TAP Labo 23/11/2016

Para completar las posibilidades de confección de GUIs vamos a ver cómo pintar sobre un componente.

El `Canvas` (lienzo) es un componente de `awt` con la representación gráfica vacía en el que podemos "pintar" lo que queramos. En la versión posterior para generación de GUIs (`Swing`) no hay un componente semejante, por lo que podríamos seguir usando el `Canvas`, pero es mejor no hacerlo porque los componentes `awt` presentan ciertas desventajas frente a `Swing`. Lo que podemos hacer es fabricar nuestro propio `JComponent` con una clase que lo extiende.

Por otro lado, es cierto que hoy en día `JavaFX` nos permite hacer cosas infinitamente más complejas que lo que podemos hacer en un `Canvas` o un `JComponent` con un esfuerzo razonable, pero esa es otra historia…

También relacionado con lo que vamos a ver estaría lo que se refiere a `Java2D` o incluso `Java3D`, pero vamos a comenzar por lo básico.

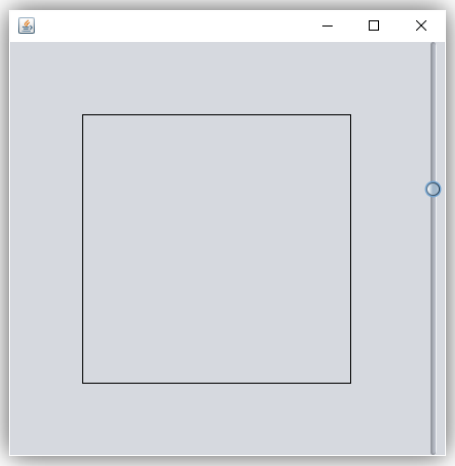
Todos los componentes gráficos se muestran en el interfaz como resultado de la ejecución del método `paint(.)` que heredan de `Component` y reescriben. Así que todo lo que tenemos que hacer es que nuestro `JComponent` sepa "autodibujarse" para lo que reescribimos el método `paint(.)`.

El prototipo de la función es el siguiente: `public void paint(Graphics g)`

El objeto de clase `Graphics` es el que proporciona una serie de métodos que permiten dibujar diversas figuras geométricas, textos, incrustar imágenes, etc. (revisar la documentación).

Lo único que haremos inicialmente será dibujar un rectángulo centrado en el área del `JComponent` con un alto y ancho de la mitad de sus dimensiones.

Para ver cómo nos queda lo meteremos en un JFrame (restablezcamos el BorderLayout que Netbeans cambia). No podemos poner gráficamente nuestro `JComponent` en el centro porque no estará en la paleta de componentes, por lo que podemos hacer dos cosas: forzar a que aparezca en la paleta de componentes, o ponerlo a mano. Para lo primero tenemos una opción en el menú contextual que podemos sacar sobre el `JComponent` en la ventana Projects (seguir `tools->add to palette`). Para lo segundo bastará con generar el `JComponent` en el constructor y añadirlo al `JFrame` (antes de la llamada a initComponents que pone Netbeans). Cada estrategia tiene sus ventajas e inconvenientes.



Deberemos dar al `JFrame` o a nuestro `JComponent` una dimensión mínima razonable porque de no ser así aparecería a cero (sólo veríamos la barra del `JFrame`). Razonablemente podemos pensar que nuestro `JComponent` no ha de forzar esto, y que queda a cargo del `JFrame`.

Veamos ahora como podemos hacer esto un poco "interactivo":

Añadamos al `JFrame` un `JSlider` en el este y diremos que nuestro `JComponent` es un `ChangeListener` de modo que podamos "decirle" al `JSlider` que nuestro `JComponent` se suscribe a sus cambios (lo hacemos en el mismo constructor del `JFrame`). Solo nos queda hacer que nuestro `paint(.)` no sea fijo sino que dependa de un parámetro y ese parámetro sea cambiado por la rutina de atención a los cambios del `JSlider` convenientemente.

(nota sobre la complejidad de la práctica: la solución final con Netbeans puede conseguirse con el retoque de una línea de código, la inserción de otras 7, el uso de la opción de menú "insert code" para reescribir el **`paint(.)`**, y aceptando la sugerencia "implement all abstract methods" para cumplir con el Listener -no en este orden-)