Código básico sin parte gráfica para ejecutar en java online

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

// Listas para almacenar especialistas, especialidades y citas

static ArrayList<String> specialists = new ArrayList<>();

static ArrayList<String> specialties = new ArrayList<>();

static ArrayList<String> appointments = new ArrayList<>();

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int option;

do {

// Mostrar el menú

System.out.println("\nMenú:");

System.out.println("1. Registrar Especialista");

System.out.println("2. Gestión de Especialidades");

System.out.println("3. Gestión de Citas");

System.out.println("0. Salir");

System.out.print("Seleccione una opción: ");

option = scanner.nextInt();

scanner.nextLine(); // Consumir el salto de línea

// Manejar la opción seleccionada

switch (option) {

case 1:

registerSpecialist(scanner);

break;

case 2:

manageSpecialties(scanner);

break;

case 3:

manageAppointments(scanner);

break;

case 0:

System.out.println("Saliendo del sistema...");

break;

default:

System.out.println("Opción no válida. Intente de nuevo.");

break;

}

} while (option != 0);

scanner.close(); // Cerrar el escáner

}

// Método para registrar un especialista

private static void registerSpecialist(Scanner scanner) {

System.out.print("Ingrese el nombre del especialista: ");

String specialistName = scanner.nextLine();

specialists.add(specialistName);

System.out.println("Especialista registrado: " + specialistName);

}

// Método para gestionar especialidades

private static void manageSpecialties(Scanner scanner) {

System.out.print("Ingrese la especialidad: ");

String specialty = scanner.nextLine();

specialties.add(specialty);

System.out.println("Especialidad agregada: " + specialty);

}

// Método para gestionar citas

private static void manageAppointments(Scanner scanner) {

System.out.print("Ingrese el nombre del paciente: ");

String patientName = scanner.nextLine();

System.out.print("Ingrese la especialidad requerida: ");

String specialty = scanner.nextLine();

System.out.print("Ingrese la fecha y hora (dd/MM/yyyy HH:mm): ");

String dateTime = scanner.nextLine();

String appointment = "Paciente: " + patientName + ", Especialidad: " + specialty + ", Fecha y Hora: " + dateTime;

appointments.add(appointment);

System.out.println("Cita registrada: " + appointment);

}

}

Código con la parte gráfica para JavaFX instalado en el computador(No corre en compilador online por limitaciones)

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.util.ArrayList;

public class MedicalAppointmentSystem extends Application {

// Listas para almacenar especialistas, especialidades y citas

static ArrayList<String> specialists = new ArrayList<>();

static ArrayList<String> specialties = new ArrayList<>();

static ArrayList<String> appointments = new ArrayList<>();

public static void main(String[] args) {

launch(args); // Lanzar la aplicación JavaFX

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Sistema de Citas Médicas");

// Crear los botones de la interfaz principal

Button registerSpecialistButton = new Button("Registrar Especialista");

Button manageSpecialtiesButton = new Button("Gestión de Especialidades");

Button manageAppointmentsButton = new Button("Gestión de Citas");

// Configurar la acción de los botones

registerSpecialistButton.setOnAction(e -> showRegisterSpecialistWindow());

manageSpecialtiesButton.setOnAction(e -> showManageSpecialtiesWindow());

manageAppointmentsButton.setOnAction(e -> showManageAppointmentsWindow());

// Layout de la ventana principal

VBox layout = new VBox(10);

layout.setPadding(new Insets(10));

layout.getChildren().addAll(registerSpecialistButton, manageSpecialtiesButton, manageAppointmentsButton);

Scene scene = new Scene(layout, 300, 200);

primaryStage.setScene(scene);

primaryStage.show();

}

// Método para mostrar la ventana de registro de especialista

private void showRegisterSpecialistWindow() {

Stage registerStage = new Stage();

registerStage.setTitle("Registrar Especialista");

// Layout para el registro de especialista

GridPane grid = new GridPane();

grid.setPadding(new Insets(10, 10, 10, 10));

grid.setVgap(8);

grid.setHgap(10);

// Elementos de entrada

Label nameLabel = new Label("Nombre del Especialista:");

TextField nameInput = new TextField();

Button submitButton = new Button("Registrar");

// Agregar elementos al grid

grid.add(nameLabel, 0, 0);

grid.add(nameInput, 1, 0);

grid.add(submitButton, 1, 1);

// Acción del botón de registro

submitButton.setOnAction(e -> {

String specialistName = nameInput.getText();

specialists.add(specialistName);

System.out.println("Especialista registrado: " + specialistName);

nameInput.clear(); // Limpiar el campo

});

Scene scene = new Scene(grid, 300, 150);

registerStage.setScene(scene);

registerStage.show();

}

// Método para mostrar la ventana de gestión de especialidades

private void showManageSpecialtiesWindow() {

Stage specialtiesStage = new Stage();

specialtiesStage.setTitle("Gestión de Especialidades");

// Layout para gestión de especialidades

GridPane grid = new GridPane();

grid.setPadding(new Insets(10, 10, 10, 10));

grid.setVgap(8);

grid.setHgap(10);

// Elementos de entrada

Label specialtyLabel = new Label("Especialidad:");

TextField specialtyInput = new TextField();

Button addSpecialtyButton = new Button("Agregar Especialidad");

// Agregar elementos al grid

grid.add(specialtyLabel, 0, 0);

grid.add(specialtyInput, 1, 0);

grid.add(addSpecialtyButton, 1, 1);

// Acción del botón para agregar especialidad

addSpecialtyButton.setOnAction(e -> {

String specialty = specialtyInput.getText();

specialties.add(specialty);

System.out.println("Especialidad agregada: " + specialty);

specialtyInput.clear(); // Limpiar el campo

});

Scene scene = new Scene(grid, 300, 150);

specialtiesStage.setScene(scene);

specialtiesStage.show();

}

// Método para mostrar la ventana de gestión de citas

private void showManageAppointmentsWindow() {

Stage appointmentStage = new Stage();

appointmentStage.setTitle("Gestión de Citas");

// Layout para gestión de citas

GridPane grid = new GridPane();

grid.setPadding(new Insets(10, 10, 10, 10));

grid.setVgap(8);

grid.setHgap(10);

// Elementos de entrada

Label patientLabel = new Label("Nombre del Paciente:");

TextField patientInput = new TextField();

Label specialtyLabel = new Label("Especialidad:");

TextField specialtyInput = new TextField();

Label dateLabel = new Label("Fecha y Hora:");

TextField dateInput = new TextField();

Button scheduleButton = new Button("Agendar Cita");

// Agregar elementos al grid

grid.add(patientLabel, 0, 0);

grid.add(patientInput, 1, 0);

grid.add(specialtyLabel, 0, 1);

grid.add(specialtyInput, 1, 1);

grid.add(dateLabel, 0, 2);

grid.add(dateInput, 1, 2);

grid.add(scheduleButton, 1, 3);

// Acción del botón para agendar cita

scheduleButton.setOnAction(e -> {

String patientName = patientInput.getText();

String specialty = specialtyInput.getText();

String dateTime = dateInput.getText();

String appointment = "Paciente: " + patientName + ", Especialidad: " + specialty + ", Fecha y Hora: " + dateTime;

appointments.add(appointment);

System.out.println("Cita registrada: " + appointment);

patientInput.clear(); // Limpiar campos

specialtyInput.clear();

dateInput.clear();

});

Scene scene = new Scene(grid, 400, 250);

appointmentStage.setScene(scene);

appointmentStage.show();

}

}