Universidad Nacional de Colombia

Tarea3 - Parte2

Autores:

Juan José Jiménez Maya

Programa: Programación Orientada a Objetos

Grupo: 3

Código: Rombo.java

```
package Tareas.Tarea3.Parte2;

public class Rombo extends Cuadrado {
    public Rombo(double lado) {
        super(lado);
    }

    @Override
    public String toString() {
        return """
                Rombo con:
                - Lado: %.2f
                - Area: %.2f
                - Perimetro: %.2f
                """.formatted(altura, area(), perimetro());
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Rombo.java

Código: MDIElement.java

```
package Tareas.Tarea3.Parte2;

import javax.swing.*;
import java.awt.*;
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);
        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/Parte2/MDIElement.java

Código: Figura.java

```
package Tareas.Tarea3.Parte2;

public interface Figura {
    double area();
    double perimetro();
}
```

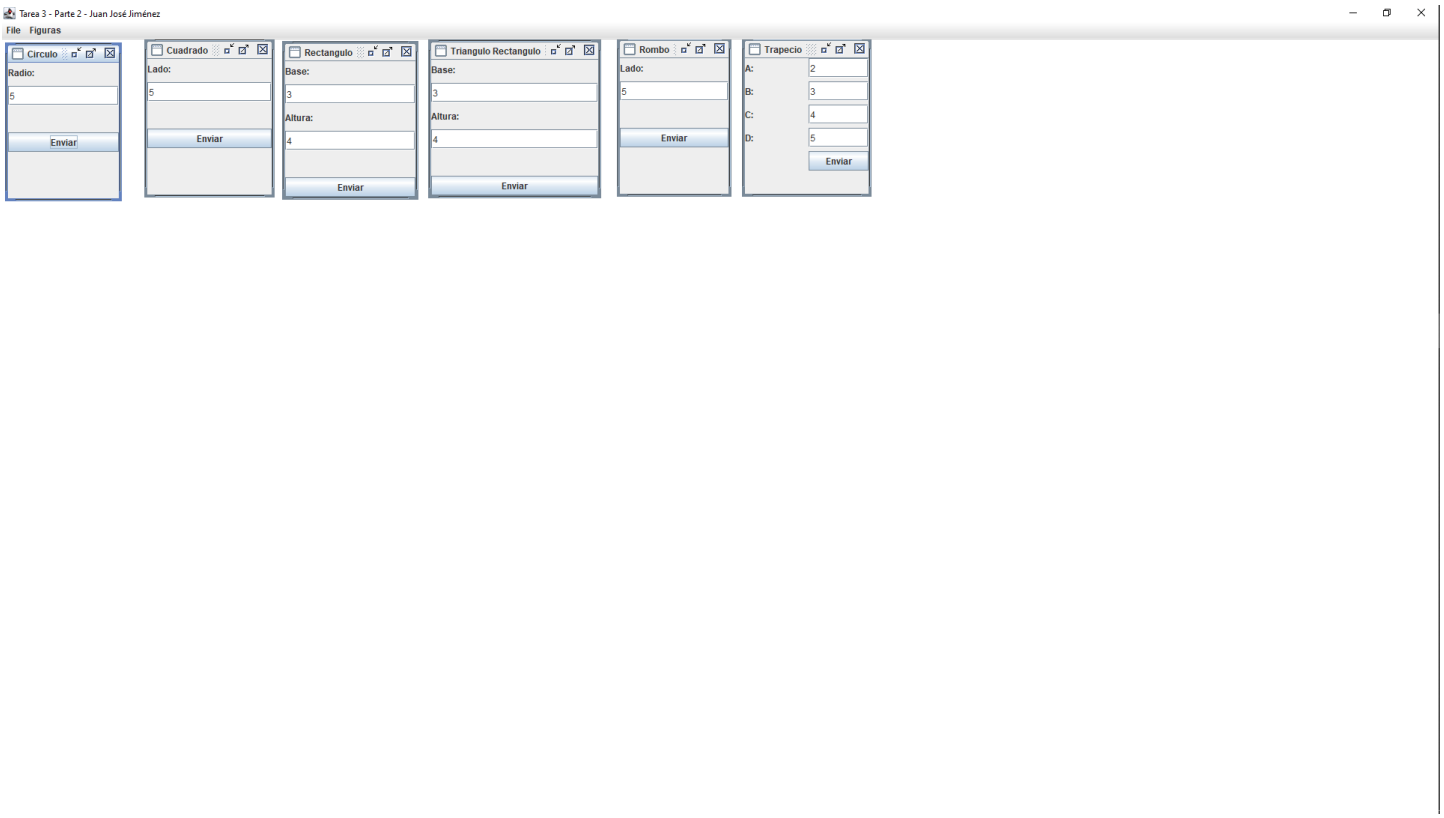
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Figura.java

Código: Rectangulo.java

```
package Tareas.Tarea3.Parte2;

public class Rectangulo implements Figura {
    protected final double base;
    protected final double altura;

    public Rectangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }

    @Override
    public double area() {
        return base * altura;
    }

    @Override
    public double perimetro() {
        return 2 * (base + altura);
    }

    @Override
    public String toString() {
        return """
                Rectangulo con:
                - Base: %.2f
                - Altura: %.2f
                - Area: %.2f
                - Perimetro: %.2f
                """.formatted(base, altura, area(), perimetro());
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Rectangulo.java

Imagen: Interfaz.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Interfaz.png

Código: Cuadrado.java

```
package Tareas.Tarea3.Parte2;

public class Cuadrado extends Rectangulo {
    public Cuadrado(double lado) {
        super(lado, lado);
    }

    @Override
    public String toString() {
        return """
                Cuadrado con:
                - Lado: %.2f
                - Area: %.2f
                - Perimetro: %.2f
                """.formatted(base, area(), perimetro());
    }
}
```

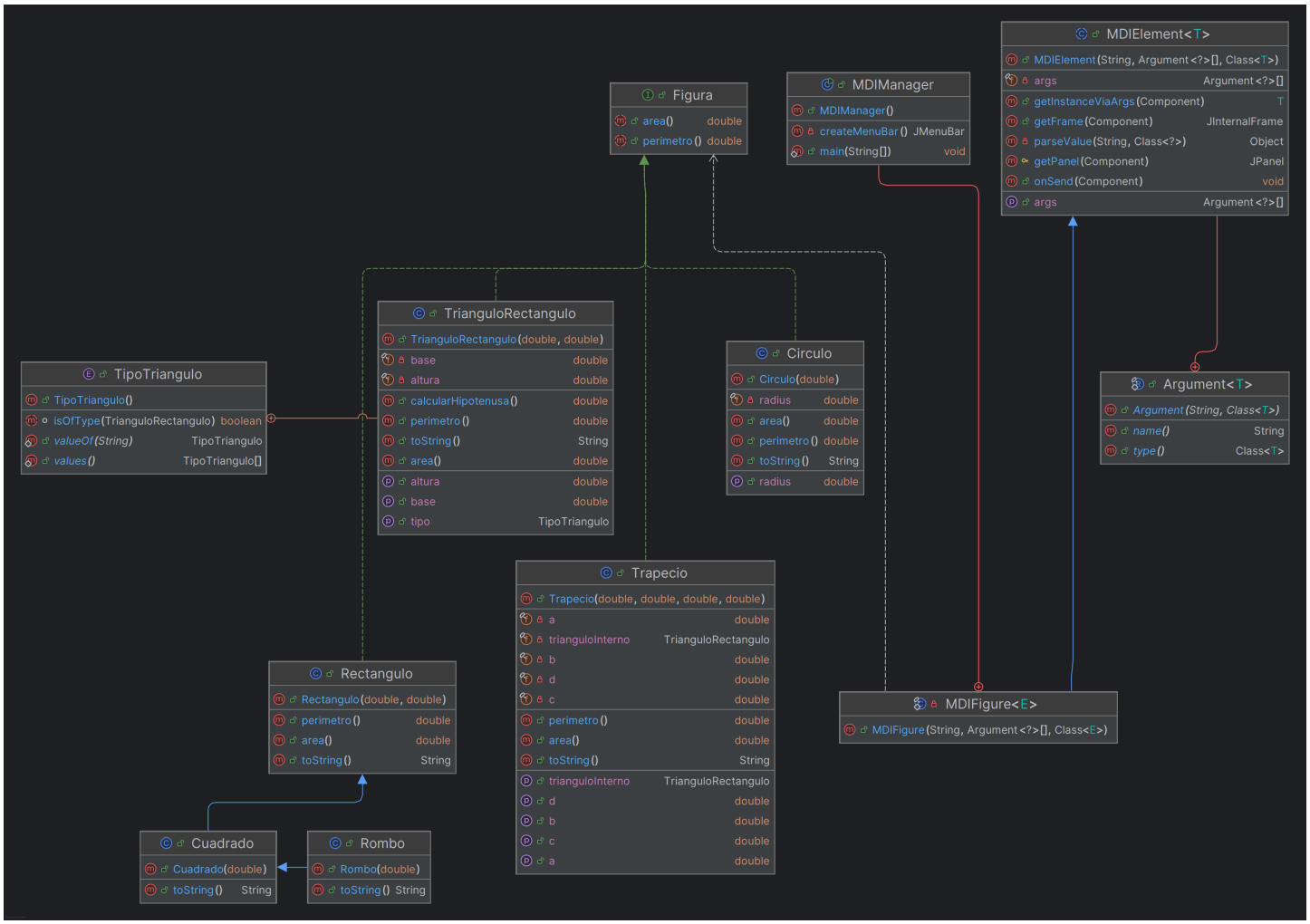
Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Cuadrado.java

Código: Circulo.java

```
package Tareas.Tarea3.Parte2;

public class Circulo implements Figura {
    private final double radius;

    public Circulo(double radius) {
        this.radius = radius;
    }

    public double getRadius() {
        return radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }

    @Override
    public double perimetro() {
        return 2 * Math.PI * radius;
    }

    @Override
    public String toString() {
        return """
                Circulo:
                Radio: %.2f
                Area: %.2f
                Perimetro: %.2f
                """.formatted(radius, area(), perimetro());
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Circulo.java

Imagen: DiagramaUML.png



Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/DiagramaUML.png

Código: TrianguloRectangulo.java

```
package Tareas.Tarea3.Parte2;

public class TrianguloRectangulo implements Figura {

    private final double base;
    private final double altura;
    private final double hipotenusa;
    public TrianguloRectangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
        this.hipotenusa = Math.pow(base*base + altura*altura, 0.5);
    }

    public double getBase() {
        return base;
    }

    public double getAltura() {
        return altura;
    }

    public double calcularHipotenusa() {
        return hipotenusa;
    }

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

    @Override
    public double perimetro() {
        return base + altura + hipotenusa;
    }

    public TipoTriangulo getTipo() {
        for (TipoTriangulo value : TipoTriangulo.values()) {
            if (value.isOfType(this)) return value;
        }
        throw new IllegalStateException();
    }

    @Override
    public String toString() {
        return """
                Triangulo Rectangulo con:
                |Base: %.2f
                |Altura: %.2f
                |Hipotenusa: %.2f
                |Area: %.2f
                |Perimetro: %.2f
                |Tipo: %s
                """.formatted(base, altura, hipotenusa, area(), perimetro(), getTipo());
    }

    public enum TipoTriangulo {
        EQUILATERO {
            @Override
            boolean isOfType(TrianguloRectangulo triangulo) {
                return triangulo.base == triangulo.altura && triangulo.base == triangulo.hipotenusa;
            }
        },
```

```
        ISOCELES {
            @Override
            boolean isOfType(TrianguloRectangulo triangulo) {
                return !EQUILATERO.isOfType(triangulo) && !ESCALENO.isOfType(triangulo);
            }
        },
        ESCALENO {
            @Override
            boolean isOfType(TrianguloRectangulo triangulo) {
                return triangulo.base != triangulo.altura && triangulo.base != triangulo.hipotenusa;
            }
        };

        abstract boolean isOfType(TrianguloRectangulo triangulo);
    }
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/TrianguloRectangulo.java

Código: Trapecio.java

```
package Tareas.Tarea3.Parte2;

public class Trapecio implements Figura {

    private final double a;
    private final double b;
    private final double c;
    private final double d;

    private final TrianguloRectangulo trianguloInterno;
    public Trapecio(double a, double b, double c, double d) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.d = d;

        double n = ((d - b) / 2);
        this.trianguloInterno = new TrianguloRectangulo(n, Math.sqrt((a * a) - (n * n)));
    }

    public double getA() {
        return a;
    }

    public double getB() {
        return b;
    }

    public double getC() {
        return c;
    }

    public double getD() {
        return d;
    }

    public TrianguloRectangulo getTrianguloInterno() {
        return trianguloInterno;
    }

    @Override
    public double area() {
        return (trianguloInterno.getAltura() / 2) * (b + d);
    }

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

    @Override
    public String toString() {
        return """
                Trapecio con:
                A = %.2f
                B = %.2f
                C = %.2f
                D = %.2f
                Área = %.2f
                Perímetro = %.2f
                """.formatted(a, b, c, d, area(), perimetro());
    }
```

```
}
```

Enlace: https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea3/Parte2/Trapecio.java

Código: MDIManager.java

```
package Tareas.Tarea3.Parte2;

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

public class MDIManager extends JFrame {

    private static final class MDIFigure<E extends Figura> extends MDIElement<E> {
        public MDIFigure(String title, Argument<?>[] args, Class<E> targetClass) {
            super(title, args, targetClass);
        }
    }

    public static final Collection<MDIElement<?>> components = Arrays.asList(
            new MDIFigure<>("Circulo", new MDIElement.Argument<?>[] {
                    new MDIElement.Argument<>("Radio", double.class)
            }, Circulo.class),
            new MDIFigure<>("Cuadrado", new MDIElement.Argument<?>[] {
                    new MDIElement.Argument<>("Lado", double.class)
            }, Cuadrado.class),
            new MDIFigure<>("Rectangulo", new MDIElement.Argument<?>[] {
                    new MDIElement.Argument<>("Base", double.class),
                    new MDIElement.Argument<>("Altura", double.class)
            }, Rectangulo.class),
            new MDIFigure<>("Triangulo Rectangulo", new MDIElement.Argument<?>[] {
                    new MDIElement.Argument<>("Base", double.class),
                    new MDIElement.Argument<>("Altura", double.class)
            }, TrianguloRectangulo.class),
            new MDIFigure<>("Rombo", new MDIElement.Argument<?>[] {
                    new MDIElement.Argument<>("Lado", double.class)
            }, Rombo.class),
            new MDIFigure<>("Trapecio", new MDIElement.Argument<?>[] {
                    new MDIElement.Argument<>("A", double.class),
                    new MDIElement.Argument<>("B", double.class),
                    new MDIElement.Argument<>("C", double.class),
                    new MDIElement.Argument<>("D", double.class)
            }, Trapecio.class)
    );

    private final JDesktopPane desktopPane;

    public MDIManager() {
        // Configure the main window
        setTitle("Tarea 3 - Parte 2 - 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("Figuras");
        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/Parte2/MDIManager.java