TAP 2024-25. Laboratorio.

1. Calculadora

Muchos habéis presentado un interfaz de calculadora como primer ejercicio evaluable, ahora se trata de que dicho interfaz sirva realmente para disponer de una calculadora operativa. Quienes no tengan un interfaz propio pueden trabajar con este que se corresponde con el ejemplo que vamos a explicar más adelante.

Antes de ir con el ejemplo, algunas consideraciones sobre cómo plantear la solución:

- En cuanto a los "listeners": una solución con la ayuda del IDE pasa por asociar un `ActionListener` a cada botón y escribir cada uno de los métodos correspondientes, pero esto da lugar a código innecesariamente extenso y repetitivo. En el extremo opuesto podemos escribir un único `ActionListener` que atienda a todos los botones, y que comience por descubrir qué botón es el que se ha pulsado (una pista en el ejemplo posterior). Como aproximación intermedia podemos usar diferentes `ActionListener` para diferentes funciones, como por ejemplo uno para formar números, que atiende a todas las teclas numéricas y a la del punto, u otro para las operaciones aritméticas que atiende a las teclas +, -, X, /.
- En cuanto a qué hacer una vez pulsada una tecla podemos optar por varios enfoques:
    - El trivial, que actúa tras cada pulsación con un código muy específico, de modo que mientras sean dígitos o el punto se acumulan para formar un número decimal, y tras cada tecla de operación va resolviendo (ojo porque esto va con un decalaje, ya que sólo se puede resolver un operador cuando se dispone de los dos operandos)
    - El "correcto", que se ocupa de aceptar sólo las pulsaciones admisibles (no es admisible p.ej. 3.24.513, ni 3x6+-4), y si se admite la pulsación somete a la expresión completa a evaluación mediante el método óptimo, que pasa por convertirla a la Notación Polaca Inversa (RPN) y resolver esta, para lo que se usan estructuras de pila (podéis usar esta aproximación buscando en la literatura informática o ayudándoos de alguna IA)
    - El "eficaz", que es un enfoque similar al anterior, pero delegando la resolución de la expresión en algún software ya disponible. Una manera, entre otras, de resolver esto con Java es delegar en una "máquina de ejecución de scripts" (`ScriptEngine`). Podríamos usar diversas `ScriptEngine`, pero la primera opción será una máquina que procese JavaScript. Esto se hace mediante un gestor de máquinas de scripts del siguiente modo[1]:

      ```
      ScriptEngine jsEngine=new ScriptEngineManager().getEngineByName("javascript");
      ```

      Con esto podremos evaluar expresiones mediante `jsEngine.eval(expresion)`.

En los dos últimos enfoques podremos también someter a evaluación a expresiones incorrectas y actuar en consecuencia cuando el retorno indique que no son evaluables (tema pendiente gestión de errores).

Vamos ahora con un ejemplo que plantea un único `Listener` y el enfoque trivial usando el interfaz aportado en el zip.

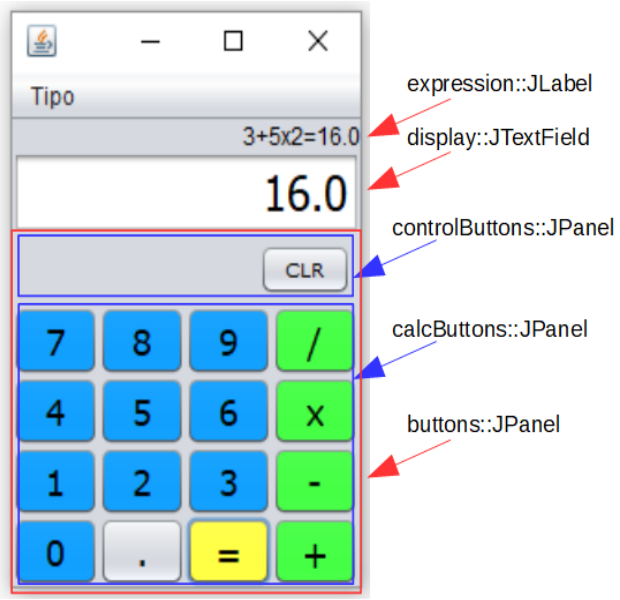
Tomaremos como ejemplo la calculadora simple que encontramos en el Sistema Operativo Windows. El aspecto del GUI es el de la figura. Llamaremos a esta clase "CalcuGUI", y tendrá los siguientes elementos (entre otros):

- 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"[2]

---

[1] En la versión 8 de JDK la máquina de JavaScript está incluida. Esta fue retirada en alguna versión posterior, de modo que a partir de entonces es necesario añadirla como librería (se denomina Nashorn)

2 Obsérvese en la imagen que, como hablamos de una calculadora extremadamente simple, 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 ninguna 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).



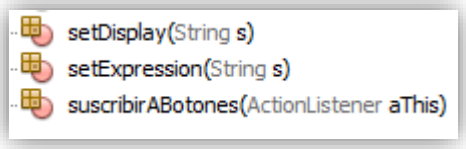
- Un display donde se mostrarán las cifras según se tecleen los dígitos y el punto, así como el resultado de las operaciones que se ejecutarán cuando se pulsan los operadores o el "igual".

- 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 dígitos, el punto, las cuatro operaciones básicas y el "igual".

Utilizaremos la ayuda gráfica de Netbeans en todo menos en el montaje del panel que hemos llamado "buttons", que puede ser más "cómodo" hacerlo de modo "manual" (escribiendo código).

Una vez construido el GUI, para hacer que nuestra calculadora "funcione", haremos lo siguiente:

1. Definir una clase **`CalcuControl`** como un **`ActionListener`** para poder generar un objeto que atienda a todas las pulsaciones de teclas en el GUI (no pondremos un Listener para cada botón, sino que atenderemos a todos con el mismo -mirando en el ActionEvent sabremos quién está "avisando").
2. Definir una clase de arranque que se limite a generar un objeto CalcuGUI y un objeto CalcuControl. Este último deberá poder recibir una referencia al GUI para, a través de ella, suscribirse a los botones.
3. Retocar **`CalcuGUI`** añadiendo un método (**`subscribirABotones`**) que permita a un **`ActionListener`** suscribirse a todos sus botones (ver la imagen junto al punto 5). 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 **`CalcuControl`**, definir el método de implementación del interfaz escribiendo 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")
6. La clase **`CalcuControl`** incluirá todo lo que se considere necesario para mantener el estado de la calculadora y representarlo en el GUI (el display y la expresión) 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 ejecutar 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 podremos decidir qué hacer.