



Programación con Java

TAREA 19

Índice.....	1
Introducción.....	2
Swing y Awt.....	3
Ejercicio 01:.....	3
Ejercicio 02:.....	4
Ejercicio 03:.....	5
Ejercicio 04:.....	6
Imágenes complementarias.....	10
Webgrafía.....	17

En esta práctica manejaremos las librerías de “*javax.swing.*;*” y “*java.awt.*;*” para hacer programas más visuales con *Swing* y añadir “acciones” con *AWT*. Como los ejercicios son sencillos y ya había tocado e investigado estas librerías me he propuesto llevarlas a un nivel más de dificultad donde a todos les añadimos colores, fuentes al texto y, a excepción del último, imágenes. Al último programa, individual, hemos hecho que las preguntas sean generativas, hasta que no responda una no tendrá opción a ver las demás

Ejercicio 01:

En el primer ejercicio tendremos que crear una ventana que nos pedirá introducir un nombre para saludarnos en una ventana emergente al pulsar un botón. He puesto un texto que aparecerá como recomendación al no estar encima y una imagen junto al saludo de bienvenida.

MouseListener para la desaparición del texto

```
JTextField nombreUsuario = new JTextField(text:"Perro Sánchez Kahbron", columns:20);
nombreUsuario.setForeground(Color.GRAY);
nombreUsuario.setAlignmentX(Component.CENTER_ALIGNMENT);
nombreUsuario.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        if (nombreUsuario.getText().equals(anObject:"Perro Sánchez Kahbron")) {
            nombreUsuario.setText(t:"");
            nombreUsuario.setForeground(Color.BLACK);
        }
    }

    @Override
    public void mouseExited(MouseEvent e) {
        if (nombreUsuario.getText().isEmpty()) {
            nombreUsuario.setText(t:"Perro Sánchez Kahbron");
            nombreUsuario.setForeground(Color.GRAY);
        }
    }
});
```

ActionListener del botón

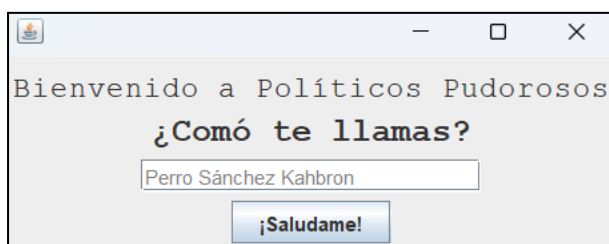
```
Button saludar = new JButton(text:"¡Saludame!");
saludar.setAlignmentX(Component.CENTER_ALIGNMENT);
saludar.setSize(width:30, height:120);
saludar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Image perro = new ImageIcon(filename:"imagenes/perroSanche.png").getImage();
        perro = perro.getScaledInstance(width:250, height:250, Image.SCALE_SMOOTH);
        ImageIcon perroSanche = new ImageIcon(perro);

        JPanel panelSaludo = new JPanel();
        panelSaludo.setLayout(new BoxLayout(panelSaludo, BoxLayout.Y_AXIS));
        JLabel bienvenida = new JLabel(text:"Bienvenido");
        JLabel nombre = new JLabel(nombreUsuario.getText());
        JLabel imagen = new JLabel(perroSanche);
        bienvenida.setAlignmentX(Component.CENTER_ALIGNMENT);
        nombre.setAlignmentX(Component.CENTER_ALIGNMENT);
        imagen.setAlignmentX(Component.CENTER_ALIGNMENT);
        bienvenida.setFont(new Font(Font.MONOSPACED, Font.PLAIN, size:22));
        nombre.setFont(new Font(Font.MONOSPACED, Font.BOLD, size:18));

        panelSaludo.add(bienvenida);
        panelSaludo.add(nombre);
        panelSaludo.add(imagen);

        JOptionPane.showOptionDialog(parentComponent:null, new JScrollPane
        (panelSaludo), title:"Saludo", JOptionPane.DEFAULT_OPTION, JOptionPane.
        PLAIN_MESSAGE, icon:null, new Object[{}], initialValue:null);
    }
});
```

Ventana principal



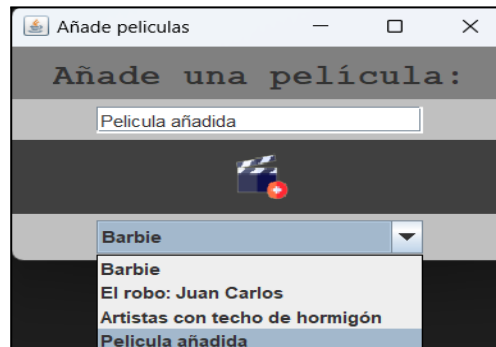
Ventana saludo



Ejercicio 02:

Para el segundo ejercicio tuvimos que hacer una aplicación para insertar un nombre de una película y luego al pulsar en el botón tendrá que añadirse al JComboBox donde se mostrarán todas las películas.

Ventana



MouseListener para la desaparición del texto

```
JPanel panelEntrada = new JPanel();
panelEntrada.setBackground(Color.LIGHT_GRAY);
JTextField peliculaEntrada = new JTextField(text:"Avatar 2", columns:20);
peliculaEntrada.setForeground(Color.GRAY);
peliculaEntrada.setAlignmentX(Component.CENTER_ALIGNMENT);
peliculaEntrada.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        if (peliculaEntrada.getText().equals(anObject:"Avatar 2")) {
            peliculaEntrada.setText("");
            peliculaEntrada.setForeground(Color.BLACK);
        }
    }

    @Override
    public void mouseExited(MouseEvent e) {
        if (peliculaEntrada.getText().isEmpty()) {
            peliculaEntrada.setText("Avatar 2");
            peliculaEntrada.setForeground(Color.GRAY);
        }
    }
});
panelEntrada.add(peliculaEntrada);
panel.add(panelEntrada);
```

ActionListener del botón

```
JPanel separador = new JPanel();
separador.setBackground(Color.DARK_GRAY);
JButton boton = new JButton(agregar);
boton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String peli = peliculaEntrada.getText();
        if (!peliculas.contains(peli)) {
            peliculas.add(peli);
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    generarInferior(panelPelis, peliculas);
                }
            });
        }
    }
});
```

Ejercicio 03:

En el tercer ejercicio tenemos que crear una encuesta donde nos harán tres preguntas: nuestro SO favorito, especialidades y horas en el ordenador; y un botón al final para mostrar las opciones seleccionadas en una nueva ventana. Para ello he hecho que las preguntas vayan apareciendo según respondemos la pregunta anterior.

Esperar respuesta para continuar

```
JPanel panelSO = sistemasOperativos();
panelPrincipal.add(panelSO);

frame.add(panelPrincipal);

frame.pack();
frame.setLocationRelativeTo(c:null);
frame.setVisible(b:true);

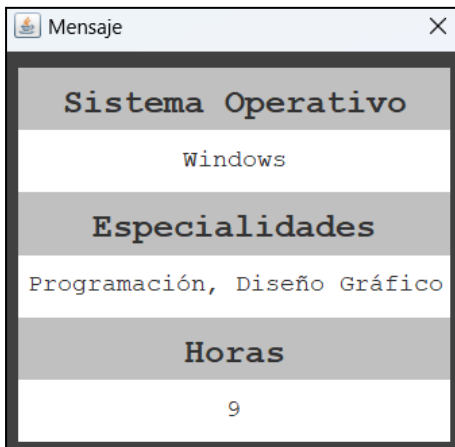
boolean parar = true;
while (parar) {
    for (Component paneles : panelSO.getComponents()) {
        if (paneles instanceof JPanel) {
            JPanel panelBoton = (JPanel) paneles;
            for (Component boton : panelBoton.getComponents()) {
                if (boton instanceof JRadioButton) {
                    JRadioButton radBut = (JRadioButton) boton;
                    if (radBut.isSelected()) {
                        parar = false;
                    }
                }
            }
        }
    }
}
```

ActionListener del botón

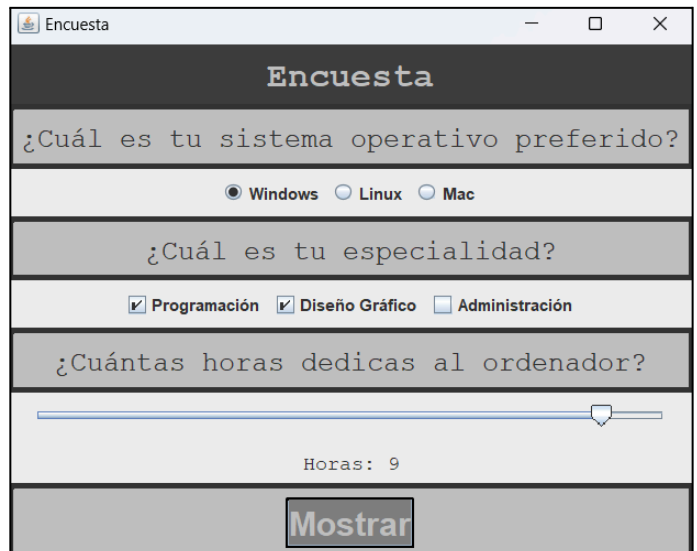
```
mostrar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String sisOp = obtenerSO(panelSO);
        String especialidades = obtenerEspecialidades(panelEsp);
        String horas = obtenerHoras(panelHoras);

        panelMensaje(frame, panelPrincipal, sisOp, especialidades, horas);
    }
});
```

Ventana mensaje a



Ventana principal



Ejercicio 04:

Para el último ejercicio tendremos que crear una calculadora básica que tenga las operaciones básicas con “[javax.swing.*](#)”.

Hemos comenzado creando un frame con un tamaño establecido y en él hemos añadido una barra de navegación “[JMenuBar barraDeNavegacion](#)” y un panel principal en el que añadiremos la [pantalla de cálculo](#) y el [panel de los botones](#).

Para el control de los botones he decidido iterar sobre los componentes del panel y obtener el botón pulsado, y una vez aquí llamar a la función “[funcionBoton\(\)](#)” para actuar en función al botón pulsado.

Frame y variable definidas al principio

```
public class ejercicio04 {
    Run | Debug
    public static void main(String[] args) {
        int numeros[] = {7,8,9,4,5,6,1,2,3,0};
        String operadores[] = {"/","*","-","+", "C", "="};
        StringBuilder historialTxt = new StringBuilder();
        StringBuilder operacion = new StringBuilder();

        // VENTANA //
        JFrame frame = new JFrame(title:"¡MAS! - Calculadora");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(width:350, height:500);

        // BARRA DE NAVEGACIÓN //
        JMenuBar barraDeNavegacion = new JMenuBar();
        JMenu opciones = new JMenu(s:"Opciones");
```

Panel principal y pantalla

```
JPanel panelDePaneles = new JPanel();
panelDePaneles.setLayout(new BorderLayout(panelDePaneles, BorderLayout.Y_AXIS));
panelDePaneles.setBorder(new EmptyBorder(top:10, left:20, bottom:10, right:20));

// PANTALLA DE CÁLCULO //
JTextPane pantalla = new JTextPane();
pantalla.setBorder(new LineBorder(new Color(r:160, g:80, b:190), thickness:4));
pantalla.setMaximumSize(new Dimension(frame.getWidth()-10, height:300));
pantalla.setFont(new Font(name:"arial", Font.BOLD, size:32));
pantalla.setEditable(b:false);
pantalla.setText(t:"0");
panelDePaneles.add(pantalla);
// PANTALLA DE CÁLCULO //
```

Panel y funciones de los botones

```
// PANEL BOTONES NÚMEROS //
JPanel panelDeBotones = new JPanel();
crearPanelBotones(panelDeBotones, numeros, operadores);
panelDePaneles.add(panelDeBotones);
// PANEL BOTONES NÚMEROS //

// FUNCIONES BOTONES //
for (Component componente : panelDeBotones.getComponents()) {
    if (componente instanceof JPanel) {
        JPanel panelBoton = (JPanel) componente;
        Component[] botones_array = panelBoton.getComponents();
        for (Component componente_boton : botones_array) {
            if (componente_boton instanceof JButton) {
                JButton boton = (JButton) componente_boton;
                boton.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        String entrada = boton.getText();
                        funcionBoton(panelDeBotones, entrada, pantalla,
                                    operacion, historialTxt, numeros, operadores);
                    }
                });
            }
        }
    }
}
// FUNCIONES BOTONES //
```

Dicha función, llamará a otra para saber el tipo de botón que se ha pulsado [“queBotonEs\(String\)”](#), y en un `switch` haremos las acciones.

[A la hora de las capturas he separado la función en los “case”, pero esta de principio a fin en orden]

Método “funcionBoton()”

case “numero”

Verificará si contiene coma y un decimal, valor obtenido en [“cantidadDecimales\(String\)”](#), deshabilitará los números para limitarlo a dos decimales; y en caso contrario (tendrá coma pero no decimal) lo concatenará, habilitará los operadores y deshabilitaremos la coma (porque dentro de “operadores” está también la coma).

En caso contrario, se habilitarán los operadores, por la posibilidad de que no lo estén.

[“estadoBotones\(JPanel, StringBuilder, String, String, int\[\], String\[\]\)”](#)

```
public static void funcionBoton(JPanel panelDeBotones, String entrada, boolean entreParentesis, boolean hayAntes, JTex
int numeros[], String operadores[]) {
    boolean pintar = false;
    String partes[] = separarOperacion(operacion);
    switch (queBotonEs(entrada)) {
        case "numero":
            int entradaEntero = Integer.parseInt(entrada);
            if (partes[partes.length - 1].contains(s:".")) {
                if (cantidadDecimales(partes[partes.length - 1]) == 1) {
                    estadoBotones(panelDeBotones, operacion, estado:"deshabilitado", tipo:"numeros", numeros, operadores);
                }
                operacion.append(entradaEntero);
                pintar = true;
                estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"operadores", numeros, operadores);
                RoundedButton coma = buscarBoton(panelDeBotones, busqueda:"," );
                coma.setEnabled(b:false);
            } else {
                operacion.append(entradaEntero);
                pintar = true;
                estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"operadores", numeros, operadores);
            }
        }
        break;
    }
```

case “cancelar”

Pondrá el `StringBuilder` con longitud “0”, por lo que elimina todo su contenido; habilitará todos los botones, por la posibilidad de que no lo estén; y pintará un “0” en la pantalla, con [“repintarPantalla\(JTextField, String\)”](#)

El método: [“habilitarTodo\(JPanel, StringBuilder, int\[\], String\[\]\)”](#); utilizará [“estadoBotones”](#).

```
case "cancelar":
    operacion.setLength(newLength:0);
    habilitarTodo(panelDeBotones, operacion, numeros, operadores);
    repintarPantalla(pantalla, infoPantalla:"0");
    pintar = false;
    break;
```

case “igual”

Verificará si el array `String[] partes` tiene una longitud igual a tres para poder operar, ya que estaría el primer número, el operador y el segundo número; solo si se cumple esta condición utilizará [“separarYOperar\(StringBuilder\)”](#) para hacer la operación y mostrarlo, guardándolo para mostrar en [“panelHistorial”](#) y vaciando “operacion”.

```
case "igual":
    if (partes.length == 3) {
        String resultado = separarYOperar(operacion);
        pintar = false;
        repintarPantalla(pantalla, resultado);
        habilitarTodo(panelDeBotones, operacion, numeros, operadores);
        historialTxt.append(operacion.toString() + " = " + resultado + "%");
        operacion.setLength(newLength:0);
    }
    break;
```


case “coma”

Verificará el estado de la operación para decidir qué pintar en la pantalla. En el caso de que la longitud de “operacion” será de 0, pero ya tengamos historial recogeremos el resultado de la operación anterior, con [“ultimoResultado\(StringBuilder\)”](#) en caso de poder ponerla; si por el contrario no hay historial, o la longitud de “partes” es de 2 (el último es un operador) pondremos un “0.”, si no se cumple nada de lo anterior se concatenará una coma y se deshabilitarán los operadores, con [“estadoBotones”](#)

```
case "coma":
if (operacion.length() == 0 && historialTxt.length() != 0 && !ultimoResultado(historialTxt).contains(s:",")) {
    String ultimoResultado = ultimoResultado(historialTxt);
    operacion.append(ultimoResultado + entrada);
} else if (operacion.length() == 0 || partes.length == 2) {
    operacion.append(str:"0.");
} else {
    operacion.append(str:"," );
}
estadoBotones(panelDeBotones, operacion, estado:"deshabilitado", tipo:"operadores", numeros, operadores);
pintar = true;
break;
```

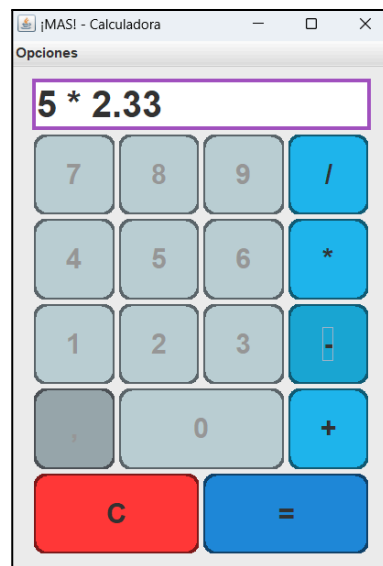
case “operador”

Verificará si la longitud de “partes” es igual a tres para operar con [“SepararYOperar”](#), añadir la operación resulta al historial, vaciará “operacion” y le añadirá el resultado de la operación y la entrada, el operador. Si “operacion” está vacío y “historialTxt” no lo está, se recogerá el último resultado de éste y se concatenar en “operacion” junto con la entrada. Si ninguna de las condiciones anteriores se cumplen, se concatenará la entrada a “operacion”. Por último, al salir de la condición, llamará al método [“estadoBotonesParaOperadores\(JPanel, StringBuilder, int\[\], String\[\]\)”](#).

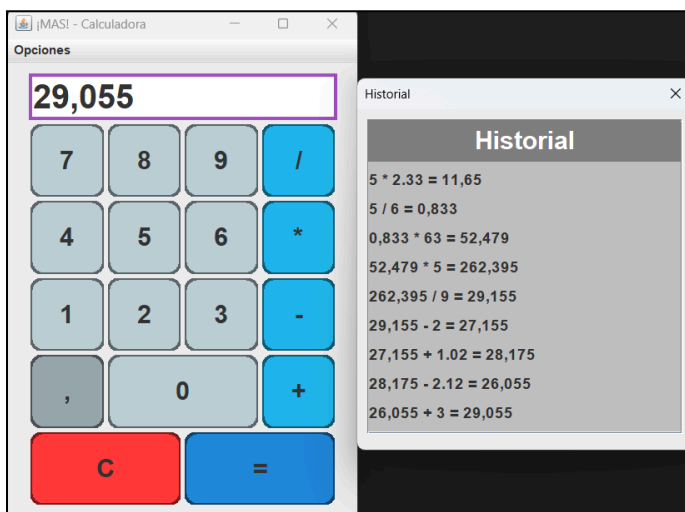
```
case "operador":
if (partes.length == 3) {
    String resultado = separarYOperar(operacion);
    historialTxt.append(operacion.toString() + " = " + resultado + "%");
    operacion.setLength(newLength:0);
    operacion.append(resultado + " " + entrada + " ");
} else if (operacion.length() == 0 && historialTxt.length() != 0) {
    String ultimoResultado = ultimoResultado(historialTxt);
    operacion.append(ultimoResultado + " " + entrada + " ");
} else if (operacion.length() != 0 && partes.length != 2) {
    operacion.append(" " + entrada + " ");
}
pintar = true;
estadoBotonesParaOperadores(panelDeBotones, operacion, numeros, operadores);
break;
}
if (pintar) {
    repintarPantalla(pantalla, operacion.toString());
}
}
```

Visualización final

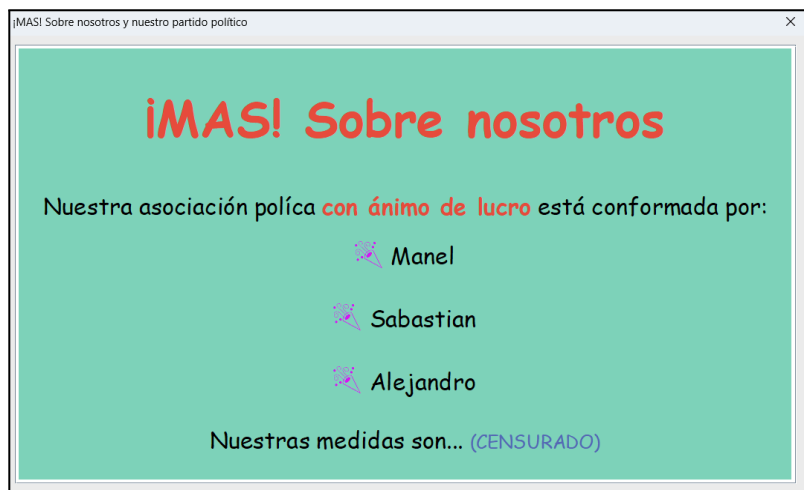
Con botones deshabilitados



Historial



Página "Sobre nosotros"



Imágenes complementarias

queBotonEs(String)

[“esEntero\(String\)”](#)

```
public static String queBotonEs(String entrada) {
    if (esEntero(entrada)) {
        return "numero";
    } else if (entrada.equals(anObject:"c")) {
        return "cancelar";
    } else if (entrada.equals(anObject:"=")) {
        return "igual";
    } else if (entrada.equals(anObject:",")) {
        return "coma";
    } else {
        return "operador";
    }
}
```

cantidadDecimales(String)

```
public static int cantidadDecimales(String numeroDecimal) {
    String partesNumero[] = numeroDecimal.split(regex:"\\.");
    if (partesNumero.length > 1) {
        return partesNumero[1].length();
    } else {
        return 0;
    }
}
```

habilitarTodo(JPanel, StringBuilder, int[], String[])

[“estadoBotones\(JPanel, StringBuilder, String, String, int\[\], String\[\]\)”](#)

```
public static void habilitarTodo(JPanel panelDeBotones, StringBuilder operacion, int numeros[], String operadores[]) {
    estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"operadores", numeros, operadores);
    estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"numeros", numeros, operadores);
}
```

estadoBotones(JPanel, StringBuilder, String, String, int[], String[])

[“buscarBoton\(JPanel, StringBuilder\)”](#)

```
public static void estadoBotones(JPanel panelDeBotones, StringBuilder operacion, String estado, String[] operadores[]) {
    switch (tipo) {
        case "numeros":
            if (estado.equals(anObject:"habilitado")) {
                for (int numero : numeros) {
                    RoundedButton botonNum = buscarBoton(panelDeBotones, String.valueOf(numero));
                    botonNum.setEnabled(b:true);
                }
            } else if (estado.equals(anObject:"deshabilitado")) {
                for (int numero : numeros) {
                    RoundedButton botonNum = buscarBoton(panelDeBotones, String.valueOf(numero));
                    botonNum.setEnabled(b:false);
                }
            }
            break;

        case "operadores":
            if (estado.equals(anObject:"habilitado")) {
                for (String operador : operadores) {
                    if (!operador.equals(anObject:"c")) {
                        RoundedButton botonOp = buscarBoton(panelDeBotones, operador);
                        botonOp.setEnabled(b:true);
                    }
                }
            } else if (estado.equals(anObject:"deshabilitado")) {
                for (String operador : operadores) {
                    if (!operador.equals(anObject:"c")) {
                        RoundedButton botonOp = buscarBoton(panelDeBotones, operador);
                        botonOp.setEnabled(b:false);
                    }
                }
            }
            break;
    }
}
```

Imágenes complementarias

separarYOperar(StringBuilder)

[“separarOperacion\(StringBuilder\)”](#)
[“operar\(double, double, String\)”](#)
[“resultadoStringFormat\(double\)”](#)

```
public static String separarYOperar(StringBuilder operacion) {
    String partes[] = separarOperacion(operacion);
    String operador1 = partes[0].replace(target:",", replacement:".");
    double num1 = Double.parseDouble(operador1);
    String operador = partes[1];
    double num2 = Double.parseDouble(partes[2]);
    double resultado = operar(num1, num2, operador);
    String resultadoFormateado = resultadoStringFormat(resultado);

    return resultadoFormateado;
}
```

esEntero(String)

```
public static boolean esEntero(String entrada) {
    try {
        double numero = Double.parseDouble(entrada);
        if (numero % 1 == 0) {
            return true;
        } else {
            return false;
        }
    } catch (Exception e) {
        return false;
    }
}
```

ultimoResultado(StringBuilder)

```
public static String ultimoResultado(StringBuilder historialtxt) {
    String partes[] = historialtxt.toString().split(regex:"%");
    String resultado[] = partes[partes.length-1].split(regex:" = ");
    return resultado[resultado.length - 1];
}
```

CaracteristicasBotones(RoundedButton)

```
public static RoundedButton caracteristicasBotones(RoundedButton boton) {
    boton.setBorder(new EmptyBorder(top:2,left:2,bottom:2,right:2));
    boton.setFont(new Font(name:"Arial", Font.BOLD, size:24));
    return boton;
}
```

repintarPantalla(JTextField, String)

```
public static JTextPane repintarPantalla(JTextPane pantalla, String infoPantalla) {
    pantalla.setText(infoPantalla);
    pantalla.repaint();
    pantalla.revalidate();

    return pantalla;
}
```

Imágenes complementarias

operar(double, double, String)

```
public static double operar(double op1, double op2, String operador) {
    double result = 0;
    switch (operador) {
        case "+":
            result = (op1 + op2);
            break;

        case "-":
            result = (op1 - op2);
            break;

        case "*":
            result = (op1 * op2);
            break;

        case "/":
            result = (op1 / op2);
            break;

        default:
            System.out.println(x:"UPS...");
            break;
    }
    return result;
}
```

resultadoStringFormat(double)
"esEntero(String)"

```
public static String resultadoStringFormat(double resultado) {
    DecimalFormat formato = new DecimalFormat(pattern:"#.###");
    if (esEntero(String.valueOf(resultado))) {
        return String.valueOf((int) resultado);
    } else {
        return formato.format(resultado);
    }
}
```

separarOperacion(StringBuilder)

```
public static String[] separarOperacion(StringBuilder operacion) {
    String partes[] = operacion.toString().split(regex:" ");
    return partes;
}
```

buscarBoton(JPanel, String)

Etiqueta "bucleExterior" para romperlo desde el bucle interior con "break bucleExterior;"

```
public static RoundedButton buscarBoton(JPanel panelDeBotones, String busqueda) {
    RoundedButton boton = new RoundedButton(text:null);
    bucleExterior:
    for (Component componente : panelDeBotones.getComponents()) {
        if (componente instanceof JPanel) {
            JPanel panelBoton = (JPanel) componente;
            Component[] botones_array = panelBoton.getComponents();
            for (Component componente_boton : botones_array) {
                if (componente_boton instanceof RoundedButton) {
                    boton = (RoundedButton) componente_boton;
                    if (boton.getText().equals(busqueda)) {
                        break bucleExterior;
                    }
                }
            }
        }
    }
    return boton;
}
```

Imágenes complementarias

panelHistorial(JPanel, StringBuilder)

[“panelOperaciones\(JPanel, StringBuilder\)”](#)

```
public static void panelHistorial(JPanel panelHistorial, StringBuilder historialTxt) {
    panelHistorial.setLayout(new BorderLayout(panelHistorial, BorderLayout.Y_AXIS));
    panelHistorial.setPreferredSize(new Dimension(width:300, height:300));
    JPanel tituloHistorial = new JPanel();
    tituloHistorial.setBackground(Color.GRAY);
    JLabel tituloLabel = new JLabel(text:"Historial");
    tituloLabel.setForeground(Color.WHITE);
    tituloLabel.setFont(new Font(name:"Arial", Font.BOLD, size:25));
    tituloLabel.setHorizontalAlignment(SwingConstants.CENTER);
    tituloHistorial.add(tituloLabel);
    panelHistorial.add(tituloHistorial);
    JPanel operacionesPanel = new JPanel();
    operacionesPanel = panelOperaciones(operacionesPanel, historialTxt);
    panelHistorial.add(operacionesPanel);
}
```

panelOperaciones(JPanel, StringBuilder)

```
public static JPanel panelOperaciones(JPanel operacionesPanel, StringBuilder historialTxt) {
    operacionesPanel.setLayout(new GridBagLayout());
    operacionesPanel.setBackground(Color.LIGHT_GRAY);
    GridBagConstraints posicion = new GridBagConstraints();
    posicion.fill = GridBagConstraints.BOTH;
    posicion.weightx = 1;
    posicion.weighty = 1;
    String operaciones[] = historialTxt.toString().split(regex:"%");
    int pos = 0;
    for (String op : operaciones) {
        JLabel operacionLabel = new JLabel(op);
        operacionLabel.setFont(new Font(name:"Arial", Font.BOLD, size:15));
        operacionLabel.setHorizontalAlignment(SwingConstants.LEFT);
        posicion.gridx = pos;
        operacionesPanel.add(operacionLabel, posicion);
        pos++;
    }
    return operacionesPanel;
}
```

estadoBotonesParaOperadores(JPanel, StringBuilder, int[], String[])

```
public static void estadoBotonesParaOperadores(JPanel panelDeBotones, StringBuilder operacion, int numeros[], String operadores[]) {
    estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"numeros", numeros, operadores);
    estadoBotones(panelDeBotones, operacion, estado:"deshabilitado", tipo:"operadores", numeros, operadores);
    RoundedButton coma = buscarBoton(panelDeBotones, busqueda:","");
    RoundedButton igual = buscarBoton(panelDeBotones, busqueda:"=");
    coma.setEnabled(b:true);
    igual.setEnabled(b:true);
}
```

Imágenes complementarias

crearPanelBotones(JPanel, int[], String[])

```
public static void crearPanelBotones(JPanel panelDeBotones, int numeros[], String operadores[]) {
    panelDeBotones.setLayout(new GridBagLayout());
    GridBagConstraints carCelda = new GridBagConstraints();
    carCelda.fill = GridBagConstraints.BOTH;
    carCelda.weightx = 1;
    carCelda.weighty = 1;
    crearBotones(panelDeBotones, carCelda, numeros, operadores);
}
```

crearBotones(JPanel, GridBagConstraints int[], String[])

```
public static void crearBotones(JPanel panelDeBotones, GridBagConstraints carCelda, int numeros[], String operadores[]) {
    int num_cel = 0;
    int indice_op = 0;
    int indice_nu = 0;
    for (int j = 1; j <= 5; j++) {
        for (int i = 1; i <= 4; i++) {
            carCelda.gridwidth = 1;
            boolean doble = false;
            if (num_cel == 13 || num_cel == 15 || num_cel == 16) {
                carCelda.gridwidth = 2;
                doble = true;
            }
            JPanel panelBoton = new JPanel(new BorderLayout());
            panelBoton.setBorder(new EmptyBorder(top:2, left:2, bottom:2, right:2));
            RoundedButton boton = new RoundedButton(text:null);
            if (!(i%4 == 0) && !(num_cel == 12) && !(j == 5)) {
                boton = new RoundedButton(String.valueOf(numeros[indice_nu]), arco:25, tipo:"numeros");
                boton = caracteristicasBotones(boton);
                indice_nu++;
            } else {
                switch (num_cel) {
                    case 12:
                        boton = new RoundedButton(operadores[indice_op], arco:25, tipo:"coma");
                        break;
                    case 15:
                        boton = new RoundedButton(operadores[indice_op], arco:25, tipo:"cancelar");
                        break;
                    case 16:
                        boton = new RoundedButton(operadores[indice_op], arco:25, tipo:"resultado");
                        break;
                    default:
                        boton = new RoundedButton(operadores[indice_op], arco:25, tipo:"string");
                        break;
                }
                boton = caracteristicasBotones(boton);
                indice_op++;
            }
            panelBoton.add(boton, BorderLayout.CENTER);
            carCelda.gridx = i;
            carCelda.gridy = j;
            panelDeBotones.add(panelBoton, carCelda);
            if (doble) {
                i++;
            }
            num_cel++;
        }
    }
}
```

panelHtmlSobreNosotros(JEditorPane)

```
public static void panelHtmlSobreNosotros(JEditorPane panelHTML) {
    panelHTML.setContentType(type:"text/html"); // Establecer el tipo de contenido como HTML
    panelHTML.setEditable(b:false);

    String paginaSobreNosotros = "<html> <head> <title>MAS! Sobre nosotros</title> <style> body { background-color: #f4f4f9; color: #333333; font-family: 'Comic Sans MS'; font-size: 16px; margin: 0; padding: 20px; display: flex; justify-content: center; align-items: center; height: 100vh; } .container { background-color: #81d3bd; border-radius: 10px; box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); padding: 20px; max-width: 600px; text-align: center; } h1 { color: #e74c3c; font-size: 2.5em; margin-bottom: 20px; } p { font-size: 1.2em; line-height: 1.6; } .highlight { color: #e74c3c; font-weight: bold; } .censored { font-size: 0.8em; color: #556bb7; } .list-item { margin: 10px 0; } .emoji { color: #d718f8; font-size: 1.2em; margin-right: 5px; } </style> </head> <body> <div class=\"container\"> <h1>MAS! Sobre nosotros</h1> <p>Nuestra asociación polica <span class=\"highlight\">con ánimo de lucro</span> está conformada por:</p> <p class=\"list-item\"><span class=\"emoji\">👮 </span>Mamel</p> <p class=\"list-item\"><span class=\"emoji\">👮 </span>Sabastian</p> <p class=\"list-item\"><span class=\"emoji\">👮 </span>Alejandro</p> <p>Nuestras medidas son... <span class=\"censored\">(CENSURADO)</span></p> </div> </body> </html>";
    panelHTML.setText(paginaSobreNosotros);
}
```

Imágenes complementarias

JFrame frame

```
public static void main(String[] args) {
    int numeros[] = {7,8,9,4,5,6,1,2,3,0};
    String operadores[] = {"/", "*", "-", "+", "C", "="};
    StringBuilder historialTxt = new StringBuilder();
    StringBuilder operacion = new StringBuilder();

    // VENTANA //
    JFrame frame = new JFrame(title:"¡MAS! - Calculadora");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(width:350, height:500);
}
```

JMenuBar barraDeNavegacion

["panelHistorial\(JPanel, StringBuilder\)"](#)
["panelHtmlSobreNosotros\(JEditorPane\)"](#)

```
JMenuBar barraDeNavegacion = new JMenuBar();
JMenu opciones = new JMenu(s:"Opciones");
```

```
JMenuItem salir = new JMenuItem(text:"Salir");
salir.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(status:0);
    }
});
```

```
JMenuItem historial = new JMenuItem(text:"Historial");
historial.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JPanel panelHistorial = new JPanel();
        panelHistorial(panelHistorial, historialTxt);
        JOptionPane.showOptionDialog(parentComponent:null,
        new JScrollPane(panelHistorial), title:"Historial",
        JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE,
        icon:null, new Object[] {}, initialValue:null);
    }
});
```

```
JMenuItem sobreNosotros = new JMenuItem(text:"Sobre Nosotros");
sobreNosotros.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JEditorPane panelHTML = new JEditorPane();
        panelHtmlSobreNosotros(panelHTML);
        JOptionPane.showOptionDialog(parentComponent:null,
        new JScrollPane(panelHTML),
        title:"¡MAS! Sobre nosotros y nuestro partido político",
        JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE,
        icon:null, new Object[] {}, initialValue:null);
    }
});
```

JTextPane pantalla

```
JTextPane pantalla = new JTextPane();
pantalla.setBorder(new LineBorder(new Color(r:160, g:80, b:190), thickness:4));
pantalla.setMaximumSize(new Dimension(frame.getWidth()-10, height:300));
pantalla.setFont(new Font(name:"arial", Font.BOLD, size:32));
pantalla.setEditable(b:false);
pantalla.setText(t:"0");

panelDePaneles.add(pantalla);
```


Importaciones

```
import java.text.*;

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import javax.swing.border.*;
```

`funcionBoton(JPanel, String, JTextPane, StringBuilder, StringBuilder, int[], String[])`

```
public static void funcionBoton(JPanel panelDeBotones, String entrada, JTextPane pantalla, StringBuilder operacion, String historialTxt, int[] numeros, String[] operadores) {
    boolean pintar = false;
    String partes[] = separarOperacion(operacion);
    switch (queBotonEs(entrada)) {
        case "numero":
            int entradaEntero = Integer.parseInt(entrada);
            if (partes[partes.length - 1].contains(s:".")) {
                if (cantidadDecimales(partes[partes.length - 1]) == 1) {
                    estadoBotones(panelDeBotones, operacion, estado:"deshabilitado", tipo:"numeros", numeros, operadores);
                }
                operacion.append(entradaEntero);
                pintar = true;
                estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"operadores", numeros, operadores);
                RoundedButton coma = buscarBoton(panelDeBotones, busqueda:","");
                coma.setEnabled(b:false);
            } else {
                operacion.append(entradaEntero);
                pintar = true;
                estadoBotones(panelDeBotones, operacion, estado:"habilitado", tipo:"operadores", numeros, operadores);
            }
            break;

        case "cancelar":
            operacion.setLength(newLength:0);
            habilitarTodo(panelDeBotones, operacion, numeros, operadores);
            repintarPantalla(pantalla, infoPantalla:"0");
            pintar = false;
            break;

        case "igual":
            if (partes.length == 3) {
                String resultado = separarYOperar(operacion);
                pintar = false;
                repintarPantalla(pantalla, resultado);
                habilitarTodo(panelDeBotones, operacion, numeros, operadores);
                historialTxt.append(operacion.toString() + " = " + resultado + "%");
                operacion.setLength(newLength:0);
            }
            break;

        case "coma":
            if (operacion.length() == 0 && historialTxt.length() != 0 && ultimoResultado(historialTxt).contains(s:",")) {
                String ultimoResultado = ultimoResultado(historialTxt);
                operacion.append(ultimoResultado + entrada);
            } else if (operacion.length() == 0 || partes.length == 2) {
                operacion.append(str:"0.");
            } else {
                operacion.append(str:".");
            }
            estadoBotones(panelDeBotones, operacion, estado:"deshabilitado", tipo:"operadores", numeros, operadores);
            pintar = true;
            break;

        case "operador":
            if (partes.length == 3) {
                String resultado = separarYOperar(operacion);
                historialTxt.append(operacion.toString() + " = " + resultado + "%");
                operacion.setLength(newLength:0);
                operacion.append(resultado + " " + entrada + " ");
            } else if (operacion.length() == 0 && historialTxt.length() != 0) {
                String ultimoResultado = ultimoResultado(historialTxt);
                operacion.append(ultimoResultado + " " + entrada + " ");
            } else if (operacion.length() != 0 && partes.length != 2) {
                operacion.append(" " + entrada + " ");
            }
            pintar = true;
            estadoBotonesParaOperadores(panelDeBotones, operacion, numeros, operadores);
            break;
    }
    if (pintar) {
        repintarPantalla(pantalla, operacion.toString());
    }
}
```

- Para ver API's de java:
[Java API](#)
- El copiloto de confianza:
[ChatGPT](#)