

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

Ejercicio 3	3
Ejercicio 4	7
Ejercicio 5	12

Listado de Figuras

1	Diagrama de clases	4
2	Interfaz usuario	5
3	Diagrama de Usos	5
4	Diagrama de clases	8
5	Interfaz usuario	8
6	Diagrama de usos	9
7	Diagrama de clases	12
8	Interfaz de usuario	13
9	Diagrama de Usos	14

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.

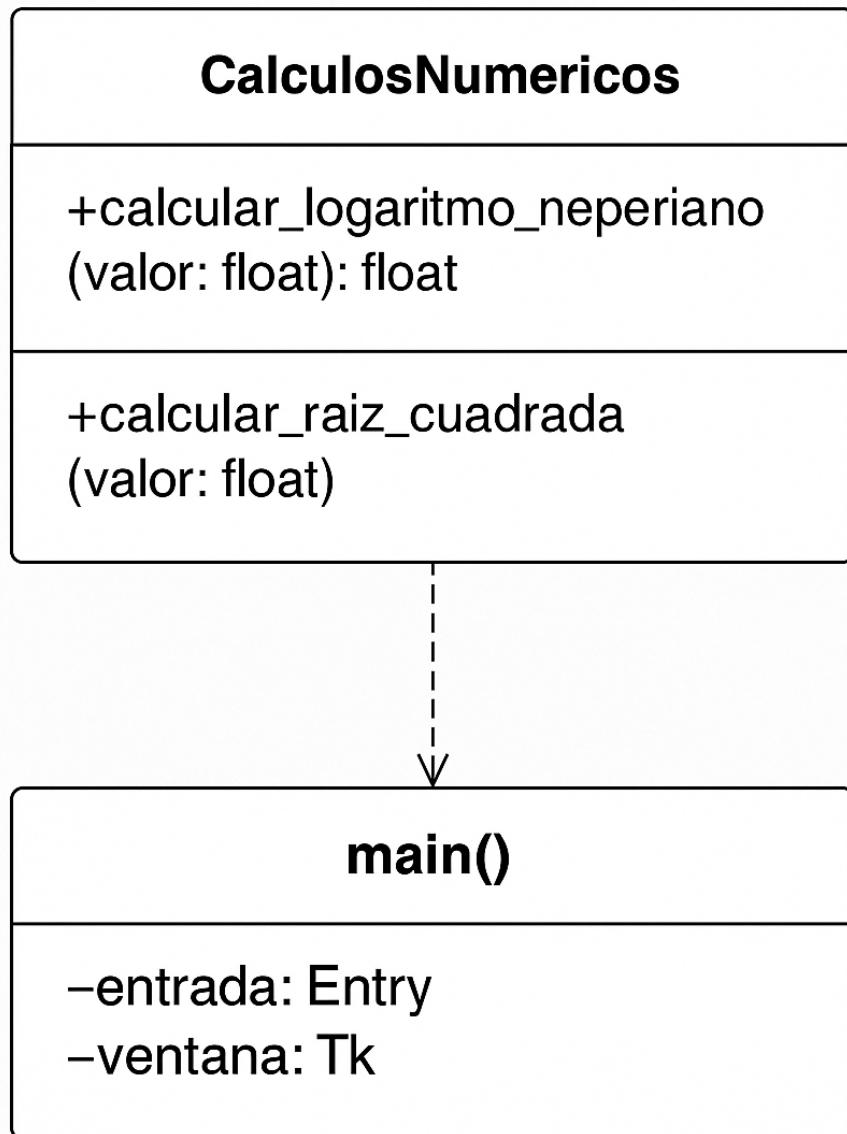


Figura 1: Diagrama de clases

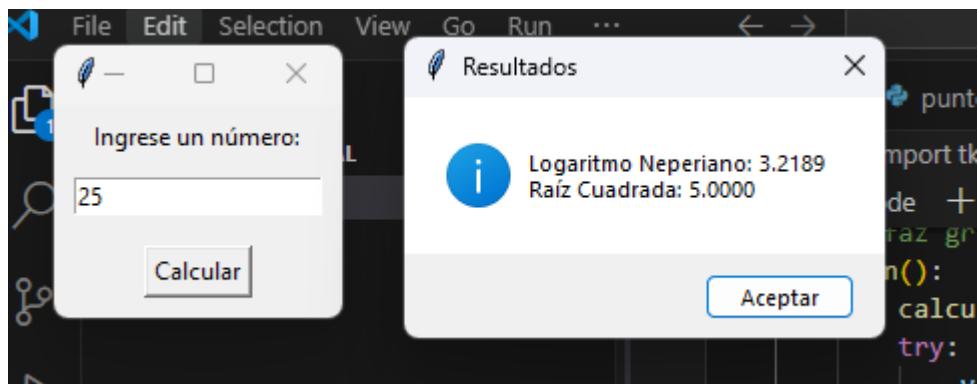


Figura 2: Interfaz usuario

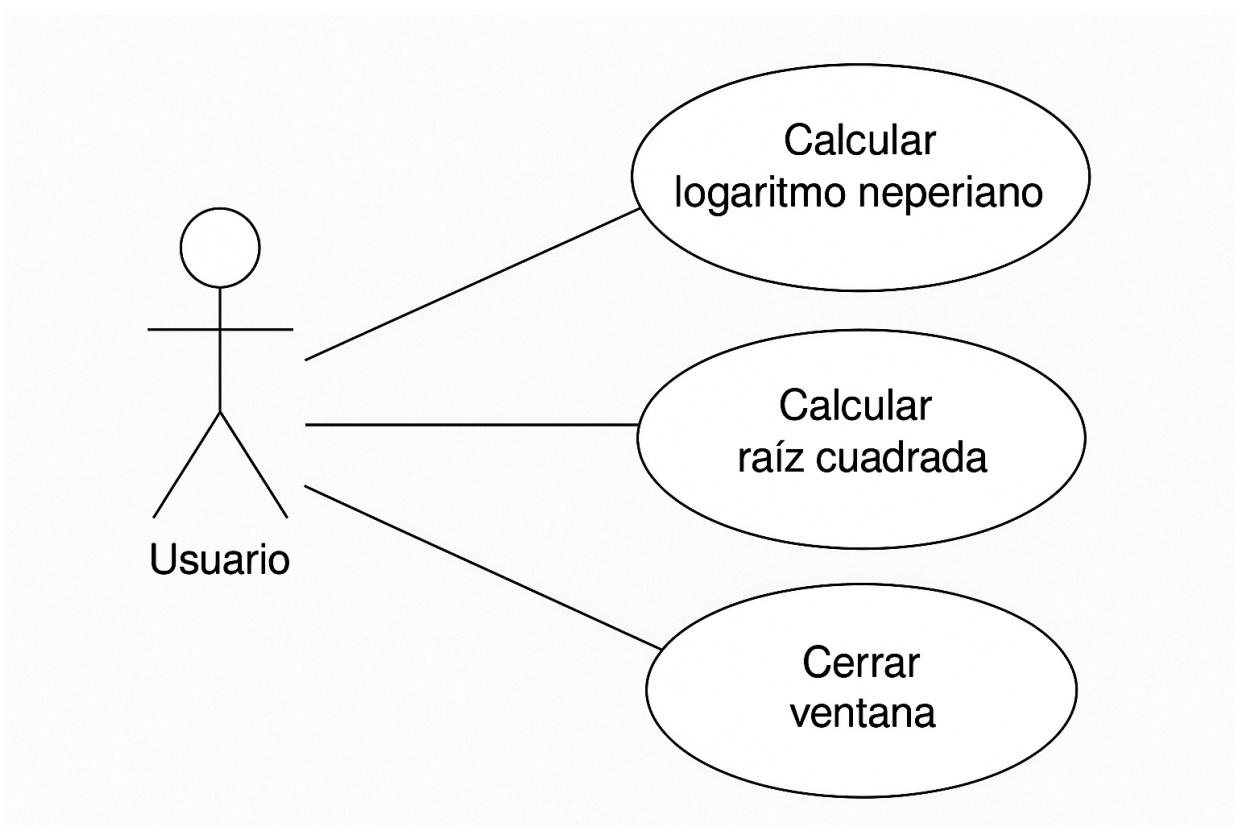


Figura 3: Diagrama de Usos

```
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.

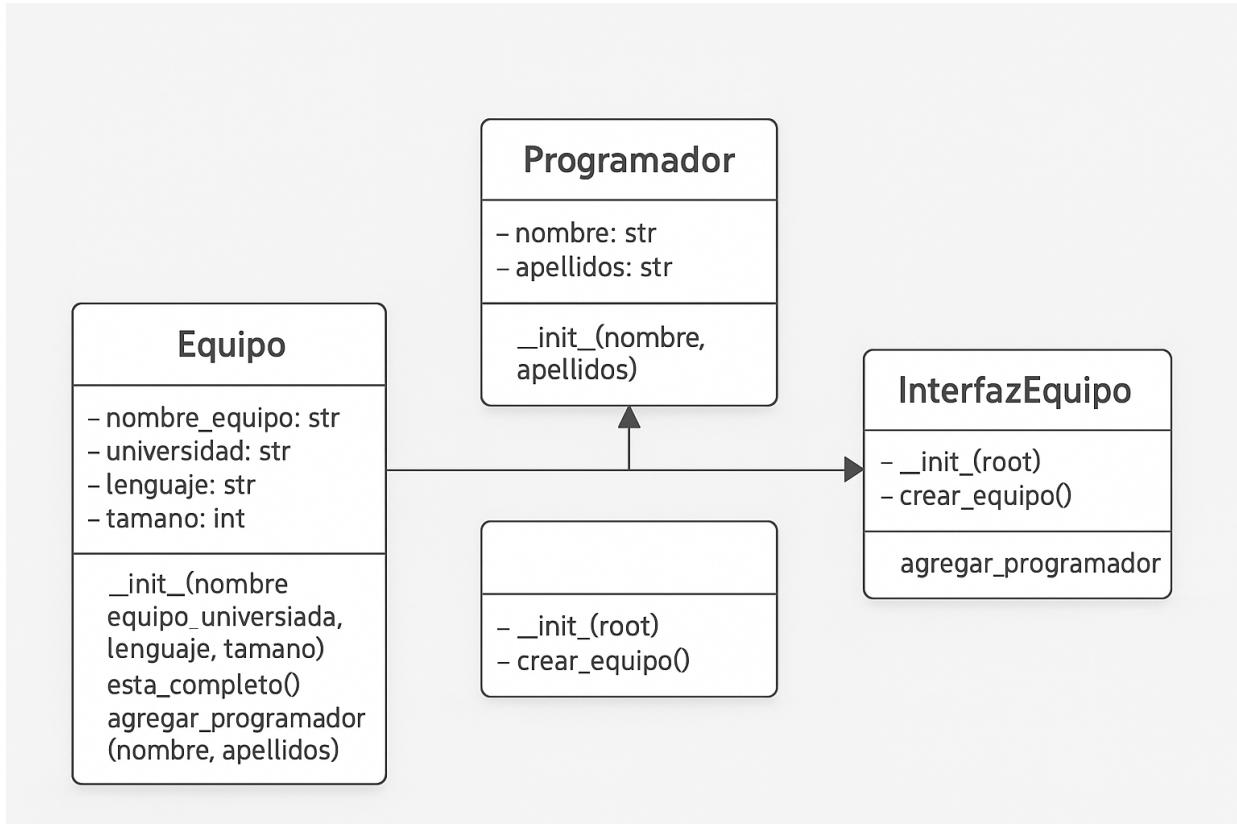


Figura 4: Diagrama de clases

The screenshot shows a user interface window titled "Registro de Equipo de Programación". The interface consists of several input fields and buttons:

- Text input fields for "Nombre del equipo", "Universidad", "Lenguaje", and "Tamaño (2-3)".
- Text input fields for "Nombre del prog" and "Apellidos del programador".
- Buttons labeled "Crear equipo" and "Agregar programador".

Figura 5: Interfaz usuario

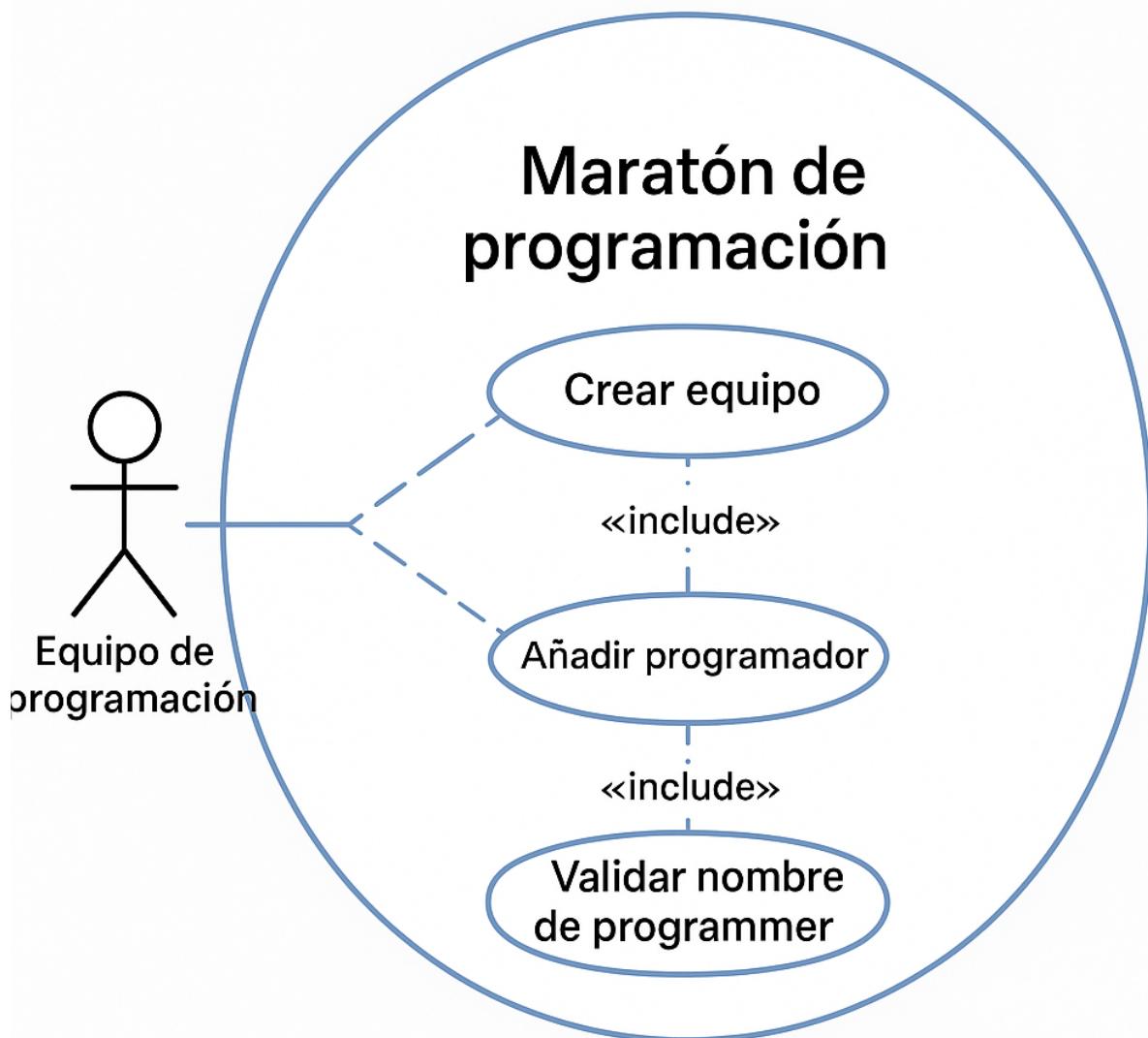


Figura 6: Diagrama de usos

```

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.

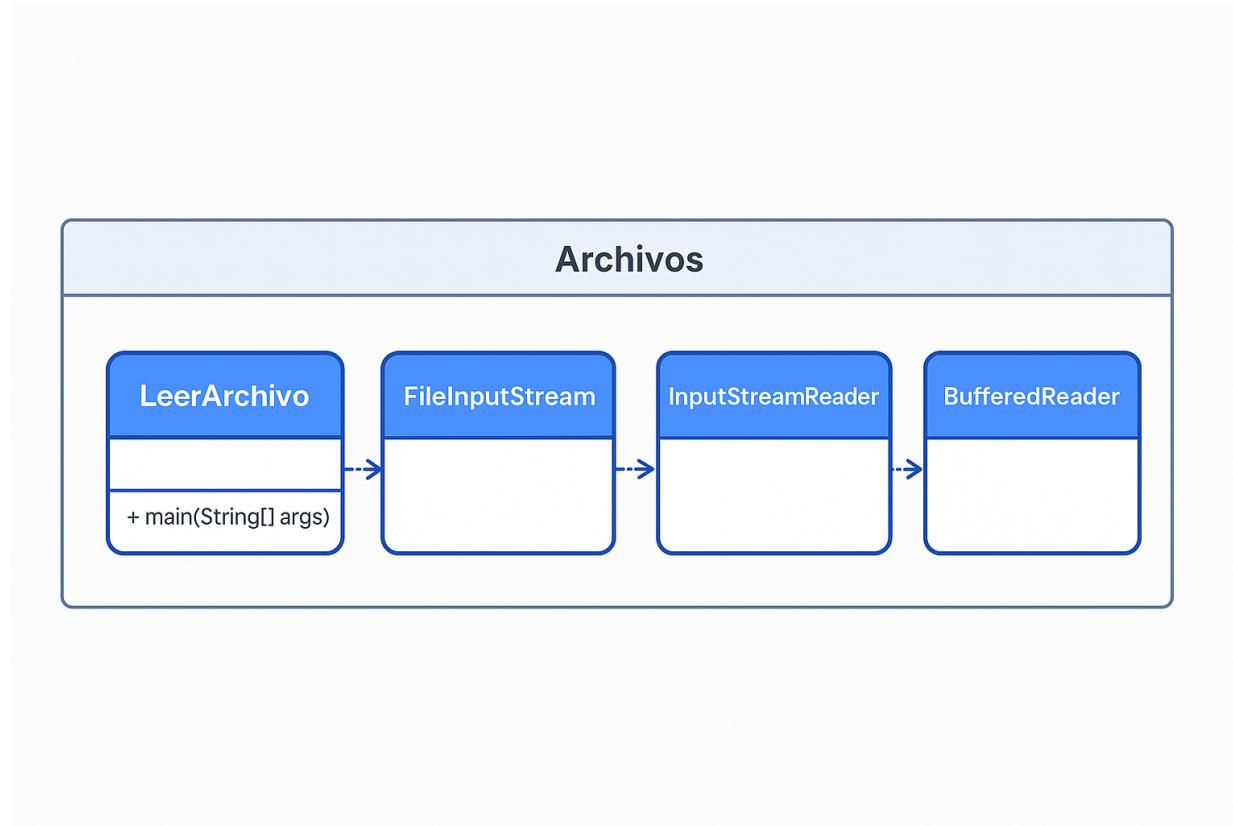


Figura 7: Diagrama de clases



Figura 8: Interfaz de usuario

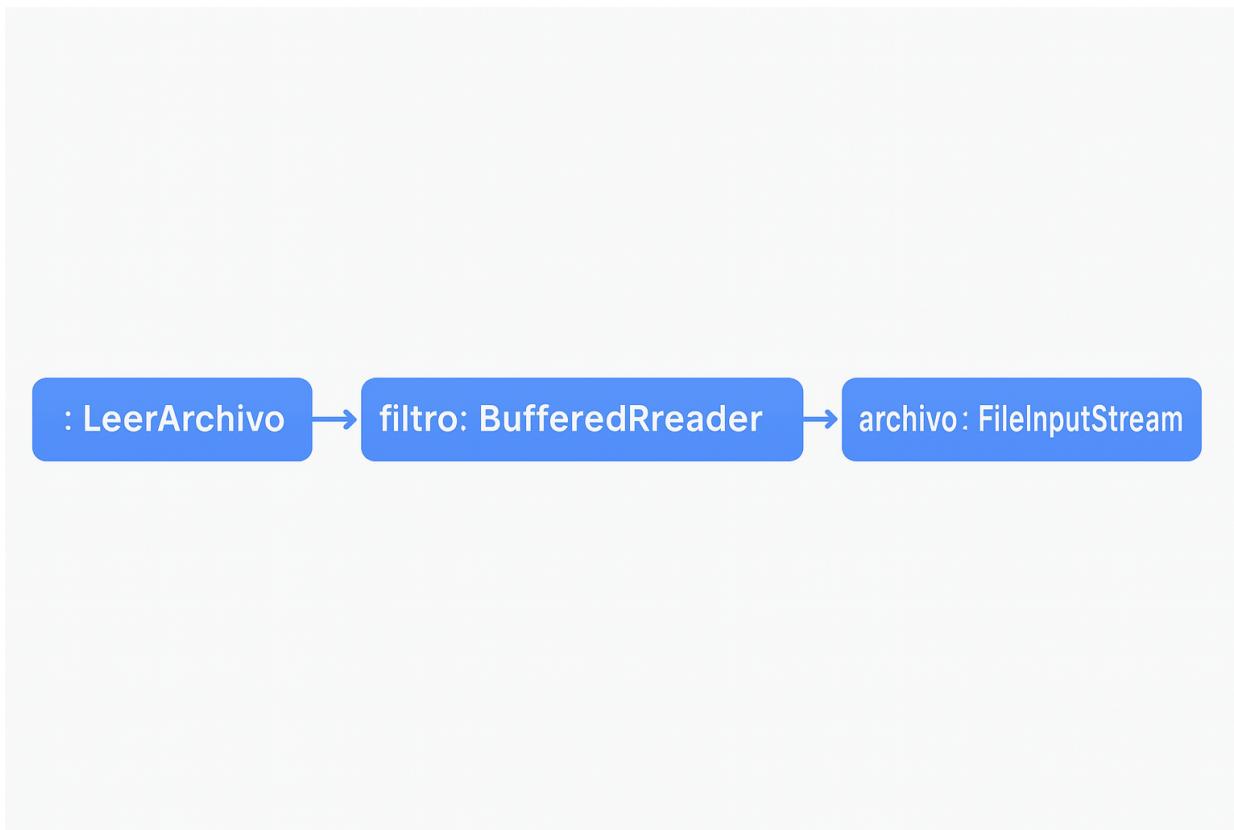


Figura 9: Diagrama de Usos

```

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()
```