Java Collections

🧩 O que são Collections?

As Collections são estruturas de dados prontas do Java. Servem pra guardar, buscar e organizar informações sem dor de cabeça. Em vez de usar arrays fixos, as collections crescem, diminuem e têm funções úteis.

Três tipos principais que você vai ver o tempo todo:

Tipo	Pra que serve	Permite repetição?	Mantém ordem?
ArrayList	Lista com ordem e repetição	☑	☑
HashSet	Conjunto sem repetição	❌	❌
HashMap	Pares chave → valor	Chave ❌ / Valor ☑	❌

📋 ArrayList – A lista que cresce sozinha

Quando usar

Quando você precisa guardar elementos em ordem e aceita repetição. É como um array que cresce sozinho.

Exemplo:

```
import java.util.ArrayList;

public class ExemploArrayList {
    public static void main(String[] args) {
        ArrayList<String> alunos = new ArrayList<>();

        alunos.add("Ana");
        alunos.add("Bruno");
        alunos.add("Ana"); // pode repetir
        alunos.add("Carlos");

        System.out.println(alunos); // [Ana, Bruno, Ana, Carlos]

        System.out.println(alunos.get(1)); // Bruno
        alunos.remove("Ana");
        System.out.println(alunos); // [Bruno, Ana, Carlos]
    }
}
```

Dicas rápidas

- add() → adiciona
- get(i) → pega o elemento da posição

- remove() → remove valor ou índice
- size() → mostra quantos tem
- Aceita elementos repetidos

💬 Resumo: o ArrayList é sua lista de tarefas — organizada, mas aceita repetidos.

🧱 HashSet – O conjunto exclusivo

Quando usar

Quando você não quer elementos repetidos. O HashSet não se importa com a ordem, ele só quer unicidade.

Exemplo:

```
import java.util.HashSet;

public class ExemploHashSet {
    public static void main(String[] args) {
        HashSet<String> cpfs = new HashSet<>();

        cpfs.add("111");
        cpfs.add("222");
        cpfs.add("111"); // ignorado, já existe

        System.out.println(cpfs); // [111, 222]
        System.out.println(cpfs.contains("222")); // true
    }
}
```

Dicas rápidas

- add() → adiciona (mas ignora se já existir)
- contains() → verifica se está lá
- remove() → tira o valor
- Não tem índice nem ordem

💬 Resumo: o HashSet é como um grupo exclusivo — cada elemento só entra uma vez.

🗺️ HashMap – O dicionário chave→valor

Quando usar

Quando você quer ligar uma coisa à outra: uma chave identifica um valor (exemplo: matrícula → nome do aluno).

Exemplo:

```java
import java.util.HashMap;

public class ExemploHashMap {
    public static void main(String[] args) {
        HashMap<Integer, String> alunos = new HashMap<>();

        alunos.put(1001, "Ana");
        alunos.put(1002, "Bruno");
        alunos.put(1003, "Carlos");

        System.out.println(alunos); // {1001=Ana, 1002=Bruno, 1003=Carlos}
        System.out.println(alunos.get(1002)); // Bruno

        alunos.remove(1003);
        System.out.println(alunos.containsKey(1003)); // false
    }
}
```

Dicas rápidas

- put(chave, valor) → adiciona ou substitui
- get(chave) → pega o valor
- remove(chave) → apaga o par
- keySet() → todas as chaves
- values() → todos os valores
- entrySet() → pares completos

💬 Resumo: o HashMap é como uma agenda — você procura pela chave e acha o valor.

⚔️ Comparando os três

Coleção	Aceita repetidos?	Mantém ordem?	Acesso por índice?	Acesso por chave?
ArrayList	☑ Sim	☑ Sim	☑ Sim	❌ Não
HashSet	❌ Não	❌ Não	❌ Não	❌ Não
HashMap	Chave ❌ / Valor ☑	❌ Não	❌ Não	☑ Sim

🧠 Dica do professor

"Escolher a coleção certa é metade do código feito."

- Quer guardar em ordem e repetir → ArrayList
- Quer garantir que não se repita → HashSet
- Quer ligar uma coisa a outra → HashMap