

# Sesión 09: Indices y *nulles*

## Hoja de problemas

### Programación 2

Ángel Herranz

aherranz@fi.upm.es

Universidad Politécnica de Madrid

2022-2023

 **Ejercicio 1.** En clase nos pedían un programa que...

“[...] lea de la entrada estándar órdenes para insertar y borrar canciones y que imprima la *playlist* resultante final”

Para probar su funcionamiento construíamos un fichero `ordenes.txt` con las siguientes órdenes de añadir y borrar canciones:

```
a
Wish you where here
Pink Floyd
4
a
Despacito
Luis Fonsi
2
a
The logical song
Supertramp
5
r
Despacito
a
SAD!
XXXTENTACION
3
a
God's Plan
Drake
4
r
SAD!
```

```
a
Havana
Camila Cabello, Young Thug
4
```

y usamos dicho fichero como entrada estándar para nuestro programa, así:

```
C:\ Sesion09> java ProcesarOrdenes < ordenes.txt
```

y esperamos que la salida estándar sea algo como esto:

```
C:\ Sesion09> java ProcesarOrdenes < ordenes.txt
God's Plan:Drake:4
Wish you where here:Pink Floyd:4
The logical song:Supertramp:5
Havana:Camila Cabello, Young Thug:4
```

La implementación vista en clase tiene las siguientes características:

- Se usa **null** para saber si hay un hueco para añadir una canción en el array de canciones.
- Cuando se borra una canción que estaba en la posición  $i$  del array, dicha posición se pone a **null**.
- No nos importa que queden huecos entre canciones en el array.
- No nos importa perder el orden en el que se insertaron las canciones.

Si no pudiste terminar el programa en clase, este es el momento.

☐ **Ejercicio 2.** En este ejercicio vamos a modificar el programa para resolver algunos de los problemas anteriores:

- Vamos a mantener las canciones en *la parte de arriba* del array sin huecos entre las mismas. Es decir, si en la *playlist* hay  $n$  canciones entonces todas las canciones están referenciadas desde la posición 0 hasta la  $n - 1$  en el array y el resto de posiciones, desde  $n$  hasta `playlist.length-1`, son **null**.
- Para ello, el mayor trabajo se realiza a la hora de borrar: si queremos borrar la canción que está en la posición  $i$ , tendremos que desplazar todas las canciones siguientes, desde la posición  $i + 1$  hasta la posición `playlist.length-1`, una posición atrás.
- Además, para no tener que buscar el primer **null** a la hora de añadir, vamos a mantener un contador junto al array. Dicho contador, además de decirnos cuantas canciones hay en la *playlist*, nos dice en qué posición hay que añadir la siguiente canción.

Tienes que modificar el programa implementado en clase de acuerdo a dichas reglas.

☐ **Ejercicio 3.** ¿Te imaginas que *alguien* nos regala una clase `Playlist`? Una clase donde sus instancias (objetos) representan *playlists* y su API<sup>1</sup> nos permite añadir y borrar canciones. ¿Te imaginas que nuestro programa principal quedase así de limpio?

---

<sup>1</sup>Recuerda que API significa *Application Public Interface* y se refiere a cómo se usa un código cualquiera, en nuestro caso cómo se usan los objetos de la clase `Playlist`, en otras palabras, sus métodos públicos

```

1  class ProcesarOrdenes {
2      public static void main(String[] args) {
3          // Se crea un objeto Scanner para poder
4          // leer de la entrada estándar
5          java.util.Scanner stdin =
6              new java.util.Scanner(System.in);
7
8          // Se declara un array de canciones
9          Playlist playlist = new Playlist();
10
11         // Variables auxiliares para
12         int i = 0;
13         String t;
14         String a;
15         int v;
16
17         // Leer órdenes hasta que no haya más datos
18         // en la entrada estándar
19         while (stdin.hasNext()) {
20             String o = stdin.nextLine();
21             switch (o) {
22                 case "a":
23                     t = stdin.nextLine();
24                     a = stdin.nextLine();
25                     v = stdin.nextInt();
26                     stdin.nextLine();
27                     playlist.add(t, a, v);
28                     break;
29                 case "r":
30                     String titulo = stdin.nextLine();
31                     playlist.remove(titulo);
32                     break;
33             }
34         }
35         // Imprimir la playlist
36         playlist.imprimir();
37     }
38 }

```

Tu tarea consiste en implementar la clase `Playlist` con un API como el que te indican las líneas 9, 27, 31 y 36 (tienes un poco de ayuda con un esqueleto de código en la siguiente página, pero inténtalo por ti mismo primero).

```

Tu clase tendrá un aspecto parecido a esto:
class Playlist {
    // Declara aquí tus atributos: el array de canciones y el contador
    // de canciones
    ...
}
/**
 * Construye una playlist vacía.
 */
public Playlist() {
    // Implementa aquí la inicialización de los atributos
    ...
}
/**
 * Añade una canción a la playlist.
 */
public void add(Cancion c) {
    // Implementa aquí cómo se añade una canción a la playlist
    ...
}
/**
 * Borra una canción que coincida con el título indicado.
 */
public void remove(String título) {
    // Implementa aquí cómo se borra una canción de la playlist
    ...
}
/**
 * Imprime la playlist
 */
public void imprimir() {
    // Implementa aquí cómo se imprime la playlist en la salida estándar
    ...
}
}

```