

## **Actividad 5 grupal POO**

Estudiantes

**David Esteban Cartagena Mejia**

C.C: 1001361568

**Josué González Otálvaro**

C.C: 1001236911

**Juan Manuel Saldarriaga V.**

C.C: 1001226798

Docente

**Walter Hugo Arboleda Mazo**

Asignatura

**Programación Orientada a Objetos (POO)**



Universidad Nacional de Colombia Sede Medellín

Julio de 2025

## **Tabla de contenidos**

<b>Ejercicio 1</b>	<b>3</b>
<b>Ejercicio 2</b>	<b>4</b>
<b>Ejercicio 3</b>	<b>9</b>
<b>Ejercicio 4</b>	<b>13</b>
<b>Ejercicio 5</b>	<b>18</b>

## **Listado de Figuras**

1	Diagrama de clases . . . . .	3
2	Interfaz de usuario . . . . .	3
3	Diagrama de Usos . . . . .	4
4	Diagrama de clases . . . . .	5
5	Interfaz de usuario . . . . .	6
6	Diagrama de Usos . . . . .	7
7	Diagrama de clases . . . . .	10
8	Interfaz usuario . . . . .	11
9	Diagrama de Usos . . . . .	11
10	Diagrama de clases . . . . .	14
11	Interfaz usuario . . . . .	14
12	Diagrama de usos . . . . .	15
13	Diagrama de clases . . . . .	18
14	Interfaz de usuario . . . . .	19
15	Diagrama de Usos . . . . .	20

# Ejercicio 1

¿Cuál es el resultado de la ejecución del método main del siguiente programa? Determinar qué se imprime en pantalla.

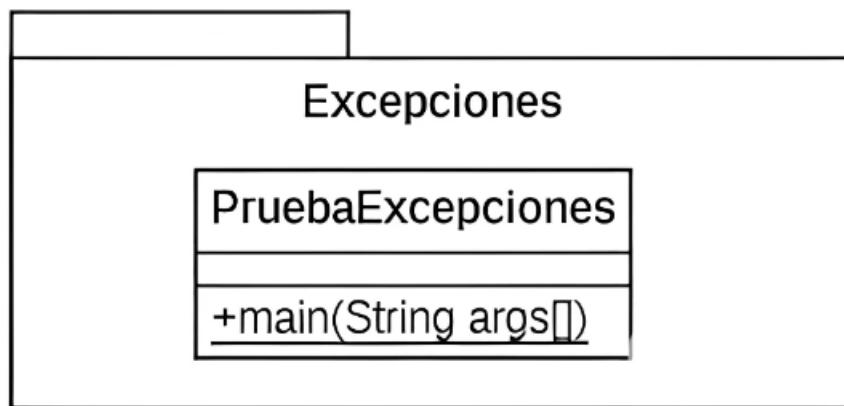


Figura 1: Diagrama de clases

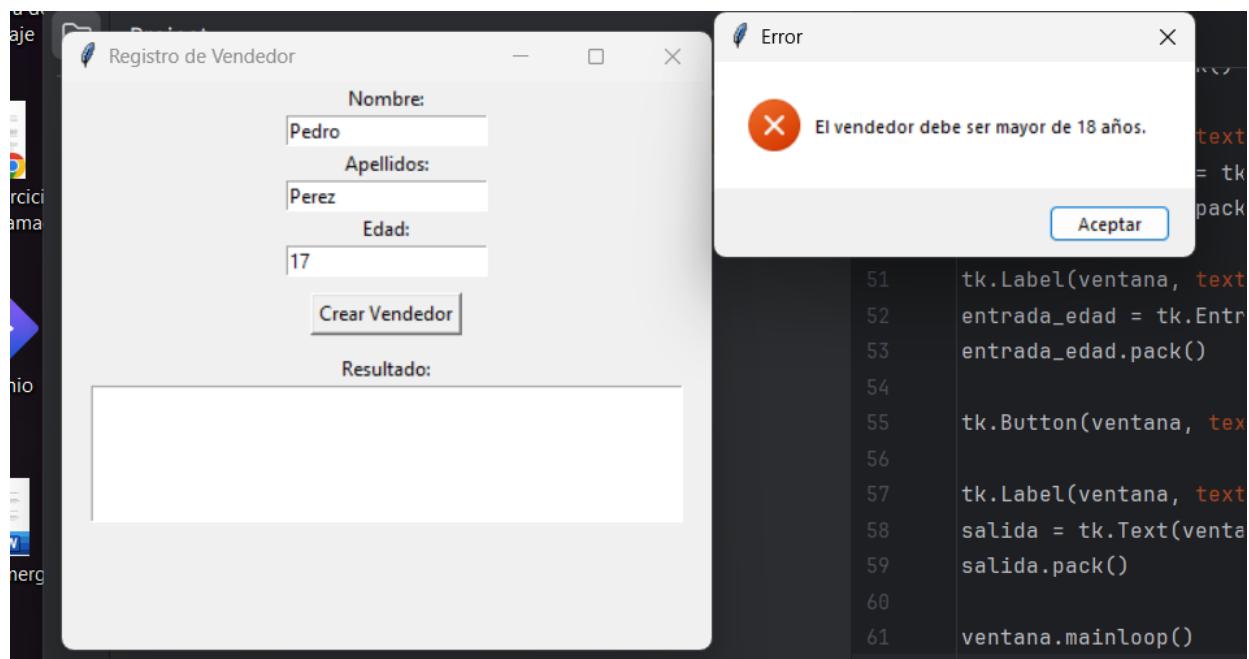


Figura 2: Interfaz de usuario

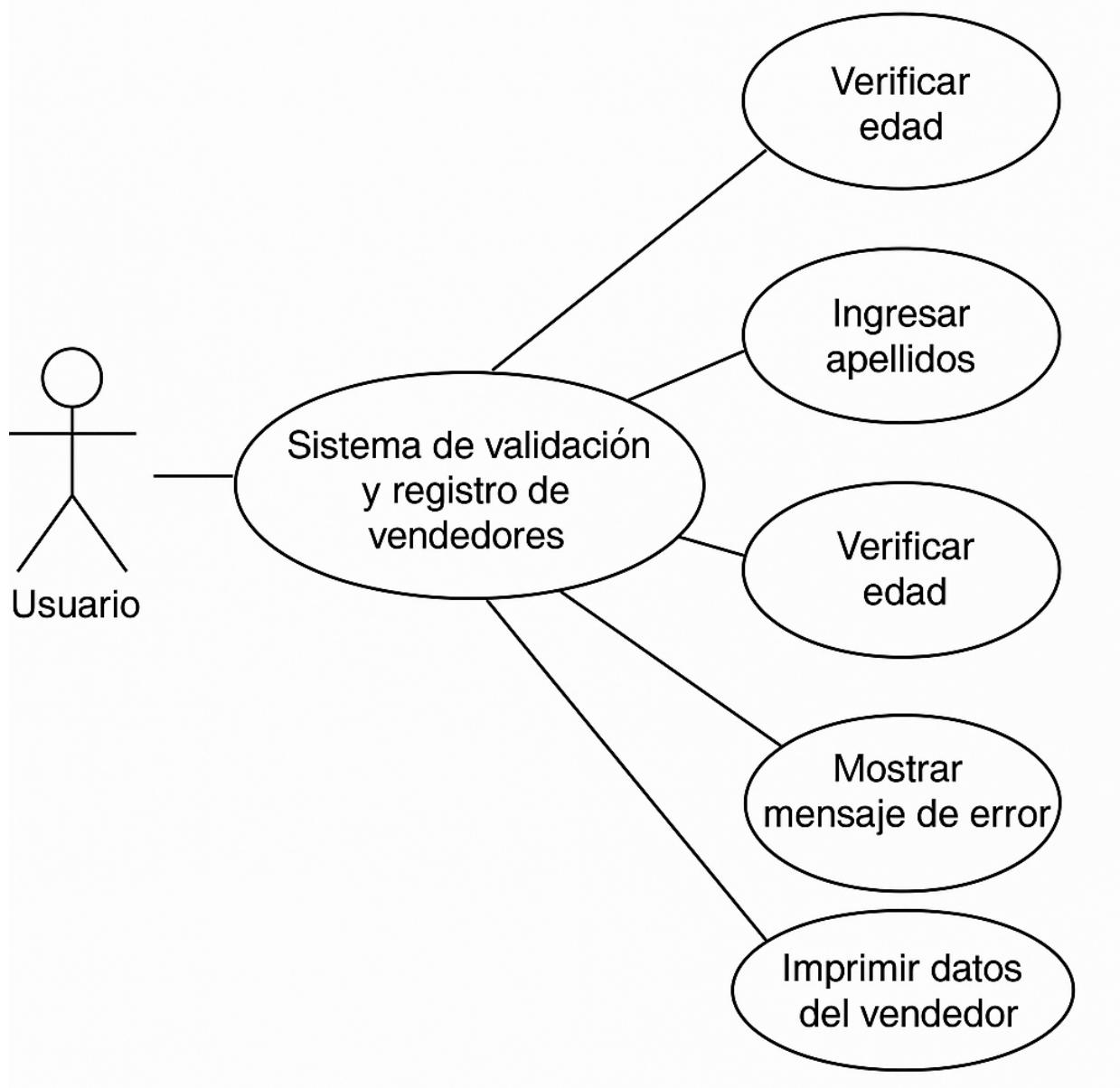


Figura 3: Diagrama de Usos

## Ejercicio 2

Se requiere implementar una clase vendedor que posee los siguientes atributos: nombre (tipo String), apellidos (tipo String) y edad (tipo int).

La clase contiene un constructor que inicialice los atributos de la clase. Además, la clase posee los siguientes métodos:

Imprimir: muestra por pantalla los valores de sus atributos.

Verificar edad: este método recibe como parámetro un valor entero que representa la edad del vendedor. Para que un vendedor pueda desempeñar sus labores se requiere que sea mayor de edad (mayor de 18 años). Si esta condición no se cumple, se lanza una excepción de tipo `IllegalArgumentException` con el mensaje “El vendedor debe ser mayor de 18 años”. Además, se evalúa si la edad se encuentra en el rango de 0 a 120, si no se cumple, se genera una excepción de tipo `IllegalArgumentException` con el mensaje “La edad no puede ser negativa ni mayor a 120”. Si la edad cumple estos requerimientos se pueden instanciar el objeto vendedor.

Además, se requiere que los datos del vendedor se ingresen por teclado

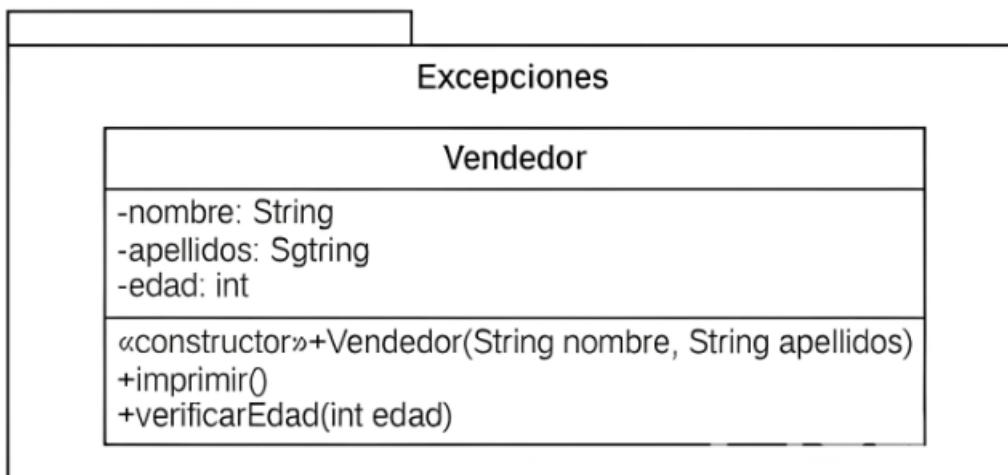


Figura 4: Diagrama de clases

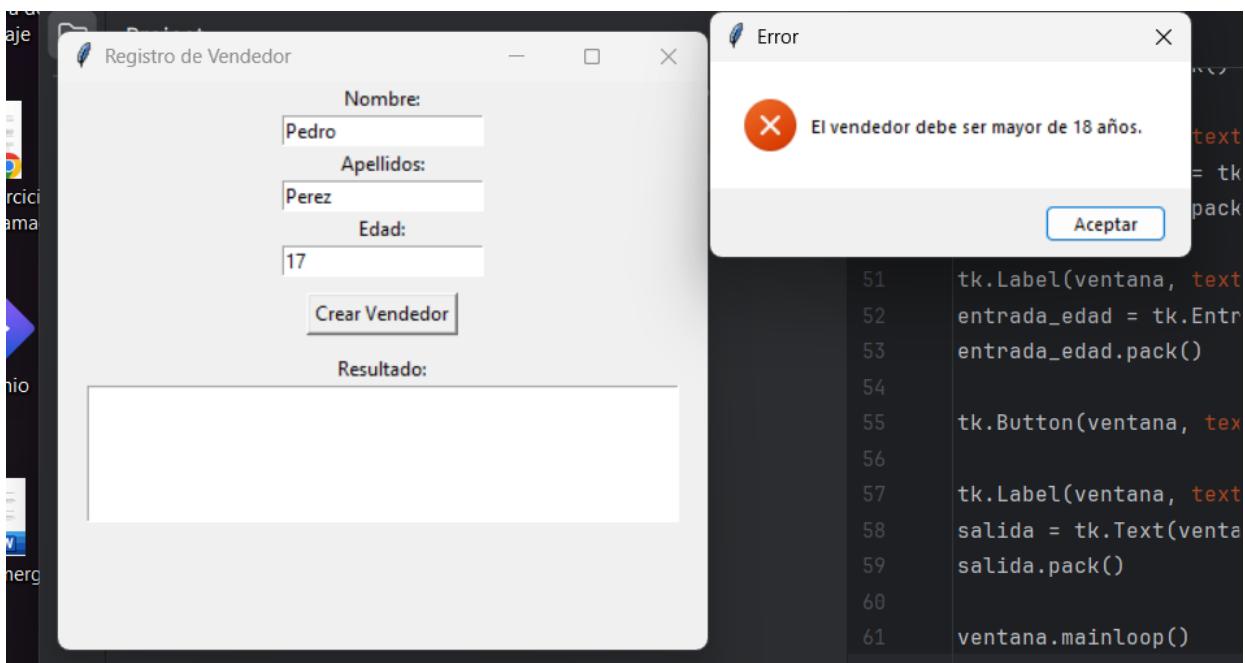


Figura 5: Interfaz de usuario

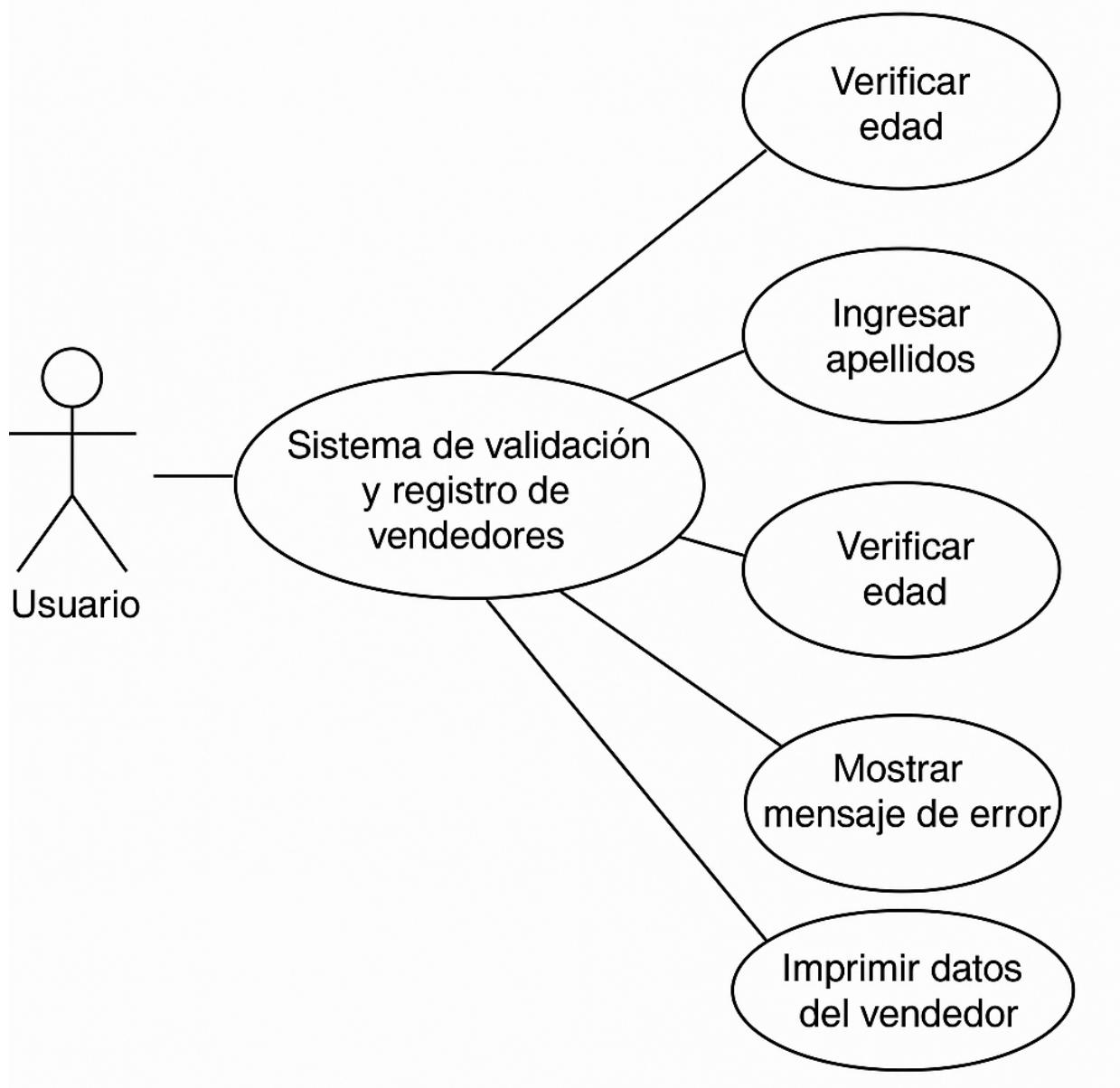


Figura 6: Diagrama de Usos

```

import tkinter as tk
from tkinter import messagebox

class Vendedor:
    def __init__(self, nombre, apellidos):
        self.nombre = nombre
        self.apellidos = apellidos
    
```

```

        self.edad = None

    def verificar_edad(self, edad):
        if edad < 0 or edad > 120:
            raise ValueError("La edad no puede ser negativa ni mayor a 120.")
        if edad < 18:
            raise ValueError("El vendedor debe ser mayor de 18 años.")
        self.edad = edad

    def imprimir(self):
        return f"Nombre del vendedor: {self.nombre}\nApellidos del vendedor: {self.apellidos}"

# Función para crear y mostrar el vendedor
def crear_vendedor():
    try:
        nombre = entrada_nombre.get()
        apellidos = entrada_apellidos.get()
        edad = int(entrada_edad.get())

        vendedor = Vendedor(nombre, apellidos)
        vendedor.verificar_edad(edad)
        resultado = vendedor.imprimir()
        salida.config(state='normal')
        salida.delete(1.0, tk.END)
        salida.insert(tk.END, resultado)
        salida.config(state='disabled')

    except ValueError as e:
        messagebox.showerror("Error", str(e))

# Interfaz gráfica
ventana = tk.Tk()
ventana.title("Registro de Vendedor")
ventana.geometry("400x350")

tk.Label(ventana, text="Nombre:").pack()
entrada_nombre = tk.Entry(ventana)
entrada_nombre.pack()

tk.Label(ventana, text="Apellidos:").pack()
entrada_apellidos = tk.Entry(ventana)
entrada_apellidos.pack()

```

```
tk.Label(ventana, text="Edad:").pack()
entrada_edad = tk.Entry(ventana)
entrada_edad.pack()

tk.Button(ventana, text="Crear Vendedor", command=crear_vendedor).pack(pady=10)

tk.Label(ventana, text="Resultado:").pack()
salida = tk.Text(ventana, height=5, width=45, state='disabled')
salida.pack()

ventana.mainloop()
```

## Ejercicio 3

Se requiere definir una clase denominada CálculosNúmericos que realice las siguientes operaciones:

Calcular el logaritmo neperiano recibiendo un valor double como parámetro. Este método debe ser estático. Si el valor no es positivo se genera una excepción aritmética.

Calcular la raíz cuadrada recibiendo un valor double como parámetro. Este método debe ser estático. Si el valor no es positivo se genera una excepción aritmética.

Se debe crear un método main que utilice dichos métodos mediante interfaz gráfica de usuario.

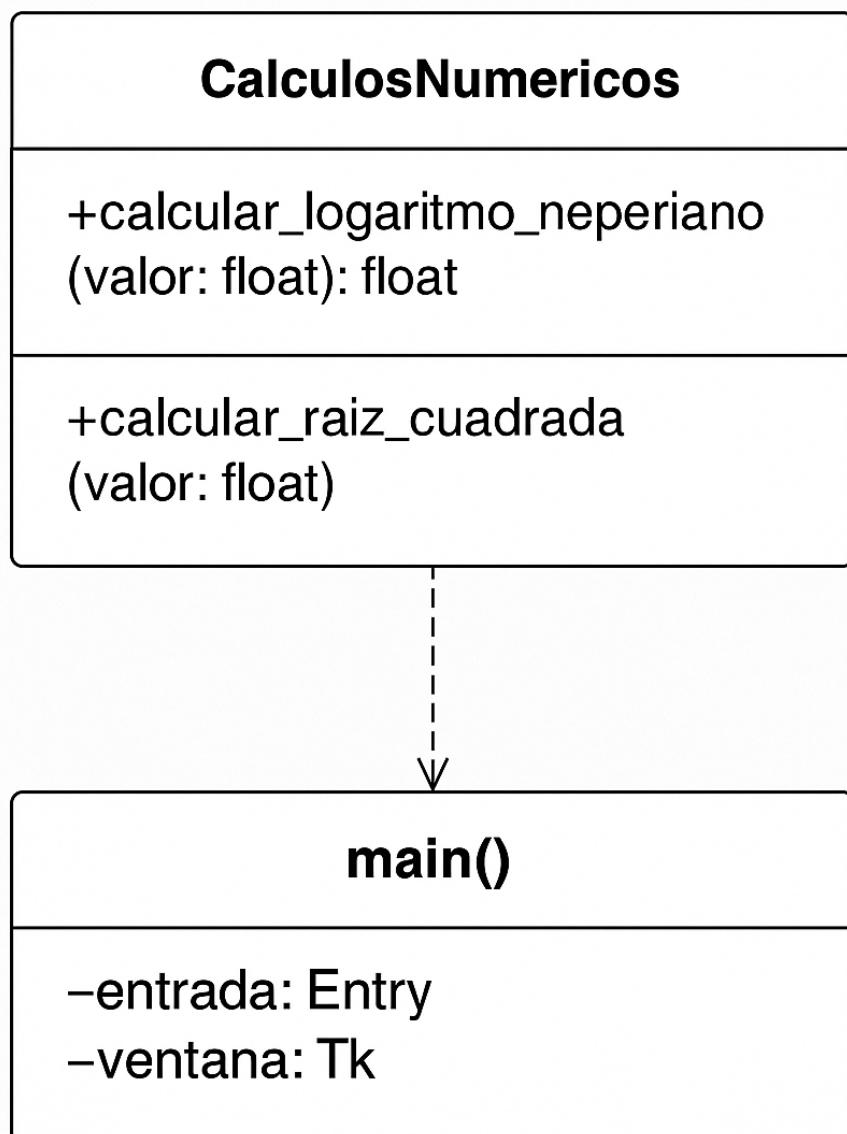


Figura 7: Diagrama de clases

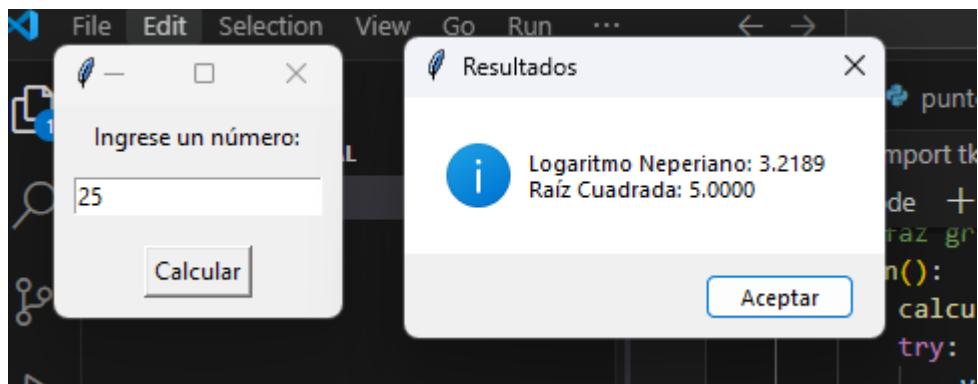


Figura 8: Interfaz usuario

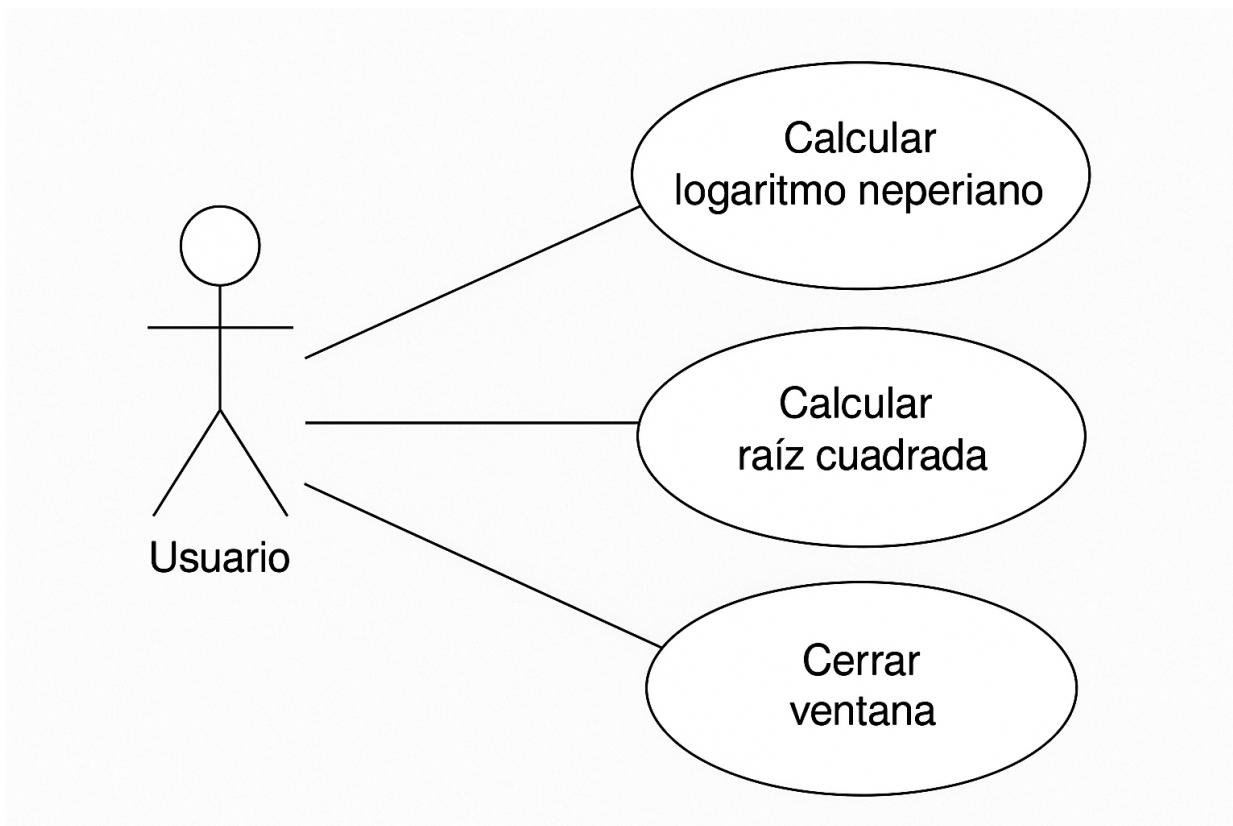


Figura 9: Diagrama de Usos

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

class CalculosNumericos:
```

```

@staticmethod
def calcular_logaritmo_neperiano(valor):
    if valor <= 0:
        raise ArithmeticError("El valor debe ser un número positivo para calcular el logaritmo neperiano")
    return math.log(valor)

@staticmethod
def calcular_raiz_cuadrada(valor):
    if valor < 0:
        raise ArithmeticError("El valor debe ser un número positivo para calcular la raíz cuadrada")
    return math.sqrt(valor)

# Interfaz gráfica
def main():
    def calcular():
        try:
            valor = float(entrada.get())
            resultado_log = CalculosNumericos.calcular_logaritmo_neperiano(valor)
            resultado_raiz = CalculosNumericos.calcular_raiz_cuadrada(valor)
            messagebox.showinfo("Resultados",
                                f"Logaritmo Neperiano: {resultado_log:.4f}\nRaíz Cuadrada: {resultado_raiz:.4f}")
        except ValueError:
            messagebox.showerror("Error", "Debe ingresar un valor numérico.")
        except ArithmeticError as e:
            messagebox.showerror("Error aritmético", str(e))

    ventana = tk.Tk()
    ventana.title("Cálculos Numéricicos")

    tk.Label(ventana, text="Ingrese un número:").pack(padx=10, pady=5)
    entrada = tk.Entry(ventana)
    entrada.pack(padx=10, pady=5)

    boton = tk.Button(ventana, text="Calcular", command=calcular)
    boton.pack(padx=10, pady=10)

    ventana.mainloop()

if __name__ == "__main__":
    main()

```

## Ejercicio 4

Un equipo de programadores desea participar en una maratón de programación. El equipo tiene los siguientes atributos:

Nombre del equipo (tipo String).

Universidad que está representando el equipo (tipo String).

Lenguaje de programación que va a utilizar el equipo en la competencia (tipo String).

Tamaño del equipo (tipo int).

Se requiere un constructor que inicialice los atributos del equipo. El equipo está conformado por varios programadores, mínimo dos y máximo tres. Cada programador posee nombre y apellidos (de tipo String).

Se requieren además los siguientes métodos:

Un método para determinar si el equipo está completo.

Un método para añadir programadores al equipo. Si el equipo está lleno se debe lanzar la excepción correspondiente.

Un método para validar los atributos nombre y apellidos de un programador, para que reciban datos que sean solo texto. Si se reciben datos numéricos, se debe generar la excepción correspondiente. Además, no se permite que los campos String tengan una longitud igual o superior a 20 caracteres.

En un método main se debe crear un equipo solicitando sus datos por teclado y se deben validar los nombres y apellidos de los programadores.

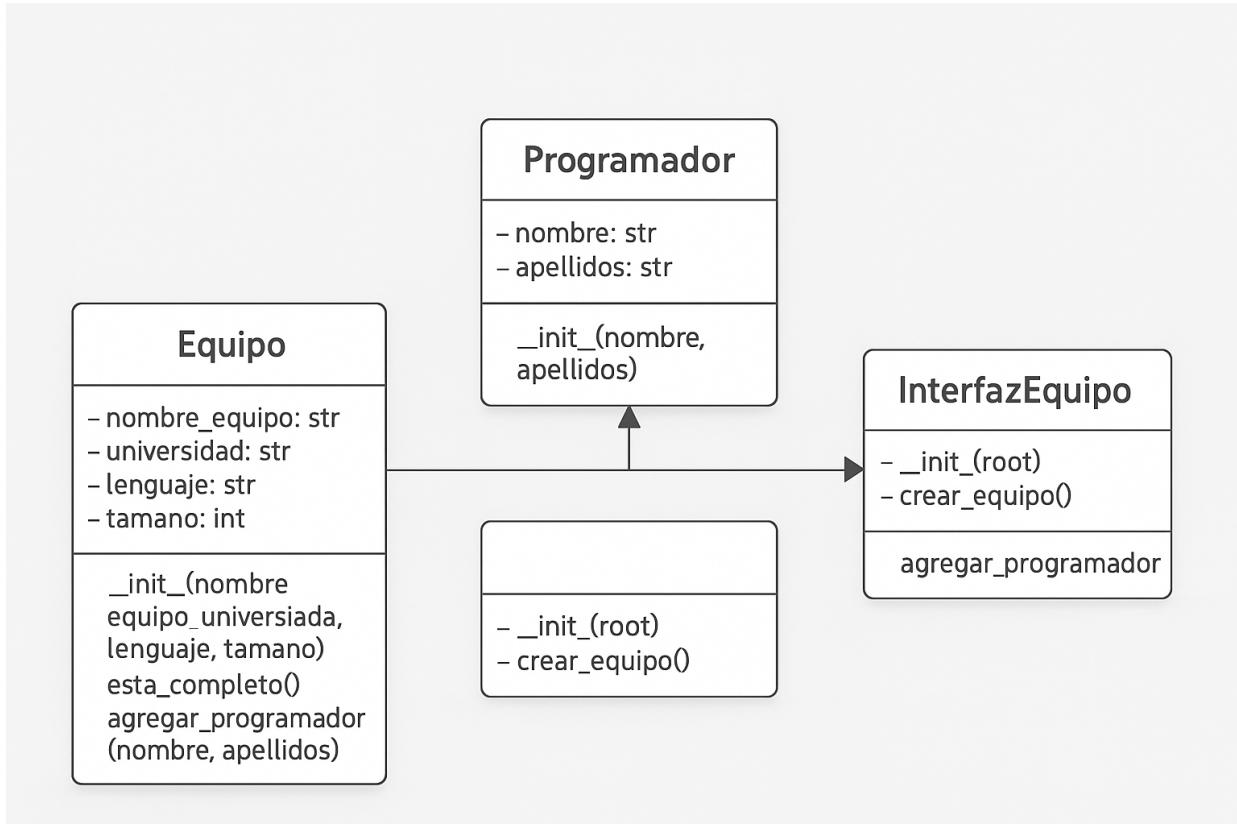


Figura 10: Diagrama de clases

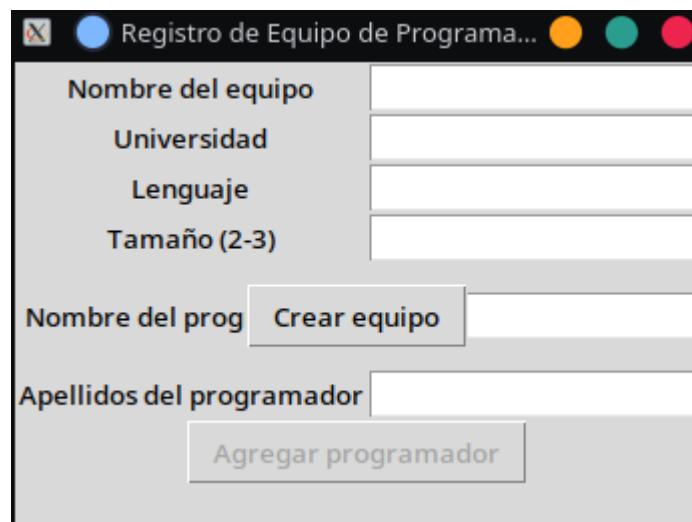


Figura 11: Interfaz usuario

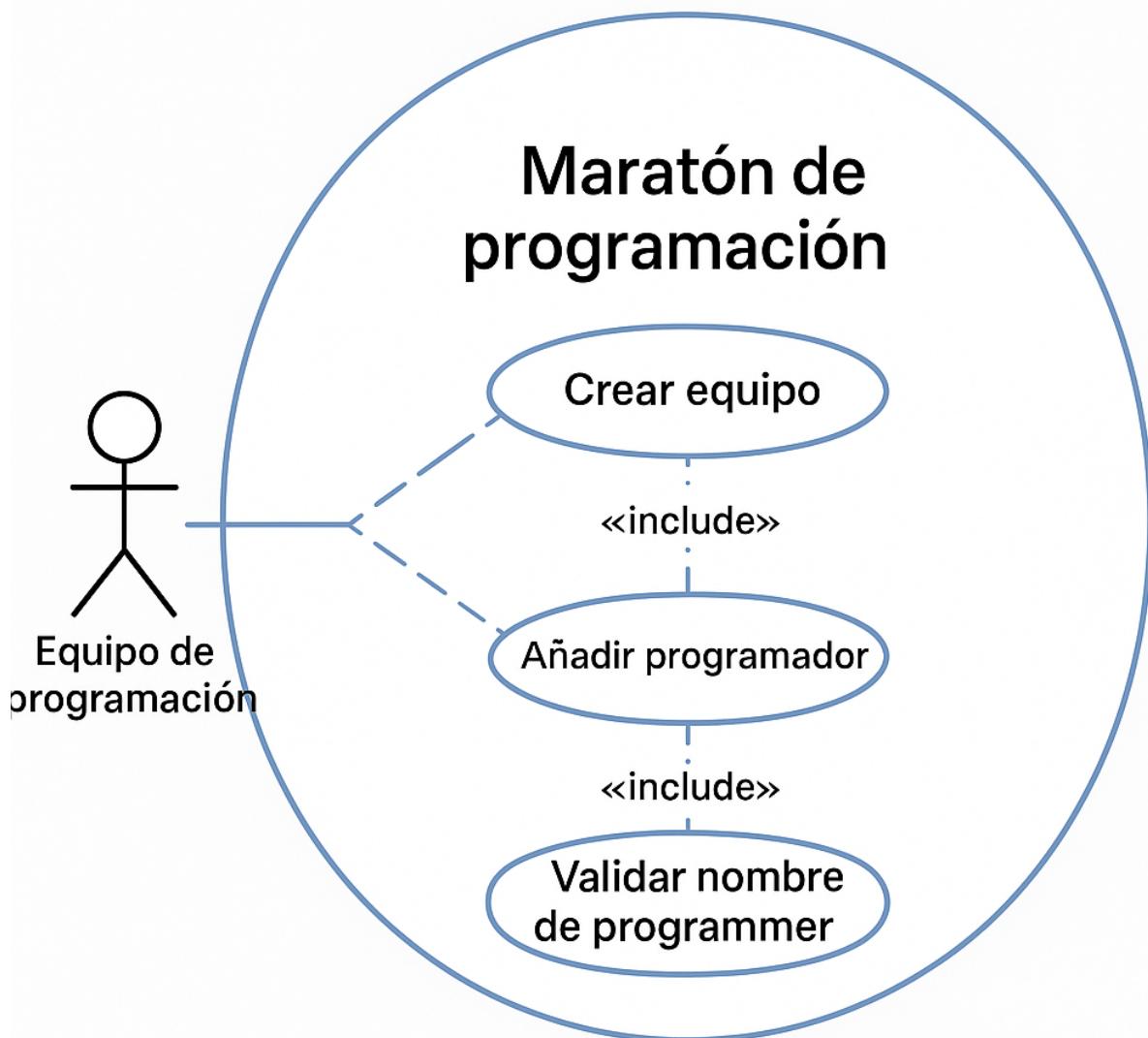


Figura 12: Diagrama de usos

```

import tkinter as tk
from tkinter import messagebox

class Programador:
    def __init__(self, nombre, apellidos):
        if not nombre.isalpha() or not apellidos.isalpha():
            raise ValueError("Nombre y apellidos deben contener solo letras.")

```

```

if len(nombre) >= 20 or len(apellidos) >= 20:
    raise ValueError("Nombre y apellidos deben tener menos de 20 caracteres.")
self.nombre = nombre
self.apellidos = apellidos

class Equipo:
    def __init__(self, nombre_equipo, universidad, lenguaje, tamano):
        if not (2 <= tamano <= 3):
            raise ValueError("El tamaño del equipo debe ser entre 2 y 3.")
        self.nombre_equipo = nombre_equipo
        self.universidad = universidad
        self.lenguaje = lenguaje
        self.tamano = tamano
        self.programadores = []

    def esta_completo(self):
        return len(self.programadores) == self.tamano

    def agregar_programador(self, nombre, apellidos):
        if self.está_completo():
            raise Exception("El equipo ya está completo.")
        nuevo = Programador(nombre, apellidos)
        self.programadores.append(nuevo)

class InterfazEquipo:
    def __init__(self, root):
        self.root = root
        self.root.title("Registro de Equipo de Programación")

        self.etiquetas = ["Nombre del equipo", "Universidad", "Lenguaje", "Tamaño (2-3)",]
        self.entradas = []

        for idx, texto in enumerate(self.etiquetas):
            label = tk.Label(root, text=texto)
            label.grid(row=idx, column=0)
            entry = tk.Entry(root)
            entry.grid(row=idx, column=1)
            self.entradas.append(entry)

        self.boton_crear = tk.Button(root, text="Crear equipo", command=self.crear_equipo)
        self.boton_crear.grid(row=4, column=0, columnspan=2, pady=10)

```

```

self.boton_agregar = tk.Button(root, text="Aregar programador", command=self.agregar_programador)
self.boton_agregar.grid(row=7, column=0, columnspan=2)

self.resultado = tk.Label(root, text="")
self.resultado.grid(row=8, column=0, columnspan=2)

self.equipo = None

def crear_equipo(self):
    try:
        nombre_equipo = self.entradas[0].get()
        universidad = self.entradas[1].get()
        lenguaje = self.entradas[2].get()
        tamano = int(self.entradas[3].get())
        self.equipo = Equipo(nombre_equipo, universidad, lenguaje, tamano)
        self.resultado.config(text="Equipo creado con éxito.")
        self.boton_agregar.config(state=tk.NORMAL)
    except Exception as e:
        messagebox.showerror("Error", str(e))

def agregar_programador(self):
    if not self.equipo:
        return
    try:
        nombre = self.entradas[4].get()
        apellidos = self.entradas[5].get()
        self.equipo.agregar_programador(nombre, apellidos)
        if self.equipo.esta_completo():
            self.resultado.config(text="Equipo completo con éxito.")
            self.boton_agregar.config(state=tk.DISABLED)
        else:
            self.resultado.config(text=f"Programador añadido. Faltan {self.equipo.tamano - len(self.equipo.programadores)} programadores")
    except Exception as e:
        messagebox.showerror("Error", str(e))

# Ejecutar la aplicación
if __name__ == "__main__":
    root = tk.Tk()
    app = InterfazEquipo(root)
    root.mainloop()

```

## Ejercicio 5

Se tiene un archivo de texto denominado prueba.txt en una cierta localización en un sistema de archivos. Se requiere desarrollar un programa que lea dicho archivo de texto utilizando un flujo de bytes que muestre los contenidos del archivo en pantalla.

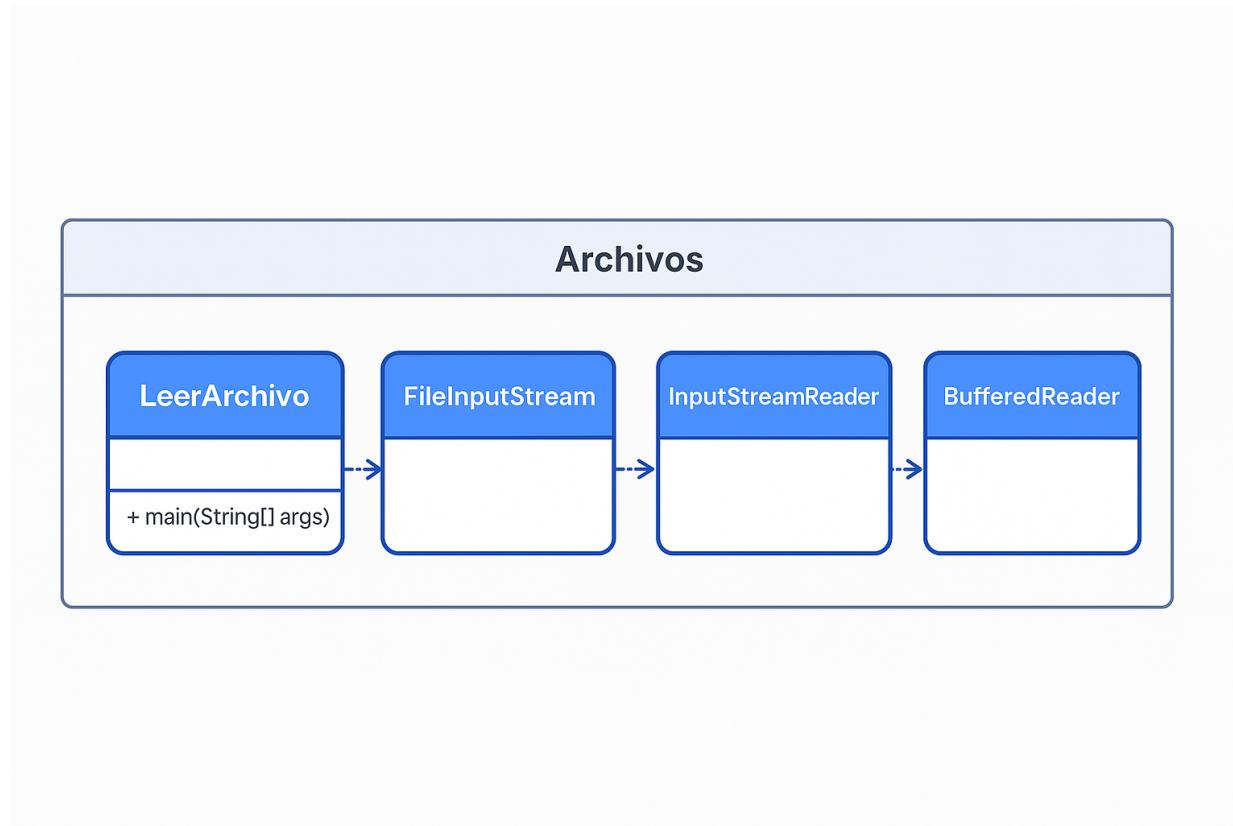


Figura 13: Diagrama de clases



Figura 14: Interfaz de usuario

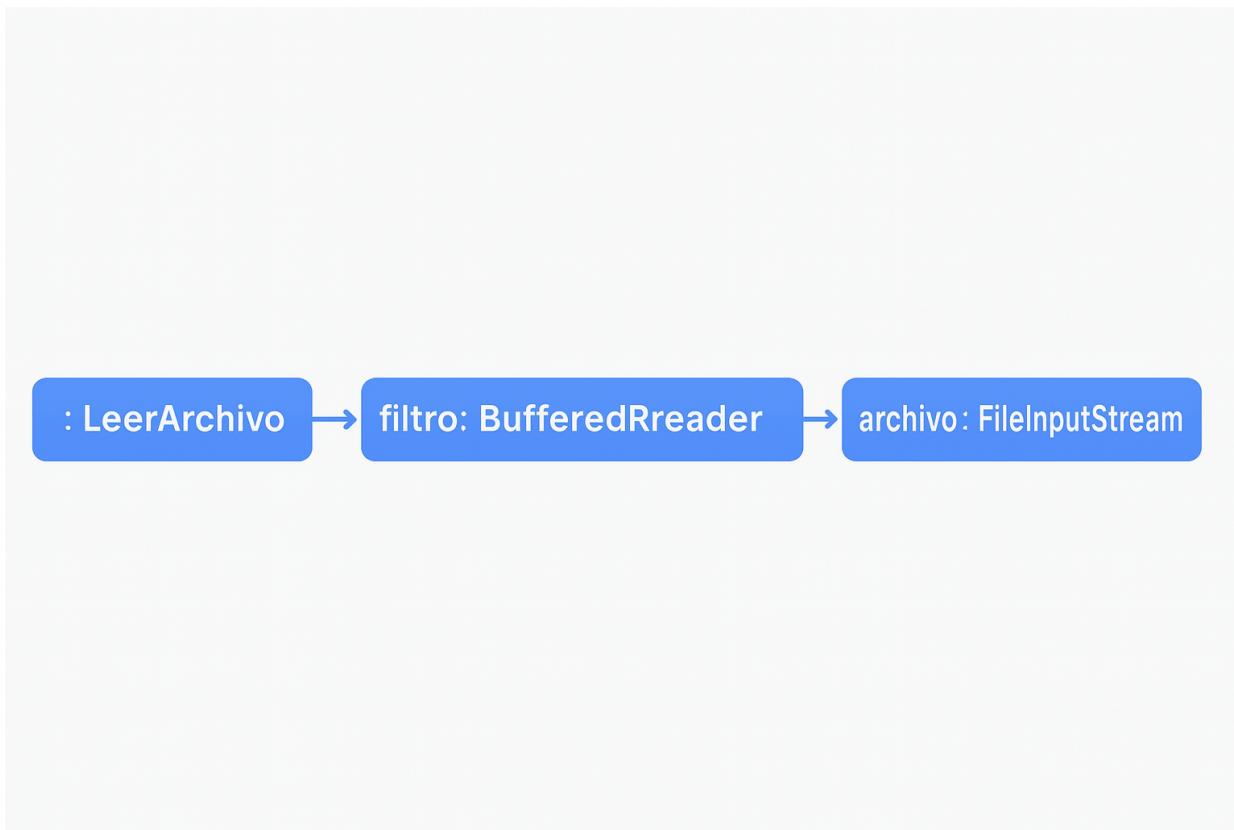


Figura 15: Diagrama de Usos

```

import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter.scrolledtext import ScrolledText

def abrir_archivo():
    archivo_path = filedialog.askopenfilename(
        title="Selecciona un archivo de texto",
        filetypes=[("Archivos de texto", "*.txt")]
    )

    if archivo_path:
        try:
            with open(archivo_path, "r", encoding="utf-8") as archivo:
                contenido = archivo.read()
                texto_area.delete("1.0", tk.END) # Borra contenido previo
                texto_area.insert(tk.END, contenido) # Muestra nuevo contenido
        except Exception as e:
            messagebox.showerror("Error", f"No se pudo leer el archivo:\n{e}")

```

```
# Crear ventana principal
ventana = tk.Tk()
ventana.title("Lector de Archivos de Texto")
ventana.geometry("600x400")

# Botón para abrir archivo
boton_abrir = tk.Button(ventana, text="Abrir archivo .txt", command=abrir_archivo)
boton_abrir.pack(pady=10)

# Área de texto con scroll
texto_area = ScrolledText(ventana, wrap=tk.WORD, width=70, height=20)
texto_area.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

# Ejecutar la ventana
ventana.mainloop()
```