MANUAL TÉCNICO

BIBLIOTECA

Introducción

El presente documento describe de forma gráfica las especificaciones de los códigos más importantes que se utilizaron en el Proyecto: “Biblioteca”.

Ciudad de Guatemala, Guatemala, 08/03/2022

Elaborado por JOSÉ ALEXANDER LÓPEZ LÓPEZ

Objetivos:

General:

- Aplicar los conceptos de programación orientada a objetos y librerías de awt que le servirán para crear su interfaz gráfica.

Específicos:

- Que el estudiante reconozca las ventajas de utilizar la programación orientada a objetos para el desarrollo de aplicaciones de escritorio.
- Que el estudiante aplique las sentencias de selección para las validaciones adecuadas en el sistema.
- Que el estudiante entienda el funcionamiento de cada uno de los componentes para la realización de la interfaz gráfica y cuáles son sus propiedades.

Lógica del Programa

1. Llamada de método principal

```
package Visual;
public class login {
    public static void main(String[] args){
     bienvenida ventanaobj1 =  new bienvenida();
    ventanaobj1.setVisible(true);
    }
}
```

A través del código anterior se comienza la ejecución del programa, es decir, la clase login es la clase principal que llamará los siguientes objetos.

2. Loggin del Sistema y su lógica de restricción

```
if(logged.getIdUsuario()!= -1)
{
    System.out.println("ID Usuario: "+logged.getIdUsuario());
    if((logged.getTipo() == 1 )) {
        Principal obj2 =  new Principal();
         obj2.setVisible(true);
    bienvenida.this.setVisible(false);
    }
    else {JOptionPane.showMessageDialog(null, "Usuario Normal - Acceso Restringido", "Su usuario no posee l
     }
}else{JOptionPane.showMessageDialog(null, "No Existe un usuario con las credenciales indicadas", "Error al I
}
} });
```

Luego de llamar a la clase bienvenida, es importante definir la lógica en el registro de usuarios del sistema, para ello se define un arreglo que guardará el Usuario ingresado, si el usuario ingresado coincide con alguno de los usuarios asignados a través de la carga de datos JSON, entonces la siguiente ventana se habilitará, Sin embargo, si los usuarios no coinciden con los ingresados, o el sistema identifica que el usuario no tiene permisos de administrador, entonces Imprimirá las restricciones y las razones por las cuales no se puede ingresar.

3. **Ingreso de los datos JSON y su almacenamiento en el programa.**

```
Object jsonarrayobyeto = objecte.get("Usuarios");
JSONArray arrayobyeto = (JSONArray) jsonarrayobyeto;
for(Object obyeto_inarray: arrayobyeto)
{
    if(contador==users.length)break;
    JSONObject VALEUR = (JSONObject) obyeto_inarray;
    ObjectU NEW = new ObjectU();

    NEW.setUsername((String) VALEUR.get("Usuario"));
    NEW.setPassword((String) VALEUR.get("Password"));
    NEW.setEscuela((String) VALEUR.get("Carrera"));
    NEW.setFacultad((String) VALEUR.get("Facultad"));
    Long value_tipo = (Long) VALEUR.get("Tipo");
    NEW.setTipo(value_tipo.intValue());
    Long value_id = (Long) VALEUR.get("ID");
    NEW.setIdUsuario(value_id.intValue());
    System.out.println("ID: "+ (Long) VALEUR.get("ID"));
    System.out.println("Usuario: "+ (String) VALEUR.get("Usuario"));
    System.out.println("Password: "+ (String) VALEUR.get("Password"));
    System.out.println("Tipo: "+ (Long) VALEUR.get("Tipo"));
    System.out.println("Facultad: "+ (String) VALEUR.get("Facultad"));
    System.out.println("Carrera: "+ (String) VALEUR.get("Carrera"));
    users[contador] = NEW;
    contador++;
}
```

El código anterior establece el orden que llevarán los datos que serán ingresados en forma de carga masiva, en el cual se especificará que los datos serán almacenados en el arreglo “Usuarios”, que contienen sub arreglos donde serán almacenados los datos que se requieren para ingresar al sistema.

4. **Llamada de elementos a un método.**

```
public libros(){


setLayout(null);
OBJECT1 =new Object[100];
contador12 = 0;
Unidades();
Textboxes();
Combo();
Btn();
tabla();
}
```

Luego de ingresar al sistema con los datos correctos, se establece un public class que obtendrá los métodos de los componentes que tendrá la ventana principal que en nuestro caso es la ventana “Libros”.

5. Establecimiento de datos a ingresar en un JTable

```
public void tabla(){

    modelol = new DefaultTableModel();
    modelol.addColumn("ID Libro");
    modelol.addColumn("Nombre Libro");
    modelol.addColumn("Autor");
    modelol.addColumn("Tipo");
    modelol.addColumn("Copias");
    modelol.addColumn("Disponibles");
    modelol.addColumn("Ocupados");


    JTable tabla = new JTable(modelol);
    deslizador2 = new JScrollPane(tabla);
    tabla.setBounds(250, 40, 800, 400);
    tabla.setFont(new java.awt.Font("Impact", 0, 14));
    deslizador2.setBounds(250, 40, 800, 400);
    add(deslizador2);

}
```

En el método graficado anteriormente se encuentra el modelo de la Tabla en el que se almacenarán los datos y además se encuentran los títulos de las columnas que serán utilizadas.

Además, se establece un JScrollPane y otros componentes gráficos para la interfaz de la tabla, como la fuente del texto y el tamaño.

6. **Establecimiento del método para almacenar datos manuales y sus restricciones.**

```
public void actionPerformed(ActionEvent en) {

    try{
        if(contador12<100){
           int id = Integer.parseInt(txt1.getText().toString());
           String libro= txt2.getText().toString();
           String autor= txt3.getText().toString();
           String tString= (String) lista.getSelectedItem();
           int tipo = Integer.parseInt(tString.toString());
           String tipos= Type(tipo);
           int copi = Integer.parseInt(txt4.getText().toString());

           Object[] newr ={id,libro,autor,tipos,copi,copi,0};
           modelo1.addRow(newr);
           contador12++;
        }else {
           JOptionPane.showMessageDialog(null, "Se ha llegado al limite de registros");  }
           }catch(Exception ex){
           JOptionPane.showMessageDialog(null, "Hacen falta datos");
    }
} });
```

La imagen anterior establece el método y el evento OnClick que cumplirá con las instrucciones establecidas. Se define el límite de datos ingresados, también se establecen las variables encargadas de almacenar los datos ingresados en las cajas de texto. Se establecen una serie de mensajes los cuáles comunican al usuario que hacen falta datos o que se ha llegado al límite permitido de datos.

7. **Lógica en el parseo de la fecha a variable String.**

```
public static Date conversordate(String fecha){

    SimpleDateFormat date = new SimpleDateFormat("dd/MM/yyyy");
      Date f =null;
  try{
    f=date.parse(fecha);

  }catch(Exception e){
    JOptionPane.showMessageDialog(null, "Información inválidad");
  }     return f;
}
```

La imagen muestra la acción lógica para almacenar los datos actuales de la fecha en un nuevo String, sin embargo, si el dato ingresado no cumple con el formato de fecha, se establece un mensaje que tiene

como finalidad informar al usuario que la información ingresada es incorrecta.