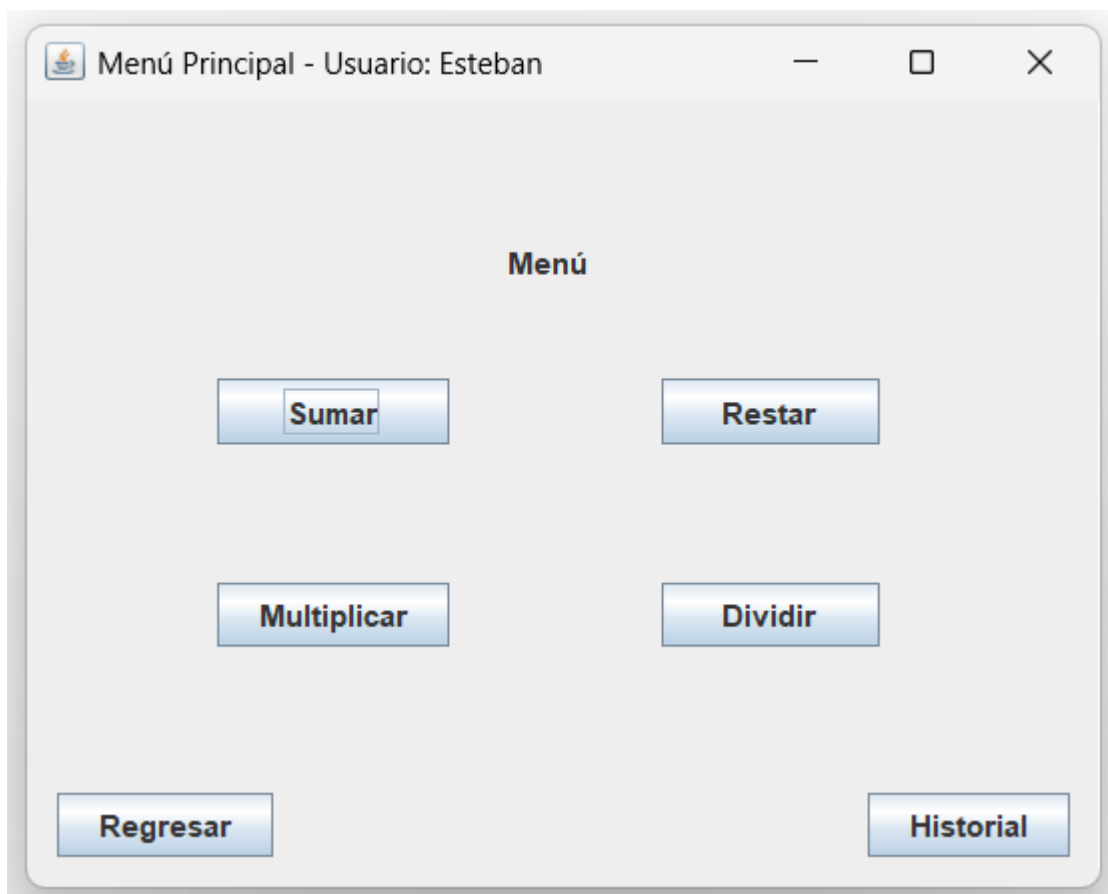


A login window with a light gray background. At the top left is a small icon of a cup with a flame. At the top right are standard window controls: a minus sign, a square, and an 'X'. The text "Ingrese el nombre de usuario:" is centered. Below it is a text input field containing "Esteban". At the bottom right is a blue button labeled "Siguiete".

Ingrese el nombre de usuario:

Esteban

Siguiete



A main menu window with a light gray background. The title bar reads "Menú Principal - Usuario: Esteban". At the top left is a small icon of a cup with a flame. At the top right are standard window controls: a minus sign, a square, and an 'X'. The text "Menú" is centered. Below it are four blue buttons arranged in a 2x2 grid: "Sumar", "Restar", "Multiplicar", and "Dividir". At the bottom left is a blue button labeled "Regresar", and at the bottom right is a blue button labeled "Historial".

Menú Principal - Usuario: Esteban

Menú

Sumar Restar

Multiplicar Dividir

Regresar Historial



Suma - Usuario: Esteban



Sumar

+

Calcular

Regresar



Resta - Usuario: Esteban



Rest...

-

Calcular

Regresar

Multiplicación - Usuario: Esteban

Multiplicar

*

Calcular

Regresar

División - Usuario: Esteban


Dividir

/


Calcular

Regresar

Error

 No se puede dividir entre cero

OK

 División - Usuario: Esteban

—

□

×

Dividir

9


/

3

Calcular

3.0

Regresar

 Historial - Usuario: Esteban

—

□

×

Historial

Usuario: Esteban

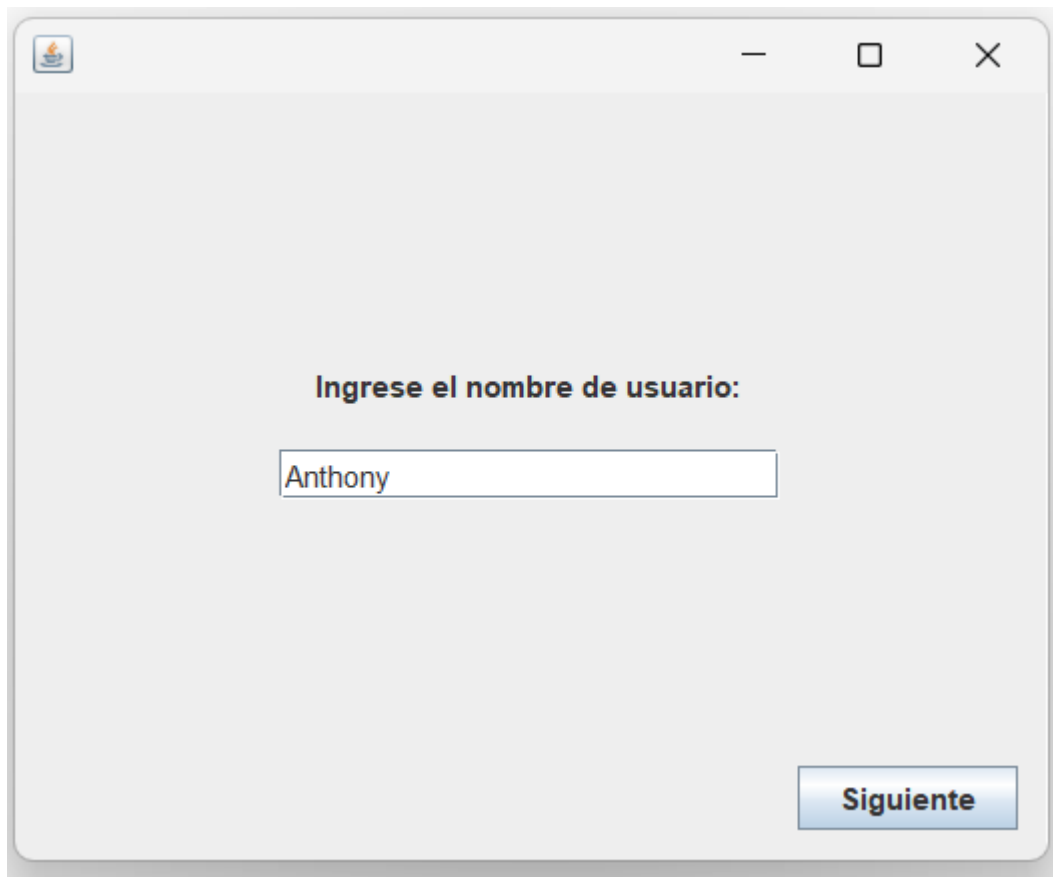
25.00 + 36.00 = 61.00

99.00 - 66.00 = 33.00

9.00 * 88.00 = 792.00

9.00 / 3.00 = 3.00

Regresar

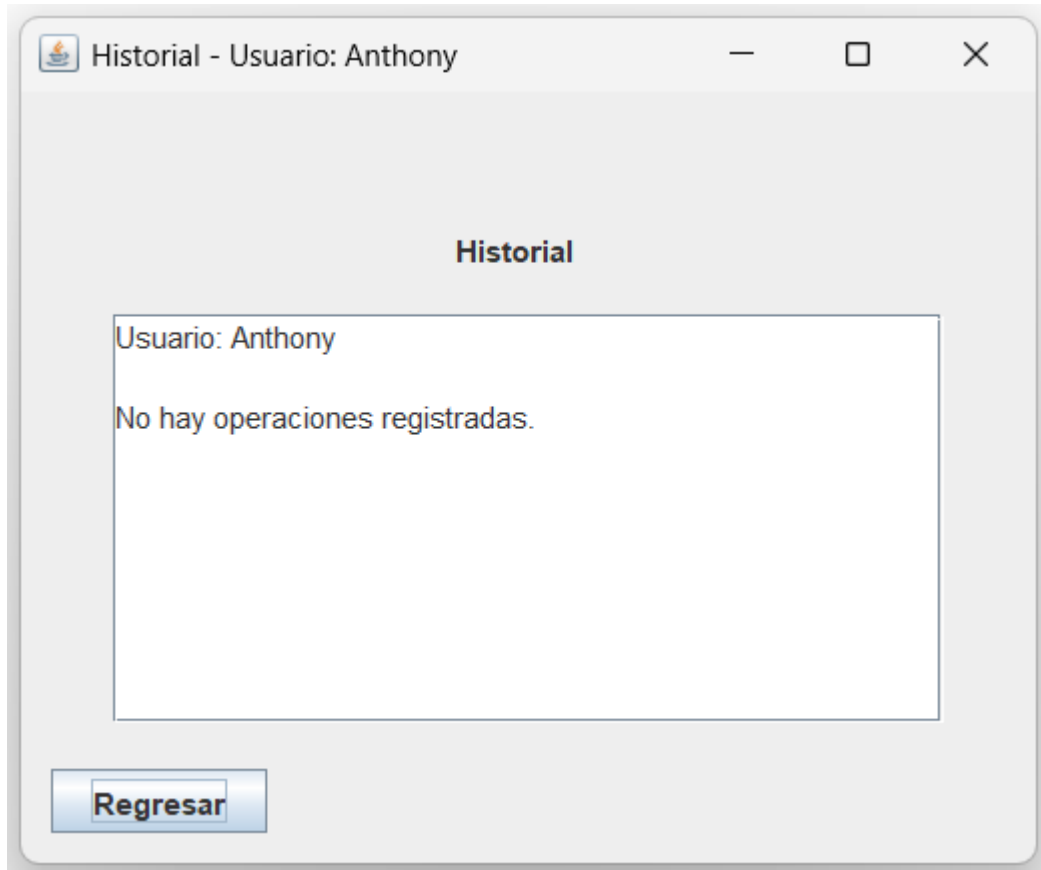


A window with a title bar containing a small icon, a minus sign, a maximize button, and a close button. The main area has a light gray background. In the center, the text "Ingrese el nombre de usuario:" is displayed. Below it is a text input field containing the name "Anthony". In the bottom right corner, there is a blue button with the text "Siguiete".

Ingrese el nombre de usuario:

Anthony

Siguiete



A window with a title bar containing a small icon, the text "Historial - Usuario: Anthony", a minus sign, a maximize button, and a close button. The main area has a light gray background. In the center, the text "Historial" is displayed. Below it is a large white rectangular area containing the text "Usuario: Anthony" and "No hay operaciones registradas.". In the bottom left corner, there is a blue button with the text "Regresar".

Historial

Usuario: Anthony

No hay operaciones registradas.

Regresar

Clase Main:

```
package actividad7;

import Controller.Controlador;
import View.MenuUsuario;

public class Actividad7 {
    public static void main(String[] args) {

        Controlador controlador = new Controlador();

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                MenuUsuario menuUsuario = new MenuUsuario();
                menuUsuario.setLocationRelativeTo(null);
                menuUsuario.setVisible(true);

                controlador.setMenuUsuario(menuUsuario);
            }
        });
    }
}
```

Clase Controlador:

```
package Controller;

import Model.Usuario;
import Model.Operacion;
import View.*;

public class Controlador {
```

```
private Usuario usuarioActual;

private MenuUsuario menuUsuario;

private MenuOpciones menuOpciones;

private MenuSuma menuSuma;

private MenuResta menuResta;

private MenuMultiplicacion menuMultiplicacion;

private MenuDivision menuDivision;

private MenuHistorial menuHistorial;


public Controlador() {

    this.menuOpciones = new MenuOpciones();

    configurarMenuOpciones();

}


public void setMenuUsuario(MenuUsuario menuUsuario) {

    this.menuUsuario = menuUsuario;

    configurarMenuUsuario();

}


private void configurarMenuUsuario() {

    menuUsuario.getSiguienteButton().addActionListener(e -> {

        String nombreUsuario = menuUsuario.getUsuarioText().getText().trim();

        if (!nombreUsuario.isEmpty()) {

            usuarioActual = new Usuario(nombreUsuario);

            menuOpciones.setTitle("Menú Principal - Usuario: " + nombreUsuario);
```

```

        menuUsuario.setVisible(false);

        menuOpciones.setLocationRelativeTo(null);

        menuOpciones.setVisible(true);
    } else {

        javax.swing.JOptionPane.showMessageDialog(menuUsuario,

            "Por favor ingrese un nombre de usuario",

            "Error",

            javax.swing.JOptionPane.ERROR_MESSAGE);

    }

});
}

```

```

private void configurarMenuOpciones() {

    menuOpciones.getSumarButton().addActionListener(e -> abrirMenuSuma());

    menuOpciones.getRestarButton().addActionListener(e -> abrirMenuResta());

    menuOpciones.getMultiplicarButton().addActionListener(e ->
abrirMenuMultiplicacion());

    menuOpciones.getDividirButton().addActionListener(e -> abrirMenuDivision());

    menuOpciones.getHistorialButton().addActionListener(e -> abrirMenuHistorial());

    menuOpciones.getRegresarButton().addActionListener(e ->
regresarMenuUsuario());

}

```

```

private void abrirMenuSuma() {

    if (menuSuma == null) {

        menuSuma = new MenuSuma();

        configurarMenuSuma();

    }

}

```



```
menuSuma.setTitle("Suma - Usuario: " + usuarioActual.getNombre());  
menuOpciones.setVisible(false);  
menuSuma.setLocationRelativeTo(null);  
menuSuma.setVisible(true);  
}
```

```
private void abrirMenuResta() {  
    if (menuResta == null) {  
        menuResta = new MenuResta();  
        configurarMenuResta();  
    }  
    menuResta.setTitle("Resta - Usuario: " + usuarioActual.getNombre());  
    menuOpciones.setVisible(false);  
    menuResta.setLocationRelativeTo(null);  
    menuResta.setVisible(true);  
}
```

```
private void abrirMenuMultiplicacion() {  
    if (menuMultiplicacion == null) {  
        menuMultiplicacion = new MenuMultiplicacion();  
        configurarMenuMultiplicacion();  
    }  
    menuMultiplicacion.setTitle("Multiplicación - Usuario: " +  
usuarioActual.getNombre());  
    menuOpciones.setVisible(false);  
    menuMultiplicacion.setLocationRelativeTo(null);  
    menuMultiplicacion.setVisible(true);  
}
```

```
private void abrirMenuDivision() {  
    if (menuDivision == null) {  
        menuDivision = new MenuDivision();  
        configurarMenuDivision();  
    }  
    menuDivision.setTitle("División - Usuario: " + usuarioActual.getNombre());  
    menuOpciones.setVisible(false);  
    menuDivision.setLocationRelativeTo(null);  
    menuDivision.setVisible(true);  
}  
  
private void abrirMenuHistorial() {  
    if (menuHistorial == null) {  
        menuHistorial = new MenuHistorial();  
        configurarMenuHistorial();  
    }  
    menuHistorial.actualizarHistorial(usuarioActual);  
    menuHistorial.setTitle("Historial - Usuario: " + usuarioActual.getNombre());  
    menuOpciones.setVisible(false);  
    menuHistorial.setLocationRelativeTo(null);  
    menuHistorial.setVisible(true);  
}  
  
private void configurarMenuSuma() {  
    menuSuma.getRegresarButton().addActionListener(e -> regresarMenuOpciones());  
    menuSuma.getCalcularButton().addActionListener(e -> calcularSuma());  
}
```

```
private void configurarMenuResta() {  
    menuResta.getRegresarButton().addActionListener(e -> regresarMenuOpciones());  
    menuResta.getCalcularButton().addActionListener(e -> calcularResta());  
}
```

```
private void configurarMenuMultiplicacion() {  
    menuMultiplicacion.getRegresarButton().addActionListener(e ->  
regresarMenuOpciones());  
    menuMultiplicacion.getCalcularButton().addActionListener(e ->  
calcularMultiplicacion());  
}
```

```
private void configurarMenuDivision() {  
    menuDivision.getRegresarButton().addActionListener(e ->  
regresarMenuOpciones());  
    menuDivision.getCalcularButton().addActionListener(e -> calcularDivision());  
}
```

```
private void configurarMenuHistorial() {  
    menuHistorial.getRegresarButton().addActionListener(e ->  
regresarMenuOpciones());  
}
```

```
private void calcularSuma() {  
    try {  
        double num1 = Double.parseDouble(menuSuma.getNum1Text().getText());  
        double num2 = Double.parseDouble(menuSuma.getNum2Text().getText());  
        double resultado = num1 + num2;
```

```
Operacion operacion = new Operacion("suma", num1, num2, resultado);
usuarioActual.agregarOperacion(operacion);

menuSuma.getResultadoText().setText(String.valueOf(resultado));
} catch (NumberFormatException ex) {
    mostrarError(menuSuma, "Por favor ingrese números válidos");
}
}

private void calcularResta() {
    try {
        double num1 = Double.parseDouble(menuResta.getNum1Text().getText());
        double num2 = Double.parseDouble(menuResta.getNum2Text().getText());
        double resultado = num1 - num2;

        Operacion operacion = new Operacion("resta", num1, num2, resultado);
        usuarioActual.agregarOperacion(operacion);

        menuResta.getResultadoText().setText(String.valueOf(resultado));
    } catch (NumberFormatException ex) {
        mostrarError(menuResta, "Por favor ingrese números válidos");
    }
}

private void calcularMultiplicacion() {
    try {
```

```
        double num1 =  
Double.parseDouble(menuMultiplicacion.getNum1Text().getText());  
  
        double num2 =  
Double.parseDouble(menuMultiplicacion.getNum2Text().getText());  
  
        double resultado = num1 * num2;  
  
        Operacion operacion = new Operacion("multiplicacion", num1, num2, resultado);  
        usuarioActual.agregarOperacion(operacion);  
  
        menuMultiplicacion.getResultadoText().setText(String.valueOf(resultado));  
    } catch (NumberFormatException ex) {  
        mostrarError(menuMultiplicacion, "Por favor ingrese números válidos");  
    }  
}
```

```
private void calcularDivision() {  
    try {  
        double num1 = Double.parseDouble(menuDivision.getNum1Text().getText());  
        double num2 = Double.parseDouble(menuDivision.getNum2Text().getText());  
  
        if (num2 == 0) {  
            mostrarError(menuDivision, "No se puede dividir entre cero");  
            return;  
        }  
  
        double resultado = num1 / num2;  
  
        Operacion operacion = new Operacion("division", num1, num2, resultado);
```

```
usuarioActual.agregarOperacion(operacion);
```

```
menuDivision.getResultadoText().setText(String.valueOf(resultado));
```

```
} catch (NumberFormatException ex) {
```

```
    mostrarError(menuDivision, "Por favor ingrese números válidos");
```

```
}
```

```
}
```

```
private void mostrarError(javax.swing.JFrame parent, String mensaje) {
```

```
    javax.swing.JOptionPane.showMessageDialog(parent, mensaje, "Error",
```

```
        javax.swing.JOptionPane.ERROR_MESSAGE);
```

```
}
```

```
private void regresarMenuOpciones() {
```

```
    if (menuSuma != null && menuSuma.isVisible()) menuSuma.setVisible(false);
```

```
    if (menuResta != null && menuResta.isVisible()) menuResta.setVisible(false);
```

```
    if (menuMultiplicacion != null && menuMultiplicacion.isVisible())
```

```
menuMultiplicacion.setVisible(false);
```

```
    if (menuDivision != null && menuDivision.isVisible()) menuDivision.setVisible(false);
```

```
    if (menuHistorial != null && menuHistorial.isVisible())
```

```
menuHistorial.setVisible(false);
```

```
    menuOpciones.setLocationRelativeTo(null);
```

```
    menuOpciones.setVisible(true);
```

```
}
```

```
private void regresarMenuUsuario() {
```

```
    menuOpciones.setVisible(false);
```

```
        menuUsuario.getUsuarioText().setText("");
        menuUsuario.setLocationRelativeTo(null);
        menuUsuario.setVisible(true);
    }

    public Usuario getUsuarioActual() {
        return usuarioActual;
    }
}
```

Clase Usuario:

```
package Model;

import java.util.ArrayList;
import java.util.List;

public class Usuario {
    private String nombre;
    private List<Operacion> historialOperaciones;

    public Usuario(String nombre) {
        this.nombre = nombre;
        this.historialOperaciones = new ArrayList<>();
    }

    public void agregarOperacion(Operacion operacion) {
        this.historialOperaciones.add(operacion);
    }
}
```

```
}
```

```
public String getNombre() {  
    return nombre;  
}
```

```
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}
```

```
public List<Operacion> getHistorialOperaciones() {  
    return new ArrayList<>(historialOperaciones);  
}
```

```
public int getTotalOperaciones() {  
    return historialOperaciones.size();  
}  
}
```

Clase Operación:

```
package Model;
```

```
public class Operacion {  
    private String tipo;  
    private double numero1;  
    private double numero2;  
    private double resultado;
```



```
public Operacion(String tipo, double numero1, double numero2, double resultado) {  
    this.tipo = tipo;  
    this.numero1 = numero1;  
    this.numero2 = numero2;  
    this.resultado = resultado;  
}
```

```
public String getTipo() {  
    return tipo;  
}
```

```
public double getNumero1() {  
    return numero1;  
}
```

```
public double getNumero2() {  
    return numero2;  
}
```

```
public double getResultado() {  
    return resultado;  
}
```

```
@Override
```

```
public String toString() {  
    return String.format("%.2f %s %.2f = %.2f",  
        numero1, getSimboloOperacion(), numero2, resultado);  
}
```

```
private String getSimboloOperacion() {  
    switch (tipo.toLowerCase()) {  
        case "suma": return "+";  
        case "resta": return "-";  
        case "multiplicacion": return "*";  
        case "division": return "/";  
        default: return "?";  
    }  
}  
}
```