Guia Rápido de Comparações em Java

1. Comparable (compareTo)

Use quando quiser definir a ordem natural de objetos da própria classe.
Exemplo:

```
class Produto implements Comparable<Produto> {
    public int compareTo(Produto outro) {
        return Double.compare(this.preco, outro.preco);
    }
}
```

2. Comparator (compare)

Use quando quiser ordenar objetos externamente à classe.
Exemplo:

```
class ComparadorPorNome implements Comparator<Aluno> {
    public int compare(Aluno o1, Aluno o2) {
        return o1.getNome().compareTo(o2.getNome());
    }
}
```

3. Comparator.comparing() e thenComparing()

Use para ordenação funcional, limpa e encadeada.
Exemplo:

```
alunos.sort(
    Comparator.comparing(Aluno::getNota)
            .thenComparing(Aluno::getNome)
            .thenComparingInt(Aluno::getIdade)
);
```

4. Comparator.comparingInt / comparingDouble

Versões otimizadas para tipos primitivos.
Exemplo:

```
Comparator.comparingInt(Aluno::getIdade);
```

5. compareToIgnoreCase (Strings)

Ignora maiúsculas/minúsculas na comparação de Strings.
Exemplo:

```
nome1.compareToIgnoreCase(nome2);
```

6. Collator (ordenar ignorando acentos)

Permite ordenação linguística (estilo humano).
Exemplo:

```
Collator collator = Collator.getInstance(new Locale("pt", "BR"));
```

Guia Rápido de Comparações em Java

```
collator.setStrength(Collator.PRIMARY);
alunos.sort((a, b) -> collator.compare(a.getNome(), b.getNome()));
```

Resumo Final

- compareTo(): ordem natural dentro da própria classe.
- compare(): ordem externa, com Comparator.
- comparing() / thenComparing(): moderno, funcional e limpo.
- Collator: ignora acentos e regras linguísticas.

Use compareTo para 1 único critério natural. Use Comparator para tudo mais. ;)