PROGRAMAÇÃO ORIENTADA A OBJETOS

Coleções
Parte 1

ROTEIRO

- O que são Collections?
- Interface Collections
- Classes Concretas Collections

O que são Collections?

- Coleções são objetos que agrupam vários elementos
- O framework Collections de Java permite trabalhar com:
  - **Interfaces:** Definição abstrata de coleções
  - **Implementações**: Objetos concretos
  - **Algoritmos**: Vários métodos para trabalhar com coleções

O que são Collections?

- Com as Collections utilizamos estruturas de dados existentes, sem nos preocuparmos com a maneira como são implementadas
- Java possui uma interface raiz chamada de Collections
- Elas são dinâmicas, isto é, podem crescer conforme a necessidade de expansão, diferente de arrays
- Nós estudamos Generics antes, pois Collections utiliza Generics

Interfaces Collections

- **Collection é o topo da API Collections, sendo disponível no pacote java.util.**
- **Collections é a API que oferece diversas coleções (interfaces e classes concretas).**
- **Interfaces e classes concretas que implementam a interface Collection, terão obrigatoriamente que fazer a implementação de diversos métodos que manipulam coleções de dados, tais como adicionar e remover elementos.**

Interfaces Collections

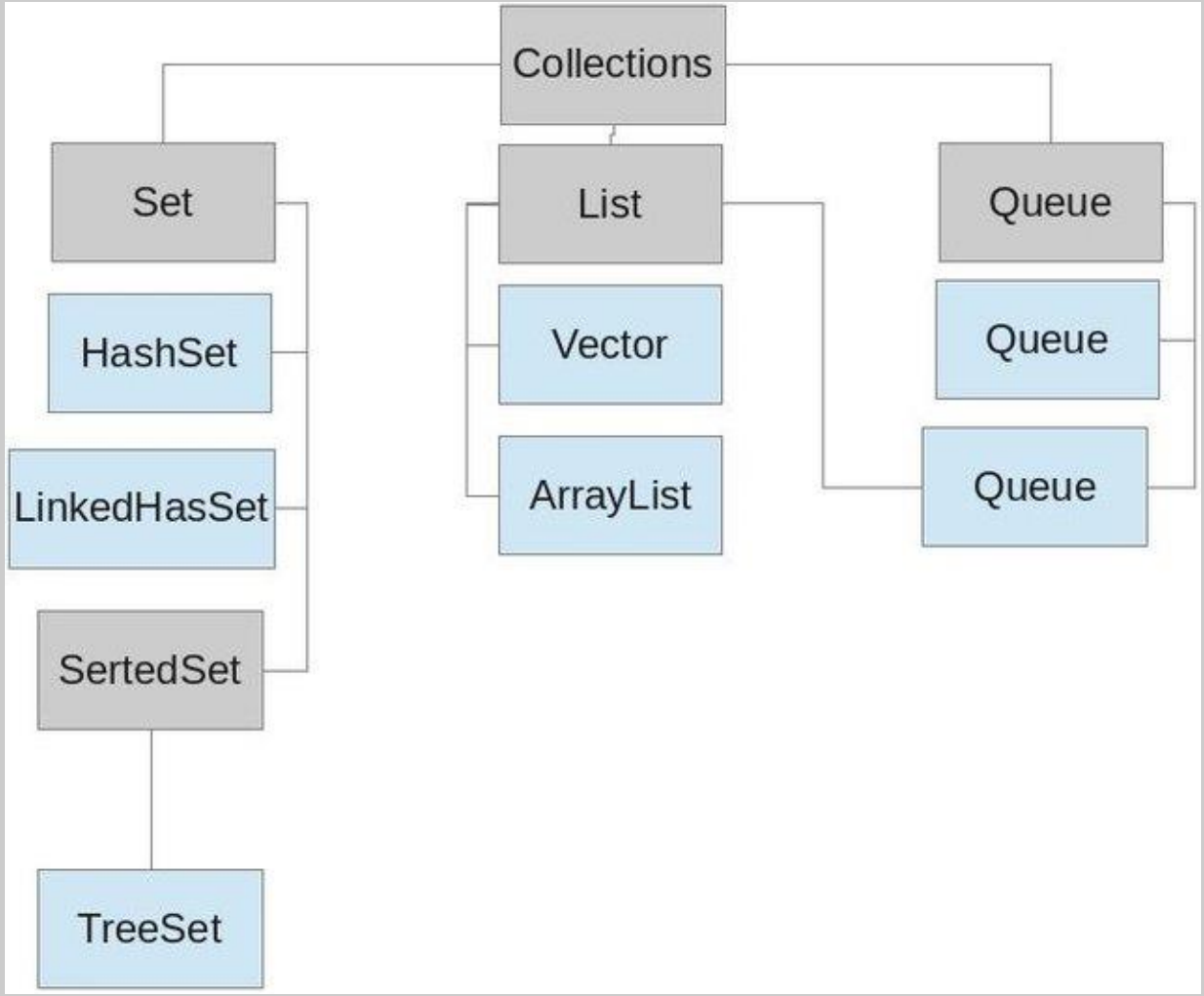
- As Collections podem ser organizadas e/ou Ordenadas
  - Quando organizadas: garante que as coleções serão percorridas na mesma ordem em que os elementos foram inseridos
    - Exemplos:
      - Organizada: LinkedHashSet, ArrayList, Vector, LinkedList, LinkedHashMap
      - Não-Organizada: HashSet, TreeSet, PriorityQueue, HashMap, HashTable, TreeMap

- **Quando ordenadas: possui métodos, regras para ordenação dos elementos.**
  - **Exemplos**
    - **Ordenada: TreeSet, PriorityQueue, TreeMap**
    - **Não-Ordenada: HashSet, LinkedHashSet, ArrayList, Vector, LinkedList, HashMap, HashTable, LinkedHashMap**

Interfaces Collections

- Nós estudamos Generics antes, pois Collections utiliza Generics
- O padrão Generics é aceitar qualquer tipo de elemento, incluindo elementos distintos (String, Double, Integer)
    - Também é possível especificar o tipo a ser aceito

Interfaces Collections



Fonte: 3

Interfaces Collections

- **Set**
  - Não permite elementos duplicados
- **SortedSet**
  - Igual a Set
  - Mantém os elementos ordenados
- **List**
  - Coleção de elementos ordenada
  - Pode ter elementos duplicados
  - Precisão da posição de cada elemento (indexação)

Interfaces Collections

- **Queue**
  - Representa uma fila genérica
  - Utiliza a regra FIFO

- **Deque**
  - Semelhante a Queue
  - Inserções e remoções podem ser feitas em qualquer extremidade

Classes Concretas Collections

- **Set**
  - **HashSet:** Armazena elemento numa tabela Hash e sem garantias de ordem dos elementos
  - **TreeSet:** Utiliza uma estrutura de árvore para armazenar os elementos. Mantém a ordem dos elementos
  - **LinkedHashSet:** Utiliza a tabela hash como uma lista ligada. Mantém a ordem de inserções dos elementos

Classes Concretas Collections

- **List**
  - **ArrayList:** implementação indexada dos elementos
  - **LinkedList:** Utiliza uma lista duplamente encadeada para armazenar os elementos

Classes Concretas Collections

- **Queue**
  - **Métodos como:**
    - **add e offer: inserem um elemento na fila**
    - **remove e pool: removem e retornam o elemento do início**
    - **element e peek: retornam, porém não removem, o elemento do início**

Classes Concretas Collections

- **Deque**
  - **Métodos**
    - **Inserção:** addFirst. offerFirst, addLast, offerLast
    - **Remoção:** removeFirst, poolFirst, removeLast, poolLast

Exemplo de Collections com Generics

- **Para mostrar um exemplo prático de uso de Collections, é importante discutir o *For-each* e o *Iterator***
  - **For-each: Trata-se de um ciclo for, embora adaptado para ser utilizado em Collections. Server para percorrer todos os elementos de qualquer Collection contida na API Collections**
  - **Iterator: É uma interface presente no java.útil que permite percorrer coleções da API Collection, desde que implementam a Collection.**
    - **Fornece métodos como**
      - **next(), hashnext(), remove()**

Exemplo de Collections com Generics

- **For-each**

  Sintaxe

  for(tipo elemento:tipo)

  Exemplo

```
List<Integer> myList = new ArrayList<Integer>();
        myList.add(1);
        myList.add(2);
        for (Integer listElements : myList) {
                System.out.println(listElements);
        }
```

Exemplo de Collections com Generics

- **Iterator**
  - **Exemplo**

```
List<Integer> myList = new ArrayList<Integer>();
        myList.add(1);
        myList.add(2);
        Iterator iMyList = myList.iterator();

        for(Integer listElements: myList){
                System.out.println(iMyList.next());
        }
```

Referências

1. Java Como Programar: Paul Deitel & Harvey Deitel - 10ª Edição
2. Java Como Programar: Paul Deitel & Harvey Deitel - 8ª Edição
3. Devemedia - https://www.devmedia.com.br/api-collections-em-java-fundamentos-e-implementacao-basica/28445

PROGRAMAÇÃO ORIENTADA A OBJETOS

Coleções
Parte 1