PROYECTO CACULADORA V1

by

Jonathan Carrero

Abstract. — El objetivo de este proyecto es terminar de comprender qué es *Swing* y cómo funciona la creación de ventanas básicas haciendo uso de esta librería. Para ello, se propone crear una aplicación GUI que permita hacer calculos sencillos. El proyecto tendrá varias versiones que habrá que ir implementando.

1. Funcionamiento

La aplicación lanza una ventana (ver demostración) que permite realizar operaciones sencillas, es decir, una calculadora de toda la vida. La calculadora debe permitir realizar operaciones con números negativos, pero no es necesario implementar decimales. Por esta razón, la operación de división ignorará el resto y consistirá en hacer divisiones enteras.

2. Detalles de implementación

El proyecto estará formado por dos clases: *Main* y *Engine*. La primera de ellas tan sólo se encarga de instanciar un objetivo de la clase *Engine*. Por otro lado, vamos a ver qué partes tiene la clase *Engine*. Lo primero que tenemos que tener en cuenta son los atributos de la clase. Este es el listado de atributos de la clase Engine, en el cual se debe respetar el nombre de los atributos. Cabe destacar que estos son los atributos mínimos que deberías tener, pero no quita que puedas añadir más (por ejemplo, para el tema de colores o cualquier diseño adicional que consideres).

```
1  // Marco de la ventana
2  private JFrame frame;
3  // Panel general que ocupa toda la ventana
4  private JPanel contentPanel;
5  // Panel norte que contiene el display
6  private JPanel displayPanel;
7  // Panel sur que contiene los botones
8  private JPanel buttonPanel;
9  // Display
10 private JTextField display;
11 // Botones
12 private JButton n0;
13 private JButton n1;
14 private JButton n2;
15 private JButton n3;
16 private JButton n4;
17 private JButton n5;
18 private JButton n6;
19 private JButton n7;
20 private JButton n8;
21 private JButton n9;
22 private JButton divide;
23 private JButton multiply;
24 private JButton subtract;
25 private JButton add;
26 private JButton equal;
27 private JButton reset;
28 // Tipos de boton
29 private enum ButtonType {REGULAR, OPERATOR}
30 // Almacenar temporalmente ciertos valores
31 private int num1, num2, result;
32 private char operation;
```

Si nos fijamos, la aplicación tiene un `JFrame` principal. Dentro de este, hay dos paneles. El primero de ellos, llamado `contentPanel`, mostrará el display al igual que ocurre en una calculadora común. Y el segundo, llamado `displayPanel`, mostrará los botones de la aplicación. También tenemos un atributo de tipo `JTextField`, el cual estará a cargo de mostrar las operaciones o números que vayan apareciendo en el display. A continuación, tenemos los atributos para instancias los botones y, un poco más abajo, un enumerado. Como se puede apreciar en el ejemplo, existen dos tipos de botones según su aspecto visual (los colores que estos tienen, vaya). Estas características visuales serán establecidas según el tipo enumerado de cada botón. Por último, tenemos algunos atributos que serán necesarios para almacenar valores temporales a medida que realizamos operaciones con la aplicación. Vamos a echar un vistazo a la constructora y los métodos.

- `Engine()`: La constructora debe instanciar, entre otras cosas, los atributos que se han mencionado anteriormente. Si añades funcionalidad adicional (ya sean acciones o simplemente algunos elementos de diseño), también debería ser instanciado en la constructora. Al final de la constructora debe aparecer una llamada a `setSettings()` y `addActionEvent()`.
- `setSettings()`: Este método establece la configuración principal de todos los componentes visuales de la ventana. Concretamente, se encarga de, entre otras cosas: poner los layouts de los paneles y añadirlos, añadir los botones y llamar al método `setFeaturesButton()`, el display, establecer las características de los botones, tamaño de la ventana, localización, etc.
- `setFeaturesButton(JButton _button, ButtonType _type)`: Contiene una condición que permite distinguir si el tipo de botón pasa como parámetro es de tipo `REGULAR` u `OPERATOR`. En función de esto, pintará el botón de un color u otro. Puedes añadirle (y es algo que se tendrá en cuenta) más características tales como cambio del tipo de letra, bordes, etc.
  - `_button` identifica el botón sobre el que se van a cambiar las características.
  - `_type` identifica de qué tipo es el botón sobre el que se van a cambiar las características.
- `addActionEvent()`: Este método registra los `ActionListener` para todos los botones de la aplicación. Es decir, para cada botón, añade un `ActionListener` que recibe como parámetro el objeto `this` para poder identificar el objeto (botón) que se pulsa.
- `operation()`: Comprueba qué operación se debe realizar. En otras palabras: mira el estado actual del atributo `this.operation` y, en función de ese valor, lleva a cabo una operación u otra (con los atributos `this.num1` y `this.num2`, que representan los dos únicos operando que maneja nuestra calculadora), modificando el tributo `this.result` y actualizando el texto en el display.
- `actionPerformed(ActionEvent e)`: Este método se encarga de obtener la información que haya en el display (números introducidos y operación que se debe realizar) y llamar al método `operation()` para ejecutar dicha operación. Hay muchas formas de llevar a cabo esta lógica... Piensa cuál podría ser la mejor manera de hacerlo. Por ejemplo, una manera podría ser identificar el botón que se ha pulsado y añadir su texto al display y, cuando se pulse sobre el botón `=`, entonces hacemos que se ejecute la operación que se haya indicado con los botones de operación (sumar, restar, multiplicar o dividir). En cualquier caso e independientemente de la decisión que se tome, lo primero que debemos hacer en este método es recoger el tipo de botón que se ha pulsado (para poder hacer la distinción) y el texto asociado a ese botón. Esto se puede hacer de la siguiente forma:

```
33 // Recogemos el tipo de boton que se ha pulsado y su texto
34 Object source = e.getSource();
35 String input_text = e.getActionCommand();
```

Para no complicarte, puedes asumir que, tanto cuando se inicia la calculadora como cuando se pulsa sobre el botón `R`, en el display no hay nada (está vacío). Además, cuando se pulsa el botón `R` debes resetear los dos operandos (ponerlos a 0).

Nota: como he dicho, hay muchas maneras de implementar la lógica del método `actionPerformed()`. Te aconsejo que busques más información sobre las expresiones regulares (para saber más sobre expresiones regulares, ver [**4, 1, 3, 2**]), ya que estas pueden ser realmente útiles para coger las partes del texto del display que te interesen en cada momento, así como hacer un `split()` de dicho texto e identificar los distintos elementos (que en nuestro caso, al ser una calculadora que únicamente trabaja con dos operandos, los tres elementos presentes son: operando 1, símbolo de la operación y operando 2).

References

[1] N. R. Calle, *Expresiones regulares en Java*. `https://refactorizando.com/expresiones-regulares-java`. [Online; accessed 31-October-2023].

[2] Gskinner, *Herramienta para comprobar expresiones regulares*. `https://regexr.com`. [Online; accessed 31-October-2023].

[3] Puntocomnoesunlenguaje, *Ejemplos de expresiones regulares en Java*. `https://puntocomnoesunlenguaje.blogspot.com/2013/07/ejemplos-expresiones-regulares-java-split.html`. [Online; accessed 31-October-2023].

[4] Wikipedia, *Expresión regular*. `https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular`. [Online; accessed 31-October-2023].

December 11, 2024

J.C., Colegio Litterator • *E-mail :* `jonathan.carrero@colegiolitterator.com`