LAB 2

Votación:

En la facultad de ingeniería y ciencias de la universidad Diego Portales se llevarán a cabo las elecciones para presidente del centro de alumnos de la escuela de informática y telecomunicaciones.

A usted se le ha solicitado crear un sistema de votaciones llamado "Elector", que permitirá gestionar candidatos, votantes y resultados de elecciones utilizando listas enlazadas, pilas y colas. El sistema deberá garantizar la integridad de los votos, evitar duplicaciones y permitir consultas eficientes sobre los resultados.

2. Implementación

Debes implementar las siguientes clases, según el diagrama proporcionado (similar al de *Slok*, pero adaptado a un sistema de votaciones):

Clase `Voto`

Atributos:

- `id` (entero, único para cada voto).
- `votanteID` (ID del votante, entero, utilizado para evitar votos duplicados).
- `candidatoID` (ID del candidato seleccionado, entero).
- `timestamp` (fecha y hora del voto en formato “hh:mm:ss”).

Métodos:

- Constructor, getters y setters.

Clase `Candidato`

Atributos:

- `id` (identificador único, entero).
- `nombre` (cadena de caracteres).
- `partido` (cadena de caracteres).
- `votosRecibidos` (cola de votos asociados, Voto).

Métodos:

- `agregarVoto(Voto v)`: Añade un voto a la cola de votos del candidato.

Clase `Votante`

Atributos:

- `id` (entero, único).
- `nombre` (cadena de caracteres).
- `yaVotó` (booleano).

Métodos:

- `marcarComoVotado()`: Cambia `yaVotó` a `true`.

Clase `UrnaElectoral`

Atributos:

- `listaCandidatos` (lista enlazada de candidatos).
- `historialVotos` (pila para almacenar votos en orden cronológico inverso (la pila recibirá sólo Votos)).
- `votosReportados` (cola para votos anulados o impugnados (La cola recibirá solo Votos)).
- `idCounter` (contador de IDs para votos, entero).

Métodos clave:

- `verificarVotante(Votante votanteID)`: Verifica si el votante ya ha votado.
- `registrarVoto(Votante votanteID, int candidatoID)`:Utiliza el metodo `verificarVotante` , luego registra el voto (Se tiene que crear un nuevo voto para agregarlo al candidato) en el candidato correspondiente y lo añade al historial. Posteriormente tiene que cambiar el estado del votante.
- `reportarVoto(Candidato candidatoID, int idVoto)`: Mueve un voto de la cola correspondiente al candidato a la cola de votos reportados (Entrega un mensaje en caso de existir ya el voto en la cola por ejemplo, en caso de fraude).
- `obtenerResultados()`: Retorna un mapa con el conteo de votos por candidato.

3. Análisis

Complejidad Temporal:

Crea una tabla de notación Big-O para los métodos clave (, `registrarVoto`, `obtenerResultados, reportarVoto`).

Gestión de Memoria:

Calcula el espacio requerido para almacenar todos los votos si:

- Cada voto ocupa 64 bytes.
- El sistema admite hasta 10 millones de votantes.

Propuesta de Mejora:

¿Cómo modificarías el sistema para soportar votaciones en múltiples facultades?

Obs: ¿Cómo podrías mantener el historial de votos en orden cronológico inverso?

Explica brevemente.

4. Informe

El informe debe incluir:

- Introducción al problema.
- Descripción de la implementación (clases auxiliares, pruebas realizadas).
- Análisis de complejidad y uso de memoria.
- Ventajas y desventajas de utilizar listas enlazadas en lugar de arreglos para este sistema.
- Conclusiones y posibles extensiones (por ejemplo, votación electrónica segura).
-

Objetivos del laboratorio:

- Evaluar el dominio de listas, pilas y colas en un contexto realista.
- Practicar diseño de clases y gestión eficiente de memoria.
- Analizar ventajas y desventajas entre distintas estructuras de datos.