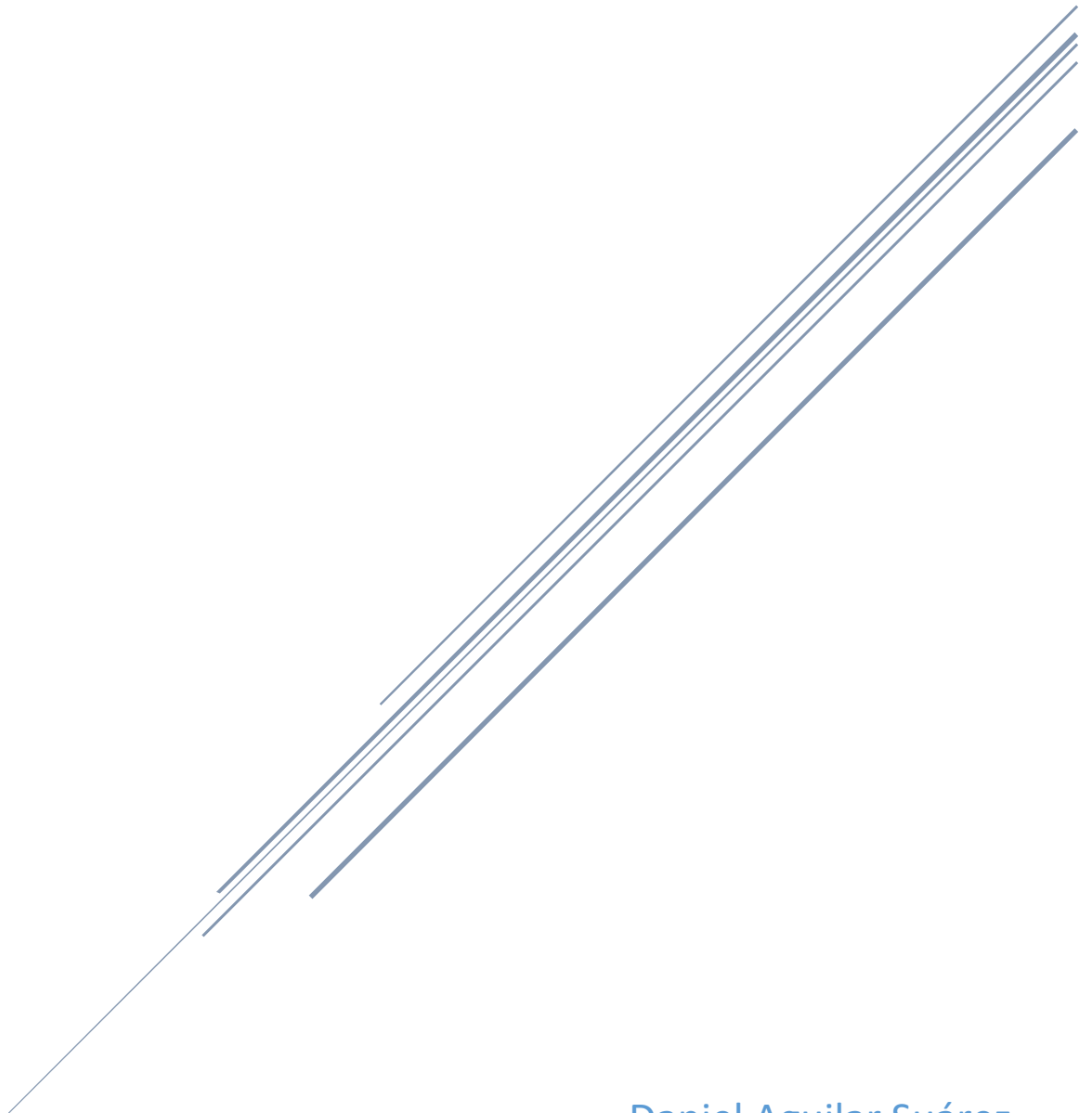


# SERIALIZACIÓN Y CREACIÓN DE INTERFACES DE USUARIO

Programación II



Daniel Aguilar Suárez  
DAW

## Tabla de contenido

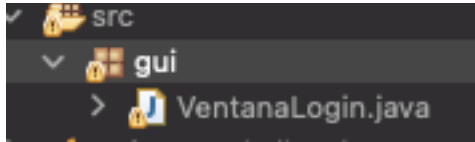
<b>1. Crear Ventana de Login .....</b>	<b>3</b>
a. Usar un fichero de nombres para comprobar si los datos son correctos .....	6
TEST CASE:.....	11
<b>2. Segunda Ventana .....</b>	<b>13</b>
<b>3. Crear una nueva clase Empleado .....</b>	<b>18</b>
<b>4. Crear nuevo empleado.....</b>	<b>19</b>
a. Comprobar si existen empleados previamente serializados en un archivo y crear el Arraylist con esos empleados.....	19
b. Crear un nuevo empleado a partir de los datos introducidos por el usuario .....	20
c. Añadir al arrayList de empleados el nuevo empleado .....	21
d. Serializar la lista de empleados y guardarlos en un archivo de nombre datos_empleados.....	21
e. Cuadro de dialogo si todo es correcto o si no se ha rellenado nada .....	22
TEST CASE:.....	22
<b>5. Boton Muestra empleado .....</b>	<b>24</b>
a. Obtener el listado de empleados serializados .....	24
b. Cuadro de dialogo con todos los empleados .....	24
TEST CASE.....	25
<b>6. TODOS LOS CÓDIGOS COMPLETOS.....</b>	<b>25</b>
a. APLICACIÓN/Main.java.....	25
b. CLASES/Empleado.java .....	26
c. CLASES/Usuarios.java .....	27
d. FUNCIONES/Funcs.java .....	28
e. GUI/VentanaLogin.java.....	29
f. GUI/VentanaAddTrab.java.....	31
<b>7. Fe de erratas.....</b>	<b>35</b>

**Información del alumno**

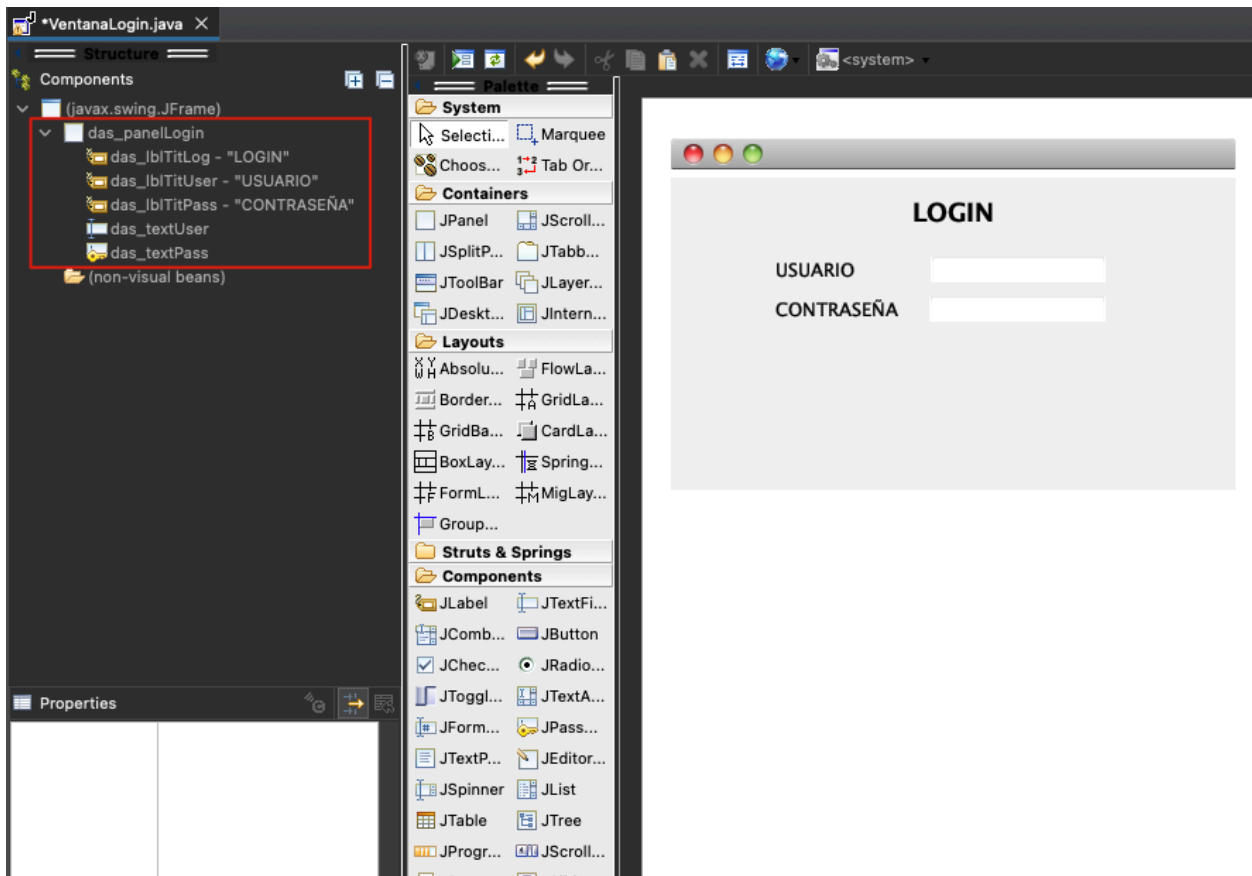
Nombre	Daniel
Apellidos	Aguilar Suárez
Módulo/Crédito	M03
UF (solo ciclos LOE)	UF5
Título de la actividad	Serialización y creación de interfaces de usuario

## 1. Crear Ventana de Login

Lo primero de todo, creo un Proyecto nuevo y una carpeta “gui” con una clase Java JFrame llamada VentanaLogin.java:

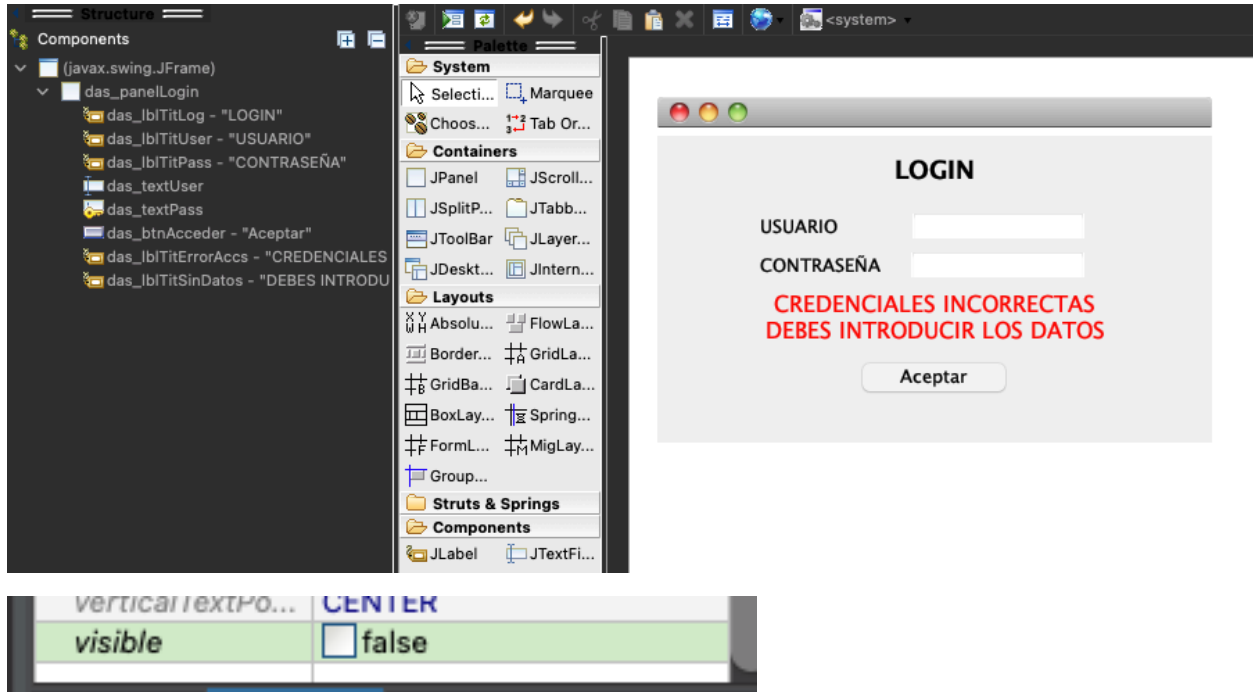


Para empezar a crear la estructura de la ventana, añado 3 JLabel, un JTextField, y un JPasswordField, además les he cambiado el nombre de las variables:

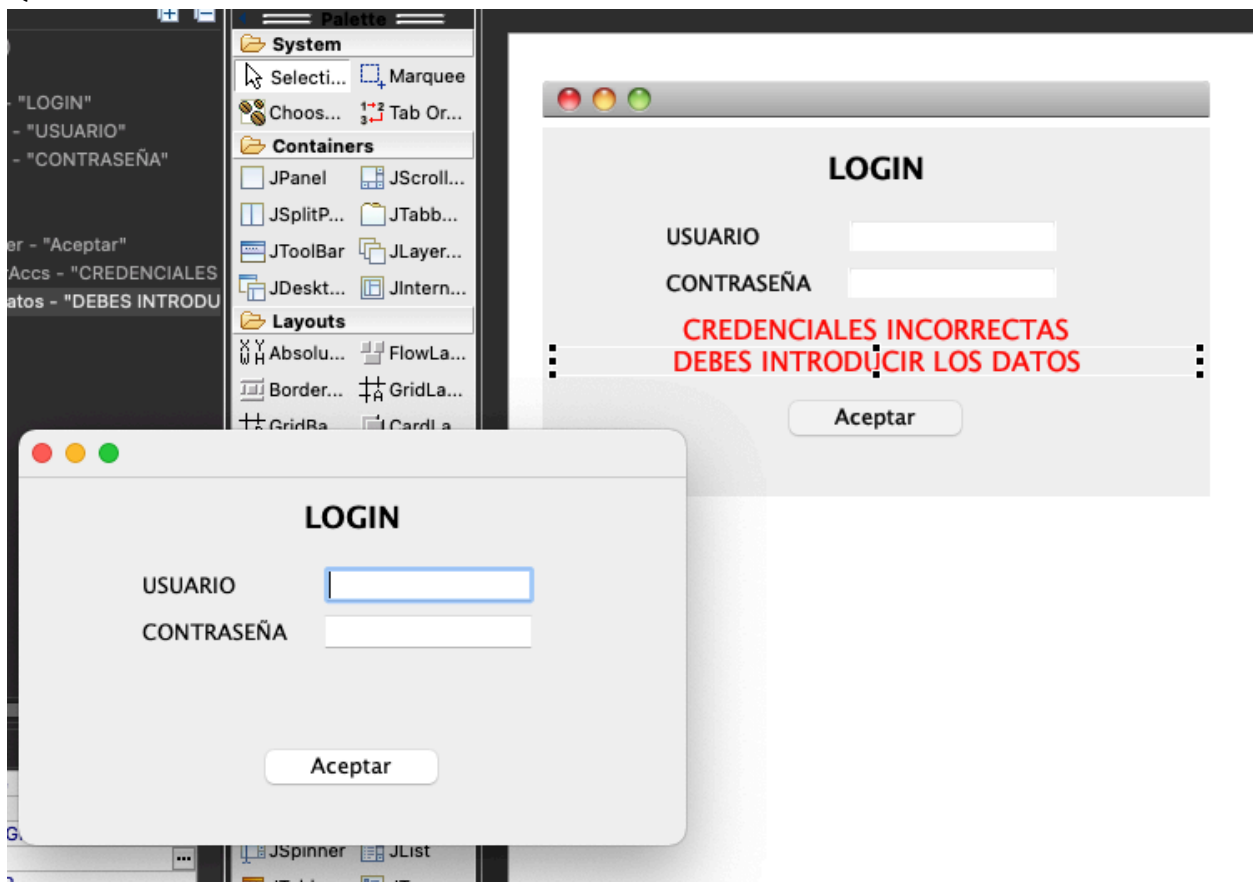


Ahora añado un botón y dos títulos que configuraré para que estén ocultos, los configuro con sus correspondientes variables:

Daniel Aguilar Suarez  
SERIALIZACIÓN Y CREACIÓN DE INTERFACES DE USUARIO



Quedaría así:



Ahora he creado una clase Main para tenerlo todo mas ordenado, dentro he exportado el método main que me crea eclipse de la clase JFrame:

```
package aplicacion;

import java.awt.EventQueue;

import gui.VentanaLogin;

public class Main {

    public static void main(String[] args) {
        /**
         * Launch the application.
         */
        // He utilizado la sintaxis por defecto que me ha generado eclipse y la he
        exportado a una clase main aparte
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    VentanaLogin frame = new VentanaLogin();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

Además, he añadido dos líneas más al constructor de la ventana, dentro de VentanaLogin he añadido las siguientes líneas para tener un título en la ventana, y centrarla en la pantalla:

```
setTitle("VENTANA DE LOGIN"); // TTITULO DE LA VENTANA
setLocationRelativeTo(null); // CENTRAR VENTANA
```

a. Usar un fichero de nombres para comprobar si los datos son correctos

Lo realizare creando una clase Serializable. La clase la llamo Usuarios:

```
package clases;

import java.io.Serializable;

public class Usuarios implements Serializable{

    private String nombre;
    private String password;

    public Usuarios(String nombre, String password) {
        super();
        this.nombre = nombre;
        this.password = password;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPassword() {
        return password;
    }

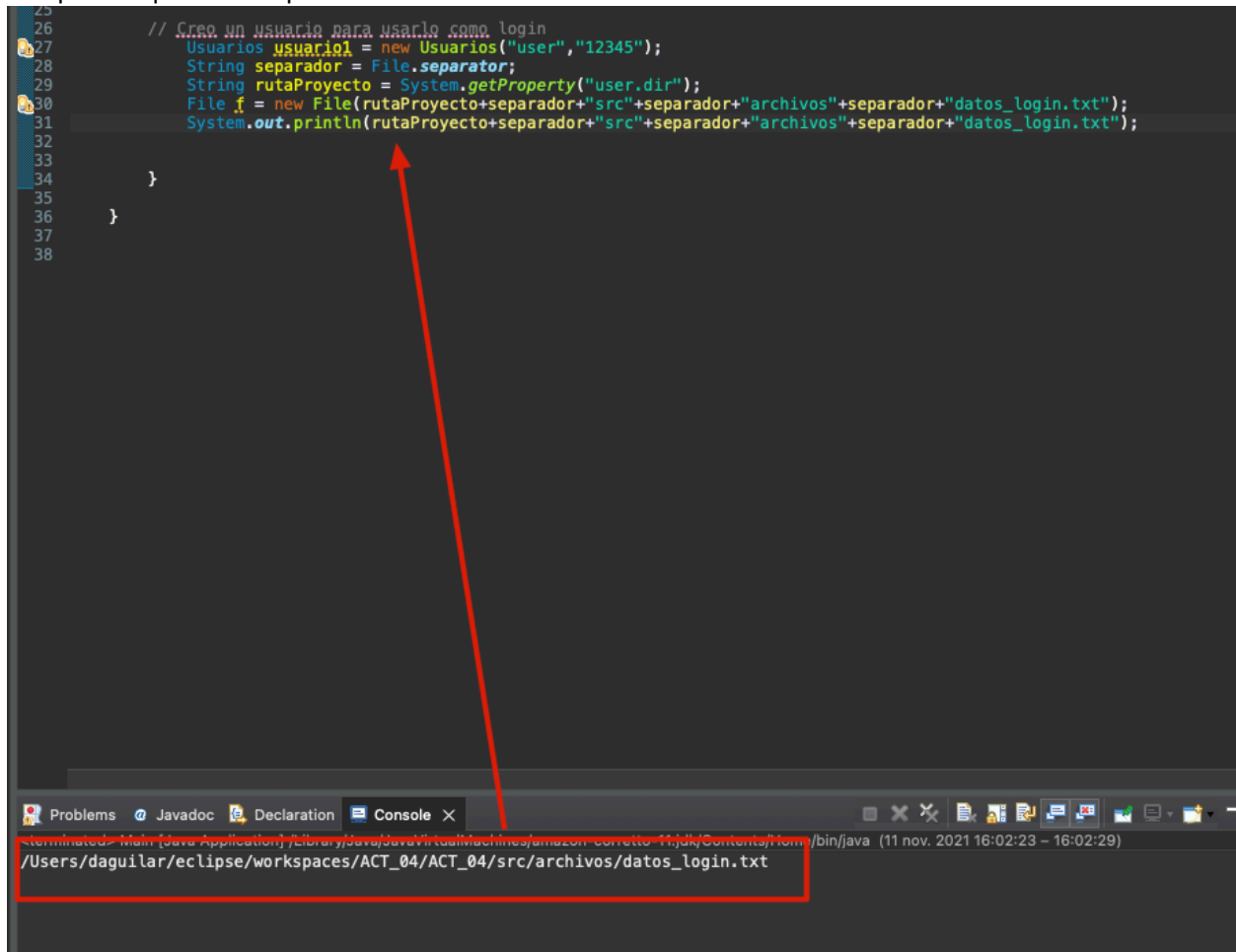
    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String toString() {
        return "Usuarios [nombre=" + nombre + ", password=" + password + "];"
    }
}
```

Ahora creo un usuario nuevo:

```
Usuarios usuario1 = new Usuarios("user","12345");
```

Para crear la ruta del File, tengo que hacer pruebas, por lo que tendré que hacer algunos Test Case para comprobar que la ruta que me muestra sería la ruta correcta:



```
25
26 // Crea un usuario para usarlo como login
27 Usuarios usuario1 = new Usuarios("user", "12345");
28 String separador = File.separator;
29 String rutaProyecto = System.getProperty("user.dir");
30 File f = new File(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_login.txt");
31 System.out.println(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_login.txt");
32
33
34 }
35
36 }
37
38
```

Console Output:

```
terminated: Main [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (11 nov. 2021 16:02:23 - 16:02:29)
/Users/daguilar/eclipse/workspaces/ACT_04/ACT_04/src/archivos/datos_login.txt
```

Código:

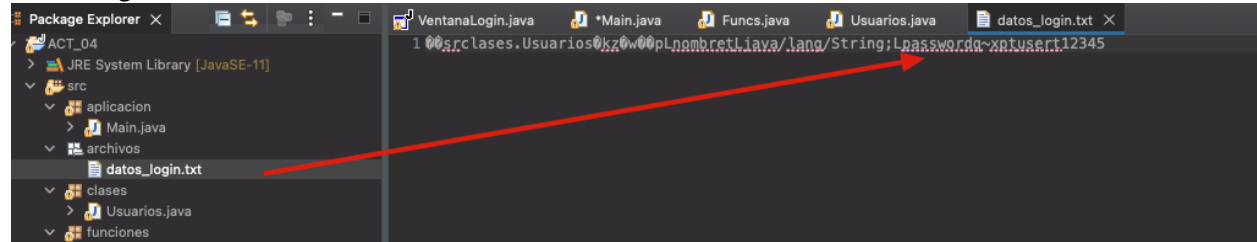
```
Usuarios usuario1 = new Usuarios("user", "12345");
String separador = File.separator;
String rutaProyecto = System.getProperty("user.dir");
File f = new File(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_login.txt");
System.out.println(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_login.txt");
;
```



Para crear el archivo donde estará almacenado el usuario y contraseña, usare una función que recibe por parámetro la ruta y el usuario:

```
public static void creaBaseDatos (File das_ruta, Usuarios das_usuario) throws IOException {  
    ObjectOutputStream das_oos = new ObjectOutputStream (new FileOutputStream(das_ruta));  
    das_oos.writeObject(das_usuario);  
}
```

Crea el siguiente archivo:



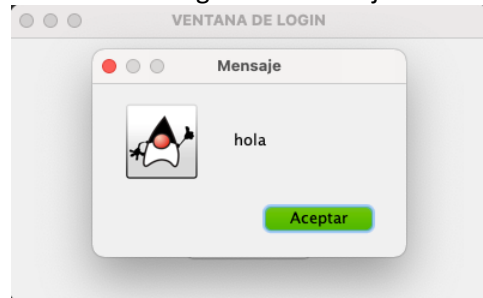
Vamos ahora a crear las acciones sobre el botón. Para ello haremos doble click sobre el botón en la vista de Design, y se nos mostrará la función:

```
JButton das_btnAcceder = new JButton("Aceptar");  
das_btnAcceder.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
    }  
});
```

Desde aquí primero añadiremos que si están vacíos los campos, aparezca el mensaje de que hay que introducir datos. Como test case para comprobar que funciona, hice lo siguiente:

```
JButton das_btnAcceder = new JButton("Aceptar");  
das_btnAcceder.addActionListener(new ActionListener() {  
    @SuppressWarnings("deprecation")  
    public void actionPerformed(ActionEvent e) {  
        // Si pulsamos el botón...  
        if (das_btnAcceder==e.getSource()) {  
            // Si está vacío...  
            if (das_textUser.getText().isEmpty() || das_textPass.getText().isEmpty() )  
            {  
                JOptionPane.showMessageDialog(null, "hola");  
            }  
        }  
    }  
});
```

Mostrando el siguiente mensaje al no introducir datos:



Una vez comprobado esto, procedí a configurarlo correctamente, me dí cuenta que si tenía el JLabel del título oculto "DEBES INTRODUCIR DATOS" por debajo de la función no podía llamar la función

“das\_libTitSinDatos.setVisible(true)” así que lo puse encima, esta sería la primera parte para mostrar el mensaje si no se introducen datos:

```
JButton das_btnAcceder = new JButton("Aceptar");
das_btnAcceder.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        // Si pulsamos el botón...
        if (das_btnAcceder==e.getSource()) {
            // Si está vacío...
            if (das_textUser.getText().isEmpty() || das_textPass.getText().isEmpty() )
        {
            das_lblTitSinDatos.setVisible(true);
        }
    }
});
```

Para controlar el usuario y password, he decidido hacerlo desde una función. Para ello he creado una función llamada checking que usando el ObjectInputStream lee el fichero creado anteriormente y crea un nuevo objeto “das\_acceso”. Una vez creado hace una comprobación recibiendo por parámetro los contenidos de los JTextFields para compararlos y retornar un true o un false.

Para comprobar primeramente que lee el archivo correctamente, he creado la función de esta manera para realizar un Test Case:

```
// Funcion para comprobar si el usuario y el password son correctos, si lo son devuelve true, si
no devuelve false
public static void checking (String das_usuario, String das_pass) throws FileNotFoundException,
IOException, ClassNotFoundException {
    String das_separador = File.separator;
    String das_rutaProyecto = System.getProperty("user.dir");
    File das_f = new File(das_rutaProyecto + das_separador + "src" + das_separador +
"archivos" + das_separador + "datos_login.txt");
    Usuarios das_acceso = new Usuarios();
    ObjectInputStream das_ois = new ObjectInputStream (new FileInputStream(das_f));

    das_acceso = (Usuarios) das_ois.readObject();

    System.out.println(das_acceso);
}
```

Resultado:

```
<terminated> Main [Java Application] /Library/Java/Java
Usuarios [nombre=user, password=12345]
```

Una vez hecho esto, simplemente he comprobado usando la función equals, que el parámetro recibido por parámetro del nombre de usuario y el parámetro recibido por parámetro del password son iguales que los almacenados en el archivo y leídos en la función con el ObjectInputStream. Si son correctos devuelve un true:

```
public static boolean checking (String das_usuario, String das_pass) throws
FileNotFoundException, IOException, ClassNotFoundException {
    boolean das_resultado = false;
    String das_separador = File.separator;
    String das_rutaProyecto = System.getProperty("user.dir");
    File das_f = new File(das_rutaProyecto + das_separador + "src" + das_separador +
"archivos" + das_separador + "datos_login.txt");
    Usuarios das_acceso = new Usuarios();
    ObjectInputStream das_ois = new ObjectInputStream (new FileInputStream(das_f));

    das_acceso = (Usuarios) das_ois.readObject();

    if (das_usuario.equals(das_acceso.getNombre()) &&
das_pass.equals(das_acceso.getPassword())) {
        das_resultado = true;
    }

    return das_resultado;
}
```

En la clase de la ventana quedaría de la siguiente forma, he añadido un test case que he realizado para comprobar que funcionaba correctamente. He tenido que almacenar el resultado de la función en una variable ya que al usarla en un IF daba problemas. La he dejado dentro de un try catch tal como me ha recomendado el IDE para poder usarla, ya que si no lo hacía me daba problemas igualmente

```
JButton das_btnAcceder = new JButton("Aceptar");
das_btnAcceder.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        // Realizo este paso ya que me daba problemas al usar la función directamente en un if,
        // por lo que almaceno el resultado de la función previamente en una variable // para utilizarla
        boolean das_result = false;;
        try {
            das_result = Funcs.checking(das_textUser.getText(), das_textPass.getText());
        } catch (ClassNotFoundException | IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        // Si pulsamos el botón...
        if (das_btnAcceder==e.getSource()) {
            // Si está vacío...
            if (das_textUser.getText().isEmpty() || das_textPass.getText().isEmpty() ) {
                das_lblTitSinDatos.setVisible(true);
            } else if (das_result == false) {
                das_lblTitErrorAccs.setVisible(true);
                // TEST CASE
            } else {
                JOptionPane.showMessageDialog(null, "hola");
            }
        }
    }
});
```

TEST CASE:



El código de la clase VentanaLogin ha quedado de la siguiente manera:

```
public class VentanaLogin extends JFrame {  
  
    private JPanel das_panelLogin;  
    private JTextField das_textUser;  
    private JPasswordField das_textPass;  
  
    public VentanaLogin() {  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        setBounds(100, 100, 400, 249);  
        das_panelLogin = new JPanel();  
        das_panelLogin.setBorder(new EmptyBorder(5, 5, 5, 5));  
        setContentPane(das_panelLogin);  
        das_panelLogin.setLayout(null);  
  
        setTitle("VENTANA DE LOGIN"); // TTITULO DE LA VENTANA  
        setLocationRelativeTo(null); // CENTRAR VENTANA  
  
        JLabel das_lblTitLog = new JLabel("LOGIN");  
        das_lblTitLog.setHorizontalAlignment(SwingConstants.CENTER);  
        das_lblTitLog.setFont(new Font("Dialog", Font.BOLD, 18));  
        das_lblTitLog.setBounds(6, 16, 388, 16);  
        das_panelLogin.add(das_lblTitLog);  
  
        JLabel das_lblTitUser = new JLabel("USUARIO");  
        das_lblTitUser.setBounds(74, 57, 61, 16);  
        das_panelLogin.add(das_lblTitUser);  
  
        JLabel das_lblTitPass = new JLabel("CONTRASEÑA");  
        das_lblTitPass.setBounds(74, 85, 94, 16);  
        das_panelLogin.add(das_lblTitPass);  
  
        das_textUser = new JTextField();  
        das_textUser.setBounds(181, 52, 130, 26);  
        das_panelLogin.add(das_textUser);  
        das_textUser.setColumns(10);  
  
        das_textPass = new JPasswordField();  
        das_textPass.setBounds(180, 80, 131, 26);  
        das_panelLogin.add(das_textPass);  
  
        JLabel das_lblTitSinDatos = new JLabel("DEBES INTRODUCIR LOS DATOS");  
        das_lblTitSinDatos.setVisible(false);  
        das_lblTitSinDatos.setHorizontalAlignment(SwingConstants.CENTER);  
    }  
}
```

```
das_lblTitSinDatos.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
das_lblTitSinDatos.setForeground(Color.RED);
das_lblTitSinDatos.setBounds(6, 132, 388, 16);
das_panelLogin.add(das_lblTitSinDatos);

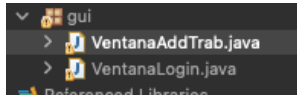
JLabel das_lblTitErrorAccs = new JLabel("CREDENCIALES INCORRECTAS");
das_lblTitErrorAccs.setVisible(false);
das_lblTitErrorAccs.setForeground(Color.RED);
das_lblTitErrorAccs.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
das_lblTitErrorAccs.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitErrorAccs.setBounds(6, 113, 388, 16);
das_panelLogin.add(das_lblTitErrorAccs);

JButton das_btnAcceder = new JButton("Aceptar");
das_btnAcceder.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        // Realizo este paso ya que me daba problemas al usar la función
        // almaceno el resultado de la función previamente en una variable
        boolean das_result = false;;
        try {
            das_result = Funcs.checking(das_textUser.getText(),
das_textPass.getText());
        } catch (ClassNotFoundException | IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        // Si pulsamos el botón...
        if (das_btnAcceder==e.getSource()) {
            // Si está vacío...
            if (das_textUser.getText().isEmpty() ||
das_textPass.getText().isEmpty() ) {
                das_lblTitSinDatos.setVisible(true);
            } else if (das_result == false) {
                das_lblTitErrorAccs.setVisible(true);
            }
            // TEST CASE
        } else {
            JOptionPane.showMessageDialog(null, "hola");
        }
    }
});

das_btnAcceder.setBounds(141, 160, 117, 29);
das_panelLogin.add(das_btnAcceder);
```

## 2. Segunda Ventana

Antes de enlazar una nueva ventana al botón anterior, voy a crearla. Para ello creo una nueva clase JFrame dentro del Package “gui”:



Abro el Design y empiezo a editar.

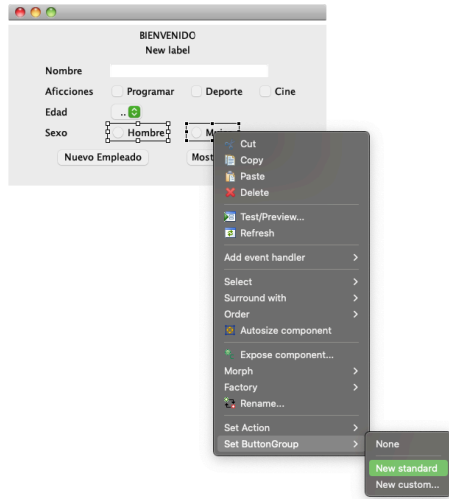
He creado la siguiente estructura, formada por:

- 6 JLabel
- 1 JTextField
- 3 JCheckBox
- 2 JRadioButton (Formando con ellos un ButtonGroup)
- 1 JComboBox
- 2 JButtons

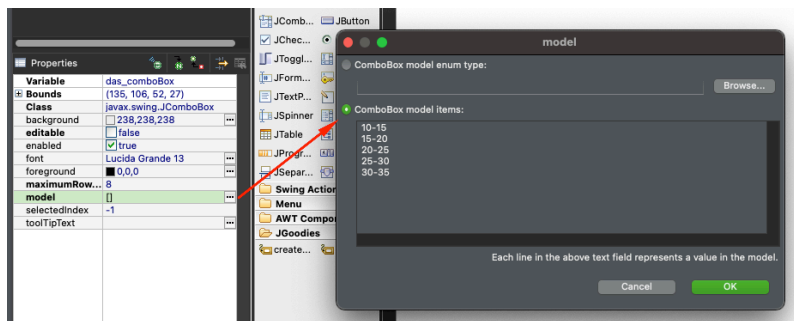
El resultado (Sin editar el JComboBox) es el siguiente:



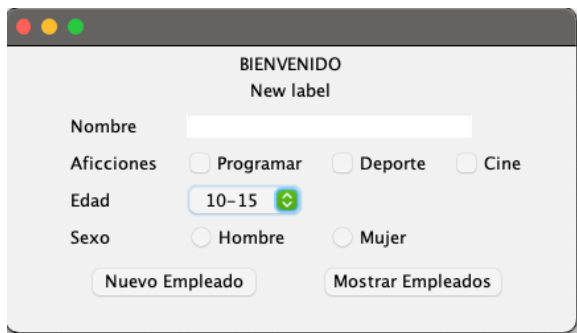
Para crear el ButtonGroup y que al clicar sobre los radio button, se active uno, u otro, selecciono los dos y con click derecho selecciono la siguiente opción:



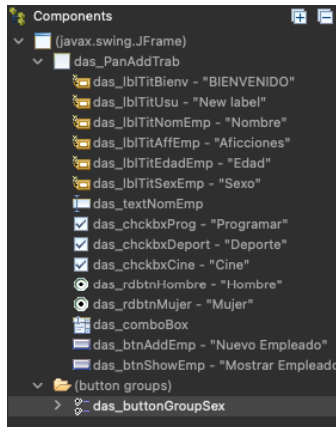
Y ahora, edito el JComboBox para añadir el rango de edades:



El resultado final sería el siguiente:



Las variables:



Para que aparezca en el titulo el nombre del usuario que se ha logeado, lo que hago es modificar el constructor de la ventana de añadir trabajadores, recibiendo por parámetro el usuario y usándolo para añadir el titulo. De esta forma quedaría el código de la clase VentanaAddTrab asi:

```
public class VentanaLogin extends JFrame {

    private JPanel das_panelLogin;
    private JTextField das_textUser;
    private JPasswordField das_textPass;

    public VentanaLogin() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 400, 249);
        das_panelLogin = new JPanel();
        das_panelLogin.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(das_panelLogin);
        das_panelLogin.setLayout(null);

        setTitle("VENTANA DE LOGIN"); // TTITULO DE LA VENTANA
        setLocationRelativeTo(null); // CENTRAR VENTANA

        JLabel das_lblTitLog = new JLabel("LOGIN");
        das_lblTitLog.setHorizontalAlignment(SwingConstants.CENTER);
        das_lblTitLog.setFont(new Font("Lucida Grande", Font.BOLD, 18));
        das_lblTitLog.setBounds(6, 16, 388, 16);
        das_panelLogin.add(das_lblTitLog);

        JLabel das_lblTitUser = new JLabel("USUARIO");
        das_lblTitUser.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
        das_lblTitUser.setBounds(74, 57, 61, 16);
        das_panelLogin.add(das_lblTitUser);

        JLabel das_lblTitPass = new JLabel("CONTRASEÑA");
        das_lblTitPass.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
        das_lblTitPass.setBounds(74, 85, 94, 16);
        das_panelLogin.add(das_lblTitPass);

        das_textUser = new JTextField();
        das_textUser.setBounds(181, 52, 130, 26);
        das_panelLogin.add(das_textUser);
        das_textUser.setColumns(10);

        das_textPass = new JPasswordField();
        das_textPass.setBounds(180, 80, 131, 26);
        das_panelLogin.add(das_textPass);
    }
}
```



```
JLabel das_lblTitSinDatos = new JLabel("DEBES INTRODUCIR LOS DATOS");
das_lblTitSinDatos.setVisible(false);
das_lblTitSinDatos.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitSinDatos.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
das_lblTitSinDatos.setForeground(Color.RED);
das_lblTitSinDatos.setBounds(6, 132, 388, 16);
das_panelLogin.add(das_lblTitSinDatos);

JLabel das_lblTitErrorAccs = new JLabel("CREDENCIALES INCORRECTAS");
das_lblTitErrorAccs.setVisible(false);
das_lblTitErrorAccs.setForeground(Color.RED);
das_lblTitErrorAccs.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
das_lblTitErrorAccs.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitErrorAccs.setBounds(6, 113, 388, 16);
das_panelLogin.add(das_lblTitErrorAccs);

JButton das_btnAcceder = new JButton("Aceptar");
das_btnAcceder.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
das_btnAcceder.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        // Realizo este paso ya que me daba problemas al usar la función
        // almaceno el resultado de la función previamente en una variable
        // directamente en un if, por lo que para utilizarla
        boolean das_result = false;;
        try {
            das_result = Funcs.checking(das_textUser.getText(),
das_textPass.getText());
        } catch (ClassNotFoundException | IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        // Si pulsamos el botón...
        if (das_btnAcceder==e.getSource()) {
            // Si está vacío...
            if (das_textUser.getText().isEmpty() ||
das_textPass.getText().isEmpty() ) {
                das_lblTitSinDatos.setVisible(true);
            } else if (das_result == false) {
                das_lblTitErrorAccs.setVisible(true);
            } else {
                // JOptionPane.showMessageDialog(null, "hola"); // TEST
                CASE
                VentanaAddTrab frame = new
VentanaAddTrab(das_textUser.getText());
                frame.setVisible(true);
                dispose();
            }
        }
    }
});

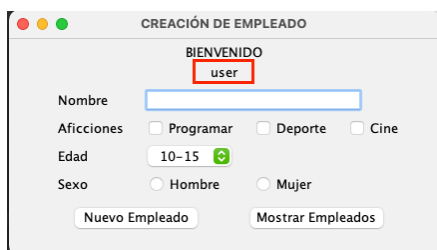
das_btnAcceder.setBounds(141, 160, 117, 29);
das_panelLogin.add(das_btnAcceder);
```

Voy a enlazar la ventana de Login con la ventana de creación de usuarios, el proceso será instanciar una ventana nueva, luego abrir la ventana de creación de trabajadores y cerrar la de login, añado el código completo del botón:

```
JButton das_btnAcceder = new JButton("Aceptar");
das_btnAcceder.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
das_btnAcceder.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        // Realizo este paso ya que me daba problemas al usar la función
        // almaceno el resultado de la función previamente en una variable
        boolean das_result = false;;
        try {
            das_result = Funcs.checking(das_textUser.getText(),
das_textPass.getText());
        } catch (ClassNotFoundException | IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        // Si pulsamos el botón...
        if (das_btnAcceder==e.getSource()) {
            // Si está vacío...
            if (das_textUser.getText().isEmpty() ||
das_textPass.getText().isEmpty() ) {
                das_lblTitSinDatos.setVisible(true);
            } else if (das_result == false) {
                das_lblTitErrorAccs.setVisible(true);
            } else {
                // JOptionPane.showMessageDialog(null, "hola"); // TEST
                CASE
                VentanaAddTrab frame = new
VentanaAddTrab(das_textUser.getText());
                frame.setVisible(true);
                dispose();
            }
        }
    }
});

das_btnAcceder.setBounds(141, 160, 117, 29);
das_panelLogin.add(das_btnAcceder);    };
```

Prueba de título:



### 3. Crear una nueva clase Empleado

Creo la siguiente clase Empleado que es Serializable:

```
package clases;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;

public class Empleado implements Serializable{

    private String nombre;
    private String edad;
    private String sexo;
    private ArrayList<String> aficciones = new ArrayList<String>();

    // CONSTRUCTOR
    public Empleado(String nombre, String edad, String sexo, ArrayList<String> aficciones) {
        super();
        this.nombre = nombre;
        this.edad = edad;
        this.sexo = sexo;
        this.aficciones = aficciones;
    }

    // GETTERS AND SETTERS

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEdad() {
        return edad;
    }

    public void setEdad(String edad) {
        this.edad = edad;
    }

    public String getSexo() {
        return sexo;
    }

    public void setSexo(String sexo) {
        this.sexo = sexo;
    }

    public ArrayList<String> getAficciones() {
        return aficciones;
    }

    public void setAficciones(ArrayList<String> aficciones) {
        this.aficciones = aficciones;
    }

    // toString
    @Override
    public String toString() {
        return "Empleado [nombre=" + nombre + ", edad=" + edad + ", sexo=" + sexo + ",\naficciones=" + aficciones + "]\n";
    }
}
```

Para comprobar que funciona correctamente, hago un Test Case en el Main creando un Empleado y pintándolo por consola:

```
// TEST CASE
// Creacion de un empleado
ArrayList<String> aficciones = new ArrayList<String>();
aficciones.add("Programar");
aficciones.add("Cine");
Empleado empleado1 = new Empleado("Dani","25-35","Hombre",aficciones);
System.out.println(empleado1);

Empleado [nombre=Dani, edad=25-35, sexo=Hombre, aficciones=[Programar, Cine]]
```

## 4. Crear nuevo empleado

### a. Comprobar si existen empleados previamente serializados en un archivo y crear el ArrayList con esos empleados

Lo primero de todo, voy a crear una función que comprobara si existe o no un archivo llamado "datos\_empleados.txt", este archivo seria el que tendría empleados previamente serializados. Si existe la función retornara un true, si no retornara un false:

```
// Funcion que comprueba si existe el archivo datos_empleados.txt, si existe retorna
true, si no existe retorna false y lo crea
public static boolean compruebaEmpleados () throws IOException {
    boolean resultado = true;
    String das_separador = File.separator;
    String das_rutaProyecto = System.getProperty("user.dir");
    File das_f = new File(das_rutaProyecto + das_separador + "src" + das_separador +
"archivos" + das_separador + "datos_empleados.txt");
    if (das_f.exists()) {
        resultado = true;
    } else {
        resultado = false;
        das_f.createNewFile();
    }
    return resultado;
}
```

Ahora en el Main creare un empleado mas, y un archivo datos\_empeledados.txt para este ejercicio. Para ello realizo el siguiente código:

```
// TEST CASES CON EMPLEADOS
// Creacion de un empleado
ArrayList<String> aficciones = new ArrayList<String>();
aficciones.add("Programar");
aficciones.add("Cine");
Empleado empleado1 = new Empleado("Dani","25-35","Hombre",aficciones);
// System.out.println(empleado1);

// Creacion de otro empleado y lista de empleados serializable para ejercicio 4.a
ArrayList<String> aficciones1 = new ArrayList<String>();
aficciones1.add("Deporte");
Empleado empleado2 = new Empleado("Alicia","15-25","Mujer",aficciones1);

// Creacion de archivo de empleados
File f1 = new
File(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_empleados.txt");
ArrayList<Empleado> aListEmpleados = new ArrayList<Empleado>();
aListEmpleados.add(empleado1);
aListEmpleados.add(empleado2);
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(f1));
oos.writeObject(aListEmpleados);
oos.close();
```

Ahora, pasamos a la acción del botón, lo primero de todo ejecutará la función que nos dirá si existe o no el archivo, si existe leera el archivo empleados.txt y creara un ArrayList con los empleados que contiene para devolverlo

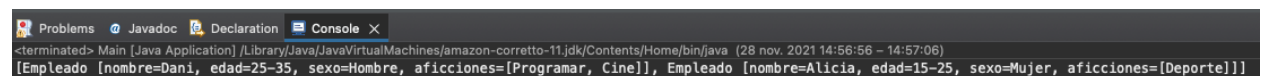
```
// Funcion que crea un ArrayList con los empleados del archivo empleados.txt, devuelve el
ArrayList
public static ArrayList<Empleado> creaArrayList() throws ClassNotFoundException,
IOException {
    String das_separador = File.separator;
    String das_rutaProyecto = System.getProperty("user.dir");
    File das_f = new File(das_rutaProyecto + das_separador + "src" + das_separador +
"archivos" + das_separador + "empleados.txt");
    ArrayList<Empleado> nuevosEmpleados = new ArrayList<Empleado>();

    ObjectInputStream ois = new ObjectInputStream(new FileInputStream(das_f));
    nuevosEmpleados = (ArrayList<Empleado>)ois.readObject();
    return nuevosEmpleados;
}
```

Hasta el momento, el botón quedaría de la siguiente manera, incluyo un Test Case con un ToString para comprobar que funciona correctamente:

```
JButton das_btnAddEmp = new JButton("Nuevo Empleado");
das_btnAddEmp.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Usa la funcion para comprobar si existen empleados, si no
        existen empleados...
        if (Funcs.compruebaEmpleados()) {
            ArrayList<Empleado> nuevosEmpleados = new
ArrayList<Empleado>();

            try {
                nuevosEmpleados = Funcs.creaArrayList();
                System.out.println(nuevosEmpleados.toString());
            } catch (ClassNotFoundException | IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    }
});
```



```
Problems Javadoc Declaration Console X
<terminated> Main [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (28 nov. 2021 14:56:56 - 14:57:06)
[Empleado [nombre=Dani, edad=25-35, sexo=Hombre, aficciones=[Programar, Cine]], Empleado [nombre=Alicia, edad=15-25, sexo=Mujer, aficciones=[Deporte]]]
```

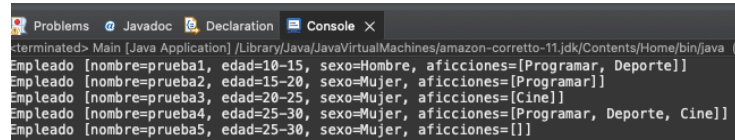
## b. Crear un nuevo empleado a partir de los datos introducidos por el usuario

Ahora vamos a extraer los datos para crear un nuevo empleado. Para ello realizamos el siguiente código:

```
// Creacion de nuevos empleados
// Filtrado de los checkbox
ArrayList<String> aficciones = new ArrayList<String>();
if (das_chckbxProg.isSelected()) {
    aficciones.add("Programar");
}
if (das_chckbxDeport.isSelected()) {
    aficciones.add("Deporte");
}
if (das_chckbxCine.isSelected()) {
    aficciones.add("Cine");
}
// Filtrado sexo para creacion final
if (das_rdbtnHombre.isSelected()) {
    Empleado empleadoNuevo = new
Empleado(das_textNomEmp.getText(), (String)das_comboBox.getSelectedItem(), das_rdbtnHombre.getText(
),aficciones);
    // TEST CASE
    System.out.println(empleadoH.toString());
}
```

```
    } else {  
        Empleado empleadoNuevo = new  
Empleado(das_textNomEmp.getText(), (String)das_comboBox.getSelectedItemAt(), das_rdbtnMujer.getText()  
, aficciones);  
        // TEST CASE  
        System.out.println(empleadoM.toString());  
    }
```

Lo que estamos haciendo es primero creando un arrayList para las aficciones, luego filtrar dependiendo de las opciones que se hayan seleccionado. Por último, dependiendo de si se ha seleccionado hombre o mujer, se crea un empleado. He creado un pequeño Test Case para pintar por consola y probar que funcionaba correctamente:



```
Empleado [nombre=prueba1, edad=10-15, sexo=Hombre, aficciones=[Programar, Deporte]]  
Empleado [nombre=prueba2, edad=15-20, sexo=Mujer, aficciones=[Programar]]  
Empleado [nombre=prueba3, edad=20-25, sexo=Mujer, aficciones=[Cine]]  
Empleado [nombre=prueba4, edad=25-30, sexo=Mujer, aficciones=[Programar, Deporte, Cine]]  
Empleado [nombre=prueba5, edad=25-30, sexo=Mujer, aficciones=[]]
```

#### c. Añadir al arrayList de empleados el nuevo empleado

Para esto, agregamos la siguiente línea debajo de los if de agregar nuevos empleados mostrados en el punto anterior:

```
        if (das_rdbtnHombre.isSelected()) {  
            Empleado empleadoNuevo = new  
Empleado(das_textNomEmp.getText(), (String)das_comboBox.getSelectedItemAt(), das_rdbtnHombre.getText()  
, aficciones);  
            // Añadir al arrayList de empleados el empleado  
            nuevosEmpleados.add(empleadoNuevo);  
            // TEST CASE  
            System.out.println(empleadoH.toString());  
        } else {  
            Empleado empleadoNuevo = new  
Empleado(das_textNomEmp.getText(), (String)das_comboBox.getSelectedItemAt(), das_rdbtnMujer.getText()  
, aficciones);  
            // Añadir al arrayList de empleados el empleado  
            nuevosEmpleados.add(empleadoNuevo);  
            // TEST CASE  
            System.out.println(empleadoM.toString());  
        }
```

#### d. Serializar la lista de empleados y guardarlos en un archivo de nombre datos\_empleados

Para serializar empleados, hare uso de una función extra, que recibe por parámetro el array y crea el archivo serializado:

```
// Funcion serializar empleados  
public static void serializaEmpleados (ArrayList<Empleado> empleados) throws  
FileNotFoundException, IOException {  
    String das_separador = File.separator;  
    String das_rutaProyecto = System.getProperty("user.dir");  
    File das_f = new File(das_rutaProyecto + das_separador + "src" + das_separador +  
"archivos" + das_separador + "datos_empleados.txt");  
    ObjectOutputStream das_oos = new ObjectOutputStream (new FileOutputStream(das_f));  
    das_oos.writeObject(empleados);  
    das_oos.close();  
}
```

En el botón se le llamaría de la siguiente manera:

```
// Serializar nuevos empleados  
try {  
    Funcs.serializaEmpleados(nuevosEmpleados);  
} catch (IOException e1) {  
    // TODO Auto-generated catch block  
    e1.printStackTrace();  
}
```

e. Cuadro de dialogo si todo es correcto o si no se ha rellenado nada

En este case se trata de JDialog, serán dos, uno para si se ha realizado con éxito, y otro si no. Para ello añadido el siguiente filtrado una vez se hace click en el botón de nuevo empleado:

```
// Comprobaciones previas campos vacios
// Variables para controlar que las dos validaciones
boolean comp1 = false;
boolean comp2 = false;
// Validacion de nombre
if (das_textNomEmp.getText().isEmpty()) {
    JDialog das_sinNombre = new JDialog();
    JLabel das_mensajeSinNombre = new JLabel("Debes introducir
un nombre");

    das_sinNombre.getContentPane().add(das_mensajeSinNombre);
    das_sinNombre.setSize(300, 100);
    das_sinNombre.setLocationRelativeTo(null);
    das_sinNombre.setVisible(true);
    comp1 = false;
} else {
    comp1 = true;
}

// Validacion de los checkbox
if (!das_chckbxProg.isSelected() && !das_chckbxDeport.isSelected() &&
!das_chckbxCine.isSelected()) {
    JDialog das_sinAficcion = new JDialog();
    JLabel das_mensajeSinAficcion = new JLabel("Debes
introducir una aficcion");

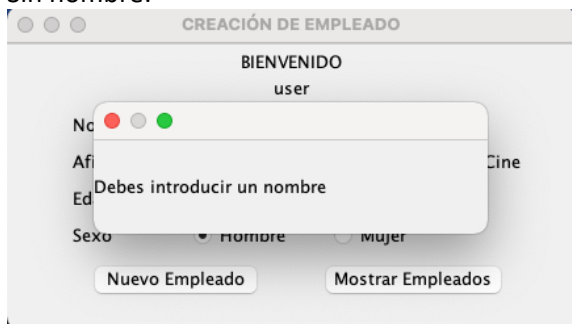
    das_sinAficcion.getContentPane().add(das_mensajeSinAficcion);
    das_sinAficcion.setSize(300, 100);
    das_sinAficcion.setLocationRelativeTo(null);
    das_sinAficcion.setVisible(true);
    comp2 = false;
} else {
    comp2 = true;
}
```

Para el la comprobación de si se ha seleccionado el sexo no lo he puesto ya que tengo seleccionado que este ya seleccionado uno de los dos radio buttons por defecto.

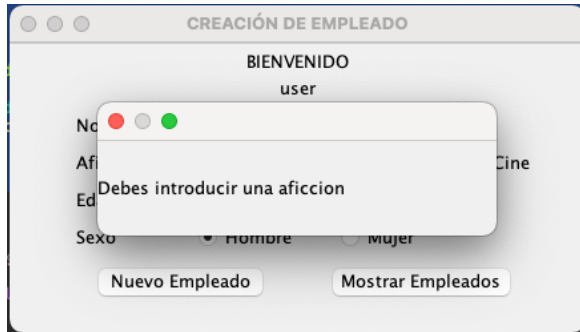
La comprobación tiene dos variables, estas tienen que encontrarse las dos en true para poder crearse un usuario.

TEST CASE:

Sin nombre:



Sin afición:



Para el mensaje de todo es correcto, lo pondremos una vez termina la serialización, quedando de esta manera

```
// Dialogo una vez se han creado los usuarios
JDialog das_gracias = new JDialog();
JLabel das_mensajeGracias = new JLabel("Gracias por usar la
aplicacion");

das_gracias.getContentPane().add(das_mensajeGracias);
das_gracias.setSize(300, 100);
das_gracias.setLocationRelativeTo(null);
das_gracias.setVisible(true);
```





## 5. Boton Muestra empleado

### a. Obtener el listado de empleados serializados

Usando la función creaArrayList utilizada anteriormente, extraigo un ArrayList de los empleados para crear un listado actual de los empleados que hay almacenados

```
public void actionPerformed(ActionEvent e) {  
    ArrayList<Empleado> das_listadoEmpleados = new  
ArrayList<Empleado>();  
    try {  
        das_listadoEmpleados = Funcs.creaArrayList();  
    } catch (ClassNotFoundException | IOException e1) {  
        // TODO Auto-generated catch block  
        e1.printStackTrace();  
    }  
}
```

### b. Cuadro de dialogo con todos los empleados

Para esto diseño un botón que primero hace el paso anterior de obtener el arraylist de empleados, después instancia un JDialog, le establece un titulo, lo localiza en el centro de la pantalla y lo hace visible.

Después crea un JLabel y una variable para su contenido. Realizando un bucle for, crea ese contenido junto con contenido HTML.

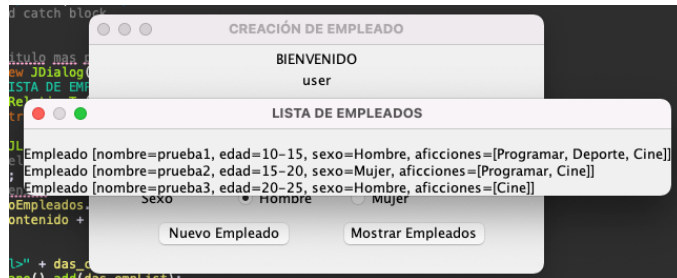
Una vez se tiene todo el contenido, se inyecta con .setText en el JLabel, y se publica.

Por último, uso la función .pack() para ajustar dinámicamente el tamaño de la pantalla.

Código completo del botón:

```
JButton das_btnShowEmp = new JButton("Mostrar Empleados");  
das_btnShowEmp.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        // ArrayList para almacenar los empleados  
        ArrayList<Empleado> das_listadoEmpleados = new  
ArrayList<Empleado>();  
        try {  
            // Usa la funcion devuelveEmpleados() que recoge los  
empleados serializados en el archivo datos_empleados.txt, los almacena en el ArrayList  
            das_listadoEmpleados = Funcs.devuelveEmpleados();  
        } catch (ClassNotFoundException | IOException e1) {  
            // TODO Auto-generated catch block  
            e1.printStackTrace();  
        }  
        // JDialog y JLabel del titulo mas propiedades  
        JDialog das_listEmple = new JDialog();  
        das_listEmple.setTitle("LISTA DE EMPLEADOS");  
        das_listEmple.setLocationRelativeTo(null);  
        das_listEmple.setVisible(true);  
  
        JLabel das_empList = new JLabel("");  
        // Contenido para el JLabel  
        String das_contenido = "";  
        // For para crear el contenido  
        for(int i=0; i<das_listadoEmpleados.size();i++) {  
            das_contenido = das_contenido + "<br>" +  
das_listadoEmpleados.get(i);  
        }  
        // Insertar el contenido  
        das_empList.setText("<html>" + das_contenido + "</html>");  
        das_listEmple.getContentPane().add(das_empList);  
        // Establecer un tamaño relativo según el contenido  
        das_listEmple.pack();  
    }  
});
```

#### TEST CASE



## 6. TODOS LOS CÓDIGOS COMPLETOS

Dado que durante el propio ejercicio he ido modificando el código para hacer ciertos cambios, añado el código completo de cada una de las clases

### a. APLICACIÓN/Main.java

```
package aplicacion;

import java.awt.EventQueue;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

import clases.Empleado;
import clases.Usuarios;
import funciones.Funcs;
import gui.VentanaLogin;

public class Main {

    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // INICIAR VENTANA DE LOGIN
        // He utilizado la sintaxis por defecto que me ha generado eclipse y la he
        // exportado a una clase main aparte
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    VentanaLogin frame = new VentanaLogin();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

        // Creo un usuario para usarlo como login
        Usuarios usuario1 = new Usuarios("user","12345");

        // Variables para la ruta del File
        String separador = File.separator;
```

```
String rutaProyecto = System.getProperty("user.dir");
File f = new
File(rutaProyecto+separador+"archivos"+separador+"datos_login.txt");
if (!f.exists()) {
    f.createNewFile();
}
// TEST CASE
//
System.out.println(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_login.txt")
;

// Escribir un archivo con los datos de una clase
Funcs.creaBaseDatos(f,usuario1);

/*
// TEST CASES CON EMPLEADOS
// Creacion de un empleado
ArrayList<String> aficciones = new ArrayList<String>();
aficciones.add("Programar");
aficciones.add("Cine");
Empleado empleado1 = new Empleado("Dani","25-35","Hombre",aficciones);
// System.out.println(empleado1);

// Creacion de otro empleado y lista de empleados serializable para ejercicio 4.a
ArrayList<String> aficciones1 = new ArrayList<String>();
aficciones1.add("Deporte");
Empleado empleado2 = new Empleado("Alicia","15-25","Mujer",aficciones1);

// Creacion de archivo de empleados
File f1 = new
File(rutaProyecto+separador+"src"+separador+"archivos"+separador+"datos_empleados.txt");
ArrayList<Empleado> aListEmpleados = new ArrayList<Empleado>();
aListEmpleados.add(empleado1);
aListEmpleados.add(empleado2);
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(f1));
oos.writeObject(aListEmpleados);
oos.close();

*/
}
```

## b. CLASES/Empleado.java

```
package clases;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;

public class Empleado implements Serializable{

    private String nombre;
    private String edad;
    private String sexo;
    private ArrayList<String> aficciones = new ArrayList<String>();

    // CONSTRUCTOR
    public Empleado(String nombre, String edad, String sexo, ArrayList<String> aficciones) {
        super();
        this.nombre = nombre;
        this.edad = edad;
        this.sexo = sexo;
        this.aficciones = aficciones;
    }

    // GETTERS AND SETTERS

    public String getNombre() {
        return nombre;
    }
}
```

```
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getEdad() {
    return edad;
}

public void setEdad(String edad) {
    this.edad = edad;
}

public String getSexo() {
    return sexo;
}

public void setSexo(String sexo) {
    this.sexo = sexo;
}

public ArrayList<String> getAficciones() {
    return aficciones;
}

public void setAficciones(ArrayList<String> aficciones) {
    this.aficciones = aficciones;
}

// toString
@Override
public String toString() {
    return "Empleado [nombre=" + nombre + ", edad=" + edad + ", sexo=" + sexo + ",\naficciones=" + aficciones + "]\n";
}
```

### c. CLASES/Usuarios.java

```
package clases;

import java.io.Serializable;

public class Usuarios implements Serializable{

    private String nombre;
    private String password;

    public Usuarios(String nombre, String password) {
        super();
        this.nombre = nombre;
        this.password = password;
    }

    public Usuarios() {
        super();
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
```

```
        this.password = password;
    }

    @Override
    public String toString() {
        return "Usuarios [nombre=" + nombre + ", password=" + password + "]";
    }
}
```

#### d. FUNCIONES/Funcs.java

```
package funciones;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

import clases.Empleado;
import clases.Usuarios;

public class Funcs {

    // Función Para crear archivo de datos de usuario de login
    public static void creaBaseDatos (File das_ruta, Usuarios das_usuario) throws IOException
    {
        ObjectOutputStream das_oos = new ObjectOutputStream (new
        FileOutputStream(das_ruta));
        das_oos.writeObject(das_usuario);
        das_oos.close();
    }

    // Funcion para comprobar si el usuario y el password son correctos, si lo son devuelve
    true, si no
    // devuelve false. Recibe por parametros el usuario y password de los JTextFields
    public static boolean checking (String das_usuario, String das_pass) throws
    FileNotFoundException, IOException, ClassNotFoundException {
        boolean das_resultado = false;
        String das_separador = File.separator;
        String das_rutaProyecto = System.getProperty("user.dir");
        File das_f = new File(das_rutaProyecto + das_separador + "archivos" +
        das_separador + "datos_login.txt");
        Usuarios das_acceso = new Usuarios();
        ObjectInputStream das_ois = new ObjectInputStream (new FileInputStream(das_f));

        das_acceso = (Usuarios) das_ois.readObject();

        if (das_usuario.equals(das_acceso.getNombre()) &&
        das_pass.equals(das_acceso.getPassword())) {
            das_resultado = true;
        }
        das_ois.close();
        return das_resultado;
    }

    // Funcion que comprueba si existe el archivo datos_empleados.txt, si existe retorna
    true, si no existe retorna false y lo crea
    public static boolean compruebaEmpleados () throws IOException {
        boolean resultado = true;
        String das_separador = File.separator;
        String das_rutaProyecto = System.getProperty("user.dir");
        File das_f = new File(das_rutaProyecto + das_separador + "archivos" +
        das_separador + "datos_empleados.txt");
        if (das_f.exists()) {
            resultado = true;
        }
    }
}
```

```
        } else {
            resultado = false;
            das_f.createNewFile();
        }
        return resultado;
    }

    // Funcion que crea un ArrayList con los empleados del archivo datos_empleados.txt,
    // devuelve el arrayList
    public static ArrayList<Empleado> creaArrayList() throws ClassNotFoundException,
    IOException {
        String das_separador = File.separator;
        String das_rutaProyecto = System.getProperty("user.dir");
        File das_f = new File(das_rutaProyecto + das_separador + "archivos" +
das_separador + "datos_empleados.txt");
        ArrayList<Empleado> nuevosEmpleados = new ArrayList<Empleado>();

        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(das_f));
        nuevosEmpleados = (ArrayList<Empleado>)ois.readObject();
        ois.close();
        return nuevosEmpleados;
    }

    // Funcion serializar empleados
    public static void serializaEmpleados (ArrayList<Empleado> empleados) throws
    FileNotFoundException, IOException {
        String das_separador = File.separator;
        String das_rutaProyecto = System.getProperty("user.dir");
        File das_f = new File(das_rutaProyecto + das_separador + "archivos" +
das_separador + "datos_empleados.txt");
        ObjectOutputStream das_oos = new ObjectOutputStream (new FileOutputStream(das_f));
        das_oos.writeObject(empleados);
        das_oos.close();
    }
}
```

#### e. GUI/VentanaLogin.java

```
package gui;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import funciones.Funcs;

import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.awt.event.ActionEvent;

public class VentanaLogin extends JFrame {

    private JPanel das_panelLogin;
    private JTextField das_textUser;
    private JPasswordField das_textPass;

    public VentanaLogin() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 400, 249);
        das_panelLogin = new JPanel();
        das_panelLogin.setBorder(new EmptyBorder(5, 5, 5, 5));
    }
}
```

```
setContentPane(das_panelLogin);
das_panelLogin.setLayout(null);

setTitle("VENTANA DE LOGIN"); // TITULO DE LA VENTANA
setLocationRelativeTo(null); // CENTRAR VENTANA

JLabel das_lblTitLog = new JLabel("LOGIN");
das_lblTitLog.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitLog.setFont(new Font("Lucida Grande", Font.BOLD, 18));
das_lblTitLog.setBounds(6, 16, 388, 16);
das_panelLogin.add(das_lblTitLog);

JLabel das_lblTitUser = new JLabel("USUARIO");
das_lblTitUser.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
das_lblTitUser.setBounds(74, 57, 61, 16);
das_panelLogin.add(das_lblTitUser);

JLabel das_lblTitPass = new JLabel("CONTRASEÑA");
das_lblTitPass.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
das_lblTitPass.setBounds(74, 85, 94, 16);
das_panelLogin.add(das_lblTitPass);

das_textUser = new JTextField();
das_textUser.setBounds(181, 52, 130, 26);
das_panelLogin.add(das_textUser);
das_textUser.setColumns(10);

das_textPass = new JPasswordField();
das_textPass.setBounds(180, 80, 131, 26);
das_panelLogin.add(das_textPass);

JLabel das_lblTitSinDatos = new JLabel("DEBES INTRODUCIR LOS DATOS");
das_lblTitSinDatos.setVisible(false);
das_lblTitSinDatos.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitSinDatos.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
das_lblTitSinDatos.setForeground(Color.RED);
das_lblTitSinDatos.setBounds(6, 132, 388, 16);
das_panelLogin.add(das_lblTitSinDatos);

JLabel das_lblTitErrorAccs = new JLabel("CREDENCIALES INCORRECTAS");
das_lblTitErrorAccs.setVisible(false);
das_lblTitErrorAccs.setForeground(Color.RED);
das_lblTitErrorAccs.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
das_lblTitErrorAccs.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitErrorAccs.setBounds(6, 113, 388, 16);
das_panelLogin.add(das_lblTitErrorAccs);

JButton das_btnAcceder = new JButton("Aceptar");
das_btnAcceder.setFont(new Font("Lucida Grande", Font.PLAIN, 13));
das_btnAcceder.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        // Realizo este paso ya que me daba problemas al usar la función
        // almaceno el resultado de la función previamente en una variable
        boolean das_result = false;;
        try {
            das_result = Funcs.checking(das_textUser.getText(),
das_textPass.getText());
        } catch (ClassNotFoundException | IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        // Si pulsamos el botón...
        if (das_btnAcceder==e.getSource()) {
```

```
        // Si está vacío...
        if (das_textUser.getText().isEmpty() ||
das_textPass.getText().isEmpty() ) {
            das_lblTitSinDatos.setVisible(true);
        } else if (das_result == false) {
            das_lblTitErrorAccs.setVisible(true);
        } else {
            // JOptionPane.showMessageDialog(null, "hola"); // TEST
CASE
            VentanaAddTrab frame = new
VentanaAddTrab(das_textUser.getText());
            frame.setVisible(true);
            dispose();
        }
    }
    });

    das_btnAcceder.setBounds(141, 160, 117, 29);
    das_panelLogin.add(das_btnAcceder);
}
```

#### f. GUI/VentanaAddTrab.java

```
package gui;
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import clases.Empleado;
import clases.Usuarios;
import funciones.Funcs;

import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JCheckBox;
import javax.swing.JRadioButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JButton;
import javax.swing.ButtonGroup;
import javax.swing.DefaultComboBoxModel;
import java.awt.event.ActionListener;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.awt.event.ActionEvent;

public class VentanaAddTrab extends JFrame {

    private JPanel das_PanAddTrab;
    private JTextField das_textNomEmp;
    private final ButtonGroup das_buttonGroupSex = new ButtonGroup();

    public VentanaAddTrab(String titulo) {

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 433, 246);
        das_PanAddTrab = new JPanel();
        das_PanAddTrab.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(das_PanAddTrab);
        das_PanAddTrab.setLayout(null);

        setTitle("CREACIÓN DE EMPLEADO"); // TTITULO DE LA VENTANA
    }
}
```



```
setLocationRelativeTo(null); // CENTRAR VENTANA

JLabel das_lblTitBienv = new JLabel("BIENVENIDO");
das_lblTitBienv.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitBienv.setBounds(6, 6, 421, 16);
das_PanAddTrab.add(das_lblTitBienv);

JLabel das_lblTitUsu = new JLabel(titulo);
das_lblTitUsu.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTitUsu.setBounds(6, 26, 421, 16);
das_PanAddTrab.add(das_lblTitUsu);

JLabel das_lblTitNomEmp = new JLabel("Nombre");
das_lblTitNomEmp.setBounds(50, 54, 61, 16);
das_PanAddTrab.add(das_lblTitNomEmp);

JLabel das_lblTitAffEmp = new JLabel("Aficciones");
das_lblTitAffEmp.setBounds(50, 82, 72, 16);
das_PanAddTrab.add(das_lblTitAffEmp);

JLabel das_lblTitEdadEmp = new JLabel("Edad");
das_lblTitEdadEmp.setBounds(50, 110, 61, 16);
das_PanAddTrab.add(das_lblTitEdadEmp);

JLabel das_lblTitSexEmp = new JLabel("Sexo");
das_lblTitSexEmp.setBounds(50, 138, 61, 16);
das_PanAddTrab.add(das_lblTitSexEmp);

das_textNomEmp = new JTextField();
das_textNomEmp.setBounds(135, 49, 222, 26);
das_PanAddTrab.add(das_textNomEmp);
das_textNomEmp.setColumns(10);

JCheckBox das_chckbxProg = new JCheckBox("Programar");
das_chckbxProg.setBounds(134, 78, 96, 23);
das_PanAddTrab.add(das_chckbxProg);

JCheckBox das_chckbxDeport = new JCheckBox("Deporte");
das_chckbxDeport.setBounds(242, 78, 82, 23);
das_PanAddTrab.add(das_chckbxDeport);

JCheckBox das_chckbxCine = new JCheckBox("Cine");
das_chckbxCine.setBounds(336, 78, 72, 23);
das_PanAddTrab.add(das_chckbxCine);

JRadioButton das_rdbtnHombre = new JRadioButton("Hombre");
das_rdbtnHombre.setSelected(true);
das_buttonGroupSex.add(das_rdbtnHombre);
das_rdbtnHombre.setBounds(135, 134, 82, 23);
das_PanAddTrab.add(das_rdbtnHombre);

JRadioButton das_rdbtnMujer = new JRadioButton("Mujer");
das_buttonGroupSex.add(das_rdbtnMujer);
das_rdbtnMujer.setBounds(242, 134, 72, 23);
das_PanAddTrab.add(das_rdbtnMujer);

JComboBox das_comboBox = new JComboBox();
das_comboBox.setModel(new DefaultComboBoxModel(new String[] {"10-15", "15-20",
"20-25", "25-30", "30-35"}));
das_comboBox.setBounds(135, 106, 95, 27);
das_PanAddTrab.add(das_comboBox);

JButton das_btnAddEmp = new JButton("Nuevo Empleado");
das_btnAddEmp.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ArrayList<Empleado> nuevosEmpleados = new ArrayList<Empleado>();
        // La declaracion del array para los empleados es global
        // Comprobaciones previas campos vacios
```

```
// Variables para controlar que las dos validaciones
boolean comp1 = false;
boolean comp2 = false;
// Validacion de nombre
if (das_textNomEmp.getText().isEmpty()) {
    JDialog das_sinNombre = new JDialog();
    JLabel das_mensajeSinNombre = new JLabel("Debes introducir
un nombre");

    das_sinNombre.getContentPane().add(das_mensajeSinNombre);
    das_sinNombre.setSize(300, 100);
    das_sinNombre.setLocationRelativeTo(null);
    das_sinNombre.setVisible(true);
    comp1 = false;
} else {
    comp1 = true;
}

// Validacion de los checkbox
if (!das_chckbxProg.isSelected() && !das_chckbxDeport.isSelected() &&
!das_chckbxCine.isSelected()) {
    JDialog das_sinAficcion = new JDialog();
    JLabel das_mensajeSinAficcion = new JLabel("Debes
introducir una aficcion");

    das_sinAficcion.getContentPane().add(das_mensajeSinAficcion);
    das_sinAficcion.setSize(300, 100);
    das_sinAficcion.setLocationRelativeTo(null);
    das_sinAficcion.setVisible(true);
    comp2 = false;
} else {
    comp2 = true;
}

// Si las dos validaciones son correctas, crea los usuarios
if (comp1 == true && comp2 == true) {
    // Usa la funcion para comprobar si existen empleados, si existen
    // crea un arrayList nuevosEmpleados donde añade los empleados del archivo donde estan ya
    // almacenados
    try {
        if (Funcs.compruebaEmpleados()) {

            // Si se hace esto (Si existe ya un fichero, se pisa
            // sobre si mismo y jamas habra mas empleados que 2 + 1), editar en el fichero de texto

            try {
                nuevosEmpleados = Funcs.creaArrayList();

                // TEST CASE
                System.out.println(nuevosEmpleados.toString());
            } catch (ClassNotFoundException | IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }

            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    // Creacion de nuevos empleados
    // Filtrado de los checkbox
    ArrayList<String> aficciones = new
ArrayList<String>();

    if (das_chckbxProg.isSelected()) {
        aficciones.add("Programar");
    }
    if (das_chckbxDeport.isSelected()) {
```

```
        aficciones.add("Deporte");
    }
    if (das_chckbxCine.isSelected()) {
        aficciones.add("Cine");
    }
    // Filtrado sexo para creacion final
    if (das_rdbtnHombre.isSelected()) {
        Empleado empleadoNuevo = new
Empleado(das_textNomEmp.getText(),(String)das_comboBox.getSelectedItemAt(),das_rdbtnHombre.getText(
),aficciones);
        // Añadir al arrayList de empleados el
empleado nuevo
        nuevosEmpleados.add(empleadoNuevo);
        // TEST CASE
        //
        System.out.println(empleadoNuevo.toString());
    } else {
        Empleado empleadoNuevo = new
Empleado(das_textNomEmp.getText(),(String)das_comboBox.getSelectedItemAt(),das_rdbtnMujer.getText(
),aficciones);
        // Añadir al arrayList de empleados el
empleado nuevo
        nuevosEmpleados.add(empleadoNuevo);
        // TEST CASE
        //
        System.out.println(empleadoNuevo.toString());
    }

    // Serializar nuevos empleados
    try {
        Funcs.serializaEmpleados(nuevosEmpleados);
    } catch (IOException e1) {
        System.out.println("Error");
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    // Dialogo una vez se han creado los usuarios
    JDialog das_gracias = new JDialog();
    JLabel das_mensajeGracias = new JLabel("Gracias por usar la
aplicacion");

    das_gracias.getContentPane().add(das_mensajeGracias);
    das_gracias.setSize(300, 100);
    das_gracias.setLocationRelativeTo(null);
    das_gracias.setVisible(true);

    } else {
        // Mensaje que se muestra por consola cuando las validaciones no
son correctas y no se está creando el usuario
        System.out.println("No se ha creado ningun usuario");
    }

    }

});
das_btnAddEmp.setBounds(60, 166, 137, 29);
das_PanAddTrab.add(das_btnAddEmp);

JButton das_btnShowEmp = new JButton("Mostrar Empleados");
das_btnShowEmp.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // ArrayList para almacenar los empleados
        ArrayList<Empleado> das_listadoEmpleados = new
ArrayList<Empleado>();
        try {
            // Usa la funcion devuelveEmpleados() que recoge los
empleados serializados en el archivo datos_empleados.txt, los almacena en el ArrayList
```

```
        das_listadoEmpleados = Funcs.creaArrayList();
    } catch (ClassNotFoundException | IOException e1) {
        // TODO Auto-generated catch block
        System.out.println("No existen empleados serializados");
    }
    // JDialog y JLabel del titulo mas propiedades
    JDialog das_listEmple = new JDialog();
    das_listEmple.setTitle("LISTA DE EMPLEADOS");
    das_listEmple.setLocationRelativeTo(null);
    das_listEmple.setVisible(true);

    JLabel das_emplList = new JLabel("");
    // Contenido para el JLabel
    String das_contenido = "";
    // For para crear el contenido
    for(int i=0; i<das_listadoEmpleados.size();i++) {
        das_contenido = das_contenido + "<br>" +
das_listadoEmpleados.get(i);
    }
    // Insertar el contenido
    das_emplList.setText("<html>" + das_contenido + "</html>");
    das_listEmple.getContentPane().add(das_emplList);
    // Establecer un tamaño relativo según el contenido
    das_listEmple.pack();
    }
});

das_btnShowEmp.setBounds(236, 166, 147, 29);
das_PanAddTrab.add(das_btnShowEmp);
}
```

## 7. Fe de erratas

En el código escrito aquí puede leerse el error ortográfico “aficciones”, esté esta corregido en el código final ejecutable