Laboratorio 2

Implementación de Listas Enlazadas en Python con Programación Orientada a Objetos

Temas: Listas Enlazadas Simples y Listas Enlazadas Circulares.

Modalidad de entrega: Implementación en Python (Obligatorio el uso de Programación Orientada a Objetos) **Debe entregarse en Google Collab**
Fecha de entrega: Jueves 31 de octubre

Objetivos:

- Aplicar los conceptos de Programación Orientada a Objetos (POO) para diseñar e implementar estructuras de datos dinámicas.
- Resolver un problema real utilizando listas enlazadas simples, listas doblemente enlazadas, y listas enlazadas circulares.
- Desarrollar habilidades para diseñar soluciones eficientes mediante operaciones sobre estructuras de datos.

Descripción del problema:

Imagina que estás diseñando un sistema para gestionar un centro de atención telefónica de una gran empresa. El centro recibe llamadas de distintos clientes, que deben ser atendidas según diferentes prioridades y reglas. El reto es implementar un sistema eficiente que permita gestionar las llamadas entrantes, cambiar el estado de las llamadas en la lista y eliminar aquellas que ya han sido atendidas.

El sistema deberá soportar tres tipos de llamadas:

- Llamadas regulares: Que se atienden en el orden de llegada.
- Llamadas VIP: Que tienen prioridad y deben insertarse al inicio de la lista.
- Llamadas circulares: Un conjunto de clientes premium que pueden estar en una lista circular para ser atendidos repetidamente si no se solucionan sus problemas en un solo llamado.

Deberás modelar esta solución utilizando listas enlazadas para representar la lista de llamadas en el sistema de atención.

Requerimientos:

1. **Lista Enlazada Simple**: lista de Llamadas Regulares

Implementar una clase **Llamada** que represente una llamada telefónica con la siguiente información:

- ID del cliente.
- Nombre del cliente.
- Prioridad de la llamada (regular, VIP)
- Estado de la llamada (pendiente, en proceso, finalizada).

Implementar una clase **ListaEnlazadaSimple** para gestionar la lista de llamadas regulares.

Esta estructura debe permitir:

- Añadir una nueva llamada al final de la lista (simulando el orden de llegada).
- Eliminar una llamada cuando se haya atendido (puede ser la primera en la lista).
- Mostrar todas las llamadas en la lista (con su estado actual).
- Buscar una llamada específica por su ID.

2. **Lista Enlazada Circular**: Lista de Clientes Premium

Implementar una clase **LlamadaPremium**, que incluya una prioridad aún mayor.
Implementar una clase **ListaEnlazadaCircular** para gestionar la lista circular de clientes premium, que están en un ciclo de atención constante.

Esta estructura debe permitir:

- Añadir un cliente premium al ciclo de atención.
- Eliminar un cliente del ciclo si ya no requiere atención continua.
- Mostrar el ciclo de atención actual recorriendo la lista de manera circular.
- Buscar un cliente premium específico por su ID.

Casos de Uso:

- **Caso 1:** Un cliente regular llama al centro de atención. El sistema debe añadir la llamada al final de la lista regular. Luego, una vez que la llamada es atendida, debe ser eliminada de la lista.

- **Caso 2:** Un cliente VIP llama al centro de atención. La llamada VIP debe añadirse al principio de la lista, para asegurarse de que se atiende antes que cualquier cliente regular.

- **Caso 3:** Un cliente premium no logra resolver su problema en la primera llamada y pasa a la lista de clientes en atención continua. Este cliente debe estar en una lista circular que permite su atención repetida hasta que su problema esté completamente resuelto.

- **Caso 4:** Se realiza una búsqueda por el ID de un cliente en cualquiera de las listas para mostrar su estado actual y los detalles de su llamada.

Criterios de Evaluación:

- Correctitud del código: Se evaluará si la solución cumple con los requisitos del problema y si las operaciones de gestión de llamadas se realizan correctamente.
- Uso de Programación Orientada a Objetos (POO): La solución debe estar diseñada utilizando clases y métodos para cada una de las estructuras y tipos de llamadas.
- Eficiencia y calidad del código: Se valorará el uso eficiente de las listas enlazadas y la organización del código, incluyendo comentarios y buenas prácticas de programación.
- Casos de prueba y funcionalidad: Deberán incluir ejemplos que muestren cómo el sistema gestiona diferentes tipos de llamadas y su evolución en la lista.

Entregables:

- Código fuente en un archivo .py que implemente las estructuras de datos solicitadas y las operaciones descritas.
- Informe en formato PDF que incluya:
    - Explicación de la solución al problema planteado.
    - Descripción de las clases, métodos y operaciones.
    - Casos de prueba y resultados obtenidos.
    - Comentarios y documentación en el código para facilitar su lectura y entendimiento.

Consideraciones:

- Asegúrate de manejar de manera adecuada la gestión de memoria en las listas enlazadas, especialmente en la lista circular.
- Los métodos deben estar diseñados de forma clara y eficiente, buscando evitar redundancias en el código.
- El sistema debe ser capaz de manejar grandes volúmenes de llamadas sin degradar su rendimiento.

Evaluación tridimensional

La evaluación del laboratorio será de la siguiente forma:

Dimensión	**Componente**	**Porcentaje**
El producto (lo que hizo)	Requerimientos funcionales	40%
	Requerimientos no funcionales	30%
El Proceso (cómo lo hizo)	Metodología de diseño (algoritmos y estructuras)	20%
Informe (justificación de porqué lo hizo así) (Individual)	Informe en PDF	10%

¡Mucho éxito en el desarrollo del laboratorio!