

Práctica 23: Eventos inmobiliaria

Diseña un interfaz gráfico para la aplicación que hiciste en la práctica de la inmobiliaria sobre colecciones. El aspecto debe ser lo más parecido posible al siguiente:

Agencia Inmobiliaria

Datos generales

Código: Tipo propiedad: Precio: Superficie:

Dirección:

Descripción:

Datos específicos para vivienda

Tipo de vivienda: Número dormitorios: Número baños:

Datos específicos para fincas rústicas

Tipo terreno:

Suministro eléctrico: ☒ Si ☐ No

Suministro de agua: ☒ Si ☐ No

Dispones de vivienda: ☒ Si ☐ No

Operaciones disponibles

Nota: Inicialmente los componentes con información sobre el tipo de propiedad “Vivienda”, deben estar deshabilitados, ya que en la lista con los tipos de propiedad, la que aparece por defecto es “Finca Rústica”, por tanto el aspecto inicial será como el de la figura anterior.

Módulo Vista Controlador

Desarrollar el ejercicio siguiendo el modelo vista controlador, en este caso el modelo será lo que ya desarrollasteis en la práctica de la inmobiliaria.

Para tener disponible el modelo en el **controlador**, tendréis que definir, y pedir memoria en el constructor, una variable de tipo Inmobiliaria, algo así:

```
/**
 * Variables de instancia
 */
private Vista miVista;
private Inmobiliaria miInmobiliaria;

/**
 * Constructor
 */
public Controlador(Vista v) {
    this.miVista = v;
    this.miInmobiliaria = new Inmobiliaria();
}
```

Parte gráfica (Vista)

No creo que encontréis ningún problema en el diseño de la parte gráfica, como gestor de esquemas el que os resultará más cómodo es BorderLayout. Como estáis observando los cuatro paneles tienen borde, yo he optado por hacer un método para esto, así solo tengo que llamarlo para que establezca el borde, y me ahorro tener que hacerlo por cada uno de los paneles:

```
/**
 * Método que coloca un borde compuesto a un panel
 * - Un borde vacío
 * - Y un borde con línea y texto
 *
 * @param p --> Panel para ponerle el borde
 * @param textoBorde --> Texto que vamos a poner en la leyenda del borde
 */
private void estableceBorde(JPanel p, String textoBorde) {
    p.setBorder(new CompoundBorder(
        new EmptyBorder(5,5,5,5),
        new TitledBorder(new LineBorder(Color.gray,1,true),textoBorde))
    );
}
```

Por otro lado, tenemos enumeraciones, para poder volcar el contenido de una enumeración en un JComboBox, tendréis que parametrizarlas a la enumeración que queráis volcar en ella. Por ejemplo:

```
private JComboBox<EnumTipoPropiedad> tipoPropiedad;
```

Para poder añadir los valores de la enumeración al JComboBox, podéis hacerlo de una de estas dos formas:

- A la hora de crearlo:

```
tipoPropiedad = new JComboBox<EnumTipoPropiedad>(EnumTipoPropiedad.values());
```

- Una vez que ya lo tenemos creado, con la ayuda del método setModel, que me permite poder añadir opciones a un JComboBox, durante la ejecución:

```
tipoPropiedad = new JComboBox<EnumTipoPropiedad>();
tipoPropiedad.setModel(
    new DefaultComboBoxModel<EnumTipoPropiedad>(
        EnumTipoPropiedad.values()));
```

El funcionamiento de esta aplicación será el siguiente (Controlador)

➤ Según el tipo de propiedad que se seleccione en la **lista desplegable “Tipo propiedad”**, deben habilitarse o deshabilitarse componentes en la parte derecha de la ventana, siguiendo las siguientes reglas:

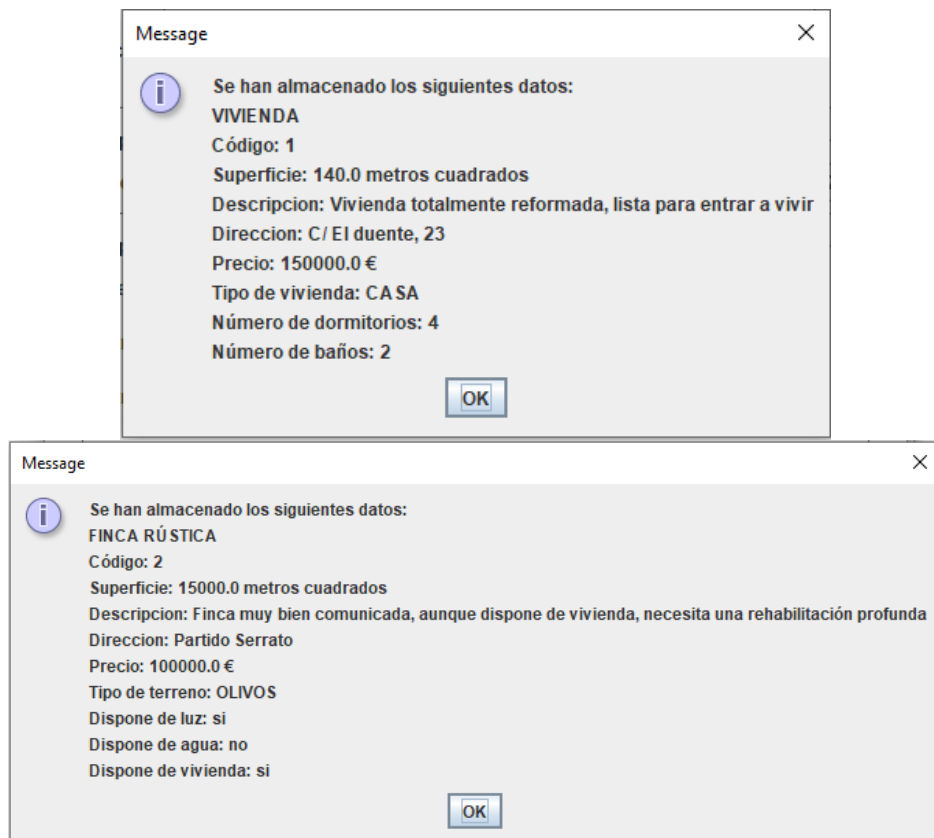
- Los campos que recogen “Datos generales”, siempre estarán habilitados.
- Si se selecciona “Local comercial”, “Plaza de aparcamiento” o “Solar” deben deshabilitarse todos los campos que recogen información específica sobre “Viviendas” y sobre “Fincas Rústicas”.
- Si se selecciona “Vivienda”, se deben habilitar todos los campos que recogen información específica sobre “Viviendas”, y deshabilitar los que recogen información sobre “Fincas rústicas”.
- Si se selecciona “Finca rústica”, se deben habilitar todos los componentes que recogen información específica sobre “Fincas rústicas”, y deshabilitar los que recogen información sobre “Viviendas”.

Pista para programar esta parte:

Mi consejo es que desarrolléis un par de métodos para activar/desactivar los componentes de los paneles de la vivienda y la finca rústica, ya que son acciones que vais a tener que realizar varias veces.

```
/**
 * Deshabilita/habilita los componentes de "Vivienda"
 * @param estado --> true (habilita) o false (deshabilita)
 */
private void estadoVivienda(boolean estado) {
    miVista.getTipoVivienda().setEnabled(estado);
    miVista.getNumDor().setEnabled(estado);
    miVista.getNumBaños().setEnabled(estado);
}
```

- En cuanto a los **botones** su comportamiento será el siguiente:
 - Botón **"Guardar propiedad"**: al pulsar este botón se deben recoger los datos del formulario y añadir una nueva propiedad a la lista de propiedades que tiene la inmobiliaria. También se debe mostrar una ventana emergente con toda la información que se ha recopilado (método toString de los distintos tipos de propiedades), el aspecto de esta ventana puede ser como las siguientes imágenes:



Si no podemos almacenar la propiedad, también se indicará con un mensaje informativo.

Pista para programar esta parte:

- ✓ Como ya estaréis suponiendo tenéis que sacar todos los datos del formulario, y además pueden ser objetos tipo Propiedad, tipo Vivienda o tipo Finca Rústica. Nos va a venir muy bien el polimorfismo, ya que os aconsejo que la variable donde vais a recoger los datos la defináis tipo Propiedad, y cuando solicitéis memoria para el objeto ya concretáis si es una Propiedad, una Vivienda o una FincaRustica.

Propiedad nuevaPropiedad;

Y posteriormente si se trata de una vivienda, tendréis algo parecido a esto

```
nuevaPropiedad =  
    new Vivienda(cod, tipoPropiedad, pre, dir, superf, des, tVivEnum, nDor, nBaños);
```

o bien, si por ejemplo es un local, algo así;

```
nuevaPropiedad =  
    new Propiedad(cod, tipoPropiedad, pre, dir, superf, des);
```

Así a la hora de añadir la propiedad a la inmobiliaria, tendréis un único tipo de objeto.

- ✓ Para sacar los datos de los JComboBox, y convertirlos en enumeraciones, por ejemplo, la enumeración del tipo de propiedad, tendréis que hacer algo parecido a esto:

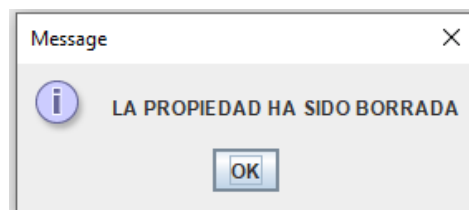
```
EnumTipoPropiedad tipoPropiedad =  
    (EnumTipoPropiedad)miVista.getTipoPropiedad().getSelectedItem();
```

- ✓ Os recomiendo sacar por un lado todos los datos que son comunes (código, tipo propiedad, precio, ...), y ya en función del tipo de propiedad, sacar los específicos de vivienda o de finca rústica si es el caso.

- ✓ Una vez que tengáis el objeto nuevaPropiedad creado, el añadirlo a la inmobiliaria será tan simple como el siguiente código:

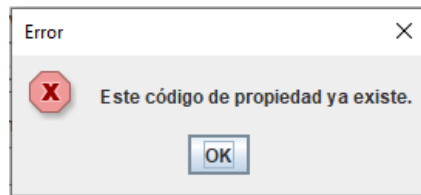
```
// Añadir la nueva propiedad a la Inmobiliaria  
if (miInmobiliaria.añade(nuevaPropiedad)) {  
    JOptionPane.showMessageDialog(miVista,  
        "Se han almacenado los siguientes datos: \n" +nuevaPropiedad.toString());  
    limpiar(); // Añadir el formulario  
}  
else {  
    JOptionPane.showMessageDialog(miVista,  
        "No se ha podido almacenar la propiedad.",  
        "Error",JOptionPane.ERROR_MESSAGE);  
}
```

- Botón **"Borrar propiedad"**: al pulsar este botón se borrará de la colección la propiedad que en ese momento se está visualizando, y si no se producido ningún error nos devolverá un mensaje indicándolo. Os recuerdo que en la clase Inmobiliaria ya tenáis un método que os permitía borrar por código.

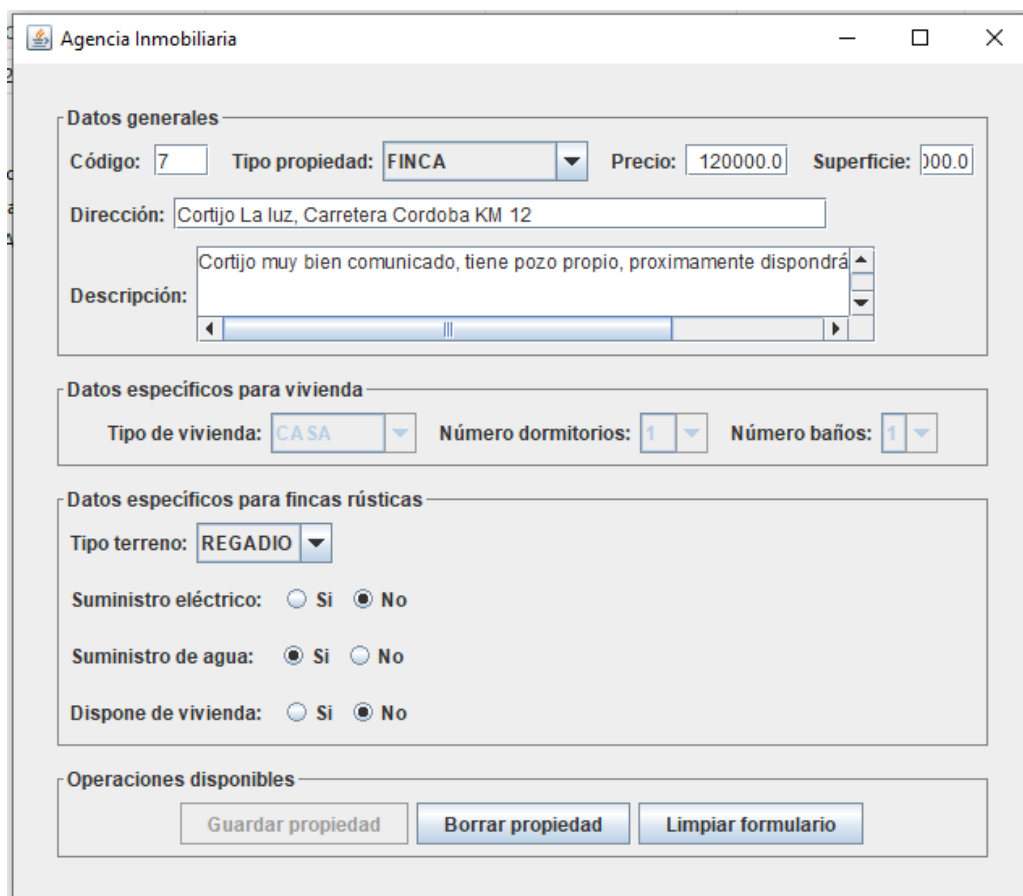


- Botón **"Limpiar Formulario"**: al pulsar este botón el formulario debe quedar como se muestra en la imagen primera de esta práctica, es decir, con todos los campos en blanco, las listas desplegables con la primera opción seleccionada por defecto, excepto, en el caso de Tipo de Propiedad, que deberá estar seleccionada la opción de "Finca", y los radio en el valor de "Sí".

- **Tras introducir el código de la propiedad**, se debe comprobar si ya existe otra propiedad con ese código previamente almacenada, en caso de que así sea se mostrarán todos los datos de esa propiedad y se deshabilitará el botón “Guardar Propiedad”, también se mostrará un mensaje de error similar al de la imagen.



Es decir, mostrará los datos como si fuese una “consulta”, pero sólo nos dejará poder ver esos datos, aunque los modifiquemos no se guardarán los cambios, lo que si podremos hacer es borrar la propiedad que estamos visualizando. En la siguiente imagen se muestran los datos de la propiedad de código 7, que previamente había sido guardada.



The screenshot shows a window titled "Agencia Inmobiliaria" with the following sections:

- Datos generales:**
 - Código: 7
 - Tipo propiedad: FINCA
 - Precio: 120000.0
 - Superficie: 000.0
 - Dirección: Cortijo La luz, Carretera Cordoba KM 12
 - Descripción: Cortijo muy bien comunicado, tiene pozo propio, proxicamente dispondrá
- Datos específicos para vivienda:**
 - Tipo de vivienda: CASA
 - Número dormitorios: 1
 - Número baños: 1
- Datos específicos para fincas rústicas:**
 - Tipo terreno: REGADIO
 - Suministro eléctrico: ☐ Si ☒ No
 - Suministro de agua: ☒ Si ☐ No
 - Dispone de vivienda: ☐ Si ☒ No
- Operaciones disponibles:**
 - Guardar propiedad (disabled)
 - Borrar propiedad
 - Limpiar formulario

Pista para programar esta parte:

- ✓ De nuevo, al igual que ocurrió cuando se guardaba una propiedad, deberías definir la variable tipo Propiedad, pero os vais a encontrar con el problema de no poder acceder a los campos específicos de la propiedades tipo Vivienda o tipo FincaRustica, en estos casos podéis “jugar” con los casting, ya que si sabéis que la propiedad es una vivienda, por ejemplo, podréis hacer algo parecido a esto:

```
miVista.getTipoVivienda().setSelectedItem(((Vivienda) p).getTipoVivienda());  
// Convierto mediante un casting el objeto propiedad "p", en objeto Vivienda, y  
// así podré acceder a sus métodos propios, por ejemplo, en este caso al que  
// devuelve el tipo de vivienda.
```

- ✓ Tenéis que controlar si la propiedad ya existe de varias formas, si se pulsa el intro (dejar esto para el final, porque os va dar dolor de cabeza, os saltará más de un evento a la vez, y lo tendréis que controlar mediante algún boolean), o bien si el campo código pierde el foco (os vais a otro campo con el ratón o pulsáis el tabulador).

La forma de controlar si se ha pulsado el intro es:

```
public void keyPressed(KeyEvent arg0) {  
    if (arg0.getKeyCode() == KeyEvent.VK_ENTER) {  
        System.out.println("Estoy en keyPressed");  
        mostrar();  
    }  
}
```

La forma de controlar el foco, imagino que lo tenéis más claro:

```
public void focusLost(FocusEvent e) {  
    System.out.println("Estoy en foco perdido");  
    mostrar();  
}
```

Pero insisto en que esto os va a costar más trabajo, porque es posible que algunas veces os salte más de un evento a la vez, y se ejecute lo mismo dos veces. Si no os sale, dejarlo solo con el método keyPressed, para el intro, o el focusLost, por la pérdida de foco.

Resumiendo, los eventos que deberéis programar serán:

- Evento **ItemEvent** para el JComboBox de los tipos de Propiedad (Interfaz ItemListener)
- Evento **ActionEvent** para los tres botones (Intefaz ActionListener)
- Eventos **KeyEvent** (por si se pulsa intro en el campo de texto del código), o **FocusEvent** (se pierde el foco en el campo de texto del código). El intro en el campo de texto también se podía haber programado con el evento ActionEvent (Interfaces FocusListener y KeyListener)

Controlad las excepciones que consideréis necesarias, como por ejemplo, que los datos introducidos no se correspondan con los tipos de datos.