Ejercicio 3

Finalmente, cuando se presione en el botón "Enviar", vamos a validar que cada uno de los campos se encuentren rellenos. Para esto debemos agregar un último JLabel vacío debajo del botón "Enviar". Además, después de pasar las validaciones, debemos construir un objeto de tipo Persona con los datos ingresados y sobrescribir el método toString para que muestre los tres campos concatenados con comas.

Solución:

```
1.  class Persona {
2.
3.      private String nombre;
4.      private String apellidos;
5.      private String direccion;
6.
7.      public Persona (String nombre, String apellidos, String direccion)
    {
8.          this.nombre = nombre;
9.          this.apellidos = apellidos;
10.         this.direccion = direccion;
11.     }
12.
13.     public String getNombre() {
14.         return nombre;
15.     }
16.
17.     public void setNombre(String nombre) {
18.         this.nombre = nombre;
19.     }
20.
21.     public String getApellidos() {
22.         return apellidos;
23.     }
24.
25.     public void setApellidos(String apellidos) {
26.         this.apellidos = apellidos;
27.     }
28.
29.     public String getDireccion() {
30.         return direccion;
31.     }
32.
33.     public void setDireccion(String direccion) {
```

```
34.         this.direccion = direccion;
35.     }
36.
37.     @Override
38.     public String toString(){
39.       return nombre + "," + apellidos + "," + direccion;
40.     }
41.
42. }
```

La clase de la interfaz grafica quedaría:

```
1.  import java.awt.event.ActionEvent;
2.  import java.awt.event.ActionListener;
3.  import javax.swing.*;
4.
5.  public class example {
6.
7.      public static void main(String args[]) {
8.          JFrame a = new JFrame("Ejemplo");
9.
10.         JLabel jlNombres = new JLabel("Nombres:");
11.         jlNombres.setBounds(50,50,200,30);
12.         a.add(jlNombres);
13.
14.         JTextField jtfNombres = new JTextField("");
15.         jtfNombres.setBounds(50,100,200,30);
16.         a.add(jtfNombres);
17.
18.         JLabel jlApellidos = new JLabel("Apellidos:");
19.         jlApellidos.setBounds(50,150,200,30);
20.         a.add(jlApellidos);
21.
22.         JTextField jtfApellidos = new JTextField("");
23.         jtfApellidos.setBounds(50,200,200,30);
24.         a.add(jtfApellidos);
25.
26.         JLabel jlDireccion = new JLabel("Direccion:");
27.         jlDireccion.setBounds(50,250,200,30);
28.         a.add(jlDireccion);
29.
30.         JTextField jtfDireccion = new JTextField("");
31.         jtfDireccion.setBounds(50,300,200,30);
32.         a.add(jtfDireccion);
33.
```

```
34.       JButton jbEnviar = new JButton("Enviar");
35.       jbEnviar.setBounds(100,350,100,30);
36.       a.add(jbEnviar);
37.
38.       final JLabel jlMensaje = new JLabel("");
39.       jlMensaje.setBounds(80,380,200,70);
40.       a.add(jlMensaje);
41.
42.
43.       jbEnviar.addActionListener(new ActionListener() {
44.          @Override
45.          public void actionPerformed(ActionEvent e) {
46.             String nombres = jtfNombres.getText();
47.             String apellidos = jtfApellidos.getText();
48.             String direccion = jtfDireccion.getText();
49.             String mensaje = "";
50.             if(nombres.equals("")){
51.                mensaje += "Debe ingresar un nombre<br></br>";
52.             }
53.
54.             if(apellidos.equals("")){
55.                mensaje += "Debe ingresar un apellido<br></br>";
56.             }
57.
58.             if(direccion.equals("")){
59.                mensaje += "Debe ingresar un direccion<br></br>";
60.             }
61.
62.             if(!mensaje.equals("")){
63.                jlMensaje.setText("<html><body>" + mensaje + "</b
   ody></html>");
64.                return;
65.             }
66.
67.             Persona persona = new Persona(nombres, apellidos, direc
   cion);
68.             System.out.println(persona);
69.          }
70.       });
71.
72.       a.setSize(340,500);
73.       a.setLayout(null);
74.       a.setVisible(true);
75.    }
76.}
```

Analicemos la clase de la interfaz.

En primer lugar, en la línea 38, se ha añadido un nuevo label para añadir el mensaje para las validaciones de los campos. Este nuevo label se ha ubicado justo debajo del botón "Enviar".

Posteriormente, se añade el evento para capturar el click del botón, de esta forma en la línea 45 vamos a poner las validaciones que nos requiere el problema. Primero obtenemos el valor de cada uno de los campos y lo guardamos en variables de tipo String.

Ahora, por cada uno de los campos validamos con un "if" si es que están vacíos o no. Si están vacíos, entonces concatenamos la variable mensaje con el texto literal de cada "if".

Tengamos en cuenta que el operador += nos permite acumular un valor en una variable. En este caso estamos añadiendo al final de la variable "mensaje" cada una de las validaciones. Finalmente validamos si la variable mensaje se encuentra vacía o no. En el caso de que tenga contenido lo mostrará en el label de mensajes. El texto final que estamos introduciendo en el JLabel es un código HTML.

Hay que recordar que todos los componentes de Java Swing soportan el código HTML. Tampoco debemos olvidar que estamos agregando un "return ;" dentro del "if" del mensaje. Esto significa que si existen mensajes que mostrar el flujo del método terminará en la línea 64 y ya no continuará con las siguientes líneas de código.

Por último, en la línea 67 estamos creando un objeto de tipo Persona con los datos que ingresamos por la interfaz y los mostramos en la salida del programa con el método System.out.println ya que sobrecargamos el método "toString" de la clase Persona.