Universidad Nacional de Colombia

Tarea3 - Parte1

Autores:

Juan José Jiménez Maya

Programa: Programación Orientada a Objetos

Grupo: 3

Código: Propuesto23.java

```
package Tareas.Tarea3.Parte1;

import java.util.HashSet;
import java.util.Set;

public class Propuesto23 extends MDIElement<Propuesto23.Ecuacion> {

    public Propuesto23() {
        super("Propuesto 23", new Argument[]{
                new Argument<>("a", double.class),
                new Argument<>("b", double.class),
                new Argument<>("c", double.class)
        }, Ecuacion.class);
    }

    public static class Ecuacion {
        public double a;
        public double b;
        public double c;
        public Ecuacion(double a, double b, double c) {
            if (a == 0) throw new IllegalArgumentException("El valor a no puede ser 0");

            this.a = a;
            this.b = b;
            this.c = c;
        }

        public Set<Double> soluciones() {
            Set<Double> res = new HashSet<>();

            double sqrt = Math.sqrt(Math.pow(b, 2) - 4 * a * c);
            res.add((-b + sqrt) / (2 * a));
            res.add((-b - sqrt) / (2 * a));

            return res;
        }

        @Override
        public String toString() {
            StringBuilder stringBuilder = new StringBuilder("Las soluciones son: ");
            for (Double sol : this.soluciones()) {
                stringBuilder.append(sol).append(", ");
            }
            stringBuilder.delete(stringBuilder.length() - 2, stringBuilder.length());
            return stringBuilder.toString();
        }
    }

}
```

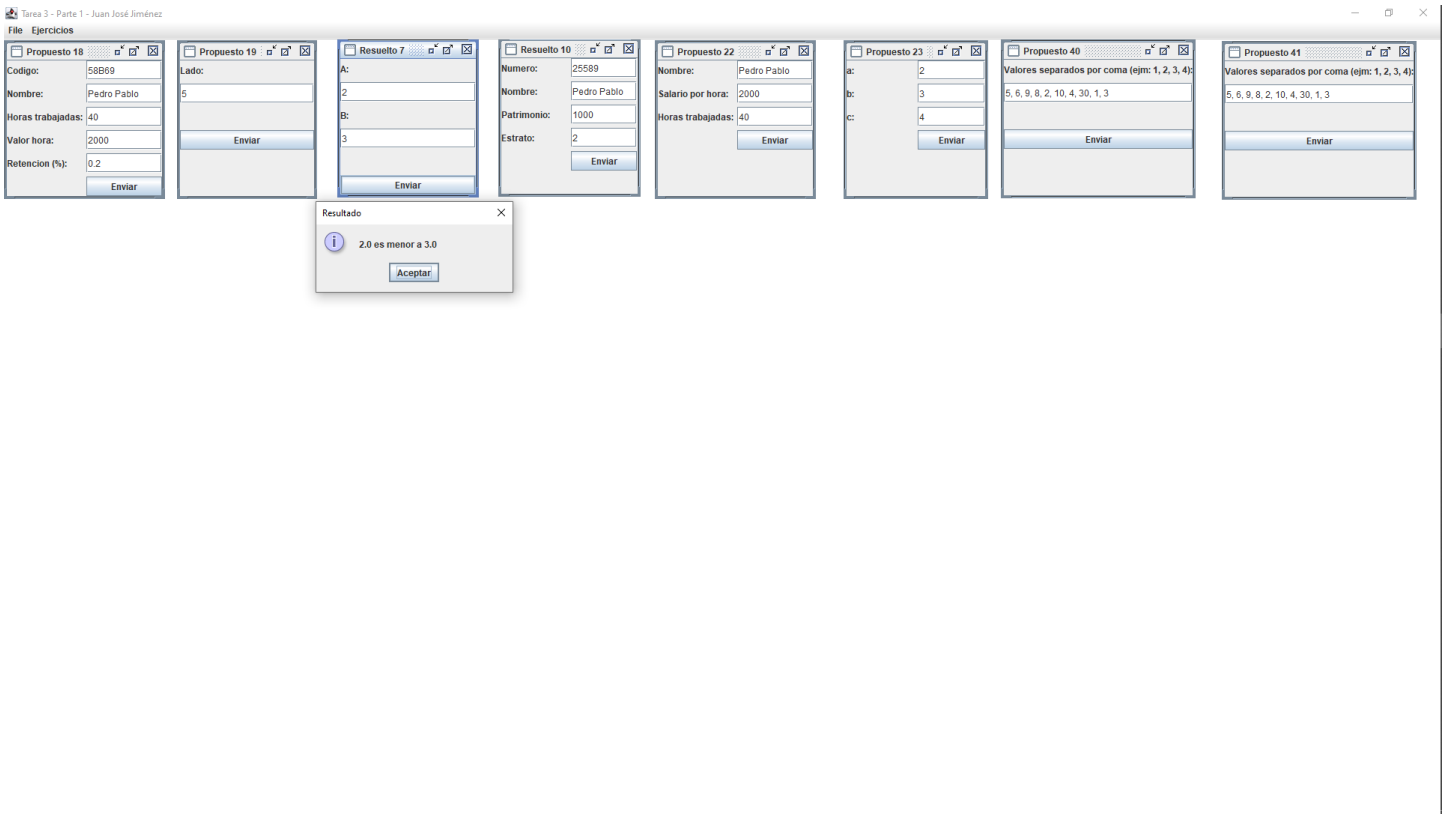
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto23.java

Código: Propuesto19.java

```
package Tareas.Tarea3.Parte1;

public class Propuesto19 extends MDIElement<Propuesto19.TrianguloEquilatero> {

    public Propuesto19() {
        super("Propuesto 19", new Argument[]{
                new Argument<>("Lado", float.class)
        }, TrianguloEquilatero.class);
    }

    private interface Figura {
        double perimetro();
        double altura();
        double area();
    }

    private abstract static class Triangulo implements Figura {
        protected final float a;
        protected final float b;
        protected final float c;
        public Triangulo(float a, float b, float c) {
            this.a = a;
            this.b = b;
            this.c = c;
        }

        @Override
        public double perimetro() {
            return a + b + c;
        }

        @Override
        public double altura() {
            return Math.sqrt(((4 * Math.pow(c, 2)) - Math.pow(b, 2)) / 4);
        }

        @Override
        public double area() {
            return (altura() * b) / 2;
        }

        @Override
        public String toString() {
            return String.format("""
                    Perimetro: %s,
                    Altura: %s,
                    Area: %s
                    """, perimetro(), altura(), area());
        }
    }

    public static class TrianguloEquilatero extends Triangulo {
        public TrianguloEquilatero(float a) {
            super(a, a, a);
        }
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto19.java

Imagen: Resuelto7.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Resuelto7.png

Código: MDIManager.java

```
package Tareas.Tarea3.Parte1;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.util.Arrays;
import java.util.Collection;

public class MDIManager extends JFrame {
    public static final Collection<MDIElement<?>> components = Arrays.asList(
            new Propuesto18(),
            new Propuesto19(),

            new Resuelto7(),
            new Resuelto10(),

            new Propuesto22(),
            new Propuesto23(),
            new Propuesto40(),
            new Propuesto41()
    );

    private final JDesktopPane desktopPane;

    public MDIManager() {
        // Configure the main window
        setTitle("Tarea 3 - Parte 1 - Juan José Jiménez");
        setSize(800, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // Create a JDesktopPane for MDI
        desktopPane = new JDesktopPane();
        setContentPane(desktopPane);

        // Set up menu bar
        setJMenuBar(createMenuBar());
    }

    private JMenuBar createMenuBar() {
        JMenuBar menuBar = new JMenuBar();

        // File menu
        JMenu fileMenu = new JMenu("File");
        JMenuItem exitItem = new JMenuItem(new AbstractAction("Exit") {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        fileMenu.add(exitItem);
        menuBar.add(fileMenu);

        // Components menu
        JMenu componentsMenu = new JMenu("Ejercicios");
        for (MDIElement<?> component : components) {
            JMenuItem componentItem = new JMenuItem(component.title);
            componentItem.addActionListener(e -> desktopPane.add(component.getFrame(this)));
            componentsMenu.add(componentItem);
        }
        menuBar.add(componentsMenu);

        return menuBar;
    }
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            MDIManager mdiManager = new MDIManager();
            mdiManager.setVisible(true);
        });
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/MDIManager.java

Código: Propuesto22.java

```
package Tareas.Tarea3.Parte1;

public class Propuesto22 extends MDIElement<Propuesto22.Empleado> {
    public Propuesto22() {
        super("Propuesto 22", new Argument[]{
                new Argument<>("Nombre", String.class),
                new Argument<>("Salario por hora", double.class),
                new Argument<>("Horas trabajadas", int.class)
        }, Empleado.class);
    }

    public record Empleado(String nombre, double salarioPorHora, int horasTrabajadas) {
        public static double salarioMensual(Empleado empleado) {
            return empleado.salarioPorHora * empleado.horasTrabajadas;
        }

        @Override
        public String toString() {
            double salario = salarioMensual(this);
            return nombre + (salario <= 450000 ? "" : String.format("\n$%s", salario));
        }
    }
}
```

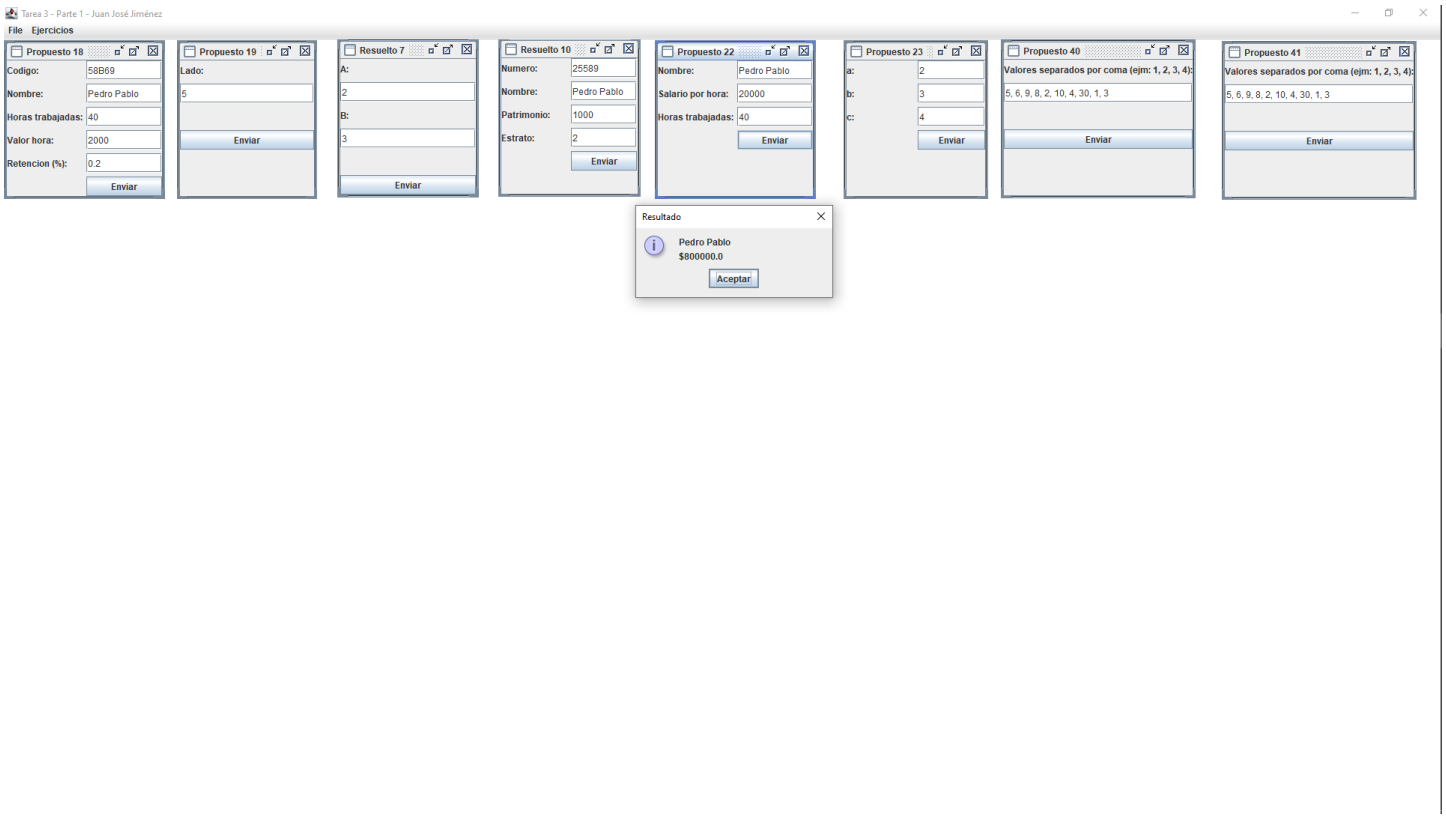
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto22.java

Código: Propuesto41.java

```
package Tareas.Tarea3.Parte1;

import java.util.Arrays;
import java.util.List;

public class Propuesto41 extends MDIElement<Propuesto41.GrupoDeNumeros> {
    public Propuesto41() {
        super("Propuesto 41", new Argument[]{
                new Argument<>("Valores separados por coma (ejm: 1, 2, 3, 4)", String.class)
        }, GrupoDeNumeros.class);
    }

    public static class GrupoDeNumeros {
        private final List<Double> valores;

        /**
         * Recibe una cadena de texto con los valores de los enteros separados por comas.
         * <p>Ejm: "1, 2, 3, 4, 5"
         * @param valores Cadena de texto con los valores de los enteros separados por comas.
         */
        public GrupoDeNumeros(String valores) {
            this.valores = Arrays.stream(valores.replaceAll("\\s+", "").split(","))
                    .map(Double::parseDouble)
                    .filter(x -> x >= 0) // El enunciado dice que los valores son positivos
                    .toList();
        }

        public double max() {
            return valores.stream().max(Double::compare).orElseThrow();
        }

        @Override
        public String toString() {
            return """
                    Grupo de Numeros: %s
                    Mayor: %s
                    """.formatted(valores, max());
        }
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto41.java

Imagen: Propuesto22.png



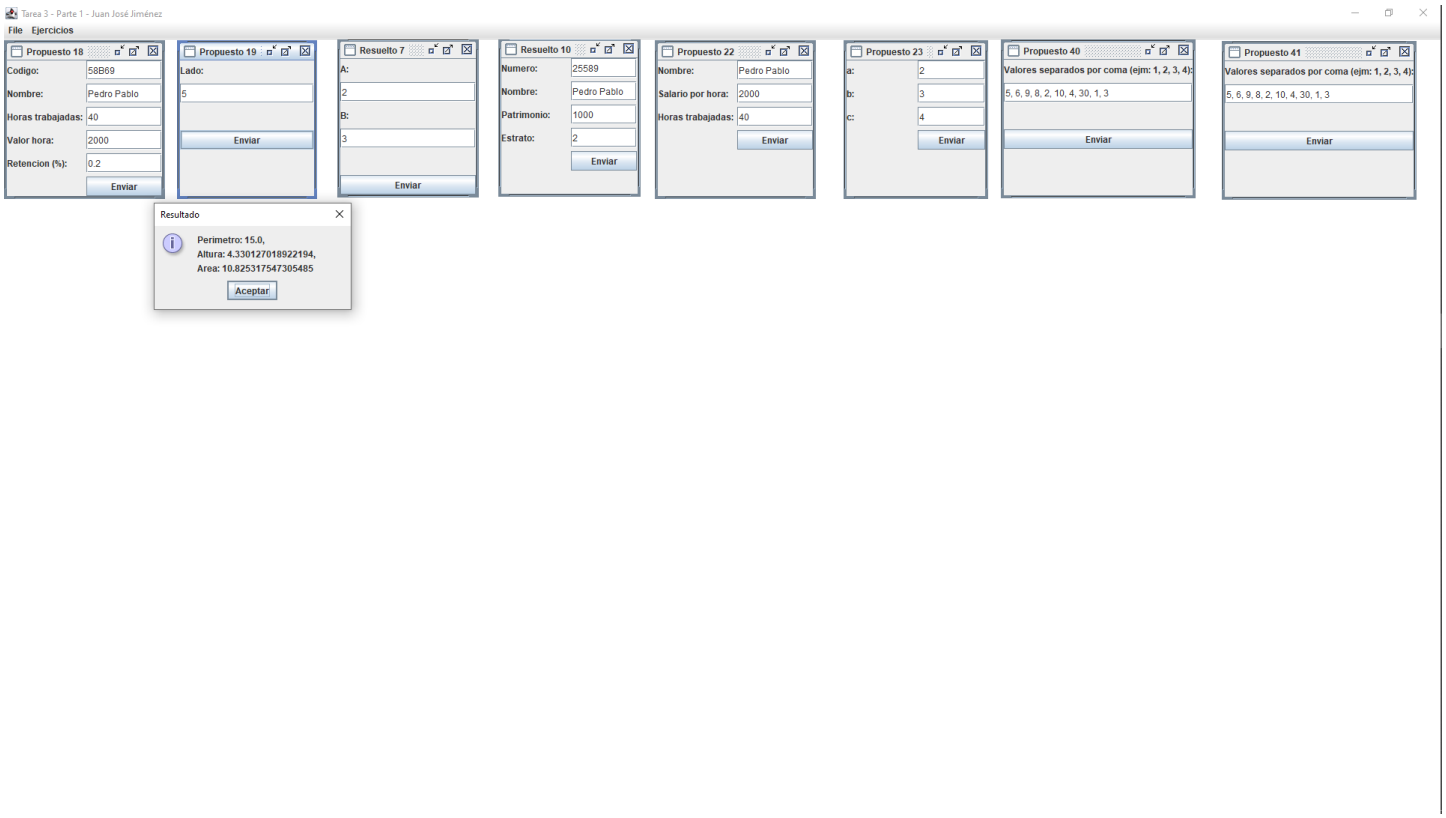
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto22.png

Código: Propuesto18.java

```
package Tareas.Tarea3.Parte1;

public class Propuesto18 extends MDIElement<Propuesto18.Empleado> {
    public Propuesto18() {
        super("Propuesto 18", new Argument<?>[]{
                new Argument<>("Codigo", String.class),
                new Argument<>("Nombre", String.class),
                new Argument<>("Horas trabajadas", int.class),
                new Argument<>("Valor hora", float.class),
                new Argument<>("Retencion (%)", float.class)
        }, Empleado.class);
    }

    public record Empleado(String codigo, String nombre, int horasTrabajadas, float valorHorasTrabajadas,
                           float porcentajeRetencionFuente) {

        public float salarioBruto() {
                return horasTrabajadas * valorHorasTrabajadas;
            }

            public float salarioNeto() {
                return salarioBruto() * (1 - porcentajeRetencionFuente);
            }

            @Override
            public String toString() {
                return String.format(
                        """
                                Código: %s
                                Nombre: %s
                                Salario bruto: %.2f
                                Salario neto: %.2f
                                """,
                        codigo, nombre, salarioBruto(), salarioNeto()
                );
            }
        }
}
```

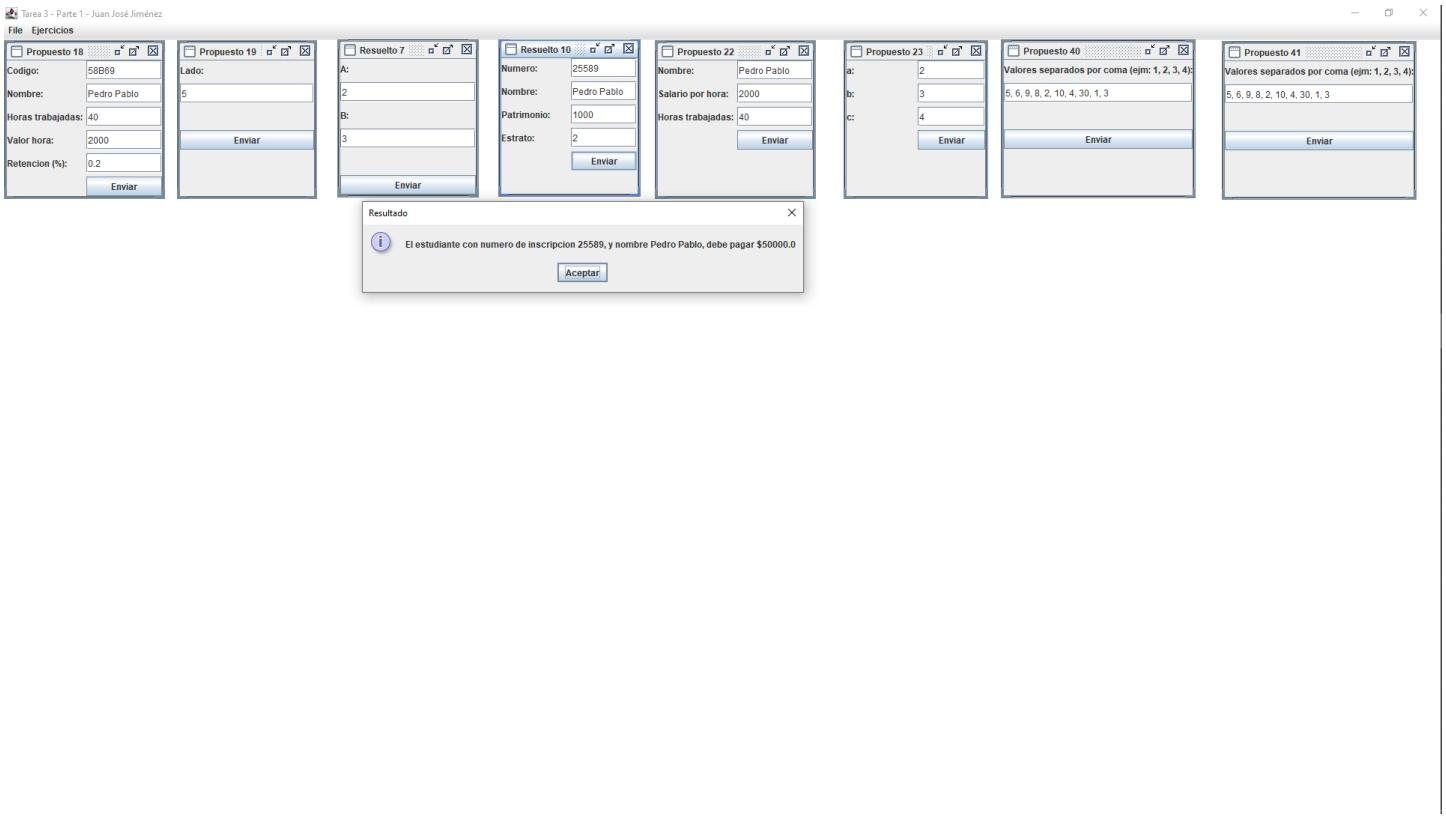
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto18.java

Imagen: Propuesto19.png



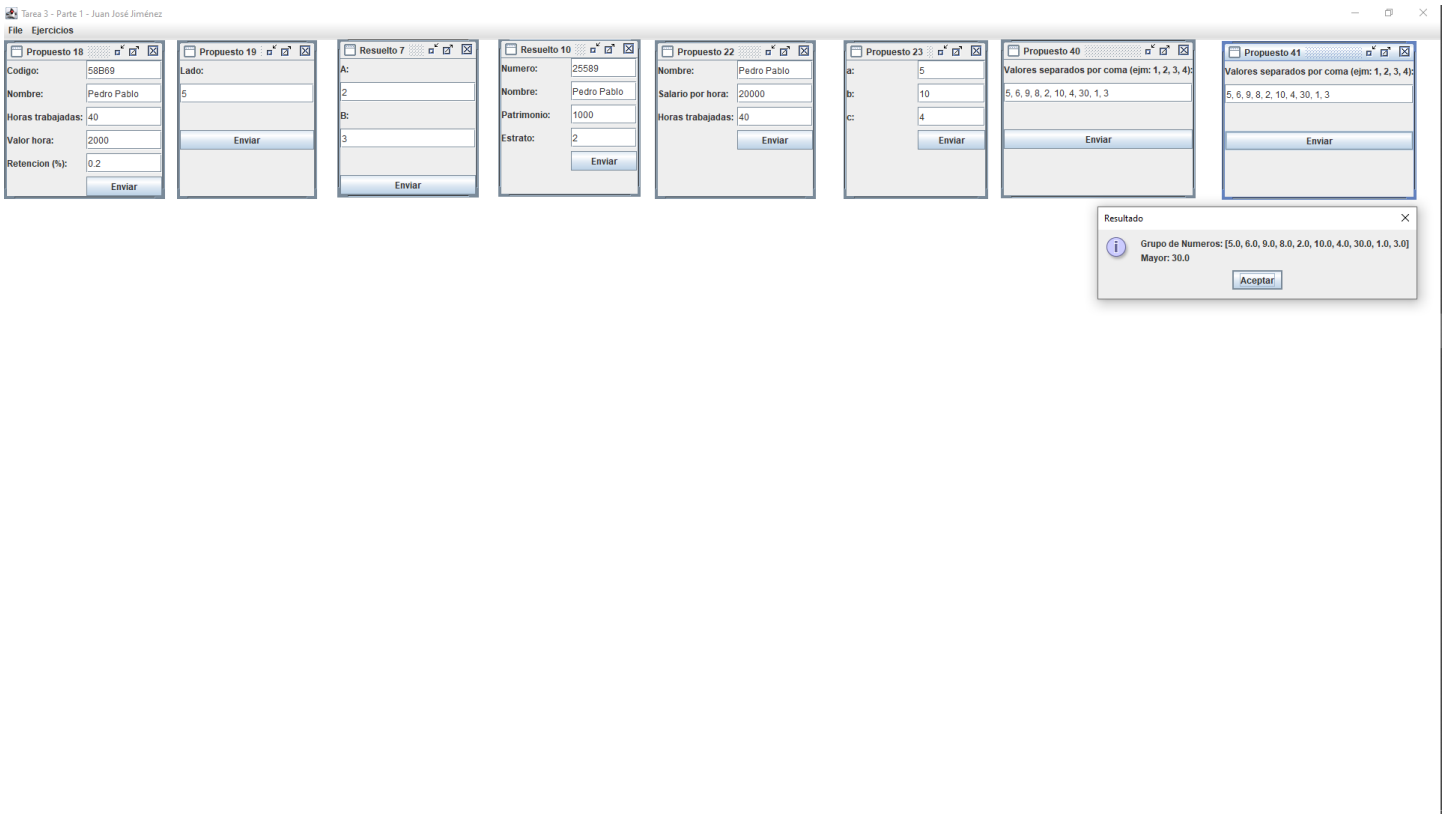
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto19.png

Imagen: Resuelto10.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Resuelto10.png

Imagen: Propuesto41.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto41.png

Código: Resuelto7.java

```
package Tareas.Tarea3.Parte1;

public class Resuelto7 extends MDIElement<Resuelto7.Comparador> {

    public Resuelto7() {
        super("Resuelto 7", new Argument<?>[] {
            new Argument<>("A", double.class),
            new Argument<>("B", double.class)
        }, Comparador.class);
    }

    public static class Comparador {
        double a, b;

        public Comparador(double a, double b) {
            this.a = a;
            this.b = b;
        }

        public int compare() {
            if (a > b) return 1;
            else if (a < b) return -1;
            return 0;
        }

        @Override
        public String toString() {
            return "%s es %s a %s".formatted(a, switch (compare()) {
                case 1 -> "mayor";
                case -1 -> "menor";
                default -> "igual";
            }, b);
        }
    }
}
```

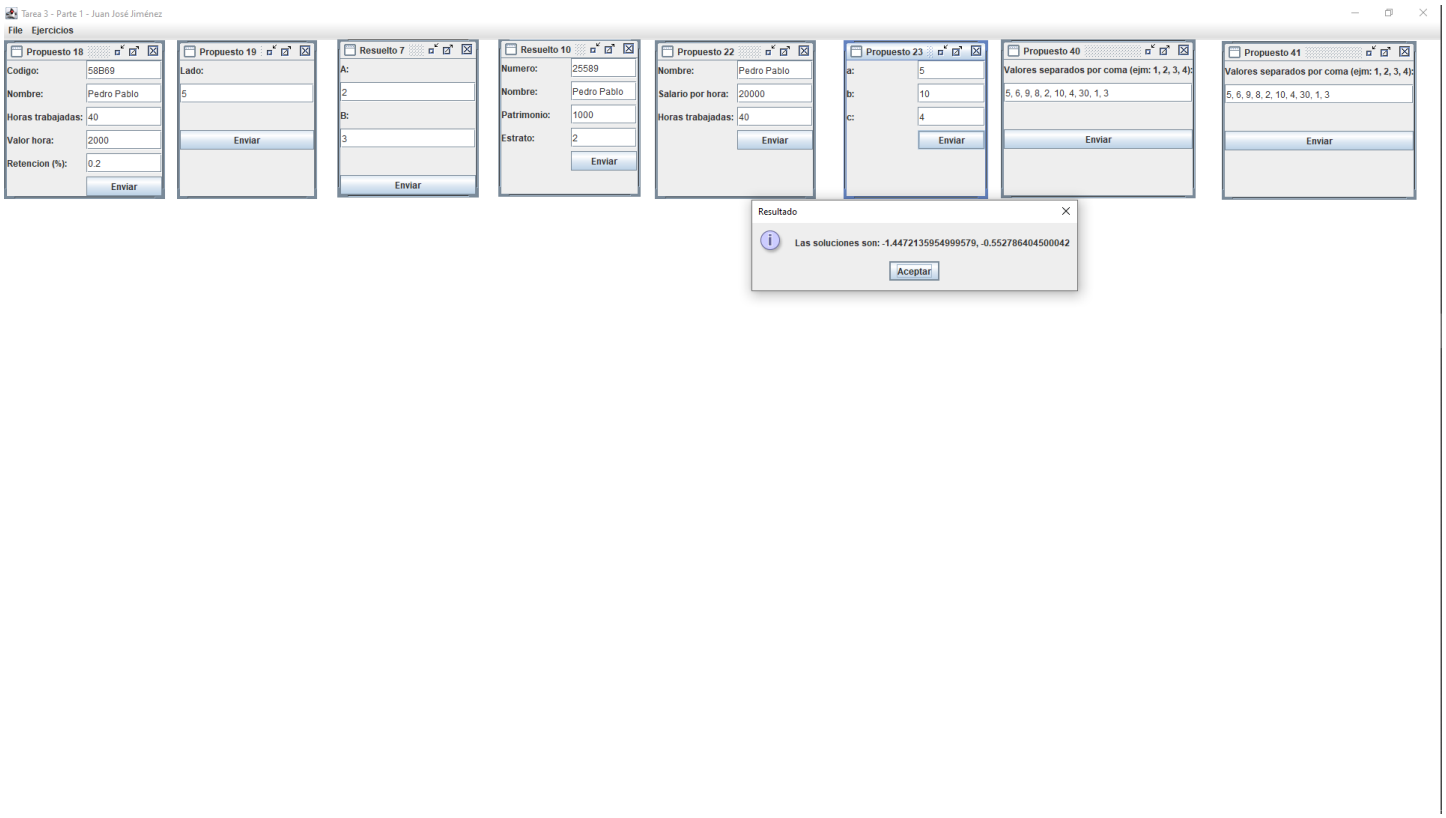
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Resuelto7.java

Código: Propuesto40.java

```
package Tareas.Tarea3.Parte1;

import java.util.Arrays;
import java.util.List;

public class Propuesto40 extends MDIElement<Propuesto40.GrupoDeNumeros> {
    public Propuesto40() {
        super("Propuesto 40", new Argument[]{
                new Argument<>("Valores separados por coma (ejm: 1, 2, 3, 4)", String.class)
        }, GrupoDeNumeros.class);
    }

    public static class GrupoDeNumeros {
        private final List<Double> valores;

        /**
         * Recibe una cadena de texto con los valores de los enteros separados por comas.
         * <p>Ejm: "1, 2, 3, 4, 5"
         * @param valores Cadena de texto con los valores de los enteros separados por comas.
         */
        public GrupoDeNumeros(String valores) {
            this(Arrays.stream(valores.replaceAll("\\s+", "").split(","))
                    .map(Double::parseDouble)
                    .filter(x -> x >= 0) // El enunciado dice que los valores son positivos
                    .toList());
        }

        public GrupoDeNumeros(List<Double> valores) {
            this.valores = valores;
        }

        public GrupoDeNumeros sqrt() {
            return new GrupoDeNumeros(valores.stream().map(Math::sqrt).toList());
        }

        public GrupoDeNumeros pow2() {
            return new GrupoDeNumeros(valores.stream().map(x -> x * x).toList());
        }

        public GrupoDeNumeros pow3() {
            return new GrupoDeNumeros(valores.stream().map(x -> x * x * x).toList());
        }

        @Override
        public String toString() {
            return """
                    Grupo de Numeros: %s
                    Raiz cuadrada: %s
                    Potencia al cuadrado: %s
                    Potencia al cubo: %s
                    """.formatted(valores, sqrt().valores, pow2().valores, pow3().valores);
        }
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto40.java

Imagen: Propuesto23.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto23.png

Código: MDIElement.java

```
package Tareas.Tarea3.Parte1;

import javax.swing.*;
import java.awt.*;
import java.beans.PropertyVetoException;
import java.lang.reflect.Constructor;
import java.util.Arrays;
import java.util.HashMap;

public abstract class MDIElement<T> {
    private final Argument<?>[] args;
    private final Class<T> targetClass;
    protected final HashMap<String, JTextField> labels = new HashMap<>();

    public final String title;

    /**
     * Crea un nuevo elemento de interfaz de usuario con argumentos y clase objetivo.
     * @param title Titulo de la ventana
     * @param args Argumentos necesarios para crear la instancia de la clase objetivo.
     * @param targetClass Clase objetivo a instanciar.
     */
    public MDIElement(String title, Argument<?>[] args, Class<T> targetClass) {
        this.title = title;
        this.args = args;
        this.targetClass = targetClass;
    }

    public Argument<?>[] getArgs() {
        return args;
    }

    /**
     * Crea un JInternalFrame con el formulario para los argumentos.
     * @param parent Componente padre, usado para enviar mensajes de error.
     * @return El JInternalFrame creado.
     */
    public JInternalFrame getFrame(Component parent) {
        JInternalFrame frame = new JInternalFrame(title, true, true, true, true);
        frame.add(getPanel(parent));
        frame.pack();
        frame.setVisible(true);
        try {
            frame.setSelected(true);
        } catch (PropertyVetoException e) {
            throw new RuntimeException(e);
        }
        return frame;
    }

    /**
     * Crea un panel con los campos de texto para los argumentos.
     * @param parent Componente padre, usado para enviar mensajes de error.
     * @return El panel creado.
     */
    protected JPanel getPanel(Component parent) {
        JPanel panel = new JPanel(new GridLayout(6, 2, 5, 5));

        // Entrada para cada argumento
        for (Argument<?> arg : this.args) {
            JLabel lbl = new JLabel(arg.name + ":");
            JTextField txt = new JTextField();
```

```
            labels.put(arg.name, txt);
            panel.add(lbl);
            panel.add(txt);
        }

        // Botón de enviar
        panel.add(new JLabel()); // Espaciador
        JButton btnSend = new JButton("Enviar");
        btnSend.addActionListener(e -> this.onSend(parent));
        panel.add(btnSend);

        return panel;
    }

    public void onSend(Component parent) {
        T instance = getInstanceViaArgs(parent);
        if (instance == null) {
            return;
        }
        JOptionPane.showMessageDialog(
                parent,
                    instance.toString(),
                "Resultado",
                JOptionPane.INFORMATION_MESSAGE);
    }


    /**
     * Basicamente este es el método más importante de esta clase
     * Se encarga de crear una instancia de la clase objetivo a partir de los argumentos
     * almacenados previamente en el HashMap labels.
     * A partir del tipo del argumento, se encuentra el constructor adecuado y se crea la instancia.
     * @param parent Componente padre, usado para enviar mensajes de error.
     * @return La instancia creada, o null si hubo un error.
     */
    public T getInstanceViaArgs(Component parent) {
        try {
            Class<?>[] parameterTypes = Arrays.stream(args)
                    .map(arg -> arg.type)
                    .toArray(Class<?>[]::new);

            Constructor<T> constructor = targetClass.getConstructor(parameterTypes);

            Object[] parameterValues = Arrays.stream(args)
                    .map(arg -> {
                        JTextField textField = labels.get(arg.name);
                        if (textField == null) {
                            throw new IllegalArgumentException("Error en la creación del formulario, no se creó
para: " + arg.name);
                        }
                        String text = textField.getText();

                        if (text == null) {
                            throw new IllegalArgumentException("No se puede dejar en blanco: " + arg.name);
                        }
                        return parseValue(text, arg.type);
                    })
                    .toArray();
            return constructor.newInstance(parameterValues);

        } catch (NoSuchMethodException ex) {
                JOptionPane.showMessageDialog(parent, "No hay constructor para la clase objetivo.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
        catch (Exception ex) {
             JOptionPane.showMessageDialog(parent, "Error al crear la instancia: " + ex.getMessage(), "Error",
```

```
JOptionPane.ERROR_MESSAGE);
        }

        return null;
    }

    /**
     * Convierte un string a el tipo especificado.
     *
     * @throws IllegalArgumentException Si el tipo no es soportado.
     * @param value La cadena a convertir.
     * @param type  El tipo al que se convertirá.
     * @return El valor convertido.
     */
    private Object parseValue(String value, Class<?> type) {
        if (type == String.class) {
            return value;
        } else if (type == int.class || type == Integer.class) {
            return Integer.parseInt(value);
        } else if (type == double.class || type == Double.class) {
            return Double.parseDouble(value);
        } else if (type == boolean.class || type == Boolean.class) {
            return Boolean.parseBoolean(value);
        } else if (type == float.class || type == Float.class) {
            return Float.parseFloat(value);
        } else if (type == long.class || type == Long.class) {
            return Long.parseLong(value);
        } else {
            try {
                return type.cast(value);
            } catch (Exception e) {
                throw new IllegalArgumentException("Unsupported argument type: " + type);
            }
        }
    }

    public record Argument<T>(String name, Class<T> type) { }
}
```

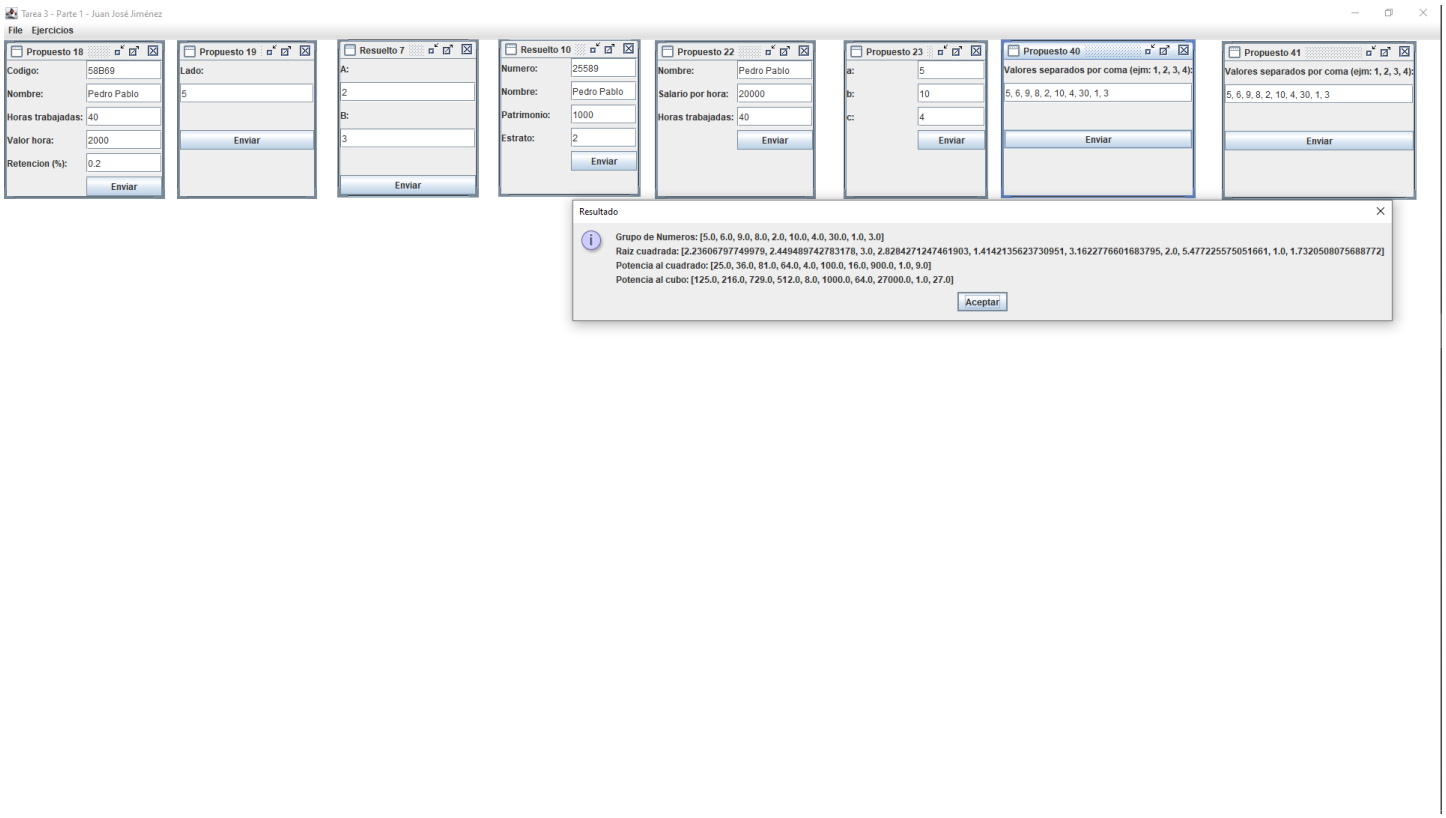
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/MDIElement.java

Código: Resuelto10.java

```
package Tareas.Tarea3.Parte1;

public class Resuelto10 extends MDIElement<Resuelto10.Estudiante> {
    public Resuelto10() {
        super("Resuelto 10", new Argument[]{
                new Argument<>("Numero", long.class),
                new Argument<>("Nombre", String.class),
                new Argument<>("Patrimonio", double.class),
                new Argument<>("Estrato", int.class)
        }, Estudiante.class);
    }

    public static class Estudiante {
        public final long numero;
        public final String nombre;
        public double patrimonio;
        public int estrato;
        public Estudiante(long numero, String nombre, double patrimonio, int estrato) {
            this.numero = numero;
            this.nombre = nombre;
            this.patrimonio = patrimonio;
            this.estrato = estrato;
        }

        public static final double VALOR_BASICO = 50000;

        public double valorMatricula() {
            return VALOR_BASICO + (patrimonio > 2000000 && estrato > 3 ? patrimonio * 0.03 : 0);
        }

        @Override
        public String toString() {
              return String.format("El estudiante con numero de inscripcion %s, y nombre %s, debe pagar $%s",
numero, nombre, valorMatricula());
        }
    }
}
```

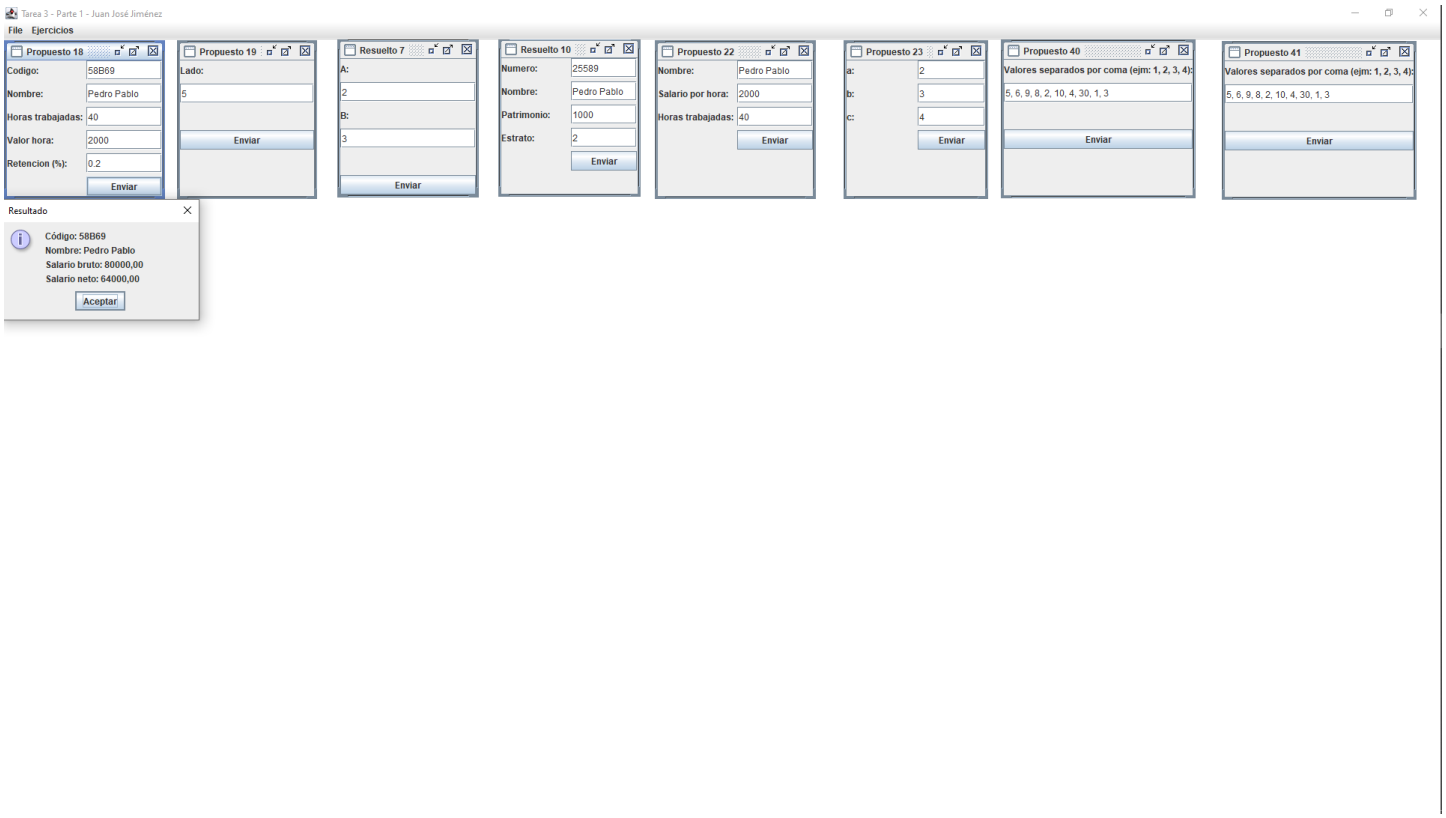
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Resuelto10.java

Imagen: Propuesto40.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto40.png

Imagen: Propuesto18.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte1/Propuesto18.png