Nombre: Jose Daniel Diaz Veloz
Matricula: al07138421
Profesora: Silvia Tello Zúñiga
Materia: Computación en Java

EVIDENCIA – COMPUTACIÓN EN JAVA

Descripción y requerimientos del programa:

El producto final será un programa que simulará un sistema de administración de citas con las siguientes funcionalidades:

- **Dar de alta doctores: se deberán poder dar de alta los doctores del consultorio médico, los datos básicos serán:**
- **Identificador único.**
- **Nombre completo.**
- **Especialidad.**

• Dar de alta pacientes: se deberán poder registrar los pacientes que acudan al consultorio médico, los datos
básicos serán:

- **Identificador único.**
- **Nombre completo.**

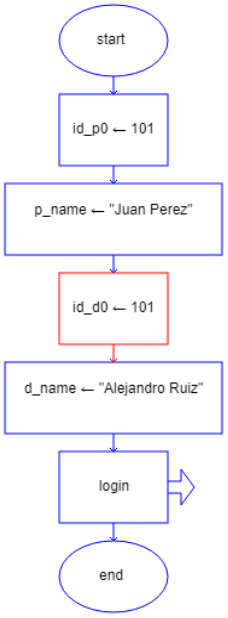
• Crear una cita con fecha y hora: se deberán poder crear múltiples citas, los datos básicos serán:

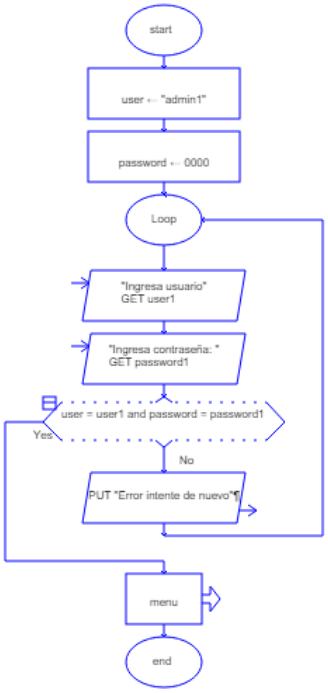
- **Identificador único.**
- **Fecha y hora de la cita.**
- **Motivo de la cita.**

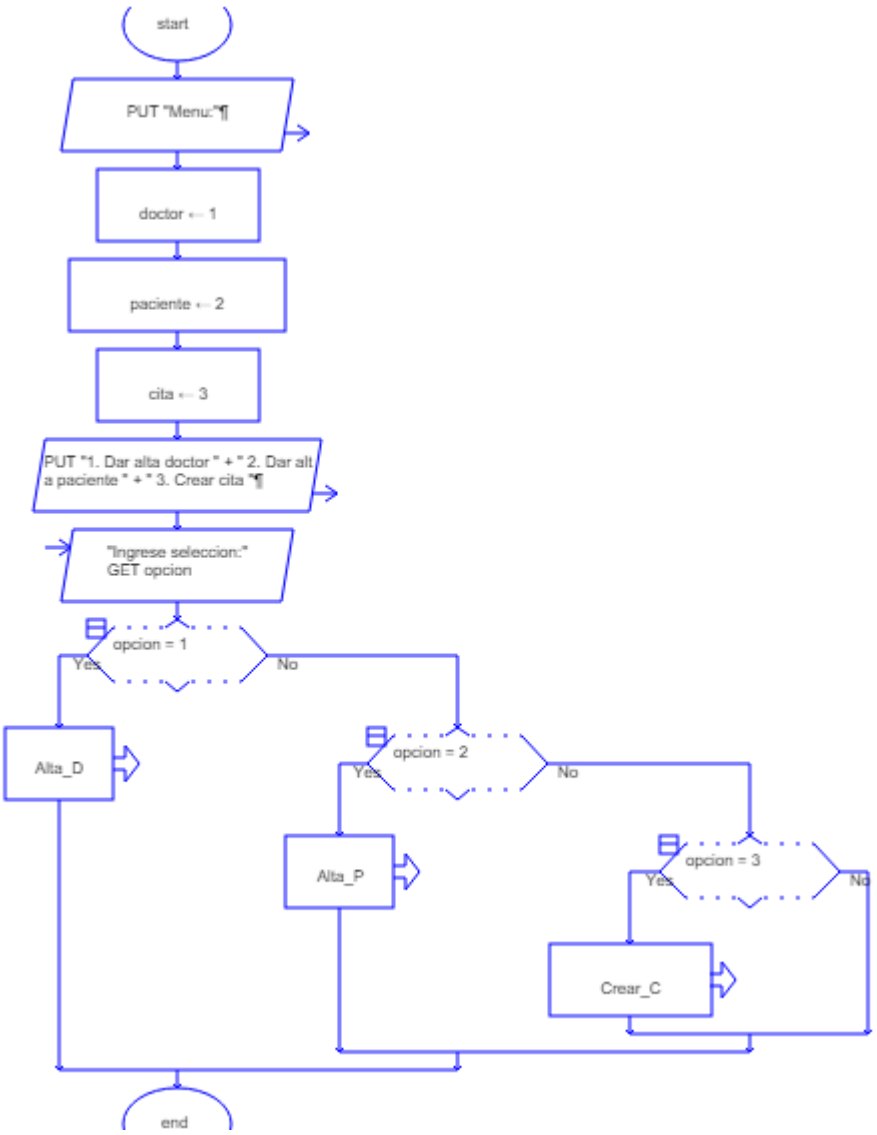
• Relacionar una cita con un doctor y un paciente: cada una de las citas creadas deberá estar relacionada con un doctor y un paciente.
• Tener control de acceso mediante administradores: solo ciertos usuarios podrán acceder al sistema mediante un identificador y una contraseña

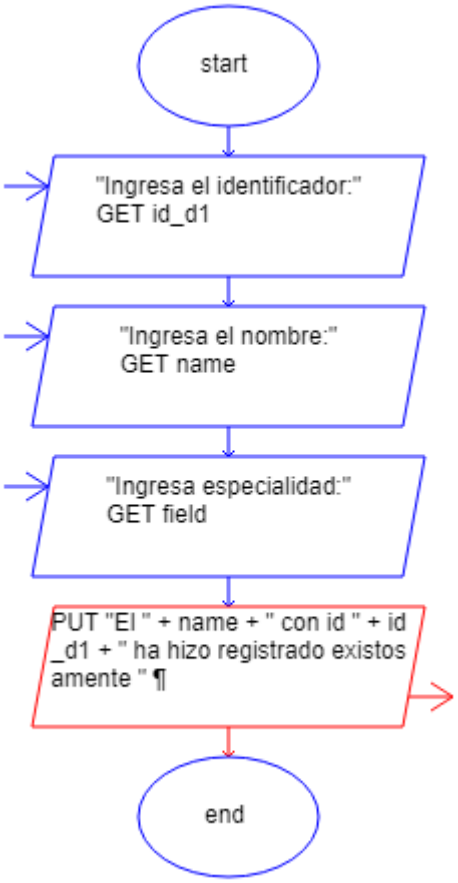
Link de Github: https://github.com/Veloz2/Evidencia---Computaci-n-en-JAVA.git
En este link se encuentran todos los documentos que sirvieron para la creación de este documento.

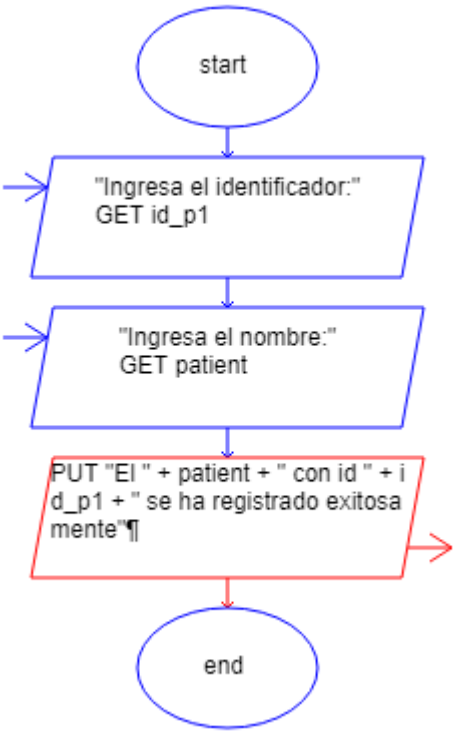
Diagrama del flujo (Raptor):

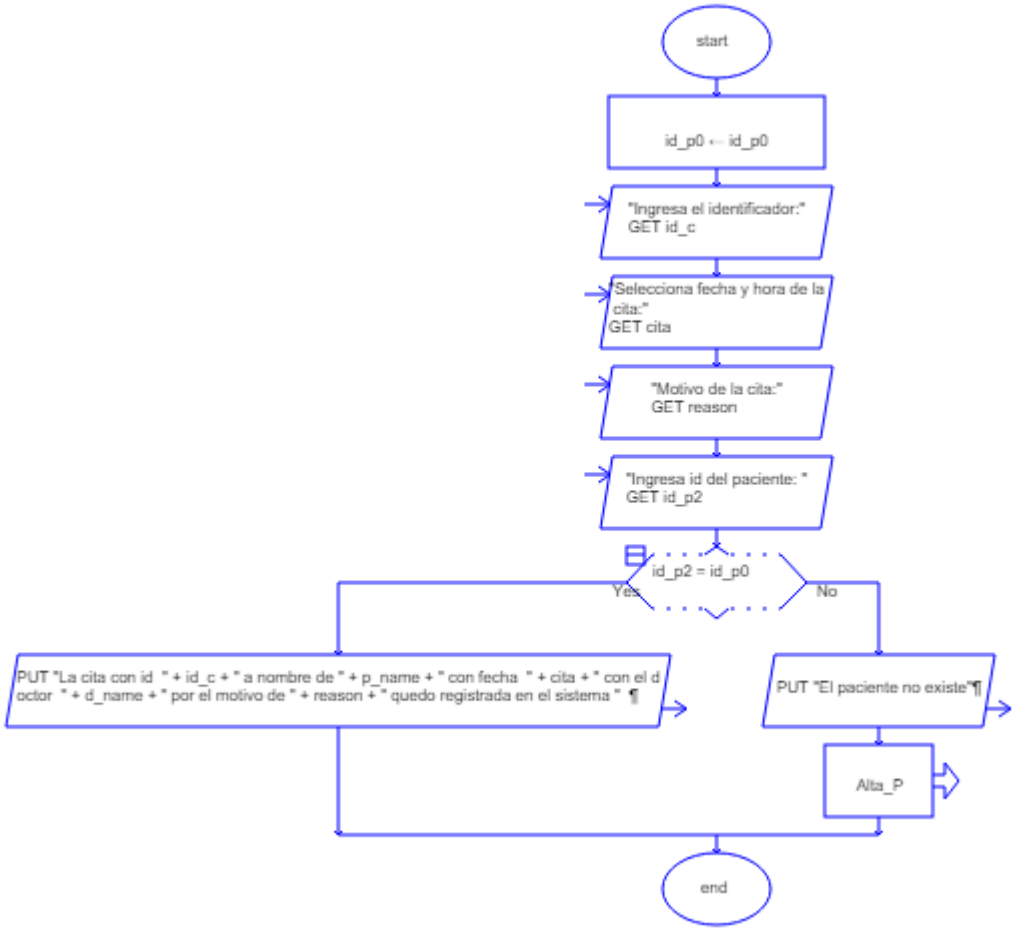





start
PUT "Menu:"¶
doctor <-- 1
paciente <-- 2
cita <-- 3
PUT "1. Dar alta doctor " + " 2. Dar alt
a paciente " + " 3. Crear cita "¶
"Ingrese seleccion:"
GET opcion
opcion = 1
Yes
No
Alta_D
opcion = 2
Yes
No
Alta_P
opcion = 3
Yes
No
Crear_C
end


start
"Ingresa el identificador:"
GET id_d1
"Ingresa el nombre:"
GET name
"Ingresa especialidad:"
GET field
PUT "El " + name + " con id " + id
_d1 + " ha hizo registrado existos
amente " ¶
end


start
"Ingresa el identificador:"
GET id_p1
"Ingresa el nombre:"
GET patient
PUT "El " + patient + " con id " + i
d_p1 + " se ha registrado exitosa
mente"¶
end

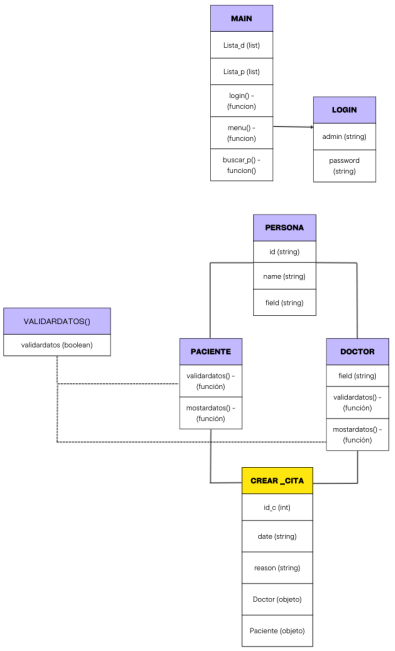


Justificación:

Se crearon las variables main, login, menu, Alta_D (alta doctor), Alta_P (alta paciente), Crear_C (crear cita) con el fin de separar responsabilidades. Esto nos permite tener un mejor registro de doctores, pacientes y citas de una forma independiente pero organizada.
En donde el login funciona como un sistema que garantiza que solo el personal autorizado (administrador) pueda acceder al menú principal, validando las credenciales antes de permitir cualquier operación, el diseño del menú se justifica con el hecho de crear una navegación intuitiva mediante una estructura de selección múltiple según la necesidad y por último la integración y relación de datos, en donde existe condicionales en donde se necesita revisar que la información si coincida con la registrada en el sistema y si no existe se dirige a crear un nuevo paciente.

Diseño del programa (diagrama de clases):

• Clase Principal.
• Clase para Doctor.
• Clase para Paciente.
• Clase para Cita.



Justificación:

Se creó la clase abstracta **Persona** que actúa como el pilar que contiene los atributos id, name y field (especialidad). Su propósito es centralizar la información básica, permitiendo que otras clases la hereden sin repetir código. Por otro lado la clase **Doctor** hereda de persona y por otro lado la clase **Paciente** se enfoca en el registro único de los usuarios validando su existencia, la clase crear cita funciona como un puente entre los objetos completos de **Doctor** y **Paciente** y por último la clase validar datos solamente es una operación booleana que asegura que los datos cumplan con los requisitos antes de ser procesados en el sistema.
La clase **Main** y **Login**, una gestiona las listas globales de los doctores y pacientes, mientras que otra se encarga de la autenticación para proteger el acceso al menú principal.

Pseudocódigo en PSeInt:

```
//id_p = identificador del usuario
//p_name =  nombre del paciente
//id_d = indentificador del doctor
//d_name = nombre del paciente
subproceso exito <- login(admin, password)
	definir user, pass como cadena
	exito <- Falso

	Mientras exito = Falso Hacer
		Escribir "Ingrese usuario:"
		Leer user
		Escribir "Ingrese contraseña:"
		Leer pass

		Si user = admin y pass = password Entonces
			Escribir "Bienvenido"
			exito <- Verdadero
		SiNo
			Escribir "Usuario y/o contraseña incorrectos"
		FinSi
	FinMientras
FinSubProceso


SubProceso Alta_D(id_d0, d_name0 por referencia)
	Repetir
		Escribir "Ingresa el ID del doctor"
		Leer id_d
		Escribir "El ID ya se encuentra registrado"
		Si id_d = id_d0 Entonces
		FinSi
	Hasta Que id_d <> id_d0
	Escribir "Ingresa el nombre del doctor: "
	Leer d_name
	Escribir "Se ha registrado con exito. "
FinSubProceso


SubProceso Alta_P(id_p0, p_name0 por referencia)
	Repetir
		Escribir "Ingresa el ID del paciente"
		Leer id_p
		Escribir "El ID ya se encuentra registrado"
		Si id_p = id_p0 Entonces
		FinSi
	Hasta Que id_d <> id_d0
	Escribir "Ingresa el nombre del paciente: "
	Leer d_name
	Escribir "Se ha registrado con exito. "
FinSubProceso


Subproceso Crear_C(id_p0, p_name0, id_d0, d_name0 Por Referencia)
	Definir id_c, fecha, motivo como Cadena
	Definir id_p2, id_d2 Como Entero
	Escribir "ID de la cita: "
	Leer id_c
	Escribir "Ingresa Fecha y hora: "
	Leer fecha
	Escribir "Motivo de la cita"
	Leer motivo
	Escribir "Ingresa el ID del paciente: "
	Leer id_p2
	Escribir "Ingresa el ID del doctor: "
```

```
        Si id_p2 = id_p y id_d2 = id_d Entonces
                Escribir "Cita registrada correctamente"
                Escribir "La cita con id", id_c, "a nombre de: " + p_name, "con fecha ", fecha, "con el doctor: ", d_name,
"por el motivo de " + motivo
        SiNo
                Escribir "Error: el doctor o el paciente no existe"
        FinSi
FinSubProceso


SubProceso Mostrar_datos(id_p0, p_name0, id_d0, d_name0, id_c, fecha, motivo)
        Escribir "----- Datos Registrados -----"
        Escribir "Paciente"
        Escribir " ID: ", id_p0
        Escribir " Nombre: ", p_name0

        Escribir "Doctor"
        Escribir " ID: ", id_d0
        Escribir " Nombre: ", d_name0

        Escribir "Citas"
        Escribir " ID: ", id_c
        Escribir " Fecha: ", fecha
        Escribir " Motivo: ", motivo
FinSubProceso


SubProceso  Menu(id_p0, p_name0, id_d0, d_name0)
        Definir opcion Como Entero

        Repetir
                Escribir "----- Menu Principal -----"
                Escribir "1.  Dar alta doctor"
                Escribir "2.  Dar alta paciente"
                Escribir "3.  Crear cita"
                Escribir "4.  Mostrar datos"
                Escribir "5.  Salir"
                Leer opcion
                Segun opcion Hacer
                        1:
                                Alta_D(id_d0, d_name0)
                        2:
                                Alta_P(id_p0, p_name0)
                        3:
                                Crear_C(id_p0, p_name0, id_d0, d_name0)
                        4:
                                Mostrar_Datos(id_p0, p_name0, id_d0, d_name0, id_c, fecha, motivo)
                        5:
                                Escribir "Saliendo....."
                        De otro modo:
                                Escribir "Opcion no valida"
                FinSegun
        Hasta Que Opcion = 5
FinSubProceso


Algoritmo Sistema
        Definir id_p0, id_d0 Como Entero
        Definir p_name0, d_name0 Como Caracter
        Definir admin, password Como Caracter
        Definir acceso Como Logico
        //Datos iniciales
        id_p0 <- 101
        p_name0 <- "Juan Perez"
        id_d0 <- 201
```

```
        d_name0 <- "Alejandro Ruiz"
        admin <- "admin1"
        password <- "0000"
        acceso <- login(admin, password)
        Si acceso = Verdadero Entonces
                Menu(id_p0, p_name0, id_d0, d_name0)
        FinSi
FinAlgoritmo
```

Justificación:

Este código implementa un sistema básico de administración médica utilizando pseudocódigo, diagramas de flujo y diagrama de clases en donde se controlan las principales operaciones del sistemas mediante subprocesos simulando ser objetos, esto incluye un mecanismo de autenticación para restringir el acceso al sistema, así como funciones para registrar doctores, pacientes, crear citas asegurando la relación entre ambos mediante la validación de identificados, además de un menú principal donde se facilita la navegación al usuario.

Codigo en Java:
Archivo se encarga de escribir los datos en un archivo txt con el fin de simular una base de datos.

```
import java.io.*;
import java.util.*;

//Este codigo se encarga de agregar y leer los datos disponibles en los archivos de
textos
public class Archivo{
    //Esta funcion escribe en los archivos de txt
    public static void writeLines(String archivo, String linea){
        try(BufferedWriter bw = new BufferedWriter(new FileWriter(archivo, true))){
            bw.write(linea);
            bw.newLine();
        } catch (IOException e){
            System.err.println("Error al escribir en "+ archivo);
        }
    }
    //Esta funcion lee los datos de los archivos de texto
    public static List<String> readLines(String archivo){
        List<String> lineas = new ArrayList<>();
        try(BufferedReader br = new BufferedReader(new FileReader(archivo))){
            String linea;
            while((linea = br.readLine()) != null){
                lineas.add(linea);
            }
        } catch (IOException e){
            System.out.println("Error leyendo " + archivo);
        }
        return lineas;
    }

}
```

Estas clases son una especie de ficha digital del doctor, paciente y citas dentro del sistema, con el propósito de poder leer la información de cada entidad y permitir que otras clases puedan consultarla de forma segura. Los datos se mantienen protegidos mediante el encapsulamiento y solamente pueden ser accedidos a través de los métodos getter.

```
public class Doctor{

    private int id;
    private String nombre;

    public Doctor(int id, String nombre){
        this.id = id;
        this.nombre = nombre;
    }
    public int getId(){
        return id;
    }
    public String getNombre(){
        return nombre;
    }
}

public class Paciente{
    private int id;
    private String nombre;

    public Paciente(int id, String nombre){
        this.id = id;
        this.nombre = nombre;
    }
    public int getId(){
        return id;
    }
    public String getNombre(){
        return nombre;
    }
}
public class Citas {

public String idCita;
private String fecha;
private String motivo;
private Paciente paciente;
private Doctor doctor;

public void Cita(String idCita, String fecha, String motivo, Paciente paciente,
Doctor doctor){
    this.idCita = idCita;
    this.fecha = fecha;
    this.motivo = motivo;
    this.paciente = paciente;
    this.doctor = doctor;
}
public String getIdCita(){
    return idCita;
}
public String getFecha(){
    return fecha;
```

```
}
public String getMotivo(){
    return motivo;
}
public Paciente getPaciente(){
    return paciente;
}
public Doctor getDoctor(){
    return doctor;
}

}
```

La clase Login se encarga de controlar el acceso al sistema por parte administrados, donde los datos se validan desde el archivo admin.txt, su intencion es funcionar como un filtro de seguridad, donde permite entrar al sistema, de lo contrario se muestra un mensaje de error.

```
import java.util.Scanner;
//Se encarga de validar que user y contraseña sean correctas, funciona como un
filtro de seguridad
public class Login {
    private Scanner sc = new Scanner(System.in);

    public boolean newLogin(){
        while(true){
            System.out.print("Ingrese usuario: ");
            String user = sc.nextLine();
            System.out.print("Ingrese contraseña: ");
            String pass = sc.nextLine();

            for(String linea : Archivo.readLines("admin.txt")){
                String[] datos = linea.split(",");
                if(datos[0].equals(user) && datos[1].equals(pass)){
                    System.out.println("Bienvenido al Sistema\n");
                    return true;
                }
                System.out.println("Usuario o contraseña incorrectas\n");
            }

        }

    }
}
```

Ejecución:

```
Ingrese usuario: admin
Ingrese contraseña: 1222
Usuario u contraseña incorrectas
```

```
Ingrese usuario: admin1
Ingrese contraseña: 0000
Binvenido al Sistema
```

La clase Sistema es el esqueleto del programa, ya que en ella se encuentran las principales funciones del programa, como lo es registrar doctores, pacientes, citas, validar información y mostrar en pantalla toda la información almacenada en los archivos de textos que simulan ser

una base de datos. También incluye un método para verificar si un paciente o doctor ya está registrado, asegurando que solo se creen con los datos válidos.

```
import java.util.Scanner;

//Este codigo se encarga de crear las citas, dar de alta doctores y pacientes
//ademas mostrar los datos que se encuentran en los archivos de texto

public class Sistema {
    private Scanner sc = new Scanner(System.in);

    public void altaDoctor(){
        System.out.print("ID Doctor: " );
        int id = Integer.parseInt(sc.nextLine());
        System.out.print("Nombre del Doctor: ");
        String nombre = sc.nextLine();

        Archivo.writeLines("doctores.txt", id + "," + nombre);
        System.out.println("Doctor guardado.\n");
    }
        public void altaPaciente(){
        System.out.print("ID Paciente: " );
        int id = Integer.parseInt(sc.nextLine());
        System.out.print("Nombre del Paciente: ");
        String nombre = sc.nextLine();

        Archivo.writeLines("pacientes.txt", id + "," + nombre);
        System.out.println("Paciente guardado.\n");
    }
    private boolean existe(String archivo, int id){
        for(String linea : Archivo.readLines(archivo)){
            if (linea.trim().isEmpty()) continue;
            String[] datos = linea.split(",");
            if(Integer.parseInt(datos[0]) == id) return true;
        }
        return false;
    }

    public void crearCita(){
        System.out.print("ID Cita: ");
        String idCita = sc.nextLine();
        System.out.print("Fecha y hora: ");
        String fecha = sc.nextLine();
        System.out.print("Motivo: ");
        String motivo = sc.nextLine();
        System.out.print("ID Paciente: ");
        int idPaciente = Integer.parseInt(sc.nextLine());
        System.out.print("ID Doctor: ");
        int idDoctor = Integer.parseInt(sc.nextLine());

        if(existe("pacientes.txt", idPaciente) && existe("doctores.txt",
idDoctor)){
            Archivo.writeLines("citas.txt", idCita + "," + fecha + "," + motivo +
"," + idPaciente + "," + idDoctor);
            System.out.println("Cita guardada en el sistema\n");
        } else{
            System.out.println("Paciente o doctor no existen. \n");
        }
```

```
    }

    public void mostrarDatos(){
        System.out.println("\n");
        System.out.println(" ----- DOCTORES -----");
        for (String linea : Archivo.readLines("doctores.txt")){
            String[] d = linea.split(",");
            System.out.println("ID: "+ d[0] + " | Nombre: "+ d[1]);
            System.out.println("--------------------------");

        }

        System.out.println(" ----- PACIENTES -----");
        for (String linea : Archivo.readLines("pacientes.txt")){
            String[] p = linea.split(",");
            System.out.println("ID: "+ p[0] + " | Nombre: "+ p[1]);
            System.out.println("--------------------------");

        }
        System.out.println(" ----- CITAS -----");
        for(String linea : Archivo.readLines("citas.txt")){
            String[] c = linea.split(",");
            System.out.println("Cita ID: "+ c[0]);
            System.out.println("Fecha: "+ c[1]);
            System.out.println("Motivo: "+ c[2]);
            System.out.println("Paciente ID: "+ c[3]);
            System.out.println("Doctor ID: "+ c[4]);
            System.out.println("--------------------------");

        }
        System.out.println();
    }
}
```

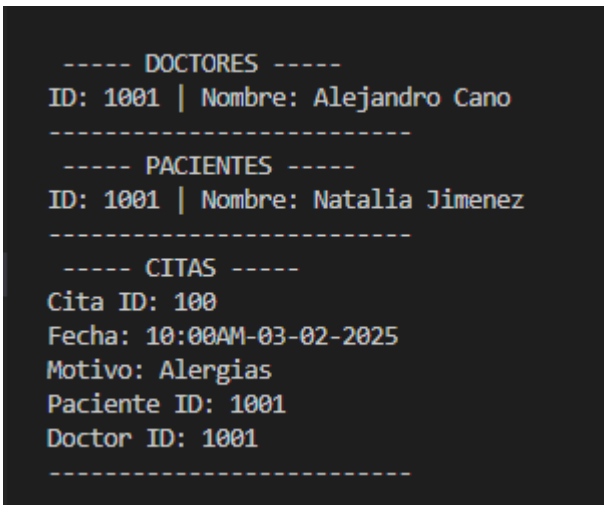
Ejecución:

```
ID Doctor: 1001
Nombre del Doctor: Alejandro Cano
Doctor guardado.
```

```
ID Paciente: 1001
Nombre del Paciente: Natalia Jimenez
Paciente guardado.
```

```
ID Cita: 100
Fecha y hora: 10:00AM-03-02-2025
Motivo: Alergias
ID Paciente: 1001
ID Doctor: 1001
Cita guardada en el sistema
```

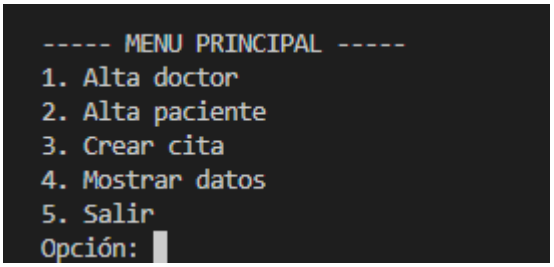
```
ID Cita: 102
Fecha y hora: 11:00AM-03-02-2025
Motivo: consulta general
ID Paciente: 1002
ID Doctor: 1001
Paciente o doctor no existen.
```



La clase menú se encarga de mostrar el menú principal del sistema y gestionar la interacción, presenta las opciones disponibles, según la opción elegida, llama los métodos correspondientes de la clase sistema.

```
import java.util.Scanner;

public class Menu {

    private Scanner sc = new Scanner(System.in);
    private Sistema sistema;

    public Menu(Sistema sistema){
        this.sistema = sistema;
    }

    public void mostrar(){
        int opcion;
        do{
            System.out.println("----- MENU PRINCIPAL -----");
            System.out.println("1. Alta doctor");
            System.out.println("2. Alta paciente");
            System.out.println("3. Crear cita");
            System.out.println("4. Mostrar datos");
            System.out.println("5. Salir");
            System.out.print("Opción: ");
            opcion = Integer.parseInt(sc.nextLine());

            switch(opcion){
                case 1 -> sistema.altaDoctor();
                case 2 -> sistema.altaPaciente();
                case 3 -> sistema.crearCita();
                case 4 -> sistema.mostrarDatos();
                case 5 -> System.out.println("Saliendo");
                default -> System.out.println("Opcion invalida\n");

            }
        } while (opcion != 5);
    }
}
```

Ejecución:



La clase Main es el punto de inicio del programa, su función es arrancar el programa.

```
//Aqui inicializamos el programa para que funcione como se espera
public class Main {
    public static void main(String[] args) {

        Login login = new Login();

        if (login.newLogin()) {
            Sistema sistema = new Sistema();
            Menu menu = new Menu(sistema);
            menu.mostrar();
        }

    }
}
```

Justificación:

Este proyecto se desarrolló con el objetivo de aplicar los principios básicos de la creación de un sistema básico de gestión de citas médicas. Se busca simular el funcionamiento de un consultorio, permitiendo registrar doctores, pacientes y agendar citas y consultar la información almacenada. Además se hace uso de archivos de texto como almacenamiento (simular base de datos) con el fin de entender el manejo de archivos, validación de datos y organización. En conjunto el proyecto integra estructuras, métodos, clases y lógica de validación, fortaleciendo las habilidades fundamentales de Java.