Universidad Diego Portales
Escuela de Informática &
Telecomunicaciones

LAB 2:

Votación Centro de Alumnos

Autores:
Martin Correa
Joakin Mac-Auliffe
Sergio Pinto

Profesor:
Marcos Fantoval

Link de GitHub

Índice

1. Introducción

En el presente laboratorio, se llevará a cabo un pedido de la Facultad de Ingeniería y Ciencias de la Universidad Diego Portales, el cual consiste en el desarrollo de un programa, que permita llevar a cabo una votación de manera íntegra y segura para elegir al presidente del Centro de Alumnos de la escuela de Informática y Telecomunicaciones. Para ello, se desarrollará un programa en Java que utilizará diversos métodos de almacenamiento de datos, incluyendo listas enlazadas, colas, pilas, entre otros.

2. Metodología

En esta sección se explicará el funcionamiento del programa, describiendo de forma detallada todas las clases, junto con sus atributos y métodos respectivamente.

2.1. Clase Votante

- **Atributos**
  - int id: Guarda en una variable de tipo entero el número de identificación del votante.
  - String nombre: Guarda en una variable de tipo cadena de caracteres el nombre del votante.
  - Boolean yaVoto: Variable de tipo booleano que es verdadero en caso de que haya ingresado un voto, y falso en caso contrario.
- **Métodos**
  - Votante(int id, String nombre): Constructor que asigna valor a 'int id' y 'String nombre'.
  - void marcarComoVotado(): Método que cambia el valor de 'Boolean yaVoto' a verdadero cuando se ingresa un voto.

2.2. Clase Voto

- **Atributos**
  - int id: Guarda en una variable de tipo entero el número de identificación del voto.
  - int votante ID: Guarda en una variable de tipo entero el número de identificación del votante.
  - int candidatoID: Guarda en una variable de tipo entero el número de identificación del candidato.

- int[] timestamp: Arreglo de tipo entero que almacena el instante de tiempo en el que se realizó la votación.

- **Métodos**
  - Voto(int votanteID, int cantidatoID, int[] timestamp): Constructor que asigna valor a 'int votanteID', 'int candidatoID' y 'int timestamp'.
  - int getId(): Retorna el ID voto.
  - int getVotanteID(): Retorna el ID del votante.
  - int getCantidatoID(): Retorna el ID del candidato.
  - int[] getTimestamp(): Retorna el instante de tiempo en el que se efectuó el voto.
  - setId(int id): Asigna valor a 'int id'.
  - setVotanteID(int votanteID): Asigna valor a 'int votanteID'.
  - setCantidatoID(int cantidatoID): Asigna valor a 'int candidatoID'.
  - void setTimestamp(int[] timestamp): Asigna valor a 'int[] timestamp'.

2.3. Clase Candidato

- **Atributos**
  - int id: Guarda en una variable de tipo entero el número de identificación del candidato.
  - String nombre: Guarda en una variable de tipo cadena de caracteres el nombre del candidato.
  - String partido: Guarda en una variable de tipo cadena de caracteres el nombre del partido al que pertenece el candidato.
  - Queue <Voto> votosRecibidos: Cola que almacena objetos de tipo 'Voto'. Representa los votos recibidos por el candidato.

- **Métodos**
  - void anadeVoto(Voto v): Método que añade un nuevo voto a la cola 'votosRecibidos'.
  - Voto retirarVoto(int idVoto): Se utiliza un iterador para recorrer la cola 'votosRecibidos' en busca de un voto que coincida con la id ingresada. Si se encuentra, el voto es eliminado de la cola y es retornado. En caso contrario, se retorna null.
  - int getID(): Retorna el número de identificación del candidato.

- String getNombre(): Retorna el nombre del Candidato.
- String getPartido(): Retorna el nombre de partido al que pertenece el candidato.
- int getVotos(): Retorna el tamaño de la cola 'votosRecibidos', cuyo valor representa la cantidad de votos que obtuvo el candidato.
- void setID(int id): Asigna valor al número de identificación del candidato.

2.4. Clase LinkedList

- **Atributos**
  - Node head: Guarda el primer nodo de la lista enlazada.
- **Métodos**
  - Linkedlist(): Constructor de la clase LinkedList, le asigna valor 'null' a head.
  - void insert(Candidato candidato): Agrega un nodo de tipo 'Candidato' a la lista enlazada, además le asigna una ID correspondiente.

2.4.1. Clase Node

Clase necesaria para el funcionamiento de la clase 'LinkedList', encargada de crear cada uno de los nodos que componen la lista.

- **Atributos**
  - Candidato value: Corresponde al tipo de valor que va a almacenar cada nodo. En este caso, objetos de tipo 'Candidato'.
  - Node next: Almacena la referencia al nodo siguiente en la lista.
- **Métodos**
  - Node(Candidato value): Constructor de la clase. Le asigna valor a 'value'.

2.5. Clase UrnaElectoral

- **Atributos**
  - LinkedList listaCandidatos: Lista enlazada que guarda los datos de cada candidato ingresado dentro de la urna electoral.
  - Stack<Voto> historialVotos: Pila que guarda los votos válidos, para así mantener un expediente.

- Queue<Voto> votosReportados: Cola utilizada para agregar votos reportados como repetidos.
- int idCounter: Variable de tipo entero, que funciona como contador para asignar un ID a cada voto realizado.

- **Métodos**
  - UrnaElectoral(Linkedlist listaCandidatos): Constructor de la clase.
  - boolean verificarVotante(Votante votanteID): Método utilizado para verificar si el votante ya tiene un voto registrado.
  - void registrarVoto(Votante votanteID, int candidatoID): Método utilizado para registrar un nuevo voto. Verifica si el voto es válido, asegurándose que el votante no haya votado anteriormente, además de comprobar que el candidato sea válido. Una vez constatado, crea el voto y lo almacena en la pila 'historialVotos'.
  - reportarVoto(Candidato candidatoID, int idVoto): Método utilizado para reportar sobre algún voto que necesite ser retirado y así añadirlo a la pila 'votosReportados'.
  - void obtenerResultados(): Método que imprime los resultados de las votaciones. Se utilizó un map como variable auxiliar para almacenar de manera más accesible la información de cada candidato. Para luego recorrer dicho map imprimiendo en pantalla el nombre del candidato, su ID, el partido al que pertenece y la cantidad de votos que recibió.

2.6. Main

Se llama a las clases del código, para obtener los resultados de la votación y verificar que cada método funcione de manera correcta. Se realizó de la siguiente manera:

- LinkedList ll = new LinkedList(): Utilizado para inicializar la clase LinkedList.
- Candidato c = new Candidato("x", ɛ"): Utilizado múltiples veces para crear objetos de tipo Candidato, ingresando distintos valores para cada uno.
- Votante v = new Votante(1, "Votante 1"): Utilizado múltiples veces para crear un objeto de tipo Votante, ingresando distintos valores para cada uno.
- ll.insert(c): Se agrega cada uno de los candidatos a la lista enlazada.
- UrnaElectoral ue = new UrnaElectoral(ll): Se instancia la clase 'UrnaElectoral' dentro del main, y se ingresa como parámetro la lista enlazada 'll' que contiene todos los candidatos.
- ue.obtenerResultados(): Llama al método void obtenerResultado() para ver los resultados de la votación.

- ue.registrarVoto(v1, c.getID()): Llama al método void registrarVoto con los parámetros del ID del votante (v1) y el ID del candidato (c.getID()). Con el objetivo de verificar si el votante ya ha emitido su voto, si su ID no ha sido registrado o si es la primera vez que vota. En caso de que sea su primer voto, se registra en la cola correspondiente. Este proceso se repite para todos los demás votos.
- ue.reportarVoto(c, 1): Se llama al método void reportarVoto con los parámetros del ID del candidato (c) y el ID del voto (1) con el propósito de verificar que el método elimine correctamente un voto inválido y lo registre en la cola votosReportados.

3. Análisis

Tabla de complejidad en notación Big-O para los métodos(registrarVoto(), obtenerResultados() y reportarVoto()).

	Notación Big-O
registrarVoto()	O(N)
obtenerResultados()	O(N)
reportarVoto()	O(N)

3.1. Gestión de Memoria

Si se considera que cada voto requiere de 64 bytes de memoria. ¿Cuánta memoria se necesita para registrar 10 millones de votos?

$$64 \times 10{,}000{,}000 = 640{,}000{,}000 \text{ bytes}$$

$$640{,}000{,}000 \div 1024 = 625000 \text{ KB}$$

$$625000 \div 1024 \approx 610,35 \text{ MB}$$

Se requieren aproximadamente 610,35 MB de espacio para almacenar 10 millones de votos.

3.2. Propuesta de Mejora

¿Como modificarías el sistema para soportar votaciones en múltiples facultades? Explica brevemente.

Para realizar votaciones en distintas facultades se podría implementar una nueva clase llamada 'Facultad de (nombre de la facultad)', con atributos de tipo 'UrnaElectoral' para poder acceder a las otras clases y gestionar sus propios métodos.

3.3. Ventajas y desventajas de usar LinkedList y Arreglos

La utilización de listas enlazadas permite un mejor tiempo de ejecución y una mayor eficiencia. Esto se debe a que las listas enlazadas pueden crecer de manera dinámica, a diferencia de los arreglos, que requieren la creación de un arreglo auxiliar para aumentar su tamaño, lo que disminuye la eficiencia del código. Además, las listas enlazadas permiten insertar o eliminar elementos en tiempo constante siempre que se disponga de la referencia al nodo correspondiente.

La principal desventaja de utilizar listas enlazadas en lugar de arreglos en el programa es la pérdida de acceso rápido a cualquier elemento, ya que en las listas enlazadas se debe recorrer nodo por nodo para llegar a un elemento específico.

4. Conclusión

En base a los objetivos planteados y tomando en cuenta los requerimientos del programa, se puede considerar que el laboratorio fue un éxito, dado que se lograron implementar correctamente todos los métodos necesarios para que el programa funcione de manera íntegra y segura. El proyecto representó un gran aporte al conocimiento del equipo en relación a la programación orientada a objetos, debido a las diversas clases lo componen. Adicionalmente, puso a prueba todos los conocimientos del equipo en el manejo de estructuras de datos tales como Linked List, colas y stacks. Por último, el proyecto ayudó a comprender mejor la importancia de un código ordenado y bien documentado, ya que al haber tantas clases y métodos, se tuvo que estructurar muy detalladamente el código, de lo contrario, avanzar de manera eficiente habría resultado considerablemente más difícil.

Como posibles extensiones, se podría considerar el desarrollo de una clase 'Facultad (nombre de la facultad)', para que otras facultades puedan realizar votaciones dentro del programa. Además, se podría implementar una interfaz gráfica (GUI) para hacer la votación más intuitiva y accesible para el usuario.