Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja

Estructura de datos aplicadas

Ruth Martínez

Examen

SAUL LUNA ESTÉVEZ

DESARROLLO DE SOFTWARE

4°B

PARCIAL 1

**1. Importaciones**

Se importan las bibliotecas necesarias para crear una interfaz gráfica (GUI) usando Tkinter y herramientas específicas para diálogos y mensajes.

**2. Clase Paciente**

Esta clase representa a un paciente individual.

* **\_\_init\_\_(self, nombre, edad, genero, telefono, diagnostico)**: Este es el constructor de la clase, donde se inicializan las propiedades del paciente: nombre, edad, género, teléfono y diagnóstico.
* Una captura de pantalla de una computadora

  Descripción generada automáticamente**\_\_str\_\_(self)**: Este método devuelve una representación en cadena del objeto Paciente para su fácil visualización.

**3. Clase RegistroPacientes**

Esta clase se utiliza para mantener un registro de pacientes en una lista.

* **\_\_init\_\_(self)**: Constructor que inicializa una lista vacía para almacenar pacientes.
* **agregar\_paciente(self, paciente)**: Método para agregar un paciente al registro.
* **buscar\_paciente(self, nombre, index=0)**: Método recursivo para buscar un paciente por nombre. Si el paciente no se encuentra, devuelve **None**.
* **buscar\_por\_diagnostico(self, diagnostico)**: Método para buscar y retornar una lista de pacientes que tienen un diagnóstico específico.
* **eliminar\_paciente(self, nombre)**: Método para eliminar un paciente del registro por su nombre.
* **estadisticas(self)**: Método que devuelve el total de pacientes y la edad promedio de los pacientes.

Texto

Descripción generada automáticamente

**4. Clase App**

Esta clase define la aplicación GUI y sus funcionalidades.

* **\_\_init\_\_(self, master)**: Constructor que inicializa la ventana principal y crea botones con acciones asociadas.
* **agregar\_paciente(self)**: Solicita la información del paciente a través de diálogos y luego lo agrega al registro.
* **buscar\_paciente\_nombre(self)**: Busca un paciente por su nombre y muestra la información o un mensaje de error si no se encuentra.
* **buscar\_paciente\_diagnostico(self)**: Busca pacientes por un diagnóstico específico y muestra la información correspondiente.
* **mostrar\_pacientes(self)**: Muestra todos los pacientes registrados.
* **eliminar\_paciente(self)**: Solicita el nombre del paciente a eliminar y luego lo elimina del registro.
* **estadisticas(self)**: Muestra las estadísticas, es decir, el total de pacientes y la edad promedio.

Una captura de pantalla de una computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

**5. Punto de entrada principal**

if \_\_name\_\_ == "\_\_main\_\_": root = tk.Tk() # Crea la ventana principal. app = App(root) # Crea la aplicación. root.mainloop() # Ejecuta el bucle principal de la aplicación.

Esta es la sección que se ejecuta cuando el script se ejecuta como un programa principal. Aquí, se crea una ventana principal **root**, luego se inicializa la aplicación GUI **App**, y finalmente se ejecuta el bucle principal para mantener la aplicación en funcionamiento.

**Resumen**: El código define una aplicación GUI para mantener un registro de pacientes. La aplicación permite agregar, buscar, mostrar, eliminar pacientes y ver estadísticas. Utiliza clases para organizar la lógica y estructurar el programa. La clase **Paciente** representa un paciente individual, la clase **RegistroPacientes** se encarga del manejo de pacientes en una lista, y la clase **App** gestiona la interfaz gráfica y su interacción con el usuario.

Una captura de pantalla de una computadora

Descripción generada automáticamente