Universidad Nacional de Colombia
Sede Medellín

Facultad de Ciencias

Actividad 4

Estudiantes

Juan Felipe Muriel Mosquera

Maria Fernanda Valencia Jimenez

Docente

Walter Hugo Arboleda Mazo

Junio 2025

1. Ejercicio 8.2 página 483

Código fuente

```
import tkinter as tk
from tkinter import messagebox
import math

class Notas:
    def __init__(self):
        self.listaNotas = [0.0] * 5

    def calcular_promedio(self):
        suma = 0.0
        for nota in self.listaNotas:
            suma += nota
        return suma / len(self.listaNotas)

    def calcular_desviacion(self):
        prom = self.calcular_promedio()
        suma = 0.0
        for nota in self.listaNotas:
            suma += (nota - prom) ** 2
        return math.sqrt(suma / len(self.listaNotas))

    def calcular_menor(self):
        if not self.listaNotas:
            return None
        menor = self.listaNotas[0]
        for nota in self.listaNotas:
            if nota < menor:
                menor = nota
        return menor

    def calcular_mayor(self):
        if not self.listaNotas:
            return None
        mayor = self.listaNotas[0]
        for nota in self.listaNotas:
            if nota > mayor:
                mayor = nota
        return mayor

class VentanaPrincipal(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Notas")
```

```
        self.geometry("280x380")
        self.resizable(False, False)
        self.center_window()
        self.create_widgets()

    def center_window(self):
        self.update_idletasks()
        width = self.winfo_width()
        height = self.winfo_height()
        x = (self.winfo_screenwidth() // 2) - (width // 2)
        y = (self.winfo_screenheight() // 2) - (height //
            2)
        self.geometry(f'{width}x{height}+{x}+{y}')

    def create_widgets(self):
        self.labels = []
        self.entry_fields = []
        for i in range(5):
            label = tk.Label(self, text=f"Nota {i+1}:")
            label.place(x=20, y=20 + i * 30, width=80,
                height=23)
            self.labels.append(label)
            entry = tk.Entry(self)
            entry.place(x=105, y=20 + i * 30, width=135,
                height=23)
            self.entry_fields.append(entry)

        self.calcular_button = tk.Button(self, text="
            Calcular", command=self.calcular_notas)
        self.calcular_button.place(x=20, y=170, width=100,
             height=23)

        self.limpiar_button = tk.Button(self, text="
            Limpiar", command=self.limpiar_campos)
        self.limpiar_button.place(x=125, y=170, width=80,
            height=23)

        self.promedio_label = tk.Label(self, text="
            Promedio = ")
        self.promedio_label.place(x=20, y=210, width=200,
            height=23)

        self.desviacion_label = tk.Label(self, text="
            Desviació'n estándar = ")
        self.desviacion_label.place(x=20, y=240, width
            =200, height=23)

        self.mayor_label = tk.Label(self, text="Nota mayor
```

```
            = ")
        self.mayor_label.place(x=20, y=270, width=200,
            height=23)

        self.menor_label = tk.Label(self, text="Nota menor
            = ")
        self.menor_label.place(x=20, y=300, width=200,
            height=23)

    def calcular_notas(self):
        notas_obj = Notas()
        try:
            for i, entry in enumerate(self.entry_fields):
                notas_obj.listaNotas[i] = float(entry.get
                    ())
        except ValueError:
            messagebox.showerror("Error de Entrada", "Por
                favor, ingrese valores numéricos válidos
                para las notas.")
            return

        promedio = notas_obj.calcular_promedio()
        desviacion = notas_obj.calcular_desviacion()
        mayor = notas_obj.calcular_mayor()
        menor = notas_obj.calcular_menor()

        self.promedio_label.config(text=f"Promedio = {
            promedio:.2f}")
        self.desviacion_label.config(text=f"Desviació'n est
            ándar = {desviacion:.2f}")
        self.mayor_label.config(text=f"Nota mayor = {mayor
            }")
        self.menor_label.config(text=f"Nota menor = {menor
            }")

    def limpiar_campos(self):
        for entry in self.entry_fields:
            entry.delete(0, tk.END)

        self.promedio_label.config(text="Promedio = ")
        self.desviacion_label.config(text="Desviació'n está
            ndar = ")
        self.mayor_label.config(text="Nota mayor = ")
        self.menor_label.config(text="Nota menor = ")

if __name__ == "__main__":
    app = VentanaPrincipal()
    app.mainloop()
```

Diagrama de clases y objetos

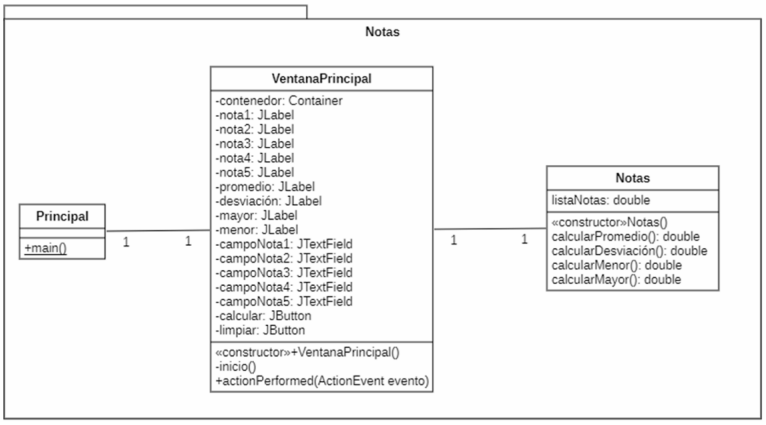


Figure 1: Diagrama de clases punto 1

Figure 2: Diagrama de objetos punto 1

Imagen de la interfaz de usuario

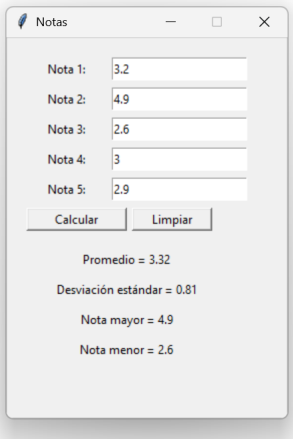


Figure 3: Interfaz gráfica punto 1

Casos de uso

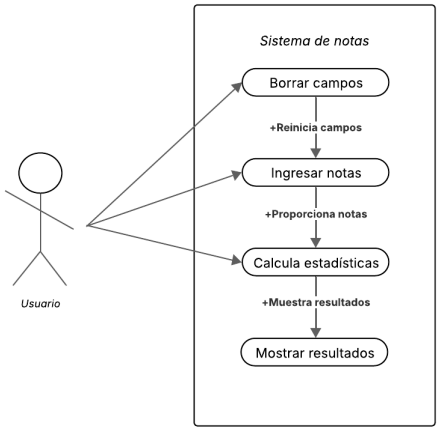


Figure 4: Casos de uso punto 1

2.Ejercicio 8.3 pagina 495

Código fuente

```
import math
import tkinter as tk
from tkinter import messagebox

## Clase: FiguraGeometrica
class FiguraGeometrica:

    def __init__(self):
        self._volumen = 0.0
        self._superficie = 0.0

    @property
    def volumen(self):
        return self._volumen

    @volumen.setter
    def volumen(self, volumen):
        self._volumen = volumen

    @property
    def superficie(self):
```

```
        return self._superficie

    @superficie.setter
    def superficie(self, superficie):
        self._superficie = superficie

## Clase: Cilindro
class Cilindro(FiguraGeometrica):

    def __init__(self, radio, altura):
        super().__init__()
        self.radio = radio
        self.altura = altura
        # Calcula el volumen y establece su atributo
        self.volumen = self.calcular_volumen()
        # Calcula la superficie y establece su atributo
        self.superficie = self.calcular_superficie()

    def calcular_volumen(self):
        volumen = math.pi * self.altura * (self.radio **
            2)
        return volumen

    def calcular_superficie(self):
        area_lado_a = 2.0 * math.pi * self.radio * self.
            altura
        area_lado_b = 2.0 * math.pi * (self.radio ** 2)
        return area_lado_a + area_lado_b

## Clase: Esfera
class Esfera(FiguraGeometrica):

    def __init__(self, radio):
        super().__init__()
        self.radio = radio
        # Calcula el volumen y establece su atributo
        self.volumen = self.calcular_volumen()
        # Calcula la superficie y establece su atributo
        self.superficie = self.calcular_superficie()

    def calcular_volumen(self):
        volumen = (4/3) * math.pi * (self.radio ** 3)
        return volumen

    def calcular_superficie(self):
        superficie = 4.0 * math.pi * (self.radio ** 2)
        return superficie
```

```
## Clase: Piramide
class Piramide(FiguraGeometrica):

    def __init__(self, base, altura, apotema):
        super().__init__()
        self.base = base
        self.altura = altura
        self.apotema = apotema
        # Calcula el volumen y establece su atributo
        self.volumen = self.calcular_volumen()
        # Calcula la superficie y establece su atributo
        self.superficie = self.calcular_superficie()

    def calcular_volumen(self):
        volumen = ((self.base ** 2) * self.altura) / 3.0
        return volumen

    def calcular_superficie(self):
        area_base = self.base ** 2
        area_lado = 2.0 * self.base * self.apotema
        return area_base + area_lado

## Clase: VentanaCilindro
class VentanaCilindro(tk.Toplevel):

    def __init__(self, master=None):
        super().__init__(master)
        self.title("Cilindro")
        self.geometry("280x210")
        self.grab_set()
        self.transient(master)
        self.resizable(False, False)
        self.inicio()

    def inicio(self):
        self.contenedor = tk.Frame(self)
        self.contenedor.pack(fill="both", expand=True)

        self.radio_label = tk.Label(self.contenedor, text
            ="Radio (cms):")
        self.radio_label.place(x=20, y=20, width=135,
            height=23)
        self.campo_radio = tk.Entry(self.contenedor)
        self.campo_radio.place(x=100, y=20, width=135,
            height=23)

        self.altura_label = tk.Label(self.contenedor, text
            ="Altura (cms):")
```

```
        self.altura_label.place(x=20, y=50, width=135,
            height=23)
        self.campo_altura = tk.Entry(self.contenedor)
        self.campo_altura.place(x=100, y=50, width=135,
            height=23)

        self.calcular_button = tk.Button(self.contenedor,
            text="Calcular", command=self.
            calcular_propiedades)
        self.calcular_button.place(x=100, y=80, width=135,
             height=23)

        self.volumen_label = tk.Label(self.contenedor,
            text="Volumen (cm3):")
        self.volumen_label.place(x=20, y=110, width=135,
            height=23)

        self.superficie_label = tk.Label(self.contenedor,
            text="Superficie (cm2):")
        self.superficie_label.place(x=20, y=140, width
            =135, height=23)

    def calcular_propiedades(self):
        error = False
        try:
            radio = float(self.campo_radio.get())
            altura = float(self.campo_altura.get())

            cilindro = Cilindro(radio, altura)

            self.volumen_label.config(text=f"Volumen (cm3)
                : {cilindro.calcular_volumen():.2f}")
            self.superficie_label.config(text=f"Superficie
                 (cm2): {cilindro.calcular_superficie():.2f
                }")

        except ValueError:
            error = True

        finally:
            if error:
                messagebox.showerror("Error", "Campo nulo
                    o error en formato de mí'mero")

## Clase: VentanaEsfera
class VentanaEsfera(tk.Toplevel):

    def __init__(self, master=None):
```

```
        super().__init__(master)
        self.title("Esfera")
        self.geometry("280x200")
        self.grab_set()
        self.transient(master)
        self.resizable(False, False)
        self.inicio()

    def inicio(self):
        self.contenedor = tk.Frame(self)
        self.contenedor.pack(fill="both", expand=True)

        self.radio_label = tk.Label(self.contenedor, text
            ="Radio (cms):")
        self.radio_label.place(x=20, y=20, width=135,
            height=23)
        self.campo_radio = tk.Entry(self.contenedor)
        self.campo_radio.place(x=100, y=20, width=135,
            height=23)

        self.calcular_button = tk.Button(self.contenedor,
            text="Calcular", command=self.
            calcular_propiedades)
        self.calcular_button.place(x=100, y=50, width=135,
             height=23)

        self.volumen_label = tk.Label(self.contenedor,
            text="Volumen (cm3):")
        self.volumen_label.place(x=20, y=90, width=135,
            height=23)

        self.superficie_label = tk.Label(self.contenedor,
            text="Superficie (cm2):")
        self.superficie_label.place(x=20, y=120, width
            =135, height=23)

    def calcular_propiedades(self):
        error = False
        try:
            radio = float(self.campo_radio.get())
            esfera = Esfera(radio)

            self.volumen_label.config(text=f"Volumen (cm3)
                : {esfera.calcular_volumen():.2f}")
            self.superficie_label.config(text=f"Superficie
                 (cm2): {esfera.calcular_superficie():.2f
                }")
```

```
        except ValueError:
            error = True

        finally:
            if error:
                messagebox.showerror("Error", "Campo nulo
                    o error en formato de mí'mero")

## Clase: VentanaPiramide
class VentanaPiramide(tk.Toplevel):

    def __init__(self, master=None):
        super().__init__(master)
        self.title("Pirámide")
        self.geometry("280x240")
        self.grab_set()
        self.transient(master)
        self.resizable(False, False)
        self.inicio()

    def inicio(self):
        self.contenedor = tk.Frame(self)
        self.contenedor.pack(fill="both", expand=True)

        self.base_label = tk.Label(self.contenedor, text="
            Base (cms):")
        self.base_label.place(x=20, y=20, width=135,
            height=23)
        self.campo_base = tk.Entry(self.contenedor)
        self.campo_base.place(x=120, y=20, width=135,
            height=23)

        self.altura_label = tk.Label(self.contenedor, text
            ="Altura (cms):")
        self.altura_label.place(x=20, y=50, width=135,
            height=23)
        self.campo_altura = tk.Entry(self.contenedor)
        self.campo_altura.place(x=120, y=50, width=135,
            height=23)

        self.apotema_label = tk.Label(self.contenedor,
            text="Apotema (cms):")
        self.apotema_label.place(x=20, y=80, width=135,
            height=23)
        self.campo_apotema = tk.Entry(self.contenedor)
        self.campo_apotema.place(x=120, y=80, width=135,
            height=23)
```

```
        self.calcular_button = tk.Button(self.contenedor,
            text="Calcular", command=self.
            calcular_propiedades)
        self.calcular_button.place(x=120, y=110, width
            =135, height=23)

        self.volumen_label = tk.Label(self.contenedor,
            text="Volumen (cm3):")
        self.volumen_label.place(x=20, y=140, width=135,
            height=23)

        self.superficie_label = tk.Label(self.contenedor,
            text="Superficie (cm2):")
        self.superficie_label.place(x=20, y=170, width
            =135, height=23)

    def calcular_propiedades(self):
        error = False
        try:
            base = float(self.campo_base.get())
            altura = float(self.campo_altura.get())
            apotema = float(self.campo_apotema.get())

            piramide = Piramide(base, altura, apotema)

            self.volumen_label.config(text=f"Volumen (cm3)
                : {piramide.calcular_volumen():.2f}")
            self.superficie_label.config(text=f"Superficie
                 (cm2): {piramide.calcular_superficie():.2f
                }")

        except ValueError:
            error = True

        finally:
            if error:
                messagebox.showerror("Error", "Campo nulo
                    o error en formato de mí'mero")

## Clase: VentanaPrincipal
class VentanaPrincipal(tk.Tk):

    def __init__(self):
        super().__init__()
        self.title("Figuras")
        self.geometry("350x160")
        self.resizable(False, False)
        # La ventana se posiciona en el centro de la
```

```
            pantalla
        self.eval('tk::PlaceWindow %s center' % self.
            winfo_toplevel())
        self.protocol("WM_DELETE_WINDOW", self.on_closing)
        self.inicio()

    def inicio(self):
        self.contenedor = tk.Frame(self)
        self.contenedor.pack(fill="both", expand=True)

        self.cilindro_button = tk.Button(self.contenedor,
            text="Cilindro", command=self.abrir_cilindro)
        self.cilindro_button.place(x=20, y=50, width=80,
            height=23)

        self.esfera_button = tk.Button(self.contenedor,
            text="Esfera", command=self.abrir_esfera)
        self.esfera_button.place(x=125, y=50, width=80,
            height=23)

        self.piramide_button = tk.Button(self.contenedor,
            text="Pirámide", command=self.abrir_piramide)
        self.piramide_button.place(x=225, y=50, width=100,
             height=23)

    def abrir_esfera(self):
        VentanaEsfera(self)

    def abrir_cilindro(self):
        VentanaCilindro(self)

    def abrir_piramide(self):
        VentanaPiramide(self)

    def on_closing(self):
        self.destroy()

## Punto de entrada principal
if __name__ == "__main__":
    mi_ventana_principal = VentanaPrincipal()
    mi_ventana_principal.mainloop()
```

Diagrama de clases y objetos

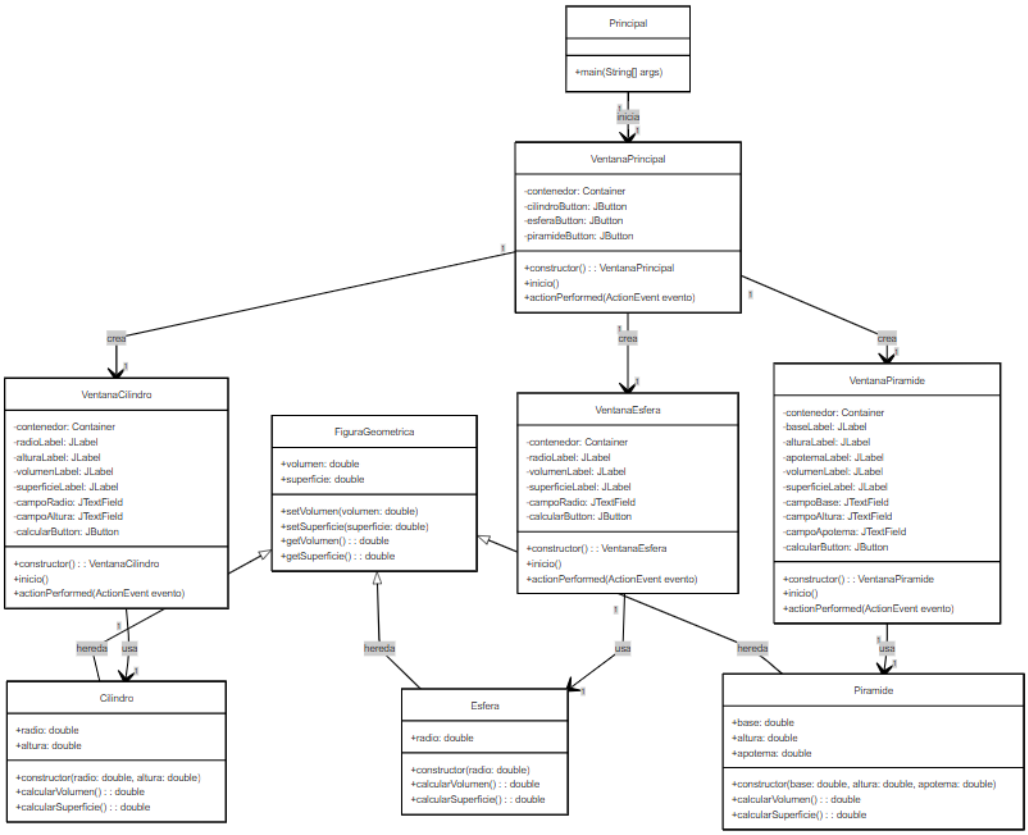


Figure 5: Diagrama de clases punto 2

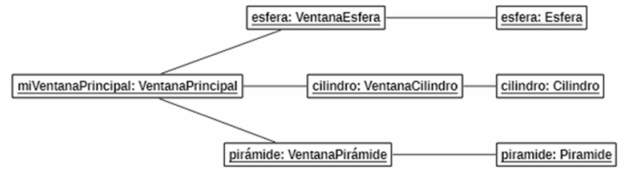


Figure 6: Diagrama de objetos punto 2

Casos de uso

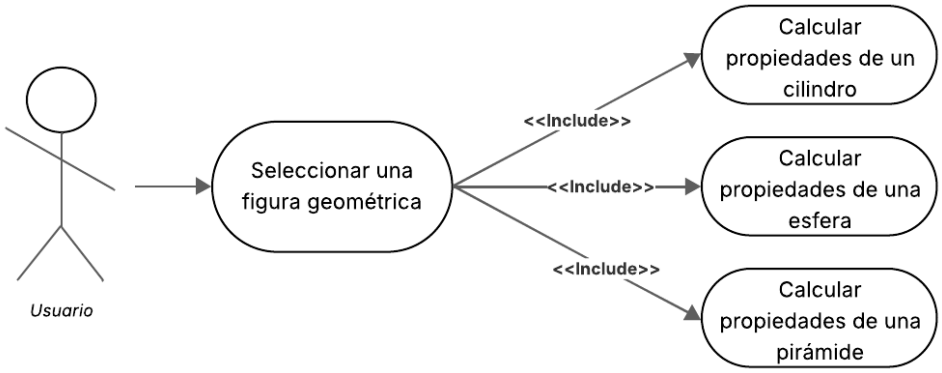
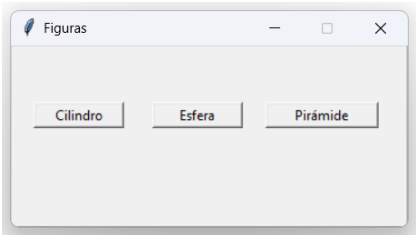
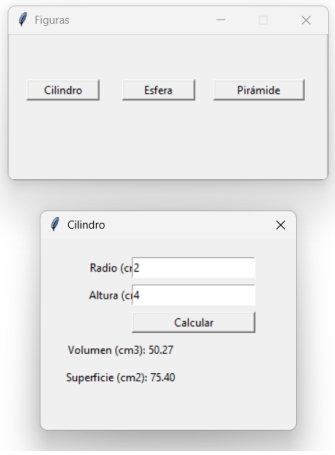


Figure 7: Casos de uso Punto 2

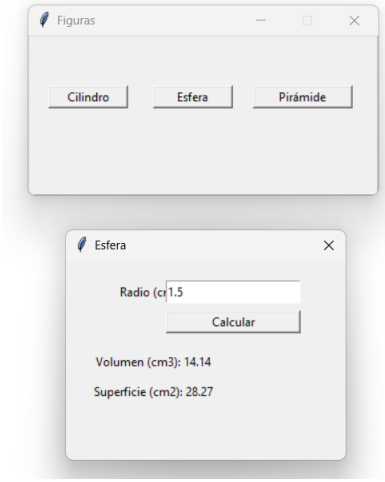
Imagen de la interfaz de usuario



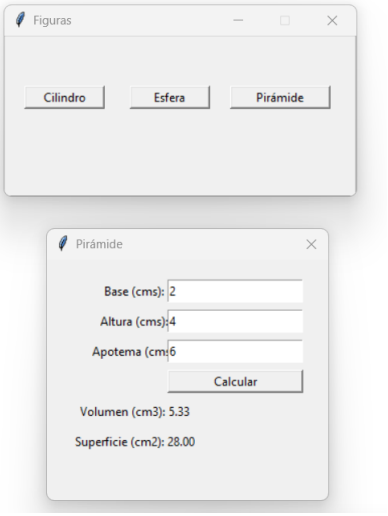
(a) : Interfaz gráfica



(b) Interfaz gráfica



(c) Interfaz gráfica



(d) Interfaz gráfica

Figure 8: Imagenes de la interfaz de usuario