsIgnoreCase(nomJefe) && contrasena.equalsIgnoreCase(contraJefe)) {
110             datosValidos = true;
111             break;
112         }
113     }
114
115     if (datosValidos) {
116         BibliotecaGUI bibliotecaGUI = new BibliotecaGUI();
117
118         bibliotecaGUI.setVisible(true);
119         Registro.this.dispose();
120     } else {
121         JOptionPane.showMessageDialog((Component)null, "Por favor, verifique los datos");
122     }
123 }
124 };
125 botonSesion.addActionListener(cuentaSecundaria);
126 }
127 }
128

```

❖ PAQUETE LIBRO

→ Libro

En la clase Libro se crearon 3 variables de tipo HashMap los cuales se usarán para guardar los nombres de los libros dependiendo de su género. Los HashMap estarán compuestos de una llave la cual es el nombre del libro y un valor que es la disponibilidad del mismo.

```
7
8 import ...
11
12 public class Libro extends JFrame {
13     no usages
14     public static HashMap<String, String> novelasClasicasH = new HashMap();
15     no usages
16     public static HashMap<String, String> terrorH;
17     no usages
18     public static HashMap<String, String> ingenieriaH;
19
20     no usages
21     public Libro() {
22     }
23
24     no usages
25     public static void eliminarNovelas(String nom) {
26         Iterator<String> iterator = novelasClasicasH.keySet().iterator();
27
28         while(iterator.hasNext()) {
29             String nomLibro = (String)iterator.next();
30             if (nomLibro.equals(nom)) {
31                 novelasClasicasH.replace(nom, "No Disponible");
32             }
33         }
34     }
35 }
```

Dentro de dicha clase se diseñaron 3 métodos de igual lógica (uno para cada género) llamados “eliminarNovelas”, “eliminarTerror” y “eliminarIngenieria”, los cuales suplirán la necesidad de cambiar el valor del HashMap cuando se preste un libro. Esto se lleva a cabo dentro de un iterador que identificará cada llave, la cual se comparará dentro de un if con el nombre del libro que ha sido prestado y de esa forma se reemplaza el valor en el HashMap.

```
27
28 }
29
30 }
31
32 no usages
33 public static void eliminarTerror(String nom) {
34     Iterator<String> iterator = terrorH.keySet().iterator();
35
36     while(iterator.hasNext()) {
37         String nomLibro = (String)iterator.next();
38         if (nomLibro.equals(nom)) {
39             terrorH.replace(nom, "No Disponible");
40         }
41     }
42 }
43
44 no usages
45 public static void eliminarIngenieria(String nom) {
46     Iterator<String> iterator = ingenieriaH.keySet().iterator();
47
48     while(iterator.hasNext()) {
49         String nomLibro = (String)iterator.next();
50         if (nomLibro.equals(nom)) {
51             ingenieriaH.replace(nom, "No Disponible");
52         }
53     }
54 }
```

En esta parte del código se aprecian los métodos “agregarNovelas”, “agregarTerror” y “agregarIngenieria” usados para identificar que el libro que se prestó con anterioridad ya ha sido devuelto y por lo tanto está disponible.

```
53     }
54
55     no usages
56     public static void agregarNovelas(String nom) {
57         Iterator<String> iterator = novelasClasicasH.keySet().iterator();
58
59         while(iterator.hasNext()) {
60             String nomLibro = (String)iterator.next();
61             if (nomLibro.equals(nom)) {
62                 novelasClasicasH.replace(nom, "Disponible");
63             }
64         }
65     }
66
67     no usages
68     public static void agregarTerror(String nom) {
69         Iterator<String> iterator = terrorH.keySet().iterator();
70
71         while(iterator.hasNext()) {
72             String nomLibro = (String)iterator.next();
73             if (nomLibro.equals(nom)) {
74                 terrorH.replace(nom, "Disponible");
75             }
76         }
77     }
78 }
```

Estos métodos iteran el HashMap seleccionado y dentro de él evalúan que el nombre del libro prestado coincida con alguna llave del Hash, si esto es verdadero, procederán a realizar el cambio del valor en el libro que ya ha sido devuelto.

```
80     no usages
81     public static void agregarIngenieria(String nom) {
82         Iterator<String> iterator = ingenieriaH.keySet().iterator();
83
84         while(iterator.hasNext()) {
85             String nomLibro = (String)iterator.next();
86             if (nomLibro.equals(nom)) {
87                 ingenieriaH.replace(nom, "Disponible");
88             }
89         }
90     }
91
92     no usages
93     public static String[] mostrarNovelas(HashMap<String, String> novela) {
94         String[] item = new String[novela.size()];
95         int i = 0;
96
97         for(Iterator<String> iterator = novela.keySet().iterator(); iterator.hasNext(); ++i) {
98             String nomLibro = (String)iterator.next();
99             item[i] = nomLibro + " - " + (String)novela.get(nomLibro);
100         }
101
102         return item;
103     }
```

Los métodos “mostrarNovelas”, “mostrarTerror” y “mostrarIngenieria” tienen la función de imprimir de manera organizada el estado de los libros, por esta razón retorna un array de tipo String. Esto lo hace debido a que dentro de un array de tipo string se agregan los componentes del HashMap para luego ser mostrados en una ventana de tipo JOptionPane.

```

104 public static String[] mostrarTerror(HashMap<String, String> terror) {
105     String[] item = new String[terror.size()];
106     int i = 0;
107
108     for(Iterator<String> iterator = terror.keySet().iterator(); iterator.hasNext(); ++i) {
109         String nomLibro = (String)iterator.next();
110         item[i] = nomLibro + " - " + (String)terror.get(nomLibro);
111     }
112
113     return item;
114 }
115
116 no usages
117 public static String[] mostrarIngenieria(HashMap<String, String> ingenieria) {
118     String[] item = new String[ingenieria.size()];
119     int i = 0;
120
121     for(Iterator<String> iterator = ingenieria.keySet().iterator(); iterator.hasNext(); ++i) {
122         String nomLibro = (String)iterator.next();
123         item[i] = nomLibro + " - " + (String)ingenieria.get(nomLibro);
124     }
125
126     return item;
127 }
128
129 static {
130     novelasClasicasH.put("Orgullo y Prejuicio", "Disponible");

```

La última parte de la clase, se encarga de agregar todos los datos a las estructuras de datos, esto se realiza dentro de una clase de tipo static. Allí se llama a los HashMap y se les agrega el nombre del libro con su estado.

```

130     novelasClasicasH.put("Romeo y Julieta", "Disponible");
131     novelasClasicasH.put("Los Miserables", "Disponible");
132     novelasClasicasH.put("Drácula", "Disponible");
133     novelasClasicasH.put("Cumbres Borrascosas", "Disponible");
134     terrorH = new HashMap();
135     terrorH.put("La casa infernal", "Disponible");
136     terrorH.put("Los Cuadernos Lovecraft", "Disponible");
137     terrorH.put("Wody", "Disponible");
138     terrorH.put("Temores Crecientes", "Disponible");
139     terrorH.put("Seria Crave", "Disponible");
140     ingenieriaH = new HashMap();
141     ingenieriaH.put("Teorema del Loro", "Disponible");
142     ingenieriaH.put("Planilandia", "Disponible");
143     ingenieriaH.put("La Mano que Piensa", "Disponible");
144     ingenieriaH.put("Continuous Delivery", "Disponible");
145     ingenieriaH.put("Soft Skills", "Disponible");
146 }
147 }
148

```

❖ PAQUETE PRÉSTAMO

→ Préstamo

En esta clase primeramente creamos los campos necesarios para recibir la información para el préstamo de un libro, también se crea unas variables string las cuales va a guardar temporalmente la información suministrada por el bibliotecario, se define el método constructor y los demas para crear y darle forma al JFrame.

```
1  @.../
5  |
6  package PaquetePrestamo;
7
8  import ...
23
1 usage
24  public class Prestamo extends JFrame {
25      no usages
26      JPanel panel;
27      no usages
28      public JTextField fechaTxt;
29      no usages
30      public JTextField nombrePrestamista;
31      no usages
32      public static String fechaIngresada;
33      no usages
34      public static String nombreCliente;
35
36      no usages
37      public Prestamo() { this.configuracion(); }
38
39      1 usage
40      public void configuracion() {
41          this.setTitle("Préstamos");
42          this.setSize(450, 170);
43          this.setLocationRelativeTo((Component)null);
44          this.Componentes();
45
46          this.setVisible(true);
47          this.setDefaultCloseOperation(3);
48      }
49
50      1 usage
51      public void Componentes() {
52          this.panel = new JPanel();
53          this.panel.setLayout((LayoutManager)null);
54          this.setBackground(Color.LightGray);
55          this.getContentPane().add(this.panel);
56          this.ColocarTexto();
57          this.CamposTexto();
58          this.ColocarBoton();
59      }
60
61      1 usage
62      public void ColocarTexto() {
63          JLabel texto1 = new JLabel();
64          texto1.setText("Fecha de préstamo:");
65          texto1.setBounds(15, 20, 120, 20);
66          this.panel.add(texto1);
67          JLabel texto2 = new JLabel();
68          texto2.setText("Nombre del cliente:");
69          texto2.setBounds(10, 60, 120, 20);
70          this.panel.add(texto2);
71      }
72
73  }
```

En la acción listener del boton tomamos la información suministrada por el bibliotecario en la venta, se comprueba que sea correcta y se envía al método actualizartabla el cual se encuentra en la clase BibliotecaGui, esto permite que luego de oprimir el boton esta información se imprima en la Jtable que está en dicha clase.

```

65 public void CamposTexto() {
66     this.fechaTxt = new JTextField();
67     this.fechaTxt.setBounds(135, 20, 100, 20);
68     this.panel.add(this.fechaTxt);
69     this.nombrePrestamista = new JTextField();
70     this.nombrePrestamista.setBounds(135, 60, 120, 20);
71     this.panel.add(this.nombrePrestamista);
72 }
73
74 1 usage
75 public void ColocarBoton() {
76     JButton prestarLibro = new JButton();
77     prestarLibro.setBounds(270, 90, 150, 20);
78     prestarLibro.setText("Hacer prestamo");
79     this.panel.add(prestarLibro);
80     ActionListener guardarDatos = actionPerformed(e) -> {
81         Prestamo.fechaIngresada = Prestamo.this.fechaTxt.getText();
82         Prestamo.nombreCliente = Prestamo.this.nombrePrestamista.getText();
83         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
84
85         try {
86             LocalDate fechaPrestamo = LocalDate.parse(Prestamo.fechaIngresada, formatter);
87             System.out.println("fecha prestamo: " + String.valueOf(fechaPrestamo));
88             System.out.println("Cliente: " + Prestamo.nombreCliente);
89             String nombre = Prestamo.nombreCliente;
90             String libro = BibliotecaGUI.nomLibro;
91             String fecha = Prestamo.fechaIngresada;

```

Otra acción que se realiza es que se elimina el libro escogido del comboBox permitiendo así que no se pueda volver a prestar ese libro.

```

92 BibliotecaGUI.actualizarTabla(nombre, libro, fecha);
93
94 if (BibliotecaGUI.genElegido.equals("Novelas Clásicas")) {
95     BibliotecaGUI.comboNovelas.removeItem(BibliotecaGUI.comboNovelas.getSelectedIndex());
96 }
97
98 if (BibliotecaGUI.genElegido.equals("Terror")) {
99     BibliotecaGUI.comboTerror.removeItem(BibliotecaGUI.comboTerror.getSelectedIndex());
100 }
101
102 if (BibliotecaGUI.genElegido.equals("Ingeniería")) {
103     BibliotecaGUI.comboIngeniería.removeItem(BibliotecaGUI.comboIngeniería.getSelectedIndex());
104 }
105
106 Prestamo.this.dispose();
107 } catch (DateTimeParseException var7) {
108     JOptionPane.showMessageDialog((Component)null, "Error de formato de fecha: " + var7.getMessage());
109 }
110
111 Prestamo.this.dispose();
112 };
113 prestarLibro.addActionListener(guardarDatos);
114 }
115 }
116

```

❖ PAQUETE VENTANA PRIN

→ BibliotecaGUI

Esta clase es la principal donde se muestra la interfaz de la biblioteca.

```
1 //...
5
6 package PaqueteVentanaPrin;
7
8 import ...
9
39
40 public class BibliotecaGUI extends JFrame {
41     no usages
42     public static String nomLibro;
43     no usages
44     public static String genElegido;
45     no usages
46     private JPanel panel;
47     no usages
48     public static JComboBox comboNovelas;
49     no usages
50     public static JComboBox comboTerror;
51     no usages
52     public static JComboBox comboIngenieria;
53     no usages
54     private JRadioButton radioNovela;
55     no usages
56     private JRadioButton radioTerror;
57     no usages
58     private JRadioButton radioIngenieria;
59     no usages
60 }
```

Se define el método constructor de esta clase y los demás métodos que nos permiten darle estructura a la ventana.

```
60 static String[] columnNames = new String[]{"Nombre", "Libro", "Fecha"};
61 no usages
62 static DefaultTableModel model;
63 no usages
64 static JTable table;
65 no usages
66 private static Prestamo prestamo;
67 no usages
68 private static HashMap<String, String> librosGenero;
69 no usages
70 private static HashMap<String, Integer> clientesDeuda;
71 no usages
72
73 no usages
74 public BibliotecaGUI() {
75     this.setTitle("Biblioteca");
76     this.setSize(700, 600);
77     this.setLocationRelativeTo((Component)null);
78     this.setVisible(true);
79     this.setDefaultCloseOperation(3);
80     this.ComponentesVentana();
81 }
82
83 1 usage
84 public void ComponentesVentana() {
85     this.ColocarPanel();
86     this.ColocarLabel();
87 }
```

Luego en el método CrearItem se crean los menús con sus nombres y parámetros.

```

89         this.ColocarPanel();
90         this.ColocarLabel();
91         this.ColocarCombo();
92         this.ColocarItem();
93         this.ColocarButton();
94         this.ColocarRadioButton();
95         this.llenartabla();
96     }
97
98     2 usages
99     public void ColocarPanel() {
100         this.panel = new JPanel();
101         this.panel.setBackground(Color.LIGHT_GRAY);
102         this.panel.setLayout((LayoutManager)null);
103         this.getContentPane().add(this.panel);
104     }
105
106     1 usage
107     public void ColocarItem() {
108         JMenuBar menuBar = new JMenuBar();
109         this.setJMenuBar(menuBar);
110         JMenu estadoDeLosLibros = new JMenu("Estado de los libros");
111         menuBar.add(estadoDeLosLibros);
112         JMenuItem estadoLibros = new JMenuItem();
113         estadoDeLosLibros.setText("Consultar");
114         estadoDeLosLibros.add(estadoLibros);
115         JMenuItem clieMora = new JMenuItem("Clientes en mora");

```

En este método también se define las acciones listener de estos menús del primero que se define el menuitemregistro este es el que comprueba que si alguien quiere crear una nueva cuenta debe ser solo con la cuenta del jefe y si es asi inicia la clase Registro.

```

94     JMenuItem menuitemClieMo = new JMenuItem("Consultar Clientes");
95     clieMora.add(menuitemClieMo);
96     JMenuItem menuitemBorrarCli = new JMenuItem("Eliminar Clientes");
97     clieMora.add(menuitemBorrarCli);
98     menuBar.add(clieMora);
99     JMenu menuRegistro = new JMenu("Registro");
100     JMenuItem menuitemregistro = new JMenuItem("Crear Cuenta");
101     menuRegistro.add(menuitemregistro);
102     menuBar.add(menuRegistro);
103     JMenu menuSalir = new JMenu("Salir");
104     JMenuItem menuitemnew = new JMenuItem("Inicar Sesión Con Otra Cuenta");
105     menuSalir.add(menuitemnew);
106     JMenuItem menuitemsalir = new JMenuItem("Salir");
107     menuSalir.add(menuitemsalir);
108     menuBar.add(menuSalir);
109     menuitemregistro.addActionListener(actionPerformed(e) -> {
110         JTextField campoTexto1 = new JTextField();
111         JTextField campoTexto2 = new JTextField();
112         Object[] message = new Object[]{"Login:", campoTexto1, "Password:", campoTexto2};
113         int option = JOptionPane.showOptionDialog((Component)null, message, "Verificion de jefe", 2, -1, (Icon)null, (Object[])null, (Object)null);
114         if (option == 0) {
115             String texto1 = campoTexto1.getText();
116             String texto2 = campoTexto2.getText();
117             if (texto1.equals("a") && texto2.equals("a")) {
118                 new Registro();
119                 BibliotecaGUI.this.setVisible(false);
120             } else {

```


La accion listener del menuItemClicMo permite abrir un JOptionPane donde muestra los clientes que están en mora y también el monto a pagar.

```
122         JOptionPane.showMessageDialog((Component)null, "ESTE USUARIO NO ES UN JEFE");
123     }
124 }
125
126 });
127
128 menuItemNew.addActionListener(actionPerformed(e) → {
129     new IniciarSesion();
130     BibliotecaGUI.this.setVisible(false);
131 });
132
133 menuItemSalir.addActionListener(actionPerformed(e) → { BibliotecaGUI.this.dispose(); });
134
135 menuItemClicMo.addActionListener(new ActionListener(this) {
136     no usages
137     public void actionPerformed(ActionEvent e) {
138         StringBuilder clientesMora = new StringBuilder();
139         Iterator var3 = BibliotecaGUI.clientesDeuda.entrySet().iterator();
140
141         while(var3.hasNext()) {
142             Map.Entry<String, Integer> entry = (Map.Entry)var3.next();
143             String nombre = (String)entry.getKey();
144             int deuda = (Integer)entry.getValue();
145             clientesMora.append(nombre).append(": $").append(deuda).append("\n");
146         }
147
148         JOptionPane.showMessageDialog((Component)null, "Clientes en mora:\n" + clientesMora.toString());
149     }
150 });
151
152 menuItemBorrarCli.addActionListener(new ActionListener(this) {
```

En los siguientes métodos se crean y configuran los JLabel, JButton.

```
183
184 2 usages
185 public void ColocarLabel() {
186     JLabel texto1 = new JLabel();
187     texto1.setText("Novelas Clasicas:");
188     texto1.setBounds(10, 160, 110, 20);
189     this.panel.add(texto1);
190     JLabel texto2 = new JLabel();
191     texto2.setText("Terror:");
192     texto2.setBounds(70, 200, 90, 20);
193     this.panel.add(texto2);
194     JLabel texto3 = new JLabel();
195     texto3.setText("Ingenieria:");
196     texto3.setBounds(50, 240, 90, 20);
197     this.panel.add(texto3);
198 }
199
200 1 usage
201 public void ColocarButton() {
202     JButton prestamoLibro = new JButton();
203     prestamoLibro.setBackground(Color.green);
204     prestamoLibro.setText("Prestar libro");
205     prestamoLibro.setBounds(330, 195, 140, 25);
206     this.panel.add(prestamoLibro);
207     ActionListener guardaPrestamo = new ActionListener(this) {
208         no usages
209         public void actionPerformed(ActionEvent e) {
```

En esta parte del código se agrega la acción al botón la cual se encarga de llamar a la acción que permite cambiar el valor de los libros a “no Disponibles”

```

207:         if (BibliotecaGUI.genElegido.equals("Novelas Clasicas")) {
208:             Libro.eliminarNovelas(BibliotecaGUI.nomLibro);
209:         }
210:
211:         if (BibliotecaGUI.genElegido.equals("Terror")) {
212:             Libro.eliminarTerror(BibliotecaGUI.nomLibro);
213:         }
214:
215:         if (BibliotecaGUI.genElegido.equals("Ingenieria")) {
216:             Libro.eliminarIngenieria(BibliotecaGUI.nomLibro);
217:         }
218:
219:         BibliotecaGUI.prestamo = new Prestamo();
220:         BibliotecaGUI.librosGenero.put(BibliotecaGUI.nomLibro, BibliotecaGUI.genElegido);
221:         BibliotecaGUI.prestamo.setDefaultCloseOperation(2);
222:     }
223: };
224: prestamoLibro.addActionListener(guardaPrestamo);
225: JButton devolver = new JButton();
226: devolver.setBackground(Color.green);
227: devolver.setText("Devolver libro");
228: devolver.setBounds(500, 195, 140, 25);
229: this.panel.add(devolver);
230: ActionListener devolucion = new ActionListener(this) {
231:     no usages
232:     public void actionPerformed(ActionEvent e) {
233:         int selectedRow = BibliotecaGUI.table.getSelectedRow();

```

En esta acción listener de devolución se comprueba la fila que esta seleccionada en la tabla, es decir el libro e informacion del préstamo, se comprueba la fecha que esta en la tabla con la actual, es decir la fecha en que se esta ejecutando el programa, de esta comparación se saca los días transcurridos y hace una condicion la cual es que si los días que estuvo prestado el libro fue mayor a 7, se cobra una multa y este cliente es adicionado a la lista de clientes morosos. Por último el libro que es devuelto se adiciona nuevamente al comboBox al que pertenece, dando lugar así a que se pueda prestar nuevamente.

```

233:         if (selectedRow != -1) {
234:             String nombre = BibliotecaGUI.table.getValueAt(selectedRow, 0).toString();
235:             String libro = BibliotecaGUI.table.getValueAt(selectedRow, 1).toString();
236:             String fechaPrestamoStr = BibliotecaGUI.table.getValueAt(selectedRow, 2).toString();
237:             DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
238:             LocalDate fechaPrestamo = LocalDate.parse(fechaPrestamoStr, formatter);
239:             LocalDate fechaActual = LocalDate.now();
240:             long diasTranscurridos = ChronoUnit.DAYS.between(fechaPrestamo, fechaActual);
241:             System.out.println("Dias transcurridos desde el préstamo: " + diasTranscurridos);
242:             int deuda = (int)((diasTranscurridos - 7L) * 1000L);
243:             if (diasTranscurridos > 7L) {
244:                 BibliotecaGUI.clientesDeuda.put(nombre, deuda);
245:                 JOptionPane.showMessageDialog((Component)null, "Este usuario tiene una multa de: $" + deuda);
246:             }
247:
248:             DefaultTableModel model = (DefaultTableModel)BibliotecaGUI.table.getModel();
249:             model.removeRow(selectedRow);
250:             String generoLibro = (String)BibliotecaGUI.librosGenero.get(libro);
251:             if (generoLibro.equals("Novelas Clasicas")) {
252:                 BibliotecaGUI.comboNovelas.addItem(libro);
253:                 Libro.agregarNovelas(libro);
254:             } else if (generoLibro.equals("Terror")) {
255:                 BibliotecaGUI.comboTerror.addItem(libro);
256:                 Libro.agregarTerror(libro);
257:             } else if (generoLibro.equals("Ingenieria")) {
258:                 BibliotecaGUI.comboIngenieria.addItem(libro);
259:                 Libro.agregarIngenieria(libro);

```

Luego de realizar el cobro por los 7 días de demora en la entrega del libro, se agregan los libros devueltos al HashMap, es decir su estado vuelve a estar

disponible de nuevo.

```
343         comboTerror.setBounds(120, 195, 175, 25);
344         Iterator<String> iterator2 = Libro.terrorH.keySet().iterator();
345
346         while(iterator2.hasNext()) {
347             String nomLibro = (String)iterator2.next();
348             if (((String)Libro.terrorH.get(nomLibro)).equals("Disponible")) {
349                 comboTerror.addItem(nomLibro);
350             }
351         }
352
353         comboTerror.setEnabled(false);
354         this.panel.add(comboTerror);
355         comboIngenieria = new JComboBox();
356         comboIngenieria.setBounds(120, 235, 175, 25);
357         Iterator<String> iterator3 = Libro.ingenieriaH.keySet().iterator();
358
359         while(iterator3.hasNext()) {
360             String nomLibro = (String)iterator3.next();
361             if (((String)Libro.ingenieriaH.get(nomLibro)).equals("Disponible")) {
362                 comboIngenieria.addItem(nomLibro);
363             }
364         }
365
366         comboIngenieria.setEnabled(false);
367         this.panel.add(comboIngenieria);
368     }
369 }
```

En el método actualizarTabla recibe tres parámetros el nombre del cliente, nombre del libro y la fecha del préstamos, al recibir estos datos los guarda en un array de de String y luego se los pasa a la tabla para que aparezcan impresos en ella.

```
370         1 usage
371         public void llenarTabla() {
372             JScrollPane scrollPane = new JScrollPane(table);
373             scrollPane.setBounds(10, 280, 660, 250);
374             this.panel.add(scrollPane);
375         }
376
377         no usages
378         public static void actualizarTabla(String nombre, String libro, String fecha) {
379             DefaultTableModel model = (DefaultTableModel)table.getModel();
380             String[] rowData = new String[]{nombre, libro, fecha};
381             model.addRow(rowData);
382         }
383
384         static {
385             model = new DefaultTableModel(columnNames, 0);
386             table = new JTable(model);
387             librosGenero = new HashMap();
388             clientesDeuda = new HashMap();
389         }
390     }
```

★ PROGRAMA EN EJECUCIÓN

INICIO

Esta es la ventana de iniciar sesión.



Inicio de sesion: Biblioteca

BIBLIOTECA U

Nombre:

JORGE

Contraseña:

..

Iniciar Sesión

Luego de iniciar sesion se muestra la siguiente ventana, la cual no permite realizar todas las acción nesarias de la biblioteca.

Biblioteca

Estado de los libros Clientes en mora Registro Salir

☒ Novelas Clásicas

☐ Terror

☐ Ingeniería

Novelas Clásicas: Cumbres Borrascosas ▼

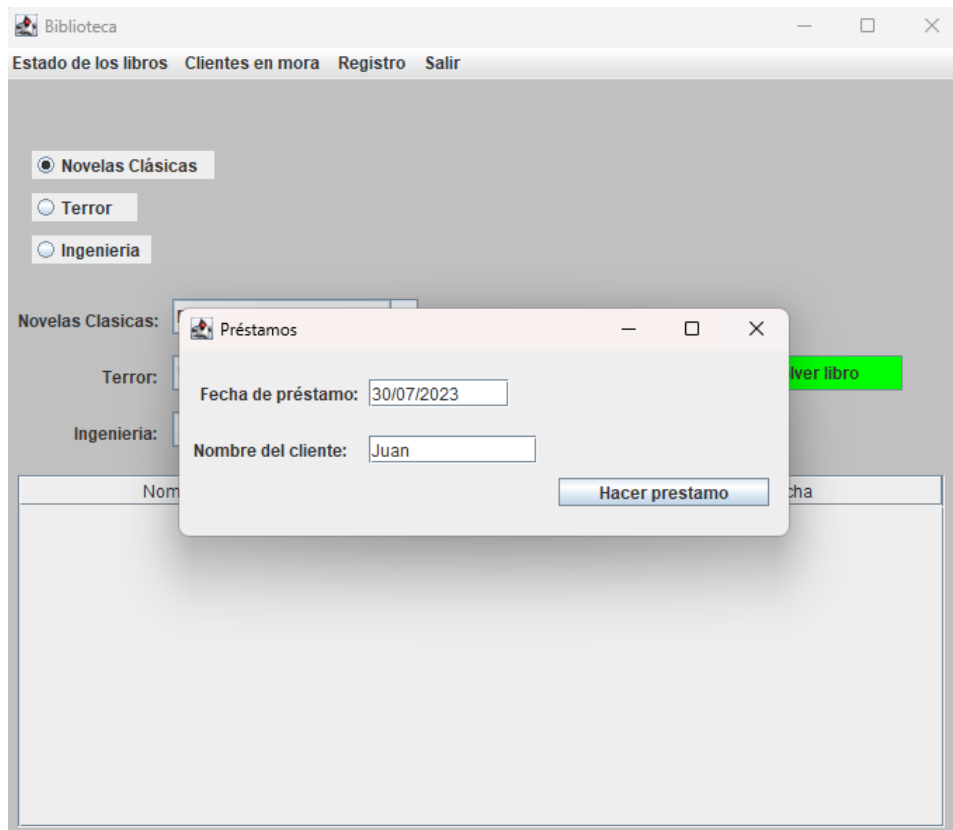
Terror: Wody ▼

Ingeniería: Soft Skills ▼

Nombre	Libro	Fecha
--------	-------	-------

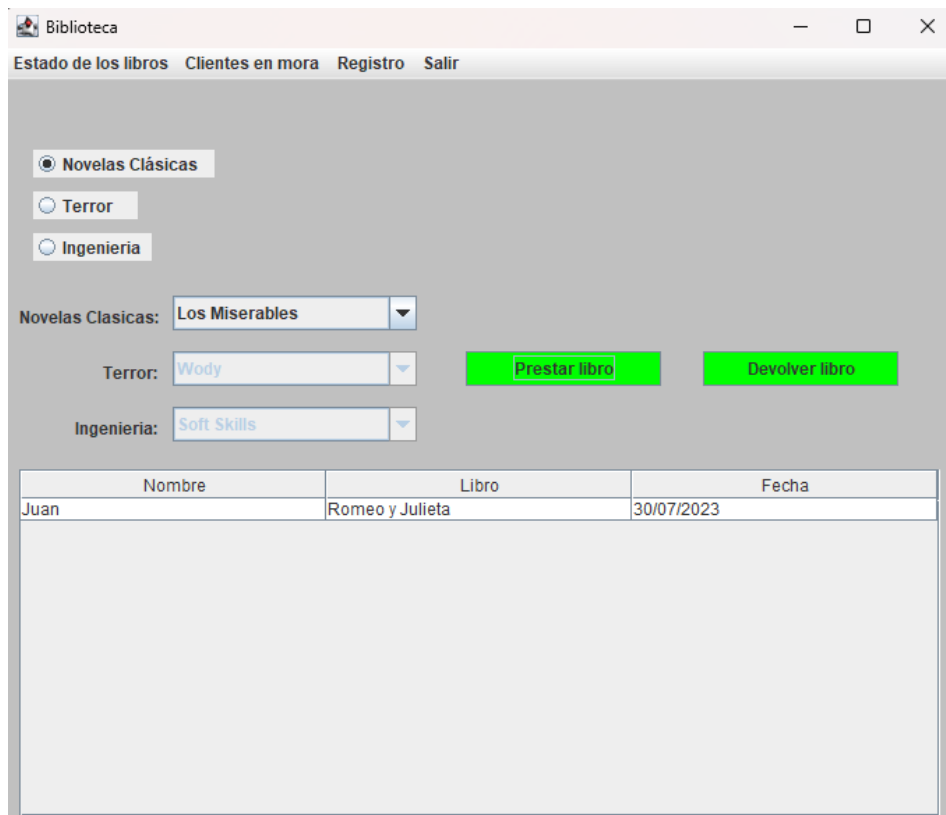
PRÉSTAMO LIBRO

Al oprimir el botón prestar libro no muestra una ventana la cual pide que se ingrese el nombre a la persona que se le va prestar el libro y también la fecha en que se hace ese préstamo.



USUARIO LIBRO PRESTADO

Luego de haber llenado lo anterior y oprimir el botón de hacer préstamo se logra que esta información se imprima en la tabla.



DIFERENTES TIPOS DE LIBROS

Para prestar un libro solo es necesario elegir uno de alguno de los comboBox.

Biblioteca

Estado de los libros Clientes en mora Registro Salir

☐ Novelas Clásicas

☐ Terror

☒ Ingeniería

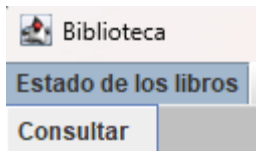
Novelas Clásicas:

Terror:

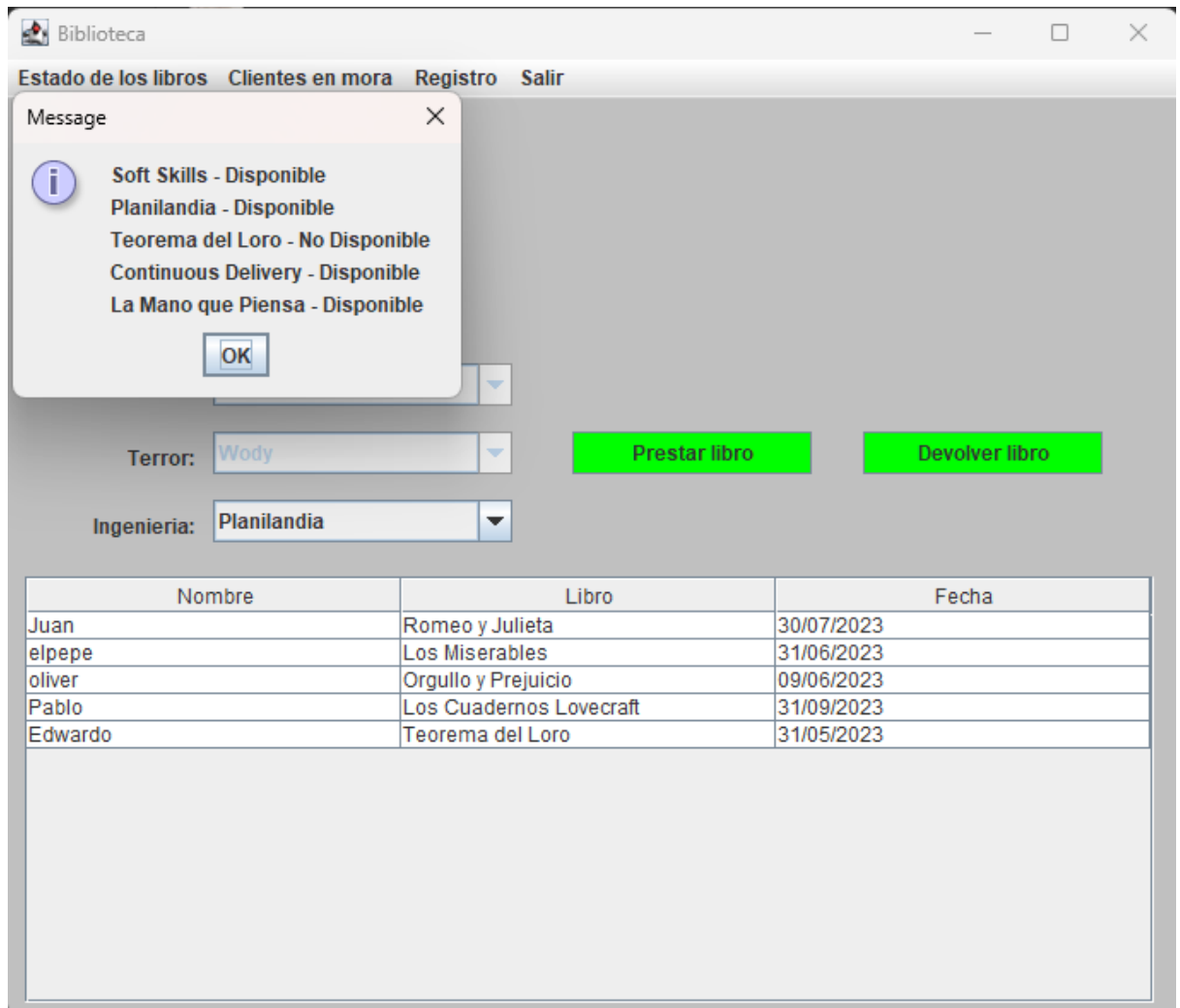
Ingeniería:

Nombre	Libro	Fecha
Juan	Romeo y Julieta	30/07/2023
elpepe	Los Miserables	31/06/2023
oliver	Orgullo y Prejuicio	09/06/2023
Pablo	Los Cuadernos Lovecraft	31/09/2023
Edwardo	Teorema del Loro	31/05/2023

CONSULTAR ESTADO DE LIBRO



Al oprimir el ítem consultar de Jmenubar muestra el estado de los libros, es decir si están prestados o disponibles, solo muestra los que pertenecen al radio botón que está seleccionado.



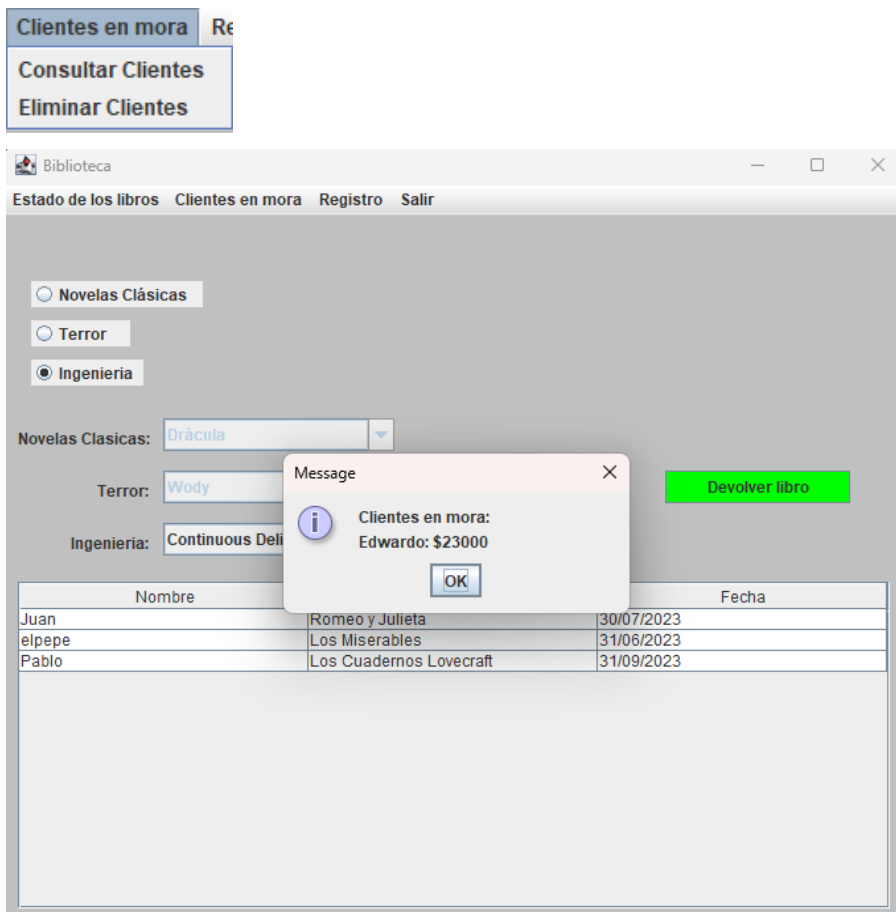
Devolucion Libro con Multa por Mora

Al elegir una fila de la tabla, es decir un libro prestado y oprimir el botón devolver libro se logra que ese préstamo se borre de la tabla, pero también hace que el libro prestado regrese al comboBox que le corresponde y así se pueda prestar de nuevo el libro, también consulta la fecha para saber si el cliente es mandado a la lista de los clientes morosos.

Nombre	Libro	Fecha
Juan	Romeo y Julieta	30/07/2023
elpepe	Los Miserables	31/06/2023
Pablo	Los Cuadernos Lovecraft	31/09/2023
Edwardo	Soft Skills	01/05/2023

Consultar cliente con multas por mora

Para consultar los clientes que están en mora, es decir aquellos que se excedieron mas de siete días con el prestamos del libro, solo basta con oprimir el item consultar clientes del menu clientes en mora, allí se mostrara los cliente y tambien el monto a cancelar.

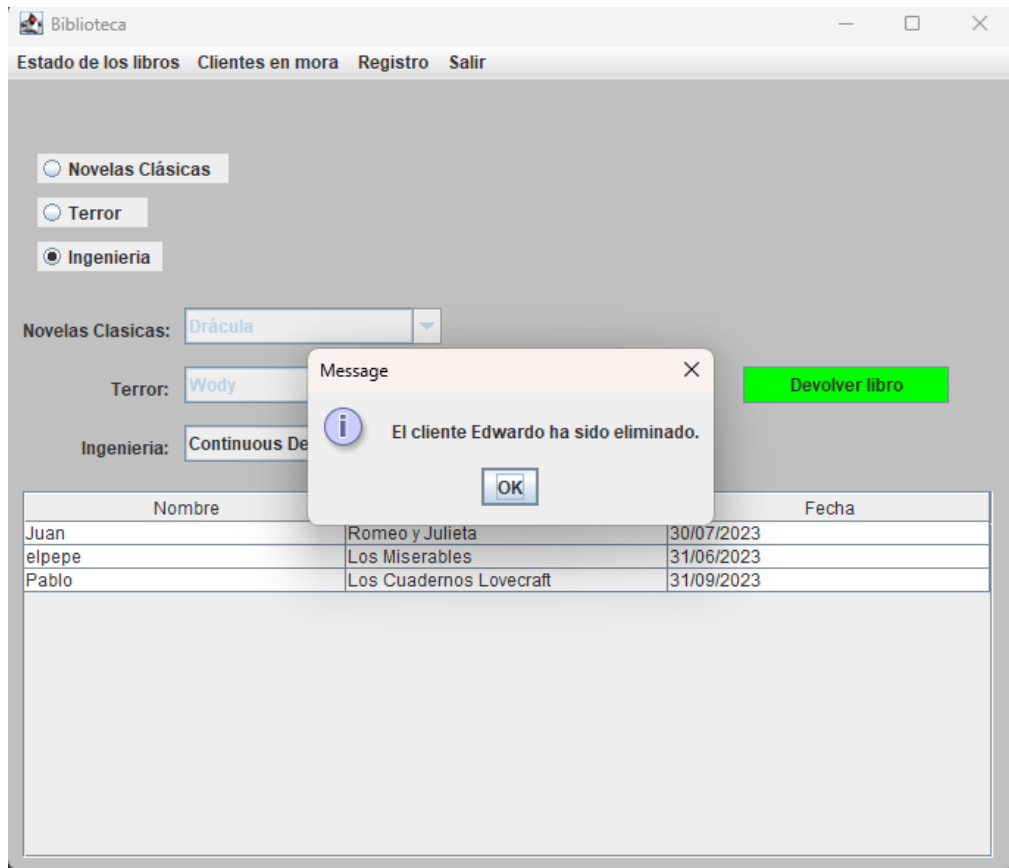


Eliminar Deuda del cliente por mora

En el mismo menu bar de clientes en mora esta el item de eliminar cliente, el cual nos permite borrar de esa lista de clientes morosos a quien ya pagaron su multa.

The screenshot shows a window titled "Biblioteca" with a menu bar containing "Estado de los libros", "Clientes en mora", "Registro", and "Salir". The main area has three radio buttons: "Novelas Clásicas", "Terror", and "Ingeniería" (which is selected). Below these are three text boxes: "Novelas Clásicas:" with "Dracula", "Terror:" with "Wody", and "Ingeniería:" with "Continuous D". A green button labeled "Devolver libro" is on the right. A table at the bottom has columns "Nombre" and "Fecha" with three rows of data. A modal dialog box titled "Input" is open, asking to enter the name of the client to be deleted, with "Edwardo" entered in the text field. The dialog has "OK" and "Cancel" buttons.

Nombre	Fecha	
Juan	Romeo y Julieta	30/07/2023
elpepe	Los Miserables	31/06/2023
Pablo	Los Cuadernos Lovecraft	31/09/2023



Crear cuenta de Bibliotecario nueva

Para crear una cuenta nueva para un bibliotecario se oprime en el menu crear cuenta.

Biblioteca

Estado de los libros Cientes en mora Registro Salir

Crear Cuenta

☐ Novelas Clásicas

☐ Terror

☒ Ingenieria

Novelas Clasicas: Dracula

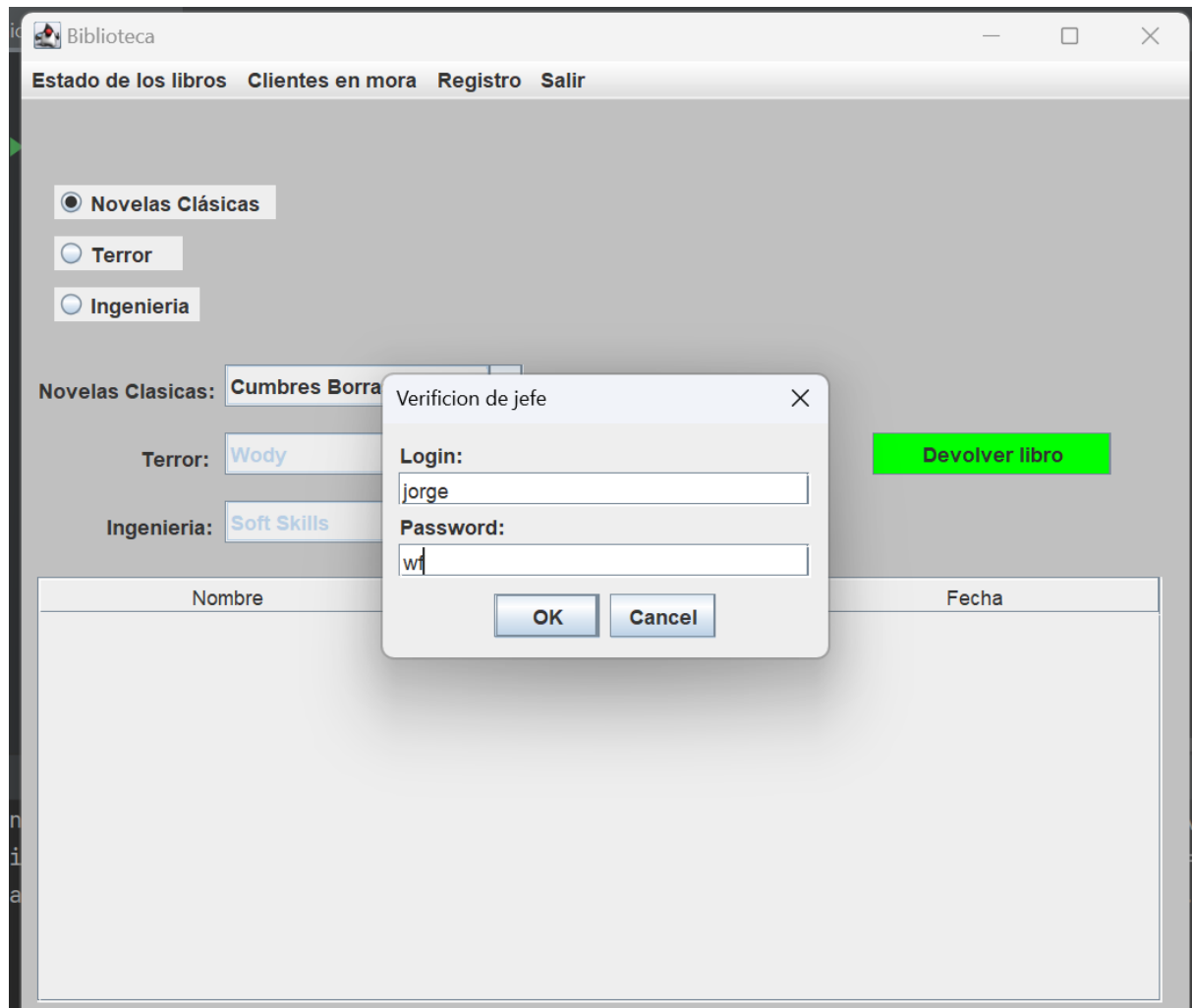
Terror: Wody

Ingenieria: Planilandia

Prestar libro Devolver libro


Nombre	Libro	Fecha
Juan	Romeo y Julieta	30/07/2023
elpepe	Los Miserables	31/06/2023
oliver	Orgullo y Prejuicio	09/06/2023
Pablo	Los Cuadernos Lovecraft	31/09/2023
Edwardo	Teorema del Loro	31/05/2023

Pero antes de pasar a la ventana para crear una nueva cuenta se debe confirmar de que quien la va crear es el bibliotecario jefe, la forma de comprobarlo es ingresando la cuenta del jefe.



Crear cuenta nueva de bibliotecario

Luego de confirmar de que si era el jefe se abre esta ventana donde se ingresa el nombre y contraseña para la nueva cuenta.



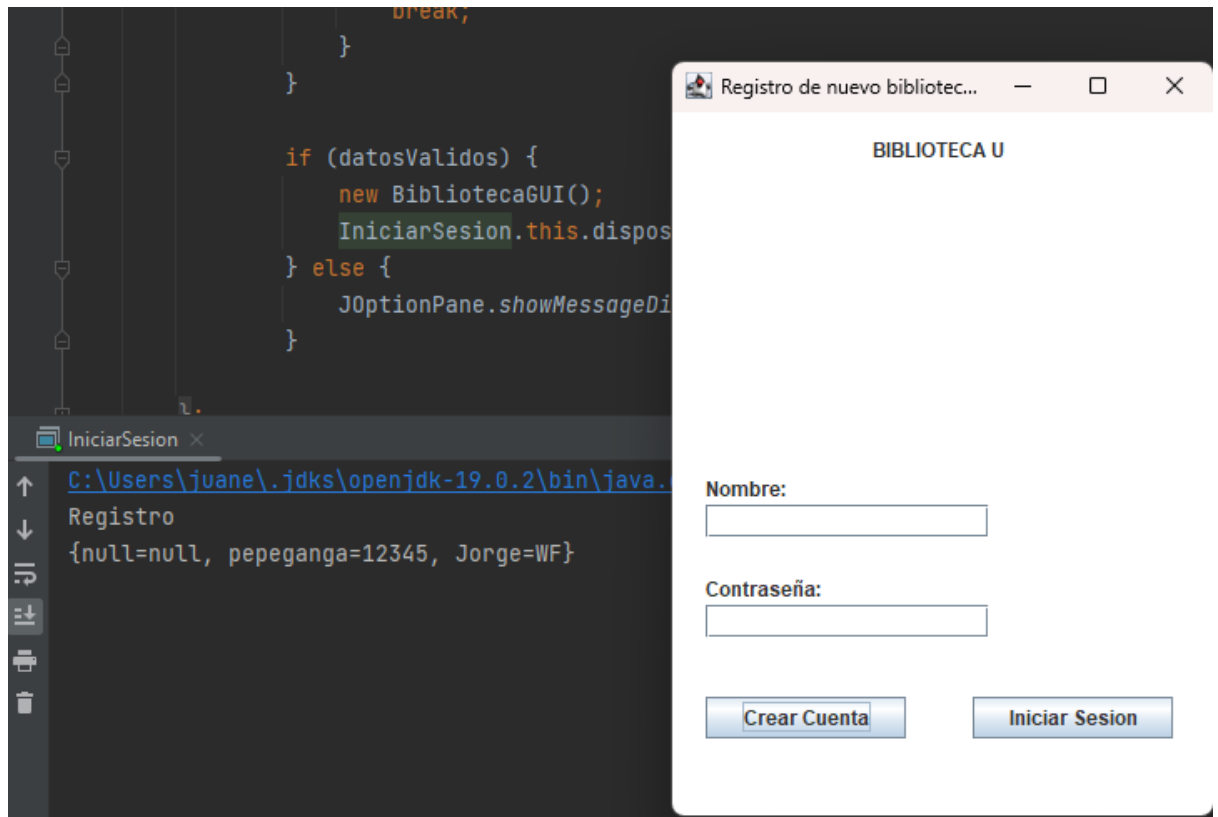
Registro de nuevo bibliotec...

BIBLIOTECA U

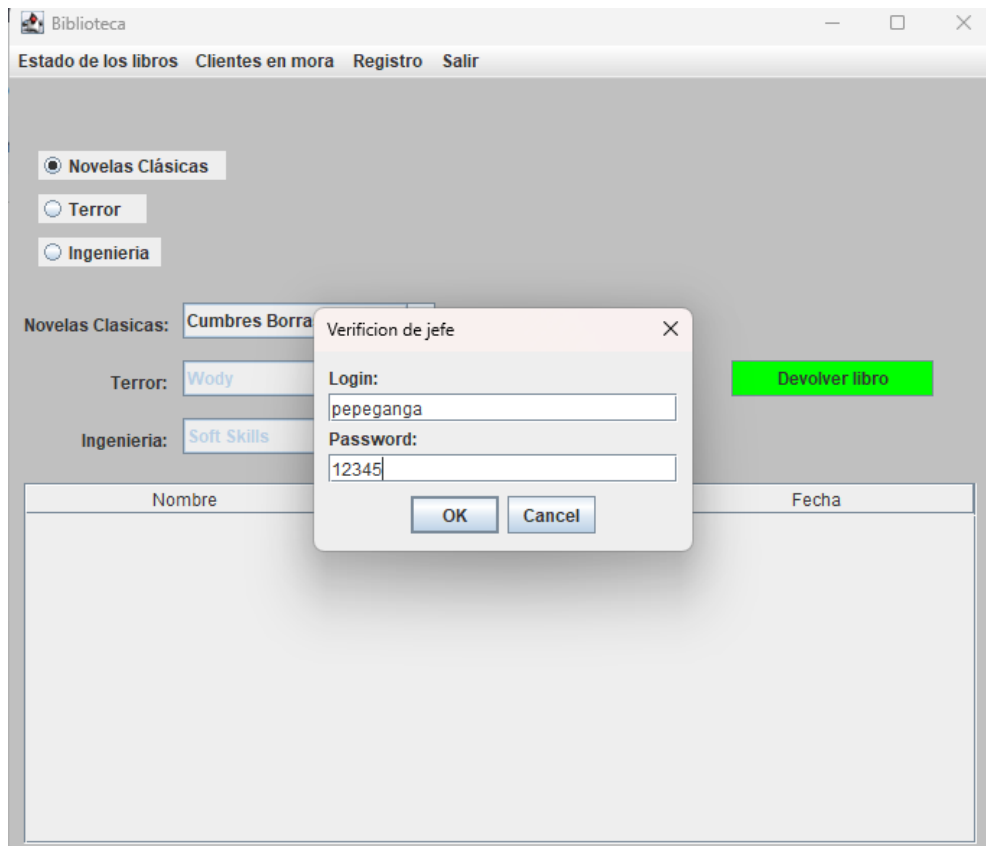
Nombre:

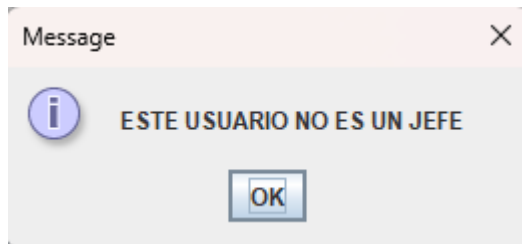
Contraseña:

Registro Bibliotecarios y BibliotecarioJefe



Cuando el bibliotecario no es BibliotecarioJefe





SALIR

Biblioteca

Estado de los libros Cientes en mora Registro **Salir**

Inicar Sesión Con Otra Cuenta
Salir

☒ Novelas Clásicas
☐ Terror
☐ Ingeniería

Novelas Clasicas: Cumbres Borrascosas ▼

Terror: Wody ▼

Ingenieria: Soft Skills ▼

Prestar libro **Devolver libro**

Nombre	Libro	Fecha
--------	-------	-------