UNIVERSIDAD AMERICANA

Facultad de Ingeniería y Arquitectura

ALGORITMOS Y ESTRUCTURAS DE DATOS - GRUPO 2

Código #1

Nombre:

- Freddy Adrian peralta
- Jose Gabriel Cano Blandón
- Marcos Alessandro Lagos Rivera
- Axell Antonio Castillo Tapia

Docente

- Silvia

Managua, Nicaragua
28 de abril del 2024

El **parámetro de ordenamiento** puede ser cualquiera de los siguientes campos: carnet, nombres, apellidos, peso, estatura, sexo o promedio

El programa está compuesto por cuatro clases principales:

- Estudiante
- Nodo
- ListaEnlazada
- Validaciones
- Un programa principal (main()) que interactúa con el usuario.

Funcionamiento del Código

1. Clase `Estudiante`

- Define los **atributos** que caracterizan a un estudiante: `carnet`, `nombres`, `apellidos`, `peso`, `estatura`, `sexo` y `promedio` .
- Implementa el método `__str__` para devolver una representación **formateada** de los datos del estudiante

```python
class Estudiante:
    def __init__(self, carnet, nombres, apellidos, peso, estatura, sexo, promedio):
        self.carnet = carnet
        self.nombres = nombres
        self.apellidos = apellidos
        self.peso = peso
        self.estatura = estatura
        self.sexo = sexo
        self.promedio = promedio

    def __str__(self):
        return f"{self.carnet} - {self.nombres} {self.apellidos}, Peso: {self.peso}kg, Estatura: {self
```

2. Clase `Nodo`

- Sirve como **contenedor** de un objeto `Estudiante` y mantiene la referencia (`siguiente`) al próximo nodo de la lista.

- Es fundamental para construir la **lista enlazada**.

```
1    class Nodo:
2        def __init__(self, estudiante):
3            self.estudiante = estudiante
4            self.siguiente = None
```

3. Clase `ListaEnlazada`

- Se utiliza para almacenar los estudiantes en orden **ascendente** según un campo especificado al momento de crear la lista (`clave_orden`) .

Métodos:

- `insertar_ordenado(self, estudiante)`:
  Inserta estudiantes de forma que se mantenga el orden basado en la clave de ordenación.
  - Si la lista está vacía o el nuevo elemento debe ir al principio, lo inserta como cabeza (líneas 9-13).
  - De lo contrario, recorre la lista y encuentra la posición adecuada para insertarlo (líneas 14-19).

- `mostrar(self)`:
  Recorre y **muestra** en pantalla los estudiantes en el orden actual de la lista (líneas 21-24).

```
from nodo import Nodo

class ListaEnlazada:
    def __init__(self, clave_orden):
        self.cabeza = None
        self.clave_orden = clave_orden

    def insertar_ordenado(self, estudiante):
        nuevo_nodo = Nodo(estudiante)
        valor_nuevo = getattr(estudiante, self.clave_orden)

        if self.cabeza is None or getattr(self.cabeza.estudiante, self.clave_orden) > valor_nuevo:
            nuevo_nodo.siguiente = self.cabeza
            self.cabeza = nuevo_nodo
        else:
            actual = self.cabeza
            while (actual.siguiente is not None and
                   getattr(actual.siguiente.estudiante, self.clave_orden) <= valor_nuevo):
                actual = actual.siguiente

            nuevo_nodo.siguiente = actual.siguiente
            actual.siguiente = nuevo_nodo

    def mostrar(self):
        actual = self.cabeza
        while actual is not None:
            print(actual.estudiante)
            actual = actual.siguiente
```

4. Clase validaciones

Este módulo contiene una función para validar los datos personales y académicos de un individuo. Su objetivo principal es asegurar que los datos ingresados cumplan con criterios básicos de formato, tipo y rango, antes de ser procesados o almacenados.

Parámetros:

- `carnet` (str): Identificación del usuario. Debe ser un número con al menos 5 dígitos.
- `nombres` (str): Nombres del usuario. Solo se permiten letras (incluye tildes y la letra ñ) y espacios.
- `apellidos` (str): Apellidos del usuario. Mismas restricciones que los nombres.
- `peso` (float): Peso corporal. Debe ser un número positivo.
- `estatura` (float): Estatura corporal en metros. Debe estar entre 0 y 3 metros.
- `sexo` (str): Sexo del usuario. Solo se aceptan los valores `'M'` (masculino) o `'F'` (femenino).
- `promedio` (float): Promedio académico. Debe estar entre 0 y 100.

5. Funciones Principales

- `pedir_estudiante()`:
  Solicita los datos del estudiante al usuario y asegura que el no se repita (líneas 6-19 del `main.py`).
- `mostrar_menu()`:
  Imprime las opciones disponibles para el usuario (líneas 21-25).
- `main()`:
  - Muestra el menú y recibe la opción seleccionada por el usuario (líneas 28-30).
  - Si elige ingresar un nuevo estudiante, utiliza `pedir_estudiante()` y agrega el estudiante a una lista (`estudiantes_guardados`) (líneas 32-35).
  - Si elige mostrar estudiantes ordenados:

        - Pregunta por el campo de ordenamiento deseado.
        - Crea una nueva `ListaEnlazada` con ese campo como clave.
        - Inserta los estudiantes ya registrados en la lista enlazada en el orden correcto.
        - Muestra los estudiantes ordenados (líneas 36-51).
    - Si elige salir, el programa finaliza (líneas 52-54).
    - Valida si la opción ingresada es válida (líneas 55-56).

Clase `ListaEnlazada`

Esta clase sirve para **guardar estudiantes** en una estructura especial llamada **lista enlazada**, asegurándose de que **siempre estén ordenados** automáticamente **según un campo** que tú decidas (por ejemplo, `promedio`, `carnet`, `peso`, etc.).

Cuando creas una `ListaEnlazada`, debes decirle **por cuál campo quieres ordenar** (eso es el `clave_orden`).

Métodos principales:

1. `insertar_ordenado(self, estudiante)`

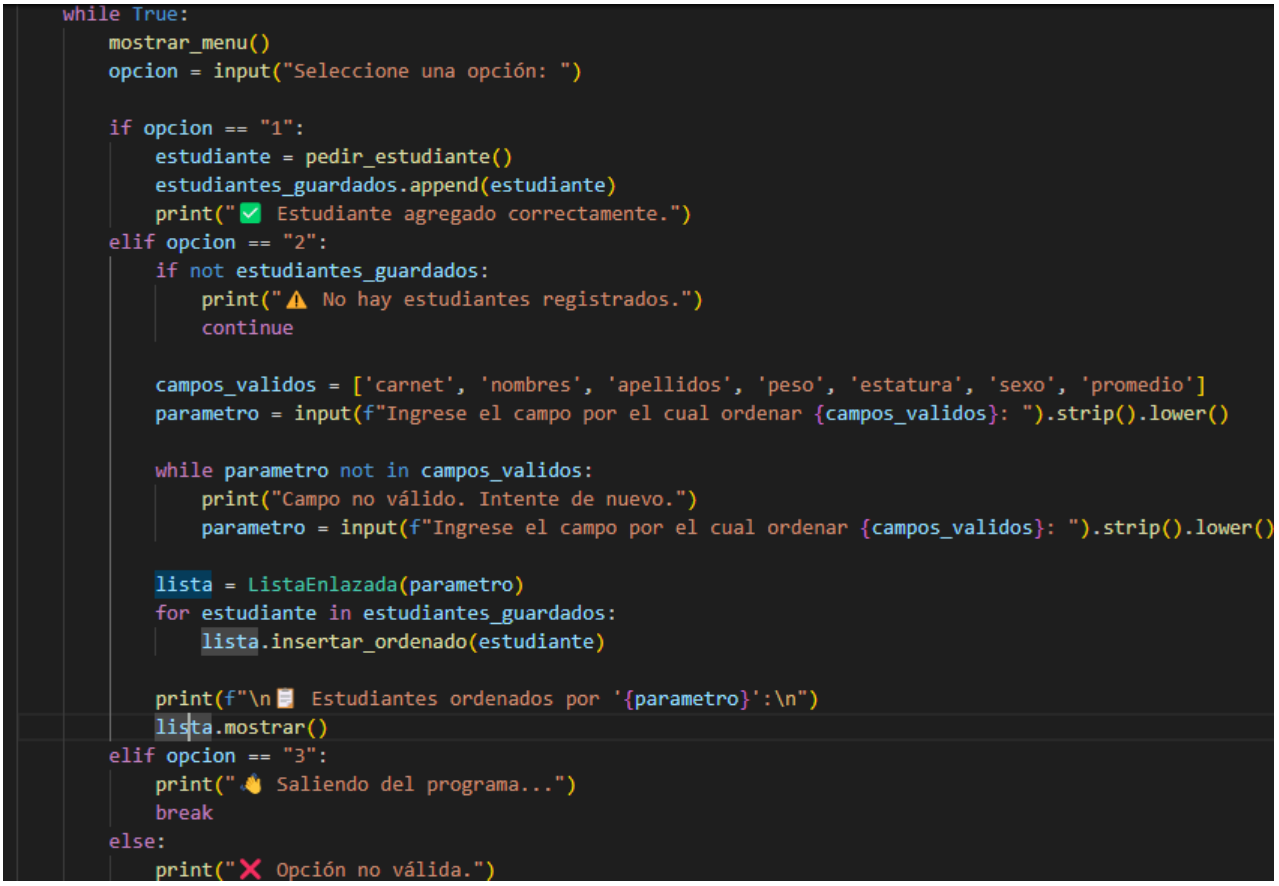
- **¿Qué hace?**
  Inserta un nuevo estudiante en la lista **respetando el orden ascendente** por el campo elegido (`clave_orden`).
- **¿Cómo lo hace?**
    - **Si la lista está vacía** o **el estudiante debe ir al inicio** (porque su valor de orden es menor que el de los demás), entonces **se convierte en la cabeza** de la lista.
    - **Si no**, **recorre uno por uno** los estudiantes ya insertados, **buscando la posición exacta** donde el nuevo estudiante debe colocarse (antes de un estudiante que tenga un valor mayor).

---

2. `mostrar(self)`

- **¿Qué hace?**
    - **Recorre toda la lista** (desde la cabeza hasta el final).
    - **Muestra cada estudiante** en el orden en que están conectados.

```python
from Estudiante import Estudiante
from listaenlazada import ListaEnlazada

ids_usados = set()

def pedir_estudiante():
    while True:
        carnet = input("Carnet (ID): ")
        if carnet in ids_usados:
            print("⚠️ Este carnet ya fue registrado. Intente con otro.")
        else:
            ids_usados.add(carnet)
            break

    nombres = input("Nombres: ")
    apellidos = input("Apellidos: ")
    peso = float(input("Peso (kg): "))
    estatura = float(input("Estatura (m): "))
    sexo = input("Sexo (M/F): ")
    promedio = float(input("Promedio: "))
    return Estudiante(carnet, nombres, apellidos, peso, estatura, sexo, promedio)

def mostrar_menu():
    print("\n--- MENÚ ---")
    print("1. Ingresar nuevo estudiante")
    print("2. Mostrar estudiantes ordenados")
    print("3. Salir")

def main():
    estudiantes_guardados = []

    while True:
```



```python
while True:
    mostrar_menu()
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        estudiante = pedir_estudiante()
        estudiantes_guardados.append(estudiante)
        print("✅ Estudiante agregado correctamente.")
    elif opcion == "2":
        if not estudiantes_guardados:
            print("⚠️ No hay estudiantes registrados.")
            continue

        campos_validos = ['carnet', 'nombres', 'apellidos', 'peso', 'estatura', 'sexo', 'promedio']
        parametro = input(f"Ingrese el campo por el cual ordenar {campos_validos}: ").strip().lower()

        while parametro not in campos_validos:
            print("Campo no válido. Intente de nuevo.")
            parametro = input(f"Ingrese el campo por el cual ordenar {campos_validos}: ").strip().lower()

        lista = ListaEnlazada(parametro)
        for estudiante in estudiantes_guardados:
            lista.insertar_ordenado(estudiante)

        print(f"\n📄 Estudiantes ordenados por '{parametro}':\n")
        lista.mostrar()
    elif opcion == "3":
        print("👋 Saliendo del programa...")
        break
    else:
        print("❌ Opción no válida.")
```

Citas de Código Clave

Validación de carnet único:

python
CopiarEditar

```
if carnet in ids_usados:
    print("⚠️ Este carnet ya fue registrado. Intente con otro.")
```

-

Inserción ordenada en la lista enlazada:

python
CopiarEditar

```
while (actual.siguiente is not None and
       getattr(actual.siguiente.estudiante, self.clave_orden) <=
valor_nuevo):
    actual = actual.siguiente
```

- (líneas 16-18 en `listaenlazada.py`)

Creación de la lista enlazada basada en el campo indicado:

python
CopiarEditar
```
lista = ListaEnlazada(parametro)
for estudiante in estudiantes_guardados:
    lista.insertar_ordenado(estudiante)
```

- (líneas 44-47 en `main.py`)

Muestra de estudiantes ordenados:

python
CopiarEditar
```
lista.mostrar()
```

- (línea 49 en `main.py`)

Conclusión

Este programa proporciona una solución completa al problema planteado: permite ingresar estudiantes, evita duplicados en carnets y muestra los datos **ordenados** según el campo que el usuario indique, utilizando estructuras de datos como **listas enlazadas**.