







## Desafío 17


### Desafío 17


 **Evaluador de Entregas de Programación**


 **Estudiante:** MILAGROS TERESITA POZZO FASINI

 **Centro:** CeRP del Suroeste


 **Curso:** PROGRAMACIÓN II: 02INPGR22025215525 Anual Semipresencial

 **Año:** 2025

 **Desafío 17: Lista de nombres**  
Implementar un programa que use un `ArrayList<String>` para almacenar nombres ingresados manualmente y luego mostrarlos en pantalla.

 **Tu código de solución:**

Escribe tu código aquí...

 **Fundamentación de tu solución:**

Explica tu estrategia, qué pensaste para resolver el problema, etc.

Para resolver este desafío lo primero que pensé fue en cómo manejar una lista de nombres cuyo tamaño no conozco desde el inicio. Si hubiera usado un arreglo tradicional (`String[]`), tendría que haber definido un tamaño fijo, lo que no sirve en este caso porque el usuario puede ingresar pocos o muchos nombres. Por esa razón elegí **`ArrayList<String>`**, ya que se trata de una lista dinámica que permite ir agregando elementos sin necesidad de preocuparme por su capacidad.

Luego, como necesitaba una forma de leer lo que escriba el usuario en la consola, opté por usar la clase **`Scanner`**, que es la herramienta más simple y clara para capturar datos desde el teclado en programas básicos de Java. Para el ingreso de datos, la estrategia fue utilizar un **bucle `while(true)`**, que repite de manera indefinida la solicitud de nombres. El corte del ciclo lo resolví con la palabra clave **`"fin"`**, de modo que el usuario tiene control sobre cuándo dejar de ingresar información. Para comparar lo que se escribe, usé el método `equalsIgnoreCase`, que permite que no importe si se escribe **`"fin"`**, **`"FIN"`** o con cualquier combinación de mayúsculas y minúsculas.

Cada vez que se ingresa un nombre distinto de **`"fin"`**, se lo guarda dentro del `ArrayList` mediante el método `add()`. De esta manera voy construyendo la lista dinámicamente. Una vez que el usuario decide terminar, el ciclo se corta con el `break` y paso a mostrar el contenido de la lista. Para la salida elegí un **bucle `for-each`**, que es la forma más



sencilla de recorrer una colección en Java, ya que no requiere manejar índices y se entiende bien incluso en un nivel inicial.

En conclusión, la estrategia general fue apoyarme en estructuras y recursos básicos de Java: ArrayList para almacenar, Scanner para leer datos, un ciclo while para repetir entradas y un for-each para mostrar resultados. Así, el programa cumple exactamente con lo que pide el enunciado de forma clara, simple y adecuada para un nivel principiante.

Enlace a GitHub [https://github.com/MilagrosPozzo/Programacion-2/blob/main/U3 Lenguaje de Programacion Java/Unidad 3 Practico4 Introduccion a JAVA/src/Desafio17.java](https://github.com/MilagrosPozzo/Programacion-2/blob/main/U3%20Lenguaje%20de%20Programacion%20Java/Unidad%203%20Practico4%20Introduccion%20a%20JAVA/src/Desafio17.java)