

EVIDENCIA – COMPUTACIÓN EN JAVA

Descripción y requerimientos del programa:

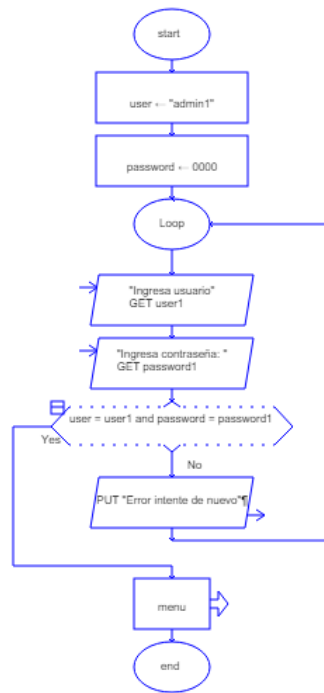
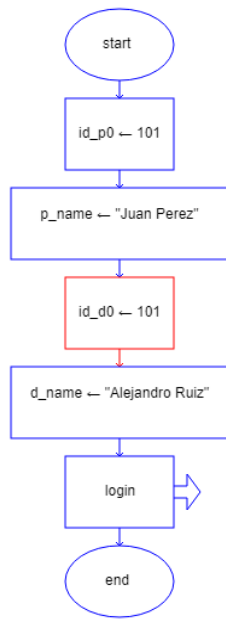
El producto final será un programa que simulará un sistema de administración de citas con las siguientes funcionalidades:

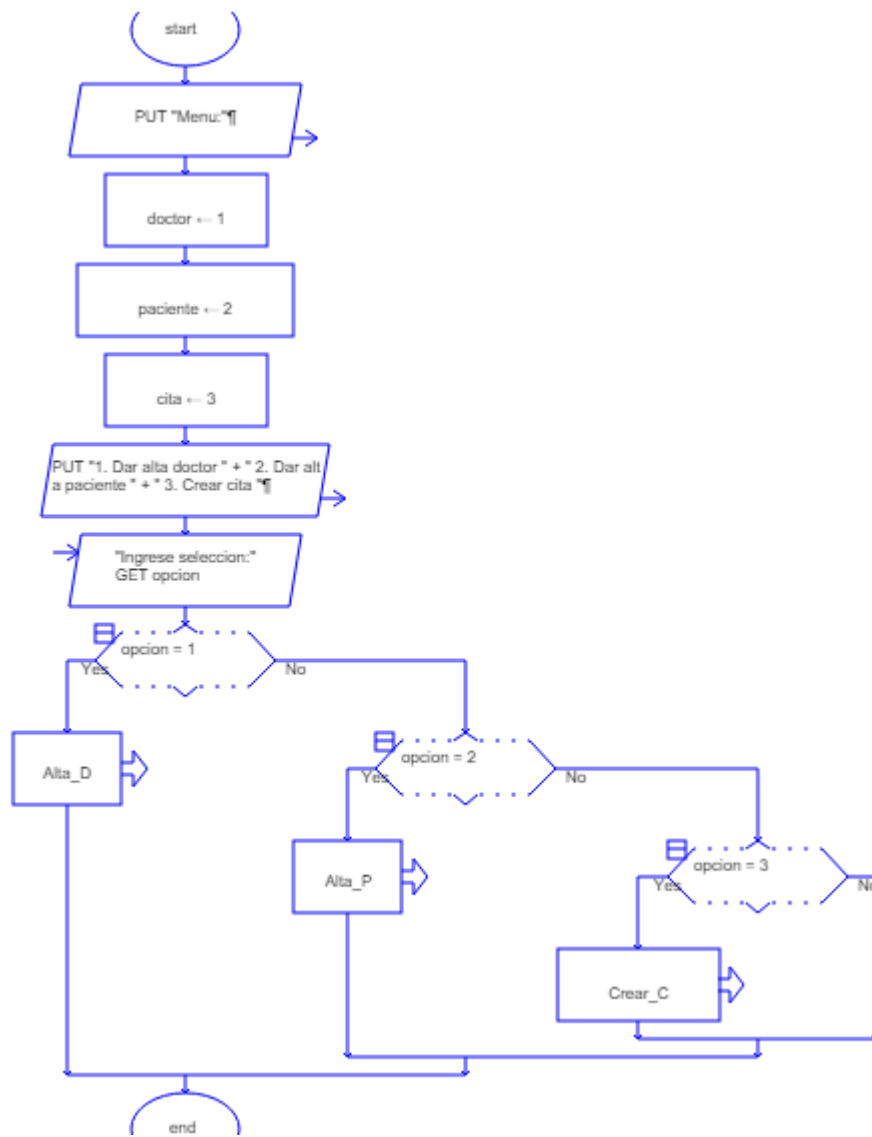
- **Dar de alta doctores: se deberán poder dar de alta los doctores del consultorio médico, los datos básicos serán:**
 - **Identificador único.**
 - **Nombre completo.**
 - **Especialidad.**
- **Dar de alta pacientes: se deberán poder registrar los pacientes que acudan al consultorio médico, los datos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

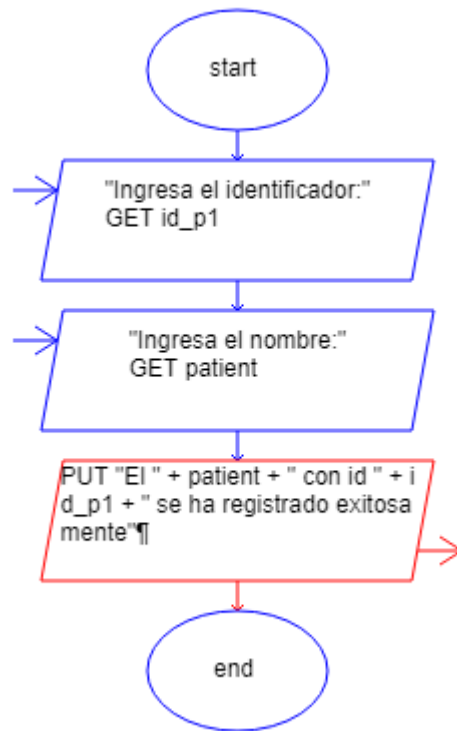
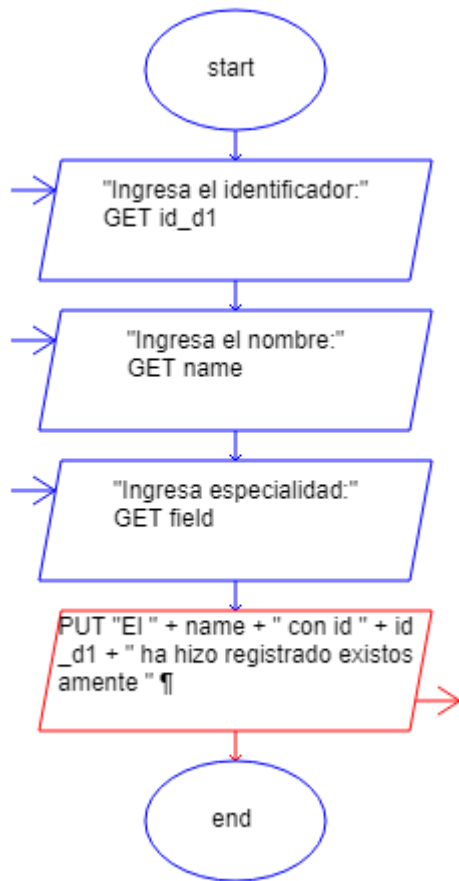
Link de Github: <https://github.com/Veloz2/Evidencia---Computaci-n-en-JAVA.git>

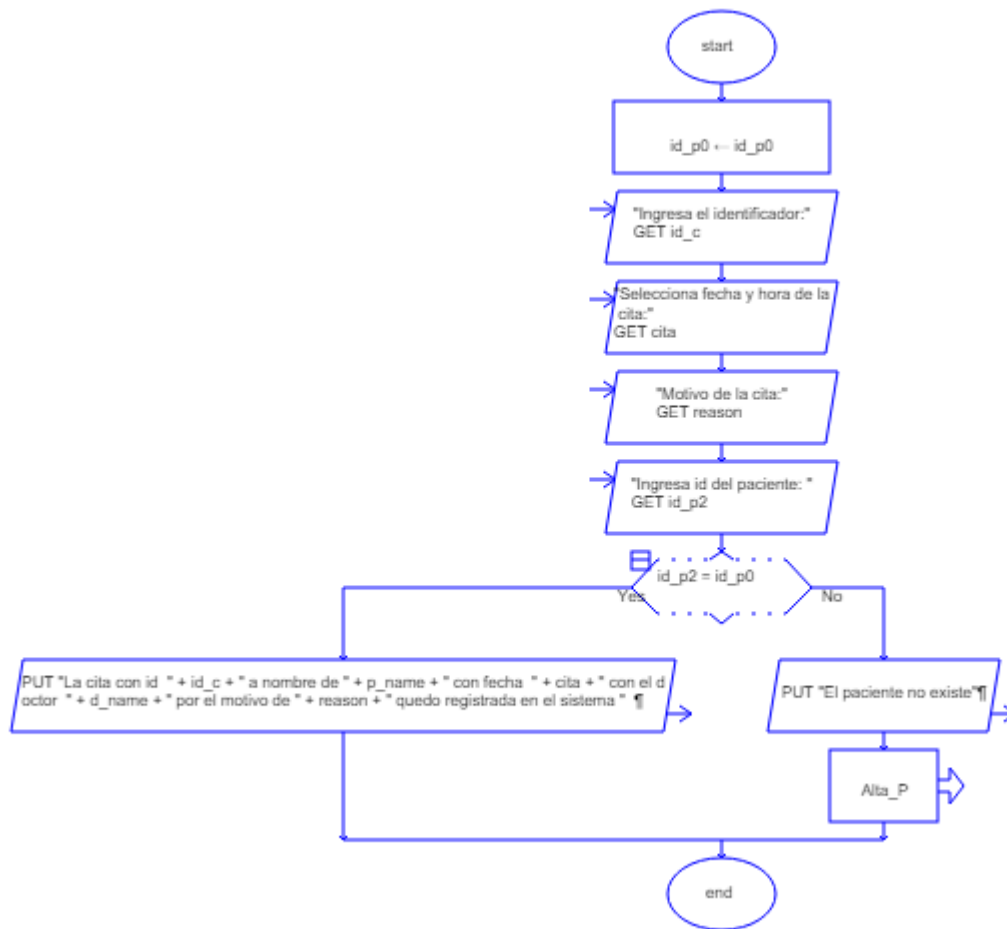
En este link se encuentran todos los documentos que sirvieron para la creación de este documento.

Diagrama del flujo (Raptor):









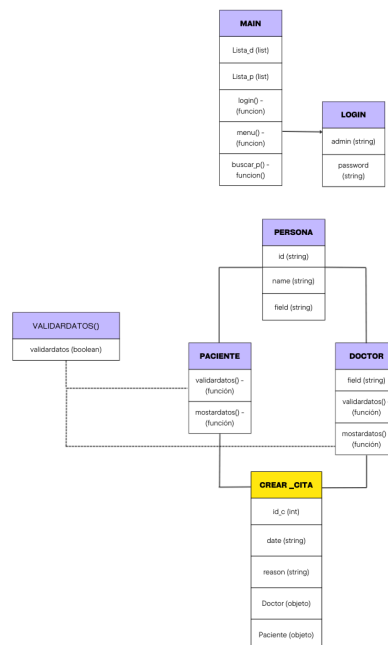
Justificación:

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 = identificador 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.