Universidade Católica de Brasília

Pró-Reitoria Acadêmica
Curso de (Engenharia De Software)
Trabalho de POO

Pró-Reitoria Acadêmica

ARTIGO SOBRE

COLLECTION E MAP

Autor: (Pablo William, Eduardo Antero e Guilherme Vieira)Orientador: (João Pedro Macleure Nunes dos Santos)

Brasília2023

Sumário

Introdução

Java Collections Framework é uma estrutura integrada na linguagem Java que fornece uma arquitetura unificada para representar e manipular coleções de objetos, além de mapear valores-chave. As coleções são usadas com frequência em desenvolvimento de software para armazenar, recuperar e manipular dados. Neste trabalho, exploraremos as classes e interfaces mais importantes relacionadas a coleções e mapeamento em Java, destacando suas características, vantagens e desvantagens, bem como fornecendo exemplos de aplicação.

A manipulação de dados é um aspecto crítico no desenvolvimento de aplicativos em Java, e para esse fim, a linguagem oferece um conjunto rico de estruturas de dados. Duas das mais fundamentais são as collections e os mapas. As collections, como List, Set e Queue, fornecem maneiras de armazenar e organizar grupos de objetos, enquanto os mapas, como HashMap e TreeMap, oferecem uma estrutura chave-valor para associar informações de forma eficiente. Nesta introdução, exploraremos a importância e a versatilidade dessas estruturas de dados em Java, destacando como elas desempenham um papel fundamental na implementação de algoritmos e no gerenciamento de dados em uma variedade de cenários de desenvolvimento de software.

Interfaces em Java Collections Framework

Iterable

A interface Iterable é o ponto de partida na hierarquia de coleções em Java. Ela define umúnico método, iterator(), que permite percorrer os elementos da coleção. Todas as classesde coleções em Java implementam essa interface. Referência:

Collection

A interface Collection estende a interface Iterable e define operações básicas para trabalhar com grupos de objetos. Ela inclui métodos para adicionar, remover e verificar a existência de elementos em uma coleção. Referência: Collection Interface

Set

A interface Set é uma subinterface de Collection que representa uma coleção que não permite elementos duplicados. Implementações comuns incluem HashSet, LinkedHashSet, e TreeSet. Referência: Set Interface

List

A interface List é outra subinterface de Collection que representa uma lista ordenada de elementos, permitindo duplicatas. Implementações notáveis incluem ArrayList, LinkedListe Vector. Referência: List Interface

Queue

A interface Queue estende Collection e representa uma fila, seguindo a ordem FIFO (First-In, First-Out). Ela inclui métodos para inserir, remover e inspecionar elementos na fila. Referência: Queue Interface

Classes Concretas em Java Collections Framework

AbstractSet

A classe AbstractSet fornece uma implementação base para as classes que implementam
a interface Set. Ela facilita a criação de novas classes Set personalizadas. Referência: AbstractSet Class

AbstractList

A classe AbstractList desempenha um papel semelhante ao AbstractSet, mas para classes que implementam a interface List. Referência: AbstractList Class

HashSet

A classe HashSet é uma implementação de Set que armazena elementos em uma tabela de dispersão. Ela oferece acesso rápido e não permite elementos duplicados, mas não garante a ordem dos elementos. Referência: HashSet Class

LinkedList

A classe LinkedList é uma implementação de List que usa uma estrutura de lista duplamente encadeada. Ela oferece inserção e remoção eficientes, mas o acesso aleatório é mais lento do que o ArrayList. Referência: LinkedList Class

Vector

A classe Vector é uma implementação de List que é thread-safe, tornando-a adequada para ambientes multithread. No entanto, sua sincronização pode causar um desempenho mais lento. Referência: Vector Class

ArrayList

A classe ArrayList é uma implementação de List que usa um array dinâmico para armazenar elementos. Ela oferece acesso rápido, mas não é thread-safe. Referência: ArrayList Class

Dictionary

A classe Dictionary é uma classe obsoleta que foi substituída por Map na versão mais recente do Java. Ela mapeia chaves para valores, mas possui limitações de funcionalidade. Referência: Dictionary Class

AbstractMap

A classe AbstractMap fornece uma implementação base para classes que implementam ainterface Map. Referência: AbstractMap Class

HashTable

A classe HashTable é uma implementação de Map que é thread-safe, mas obsoleta.Recomenda-se o uso de HashMap em seu lugar. Referência: HashTable Class

HashMap

A classe HashMap é uma implementação de Map que armazena pares chave-valor em uma tabela de dispersão. Ela oferece acesso rápido e é a escolha preferida na maioriados casos. Referência: HashMap Class

Vantagens e Desvantagens

- HashSet: Rápido, sem elementos duplicados, ordem não garantida.
- LinkedList: Boa para inserção/remoção, acesso aleatório mais lento.
- Vector: Thread-safe, mas pode ser lento devido à sincronização.
- ArrayList: Rápido, não thread-safe.
- HashTable: Thread-safe, mas obsoleto em favor de HashMap.
- HashMap: Rápido, não thread-safe, amplamente utilizado.

Exemplos de Aplicações

- HashSet pode ser usado para manter uma lista de itens exclusivos, como tags em um sistema de marcação.
- ArrayList é útil para armazenar e manipular listas de objetos, como contatos em uma lista telefônica.
- HashMap é a escolha ideal para mapear palavras-chave para significados em um dicionário.

- Set é uma coleção que não permite elementos duplicados. O exemplo poderia sem uma lista de compra simples.

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

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

        set.add("Maçã");
        set.add("Banana");
        set.add("Pêra");
        set.add("Maçã"); // Não adiciona uma segunda "Maçã" (duplicada)

        System.out.println("Elementos no Set: " + set);
    }
}
```

- List é uma coleção que permite elementos duplicados e mantém a ordem de inserção. Neste exemplo não seria possível a criação de uma lista de compras, pois teriam itens iguais. Porém seria possível uma criação de uma playlist onde a música poderia se repetir.

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

import java.util.ArrayList;
import java.util.List;

public class ExemploList {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();

        list.add("Maçã");
        list.add("Banana");
        list.add("Pêra");
        list.add("Maçã"); // Adiciona uma segunda "Maçã" (duplicada)

        System.out.println("Elementos na List: " + list);
    }
}
```

- Map é uma coleção de pares chave-valor, onde as chaves são únicas. No exemplo abaixo poderíamos criar um programa no qual pessoa usa a chave para nome na criação em uma lista de compras e valor para informar a quantidade dos itens que deviria ser comprado.

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

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

        frutaParaQuantidade.put("Maçã", 3);
        frutaParaQuantidade.put("Banana", 5);
        frutaParaQuantidade.put("Pêra", 2);

        int quantidadeDeMacas = frutaParaQuantidade.get("Maçã");

// Obtém o valor associado à chave "Maçã"

        System.out.println("Quantidade de maçãs: " + quantidadeDeMacas);
    }
}
```

- Uma Queue representa uma fila, onde o primeiro elemento adicionado é o primeiro a ser removido. Exemplo básico de uma fila de atendimento de um banco, hospital.

```
import java.util.LinkedList;
import java.util.Queue;

public class ExemploQueue{
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<>();

        queue.offer("Eduardo");
        queue.offer("Pablo");
        queue.offer("Guilherme");

// Remove e retorna o primeiro elemento
        String nextPerson = queue.poll();


        System.out.println("Próxima fruta na fila: " + nextPerson);
    }
}
```

Conclusão

O Java Collections Framework é uma parte fundamental da linguagem Java, oferecendo uma variedade de classes e interfaces para atender às necessidades de diferentes tipos de coleções e mapeamentos. A escolha da estrutura adequada depende dos requisitos do projeto, levando em consideração as vantagens e desvantagens de cada implementação. O conhecimento dessas classes e interfaces é fundamental para desenvolver aplicativos Java eficientes e escaláveis.

As collections e mapas em Java são componentes essenciais para o desenvolvimento de aplicativos eficientes e flexíveis. As collections, como List, Set e Queue, permitem armazenar e manipular grupos de objetos de maneira organizada, tornando mais fácil o gerenciamento de dados. Já os mapas, como HashMap e TreeMap, fornecem uma estrutura chave-valor que é fundamental para a rápida recuperação e associação de informações. Ao combiná-los, os desenvolvedores têm à disposição uma ampla gama de ferramentas para atender às necessidades de seus aplicativos, desde o armazenamento de dados simples até a implementação de algoritmos complexos. No entanto, é importante escolher a estrutura adequada de acordo com os requisitos do projeto, garantindo assim a eficiência e a legibilidade do código. Em suma, as collections e mapas em Java são recursos poderosos que desempenham um papel fundamental no desenvolvimento de aplicativos robustos e eficazes.

Referências Bibliográficas

Java collections framework. Disponível em: <https://en.wikipedia.org/wiki/Java_collections_framework>.

Collection (Java Platform SE 8). Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html>.

Map (Java Platform SE 8). Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/util/Map.html>.