Programação Avançada - AULA 07

Matheus Moresco
Engenharia de Software - 5º Período
2025/01

Introdução

- Entender o que são Conjuntos (Set) e Mapas (Map).
- Aprender as principais implementações e suas diferenças.
- Implementar exemplos práticos para fixação.

O que são Conjuntos e Mapas?

- **Set (Conjunto)** → Coleção sem elementos duplicados.
- **Map (Mapa)** → Estrutura que armazena pares chave-valor.

Exemplo prático:

- **Set**: Lista de nomes únicos de alunos.
- **Map**: Relaciona número de matrícula → nome do aluno.

Conjuntos (Set)

Um conjunto (Set) é uma coleção sem elementos duplicados. Ele é útil quando precisamos garantir que todos os itens armazenados sejam únicos.

Características do Set:

- Não permite duplicatas
- Não mantém a ordem dos elementos (dependendo da implementação)
- Rápido para verificar se um elemento está presente

Conjuntos (Set)

Principais Implementações:

- **HashSet** → Mais rápido, mas sem ordem garantida
- **LinkedHashSet** → Mantém a ordem de inserção
- **TreeSet** → Mantém os elementos ordenados

HashSet - implementação

```java
import java.util.HashSet;
import java.util.Set;

public class ExemploSet {
    public static void main(String[] args) {
        Set<String> nomes = new HashSet<>();

        nomes.add("Ana");
        nomes.add("Bruno");
        nomes.add("Carlos");
        nomes.add("Ana"); // Ignorado (duplicado)

        System.out.println(nomes); // Sem repetições
    }
}
```

Saída esperada:

```
[Ana, Bruno, Carlos]
```

LinkedHashSet - implementação

```java
import java.util.LinkedHashSet;
import java.util.Set;

public class ExemploLinkedHashSet {
    public static void main(String[] args) {
        Set<String> frutas = new LinkedHashSet<>();

        frutas.add("Maçã");
        frutas.add("Banana");
        frutas.add("Uva");
        frutas.add("Maçã"); // Duplicado, será ignorado
        frutas.add("Laranja");

        System.out.println(frutas); // Mantém a ordem de inserção
    }
}
```

Saída esperada:

```
[Maçã, Banana, Uva, Laranja]
```

TreeSet - implementação

```java
import java.util.Set;
import java.util.TreeSet;

public class ExemploTreeSet {
    public static void main(String[] args) {
        Set<Integer> numeros = new TreeSet<>();

        numeros.add(50);
        numeros.add(10);
        numeros.add(30);
        numeros.add(20);
        numeros.add(40);

        System.out.println(numeros); // Ordenação automática
    }
}
```

Saída esperada:

```
[10, 20, 30, 40, 50]
```

HashSet vs LinkedHashSet vs TreeSet

Característica	HashSet	LinkedHashSet	TreeSet
Ordem dos elementos	Não garantida	Mantém ordem de inserção	Ordenado (crescente)
Desempenho (média)	Mais rápido (O(1) para inserir, buscar, remover)	Rápido (O(1) para inserir, buscar, remover)	Mais lento (O(log n) devido à árvore balanceada)
Uso recomendado	Quando a ordem não importa e precisa de alto desempenho	Quando a ordem de inserção deve ser preservada	Quando os elementos precisam estar ordenados

Quando usar cada um?

- Use **HashSet** se precisar da melhor performance e a ordem não importar.
- Use **LinkedHashSet** se precisar manter a ordem de inserção dos elementos.
- Use **TreeSet** se precisar que os elementos estejam sempre ordenados automaticamente.

Mapas (Map)

Um mapa (Map) é uma estrutura que associa chaves a valores. Cada chave é única, mas os valores podem ser repetidos.

Características do Map:

- Armazena pares chave-valor
- As chaves são únicas (não podem ser repetidas)
- É eficiente para buscar valores através da chave

Mapas (Map)

Principais Implementações:

- **HashMap** → Rápido, sem ordem garantida
- **LinkedHashMap** → Mantém a ordem de inserção
- **TreeMap** → Mantém as chaves ordenadas

HashMap - implementação

```java
import java.util.HashMap;
import java.util.Map;

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

        hashMap.put(3, "Carlos");
        hashMap.put(1, "Ana");
        hashMap.put(4, "Diana");
        hashMap.put(2, "Bruno");

        System.out.println(" 🔹 HashMap (Ordem imprevisível): " + hashMap);
    }
}
```

Saída esperada:

```
HashMap (Ordem imprevisível): {1=Ana, 3=Carlos, 2=Bruno, 4=Diana}
```

LinkedHashMap - implementação

```
import java.util.LinkedHashMap;
import java.util.Map;

public class ExemploLinkedHashMap {
    public static void main(String[] args) {
        Map<Integer, String> linkedHashMap = new LinkedHashMap<>();

        linkedHashMap.put(3, "Carlos");
        linkedHashMap.put(1, "Ana");
        linkedHashMap.put(4, "Diana");
        linkedHashMap.put(2, "Bruno");

        System.out.println(" 🔹 LinkedHashMap (Ordem de inserção mantida): " + linkedHashMap);
    }
}
```

Saída esperada:

```
LinkedHashMap (Ordem de inserção mantida): {3=Carlos, 1=Ana, 4=Diana, 2=Bruno}
```

TreeMap - implementação

```
import java.util.Map;
import java.util.TreeMap;

public class ExemploTreeMap {
    public static void main(String[] args) {
        Map<Integer, String> treeMap = new TreeMap<>();

        treeMap.put(3, "Carlos");
        treeMap.put(1, "Ana");
        treeMap.put(4, "Diana");
        treeMap.put(2, "Bruno");

        System.out.println(" 🔹 TreeMap (Ordenado pelas chaves): " + treeMap);
    }
}
```

Saída esperada:

```
Alunos (ordenados automaticamente pelas chaves): {101=Ana, 102=Bruno, 103=Carlos, 104=Diana}
```

HashMap vs LinkedHashMap vs TreeMap

Característica	HashMap	LinkedHashMap	TreeMap
Ordem dos elementos	Não mantém	Ordem de inserção	Ordenado pelas chaves
Implementação interna	Tabela Hash	Tabela Hash + Lista Duplamente Encadeada	Árvore Rubro-Negra (Red-Black Tree)
Complexidade média	O(1)	O(1)	O(log n)
Uso recomendado	Máximo desempenho	Preservar ordem de inserção	Ordenação automática

Quando usar cada um?

- Use HashMap para operações rápidas quando a ordem não importa.
- Use LinkedHashMap se quiser manter a ordem de inserção dos elementos.
- Use TreeMap se precisar que os elementos sejam armazenados de forma ordenada pelas chaves.

Exemplo prático

- Implementar um Set e um Map para cadastro de veículos.
- Comparar com o uso de Listas.

Exercício Pratico

1. Refatore a classe de Empresa, para
    1. Usar um Set na listagem de e-mails
2. Crie um atributo de produtos na empresa e use um map para cadastrar os produtos.
3. Crie os métodos para cadastro, edição, remoção e listagem de produtos.