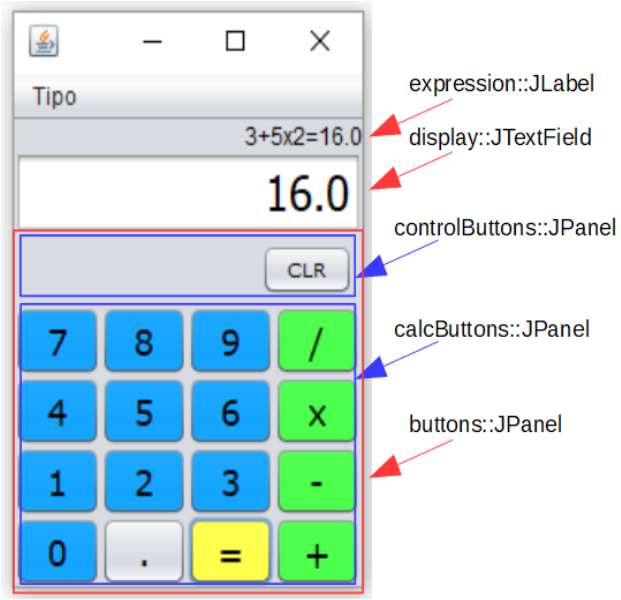
TAP 2018-19. Laboratorio.

1. Calculadora

Vamos a fabricar una calculadora partiendo de un GUI que ya está hecho. Este tiene varios elementos a los que atenderemos con código (hay uno, el menú, al que no atenderemos en esta práctica)

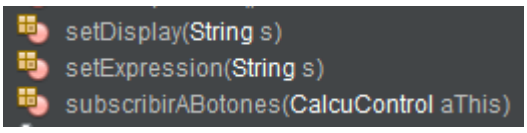


- Un panel conteniendo a otros dos, donde el primero es un teclado para operaciones de "control" -sólo con la tecla de borrado total de la operación en curso-, y el segundo un teclado con los 10 digitos, el punto, las cuatro operaciones básicas y el "igual".
- Un display donde se mostrarán las cifras según se tecleen los digitos y el punto, así como el resultado de las operaciones que se ejecutarán cuando se pulsan los operadores o el "igual".
- Una etiqueta en la que se mostrará toda la secuencia de teclas ejecutada, y el resultado al terminarla con la pulsación de la tecla "igual"[1]

Este GUI (**`CalcuGUI.java`**) se encuentra en un proyecto Netbeans que es preciso descargar de la red. Además del GUI, encontraremos también una clase de arranque (**`Calculadora`**) que lo instancia[2], y a continuación instancia un objeto de una clase de control (**`CalcuControl`**) para "dar vida" a nuestra calculadora. Esta última clase está inicialmente vacía, y su implementación es buena parte del ejercicio a realizar (no todo).

Acciones a realizar:

1. Declarar **`CalcuControl`** como un **`ActionListener`** (para poder atender a todas las pulsaciones de teclas en el GUI).
2. Hacer que pueda instanciarse recibiendo una referencia al GUI que controlará, y subscribiéndose a los eventos que atenderá de él (todas las teclas). En lugar de suscribirse a cada botón, se seguirá el enfoque planteado en el siguiente punto.
3. Retocar **`CalcuGUI`** añadiendo un método (**`subscribirABotones`**) que permita a un **`ActionListener`** suscribirse a todos sus botones. No se hará suscribiendo uno a uno a cada botón, sino de un modo más eficaz: recorriendo los paneles para asignarles a todos los botones que incluyen el **`ActionListener`** recibido (ver "pistas" al final).
4. En el método de implementación del interfaz, escribir el código para atender a cada evento, indagando "quién" es quien ha enviado el "aviso" y actuando en consecuencia (punto 6), lo que acabará mostrando un cambio en el GUI conforme al siguiente punto.
5. Para alterar el contenido del display y de la expresión en el GUI, éste proporcionará sendos métodos (que simulan ser "setters")



---

1 Obsérvese en la imagen que la expresión no es demasiado "razonable": un resultado de 13.0 parecería lo correcto. Lo que la calculadora de la imagen hace es un procesamiento de izquierda a derecha sin atender a una precedencia de operadores. Se ha hecho esto por sencillez, ya que atender a precedencias podría complicarse en exceso: veríamos de inmediato la necesidad de disponer de paréntesis, puesto que atender a una precedencia pide de forma natural atender a la posibilidad de forzarla (no hacerlo es tan "insatisfactorio" como no atender a precedencia).

2 Comentaremos la clase de arranque en el aula.

6. La clase **`CalcuControl`** incluirá todo lo que se considere necesario para mantener y actualizar el estado del display y la expresión del GUI como resultado de cada interacción con el teclado (p.ej. Para las teclas numéricas simplemente añadir el dígito a la representación; o para los operadores ejercutar y mostrar el resultado). Esto no es complicado pero tampoco es trivial: no se debe admitir cualquier número en cualquier momento; no puede haber más de un punto; cuando se pulsa un operador aún falta un operando para poder ejecutarlo...

Algunas "pistas":

"**`set Display`**" y "**`setExpression`**" son muy fáciles, no necesitan pistas...

"**`suscribirABotones`**" debe recorrer todos los paneles que contengan botones para que a todos ellos se les suscriba el objeto que le llega como parámetro.

`jp.getComponents()` al JPanel (**jp**) podemos pedirle los componentes que contiene
`b.addActionListener(al)` a un botón (**b**) podemos suscribirle un ActionListener (**al**)

Ojo a los detalles: No tenemos una colección de paneles con botones para hacer un **`for`** (pero podemos meterlos en un array); los botones los obtenemos como "**`Component`**", luego hay que asegurarle al compilador que son botones (será necesario usar un cast); ...

Al definir en **`CalcuControl`** el método que atiende a los eventos tendremos que "descubrir" cuál ha sido el botón pulsado, cosa que podemos hacer "mirando dentro" del **`ActionEvent`** que recibimos... podremos descubrir quién ha sido el componente gráfico origen ("source") -que será un botón-, y a este pedirle el texto que muestra en el GUI, con lo cual decidir a qué hacer.

Para entregar:

1) El proyecto de la práctica

2) Nuestra calculadora está formada por la combinación de dos elementos: el GUI y el controlador. Podríamos generar en el "main" más de una calculadora sin más que combinar varios pares de objetos de cada clase. ¿Podríamos hacerlo con un sólo controlador común a todos los GUIs? Si la respuesta es "no", tiene sentido plantearse hacer las modificaciones necesarias para que la respuesta sea "si", y en ese caso ¿sería razonable hacerlo?

2) Independientemente de que dispongamos de un controlador que soporte varios GUIs o no, ¿Tiene sentido planterse un controlador "universal" que pueda soportar cualquier GUI? ¿Qué problemas surgirían? ¿Tienen solución o "apaño"?