UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE Elige un elemento.

CICLO ACADÉMICO: MARZO – JULIO 2025

INFORME DE GUÍA PRÁCTICA

I. PORTADA

Tema:	Gestión de colección de datos utilizando vectores
Unidad de Organización Curricular:	BÁSICA
Nivel y Paralelo:	Tercero- "A"
Alumnos participantes:	Auz Mayorga José Mateo Guevara López Oscar Mauricio Mora Beltrán Santiago Sebastián Peña Aguilar Javier Aldair
Asignatura:	Estructura de datos
Docente:	Ing. Félix Fernández, Mg.

II. INFORME DE GUÍA PRÁCTICA

2.1 Objetivos

General:

Determinar el cálculo de la media de valores en colecciones de datos estructurados utilizando programación orientada a objetos y vectores.

Específicos:

- Facilitar la gestión eficiente de los estudiantes matriculados en un curso, permitiendo registrar, modificar y eliminar sus datos personales a través de una aplicación interactiva.
- Brindar herramientas para el seguimiento académico individual, mediante el ingreso, edición y control de calificaciones, así como el cálculo del promedio de cada estudiante.
- Ofrecer una visión general del rendimiento del curso, permitiendo al usuario conocer el promedio general de calificaciones y tomar decisiones informadas basadas en los datos registrados.

2.2 Modalidad

Presencial

2.3 Tiempo de duración

Presenciales: 4

No presenciales: 0

2.4 Instrucciones

1. Lleve a cabo las actividades indicadas.
2. Suba a plataforma el código resultante.

2.5 Listado de equipos, materiales y recursos

Listado de equipos y materiales generales empleados en la guía práctica:

- PC de escritorio o portátil con JDK instalado e IDE de su elección.

TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

☐Plataformas educativas

☐Simuladores y laboratorios virtuales

☐Aplicaciones educativas
☐Recursos audiovisuales
☐Gamificación
☒Inteligencia Artificial
Otros (Especifique): _____

2.6 Actividades por desarrollar

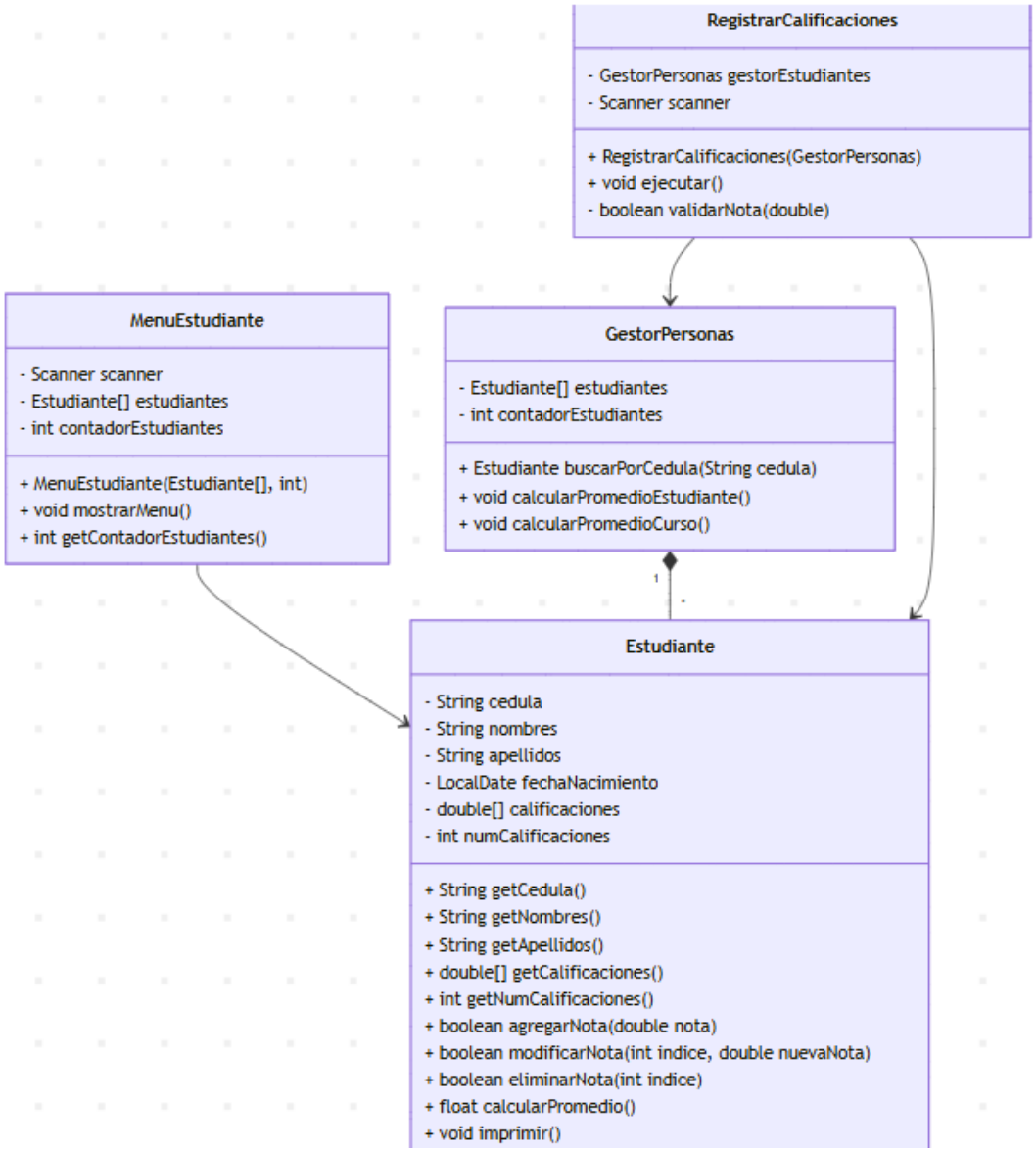
Implemente una aplicación para la gestión de los datos de los estudiantes matriculados en un curso con cupo para hasta 20 estudiantes. Para ello, deberá:
1. Definir la clase Estudiante, con atributos cédula de identidad, nombres, apellidos, fecha de nacimiento y un vector con las notas (máximo de 7 notas, asumiendo que es la cantidad máxima de actividades en que se evaluará a un estudiante).
2. Implementar adecuadamente el constructor de la clase Estudiante y un método buscar(), en la clase que considere oportuno, para que busque por cédula a un estudiante dado. 3. Implemente el siguiente menú de opciones en la aplicación Java por consola:

=== GESTOR DE PERSONAS ===
1 Estudiantes.
2 Registro de calificaciones.
3 Determinar el promedio de notas de un estudiante.
4 Determinar el promedio de notas del curso. Teclee su opción (1-4)

El comportamiento del programa cuando el usuario interactúe con el menú de opciones deberá ser como se describe a continuación. En la opción 1, al usuario se le mostrará un listado con un autonumérico, para identificar cada registro de cada estudiante, y los datos de cada estudiante previamente registrado. A partir de ello, el usuario podrá acceder a un submenú que le permita ingresar, modificar y eliminar los datos de un estudiante. Cada vez que un usuario inserte/modifique/elimine un registro, si es que tiene sentido, se le preguntará si desea realizar la misma acción una vez más. Si la cantidad máxima de estudiantes ya fue registrada, no se permitirá insertar datos de un nuevo estudiante. Si no hay estudiantes registrados, no se le permitirá intentar eliminar el registro de un estudiante. En todo caso, para las opciones de modificar y eliminar, el usuario comenzará indicando el valor de autonumérico que identifique al registro de estudiante que se quiere modificar/eliminar.
En la opción 2, el usuario podrá ingresar los datos de calificaciones de un estudiante. Comenzará introduciendo el número de cédula del estudiante en cuestión. Si el número de cédula pertenece a uno de los estudiantes registrados, se mostrarán sus datos (nombres y apellidos y edad), y se permitirá ingresar tantos valores de calificaciones como el usuario considere. Se le mostrará un listado de calificaciones previamente registradas. Se permitirá modificar o eliminar cada una de estas, o insertar nuevas calificaciones. Si se llega a ingresar la cantidad máxima de calificaciones posible, el sistema notificará que se han ingresado todas las calificaciones posibles y se da por terminado el proceso de entrada de calificaciones. Si el número de cédula ingresado inicialmente no fuera el de alguno de los estudiantes registrados, se le notificará oportunamente al usuario y se le dará la opción de ingresar otro número de cédula o regresar al menú principal.
En la opción 3, el usuario comenzará introduciendo el número de cédula del estudiante del que se quiere calcular el promedio de calificaciones. Si el número de cédula pertenece a uno de los estudiantes ya registrados, se mostrarán sus datos (nombres y apellidos, edad y promedio de calificaciones). Caso contrario, se emitirá un mensaje de error indicando que no se encontró un estudiante con el número de cédula indicado.
En la opción 4, se mostrará un mensaje indicando el valor del promedio general de calificaciones. Si ningún estudiante tiene calificaciones registradas, de igual manera, se indicará que No se han registrado calificaciones de estudiantes.

Nota: el conjunto de por estudiante, se almacenará en vectores

2.7 Resultados obtenidos



Este diagrama representa una arquitectura funcional de la aplicación de Arrays en una implementación de Gestión de Estudiantes

Estudiante: contiene datos y lógica básica.
GestorPersonas: gestiona el conjunto.
MenuEstudiante: interfaz del usuario.
RegistrarCalificaciones: manejo de notas.

Clase Estudiante

- Estructura inicial de la clase en esta parte se manejará el ingreso de un máximo de 7 notas. Esta lógica de almacenamiento se implementa usando un vector (arreglo), lo que permite organizar las notas de cada estudiante de forma estructurada y acceder a ellas fácilmente cuando se necesiten. Tal como mencionan Malhotra y Malhotra (2019), "*una estructura de datos nos ayuda a guardar y manejar la información dentro de la memoria del computador de forma eficiente"*. [1] Esto es fundamental en programas como este, donde

necesitamos mantener los datos ordenados y disponibles para realizar operaciones como calcular promedios o editar calificaciones.

```
public class Estudiante {
    String cedula, nombres,apellidos;
    LocalDate fechaNacimiento;
    double []calificaciones;
    int numCalificaciones;

    public Estudiante(String cedula, String nombres, String apellidos,
            LocalDate fechaNacimiento) {
        this.cedula = cedula;
        this.nombres = nombres;
        this.apellidos = apellidos;
        this.fechaNacimiento = fechaNacimiento;
        this.calificaciones = new double[7]; // máximo 7 notas
    }
```

- **Método imprimir:** Este método es de gran importancia ya que muestra la información general de cada estudiante y todas sus calificaciones validas registradas de manera estructurada.

```
public void imprimir() {
    System.out.print("Cedula: " + cedula + " | Nombres: " + nombres +
            " | Apellidos: " + apellidos +" | Fecha de nacimiento: " +
            fechaNacimiento + " | Notas: ");
    for (int i = 0; i < numCalificaciones; i++) {
    System.out.print(calificaciones[i] + " ");
}
    System.out.println();
}
```

- **Métodos agregar, modificar y eliminar nota:** Estos métodos sirven para la gestión de las calificaciones de cada estudiante dentro de las clases posteriores.

```java
public boolean agregarNota(double nota) {
    if (numCalificaciones < calificaciones.length) {
        calificaciones[numCalificaciones++] = nota;
        return true;
    }
    return false;
}

public boolean modificarNota(int indice, double nuevaNota) {
    if (indice >= 0 && indice < numCalificaciones) {
        calificaciones[indice] = nuevaNota;
        return true;
    }
    return false;
}


public boolean eliminarNota(int indice) {
    if (indice >= 0 && indice < numCalificaciones) {
        calificaciones[indice] = 0.0;
        for (int i = indice; i < numCalificaciones - 1; i++) {
            calificaciones[i] = calificaciones[i + 1];
        }
        numCalificaciones--;
        return true;
    }
    return false;
}
```

- **Método calcularPropemdio:** calcula el promedio de las notas validas de cada estudiante, ignorando las notas de 0 o posiciones vacias.

```java
public float calcularPromedio() {
float suma = 0;
if (numCalificaciones > 0) {
    for (int i = 0; i < numCalificaciones; i++) {
        suma += calificaciones[i];
    }
    return suma / numCalificaciones;
} else {
    return 0;
}
```

Clase Menú estudiante

Estructura inicial del programa, define los atributos principales de la clase MenuEstudiante: un Scanner estático para leer datos desde consola (compartido entre todas las instancias), un arreglo de Estudiante para almacenar los registros, y un contador para llevar el control de cuántos estudiantes hay actualmente. En el constructor, se inicializan estos atributos con los valores recibidos y se llama al método mostrarMenu() para comenzar la interacción con el usuario desde la consola. Esta estructura permite gestionar dinámicamente estudiantes en un menú interactivo.

```
public class MenuEstudiante {
    private static Scanner scanner = new Scanner(System.in);
    private Estudiante[] estudiantes;
    private int contadorEstudiantes;

    public MenuEstudiante(Estudiante[] estudiantes, int contadorEstudiantes) {
        this.estudiantes = estudiantes;
        this.contadorEstudiantes = contadorEstudiantes;
        mostrarMenu();
    }
```

getContadorEstudiantes(), es un método público que permite acceder externamente al número actual de estudiantes registrados. El segundo, mostrarEstudiantes(), es un método privado que recorre el arreglo de estudiantes hasta el número registrado (contadorEstudiantes) e imprime los datos de cada uno usando su método imprimir().

Estos métodos apoyan la visualización y gestión del listado de estudiantes de forma ordenada.

```
public int getContadorEstudiantes() {
    return contadorEstudiantes;
}

private void mostrarEstudiantes() {
    for (int i = 0; i < contadorEstudiantes; i++) {
        System.out.println((i + 1) + ".-");
        estudiantes[i].imprimir();
    }
}
```

Verificación del número de estudiantes registrados (contadorEstudiantes) ha alcanzado el límite máximo de estudiantes permitido por el tamaño del arreglo (estudiantes.length). Si es así, imprime un mensaje informando que no se pueden agregar más estudiantes y luego interrumpe el flujo de ejecución con un break, lo que sale del ciclo o método en el que se encuentra. Esto evita que se agreguen más estudiantes si el arreglo ya está lleno.

```
if (contadorEstudiantes >= estudiantes.length) {
    System.out.println("No se puede ingresar más estudiantes. El curso ya está lleno.");
    break;
}
```

Inicio de la opción de modificación y eliminación de estudiantes

```
if (contadorEstudiantes == 0) {
    System.out.println("No hay estudiantes registrados para modificar.");
    break;
}
System.out.println("Ingrese el identificador del estudiante a modificar:");
int id = scanner.nextInt();
scanner.nextLine();
```

```
if (contadorEstudiantes == 0) {
    System.out.println("No hay estudiantes registrados para eliminar.");
    break;
}

System.out.println("Ingrese el número del estudiante a eliminar:");
int id = scanner.nextInt();
scanner.nextLine();
```

Clase RegistrarCalificaciones

Siguiendo las recomendaciones de Guardati [2] sobre la eficiencia de esta estructura para datos con límites definidos. La manipulación de estos vectores, que incluye operaciones como la inserción, modificación y eliminación de registros de calificaciones, se realizó a través de métodos encapsulados en la clase estudiante, lo cual facilitó la modularidad y la reutilización del código. Esta organización permitió un acceso directo a los elementos del vector, optimizando el rendimiento de la aplicación al realizar cálculos de promedios y otras operaciones estadísticas sobre los datos de los estudiantes.

Método para validar que se ingresen notas del 0 al 10.

```
private boolean validarNota(double nota) {
    return nota >= 0 && nota <= 10;
}
```

Caso 1 para el registro de nuevas calificaciones haciendo uso del método agregarNota incorporado en estudiante y su validación de ingreso de nota valido.

```
if (validarNota(nuevaCalificacion)) {
    if (estudiante.agregarNota(nuevaCalificacion)) {
        System.out.println("Calificacion agregada.");
        numCalificaciones = estudiante.getNumCalificaciones();
    }
```

Caso 2 para la modificación de calificaciones existentes haciendo uso del método modificarNota incorporado en estudiante y su validación de ingreso de nota valido.

```
if (validarNota(nuevaCalificacionModificar)) {
    if (estudiante.modificarNota(indiceModificar, nuevaCalificacionModificar)) {
        System.out.println("Calificacion modificada.");
```

Caso para la eliminación de calificaciones existentes haciendo uso del método eliminarNota incorporado en estudiante y su validación de nota valido.

```
if (estudiante.eliminarNota(indiceEliminar)) {
    System.out.println("Calificacion eliminada.");
    numCalificaciones = estudiante.getNumCalificaciones();
} else {
    System.out.println("Numero de calificacion invalido.");
}
```

Clase Principal Main

- **Crea la clase principal GestorPersonas:** Define las variables globales para el programa: entrada por teclado, arreglo de estudiantes, x|contador, el gestor principal y el manejador de calificaciones.

```
public class GestorPersonas {

    static Scanner scanner = new Scanner(System.in);
    static Estudiante[] estudiantes = new Estudiante[20];
    static int contadorEstudiantes = 0;
    static GestorPersonas gestor = new GestorPersonas();
    private RegistrarCalificaciones registroCalificaciones;
```

- Constructor de la clase que inicializa el módulo de calificaciones.

```
public GestorPersonas() {
    this.registroCalificaciones = new RegistrarCalificaciones(this);
}
```

- Método para calcular y mostrar el promedio de notas de un estudiante específico usando su cédula.

```
public static void calcularPromedioEstudiante() {
    System.out.print("Ingrese la cédula del estudiante: ");
    String cedula = scanner.nextLine();

    Estudiante estudiante = buscarPorCedula(cedula);
    if (estudiante != null) {
        float promedio = estudiante.calcularPromedio();
        System.out.println("Estudiante: " + estudiante.getNombres() + " " + estudiante.getApellidos());
        System.out.println("Edad: " + calcularEdad(estudiante.getFechaNacimiento()));
        System.out.printf("Promedio de calificaciones: %.2f\n", promedio);
    } else {
        System.out.println("No se encontró un estudiante con esa cédula.");
    }
}
```

- Calcula el promedio general de notas de todos los estudiantes registrados con calificaciones.

```
public static void calcularPromedioCurso() {
    float suma = 0;
    if (contadorEstudiantes == 0) {
        System.out.println("No hay estudiantes registrados en el curso.");
        return;
    }
    for (int i = 0; i < contadorEstudiantes; i++) {
        suma += estudiantes[i].calcularPromedio(); // Suma el promedio de cada estudiante
    }
    System.out.printf("Promedio general del curso: %.2f\n", suma / contadorEstudiantes);
}
```

- Busca un estudiante en el arreglo usando su cédula y lo devuelve si lo encuentra.

```
public static Estudiante buscarPorCedula(String cedula) {
    for (int i = 0; i < contadorEstudiantes; i++) {
        if (estudiantes[i].getCedula().equals(cedula)) {
            return estudiantes[i];
        }
    }
    return null;
}
```

- Muestra la lista de todos los estudiantes registrados.

```
public void mostrar() {
    for (int i = 0; i < contadorEstudiantes; i++) {
        System.out.println((i + 1) + ".-");
        estudiantes[i].imprimir();
    }
}
```

2.8 Habilidades blandas empleadas en la práctica

☒ Liderazgo
☒ Trabajo en equipo
☒ Comunicación asertiva
☒ La empatía
☒ Pensamiento crítico
☒ Flexibilidad
☒ La resolución de conflictos
☒ Adaptabilidad
☒ Responsabilidad

2.9 Conclusiones

- A lo largo del desarrollo de esta aplicación, pudimos comprobar lo útil que resulta usar la programación orientada a objetos junto con vectores para organizar la información de los estudiantes. Gracias a esto, fue posible registrar, modificar y eliminar datos de forma ordenada, haciendo que la gestión de los estudiantes sea más sencilla y eficiente.

- También logramos que cada estudiante tenga un seguimiento académico personalizado. Al permitir ingresar y editar sus calificaciones, y calcular su promedio, se facilita ver el progreso de cada uno, lo cual es muy importante para apoyar mejor su proceso de aprendizaje.

- Además, al calcular el promedio general del curso, se obtuvo una visión clara del rendimiento grupal. Esto ayuda a tomar decisiones más informadas y pensar en mejoras que beneficien tanto a los estudiantes como al desarrollo del curso en general.

2.10 Recomendaciones

- Se recomienda seguir utilizando la programación orientada a objetos y estructuras como los vectores para desarrollar aplicaciones educativas, ya que permiten una mejor organización de los datos y facilitan su manipulación.

- Para mejorar la experiencia del usuario, sería ideal implementar una validación más estricta al momento de ingresar datos, como evitar repeticiones de cédulas o formatos incorrectos en fechas, lo cual reduciría errores durante el uso de la aplicación.

2.11 Referencias bibliográficas

Bibliografía

[1] D. M. y. N. Malhotra, Data Structures and Program Design Using C++, Rockland, Massachusetts: Mercury Learning and Information, 2019.

[2] S. Guardati Buemo, Estructuras de datos basicos: programacion orientada a objetivos con Java., 2016.

2.12 Anexos

Clase estudiante

```java
package APE1;

import java.time.LocalDate;

public class Estudiante {
    String cedula, nombres,apellidos;
    LocalDate fechaNacimiento;
    double []calificaciones;
    int numCalificaciones;

    public Estudiante(String cedula, String nombres, String apellidos,
LocalDate fechaNacimiento) {
        this.cedula = cedula;
        this.nombres = nombres;
        this.apellidos = apellidos;
        this.fechaNacimiento = fechaNacimiento;
        this.calificaciones = new double[7];
    }

    public String getCedula() {
        return cedula;
    }

    public String getNombres() {
        return nombres;
    }

    public String getApellidos() {
        return apellidos;
    }

    public LocalDate getFechaNacimiento() {
        return fechaNacimiento;
    }

    public double[] getCalificaciones() {
        return calificaciones;
    }

    public int getNumCalificaciones() {
        return numCalificaciones;
    }

    public boolean agregarNota(double nota) {
        if (numCalificaciones < calificaciones.length) {
```

```
            calificaciones[numCalificaciones++] = nota;
            return true;
        }
        return false;
    }

    public boolean modificarNota(int indice, double nuevaNota) {
        if (indice >= 0 && indice < numCalificaciones) {
            calificaciones[indice] = nuevaNota;
            return true;
        }
        return false;
    }


    public boolean eliminarNota(int indice) {
        if (indice >= 0 && indice < numCalificaciones) {
            calificaciones[indice] = 0.0;
            for (int i = indice; i < numCalificaciones - 1; i++) {
                calificaciones[i] = calificaciones[i + 1];
            }
            numCalificaciones--;
            return true;
        }
        return false;
    }

    public float calcularPromedio() {
    float suma = 0;
    if (numCalificaciones > 0) {
        for (int i = 0; i < numCalificaciones; i++) {
            suma += calificaciones[i];
        }
        return suma / numCalificaciones;
    } else {
        return 0;
    }
}

    public void imprimir() {
        System.out.print("Cedula: " + cedula + " | Nombres: " + nombres +
" | Apellidos: " + apellidos +
                         " | Fecha de nacimiento: " + fechaNacimiento + "
| Notas: ");
        for (int i = 0; i < numCalificaciones; i++) {
        System.out.print(calificaciones[i] + " ");
    }
```

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

Clase Menú de estudiantes

```
package APE1;

import java.time.LocalDate;
import java.util.Scanner;

public class MenuEstudiante {
    private static Scanner scanner = new Scanner(System.in);
    private Estudiante[] estudiantes;
    private int contadorEstudiantes;

    public MenuEstudiante(Estudiante[] estudiantes, int
contadorEstudiantes) {
        this.estudiantes = estudiantes;
        this.contadorEstudiantes = contadorEstudiantes;
        mostrarMenu();
    }

    public int getContadorEstudiantes() {
        return contadorEstudiantes;
    }

    private void mostrarEstudiantes() {
        for (int i = 0; i < contadorEstudiantes; i++) {
            System.out.println((i + 1) + ".-");
            estudiantes[i].imprimir();
        }
    }

    private void mostrarMenu() {
        boolean reinicio = true;
        while (reinicio) {
            System.out.println("\nListado de estudiantes:");
            mostrarEstudiantes();
            System.out.println("""
                                        Menu de estudiantes
                                1. Ingresar estudiantes
                                2. Modificar estudiantes
                                3. Eliminar estudiantes
                                4. Salir
                            Escoja el numero de la opcion a la que desea
acceder""");
```

```
            int opsub1 = scanner.nextInt();
            scanner.nextLine();
            String ver;

            switch (opsub1) {
                case 1:
                    do {
                        if (contadorEstudiantes >= estudiantes.length) {
                            System.out.println("No se puede ingresar más
estudiantes. El curso ya está lleno.");
                            break;
                        }

                        System.out.println("Ingrese la cedula del
estudiante");
                        String cedula = scanner.nextLine();
                        System.out.println("Ingrese los nombres del
estudiante");
                        String nombre = scanner.nextLine();
                        System.out.println("Ingrese los apellidos del
estudiante");
                        String apellido = scanner.nextLine();
                        LocalDate fechaNacimiento = null;
                        while (fechaNacimiento == null) {
                            try {
                                System.out.print("Ingrese la fecha de
nacimiento (yyyy-MM-dd): ");
                                String fechaStr = scanner.nextLine();
                                fechaNacimiento =
LocalDate.parse(fechaStr);
                            } catch (Exception e) {
                                System.out.println("Formato inválido.
Intente nuevamente.");
                            }
                        }
                        estudiantes[contadorEstudiantes++] = new
Estudiante(cedula, nombre, apellido, fechaNacimiento);
                        System.out.println("Desea ingresar otro
estudiante? (S/N)");
                        ver = scanner.nextLine();
                    } while (ver.equalsIgnoreCase("s"));
                    break;
                case 2:
                    do {
                        if (contadorEstudiantes == 0) {
```

```
                            System.out.println("No hay estudiantes
registrados para modificar.");
                            break;
                        }
                        System.out.println("Ingrese el identificador del
estudiante a modificar:");
                        int id = scanner.nextInt();
                        scanner.nextLine();

                        if (id < 1 || id > contadorEstudiantes) {
                            System.out.println("ID inválido.");
                        } else {
                            System.out.println("Ingrese la nueva
cédula:");
                            String cedula = scanner.nextLine();
                            System.out.println("Ingrese los nuevos
nombres:");
                            String nombre = scanner.nextLine();
                            System.out.println("Ingrese los nuevos
apellidos:");
                            String apellido = scanner.nextLine();

                            LocalDate fechaNacimiento = null;
                            while (fechaNacimiento == null) {
                                try {
                                    System.out.print("Ingrese la nueva
fecha de nacimiento (yyyy-MM-dd): ");
                                    String fechaStr = scanner.nextLine();
                                    fechaNacimiento =
LocalDate.parse(fechaStr);
                                } catch (Exception e) {
                                    System.out.println("Formato
inválido.");
                                }
                            }

                            estudiantes[id - 1] = new Estudiante(cedula,
nombre, apellido, fechaNacimiento);
                            System.out.println("Estudiante modificado
correctamente.");
                        }
                        System.out.println("Desea modificar otro
estudiante? (S/N)");
                        ver = scanner.nextLine();
                    } while (ver.equalsIgnoreCase("s"));
                    break;
                case 3:
```

```
                    do {
                        if (contadorEstudiantes == 0) {
                            System.out.println("No hay estudiantes
registrados para eliminar.");
                            break;
                        }

                        System.out.println("Ingrese el número del
estudiante a eliminar:");
                        int id = scanner.nextInt();
                        scanner.nextLine();

                        int indexEliminar = id - 1;
                        if (indexEliminar >= 0 && indexEliminar <
contadorEstudiantes) {
                            estudiantes[indexEliminar] =
estudiantes[contadorEstudiantes - 1];
                            estudiantes[--contadorEstudiantes] = null;
                            System.out.println("Estudiante eliminado
correctamente.");
                        } else {
                            System.out.println("ID inválido.");
                        }
                        System.out.println("Desea eliminar otro
estudiante? (S/N)");
                        ver = scanner.nextLine();
                    } while (ver.equalsIgnoreCase("s"));
                    break;
                case 4:
                    reinicio = false;
                    break;
                default:
                    System.out.println("Opción inválida.");
            }
        }
    }
}
```

Clase Registrar calificaciones

```
package APE1;

import java.util.InputMismatchException;
import java.util.Scanner;

class RegistrarCalificaciones {
```

```
    private GestorPersonas gestorEstudiantes;
    private Scanner scanner;

    public RegistrarCalificaciones(GestorPersonas gestorEstudiantes) {
        this.gestorEstudiantes = gestorEstudiantes;
        this.scanner = new Scanner(System.in);
    }

    private boolean validarNota(double nota) {
        return nota >= 0 && nota <= 10;
    }

    public void ejecutar() {
        System.out.println("=== REGISTRO DE CALIFICACIONES ===");
        System.out.print("Ingrese la cedula del estudiante: ");
        String cedula = scanner.nextLine();

        Estudiante estudiante = gestorEstudiantes.buscarPorCedula(cedula);

        if (estudiante != null) {
            System.out.println("Datos del estudiante:");
            System.out.println("Nombres y Apellidos: " +
estudiante.getNombres() + " " + estudiante.getApellidos());

            double[] calificaciones = estudiante.getCalificaciones();
            int numCalificaciones = estudiante.getNumCalificaciones();

            do {
                System.out.println("\nCalificaciones registradas:");
                if (numCalificaciones == 0) {
                    System.out.println("No hay calificaciones registradas
para este estudiante.");
                } else {
                    for (int i = 0; i < numCalificaciones; i++) {
                        System.out.println((i + 1) + ". " +
calificaciones[i]);
                    }
                }

                System.out.println("\nOpciones de calificaciones:");
                System.out.println("1. Ingresar nueva calificacion");
                System.out.println("2. Modificar calificacion");
                System.out.println("3. Eliminar calificacion");
                System.out.println("4. Terminar registro de
calificaciones");
```

```
                System.out.print("Seleccione una opcion: ");
                int opcionCalificacion;
                try {
                    opcionCalificacion = scanner.nextInt();
                    scanner.nextLine();
                } catch (InputMismatchException e) {
                    System.out.println("Error:  Por  favor,  ingrese  un
número para la opción.");
                    scanner.nextLine();
                    continue;
                }

                switch (opcionCalificacion) {
                    case 1:
                        if (numCalificaciones < calificaciones.length) {
                            System.out.print("Ingrese      la      nueva
calificacion (0-10): ");
                            try {
                                double         nuevaCalificacion         =
scanner.nextDouble();
                                scanner.nextLine();
                                if (validarNota(nuevaCalificacion)) {
                                    if
(estudiante.agregarNota(nuevaCalificacion)) {
                                        System.out.println("Calificacion
agregada.");
                                        numCalificaciones               =
estudiante.getNumCalificaciones();
                                    }
                                } else {
                                    System.out.println("La    calificacion
debe estar entre 0 y 10.");
                                }
                            } catch (InputMismatchException e) {
                                System.out.println("Error:    Por    favor,
ingrese  un  numero  valido  para  la  calificacion  (use  un  punto  para
decimales).");
                                scanner.nextLine();
                            }
                        } else {
                            System.out.println("Se    ha    alcanzado    la
cantidad maxima de calificaciones posibles (" + calificaciones.length +
").");
                        }
                        break;
                    case 2:
```

```
                    if (numCalificaciones > 0) {
                        System.out.print("Ingrese el numero de la
calificacion a modificar: ");
                        int indiceModificar;
                        try {
                            indiceModificar = scanner.nextInt() - 1;
                            scanner.nextLine();
                            if (indiceModificar >= 0 &&
indiceModificar < numCalificaciones) {
                                System.out.print("Ingrese la nueva
calificacion (0-10): ");
                                try {
                                    double nuevaCalificacionModificar
= scanner.nextDouble();
                                    scanner.nextLine();
                                    if
(validarNota(nuevaCalificacionModificar)) {
                                        if
(estudiante.modificarNota(indiceModificar, nuevaCalificacionModificar)) {
                                            System.out.println("Calificacion modificada.");
                                        } else {
                                            System.out.println("Numero de calificación invalido.");
                                        }
                                    } else {
                                        System.out.println("La
calificacion debe estar entre 0 y 10.");
                                    }
                                } catch (InputMismatchException e) {
                                    System.out.println("Error: Por
favor, ingrese un numero valido para la calificacion (use un punto para
decimales).");
                                    scanner.nextLine();
                                }
                            } else {
                                System.out.println("Numero de
calificación invalido.");
                            }
                        } catch (InputMismatchException e) {
                            System.out.println("Error: Por favor,
ingrese un numero para el indice de la calificacion.");
                            scanner.nextLine();
                        }
                    } else {
```

```
                            System.out.println("No hay calificaciones para
modificar.");
                        }
                        break;
                    case 3:
                        if (numCalificaciones > 0) {
                            System.out.print("Ingrese  el  numero  de  la
calificacion a eliminar: ");
                            int indiceEliminar;
                            try {
                                indiceEliminar = scanner.nextInt() - 1;
                                scanner.nextLine();
                                if
(estudiante.eliminarNota(indiceEliminar)) {
                                    System.out.println("Calificacion
eliminada.");
                                    numCalificaciones                   =
estudiante.getNumCalificaciones();
                                } else {
                                    System.out.println("Numero          de
calificacion invalido.");
                                }
                            } catch (InputMismatchException e) {
                                System.out.println("Error:   Por   favor,
ingrese un numero para el indice de la calificacion.");
                                scanner.nextLine();
                            }
                        } else {
                            System.out.println("No hay calificaciones para
eliminar.");
                        }
                        break;
                    case 4:
                        System.out.println("Registro   de   calificaciones
terminado.");
                        return;
                    default:
                        System.out.println("Opcion invalida.");
                }
            } while (true);
        } else {
            System.out.println("No se encontro un estudiante con la cedula
ingresada.");
            System.out.print("¿Desea intentar con otra cedula? (s/n): ");
            String respuesta = scanner.nextLine().toLowerCase();
            if (respuesta.equals("s")) {
```

```
                ejecutar();
            }
        }
    }
}
```

Clase Gestor estudiantes (main)

```
package APE1;

import java.time.LocalDate;
import java.util.Scanner;

public class GestorPersonas {

    static Scanner scanner = new Scanner(System.in);
    static Estudiante[] estudiantes = new Estudiante[20];
    static int contadorEstudiantes = 0;
    static GestorPersonas gestor = new GestorPersonas();
    private RegistrarCalificaciones registroCalificaciones;

    public GestorPersonas() {
        this.registroCalificaciones = new RegistrarCalificaciones(this);
    }

    public static void calcularPromedioEstudiante() {
        System.out.print("Ingrese la cedula del estudiante: ");
        String cedula = scanner.nextLine();

        Estudiante estudiante = buscarPorCedula(cedula);
        if (estudiante != null) {
            float promedio = estudiante.calcularPromedio();
            System.out.println("Estudiante: " + estudiante.getNombres() +
" " + estudiante.getApellidos());
            System.out.println("Edad: " +
calcularEdad(estudiante.getFechaNacimiento()));
            System.out.printf("Promedio de calificaciones: %.2f\n",
promedio);
        } else {
            System.out.println("No se encontro un estudiante con esa
cedula.");
        }
    }

    public static void calcularPromedioCurso() {
        float suma = 0;
```

```
        if (contadorEstudiantes == 0) {
            System.out.println("No hay estudiantes registrados en el
curso.");
            return;
        }

        for (int i = 0; i < contadorEstudiantes; i++) {
            suma += estudiantes[i].calcularPromedio(); // Suma el
promedio de cada estudiante
        }

        System.out.printf("Promedio general del curso: %.2f\n", suma /
contadorEstudiantes);
    }

    public static int calcularEdad(LocalDate fechaNacimiento) {
        return LocalDate.now().getYear() - fechaNacimiento.getYear();
    }

    public static Estudiante buscarPorCedula(String cedula) {
        for (int i = 0; i < contadorEstudiantes; i++) {
            if (estudiantes[i].getCedula().equals(cedula)) {
                return estudiantes[i];
            }
        }
        return null;
    }

    public void mostrar() {
        for (int i = 0; i < contadorEstudiantes; i++) {
            System.out.println((i + 1) + ".-");
            estudiantes[i].imprimir();
        }
    }

    public static boolean hayEstudiantesRegistrados() {
        return contadorEstudiantes > 0;
    }

    public static boolean esArregloLleno() {
        return contadorEstudiantes >= estudiantes.length;
    }

    public static void main(String[] args) {
        int opcion;
        do {
            System.out.println("\n=== GESTOR DE PERSONAS ===");
```

```
            System.out.println("1. Estudiantes");
            System.out.println("2. Registro de calificaciones");
            System.out.println("3. Determinar el promedio de notas de un
estudiante");
            System.out.println("4. Determinar el promedio de notas del
curso");
            System.out.println("0. Salir");
            System.out.print("Teclee su opcion (0-4): ");
            opcion = scanner.nextInt();
            scanner.nextLine();

            switch (opcion) {
                case 1:
                    MenuEstudiante menuEstudiantes = new
MenuEstudiante(estudiantes, contadorEstudiantes);
                    contadorEstudiantes =
menuEstudiantes.getContadorEstudiantes();
                    break;
                case 2:
                    gestor.registroCalificaciones.ejecutar();
                    break;
                case 3:
                    calcularPromedioEstudiante();
                    break;
                case 4:
                    calcularPromedioCurso();
                    break;
                case 0:
                    System.out.println("Gracias por usar el sistema.");
                    break;
                default:
                    System.out.println("Opcion inválida.");
            }
        } while (opcion != 0);
    }
}
```