**Fundamentos de Programación – Entrega Práctica**

**Trabajo por:**

**Daniel Mauricio Giraldo Moreno**

**Profesor del área:**

**Edwin Nelson Montoya Munera**

**Ingeniería en sistemas – Semestre 1 - 2024-2**

**Universidad EAFIT, Medellín**

**5/11/2024**

**Descripción del proyecto**

**Objetivo de el programa:** El objetivo del programa es crear un sistema de gestión de notas que permita a los usuarios gestionar la información de estudiantes y sus materias en una universidad. Este sistema permitirá ingresar, almacenar y ver los datos de los estudiantes y sus calificaciones, calcular sus promedios, ordenar la lista e información de los estudiantes por promedio o nombre, eliminar estudiantes que hayan sido registrados en el programa de gestión de notas, y buscar a un estudiante específico para consultar sus materias y notas.

**Funcionalidad:** El sistema está diseñado en Java utilizando los principios de la programación orientada a objetos y hace uso de estructuras de datos como arreglos y matrices. Se muestra una interfaz en consola con un menú con el cual se puede interactuar que permite realizar las siguientes acciones:

1. Agregar estudiante

2. Eliminar estudiante

3. Mostrar lista de estudiantes

4. Calcular promedios

5. Ordenar estudiantes

6. Buscar estudiante

7. Salir

**Estructura de el programa:** El programa cuenta con las siguientes clases:

**Estudiante:** Representa a cada estudiante y contiene atributos como nombre, ID, y una lista de materias (arreglo dinámico); Incluye métodos para calcular el promedio de notas y agregar nuevas materias.

**Materia:** Representa a cada materia de un estudiante, con atributos como nombre, nota, y semestre.

**Gestión Estudiantes:** Se encarga gestionar la lista de estudiantes y proporcionar métodos para agregar, eliminar, mostrar, ordenar y buscar estudiantes.

**Main:** La clase principal donde se ejecuta el programa y se maneja el menú de opciones en consola.

**Requerimientos de este programa:**

**Ingreso de Datos:** Permite ingresar los datos de estudiantes, como nombre, ID, materias y sus notas. Los datos se almacenan en arreglos.

**Visualización de Datos:** Permite ver la lista de estudiantes con sus materias y notas, y ordenar la lista por nombre, materia o nota.

**Cálculo de Promedios:** Calcula el promedio de notas por estudiante y lo muestra en un formato claro.

**Ordenamiento:** Permite ordenar la lista de estudiantes por promedio mediante el algoritmo de ordenamiento bubblesort.

**Búsqueda:** Realiza una búsqueda de un estudiante por su ID y realiza tareas tales como indicar la información del estudiante asociado a ese ID o directamente borrar el estudiante.

**Mejoras:** El diseño del programa facilita el proceso en caso se quieran agregar funcionalidades futuras, como la gestión de asignaturas o el almacenamiento de datos en archivos.

**Funcionalidad del programa:**

**Ingreso de Estudiantes:** El sistema permite al usuario ingresar estudiantes con sus respectivos datos, incluyendo nombre, ID, y materias.Cada materia tiene atributos específicos, como semestre y nota.

**Visualización de Información:** El usuario puede ver una lista estructurada de todos los estudiantes junto con sus materias y notas.

**Cálculo de Promedios:** Se calcula por medio de un método el promedio de notas para cada estudiante especifico.

**Ordenamiento:** El sistema puede ordenar la lista de estudiantes por su promedio de notas.

**Búsqueda de Estudiantes y Gestión:** Permite buscar un estudiante específico ingresando su ID. Si el estudiante es encontrado, se muestran sus materias y notas, por medio de la búsqueda funcionan los métodos que permiten buscar y eliminar estudiantes de la lista.

**Aspectos que fueron necesarios tener en cuenta:**

**Uso de Programación Orientada a Objetos:** Se sigue el enfoque de POO, utilizando clases y objetos para estructurar correctamente el sistema.

**Estructura de Datos:** Se emplean arreglos y listas dinámicas (ArrayList) para gestionar la lista de estudiantes y sus materias.

**Separación de Lógica en Métodos:** Cada acción importante, como agregar o eliminar estudiantes, calcular promedios, y ordenar la lista, está encapsulada en métodos específicos. **Validación de Datos:** Se incluirá manejo **básico** de excepciones para validar entradas de usuario y evitar errores.

**Otros aspectos adicionales:**

**Posibilidad de Mejoras Futuras:** Se puede extender el programa para incluir otros aspectos como la gestión de materias y almacenamiento en archivos.

**Comentarios en el Código:** Todo el código fuente incluye comentarios claros para facilitar su comprensión y mantenimiento a largo plazo.

**Algoritmo**

**Descripción de el algoritmo:**

1. **Agregar Estudiante(s) con Materias - Propósito:** Permitir al usuario ingresar un nuevo estudiante con sus respectivas materias y notas.

**Pasos:**

1. Solicitar el nombre del estudiante y su ID.
2. Crear una instancia de Estudiante con los datos proporcionados.
3. Preguntar al usuario si desea agregar materias (1 para Sí, 2 para No).
4. Mientras el usuario desee agregar materias:
5. Solicitar el semestre de la materia.
6. Solicitar el nombre de la materia.
7. Solicitar la nota de la materia (recordando el formato correcto para números decimales que es usando la “,”).
8. Crear una instancia de Materia y agregarla a la lista de materias del estudiante.
9. Calcular el promedio de notas del estudiante tras añadir las materias.
10. Agregar el estudiante a la lista general de estudiantes en GestionEstudiantes.

**2. Eliminar un Estudiante por ID:**

**Propósito:** Eliminar a un estudiante ingresando su ID.

**Pasos:**

1. Pedir al usuario el ID del estudiante a eliminar.
2. Iterar (buscar) sobre la lista de estudiantes.
3. Si el ID ingresado coincide con el de algún estudiante, eliminarlo de la lista y mostrar un mensaje de confirmación.
4. Si el estudiante no se encuentra, mostrar un mensaje de error.

**3. Mostrar la Lista de Estudiantes con sus Materias:**

**Propósito:** Mostrar los datos básicos de cada estudiante y sus materias.

**Pasos:**

1. Comprobar si la lista de estudiantes está vacía. Si lo está, mostrar un mensaje de error.
2. Si la lista no esta vacía, usar un Iterator para recorrer la lista de estudiantes.
3. Para cada estudiante, llamar a mostrarInformacion() para imprimir nombre, ID y lista de materias.
4. Si un estudiante no tiene materias registradas, mostrar un mensaje indicando que no hay materias.

**4.** **Calcular el Promedio de Notas de Cada Estudiante:**

**Propósito:** Calcular el promedio de notas de todas las materias de cada estudiante.

**Pasos:**

1. Comprobar si la lista de estudiantes está vacía, en caso contrario, continuar.
2. Para cada estudiante:
3. Llamar al método calcularPromedio() en Estudiante.
4. Dentro de calcularPromedio(), recorrer todas las materias del estudiante y calcular el promedio.
5. **Ordenar Estudiantes por Promedio usando Bubble Sort:**

**Propósito**: Ordenar la lista de estudiantes de menor a mayor promedio de notas.

**Pasos**:

1. Verificar si la lista de estudiantes está vacía.
2. Aplicar el algoritmo de ordenamiento Bubble Sort a la lista de estudiantes:

2.1 Comparar los promedios de estudiantes adyacentes.

2.2 Si un estudiante tiene un promedio mayor que el siguiente, intercambiar sus posiciones.

1. Mostrar la lista de estudiantes ordenada de menor a mayor promedio.
2. **Buscar un Estudiante por ID y Mostrar su Información:**

**Propósito:** Buscar un estudiante específico por su ID y mostrar su información detallada.

**Pasos**:

1. Solicitar al usuario el ID del estudiante a buscar.
2. Iterar sobre la lista de estudiantes para encontrar un estudiante con el ID proporcionado.
3. Si se encuentra el estudiante:
   * Mostrar nombre e ID.
   * Usar un Iterator para recorrer y mostrar sus materias.
4. Si el estudiante no tiene materias, mostrar un mensaje indicándolo.
5. Si no se encuentra el estudiante, mostrar un mensaje de error.

**CODIGO DE HONOR Y ÉTICA**

Declaro que el código fuente de este proyecto fue realizado por mí. Para aclaraciones, dudas y algunas consultas utilicé la herramienta de inteligencia artificial ChatGPT como apoyo en la explicación de conceptos y en la organización de los pasos del proyecto, la implementación y lógica del código fueron desarrolladas por mí. No copié ninguna parte de este proyecto a terceros, y me siento capaz de demostrar mi conocimiento y sustentar el código.

**Link – GitHub:** <https://github.com/dani-g11/Practica.git>

**El archivo adicional entregado contiene el código completo y todo lo que se ha pedido para el desarrollo de la actividad.**