Edurne Abian y Pablo Martinez

PRÁCTICA 1:

CLUB DE SOCIOS. 2ª EVAL.

# PRÁCTICA 1: Implementación de un Club de Socios

En esta práctica, se tiene que diseñar un programa adaptando la utilización de la Estructura de Datos que mejor se adapte para cada una de las funcionalidades.

El programa Java diseñado va a ser capaz de realizar las acciones comprendidas en un menú; como eliminar o añadir un socio, preguntar sobre la pertenencia o no de un socio a un club, dar de alta o baja un socio, visualizar los socios del club…

## **desarrollo:**

Para la estructura de datos del programa se ha elegido crear un objeto, al que se le llama “Club”, a modo de lista enlazada doble, que contendrá el nombre del club, el primer nodo, el último nodo y su tamaño (número de nodos asociados).

Los nodos se han creado como clase Nodo que contiene los atributos de contenido (objeto de la clase socio), un enlace al nodo siguiente, y otro al nodo anterior. Al constructor se le pasa el socio, y por defecto se enlaza como nodo siguiente y nodo anterior “null”.

La clase socio contiene los atributos: nombre, primer apellido, segundo apellido y fecha de entrada. Se ha creado un constructor al que debemos pasarle todos los atributos.

## **IMPLEMENTACIÓN DEL CLUB:**

El club tiene varios métodos para realizar las operaciones necesarias:

* **Fundación de un club sin socios:** De manera automática, con el constructor único de la clase se genera un objeto de tamaño cero, es decir, sin socios.
* **Interrogación sobre la pertenencia o no de un socio a un club**: Con dos punteros, va recorriendo desde el primer nodo hasta el final, y del último hasta el primero; e intenta buscar una equivalencia de nombre, para saber si pertenece o no un socio a dicho club.



* **Incorporación de un socio**: Recibe como parámetro un objeto de la clase socio, y se genera un nuevo nodo con él. Después, con un puntero se comienza a analizar desde la última posición de los nodos del club, para ver si alfabéticamente se va a colocar a la derecha o no del puntero; así, hasta que quede ordenado. De manera que con cada incorporación siempre queda ordenado por orden alfabético.



* **Borrado de un socio**: Primero se comprueba con el método de pertenencia si el socio existe o no, en caso de que si, recorre con un puntero la lista para encontrar su posición y finalmente gestiona su borrado.



* **Interrogación de si un club está vacío o no**: Comprueba y devuelve verdadero si el tamaño de la lista del club está vacía.



* **Unión de dos clubes**: Comprueba que no estén vacíos, y a continuación, va leyendo el contenido de los nodos de ambos clubes y añadiéndolo al nuevo club.
* **Número de socios de un club:**  devuelve con un getter el valor del atributo size.

## **desarrollo DEL MENÚ:**

El programa comienza leyendo el fichero de datos “clubs.dat” en la carpeta FILES, en el que, en el caso de existir, estarán registrados los clubs guardados de anteriores funcionamientos del programa. Con el método de *readObject* de la clase “ObjectInputStream” los lee y los que existen los registra en un ArrayList que utilizará el programa.

A continuación, se debe elegir una de las siguientes funciones del menú al comenzar el programa:

1. **Creación:** Para introducir el nombre de un nuevo club vacío.
2. **Gestión:** Primero comprueba si existen clubs, y en caso de que si, muestra los que hay guardados en el ArrayList de clubs y llama al método de *gestionClub*, que nos permitirá hacer las siguientes operaciones:
   1. Dar de alta a un nuevo socio en el club: Llama al método *altaUsuario*, que permitirá introducir los nuevos datos del socio e incorporarlo al club con el método perteneciente a la clase club llamado *incorporación.*
   2. Dar de baja a un socio del club: Llama al método *bajaUsuario*, que exige introducir el nombre del socio a borrar, y con ello, llamar al método borrado de la clase club.
   3. Visualizar los socios: Llama al método, recorre la lista de los nodos del club, y los muestra.
   4. Crear fichero de texto y guardar el club: Llama al método escribir fichero, que realiza la operación de lectura y posteriormente escribe el fichero “clubs.dat” con los que ya había anteriormente en él y actualiza el club seleccionado si ya existía. Todo ello escrito como objetos con el método *writeObject* de la clase “ObjectOutputStream”. Para la utilización de estos métodos de lectura y escritura es necesario que las clases Nodo y Socio implementen la interfaz “Serializable”.
3. **Unión de dos clubs:** Exige introducir el nombre del club en el que van a ser añadidos y muestra el ArrayList de clubs con los que trabaja el programa actualmente para elegirlos y llama al método de unión de la clase club. De forma que se añaden al club recién creado los socios de los dos clubs elegidos, pero no se eliminan esos clubs.
4. **Finaliza el programa**, de modo que sólo se habrán escrito en el fichero los clubs que hayan efectuado el proceso del **apartado 2.d**; por lo que cuando vuelva a ejecutarse el programa, se recuperarán los club y los nodos enlazados de cada uno de ellos que contienen los socios.

Diagrama de flujo resumido:

