PRÁCTICA 6- GESTIÓN DE EVENTOS EN JAVA

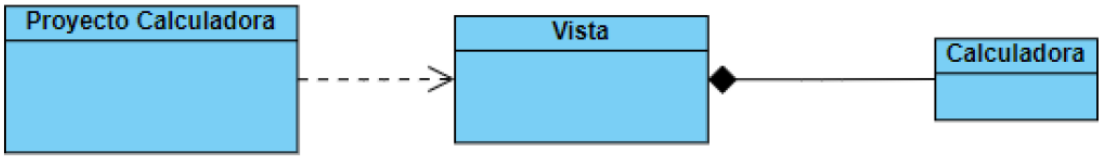
ALUMNO: SÁNCHEZ DE LOS RÍOS FLAVIO JOSUÉ

TINOCO VIDEGARAY SERGIO ERNESTO

PROFESOR: HERNÁNDEZ CRUZ MACARIO

FECHA: 11 DE NOVIEMBRE DE 2021

En esta práctica se va a desarrollar una calculadora científica implementando una interfaz gráfica de usuario y algunas funciones matemáticas de java.



Clase Calculadora

En esta clase se van a definir las operaciones matemáticas tanto aritméticas como algebraicas y algunas funciones trascendentales como las funciones trigonométricas seno, coseno y tangente.

```
public class Calculadora
{
        private double memoria=0;
        private char operadorAnterior='=';
        private boolean radianes=true;
```

Se ponen las funciones de suma, resta, multiplicación, división, inversa y raíz cuadrada

```
public void operacion(double numero, char operador)
{

    if(operadorAnterior=='=')
        memoria = numero;
    else
        switch(operadorAnterior)
        {
           case '+': memoria+=numero; break;
            case '-': memoria-=numero; break;
            case 'x': memoria*=numero; break;
            case '÷': memoria/=numero; break;
            case 'I': memoria=1/memoria; break;
            case '√': memoria=sqrt(memoria); break;
```

En el caso de que el carácter de operación corresponda con la ‘S’, se va a calcular el seno del valor almacenado en la memoria, y en caso de ‘C’ se calculara el coseno.

```
  case 'S':
      if(radianes){
            memoria=Math.sin(memoria);
      }
      else{
          memoria=Math.sin((memoria*Math.PI)/180);
      }
      break;
  case 'C': if(radianes){
            memoria=Math.cos(memoria);
      }
      else{
          memoria=Math.cos((memoria*Math.PI)/180);
      }
  break;
```

En el caso de la tangente se va a evaluar el valor que haya en memoria y si el valor corresponde con el Angulo pi/2 o bien 90° se mostrara un mensaje de error matemático y se sale del switch. En caso contrario se calcula el valor de la tangente del Angulo.

```
case 'T':
    if(memoria==90.0 || memoria==(Math.PI/2)){
        JOptionPane.showMessageDialog(null,
                "La tangente de 90° no esta definida",
                "Error matematico",
                JOptionPane.ERROR_MESSAGE);
        break;
    }

    if(radianes){
          memoria=Math.tan(memoria);
    }
    else{
        memoria=Math.tan((memoria*Math.PI)/180);
    }
break;
```

En el caso de la raíz cubica se manda a llamar al método Math.cbrt para calcular la raíz cubica de la variable memoria.

```
case '∛': memoria=Math.cbrt(memoria); break;
case '^': memoria=memoria*memoria; break;
```

Se programa el método clearMemory para borrar el valor de la variable memoria e instanciar el operador en '='.

```
public void clearMemory()
{
    this.memoria=0;
    this.operadorAnterior='=';
}
```

Método para obtener el valor almacenado en la variable memoria.

```
public double getMemoria()
{
        return memoria;
}
```

Método para asignar a la variable radianes el valor verdadero, lo que indicaría que la calculadora va a operar en radianes.

```
public void setRadianes()
{
    radianes=true;
}
```

Análogamente se programa un método que asigne el valor falso a la bandera radianes, indicando que se estará operando en grados sexagesimales.

```
public void setDegrees()
{
    radianes=false;
}
```

Clase Vista

Se procede con la programación de la clase vista para la interfaz gráfica de usuario.

```
public class Vista extends JFrame
{
        // El objeto calculadora es el que realmente realiza las operaciones
        private final Calculadora calculator = new Calculadora();
        private final JTextField display = new JTextField();

        private final JButton b0 = new JButton("0");
        private final JButton b1 = new JButton("1");
        private final JButton b2 = new JButton("2");
        private final JButton b3 = new JButton("3");
        private final JButton b4 = new JButton("4");
        private final JButton b5 = new JButton("5");
        private final JButton b6 = new JButton("6");
        private final JButton b7 = new JButton("7");
        private final JButton b8 = new JButton("8");
        private final JButton b9 = new JButton("9");
        private final JButton mas= new JButton("+");
        private final JButton menos=new JButton("-");
        private final JButton igual=new JButton("=");
        private final JButton cE   = new JButton("CE");
```

Se definen las variables de los componentes de tipo button para las operaciones y los valores numéricos de la calculadora.

```
private final JButton c = new JButton("c");
 private final JButton e = new JButton("e");
 private final JButton pi = new JButton("π");
 private final JButton multi = new JButton("x");
 private final JButton div = new JButton("÷");
 private final JButton raiz2 = new JButton("√");
 private final JButton raiz3 = new JButton("∛");
 private final JButton potencia = new JButton("^2");
 private final JButton inverso = new JButton("Inv");
 private final JButton sen = new JButton("Sen");
 private final JButton cos= new JButton("Cos");
 private final JButton tan=new JButton("Tan");
 private final JButton punto=new JButton(".");
```

Constructor de la clase vista para inicializar las variables de la GUI y asignar las coordenadas de la ventana principal

```
public Vista()
{
        initComponents();
        this.setTitle("Calculadora");
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

        this.setBounds(100,100,610,370);

        this.setVisible(true);

}
```

Método para generar la barra de menú con las opciones para salir y alternan entre el modo radianes y grados.

```
public void initComponents()
{
    // Diseña el menu
    JMenuBar barraMenus = new JMenuBar();
    JMenu archivo       = new JMenu("Archivo");
    JMenuItem salir     = new JMenuItem("Salir");
    JMenuItem Radianes     = new JMenuItem("Radianes");
    JMenuItem Grados     = new JMenuItem("Grados");
    this.setJMenuBar(barraMenus);
```

Se añaden al frame actual los componentes de menú

```
barraMenus.add(archivo);
archivo.add(salir);
 archivo.add(Radianes);
 archivo.add(Grados);
```

Se asignan las coordenadas de los componentes de la vista como los botones y el display de nuestra calculadora.

```
this.setLayout(null); // Se desh
display.setBounds(10,5,580,40);
b1.setBounds(10,50,50,50);
b2.setBounds(70,50,50,50);
b3.setBounds(130,50,50,50);
mas.setBounds(190,50,50,50);

b4.setBounds(10,110,50,50);
b5.setBounds(70,110,50,50);
b6.setBounds(130,110,50,50);
menos.setBounds(190,110,50,50);

b7.setBounds(10,170,50,50);
b8.setBounds(70,170,50,50);
b9.setBounds(130,170,50,50);
b0.setBounds(70,230,50,50);
e.setBounds(190,170,50,50);
cE.setBounds(10,230,50,50);

c.setBounds(130,230,50,50);
pi.setBounds(190,230,50,50);
multi.setBounds(250,50,50,50);
div.setBounds(250,110,50,50);
igual.setBounds(250,230,100,50);
punto.setBounds(250,170,50,50);
```

Similarmente se colocan las coordenadas y los tamaños de los botones para las funciones adicionales.

```
//Funciones
potencia.setBounds(330,50,70,50);
raiz2.setBounds(420,50,70,50);
raiz3.setBounds(510,50,70,50);
sen.setBounds(330,110,70,50);
cos.setBounds(420,110,70,50);
tan.setBounds(510,110,70,50);
inverso.setBounds(420,170,70,50);
```

Se asignan los atributos del display para darle un formato a los números y evitar que el usuario ingrese datos directamente en el display.

```
display.setBackground(Color.black);
display.setForeground(Color.orange);
display.setFont(new Font("Consolas",Font.BOLD, 26));
display.setHorizontalAlignment(JTextField.RIGHT);
display.setEditable(false);
```

Se añaden los componentes al frame actual.

```
// Agrega los elementos al JFrame
this.add(display);
this.add(b1);
this.add(b2);
this.add(b3);
this.add(b4);
this.add(b5);
this.add(b6);
this.add(b7);
this.add(b8);
this.add(b9);
this.add(b0);
this.add(mas);
this.add(menos);
this.add(igual);
this.add(cE);
```

Se asignan los eventos correspondientes a cada botón de la interfaz invocando a la función gestionarBotones.

```
b0.addActionListener(evt -> gestionarBotones(evt));
b1.addActionListener(evt -> gestionarBotones(evt));
b2.addActionListener(evt -> gestionarBotones(evt));
b3.addActionListener(evt -> gestionarBotones(evt));
b4.addActionListener(evt -> gestionarBotones(evt));
b5.addActionListener(evt -> gestionarBotones(evt));
b6.addActionListener(evt -> gestionarBotones(evt));
b7.addActionListener(evt -> gestionarBotones(evt));
b8.addActionListener(evt -> gestionarBotones(evt));
b9.addActionListener(evt -> gestionarBotones(evt));
mas.addActionListener(evt -> gestionarBotones(evt));
menos.addActionListener(evt -> gestionarBotones(evt));
igual.addActionListener(evt -> gestionarBotones(evt));
```

De igual forma se asignan los métodos gestionarCe y gestionar C en el caso de los botones para eliminar la memoria o el valor más reciente en el display.

```
cE.addActionListener(evt -> gestionarCE(evt));
 c.addActionListener(evt -> gestionarC(evt));
```

Se asignan los eventos para el botón salir y las opciones de radianes y grados.

```
salir.addActionListener(evt -> gestionarSalir(evt));
Radianes.addActionListener(evt -> calculator.setRadianes());
Grados.addActionListener(evt -> calculator.setDegrees());
```

Se programan los métodos para gestionar el comportamiento del botón salir el cual va a mostrar una ventana emergente para preguntarle al usuario si desea salir del programa.

```
    class MyWindowAdapter extends WindowAdapter
    {
        @Override
        public void windowClosing(WindowEvent e)
        {
            exit();
        }
    }
    addWindowListener(new MyWindowAdapter());

}

// Métodos para gestión de eventos

public void gestionarSalir(java.awt.event.ActionEvent evt)
{
    exit();
}

public void exit()
{
    int respuesta = JOptionPane.showConfirmDialog(rootPane,
     "Desea salir?","Federación deportiva",JOptionPane.YES_NO_OPTION);
    if(respuesta==JOptionPane.YES_OPTION) System.exit(0);
}
```

Método para asignar los valores al display de la calculadora e invocar a los métodos para las operaciones.

```
public void gestionarBotones(java.awt.event.ActionEvent evt)
{

    String textoBoton = evt.getActionCommand();

    // Cada botón tiene un simbolo numérico o operador en la cara del botón
    // Se obtiene con charAt

    char simbolo = textoBoton.charAt(0);
```

En el caso de que el botón presionado corresponda a algún número o punto decimal, únicamente se va a agregar dicho número al display haciendo una concatenación del valor en el display y el símbolo ingresado por medio del botón.

```
if( (simbolo >= '0' && simbolo <= '9') ||
        simbolo == '.' || simbolo == 'e'|| simbolo == 'π') // En caso de número o punto
{
    if(nuevo) display.setText(""); // Si es una nueva cifra se borra el display

        display.setText(display.getText()+simbolo);
    nuevo = false; // Se pone en falso cuando se pone el primer dígito de una cifra
}
```

En caso de que el botón corresponda a algún operador se va a asignar el valor del display a una variable double y se va a invocar al método operación de la clase calculadora mandándole como parámetro el número en el display y el símbolo del operador.

```
else if(simbolo=='+'||simbolo=='-'||
        simbolo=='x'||simbolo=='÷'||
        simbolo=='√'||simbolo=='∛'||
        simbolo=='^'||simbolo=='I'||
        simbolo=='S'||simbolo=='C'||
        simbolo=='T'||simbolo=='=') // En caso de operador
{
    double numero;
    if(display.getText().equals("e"))numero=Math.E;
    else if(display.getText().equals("π"))numero=Math.PI;
    else numero=Double.parseDouble(display.getText());

    calculator.operacion(numero,simbolo);        // Invoca la funcionalidad de la calculadora
    display.setText(""+calculator.getMemoria());  // Obtiene el estado de la memoria de la calculadora
    nuevo = true;
}
```

Método para eliminar el valor más reciente ingresado al display. Únicamente se va a reescribir el contenido del display con el valor de la memoria de nuestra clase calculadora y se le indicara a la bandera nuevo que vamos a ingresar un nuevo valor numérico.

```
// Gestiona botón CE
public void gestionarCE(java.awt.event.ActionEvent evt)
{
    // Codigo para procesar CE
      display.setText(""+calculator.getMemoria());
      nuevo=true;
}
```

Análogamente con el método gestionarC se va a eliminar tanto el valor reciente como el valor almacenado previamente en memoria y se vacía el contenido del display.

```
 public void gestionarC(java.awt.event.ActionEvent evt)
{
    // Codigo para procesar CE
    display.setText("");
    calculator.clearMemory();
    nuevo=true;
}
```

Clase main: Únicamente se instancia un objeto anónimo de la clase Vista para generar la interfaz de la calculadora.

```
public class ProyectoCalculadora
{

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        new Vista();
    }

}
```

Conclusiones:

El uso de clases en java hace más fácil la gestión de datos almacenados en una interfaz de usuario, permitiéndonos una retroalimentación entre las clases.

De igual forma el uso de banderas resulta ser un aspecto clave en el desarrollo de esta práctica ya que nos permite almacenar el estado de algún parámetro interno del programa como el modo de operación de grados y radianes o el ingreso de algún nuevo valor numérico en el display.