DOC4: Programació estructurada avançada - ArrayList i Col·leccions.

1. ArrayList

Un **ArrayList** és una estructura de dades que emmagatzema elements de manera similar a un array, però amb més flexibilitat i de forma dinàmica.

Diferències principals entre Array i ArrayList:

Array	ArrayList
La mida és fixa en el moment de crear-lo.	La mida pot augmentar o disminuir automàticament.
Pot contenir tipus primitius (int, double...).	Només pot contenir objectes (Integer, Double, String...).
Sintaxi més simple per accedir als elements.	Té mètodes que faciliten afegir, eliminar i buscar elements.

Quan fer servir un ArrayList?

- Quan no sabem quants elements tindrà la llista.
- Quan volem afegir o eliminar elements fàcilment.
- Quan volem que la llista creixi automàticament.

2. Creació d'un ArrayList.

Per utilitzar ArrayList hem d'importar la classe:

```
import java.util.ArrayList;
```

Per a crear un ArrayList es similar a com ho fem amb arrays però més orientat a objectes.

ArrayList<String> llista = new ArrayList<String>();

1. ArrayList d'enters (Integer):

ArrayList<Integer> numeros = new ArrayList<Integer>();

2. ArrayList de decimals (Double):

ArrayList<Double> decimals = new ArrayList<Double>();

3. ArrayList de cadenes de text (String):

ArrayList<String> noms = new ArrayList<String>();

4. ArrayList de caràcters (Character):

ArrayList<Character> lletres = new ArrayList<Character>();

5. ArrayList de booleans (Boolean):

ArrayList<Boolean> valors = new ArrayList<Boolean>();

3. Mètodes més comuns:

Mètode	Descripció
add(element)	Afegeix un element al final.
add(posicio, element)	Afegeix un element en una posició concreta. **Mou tots els elements cap a la dreta a partir d'aquella posició**
get(posicio)	Retorna l'element en la posició indicada.
set(posicio, element)	Substitueix l'element en la posició indicada.
remove(posicio)	Elimina l'element en la posició indicada
size()	Retorna el nombre d'elements
clear()	Buida tots els elements de l'ArrayList.
contains(element)	Retorna true si l'element està a la llista, false si no.
indexOf(element)	Retorno la posició de l'element dins de l'ArrayList.

EXEMPLE per a buscar elements en un array i eliminar-lo sense bucles.

```
while(arraylist.contains(element)) {
                    arraylist.remove(arraylist.indexOf(element));
            }
```

4. Fer una copia de Arraylist.

```
ArrayList<Integer> numerosCopia = new ArrayList<>(numeros);
```

5. Recorrer un ArrayList:

Un arraylist es recorre de la mateixa manera que un Array.

```
for (int i = 0; i < numeros.size(); i++) {
   System.out.println("Posició " + i + ": " + numeros.get(i));
}
```

6. Mostrar un ArrayList directament sense personalització

```
ArrayList<String> noms = new ArrayList<String>;
System.out.println(noms);
```

** Això ja mostra tot l'arrayList però de forma primitiva, sense personalització

6. FOR-EACH: -> Agafar cada element sense índex

Tot i això hi ha una forma més avançada de mostrar els elements d'un ArrayList que es amb el **FOR-EACH.**

```
ArrayList<Integer> numeros = new ArrayList<Integer>();
numeros.add(10);
numeros.add(20);
```

```
numeros.add(30);

for (Integer numero : numeros) {

    System.out.println(numero);

}
```

Aquest `for-each` és més curt, mostra l'element, **però no dóna accés a la posició de l'element, només als valors!!!!!!!!!!!!!.**

7. Apunt Important:

- Encara que als arrays fèiem `int[] numeros = new int[5];`, aquí no podem fer `ArrayList<int>`.
- Els tipus **primitives** com `int`, `double`, `char`... no es poden posar directament en un ArrayList.
- Hem d'usar les versions amb majúscula, que són **classes especials** que representen aquests valors:
  - `int` → `Integer`
  - `double` → `Double`
  - `char` → `Character`
  - `boolean` → `Boolean`

8. Exemple pràctic.

```
import java.util.ArrayList;

public class ExempleArrayList {

    public static void main(String[] args) {

        ArrayList<String> noms = new ArrayList<String>();

        // Afegim elements

        noms.add("Anna");

        noms.add("Marc");

        noms.add("Joan");

        // Accedim a elements

        System.out.println("Nom a la posició 1: " + noms.get(1));
```

```
        // Modifiquem un element
        noms.set(1, "Maria");
        // Eliminem un element
        noms.remove(2);
        // Recorrem el ArrayList
        System.out.println("Llista de noms:");
        for (String nom : noms) {
            System.out.println(nom);
        }
        // Mida de la llista
        System.out.println("Mida de la llista: " + noms.size());
    }
}
```

2. COL·LECCIONS

Les **Collections** a Java són una classe que conté **mètodes estàtics** per facilitar el treball amb llistes com els **ArrayList**.

Els mètodes de la classe Collections permeten fer operacions habituals com:

- Ordenar elements.
- Invertir l'ordre.
- Barrejar aleatòriament.
- Trobar el valor màxim o mínim.
- Comptar quantes vegades apareix un element.

1. Mètodes principals de la classe Collections per treballar amb ArrayList

Mètode	Descripció
Collections.sort(llista)	Ordena la llista en ordre creixent.

Collections.reverse(llista)	Inverteix l'ordre dels elements.
Collections.shuffle(llista)	Barreja els elements aleatòriament.
Collections.max(llista)	Retorna el valor més gran.
Collections.min(llista)	Retorna el valor més petit.
Collections.frequency(llista, x)	Retorna quantes vegades apareix l'element x.
Collections.swap(llista, i, j)	Intercanvia els elements de les posicions i i j.

2. Exemple ordenadar Arraylist:

```
// Copiem
ArrayList<String> nomsOrdenats = new ArrayList<>(noms);
// Ordenem la còpia Collections.sort(numerosOrdenats);
Collections.sort(nomsOrdenats);
// Mostrem fe forma simple
System.out.println(nomsOrdenats);
```