# CASO PRÁCTICO 1

**ESTRUCTURAS DE DATOS**

## Contexto

En el departamento de informática estáis llevando a cabo un proyecto especialmente diseñado para aplicarlo en una empresa de formación y tiene como objetivo almacenar, procesar y gestionar información utilizando diferentes estructuras de datos y siguiendo los pasos que se indican a continuación.

## Cuestiones a resolver

Crea una aplicación sencilla y sin interfaz gráfica solo usando la instrucción *System.out.println*() de Java, que permita gestionar la información de estudiantes utilizando primero un array convencional y luego un ArrayList. Los estudiantes serán almacenados con su nombre, edad y nota media. Hay un menú principal y un submenú, esta sería la estructura:

Explicación de la estructura del menú:

* **Menú Principal:** El usuario primero debe elegir si quiere trabajar con arrays o con ArrayList, y puede regresar al menú principal en cualquier momento si lo necesita.
  + **Submenú Array:** El usuario tiene dos opciones para trabajar con arrays:
    - Agregar un estudiante
    - Ver todos los estudiantes almacenados en el array.
    - Salir
  + **Submenú ArrayList:** El usuario tiene más opciones con ArrayList, como son:
    - Agregar un estudiante
    - Ver todos los estudiantes almacenados en el ArrayList
    - Eliminar un estudiante determinado por nombre
    - Buscar un estudiante determinado por nombre
    - Volver al menú principal

**Ejemplo de pantalla de salida:**

**Usando opción 1 del menú Principal (Usar Array)**

Texto

El contenido generado por IA puede ser incorrecto.

**Usando opción 2 del menú Principal (Usar ArrayList)**

Con Las opciones 1 y 2 el usuario no notará la diferencia, pero internamente se ha utilizado una estructura diferente de almacenamiento (ArrayList).

En las opciones 3 y 4, eliminar y buscar respectivamente se le pedirá al usuario el nombre del alumno a eliminar o a buscar y en pantalla debe aparecer el resultado, bien informando del borrado, de los datos del alumno encontrado o bien el mensaje de no encontrado.

Texto

El contenido generado por IA puede ser incorrecto.

En la siguiente web puedes encontrar información adicional sobre la API de Java <https://docs.oracle.com/en/java/javase/17/docs/api/> , para comprender mejor el uso de métodos de arrays y colecciones. Para la elaboración del programa se utilizará Eclipse como IDE de desarrollo.

## Recursos

Se deberá consultar el contenido de la unidad, internet, libros, revistas y utilizar medios informáticos para la presentación del caso práctico (Word, Power-Point…)

## Objetivos

Utilizar arrays para almacenar y manipular datos en programas.

Identificar y utilizar las librerías de clases relacionadas con tipos de datos avanzados en Java.

Implementar listas para organizar y manejar datos de forma eficiente en el código.

Emplear iteradores para recorrer y manipular los elementos de las listas de manera ordenada y controlada.

Comprender las distintas colecciones de datos disponibles en Java y su utilidad según el tipo de operación que se requiera realizar.

Desarrollar clases y métodos en Java que puedan trabajar con distintos tipos de datos, garantizando la reutilización del código.

Implementar operaciones agregadas (como filtros, mapeos y reducciones) sobre colecciones de datos para procesar la información de manera eficiente.

## Resultados de aprendizaje y criterios de evaluación

Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

* Se han escrito programas que utilicen matrices (arrays).
* Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.
* Se han utilizado listas para almacenar y procesar información.
* Se han utilizado iteradores para recorrer los elementos de las listas.
* Se han reconocido las características y ventajas de cada una de las colecciones de datos disponibles.
* Se han creado clases y métodos genéricos.
* Se han utilizado operaciones agregadas para el manejo de información almacenada en colecciones.

## Rúbrica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Excelente** | **Satisfactorio** | **Mejorable** | **Insuficiente** |
| **Implementación de estructuras de datos y métodos.** | Implementa correctamente arrays y ArrayLists para almacenar, procesar y gestionar la información. Todos los métodos requeridos están presentes y funcionan correctamente. | Implementa correctamente arrays y ArrayLists, pero uno o más métodos no funcionan correctamente. | Implementa arrays y/o ArrayLists, pero no están bien implementados o presentan fallos significativos. | No implementa correctamente arrays ni ArrayLists. |
| 20% | 2 | 1.5 | 1 | 0.5 |
| **Utilización de clases y métodos.** | Diseña e implementa eficientemente clases y métodos. Utiliza adecuadamente constructores, propiedades y control de visibilidad. | Diseña e implementa clases y métodos, pero hay problemas menores con los constructores, propiedades o control de visibilidad. | Implementa clases y métodos, pero presenta problemas significativos con los constructores, propiedades o control de visibilidad. | No implementa correctamente las clases ni los métodos. |
| 20% | 2 | 1.5 | 1 | 0.5 |
| **Manipulación de información** | Escribe programas que manipulan correctamente la información utilizando arrays y ArrayLists. Demuestra una comprensión clara de la gestión y manipulación de datos. | Escribe programas que manipulan la información, pero hay problemas menores con el uso de arrays y ArrayLists. | Escribe programas que manipulan la información, pero presenta problemas significativos con el uso de arrays y ArrayLists. | No manipula correctamente la información ni usa adecuadamente arrays y ArrayLists. |
| 20% | 2 | 1.5 | 1 | 0.5 |
| **Identificación y resolución de problemas** | Define claramente el problema e implementa una solución efectiva y adecuada utilizando arrays y ArrayLists. | Define el problema, pero la solución tiene problemas menores. | Define el problema, pero la solución tiene deficiencias significativas. | No define correctamente el problema o no implementa una solución adecuada. |
| 20% | 2 | 1.5 | 1 | 0.5 |
| **Código limpio con estructuras necesarias.** | Presenta código limpio con estructuras necesarias y sin redundancia. | Presenta código limpio, pero hay algunas estructuras que se pueden simplificar, eliminar o tabular para evitar redundancia. | Presenta código más o menos limpio, pero hay estructuras que se pueden simplificar, eliminar o tabular para evitar redundancia. | Presenta código desordenado, hay muchas estructuras que se deben simplificar o eliminar para evitar redundancia. |
| 10% | 1 | 0.75 | 0.5 | 0.25 |
| **Errores de compilación y/o ejecución.** | El programa se ejecuta sin errores de compilación o ejecución, realizando la función esperada. | El programa se ejecuta sin errores de compilación, pero sí con algunos errores en la ejecución, la función realizada varía ligeramente de la función esperada. | El programa se ejecuta sin errores de compilación, pero sí con varios errores en la ejecución, la función realizada varía de la función esperada. | El programa se ejecuta con errores de compilación por tanto no es ejecutable y/o el programa ejecuta sin errores de compilación, pero con numerosos errores en la ejecución, la función realizada no coincide con la función esperada. |
| 10% | 1 | 0.75 | 0.5 | 0.25 |