

# Solucions a exercicis / GestorCSVcomplet

## Exercici 1 — Afegir alumnes

Modifica el programa perquè, a més de nom i nota, també guardi:

- el **cognom**,
- i l'**edat** de l'alumne.

· Cada línia del CSV ha de quedar així:

Nom,Cognom,Edat,Nota

## Exercici 2 — Llistar alumnes amb format

Quan es mostrin els alumnes, fes que surti en un format més bonic

Nom complet: Anna Garcia | Edat: 20 | Nota: 8.5

## Exercici 3 — Calcular la mitjana

Afegeix una nova opció al menú:

- Calcular la mitjana

Ha de llegir totes les notes del fitxer i calcular la mitjana.



## Exercici 4 — Cercar alumne

Afegeix una opció al menú:

- Cercar alumne pel nom
- Demana el nom per teclat.
- Recorre el fitxer i mostra la informació de l'alumne si existeix.
- Si no el troba, mostra: *"No s'ha trobat cap alumne amb aquest nom."*

### Exercici 5 — Aprobat o suspès

Quan llistis alumnes, afegeix al final un missatge segons la nota:

- Si nota  $\geq 5 \rightarrow$  “ Aprobat”
- Si nota  $< 5 \rightarrow$  “ Suspès”

### Exercici 6 (avançat) — Eliminar un alumne

Crea una opció al menú per **eliminar un alumne** del fitxer:

1. Llegeix tots els alumnes i guarda'ls en una llista temporal (memòria).
2. Treu de la llista l'alumne amb el nom indicat.
3. Torna a escriure tot el fitxer des de zero amb la nova llista.

### Bonus — Ordenar per nota

Fes que quan es llistin els alumnes, surtin **ordenats de major a menor nota**.

al final ens queda un gestor d'alumnes força complet, amb dades persistides a `alumnes.csv`.

Aquí tens el codi complet per comparar-lo amb el teu:

```
package Paquet1;

import java.io.*;
import java.util.*;

public class GestorCSVCompleto {

    private static final File FITXER = new File("alumnes.csv");

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
```

```
int opcio;

do {

    System.out.println("\n--- GESTOR D'ALUMNES (CSV) ---");

    System.out.println("1. Afegir alumne");
    System.out.println("2. Llistar alumnes");
    System.out.println("3. Calcular nota mitjana");
    System.out.println("4. Cercar alumne pel nom");
    System.out.println("5. Eliminar alumne");
    System.out.println("0. Sortir");

    System.out.print("Opció: ");

    opcio = llegirEnter(sc);

    switch (opcio) {

        case 1 -> afegirAlumne(sc);
        case 2 -> llistarAlumnes();
        case 3 -> calcularMitjana();
        case 4 -> cercarAlumne(sc);
        case 5 -> eliminarAlumne(sc);
        case 0 -> System.out.println("Sortint...");
        default -> System.out.println("Opció no vàlida");

    }

} while (opcio != 0);

sc.close();

}

private static void afegirAlumne(Scanner sc) {

    System.out.print("Nom: ");

    String nom = sc.nextLine();

    System.out.print("Cognom: ");

    String cognom = sc.nextLine();

    System.out.print("Edat: ");

    int edat = llegirEnter(sc);

    System.out.print("Nota: ");
```

```
double nota = llegirDouble(sc);

try (PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter(FITXER, true)))) {

    pw.println(nom + "," + cognom + "," + edat + "," + nota);

    System.out.println("Alumne afegit correctament!");

} catch (IOException e) {

    System.err.println("Error escrivint: " + e.getMessage());

}

}

private static void llistarAlumnes() {

    List<String[]> alumnes = llegirFitxer();

    if (alumnes.isEmpty()) {

        System.out.println("Encara no hi ha alumnes.");

        return;

    }

    // Ordenem per nota descendent

    alumnes.sort((a, b) -> Double.compare(Double.parseDouble(b[3]),
Double.parseDouble(a[3])));

    System.out.println("\n--- LLISTA D'ALUMNES ---");

    for (String[] alumne : alumnes) {

        String nomCompleto = alumne[0] + " " + alumne[1];

        int edat = Integer.parseInt(alumne[2]);

        double nota = Double.parseDouble(alumne[3]);

        String estat = (nota >= 5) ? "✅ Aprovat" : "❌ Suspès";

        System.out.println("Nom complet: " + nomCompleto +

            " | Edat: " + edat +

            " | Nota: " + nota +

            " | " + estat);

    }

}

private static void calcularMitjana() {
```

```
List<String[]> alumnes = llegirFitxer();

if (alumnes.isEmpty()) {
    System.out.println("Encara no hi ha alumnes.");
    return;
}

double suma = 0;

for (String[] alumne : alumnes) {
    suma += Double.parseDouble(alumne[3]);
}

double mitjana = suma / alumnes.size();

System.out.println("La mitjana de notes és: " + mitjana);
}

private static void cercarAlumne(Scanner sc) {
    System.out.print("Nom a cercar: ");
    String nomBuscat = sc.nextLine().trim();

    List<String[]> alumnes = llegirFitxer();
    boolean trobat = false;

    for (String[] alumne : alumnes) {
        if (alumne[0].equalsIgnoreCase(nomBuscat)) {
            System.out.println("Trobat -> " + alumne[0] + " " + alumne[1]
+
                                " | Edat: " + alumne[2] +
                                " | Nota: " + alumne[3]);

            trobat = true;
        }
    }

    if (!trobat) {
        System.out.println("No s'ha trobat cap alumne amb aquest nom.");
    }
}

private static void eliminarAlumne(Scanner sc) {
```

```
System.out.print("Nom de l'alumne a eliminar: ");

String nomEliminar = sc.nextLine().trim();

List<String[]> alumnes = llegirFitxer();

boolean eliminat = alumnes.removeIf(a ->
a[0].equalsIgnoreCase(nomEliminar));

if (eliminat) {

    // Tornem a escriure el fitxer des de zero

    try (PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter(FITXER, false)))) {

        for (String[] alumne : alumnes) {

            pw.println(String.join(",", alumne));

        }

    } catch (IOException e) {

        System.err.println("Error reescrivint: " + e.getMessage());

    }

    System.out.println("Alumne eliminat correctament!");

} else {

    System.out.println("No s'ha trobat cap alumne amb aquest nom.");

}

}

// --- Utilitats ---

private static List<String[]> llegirFitxer() {

    List<String[]> alumnes = new ArrayList<>();

    if (!FITXER.exists()) return alumnes;

    try (BufferedReader br = new BufferedReader(new FileReader(FITXER))) {

        String linia;

        while ((linia = br.readLine()) != null) {

            String[] parts = linia.split(",");

            if (parts.length == 4) alumnes.add(parts);

        }

    } catch (IOException e) {
```

```
        System.err.println("Error llegint: " + e.getMessage());
    }

    return alumnes;
}

private static int llegirEnter(Scanner sc) {
    while (true) {
        try {
            return Integer.parseInt(sc.nextLine().trim());
        } catch (NumberFormatException e) {
            System.out.print("Introdueix un nombre enter vàlid: ");
        }
    }
}

private static double llegirDouble(Scanner sc) {
    while (true) {
        try {
            return Double.parseDouble(sc.nextLine().trim());
        } catch (NumberFormatException e) {
            System.out.print("Introdueix un nombre decimal vàlid: ");
        }
    }
}
}
```