```
1 /**
 2
   * ComplexDialoquePanel.java
 3
   * 
 4
   * Diálogo dinámico emergente con campos de texto.
 5
   * 
   * Radu Constantin Robu y Alberto Pérez
 6
 7
   */
 8 package P3. Vista;
9
10 import java.awt.GridBagConstraints;
11 import java.awt.GridBagLayout;
12 import java.awt.Insets;
13 import java.util.HashMap;
14 import java.util.Map;
15 import javax.swing.*;
16
17 /**
18
   * Clase que permite crear una ventana para introducir varios datos
   * por teclado.
19
20
   */
21 public class ComplexDialoguePanel extends JPanel {
22
       private static final String ETIQUETA_VENTANA = "Introducir información";
23
       private static final String ETIQUETA_CONTRASEÑA = "Contraseña";
24
       private String[] etiquetasDatos;
25
       private Map<String, JTextField> camposTexto = new HashMap<>();
26
27
28
        * Construye un ComplexDialogPanel.
29
30
        * @param textoVentana
31
        * @param etiquetasDatos
32
        * @param tamanyoCamposTexto
33
        */
34
       public ComplexDialoguePanel(String textoVentana, String[] etiquetasDatos,
35
                                    int tamanyoCamposTexto) {
           setLayout(new GridBagLayout());
36
37
           inicializar(etiquetasDatos, tamanyoCamposTexto);
38
           setBorder(BorderFactory.createTitledBorder(textoVentana));
39
           this.etiquetasDatos = etiquetasDatos;
40
       }
41
42
       /**
43
        * Inicializa el ComplexDialoguePanel.
44
45
        * @param etiquetasDatos
46
        * @param tamanyo
47
        */
48
       private void inicializar(String[] etiquetasDatos, int tamanyo) {
49
           for (int i = 0; i < etiquetasDatos.length; i++) {</pre>
               String etiqueta = etiquetasDatos[i];
50
51
               add(new JLabel(etiqueta), createGbc(0, i));
52
53
               if (etiqueta != ETIQUETA_CONTRASEÑA) {
54
                   JTextField textField = new JTextField(tamanyo);
55
                   camposTexto.put(etiqueta, textField);
```

```
56
                     add(textField, createGbc(1, i));
                } else if (etiqueta == ETIQUETA_CONTRASEÑA) {
 57
                     JPasswordField passwordField = new JPasswordField(tamanyo);
 58
 59
                     camposTexto.put(etiqueta, passwordField);
                     add(passwordField, createGbc(1, i));
 60
                }
 61
 62
            }
 63
        }
 64
 65
        /**
         * Devuelve el camposTexto introducido en el campo correspondiente
 66
 67
         * a la etiqueta.
 68
 69
         * @param etiqueta
 70
         * @return
 71
         */
 72
        public String getText(String etiqueta) {
 73
            JTextField campoTexto = camposTexto.get(etiqueta);
            if (campoTexto != null) {
 74
 75
                return campoTexto.getText();
 76
            } else {
 77
                throw new IllegalArgumentException(etiqueta);
            }
 78
 79
        }
 80
 81
        /**
 82
         * Crea las dimendiones del GridBagLayout.
 83
         *
 84
         * @param X
 85
         * @param y
 86
         * @return
 87
         */
 88
        public static GridBagConstraints createGbc(int x, int y) {
 89
            GridBagConstraints gbc = new GridBagConstraints();
 90
            qbc.qridx = x;
 91
            qbc.qridy = y;
 92
            gbc.weightx = 1.0;
            gbc.weighty = gbc.weightx;
 93
 94
            if (x == 0) {
 95
                gbc.anchor = GridBagConstraints.LINE_START;
 96
                qbc.fill = GridBaqConstraints.BOTH;
 97
                gbc.insets = new Insets(5, 5, 5, 8);
            } else {
 98
 99
                gbc.anchor = GridBagConstraints.LINE_END;
100
                gbc.fill = GridBagConstraints.HORIZONTAL;
                gbc.insets = new Insets(5, 5, 5, 5);
101
102
            }
103
            return gbc;
104
        }
105
106
        /**
107
         * Obtiene los datos del viajero a través de un ComplexDialoguePanel.
108
109
         * @param opciones
110
         * @return
```

```
111
         */
112
        public String[] obtenerTextoCampos(String[] opciones) {
113
            int respuesta = -1;
114
            int numDatos = this.etiquetasDatos.length;
115
            boolean datosIncorrectos = true;
116
            while (datosIncorrectos) {
117
                String[] datos = new String[numDatos];
118
                int tipoOpcion = JOptionPane.DEFAULT_OPTION;
119
120
                int tipoMensaje = JOptionPane.PLAIN_MESSAGE;
                Icon icono = null;
121
122
                Object valorInicial = opciones[0];
123
124
                respuesta = JOptionPane.showOptionDialog(null,
125
                         this, ETIQUETA_VENTANA,
126
                         tipoOpcion, tipoMensaje, icono,
127
                         opciones, valorInicial);
128
129
                if (respuesta == 0) {
                     for (int i = 0; i < numDatos; i++) {</pre>
130
                         datos[i] = this.getText(this.etiquetasDatos[i]);
131
132
                     }
133
134
                     datosIncorrectos = comprobarDatosIntroducidos(datos);
135
                     if (!datosIncorrectos) {
136
137
                         return datos;
                     }
138
                } else {
139
140
                     return null;
                }
141
142
143
            return null;
144
        }
145
146
147
         * Método que devuelve si los datos introducidos son correctos o no.
148
149
         * @param datos
150
         * @return
151
         */
152
        private boolean comprobarDatosIntroducidos(String[] datos) {
153
            int datosValidos = 0;
            boolean datosIncorrectos = true;
154
155
            for (String dato : datos) {
156
157
                if ((dato != null) && (dato.length() > 0)) {
158
                     datosValidos++;
159
                }
            }
160
161
162
            if (datosValidos == datos.length) {
163
                datosIncorrectos = false;
            }
164
165
```

