Universidad Nacional de Colombia
Sede Medellín

Facultad de Ciencias

Actividad 5

Estudiantes

Juan Felipe Muriel Mosquera

Maria Fernanda Valencia Jimenez

Docente

Walter Hugo Arboleda Mazo

Julio 2025

Índice

1. Ejercicio página 400

1.1. Código fuente

```
import tkinter as tk
from tkinter import scrolledtext, messagebox

class PruebaExcepciones:
    def __init__(self, output_widget):
        self.output_widget = output_widget

    def ejecutar_primer_bloque(self):
```

```
        self.output_widget.insert(tk.END, "--- Ejecutando
            Primer Bloque ---\n")
        try:
            self.output_widget.insert(tk.END, "Ingresando
                al primer try\n")
            cociente = 10000 / 0
            self.output_widget.insert(tk.END, "Después de
                la división\n\n")
        except ZeroDivisionError:
            self.output_widget.insert(tk.END, "Excepción:
                División por cero\n")
        finally:
            self.output_widget.insert(tk.END, "Ingresando
                al primer finally\n")
        self.output_widget.insert(tk.END,
            "--------------------------------\n\n")
        self.output_widget.see(tk.END)

    def ejecutar_segundo_bloque(self):
        self.output_widget.insert(tk.END, "--- Ejecutando
            Segundo Bloque ---\n")
        try:
            self.output_widget.insert(tk.END, "Ingresando
                al segundo try\n")
            objeto = None
            objeto.test()
            self.output_widget.insert(tk.END, "Imprimiendo
                 objeto\n")
        except ZeroDivisionError:
            self.output_widget.insert(tk.END, "Excepción:
                División por cero ( No  se esperaba aquí!)\
                n")
        except AttributeError:
            self.output_widget.insert(tk.END, "Excepción:
                Ocurrió' un AttributeError (el objeto es
                None)\n")
        except Exception:
            self.output_widget.insert(tk.END, "Excepción:
                Ocurrió' una excepción general\n")
        finally:
            self.output_widget.insert(tk.END, "Ingresando
                al segundo finally\n")
        self.output_widget.insert(tk.END,
            "--------------------------------\n\n")
        self.output_widget.see(tk.END)

class AplicacionExcepciones:
    def __init__(self, master):
```

```
        self.master = master
        master.title("Demostració'n de Excepciones")
        master.geometry("700x400")
        master.resizable(False, False)

        self.output_area = scrolledtext.ScrolledText(
            master, wrap=tk.WORD, width=70, height=20, font
            =("Consolas", 10))
        self.output_area.pack(pady=10, padx=10, fill="both
            ", expand=True)

        self.prueba_excepciones = PruebaExcepciones(self.
            output_area)

        frame_botones = tk.Frame(master)
        frame_botones.pack(pady=10)

        self.btn_primer_bloque = tk.Button(
            frame_botones,
            text="Ejecutar Primer Bloque (Divisió'n por
                Cero)",
            command=self.prueba_excepciones.
                ejecutar_primer_bloque,
            font=("Arial", 10),
            bg="#FFC107"
        )
        self.btn_primer_bloque.pack(side=tk.LEFT, padx=10)

        self.btn_segundo_bloque = tk.Button(
            frame_botones,
            text="Ejecutar Segundo Bloque (Objeto Nulo)",
            command=self.prueba_excepciones.
                ejecutar_segundo_bloque,
            font=("Arial", 10),
            bg="#2196F3"
        )
        self.btn_segundo_bloque.pack(side=tk.LEFT, padx
            =10)

        self.btn_limpiar = tk.Button(
            frame_botones,
            text="Limpiar Salida",
            command=self.limpiar_salida,
            font=("Arial", 10),
            bg="#9E9E9E"
        )
        self.btn_limpiar.pack(side=tk.LEFT, padx=10)
```

```
    def limpiar_salida(self):
        self.output_area.delete(1.0, tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    app = AplicacionExcepciones(root)
    root.mainloop()
```

1.2. Diagrama de clase

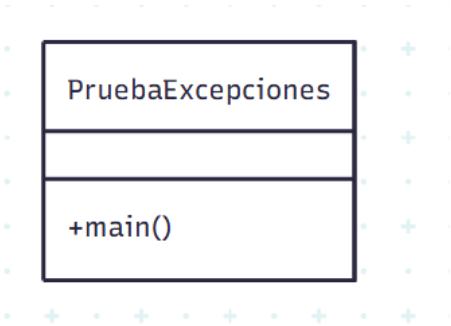

Figura 1: Diagrama de clase

1.3. Casos de uso

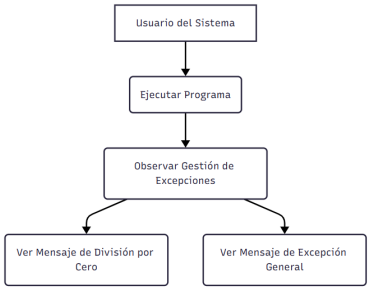

Figura 2: Casos de uso

1.4. Interfaz de usuario

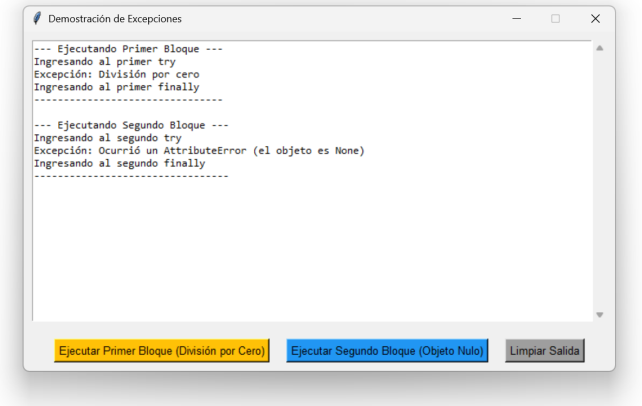

Figura 3: Interfaz de usuario

2. Ejercicio página 406

2.1. Código fuente

```
import tkinter as tk
from tkinter import messagebox

class Vendedor:
    def __init__(self, nombre, apellidos):
        self.nombre = nombre
        self.apellidos = apellidos
        self.edad = 0

    def imprimir(self):
        return (f"Nombre del vendedor: {self.nombre}\n"
                f"Apellidos del vendedor: {self.apellidos
                    }\n"
                f"Edad del vendedor: {self.edad}")

    def verificar_edad(self, edad):
        if not isinstance(edad, int):
            raise ValueError("La edad debe ser un mí'mero
                entero.")
        if edad < 18:
            raise ValueError("El vendedor debe ser mayor
                de 18 años.")
        elif 0 <= edad <= 120:
```

```
            self.edad = edad
        else:
            raise ValueError("La edad no puede ser
                negativa ni mayor a 120.")

class AplicacionVendedor:
    def __init__(self, master):
        self.master = master
        master.title("Registro de Vendedor")
        master.geometry("400x350")
        master.resizable(False, False)

        self.vendedor = None

        self.frame_datos = tk.LabelFrame(master, text="
            Datos del Vendedor", padx=10, pady=10)
        self.frame_datos.pack(pady=10, padx=10, fill="x")

        tk.Label(self.frame_datos, text="Nombre:").grid(
            row=0, column=0, sticky="w", pady=5)
        self.entry_nombre = tk.Entry(self.frame_datos,
            width=30)
        self.entry_nombre.grid(row=0, column=1, pady=5)

        tk.Label(self.frame_datos, text="Apellidos:").grid
            (row=1, column=0, sticky="w", pady=5)
        self.entry_apellidos = tk.Entry(self.frame_datos,
            width=30)
        self.entry_apellidos.grid(row=1, column=1, pady=5)

        tk.Label(self.frame_datos, text="Edad:").grid(row
            =2, column=0, sticky="w", pady=5)
        self.entry_edad = tk.Entry(self.frame_datos, width
            =30)
        self.entry_edad.grid(row=2, column=1, pady=5)

        self.btn_registrar = tk.Button(self.frame_datos,
            text="Registrar Vendedor", command=self.
            registrar_vendedor)
        self.btn_registrar.grid(row=3, column=0,
            columnspan=2, pady=10)

        self.frame_resumen = tk.LabelFrame(master, text="
            Resumen del Vendedor", padx=10, pady=10)
        self.frame_resumen.pack(pady=10, padx=10, fill="
            both", expand=True)

        self.label_resumen = tk.Label(self.frame_resumen,
```

```
            text="Ingrese los datos para ver el resumen.",
            justify=tk.LEFT)
        self.label_resumen.pack(fill="both", expand=True)

    def registrar_vendedor(self):
        nombre = self.entry_nombre.get().strip()
        apellidos = self.entry_apellidos.get().strip()
        edad_str = self.entry_edad.get().strip()

        if not nombre or not apellidos or not edad_str:
            messagebox.showwarning("Campos Vacíos", "Por
                favor, complete todos los campos.")
            return

        try:
            edad = int(edad_str)
            self.vendedor = Vendedor(nombre, apellidos)
            self.vendedor.verificar_edad(edad)

            self.label_resumen.config(text=self.vendedor.
                imprimir())
            messagebox.showinfo("É'xito", "Vendedor
                registrado correctamente.")
        except ValueError as e:
            messagebox.showerror("Error de Validació'n",
                str(e))
            self.label_resumen.config(text="Error al
                registrar: " + str(e))
        except Exception as e:
            messagebox.showerror("Error Inesperado", f"
                Ocurrió' un error inesperado: {e}")
            self.label_resumen.config(text="Error
                inesperado: " + str(e))

if __name__ == "__main__":
    root = tk.Tk()
    app = AplicacionVendedor(root)
    root.mainloop()
```

2.2. Diagrama de clase

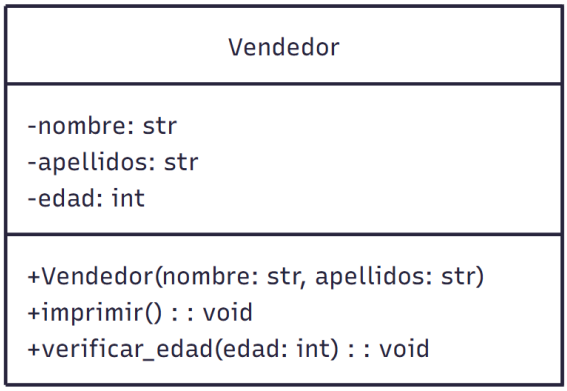

Figura 4: Diagrama de clase

2.3. Casos de uso

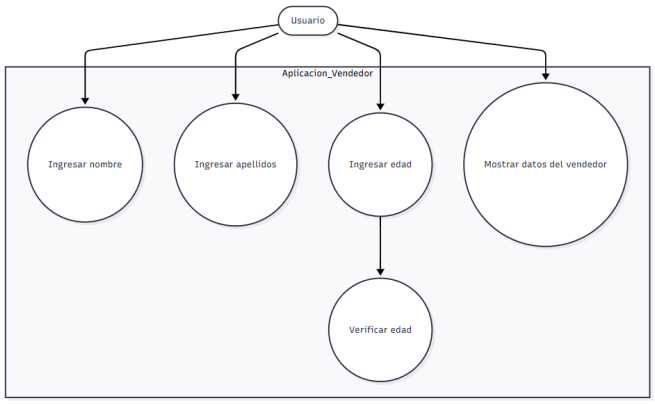

Figura 5: Casos de uso

2.4. Interfaz de usuario

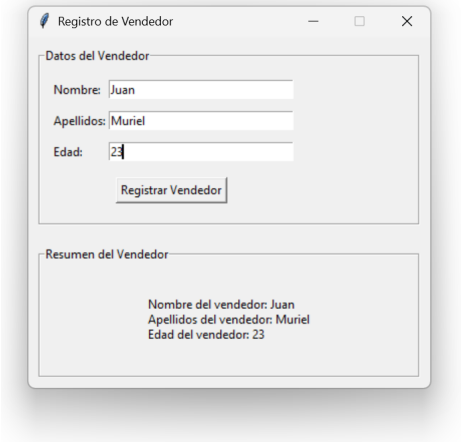

Figura 6: Interfaz de usuario

3. Ejercicio página 412

3.1. Código fuente

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

class CalculosNumericos:
    @staticmethod
    def calcular_logaritmo_neperiano(valor):
        try:
            if not isinstance(valor, (int, float)):
                raise TypeError("El valor ingresado debe
                    ser numérico")
            if valor < 0:
                raise ValueError("El valor debe ser un mú'
                    mero positivo")

            resultado = math.log(valor)
            return f"Logaritmo Neperiano = {resultado}"
        except ValueError as e:
            return f"Error: {e} para calcular el logaritmo
                ."
```

```
        except TypeError as e:
            return f"Error: {e} para calcular el logaritmo
                ."

    @staticmethod
    def calcular_raiz_cuadrada(valor):
        try:
            if not isinstance(valor, (int, float)):
                raise TypeError("El valor ingresado debe
                    ser numérico")
            if valor < 0:
                raise ValueError("El valor debe ser un mú'
                    mero positivo")

            resultado = math.sqrt(valor)
            return f"Raíz Cuadrada = {resultado}"
        except ValueError as e:
            return f"Error: {e} para calcular la raíz
                cuadrada."
        except TypeError as e:
            return f"Error: {e} para calcular la raíz
                cuadrada."

class AplicacionCalculos:
    def __init__(self, master):
        self.master = master
        master.title("Calculadora Numérica")
        master.geometry("400x250")
        master.resizable(False, False)

        self.frame_input = tk.LabelFrame(master, text="
            Ingresar Valor", padx=10, pady=10)
        self.frame_input.pack(pady=10, padx=10, fill="x")

        tk.Label(self.frame_input, text="Valor Numérico:")
            .grid(row=0, column=0, sticky="w", pady=5)
        self.entry_valor = tk.Entry(self.frame_input,
            width=30)
        self.entry_valor.grid(row=0, column=1, pady=5)

        self.btn_calcular = tk.Button(self.frame_input,
            text="Calcular", command=self.realizar_calculos
            )
        self.btn_calcular.grid(row=1, column=0, columnspan
            =2, pady=10)

        self.frame_resultados = tk.LabelFrame(master, text
            ="Resultados", padx=10, pady=10)
```

```
        self.frame_resultados.pack(pady=10, padx=10, fill
            ="both", expand=True)

        self.label_log = tk.Label(self.frame_resultados,
            text="Logaritmo Neperiano: ", anchor="w",
            justify=tk.LEFT)
        self.label_log.pack(fill="x", pady=5)

        self.label_sqrt = tk.Label(self.frame_resultados,
            text="Raíz Cuadrada: ", anchor="w", justify=tk.
            LEFT)
        self.label_sqrt.pack(fill="x", pady=5)

    def realizar_calculos(self):
        valor_str = self.entry_valor.get().strip()

        try:
            valor = float(valor_str)

            resultado_log = CalculosNumericos.
                calcular_logaritmo_neperiano(valor)
            self.label_log.config(text=resultado_log)

            resultado_sqrt = CalculosNumericos.
                calcular_raiz_cuadrada(valor)
            self.label_sqrt.config(text=resultado_sqrt)

        except ValueError:
            messagebox.showerror("Error de Entrada", "Por
                favor, ingresa un mú'mero válido.")
            self.label_log.config(text="Logaritmo
                Neperiano: ")
            self.label_sqrt.config(text="Raíz Cuadrada: ")
        except Exception as e:
            messagebox.showerror("Error Inesperado", f"
                Ocurrió' un error inesperado: {e}")
            self.label_log.config(text="Logaritmo
                Neperiano: ")
            self.label_sqrt.config(text="Raíz Cuadrada: ")

if __name__ == "__main__":
    root = tk.Tk()
    app = AplicacionCalculos(root)
    root.mainloop()
```

3.2. Diagrama de clase

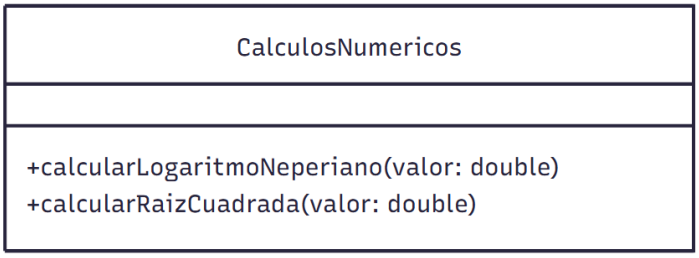

Figura 7: Diagrama de clase

3.3. Casos de uso

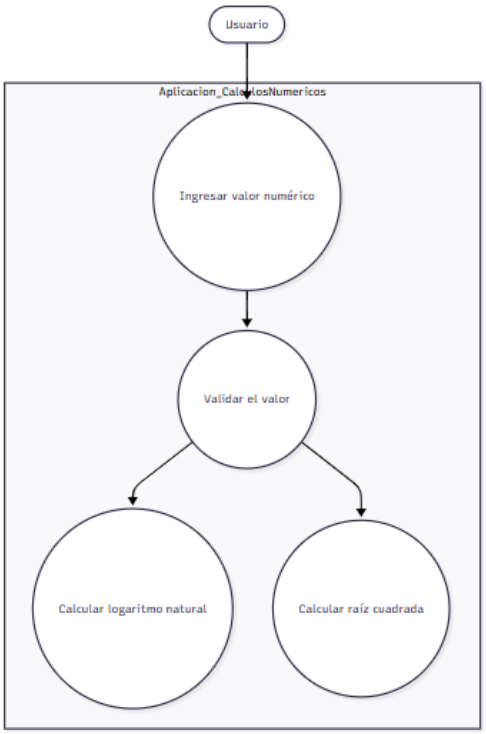

Figura 8: Casos de uso

3.4. Interfaz de usuario

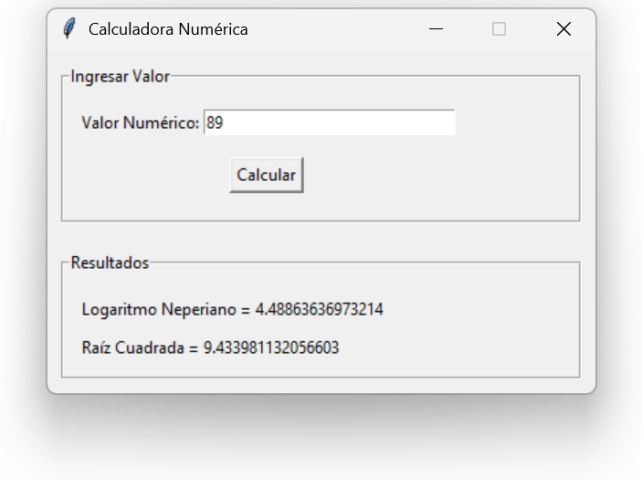

Figura 9: Interfaz de usuario

4. Ejercicio página 418

4.1. Código fuente

```
import tkinter as tk
from tkinter import messagebox, scrolledtext

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

class EquipoMaratonProgramacion:
    def __init__(self, nombre_equipo, universidad,
        lenguaje_programacion):
        self.nombre_equipo = nombre_equipo
        self.universidad = universidad
        self.lenguaje_programacion = lenguaje_programacion
        self.programadores = []
        self.tamaño_equipo = 0

    def esta_lleno(self):
```

```
        return len(self.programadores) >= 3

    def añadir(self, programador):
        if self.esta_lleno():
            raise Exception("El equipo está completo. No
                se pudo agregar programador.")
        self.programadores.append(programador)
        self.tamaño_equipo = len(self.programadores)

    @staticmethod
    def validar_campo(campo):
        if any(char.isdigit() for char in campo):
            raise Exception("El nombre no puede tener dí
                gitos.")
        if len(campo) > 20:
            raise Exception("La longitud no debe ser
                superior a 20 caracteres.")

class AplicacionEquipoMaraton:
    def __init__(self, master):
        self.master = master
        master.title("Registro de Equipo de Maratón de
            Programación")
        master.geometry("550x600")
        master.resizable(False, False)

        self.equipo = None
        self.num_programadores_registrados = 0

        self.frame_equipo = tk.LabelFrame(master, text="
            Datos del Equipo", padx=15, pady=10)
        self.frame_equipo.pack(pady=10, padx=15, fill="x")

        self.frame_programador = tk.LabelFrame(master,
            text="Agregar Programador", padx=15, pady=10)
        self.frame_programador.pack(pady=10, padx=15, fill
            ="x")

        self.frame_resumen = tk.LabelFrame(master, text="
            Resumen del Equipo", padx=15, pady=10)
        self.frame_resumen.pack(pady=10, padx=15, fill="
            both", expand=True)

        tk.Label(self.frame_equipo, text="Nombre del
            Equipo:").grid(row=0, column=0, sticky="w",
            pady=2)
        self.entrada_nombre_equipo = tk.Entry(self.
            frame_equipo, width=40)
```

```
        self.entrada_nombre_equipo.grid(row=0, column=1,
            pady=2)

        tk.Label(self.frame_equipo, text="Universidad:").
            grid(row=1, column=0, sticky="w", pady=2)
        self.entrada_universidad = tk.Entry(self.
            frame_equipo, width=40)
        self.entrada_universidad.grid(row=1, column=1,
            pady=2)

        tk.Label(self.frame_equipo, text="Lenguaje de
            Programació'n:").grid(row=2, column=0, sticky="w
            ", pady=2)
        self.entrada_lenguaje = tk.Entry(self.frame_equipo
            , width=40)
        self.entrada_lenguaje.grid(row=2, column=1, pady
            =2)

        self.btn_crear_equipo = tk.Button(self.
            frame_equipo, text="Crear Equipo", command=self
            .crear_equipo)
        self.btn_crear_equipo.grid(row=3, column=0,
            columnspan=2, pady=10)

        tk.Label(self.frame_programador, text="Nombre:").
            grid(row=0, column=0, sticky="w", pady=2)
        self.entrada_nombre_programador = tk.Entry(self.
            frame_programador, width=30)
        self.entrada_nombre_programador.grid(row=0, column
            =1, pady=2)

        tk.Label(self.frame_programador, text="Apellidos
            :").grid(row=1, column=0, sticky="w", pady=2)
        self.entrada_apellidos_programador = tk.Entry(self
            .frame_programador, width=30)
        self.entrada_apellidos_programador.grid(row=1,
            column=1, pady=2)

        self.btn_añadir_programador = tk.Button(self.
            frame_programador, text="Añadir Programador",
            command=self.añadir_programador, state=tk.
            DISABLED)
        self.btn_añadir_programador.grid(row=2, column=0,
            columnspan=2, pady=10)

        self.area_resumen = scrolledtext.ScrolledText(self
            .frame_resumen, width=60, height=15, state="
            disabled", wrap=tk.WORD)
```

```
        self.area_resumen.pack(pady=5, padx=5, fill="both
            ", expand=True)

        self.actualizar_resumen()

    def crear_equipo(self):
        nombre_equipo = self.entrada_nombre_equipo.get()
        universidad = self.entrada_universidad.get()
        lenguaje = self.entrada_lenguaje.get()

        if not nombre_equipo or not universidad or not
            lenguaje:
            messagebox.showwarning("Campos Vacíos", "Por
                favor, completa todos los campos del equipo
                .")
            return

        try:
            EquipoMaratonProgramacion.validar_campo(
                nombre_equipo)
            EquipoMaratonProgramacion.validar_campo(
                universidad)
            EquipoMaratonProgramacion.validar_campo(
                lenguaje)

            self.equipo = EquipoMaratonProgramacion(
                nombre_equipo, universidad, lenguaje)
            messagebox.showinfo("É'xito", "Equipo creado
                correctamente. Ahora puedes añadir
                programadores.")
            self.btn_crear_equipo.config(state=tk.DISABLED
                )
            self.btn_añadir_programador.config(state=tk.
                NORMAL)
            self.actualizar_resumen()
        except Exception as e:
            messagebox.showerror("Error al Crear Equipo",
                str(e))

    def añadir_programador(self):
        if not self.equipo:
            messagebox.showwarning("Error", "Primero debes
                 crear el equipo.")
            return

        if self.equipo.esta_lleno():
            messagebox.showinfo("Equipo Completo", "El
                equipo ya tiene 3 programadores. No se
```

```
            pueden añadir más.")
        self.btn_añadir_programador.config(state=tk.
            DISABLED)
        return

    nombre_prog = self.entrada_nombre_programador.get
        ()
    apellidos_prog = self.
        entrada_apellidos_programador.get()

    if not nombre_prog or not apellidos_prog:
        messagebox.showwarning("Campos Vacíos", "Por
            favor, ingresa el nombre y apellidos del
            programador.")
        return

    try:
        EquipoMaratonProgramacion.validar_campo(
            nombre_prog)
        EquipoMaratonProgramacion.validar_campo(
            apellidos_prog)

        programador = Programador(nombre_prog,
            apellidos_prog)
        self.equipo.añadir(programador)
        self.num_programadores_registrados += 1
        messagebox.showinfo("É'xito", f"Programador {
            nombre_prog} {apellidos_prog} añadido al
            equipo.")

        self.entrada_nombre_programador.delete(0, tk.
            END)
        self.entrada_apellidos_programador.delete(0,
            tk.END)

        self.actualizar_resumen()

        if self.equipo.esta_lleno():
            messagebox.showinfo("Equipo Completo", "
                El  equipo ha alcanzado el máximo de 3
                programadores!")
            self.btn_añadir_programador.config(state=
                tk.DISABLED)

    except Exception as e:
        messagebox.showerror("Error al Añadir
            Programador", str(e))
```

```
    def actualizar_resumen(self):
        self.area_resumen.config(state="normal")
        self.area_resumen.delete(1.0, tk.END)

        if self.equipo:
            self.area_resumen.insert(tk.END, "---
                Información del Equipo ---\n")
            self.area_resumen.insert(tk.END, f"Nombre del
                Equipo: {self.equipo.nombre_equipo}\n")
            self.area_resumen.insert(tk.END, f"Universidad
                : {self.equipo.universidad}\n")
            self.area_resumen.insert(tk.END, f"Lenguaje de
                 Programación: {self.equipo.
                lenguaje_programacion}\n\n")
            self.area_resumen.insert(tk.END, f"Integrantes
                 ({len(self.equipo.programadores)}/3):\n")
            if self.equipo.programadores:
                for prog in self.equipo.programadores:
                    self.area_resumen.insert(tk.END, f"- {
                        prog.nombre} {prog.apellidos}\n")
            else:
                self.area_resumen.insert(tk.END, "  Ningún
                     programador añadido aún.\n")
        else:
            self.area_resumen.insert(tk.END, "Aún no se ha
                 creado un equipo.\n")

        self.area_resumen.config(state="disabled")

if __name__ == "__main__":
    root = tk.Tk()
    app = AplicacionEquipoMaraton(root)
    root.mainloop()
```

4.2. Diagramas de clase y objeto

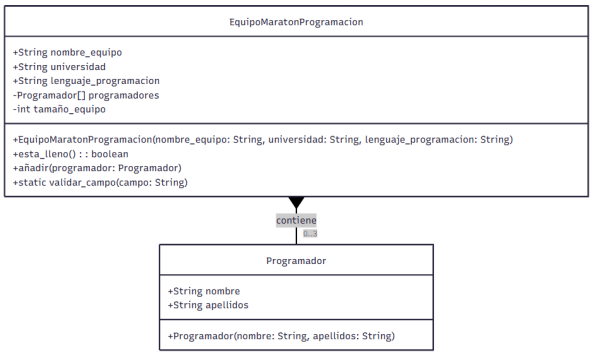

Figura 10: Diagrama de clase

Figura 11: Diagrama de objetos

4.3. Casos de uso

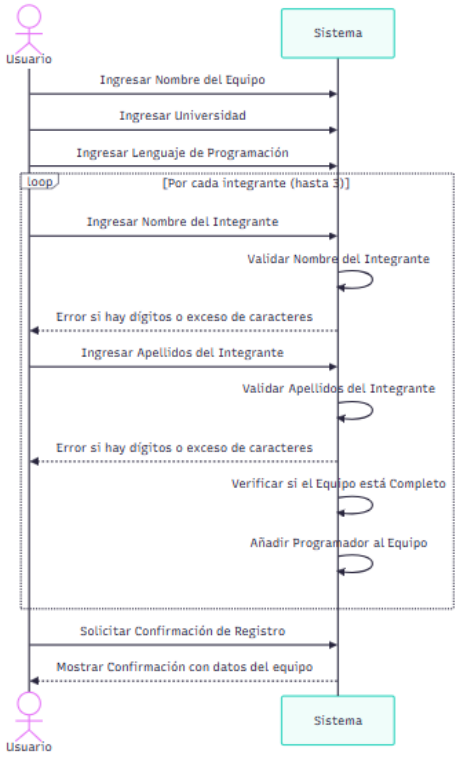

Figura 12: Casos de uso

4.4. Interfaz de usuario

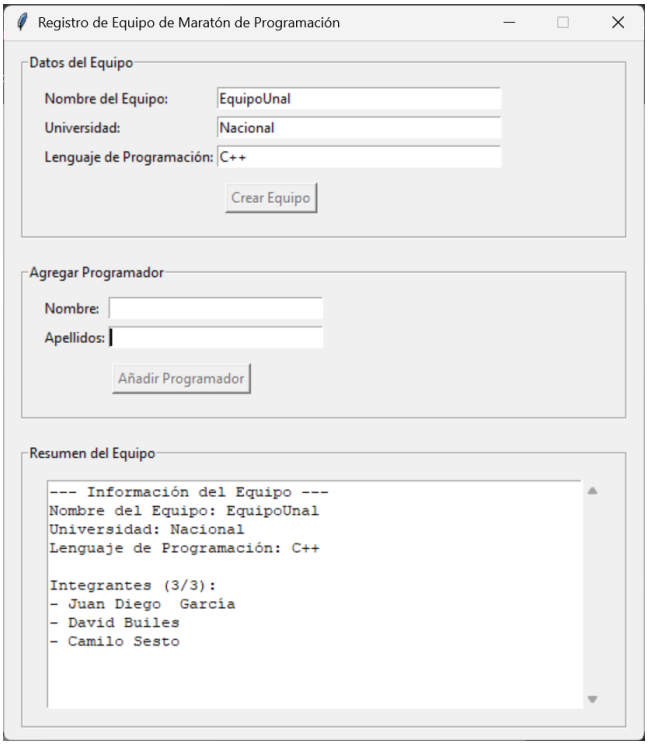

Figura 13: Interfaz de usuario

5. Ejercicio página 427

5.1. Código fuente

```
import tkinter as tk
from tkinter import messagebox

def cargar_archivo(nombre_archivo):
    try:
        with open(nombre_archivo, 'r', encoding='utf-8')
            as archivo:
            contenido = archivo.read()
            texto_area.delete(1.0, tk.END)  # Borra el
                contenido previo
            texto_area.insert(tk.END, contenido) # Inserta
                 el contenido del archivo
```

```
            etiqueta_estado.config(text=f"Archivo {
                nombre_archivo} cargado exitosamente.")
    except FileNotFoundError:
        messagebox.showerror("Error", f"No se pudo
            encontrar el archivo: '{nombre_archivo}'")
        etiqueta_estado.config(text=f"Error: Archivo {
            nombre_archivo} no encontrado.")
    except Exception as e:
        messagebox.showerror("Error", f"Ocurrió un error
            al leer el archivo: {e}")
        etiqueta_estado.config(text=f"Error al leer el
            archivo '{nombre_archivo}'.")

ventana = tk.Tk()
ventana.title("Lector de Archivos de Texto")
ventana.geometry("500x400")
etiqueta_estado = tk.Label(
    ventana,
    text="Cargando archivo...",
    font=("Arial", 10),
    fg="black",
)
etiqueta_estado.pack(pady=10)

frame_texto = tk.Frame(ventana, bd=2, relief=tk.SUNKEN)
frame_texto.pack(padx=10, pady=5, fill=tk.BOTH, expand=
    True)

scrollbar = tk.Scrollbar(frame_texto)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

texto_area = tk.Text(
    frame_texto,
    wrap="word",
    yscrollcommand=scrollbar.set,
    font=("Consolas", 11),
    bd=0,
    padx=5, pady=5
)
texto_area.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
scrollbar.config(command=texto_area.yview)

if __name__ == "__main__":
    nombre_archivo = "prueba.txt"
    ventana.after(100, cargar_archivo(nombre_archivo))
    ventana.mainloop()
```

5.2. Diagramas de clase y objeto

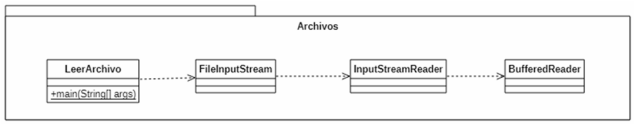

Figura 14: Diagrama de clase

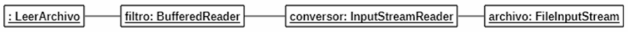

Figura 15: Diagrama de objetos

5.3. Casos de uso

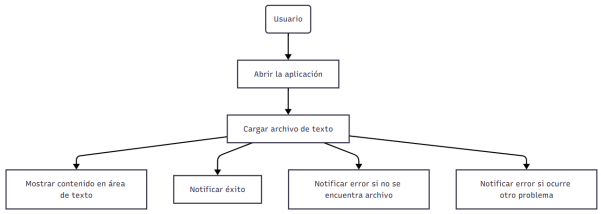

Figura 16: Casos de uso

5.4. Interfaz de usuario

Junto con el archivo .ipynb se encuentra el archivo prueba.txt acorde a el ejemplo de la interfaz de usuario

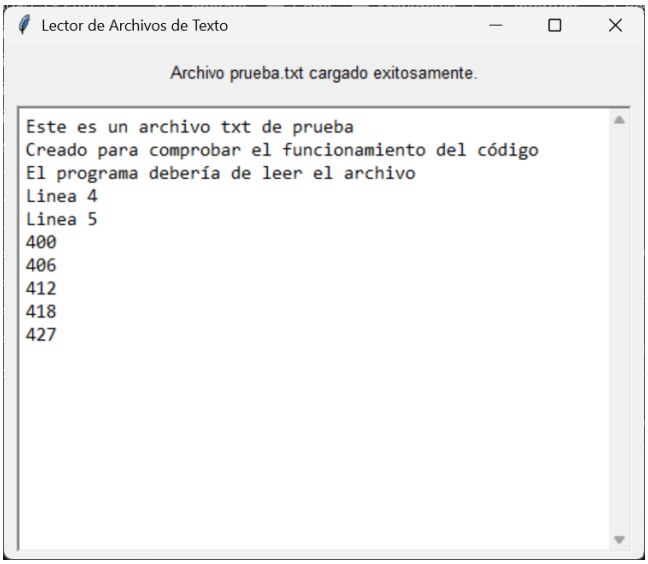

Figura 17: Interfaz de usuario