**CICLO FORMATIVO Y MÓDULO: DESARROLLO DE APLICACIONES MULTIPLATAFORMA**
**DESARROLLO DE INTERFACES 21/22**
**BENJAMIN MULERO PEDROSA**

# ***TAREA UNIDAD Nº 1: CONFECCIÓN DE INTERFACES DE USUARIO***

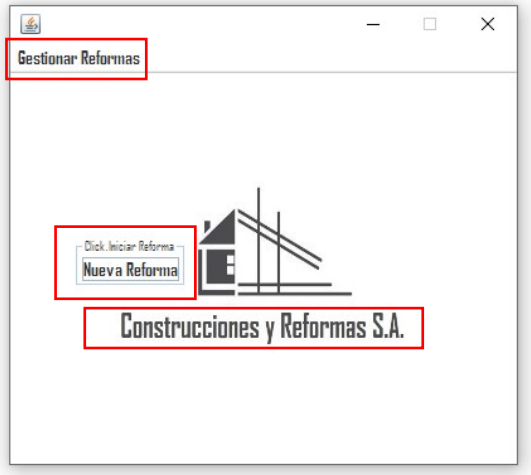
Creado el proyecto, creo otros dos directorios, una “Imagenes”, para introducir las imágenes necesarias, y otro “Formularios”, donde introduciré los formularios de la Tarea.

En el archivo creado al inicio del proyecto “Tarea01BMP.java”, establezco, que al ejecutar el proyecto se inicie por el “FormularioPrincipal”.

```
public class Tarea01BMP {


  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
    // TODO code application logic here
    FormularioPrincipal P = new FormularioPrincipal();
    P.setVisible(true);
  }
}
```

## FORMULARIO PRINCIPAL



He decidido utilizar los colores blanco y negro para el diseño de la interfaz y el formato de letra “Agency FB”, como formato distintivo de la empresa.

Elijo un “JFrame” al cual le inserto un panel, al cual le hago un “AbsoluteLayout”. Al panel creado le inserto un “icon” con el logotipo de la empresa y su nombre.

Elimino la opción “resizable”, para que no se pueda modificar el tamaño de la ventana de la interfaz.

Añado el menú, con sus dos opciones (Salir y Nueva Reforma) y el botón “Nueva Reforma” al lado del logotipo. Todas las opciones son activadas para realizar la acción señalada por la Tarea a través de “eventos”.

- Salir=Cerrar programa
- Nueva Reforma (Menú)=Inicia el Formulario Datos.
- Nueva Reforma (Botón central) = Inicia el Formulario Datos.

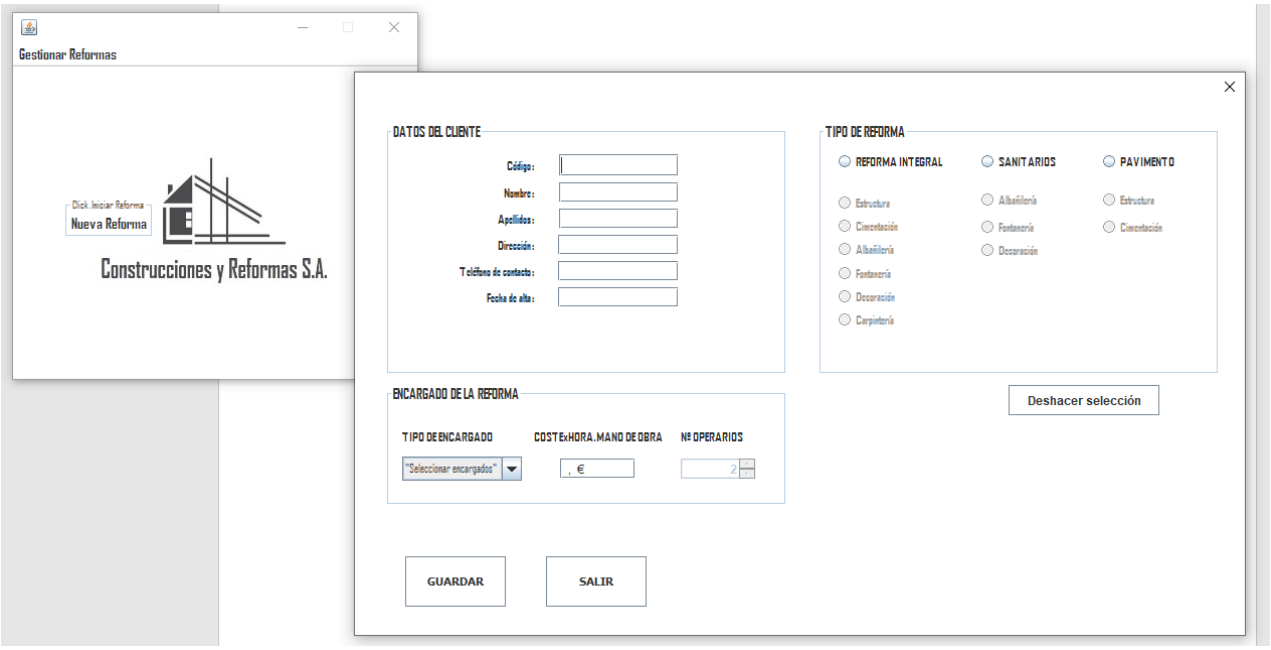
```
private void NuevaReformaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FormularioDatos formulariodatos = new FormularioDatos (this,true);
    formulariodatos.setVisible(true);
}

private void SalirActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
        dispose();
}

private void BotonNuevaReformaActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    FormularioDatos formulariodatos = new FormularioDatos (this,true);
    formulariodatos.setVisible(true);
}
```

Relleno la propiedad “toolTipText” de cada campo para aclarar al usuario los datos y requisitos a la hora de introducir los datos por teclado.

# FORMULARIO PARA LA RECOGIDA DE DATOS DE LA REFORMA




En el "Formulario de Datos", establezco un "JDialog", al cual le añado un panel que extiendo por todo el "JDialog" y le realizo un "AbsoluteLayout". A partir de este panel, voy estableciendo otros paneles que recaerán en el anterior.

He dividido "FormularioDatos" en 3 paneles:

- Datos del cliente:
  1. Para cumplir con los requisitos del ejercicio de esta sección, he establecido un "jlabel"(nombre del campo) con un "JTextField"(campo de texto) por cada dato exigido
  2. He rellenado la propiedad "toolTipText" de cada campo para aclarar al usuario los datos y requisitos a la hora de introducir los datos por teclado.
  3. Utilizando "eventos" he programado cada uno de los requisitos para cada campo.

```
private void telefonoKeyTyped(java.awt.event.KeyEvent evt) {

    // TODO add your handling code here:

    char c = evt.getKeyChar();

    if (c<'0' || c>'9')evt.consume();


    if (telefono.getText().length()>=9)

    {

       evt.consume();

    }


}
```

```
private void direccionKeyTyped(java.awt.event.KeyEvent evt) {
   // TODO add your handling code here:
   if (direccion.getText().length()>=30)
   {
      evt.consume();
   }
}

private void apellidosKeyTyped(java.awt.event.KeyEvent evt) {
   // TODO add your handling code here:
   if (apellidos.getText().length()>=30)
   {
      evt.consume();
   }
}

private void nombreKeyTyped(java.awt.event.KeyEvent evt) {
   // TODO add your handling code here:
   if (nombre.getText().length()>=20)
   {
      evt.consume();
   }
}

private void codigoKeyTyped(java.awt.event.KeyEvent evt) {
   // TODO add your handling code here:
   char c = evt.getKeyChar();
   if (c<'0' || c>'9')evt.consume();

   if (codigo.getText().length()>=5)
```

```
    {
        evt.consume();
    }
}

private void fechaKeyTyped(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
        if (fecha.getText().length()>=10)
    {
        evt.consume();
    }
}
```

4. Para la creación de la fecha y que esta tenga el formato deseado, he utilizado un "JFormattedTextField".

- Tipos de reforma
  1. He establecido el sistema de selección del "Tipo de reforma" a través de "radioButton", realizando la agrupación señalada por la tarea.
  2. Coloco un "buttonGroup" que agrupa los tres tipos de reformas.
  3. En este caso he decidido no rellenar la propiedad "toolTipText" ya que se puede leer claramente la selección que se está haciendo.
  4. Para el sistema de selección, he decido utilizar "setSelected" y "setEnabled" en "eventos".
     EJ.

```
private void
reformaIntegralActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    /*Si reformaIntegral es seleccionado, se ponen activos los de su
grupo y el resto siguen desactivados*/
    if (reformaIntegral.isSelected())
    {
        estructura.setEnabled(true);
        cimentacion.setEnabled(true);
        albañileria.setEnabled(true);
        fontaneria.setEnabled(true);
        decoracion.setEnabled(true);
        carpinteria.setEnabled(true);
        sanitarios.setEnabled(false);
        pavimento.setEnabled(false);
        albañileriaSani.setEnabled(false);
        fontaneriaSani.setEnabled(false);
        decoracionSani.setEnabled(false);
```

```
            estructuraPavi.setEnabled(false);
            cimentacionPavi.setEnabled(false);

        }
    }
```

5. Establezco un botón para deshacer los cambios y poder restear la sección de "Tipos de Reforma".

- Encargado de la reforma: Formado por tres "jlabel", y sus respectivos campos exigidos por la tarea: Tipo de encargado, coste mano de obra por hora, número de operarios.
    a. Tipo de encargado:
    - Afecta como se comportarán "Coste mano de obra" y "Número de operarios, según la selección de "Tipo de encargado".
    - Para este campo utilizo un "jcombobox" formado por 3 items (empleados propios, autónomos, subcontrata)
    - Cada item tiene una serie de restricciones de activación y de validación, que he establecido en un "evento"
        1. Empleados propios: Al seleccionarlo, el "Nº de operarios" estará desactivado, y cada vez que se seleccione se desactivará
        2. Autónomos: Al seleccionarlo, el "Nº de operarios" se activará, y cada vez que se seleccione se activará.
        3. Subcontrata: Al seleccionarlo, el "Nº de operarios" se activará, y cada vez que se seleccione se activará.

```
private void encargadoActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if (encargado.getSelectedItem().equals("Empleados propios")) {
            spinner.setEnabled(false);

        }

        if (encargado.getSelectedItem().equals("Autónomos")||encargado.getSelectedItem().equals("Subcontrata")) {
            spinner.setEnabled(true);
        }

        MiFuncion();
    }
```

    b. CosteXhora.Mano de obra.
    - Para este campo he utilizado un "Formatted Field" con mascara creada por mi, en la cual son 6 caracteres (4 dígitos, una coma y el símbolo del euros/##,##€ ), utilizando "formatterFactory".
    - Viene afectado por el "tipo de encargado"
        i. Empleados propios y Autónomos: El CosteXhora.Mano de obra deberá alcanzar como máximo 99,99 que lo he conseguido al establecer la máscara.

ii. Subcontrata: El CosteXhora.Mano de obra al seleccionar subcontrata ha de ser de un máximo de 80 euros, pero no he conseguido obtenerlo.

c. Nº de operarios
- Para este campo he utilizado un “spinner”.
- Viene afectado por el “tipo de encargado”
  I. Activando y desactivándolo según la selección de “tipo de encargado” y que solucioné de la forma señalada anteriormente.
  II. Estableciendo el spinner en un rango de como máximo 50 y como mínimo 2 y que he establecido a través de las propiedades en la interfaz gráfica de Netbeans.

## Guardar, salir y validaciones

**1. Guardar**

Para realizar la acción de guardar:

I. Establezco un botón (botón “Guardar”)
II. Creo un “private boolean” denominado validarFormulario y que me servirá posteriormente como comprobador de la validación

```
private boolean validarFormulario (){

String nom=nombre.getText();

String ape=apellidos.getText();

String dir=direccion.getText();

String tel=telefono.getText();

String fec=fecha.getText();


/*VALIDACIÓN DE DATOS DE LOS CLIENTES*/

try

{

  Integer.parseInt(codigo.getText());

}

catch (NumberFormatException n)

{

  JOptionPane.showMessageDialog(this, "El campo Código, no puede estar vacío", "ERROR", JOptionPane.ERROR_MESSAGE);

  return false;
```

```
    }


    if (nom==null||"".equals(nom))
    {
       JOptionPane.showMessageDialog(this, "El campo Nombre, no puede
estar vacío", "ERROR", JOptionPane.ERROR_MESSAGE);
       return false;
    }


    if (ape==null||"".equals(ape))
    {
       JOptionPane.showMessageDialog(this, "El campo Apellidos, no puede
estar vacío", "ERROR", JOptionPane.ERROR_MESSAGE);
       return false;
    }


    if (dir==null||"".equals(dir))
    {
       JOptionPane.showMessageDialog(this, "El campo Dirección, no puede
estar vacío", "ERROR", JOptionPane.ERROR_MESSAGE);
       return false;
    }


    if (tel==null||"".equals(tel))
    {
       JOptionPane.showMessageDialog(this, "El campo Teléfono de
contacto, no puede estar vacío", "ERROR", JOptionPane.ERROR_MESSAGE);
```

```
        return false;
    }
        if (tel.length()<9)
    {
        JOptionPane.showMessageDialog(this, "El campo Teléfono de
contacto, ha de tener 9 dígitos", "ERROR", JOptionPane.ERROR_MESSAGE);
        return false;
    }

        try{
            SimpleDateFormat formatoFecha = new
SimpleDateFormat("dd/MM/yyyy");
        formatoFecha.setLenient(false);
        formatoFecha.parse(fec);
                } catch (ParseException e)
{JOptionPane.showMessageDialog(this, "El campo Fecha, no puede estar
vacío y ha de tener el siguiente formato:Ej.12/10/2020", "ERROR",
JOptionPane.ERROR_MESSAGE);

        return false;

    }

    /*VALIDACIÓN TIPO DE REFORMA*/

    if (reformaIntegral.isSelected()==false && sanitarios.isSelected()==false
&& pavimento.isSelected()==false)
        {JOptionPane.showMessageDialog(this, "Debes seleccionar un tipo
de reforma", "ERROR", JOptionPane.ERROR_MESSAGE);
        return false;
        }
```

```
    if (estructura.isSelected()==false && cimentacion.isSelected()==false &&
albañileria.isSelected()==false&&

                fontaneria.isSelected()==false &&
decoracion.isSelected()==false && carpinteria.isSelected()==false&&

            albañileriaSani.isSelected()==false &&
fontaneriaSani.isSelected()==false && decoracionSani.isSelected()==false&&

                estructuraPavi.isSelected()==false &&
cimentacionPavi.isSelected()==false)

        {JOptionPane.showMessageDialog(this, "Debes seleccionar alguno
de los servicios de la reforma elegida", "ERROR",
JOptionPane.ERROR_MESSAGE);

        return false;

        }


/*VALIDACION COSTE X HORA. MANO DE OBRA*/

if (cos==null||" ,  €".equals(cos))

    {

      JOptionPane.showMessageDialog(this, "El campo CosteXHoraMano de
obra, debe contener una cuantía", "ERROR", JOptionPane.ERROR_MESSAGE);

        return false;

    }

return true;

}
```

III. Si la validación es correcta, aparecerá el mensaje registro guardado y todo pasará al estado original. Este pasó lo he realizado con un evento, añadiendo

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


    // TODO add your handling code here:

    /*Se comprueba que se cumplen las condiciones de la validación*/

    if (validarFormulario())

    {

    /*Todo vuelve a su forma original*/

      JOptionPane.showMessageDialog(this, "Registro guardado");
```

```
/*Datos del cliente*/
    codigo.setText("");
    nombre.setText("");
    apellidos.setText("");
    direccion.setText("");
    telefono.setText ("");
    fecha.setText ("");

/*Encargado de la reforma*/
    coste.setText("");
    encargado.setSelectedIndex(0);
    spinner.setValue(2);
    spinner.setEnabled(false);

/*Tipo de reforma*/
    buttonGroup1.clearSelection();
    reformaIntegral.setEnabled(true);
    sanitarios.setEnabled(true);
    pavimento.setEnabled(true);
    estructura.setEnabled(true);
    cimentacion.setEnabled(true);
    albañileria.setEnabled(true);
    fontaneria.setEnabled(true);
    decoracion.setEnabled(true);
    carpinteria.setEnabled(true);
    albañileriaSani.setEnabled(true);
    fontaneriaSani.setEnabled(true);
    decoracionSani.setEnabled(true);
    estructuraPavi.setEnabled(true);
    cimentacionPavi.setEnabled(true);
```

```
                reformaIntegral.setSelected(false);
                sanitarios.setSelected(false);
                pavimento.setSelected(false);
                estructura.setSelected(false);
                cimentacion.setSelected(false);
                albañileria.setSelected(false);
                fontaneria.setSelected(false);
                decoracion.setSelected(false);
                carpinteria.setSelected(false);
                albañileriaSani.setSelected(false);
                fontaneriaSani.setSelected(false);
                decoracionSani.setSelected(false);
                estructuraPavi.setSelected(false);
                cimentacionPavi.setSelected(false);


                estructura.setEnabled(false);
                cimentacion.setEnabled(false);
                albañileria.setEnabled(false);
                fontaneria.setEnabled(false);
                decoracion.setEnabled(false);
                carpinteria.setEnabled(false);
                albañileriaSani.setEnabled(false);
                fontaneriaSani.setEnabled(false);
                decoracionSani.setEnabled(false);
                estructuraPavi.setEnabled(false);
                cimentacionPavi.setEnabled(false);
        }
}
```

**2. Salir**

I. Al igual que en el botón “Guardar”, establecí un botón para “Salir”.

II. Establezco otro “private boolean”, que lo utilizaré como validador en un evento del botón Salir. Esta validación analiza si alguno de los “Datos de los Clientes” esta relleno, para advertir si quiere seguir o no con la cumplimentación de la nueva reforma.

```
private boolean validacionSalir (){
    String nom=nombre.getText();
    String ape=apellidos.getText();
    String dir=direccion.getText();
    String tel=telefono.getText();
    String fec=fecha.getText();

    if (!nom.isEmpty() || !nom.isEmpty() || !nom.isEmpty() || !ape.isEmpty() || !dir.isEmpty() || !tel.isEmpty() || !fec.isEmpty())
    {
        return true;
    }
  return false;
}
```

III. Establezco en el evento el validador de salida y el mensaje de advertencia y un evento para salir.

```
private void salirActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    if (validacionSalir())
    {int n=JOptionPane.showConfirmDialog(null, "Has introducido algún dato de la sección 'Datos del Cliente', ¿QUIERES SALIR?","Vas a salir..",JOptionPane.ERROR_MESSAGE);
        if (n==0) {dispose();}
        if (n==1) {}
        }

  }
```

```
private void salirMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    dispose();
}
```

# FIN