Guia Java — Parte 3: Coleções, Exceções e Boas Práticas

Nesta terceira parte do guia de Java, você aprenderá a trabalhar com as principais coleções da linguagem, entenderá como tratar erros com exceções e conhecerá boas práticas para escrever código limpo e profissional.

Coleções (Collections Framework): O Java Collections Framework é um conjunto de classes e interfaces que facilitam o armazenamento e manipulação de grupos de objetos. As principais interfaces são List, Set e Map.

List: É uma coleção ordenada que permite elementos duplicados. As implementações mais usadas são ArrayList e LinkedList. Exemplo: List nomes = new ArrayList<>(); nomes.add("Pedro"); nomes.add("Maria"); System.out.println(nomes.get(0));

Set: Não permite elementos duplicados e não garante ordem. As implementações mais comuns são HashSet e TreeSet. Exemplo: Set frutas = new HashSet<>(); frutas.add("Maçã"); frutas.add("Banana");

Map: Armazena pares de chave e valor. As chaves são únicas. As implementações mais usadas são HashMap e TreeMap. Exemplo: Map usuarios = new HashMap<>(); usuarios.put(1, "Pedro"); usuarios.put(2, "Ana"); System.out.println(usuarios.get(1));

Iteração de Coleções: Você pode percorrer coleções com o laço 'for-each' ou com streams. for(String nome : nomes) { System.out.println(nome); }

Tratamento de Exceções (try / catch / finally): Exceções são erros que ocorrem durante a execução do programa. Usar try/catch evita que o programa pare de funcionar ao ocorrer um erro. Exemplo: try { int resultado = 10 / 0; } catch (ArithmeticException e) { System.out.println("Erro: divisão por zero!"); } finally { System.out.println("Bloco finally sempre é executado."); }

Exceções Personalizadas: Você pode criar suas próprias exceções para tratar erros específicos. Exemplo: public class SaldoInsuficienteException extends Exception { public SaldoInsuficienteException(String mensagem) { super(mensagem); } }

Boas Práticas de Código: ✔ Nomeie variáveis e métodos de forma clara e significativa. ✔ Evite métodos muito longos — divida em funções menores. ✔ Use comentários apenas quando necessário, preferindo código autoexplicativo. ✔ Trate exceções de forma adequada, sem esconder erros. ✔ Utilize convenções de nomenclatura padrão (camelCase, PascalCase, etc). ✔ Faça uso de constantes (final) para valores fixos. ✔ Sempre feche conexões, arquivos e streams após o uso.

Boas Práticas com Coleções: ✔ Prefira interfaces (List, Set, Map) em vez de implementações concretas. ✔ Inicialize coleções apenas quando necessário. ✔ Use generics (<>). Exemplo: List nomes = new ArrayList<>(); ✔ Evite modificar coleções enquanto as percorre.

Esses conceitos são essenciais para trabalhar de forma segura e eficiente em Java. Com domínio de coleções, tratamento de erros e boas práticas, você estará preparado para criar sistemas robustos e profissionais.