Trabajo 5 POO (Grupal)

Johan Sebastián Robles Rincón

Juan Sebastián Duran Roldan

José Manuel Castro Balvin

Estudiantes de Ingeniería de Sistemas E Informática

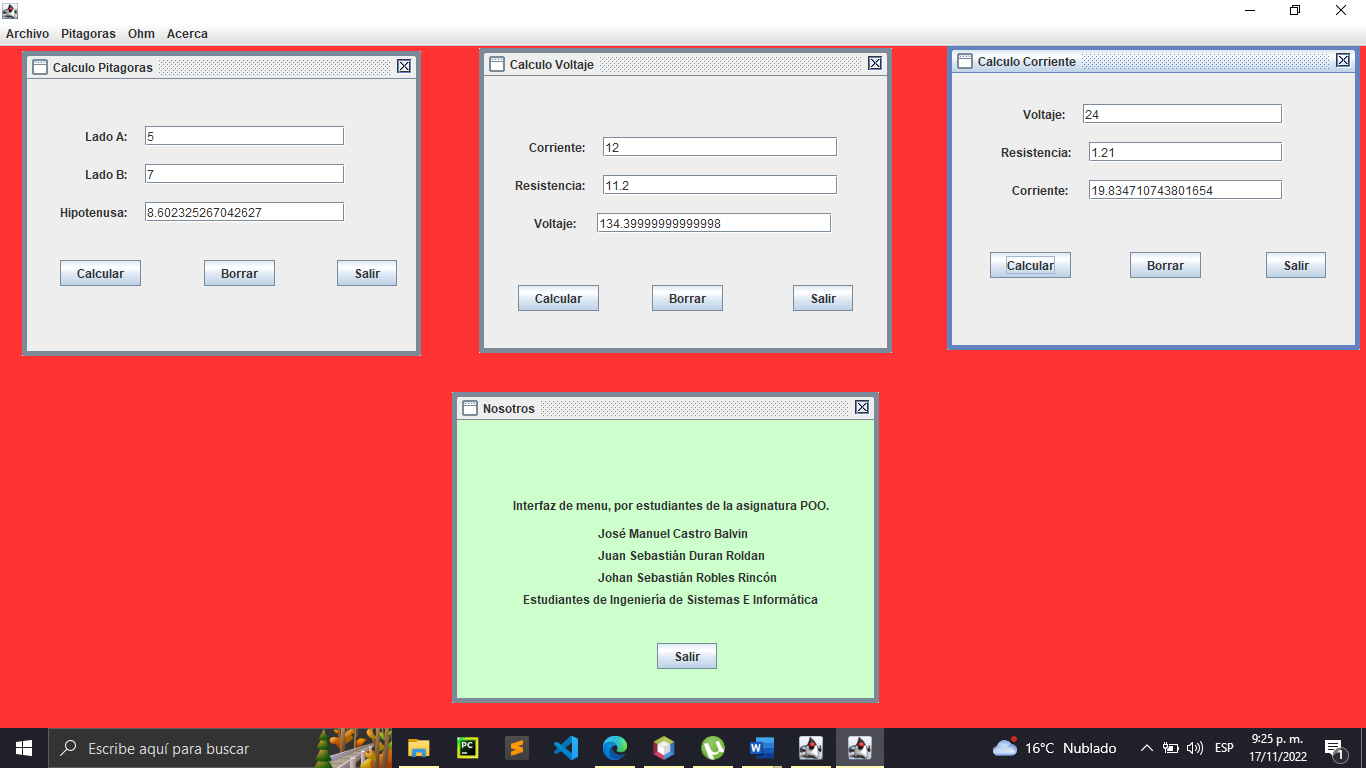
Profesor Walter Hugo Arboleda Mazo

Universidad Nacional De Colombia Sede Medellín

17/11/2022

**1.Ejercicios de menús realizado en el video del sábado 29 de octubre**

**Funcionamiento:**



**Códigos (más importantes)**

* **Menú**

package menus;

import javax.swing.JFrame;

/\*\*

\*

\* @author Johan

\*/

public class Menus {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

PantallaPrincipal formulario = new PantallaPrincipal();

formulario.setExtendedState(JFrame.MAXIMIZED\_BOTH);

formulario.setVisible(true);

}

}

* **Lógica Pitágoras**

package menus;

public class LogicaPitagoras {

public static Double CalcularHipotenusa(Double CatetoOpuesto, Double CatetoAdyacente){

Double hipotenusa = Math.sqrt(Math.pow(CatetoOpuesto, 2)+Math.pow(CatetoAdyacente, 2));

return hipotenusa;

}

}

* **Lógica Corriente**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package menus;

/\*\*

\*

\* @author Johan

\*/

public class Corriente {

public static Double CalcularCorriente(Double Voltaje, Double Resistencia){

Double corriente = Voltaje / Resistencia;

return corriente;

}

}

* **Lógica Voltaje**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package menus;

/\*\*

\*

\* @author Johan

\*/

public class Voltaje {

public static Double CalcularVoltaje(Double Corriente, Double Resistencia){

Double voltaje = Corriente \* Resistencia;

return voltaje;

}

}

Nota: los JFrame y los Internal JFrame son bastante similares los de la clase, no veo necesario colocarlo, además para no alarga tanto este documento. Igual, se pueden ver en el repositorio GitHub.

**2.Ejercicio 8.2 pagina 483 (Texto Ejercicios de programación)**

**Interfaz de usuario:**

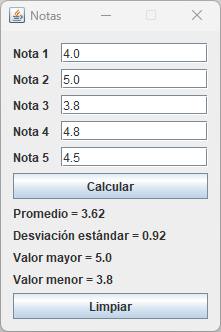
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se hacen 2 casos y los resultados son:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Código:**

Clase Notas:

package com.mycompany.notas;

public class Notas {

double[] listaNotas;

public Notas() {

listaNotas = new double[5];

}

double calcularPromedio() {

double suma = 0;

for(int i=1; i < listaNotas.length; i++) {

suma = suma + listaNotas[i];

}

return (suma / listaNotas.length);

}

double calcularDesviación() {

double prom = calcularPromedio();

double suma = 0;

for(int i=0; i < listaNotas.length; i++) {

suma += Math.pow(listaNotas[i] - prom, 2 );

}

return Math.sqrt (suma/listaNotas.length );

}

double calcularMenor() {

double menor = listaNotas[0];

for(int i=0; i < listaNotas.length; i++) {

if (listaNotas[i] < menor) {

menor = listaNotas[i];

}

}

return menor;

}

double calcularMayor() {

double mayor = listaNotas[0];

for(int i=0; i < listaNotas.length; i++) {

if (listaNotas[i] > mayor) {

mayor = listaNotas[i];

}

}

return mayor;

}

}

Clase Principal:

package com.mycompany.notas;

public class Principal {

public static void main(String[] args) {

VentanaPrincipal miVentanaPrincipal;

miVentanaPrincipal= new VentanaPrincipal();

miVentanaPrincipal.setVisible(true);

}

}

Clase VentanaPrincipal, No se agrega código generado automáticamente por JavaSwing, para código totalmente funcional revisar el enlace de GitHub:

package com.mycompany.notas;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

public class VentanaPrincipal extends javax.swing.JFrame {

public VentanaPrincipal() {

initComponents();

setTitle("Notas");

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setResizable(false);

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try{

Notas notas = new Notas();

notas.listaNotas[0] = Double.parseDouble(txtN1.getText());

notas.listaNotas[1] = Double.parseDouble(txtN2.getText());

notas.listaNotas[2] = Double.parseDouble(txtN3.getText());

notas.listaNotas[3] = Double.parseDouble(txtN4.getText());

notas.listaNotas[4] = Double.parseDouble(txtN5.getText());

notas.calcularPromedio();

notas.calcularDesviación();

labelPromedio.setText("Promedio = " + String.valueOf(String.format("%.2f",notas.calcularPromedio())));

double desv = notas.calcularDesviación();

labelDesviacion.setText("Desviación estándar = " + String.format("%.2f", desv));

labelMayor.setText("Valor mayor = " + String.valueOf(notas.calcularMayor()));

labelMenor.setText("Valor menor = " + String.valueOf(notas.calcularMenor()));

}

catch(NumberFormatException ex){

JOptionPane.showMessageDialog(null, "El valor ingresado es incorrecto.");

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

txtN1.setText("");

txtN2.setText("");

txtN3.setText("");

txtN4.setText("");

txtN5.setText("");

labelPromedio.setText("Promedio = ");

labelDesviacion.setText("Desviación estándar = ");

labelMayor.setText("Valor mayor = ");

labelMenor.setText("Valor menor = ");

}

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new VentanaPrincipal().setVisible(true);

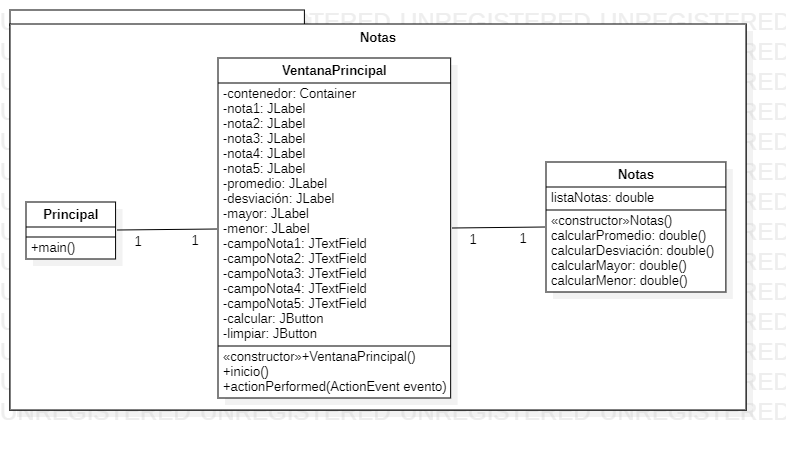
}

});

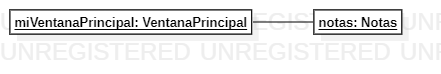
}

}

**Diagrama de clases:**

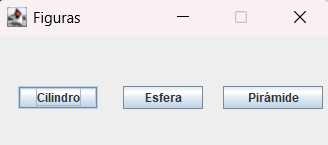


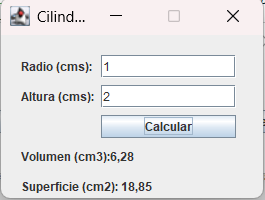
**Diagrama de objetos:**

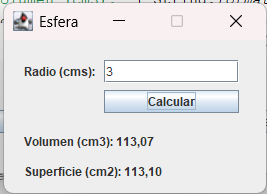


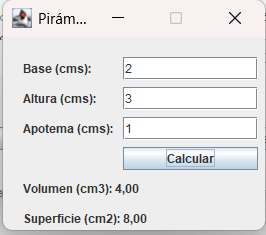
**3. Ejercicio 8.3 pagina 483 (Texto Ejercicios de programación)**

**Interfaz de usuario:**









**Clase Cilindro:**

package com.mycompany.figurageometrica;

/\*\*

\* Esta clase denominada Cilindro es una subclase de FiguraGeométrica

\* que cuenta con un radio y una altura.

\* @version 1.2/2020

\*/

public class Cilindro extends FiguraGeométrica {

private double radio; // Atributo que establece el radio de un cilindro

private double altura; // Atributo que establece la altura de un cilindro

/\*\*

\* Constructor de la clase Cilindro

\* @param radio Parámetro de define el radio de un cilindro

\* @param altura Parámetro de define la altura de un cilindro

\*/

public Cilindro(double radio, double altura) {

this.radio = radio;

this.altura = altura;

this.setVolumen(calcularVolumen()); /\* Calcula el volumen y

establece su atributo \*/

this.setSuperficie(calcularSuperficie()); /\* Calcula la superficie y

establece su atributo \*/

}

/\*\*

\* Método para calcular el volumen de un cilindro

\* @return El volumen de un cilindro

\*/

public double calcularVolumen() {

double volumen = Math.PI \* altura \* Math.pow(radio, 2.0);

return volumen;

}

/\*\*

\* Método para calcular la superficie de un cilindro

\* @return La superficie de un cilindro

\*/

public double calcularSuperficie() {

double áreaLadoA = 2.0 \* Math.PI \* radio \* altura;

double áreaLadoB = 2.0 \* Math.PI \* Math.pow(radio, 2.0);

return áreaLadoA + áreaLadoB;

}

}

**Clase Esfera:**

package com.mycompany.figurageometrica;

/\*\*

\* Esta clase denominada Esfera es una subclase de FiguraGeométrica

\* que cuenta con un radio.

\* @version 1.2/2020

\*/

public class Esfera extends FiguraGeométrica {

private double radio; // Atributo que identifica el radio de una esfera

/\*\*

\* Constructor de la clase Esfera

\* @param radio Parámetro de define el radio de una esfera

\* \*/

public Esfera(double radio) {

this.radio = radio;

this.setVolumen(calcularVolumen()); /\* Calcula el volumen y

establece su atributo \*/

this.setSuperficie(calcularSuperficie()); /\* Calcula la superficie y

establece su atributo \*/

}

/\*\*

\* Método para calcular el volumen de una esfera

\* @return El volumen de una esfera

\*/

public double calcularVolumen() {

double volumen = 1.333 \* Math.PI \* Math.pow(this.radio, 3.0);

return volumen;

}

/\*\*

\* Método para calcular la superficie de una esfera

\* @return La superficie de una esfera

\*/

public double calcularSuperficie() {

double superficie = 4.0 \* Math.PI \* Math.pow(this.radio, 2.0);

return superficie;

}

}

**Clase FiguraGeometrica:**

package com.mycompany.figurageometrica;

/\*\*

\* Esta clase denominada FiguraGeométrica modela un figura

\* geométrica que cuenta con un volumen y una superficie a ser

\* calculados de acuerdo al tipo de figura geométrica.

\* @version 1.2/2020

\*/

public class FiguraGeométrica {

private double volumen; /\* Atributo que identifica el volumen de

una figura geométrica \*/

private double superficie; /\* Atributo que identifica la superficie de

una figura geométrica \*/

/\*\*

\* Método para establecer el volumen de una figura geométrica

\* @param volumen Parámetro que define el volumen de una figura

\* geométrica

\*/

public void setVolumen(double volumen) {

this.volumen = volumen;

}

/\*\*

\* Método para establecer la superficie de una figura geométrica

\* @param superficie Parámetro que define la superficie de una

\* figura geométrica

\*/

public void setSuperficie(double superficie) {

this.superficie = superficie;

}

/\*\*

\* Método para obtener el volumen de una figura geométrica

\* @return El volumen de una figura geométrica

\*/

public double getVolumen() {

return this.volumen;

}

/\*\*

\* Método para obtener la superficie de una figura geométrica

\* @return La superficie de una figura geométrica

\*/

public double getSuperficie() {

return this.superficie;

}

}

**Clase Piramide:**

package com.mycompany.figurageometrica;

/\*\*

\* Esta clase denominada Pirámide es una subclase de FiguraGeométrica

\* que cuenta con una base, una altura y un apotema.

\* @version 1.2/2020

\*/

public class Piramide extends FiguraGeométrica {

private double base; /\* Atributo que identifica la base de una

pirámide \*/

private double altura; /\* Atributo que identifica la altura de una

pirámide \*/

private double apotema; /\* Atributo que identifica el apotema de

una pirámide \*/

/\*\*

\* Constructor de la clase Pirámide

\* @param base Parámetro de define la base de una pirámide

\* @param altura Parámetro de define la altura de una pirámide

\* @param apotema Parámetro de define el apotema de una pirámide

\*/

public Piramide(double base, double altura, double apotema) {

this.base = base;

this.altura = altura;

this.apotema = apotema;

this.setVolumen(calcularVolumen()); /\* Calcula el volumen y

establece su atributo \*/

this.setSuperficie(calcularSuperficie()); /\* Calcula la superficie y

establece su atributo \*/

}

/\*\*

\* Método para calcular el volumen de una pirámide

\* @return El volumen de una pirámide

\*/

public double calcularVolumen() {

double volumen = (Math.pow(base, 2.0) \* altura) / 3.0;

return volumen;

}

/\*\*

\* Método para calcular la superficie de una pirámide

\* @return La superficie de una pirámide

\*/

public double calcularSuperficie() {

double áreaBase = Math.pow(base, 2.0);

double áreaLado = 2.0 \* base \* apotema;

return áreaBase + áreaLado;

}

}

**Clase Principal:**

package com.mycompany.figurageometrica;

/\*\*

\* Esta clase define el punto de ingreso al programa de figuras

\* geométricas. Por lo tanto, cuenta con un método main de acceso al

\* programa.

\* @version 1.2/2020

\*/

public class Principal {

/\*\*

\* Método main que sirve de punto de entrada al programa

\*/

public static void main(String[] args) {

VentanaPrincipal miVentanaPrincipal; /\* Define la ventana

principal \*/

miVentanaPrincipal= new VentanaPrincipal(); /\* Crea la ventana

principal \*/

miVentanaPrincipal.setVisible(true); /\* Establece la ventana

como visible \*/

// Establece que la ventana no puede cambiar su tamaño

miVentanaPrincipal.setResizable(false);

}

}

**Clase VentanaCilindro:**

package com.mycompany.figurageometrica;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Esta clase denominada VentanaCilindro define una ventana para

\* ingresar los datos de un cilindro y calcular su volumen y superficie.

\* @version 1.2/2020

\*/

public class VentanaCilindro extends JFrame implements ActionListener {

// Un contenedor de elementos gráficos

private Container contenedor;

/\* Etiquetas estáticas para identificar los campos de texto a ingresar

y calcular \*/

private JLabel radio, altura, volumen, superficie;

// Campos de texto a ingresar

private JTextField campoRadio, campoAltura;

// Botón para realizar los cálculos numéricos

private JButton calcular;

/\*\*

\* Constructor de la clase VentanaCilindro

\*/

public VentanaCilindro() {

inicio();

setTitle("Cilindro"); // Establece el título de la ventana

setSize(280,210); // Establece el tamaño de la ventana

setLocationRelativeTo(null); /\* La ventana se posiciona en el

centro de la pantalla \*/

setResizable(false); /\* Establece que el botón de cerrar permitirá

salir de la aplicación \*/

}

/\*\*

\* Método que crea la ventana con sus diferentes componentes

\* gráficos

\* \*/

private void inicio() {

contenedor = getContentPane(); /\* Obtiene el panel de

contenidos de la ventana \*/

contenedor.setLayout(null); /\* Establece que el contenedor no

tiene un layout \*/

// Establece la etiqueta y campo de texto para el radio del cilindro

radio = new JLabel();

radio.setText("Radio (cms):");

radio.setBounds(20, 20, 500, 23); /\* Establece la posición de la

etiqueta de radio del cilindro \*/

campoRadio = new JTextField();

// Establece la posición del campo de texto de radio del cilindro

campoRadio.setBounds(100, 20, 135, 23);

// Establece la etiqueta y campo de texto para la altura del cilindro

altura = new JLabel();

altura.setText("Altura (cms):");

altura.setBounds(20, 50, 135, 23); /\* Establece la posición de la

etiqueta de altura del cilindro \*/

campoAltura = new JTextField();

// Establece la posición del campo de texto de altura del cilindro

campoAltura.setBounds(100, 50, 135, 23);

/\* Establece el botón para calcular el volumen y superficie del

cilindro \*/

calcular = new JButton();

calcular.setText("Calcular");

calcular.setBounds(100, 80, 135, 23); /\* Establece la posición

del botón calcular \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/

calcular.addActionListener(this);

// Establece la etiqueta y el valor del volumen del cilindro

volumen = new JLabel();

volumen.setText("Volumen (cm3):");

// Establece la posición de la etiqueta de volumen del cilindro

volumen.setBounds(20, 110, 135, 23);

// Establece la etiqueta y el valor de la superficie del cilindro

superficie = new JLabel();

superficie.setText("Superficie (cm2):");

// Establece la posición de la etiqueta de superficie del cilindro

superficie.setBounds(20, 140, 500, 23);

// Se añade cada componente gráfico al contenedor de la ventana

contenedor.add(radio);

contenedor.add(campoRadio);

contenedor.add(altura);

contenedor.add(campoAltura);

contenedor.add(calcular);

contenedor.add(volumen);

contenedor.add(superficie);

}

/\*\*

\* Método que gestiona los eventos generados en la ventana del

\* cilindro throws Exception Excepción al ingresar un campo nulo o

\* error en formato de número

\*/

public void actionPerformed(ActionEvent event) {

// Se inicializan el radio y la altura del cilindro

boolean error = false; /\* Se inicializa variable para determinar si

ocurre un error \*/

double radio = 0;

double altura = 0;

try {

// Se obtiene el radio del cilindro ingresado

radio = Double.parseDouble(campoRadio.getText());

// Se obtiene la altura del cilindro ingresada

altura = Double.parseDouble(campoAltura.getText());

Cilindro cilindro = new Cilindro(radio, altura); /\* Se crea un

objeto Cilindro \*/

// Se calcula y muestra el volumen

volumen.setText("Volumen (cm3):" + String.format("%.2f",

cilindro.calcularVolumen()));

// Se calcula y muestra la superficie

superficie.setText("Superficie (cm2): " + String.format("%.2f",

cilindro.calcularSuperficie()));

} catch (Exception e){

error = true; // Si ocurre una excepción

} finally {

if(error) { /\* Si ocurre una excepción, se muestra un mensaje

de error \*/

JOptionPane.showMessageDialog(null,"Campo nulo o error en formato de numero",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

**Clase VentanaEsfera:**

package com.mycompany.figurageometrica;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Esta clase denominada VentanaEsfera define una ventana para

\* ingresar los datos de una esfera y calcular su volumen y superficie.

\* @version 1.2/2020

\*/

public class VentanaEsfera extends JFrame implements ActionListener {

// Un contenedor de elementos gráficos

private Container contenedor;

/\* Etiquetas estáticas para identificar los campos de texto a ingresar

y calcular \*/

private JLabel radio, volumen, superficie;

private JTextField campoRadio; // Campo de texto a ingresar

private JButton calcular; /\* Botón para realizar los cálculos

numéricos \*/

/\*\*

\* Constructor de la clase VentanaEsfera

\*/

public VentanaEsfera() {

inicio();

setTitle("Esfera"); // Establece el título de la ventana

setSize(280,200); // Establece el tamaño de la ventana

setLocationRelativeTo(null); /\* La ventana se posiciona en el

centro de la pantalla \*/

setResizable(false); /\* Establece que el botón de cerrar permitirá

salir de la aplicación \*/

}

/\*\*

\* Método que crea la ventana con sus diferentes componentes

\* gráficos

\*/

private void inicio() {

contenedor = getContentPane(); /\* Obtiene el panel de

contenidos de la ventana \*/

contenedor.setLayout(null); /\* Establece que el contenedor no

tiene un layout \*/

// Establece la etiqueta y campo de texto para el radio de la esfera

radio = new JLabel();

radio.setText("Radio (cms):");

radio.setBounds(20, 20, 500, 23); /\* Establece la posición de la

etiqueta de radio de la esfera \*/

campoRadio = new JTextField();

// Establece la posición del campo de texto de radio de la esfera

campoRadio.setBounds(100, 20, 135, 23);

/\* Establece el botón para calcular el volumen y superficie de la

esfera \*/

calcular = new JButton();

calcular.setText("Calcular");

calcular.setBounds(100, 50, 135, 23); /\* Establece la posición

del botón calcular \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/

calcular.addActionListener(this);

// Establece la etiqueta y el valor del volumen de la esfera

volumen = new JLabel();

volumen.setText("Volumen (cm3):");

// Establece la posición de la etiqueta de volumen de la esfera

volumen.setBounds(20, 90, 500, 23);

// Establece la etiqueta y el valor de la superficie de la esfera

superficie = new JLabel();

superficie.setText("Superficie (cm2):");

// Establece la posición de la etiqueta de superficie de la esfera

superficie.setBounds(20, 120, 500, 23);

// Se añade cada componente gráfico al contenedor de la ventana

contenedor.add(radio);

contenedor.add(campoRadio);

contenedor.add(calcular);

contenedor.add(volumen);

contenedor.add(superficie);

}

/\*\*

\* Método que gestiona los eventos generados en la ventana de la

\* esfera throws Exception Excepción al ingresar un campo nulo o

\* error en formato de número

\*/

public void actionPerformed(ActionEvent evento) {

if (evento.getSource() == calcular) { /\* Si se pulsa el botón

Calcular \*/

boolean error = false;

try {

// Se obtiene y convierte el valor numérico del radio

double radio = Double.parseDouble(campoRadio.

getText());

Esfera esfera = new Esfera(radio); /\* Se crea un objeto

Esfera \*/

// Se muestra el volumen

volumen.setText("Volumen (cm3): " + String.

format("%.2f", esfera.calcularVolumen()));

// Se muestra la superficie

superficie.setText("Superficie (cm2): " + String.format("%.2f",esfera.calcularSuperficie()));

} catch (Exception e) {

error = true; // Si ocurre una excepción

} finally {

if(error) { /\* Si ocurre una excepción, se muestra un

mensaje de error \*/

JOptionPane.showMessageDialog(null,"Campo nulo oerror en formato de número","Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

}

**Clase Piramide:**

package com.mycompany.figurageometrica;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Esta clase denominada VentanaPirámide define una ventana para

\* ingresar los datos de una pirámide y calcular su volumen y superficie.

\* @version 1.2/2020

\*/

public class VentanaPirámide extends JFrame implements

ActionListener {

// Un contenedor de elementos gráficos

private Container contenedor;

/\* Etiquetas estáticas para identificar los campos de texto a ingresar

y calcular \*/

private JLabel base, altura, apotema, volumen, superficie;

// Campos de texto a ingresar

private JTextField campoBase, campoAltura, campoApotema;

// Botón para realizar los cálculos numéricos

private JButton calcular;

/\*\*

\* Constructor de la clase VentanaPirámide

\*/

public VentanaPirámide() {

inicio();

setTitle("Pirámide"); // Establece el título de la ventana

setSize(280,240); // Establece el tamaño de la ventana

setLocationRelativeTo(null); /\* La ventana se posiciona en el

centro de la pantalla \*/

setResizable(false); /\* Establece que el botón de cerrar permitirá

salir de la aplicación \*/

}

/\*\*

\* Método que crea la ventana con sus diferentes componentes

\* gráficos

\*/

private void inicio() {

contenedor = getContentPane(); /\* Obtiene el panel de

contenidos de la ventana \*/

contenedor.setLayout(null); /\* Establece que el contenedor no

tiene un layout \*/

/\* Establece la etiqueta y campo de texto para la base de la

pirámide \*/

base = new JLabel();

base.setText("Base (cms):");

// Establece la posición de la etiqueta de la base de la pirámide

base.setBounds(20, 20, 500, 23);

campoBase = new JTextField();

/\* Establece la posición del campo de texto de la base de la

pirámide \*/

campoBase.setBounds(120, 20, 135, 23);

/\* Establece la etiqueta y campo de texto para la altura de la

pirámide \*/

altura = new JLabel();

altura.setText("Altura (cms):");

// Establece la posición de la etiqueta de la altura de la pirámide

altura.setBounds(20, 50, 500, 23);

campoAltura = new JTextField();

/\* Establece la posición del campo de texto de la altura de la

pirámide \*/

campoAltura.setBounds(120, 50, 135, 23);

/\* Establece la etiqueta y campo de texto para el apotema de la

pirámide \*/

apotema = new JLabel();

apotema.setText("Apotema (cms):");

// Establece la posición de la etiqueta del apotema de la pirámide

apotema.setBounds(20, 80, 500, 23);

campoApotema = new JTextField();

/\* Establece la posición del campo de texto del apotema de la

pirámide \*/

campoApotema.setBounds(120, 80, 135, 23);

/\* Establece el botón para calcular volumen y superficie de la

pirámide \*/

calcular = new JButton();

calcular.setText("Calcular");

calcular.setBounds(120, 110, 135, 23); /\* Establece la posición

del botón calcular \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/

calcular.addActionListener(this);

// Establece la etiqueta y el valor del volumen de la pirámide

volumen = new JLabel();

volumen.setText("Volumen (cm3):");

// Establece la posición de la etiqueta de volumen de la pirámide

volumen.setBounds(20, 140, 500, 23);

// Establece la etiqueta y el valor de la superficie de la pirámide

superficie = new JLabel();

superficie.setText("Superficie (cm2):");

// Establece la posición de la etiqueta de superficie de la pirámide

superficie.setBounds(20, 170, 500, 23);

// Se añade cada componente gráfico al contenedor de la ventana

contenedor.add(base);

contenedor.add(campoBase);

contenedor.add(altura);

contenedor.add(campoAltura);

contenedor.add(apotema);

contenedor.add(campoApotema);

contenedor.add(calcular);

contenedor.add(volumen);

contenedor.add(superficie);

}

/\*\*

\* Método que gestiona los eventos generados en la ventana de la

\* esfera throws Exception Excepción al ingresar un campo nulo o

\* error en formato de número

\* \*/

public void actionPerformed(ActionEvent event) {

Piramide pirámide;

boolean error = false;

double base = 0;

double altura = 0;

double apotema = 0;

try {

// Se obtiene y convierte el valor numérico de la base

base = Double.parseDouble(campoBase.getText());

// Se obtiene y convierte el valor numérico de la altura

altura = Double.parseDouble(campoAltura.getText());

// Se obtiene y convierte el valor numérico del apotema

apotema = Double.parseDouble(campoApotema.getText());

// Se crea un objeto Pirámide

pirámide = new Piramide(base, altura, apotema);

// Se muestra el volumen

volumen.setText("Volumen (cm3): " + String.format("%.2f",

pirámide.calcularVolumen()));

// Se muestra la superficie

superficie.setText("Superficie (cm2): " + String.format("%.2f",

pirámide.calcularSuperficie()));

} catch (Exception e) {

error = true; // Si ocurre una excepción

} finally {

if (error) { /\* Si ocurre una excepción, se muestra un mensaje

de error \*/

JOptionPane.showMessageDialog(null, "Campo nulo o error en formato de número","Error",JOptionPane.ERROR\_MESSAGE);

}

}

}

}

**Clase VentanaPrincipal:**

package com.mycompany.figurageometrica;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Esta clase denominada VentanaPrincipal define una interfaz gráfica

\* que permitirá consultar un menú principal con tres figuras

\* geométricas.

\* @version 1.2/2020

\*/

public class VentanaPrincipal extends JFrame implements

ActionListener {

// Un contenedor de elementos gráficos

private Container contenedor;

// Botones para seleccionar una figura geométrica determinada

private JButton cilindro, esfera, pirámide;

/\*\*

\* Constructor de la clase VentanaPrincipal

\*/

public VentanaPrincipal(){

inicio();

setTitle("Figuras"); // Establece el título de la ventana

setSize(350,160); // Establece el tamaño de la ventana

setLocationRelativeTo(null); /\* La ventana se posiciona en el

centro de la pantalla \*/

// Establece que el botón de cerrar permitirá salir de la aplicación

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

/\*\*

\* Método que crea la ventana con sus diferentes componentes

\* gráficos

\*/

private void inicio() {

contenedor = getContentPane(); /\* Obtiene el panel de

contenidos de la ventana \*/

contenedor.setLayout(null); /\* Establece que el contenedor no

tiene un layout \*/

// Establece el botón del cilindro

cilindro = new JButton();

cilindro.setText("Cilindro");

cilindro.setBounds(20, 50, 80, 23); /\* Establece la posición del

botón del cilindro \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/

cilindro.addActionListener(this);

// Establece el botón de la esfera

esfera = new JButton();

esfera.setText("Esfera");

esfera.setBounds(125, 50, 80, 23); /\* Establece la posición del

botón de la esfera \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/

esfera.addActionListener(this);

// Establece el botón de la pirámide

pirámide = new JButton();

pirámide.setText("Pirámide");

pirámide.setBounds(225, 50, 100, 23); /\* Establece la posición

del botón de la pirámide \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/

pirámide.addActionListener(this);

// Se añade cada componente gráfico al contenedor de la ventana

contenedor.add(cilindro);

contenedor.add(esfera);

contenedor.add(pirámide);

}

/\*\*

\* Método que gestiona los eventos generados en la ventana principal

\*/

public void actionPerformed(ActionEvent evento) {

if (evento.getSource() == esfera) { // Si se pulsa el botón esfera

VentanaEsfera esfera = new VentanaEsfera(); /\* Crea la

ventana de la esfera \*/

esfera.setVisible(true); /\* Establece que se visualice la ventana

de la esfera \*/

}

if (evento.getSource() == cilindro) { /\* Si se pulsa el botón

cilindro \*/

VentanaCilindro cilindro = new VentanaCilindro(); /\* Crea la

ventana del cilindro \*/

cilindro.setVisible(true); /\* Establece que se visualice la

ventana del cilindro \*/

}

if (evento.getSource() == pirámide) { /\* Si se pulsa el botón

pirámide \*/

VentanaPirámide pirámide = new VentanaPirámide(); /\* Crea

la ventana de la pirámide \*/

pirámide.setVisible(true); /\* Establece que se visualice la

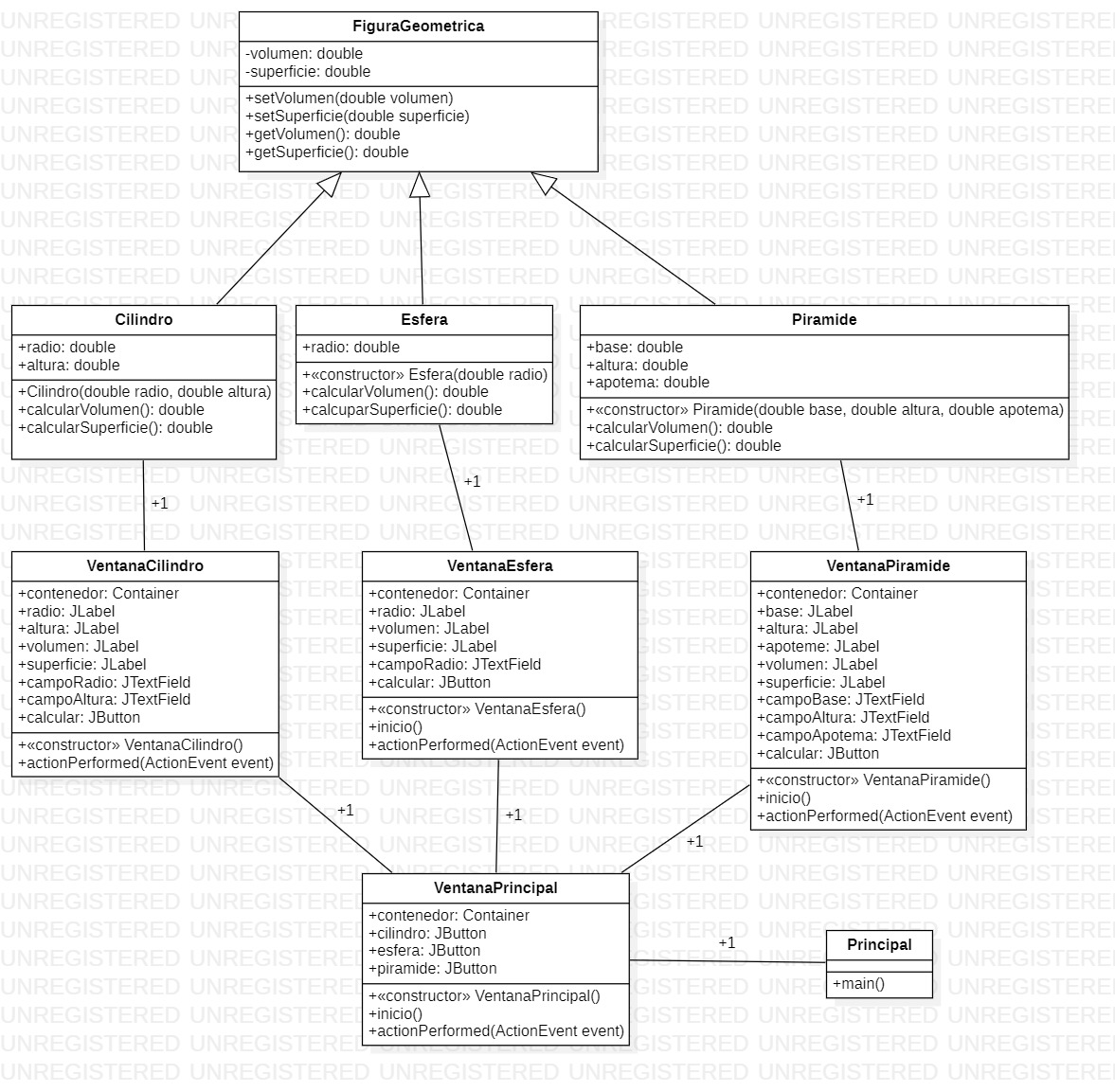
ventana de la pirámide \*/

}

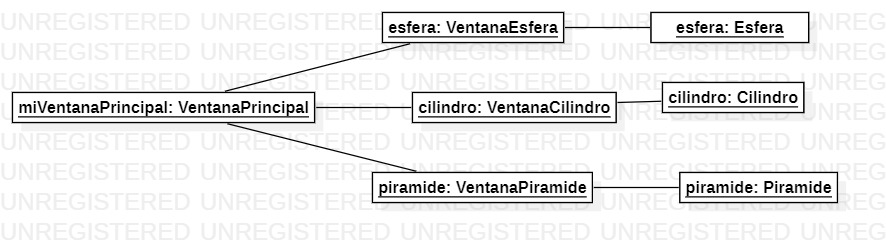
}

}

**Diagramas:**

****

**Diagrama de objetos:**

****