



10. Mapas

📅 Date	@27/10/2022
📁 Categoria	Java
📖 Curso	Java Collections: Dominando Listas Sets e Mapas

Tópicos

- Mapas
 - Características dos mapas
 - Adicionando elementos em um mapa
 - Iterando sobre um mapa
 - Garantindo a ordem de inserção de um mapa
 - Criando o mapa de alunos
 - Para saber mais: Chaves e Valores
 - Como continuar?
 - Projeto pronto
-

Características dos mapas

- Se repetimos uma chave, a chave repetida é sobrescrita pela nova.
 - Se uma chave for repetida, a antiga permanece, porém, o valor é sobrescrito pelo novo valor contido na chave passada, sendo o antigo valor "esquecido" pelo `Map`.
 - O valor pode ser repetido, diferente da chave.
 - A interface `Map` mapeia valores para chaves, ou seja, através da chave conseguimos acessar o valor.
 - Ela funciona da seguinte maneira, mapeia valores para chaves, e através da chave conseguimos acessar o valor correspondente. Por isso ela não pode ser repetida, ao contrário do valor, que podem existir iguais.
 - A interface `Map` **não** implementa a interface `Collection`, apesar de fazer parte da **API** de *Collections*.
-

Iterando sobre um mapa

Vimos que para adicionar um elemento em uma lista ou conjunto, utilizamos o método `add`. Mas por não implementar a interface `Collection`, `Map` não possui este método `add`. Qual método utilizamos para adicionar um elemento em um `Map`?

O método utilizado para adicionar um elemento em um `Map` é o método `put`. Ele recebe dois parâmetros, a **chave** e o **valor**. Os tipos desses parâmetros dependem do que definimos na hora da instância do objeto. Por exemplo, abaixo criamos um `Map` que recebe como chave a matrícula do aluno e como valor o seu nome:

```
Map<Integer, String> matriculaParaAluno = new HashMap<>();
matriculaParaAluno.put(123456, "Leonardo Cordeiro");
```

Então no `put`, nós temos que receber como parâmetro um inteiro, que representa a matrícula, e uma `String`, que representa o nome do aluno.

Iterando sobre um mapa

No código abaixo, temos um `Map` preparado que associa uma pessoa com a sua idade. Sabendo que temos acesso ao conjunto de chaves de um `Map` através do método `keySet()`, implemente um código que itere por esse conjunto e imprima os seus valores.

Podemos utilizar o `forEach` do Java 8 para iterar pelo conjunto de chaves que é retornado pelo método `keySet()`. Depois, para cada chave idade, nós pegamos o seu valor através do método `get` e imprimimos:

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

public class Exercicio {

    public static void main(String[] args) {

        Map<Integer, String> pessoas = new HashMap<>
        ();

        pessoas.put(21, "Leonardo Cordeiro");
        pessoas.put(27, "Fabio Pimentel");
        pessoas.put(19, "Silvio Mattos");
        pessoas.put(23, "Romulo Henrique");
        //...
    }
}
```

```
public class Exercicio {

    public static void main(String[] args) {

        Map<Integer,String> pessoas =new HashMap<>();

        pessoas.put(21, "Leonardo Cordeiro");
        pessoas.put(27, "Fabio Pimentel");
        pessoas.put(19, "Silvio Mattos");
        pessoas.put(23, "Romulo Henrique");

        pessoas.keySet().forEach(idade -> {
            System.out.println(pessoas.get(idade));
        });
    }
}
```

Garantindo a ordem de inserção de um mapa

Assim como o `HashSet`, o `HashMap` não mantém a ordem de inserção dos seus elementos. Mas há uma implementação de `Map` que garante essa ordem de inserção para nós, que implementação é essa?

- O `LinkedHashMap` continua nos dando a performance de um `HashMap`, mas com acesso previsível e ordenado, seguindo a inserção dos seus elementos.

Para saber mais: Chaves e Valores

Segundo o JavaDoc, um `Map` também pode ser visto como se fossem de 3 coleções:

The Map interface provides three collection views, which allow a map's contents to be viewed as a set of keys, collection of values, or set of key-value mappings.

Ou seja, temos uma coleção de chaves, uma coleção de valores, e uma coleção das associações. O interessante é que podemos acessar cada uma das coleções. Vamos tentar?

No editor abaixo já temos um `Map` preparado que associa o nome do aluno com a idade.


Para acessar apenas as chaves use o método `keySet()` do `Map`. Para acessar os valores existe o método `values()`.

Tente iterar (`foreach`) em cima das chaves e valores separadamente!

Resposta

Java-Collections/TesteMap.java at main · klayton-a-souza/Java-Collections

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters. Learn more about bidirectional Unicode characters You

 <https://github.com/klayton-a-souza/Java-Collections/blob/main/Codigo/gerenciad-or-de-cursos/src/br/com/alura/TesteMap.java>

layton-a-souza/**Java-collections**

1 Contributor 0 Issues 0 Stars 0 Forks

Como continuar?

O grande foco desse treinamento foi o uso da API de Collections, mas a viagem dentro da plataforma não para por aqui.

Se você tiver interesse em aprender mais sobre o mundo OO e design de classes, os cursos sobre SOLID e padrões de projetos podem ser interessantes para você:

- [OO: Melhores técnicas com Java](#)
- [SOLID com Java](#)
- [Design Patterns Java I](#)
- [Design Patterns Java II](#)

Ou ainda, deseja aprofundar o conhecimento nas estruturas de dados abordadas no decorrer do curso:

- [Estrutura de Dados](#)

Para quem quer entender melhor como funcionam as collections *thread-safe* e threads em geral, sugiro os seguintes cursos:

- [Threads I](#)
- [Threads](#)

E claro, tem muito mais, por exemplo a [API de Reflection](#) ou treinamentos sobre o mundo Java Web e backend em geral.
