PRÁCTICA 3

Swing

Gestión de eventos

El objetivo de esta práctica es gestionar eventos usando el entorno NetBeans y las herramientas visuales que ofrece. Los elementos que veremos se asumen conocidos por el estudiante, siendo el objetivo de esta sesión repasarlos usando el NetBeans. Para más detalles, se recomienda la lectura del tutorial "*Creating a GUI with JFC/Swing*", concretamente la sección "*Writing Event Listeners*".

Primeros ejemplos

En primer lugar, creamos un proyecto en NetBeans de tipo "Aplicación Java" y añadimos una ventana (`JFrame`) usando la plantilla por defecto de NetBeans. En la ventana, incorporamos un botón (véase Figura 1).

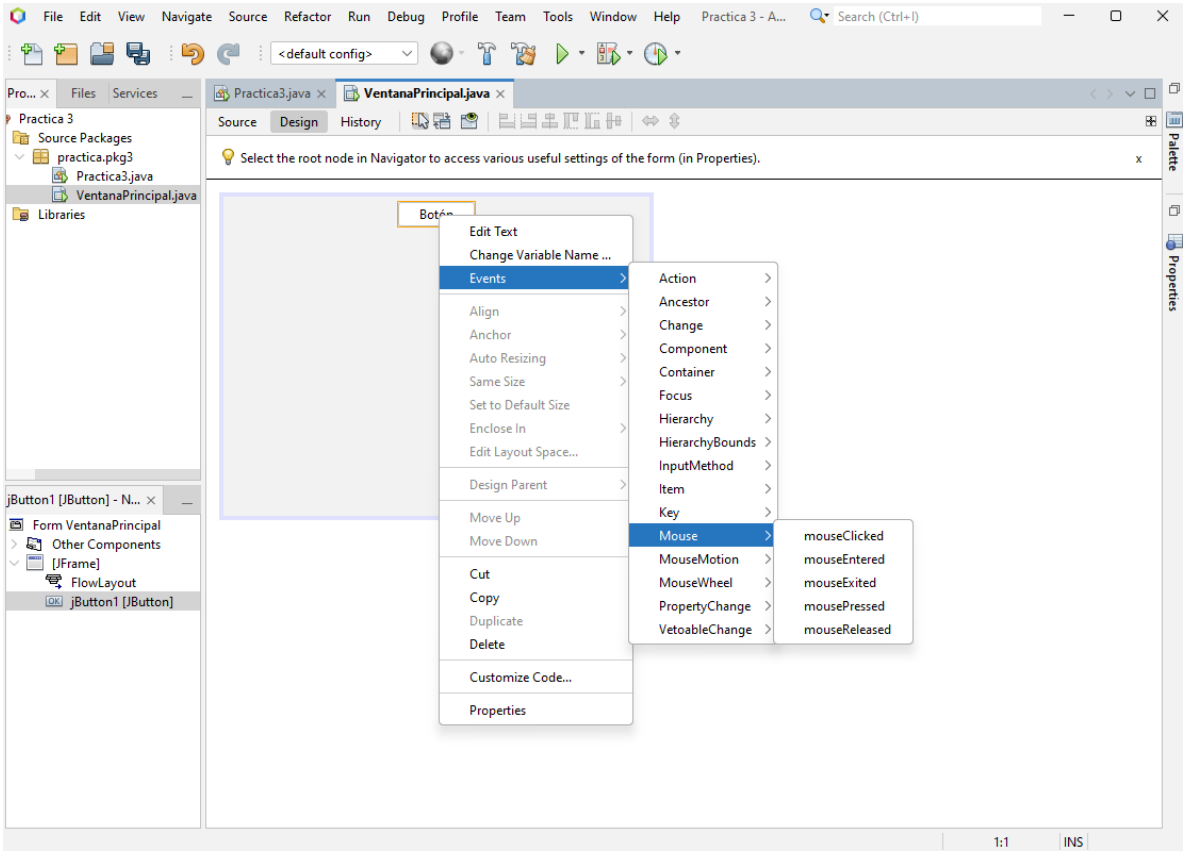

Figura 1: Área de diseño en NetBeans. Gestión de eventos

Existen varios estilos a la hora de gestionar eventos (mediante clases anónimas, clases internas, etc.). Por defecto, NetBeans usa clases anónimas, pero para este ejercicio usaremos el estilo basado en clases internas. Para ello, en el navegador de la izquierda (parte inferior) seleccionamos el ítem raíz asociado a la ventana principal (en la Figura 1, el ítem "*Form VentanaPrincipal*") y, en las propiedades, en el apartado "*Listener Generation Style*", seleccionamos la opción "*One inner class*".

En primer lugar, gestionaremos eventos de ratón generados por el botón que acabamos de incorporar. Pulsando el botón derecho, el menú “Eventos” nos muestra la lista de eventos que puede lanzar el generador (en este caso, el botón). Para este primer ejemplo, manejaremos los siguientes eventos:

- mouseClicked (al pulsar el botón, hacer que éste se ponga de color rojo)
- mouseEntered (al entrar en el botón, hacer que éste se ponga de color azul)
- mouseExited (al salir en el botón, hacer que éste se ponga de color verde)

Si observamos el código generado, vemos que se ha creado una nueva clase (interna) que implementa el interfaz `MouseListener` y que, consecuentemente, incorpora los cinco métodos incluidos en dicho interfaz:

```
private class FormListener implements MouseListener {
   FormListener() {}

   public void mouseClicked(java.awt.event.MouseEvent evt) {
     if (evt.getSource() == boton) {
       VentanaPrincipal.this.botonMouseClicked(evt);
     }
   }

   public void mouseEntered(java.awt.event.MouseEvent evt) {
     if (evt.getSource() == boton) {
       VentanaPrincipal.this.botonMouseEntered(evt);
     }
   }

   public void mouseExited(java.awt.event.MouseEvent evt) {
     if (evt.getSource() == boton) {
       VentanaPrincipal.this.botonMouseExited(evt);
     }
   }

   public void mousePressed(java.awt.event.MouseEvent evt){}

   public void mouseReleased(java.awt.event.MouseEvent evt){}
}
```

Para los eventos gestionados, el código anterior llama a otros métodos (que es donde nosotros ponemos el código, esto es, donde el NetBeans sitúa el cursor de forma automática):

```
private void botonMouseClicked(java.awt.event.MouseEvent evt) {
   boton.setBackground(Color.red);
}

private void botonMouseEntered(java.awt.event.MouseEvent evt) {
   boton.setBackground(Color.blue);
}

private void botonMouseExited(java.awt.event.MouseEvent evt) {
   boton.setBackground(Color.green);
}
```

Además del código anterior, en el método `initComponents` se añade el siguiente código:

```
FormListener formListener = new FormListener();

boton.addMouseListener(formListener);
```

A continuación, gestionar los siguientes eventos y analizar el código generado en cada caso (se aconseja ver el código cada vez que se incorpora un nuevo evento):

- Sobre el botón que actualmente tenemos en la ventana:
  - *mouseDragged*
- Sobre la ventana principal (`JFrame`)
  - *mouseClicked*
- Incorporar un nuevo botón y manejar los siguientes eventos
  - *mouseClicked*
  - *actionPerformed*

Pregunta: Analizando el código generado, ¿cuántas clases manejadoras crea NetBeans? En caso de un mismo tipo de evento, ¿cómo gestiona el hecho de que sea un generador u otro?

Ejercicio 1: Dibujar puntos y líneas

En este segundo ejemplo incorporaremos a nuestra aplicación la posibilidad de[1]:

- Pintar puntos en la zona de la ventana donde se haga un clic
- Dibujar líneas (viéndose la línea mientras se hace el *dragged*)

Para ello, el estudiante deberá elegir (1) qué eventos manejar y (2) qué hacer en cada caso.

A la hora de dibujar, recordar que el método que establece qué se visualiza en una ventana es el método `paint(Graphics)` que se hereda de `JFrame`; si queremos modificarlo para, como en este caso, pintar sobre la ventana, habrá que sobrecargarlo en nuestra `VentanaPrincipal`:

```
public void paint(Graphics g){
   super.paint(g);

   // Código (mensajes a g)

}
```

[11] Sólo se verá un punto cada vez, es decir, al pintar un nuevo punto desaparecerá el punto anterior. Lo mismo ocurrirá en el caso de la línea.

Ejercicio 2: Escritorio SM

En este último ejercicio crearemos un entorno multiventana como el que se muestra en la Figura 2 (i.e., la ventana principal tendrá un escritorio con ventanas internas). Para ello, crearemos un nuevo proyecto en NetBeans de tipo "Aplicación Java", al que llamaremos "EscritorioSM", y añadiremos una ventana (`JFrame`) usando la plantilla por defecto de NetBeans. Incorporaremos el menú "Archivo" con la opción "Nuevo", de forma que cuando se seleccione esta opción se creará una nueva ventana interna.

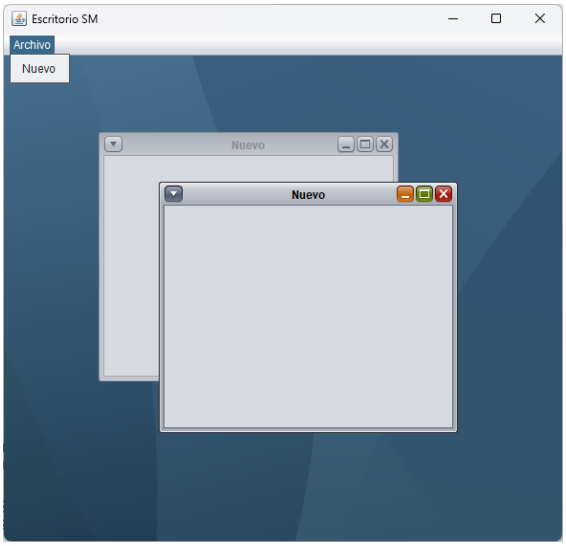

Figura 2: Entorno multiventana

Para realizar un entorno multiventana:

- Añadir un escritorio en el centro de la ventana principal. Para ello, incorporar un `JDesktopPane` usando el NetBeans (área "contenedores swing").

- Crear una clase propia "VentanaInterna" que herede de `JInternalFrame` (para ello, usar NetBeans). Activar las propiedades "*closable*", "*iconifiable*", "*maximizable*" y "*resizable*".

- Para incorporar una ventana interna, hay que (i) crear el objeto, (ii) añadirlo al escritorio y (iii) mandar el mensaje para visualizarla. Así, por ejemplo, en el manejador del evento asociado a la opción "Nuevo", tendríamos:

```
private void menuNuevoActionPerformed(ActionEvent evt) {
  VentanaInterna vi = new VentanaInterna();
  escritorio.add(vi);
  vi.setVisible(true);
}
```