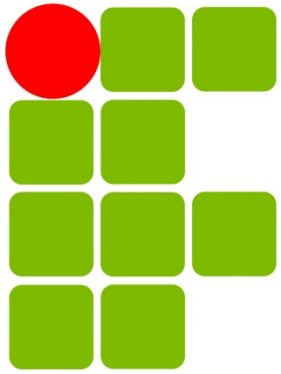


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA

Collections Framework: Map

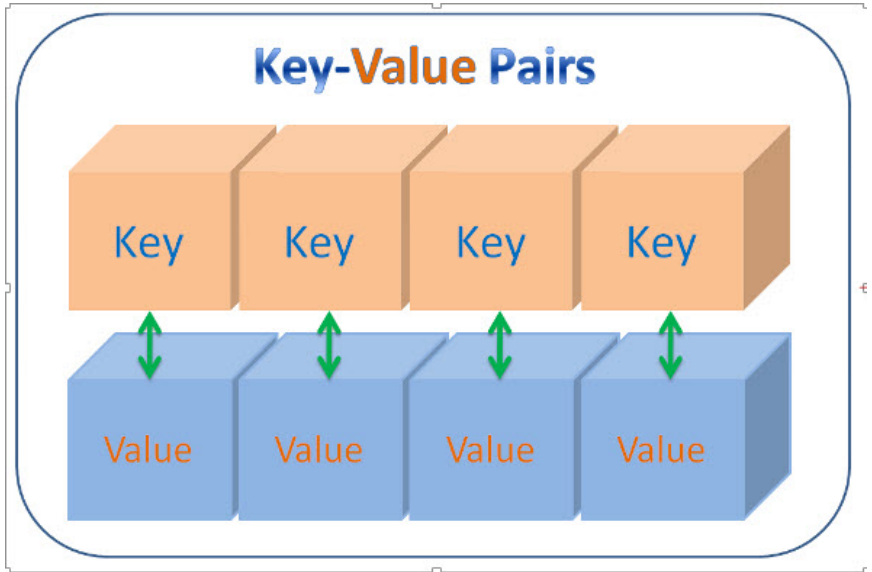
Programação Orientado a Objetos

Prof. Katyusco de F. Santos

Campina Grande/PB – 2017

O que estudaremos nesta aula?

- O que são *Mapas*, no contexto de Coleções?
- Como (e quando) utilizar *Mapas* (ou *Maps*) em Java?
- Como manipular estas estruturas?

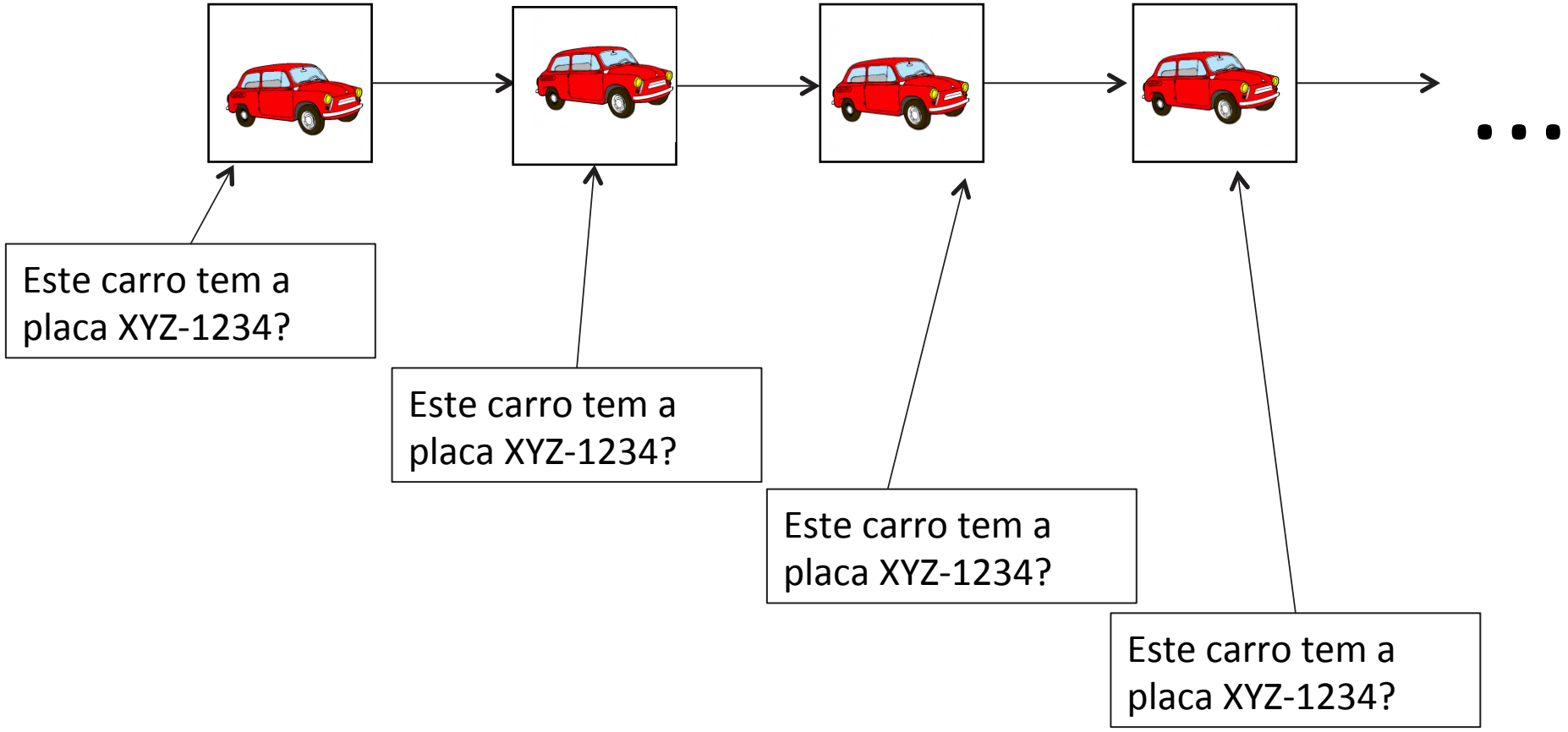


Mapas *versus* Lista

- **Em Lista, qual o procedimento para obter um certo *objeto O*, dado um *valor V*?**
  - **Por exemplo, como obter os dados de um carro, dada uma placa?**
- **Resposta: armazenar todos os objetos (carro) em uma lista, e percorrê-la em busca do valor (placa) informado.**
  - **Procedimento muito custoso.**

Mapas *versus* Lista

- **Exemplo: lista de carros.**



Mapas *versus* Lista

- Exemplo: lista de carros.

Este carro tem a placa XYZ-1234?

Este carro tem a placa XYZ-1234?

Este carro tem a placa XYZ-1234?

Este carro tem a placa XYZ-1234?

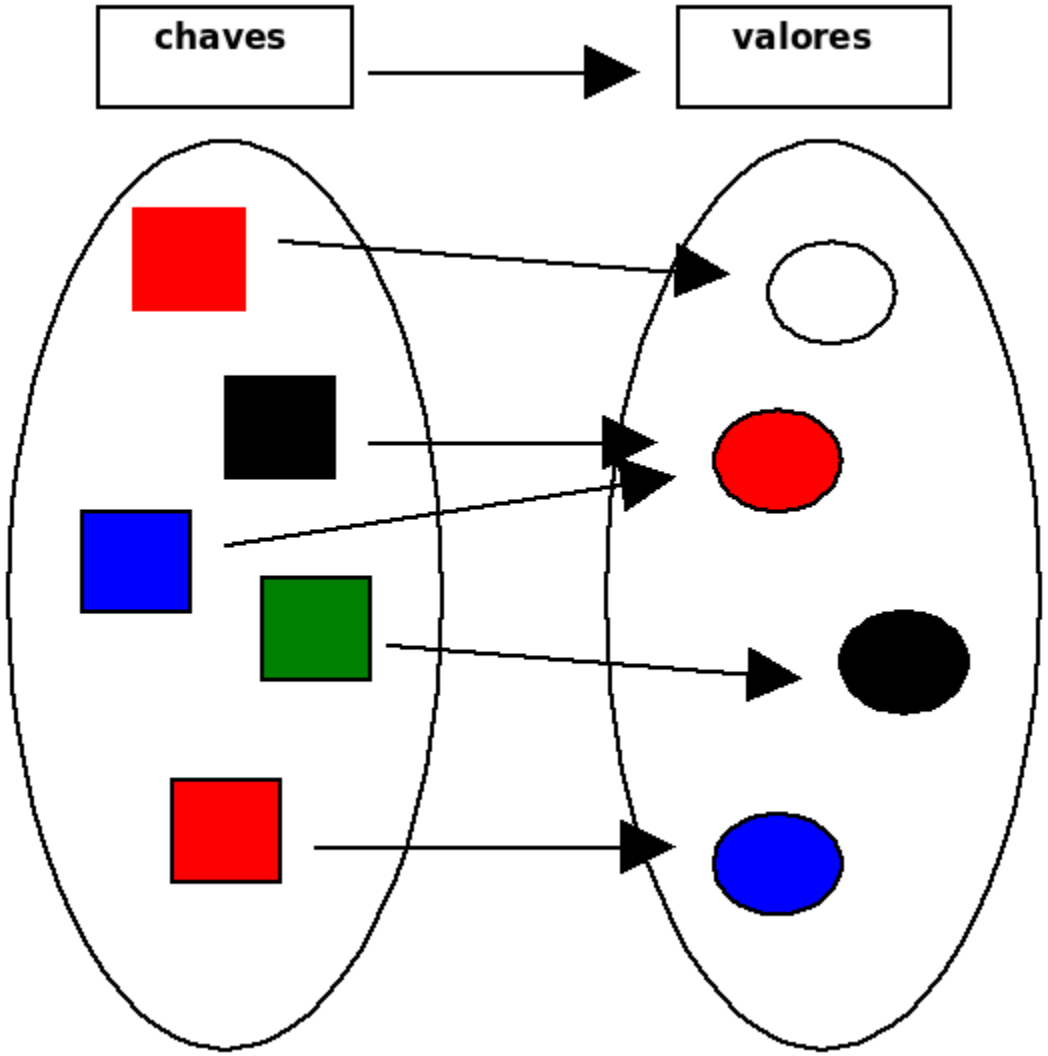
Verificar *objeto por objeto* é um procedimento lento e custoso!

Mapas

- **Um mapa é composto por um conjunto de associações entre um *objeto chave* a um *objeto valor:***
  - **É equivalente ao conceito de dicionário, usado em várias linguagens;**
  - ***{chave, valor}* ou *chave -> valor*.**
- **A Interface *java.util.Map* representa um mapa, pois é possível usá-lo para mapear uma chave a um valor, por exemplo.**

Mapas

- **Possíveis ações em um mapa:**
  - **Mapear uma chave a um valor;**
  - **O que está na chave X?;**
  - **Remapeie uma certa chave;**
  - **Obter o conjunto de chaves;**
  - **Obter o conjunto de valores;**
  - **Desmapeie a chave X.**

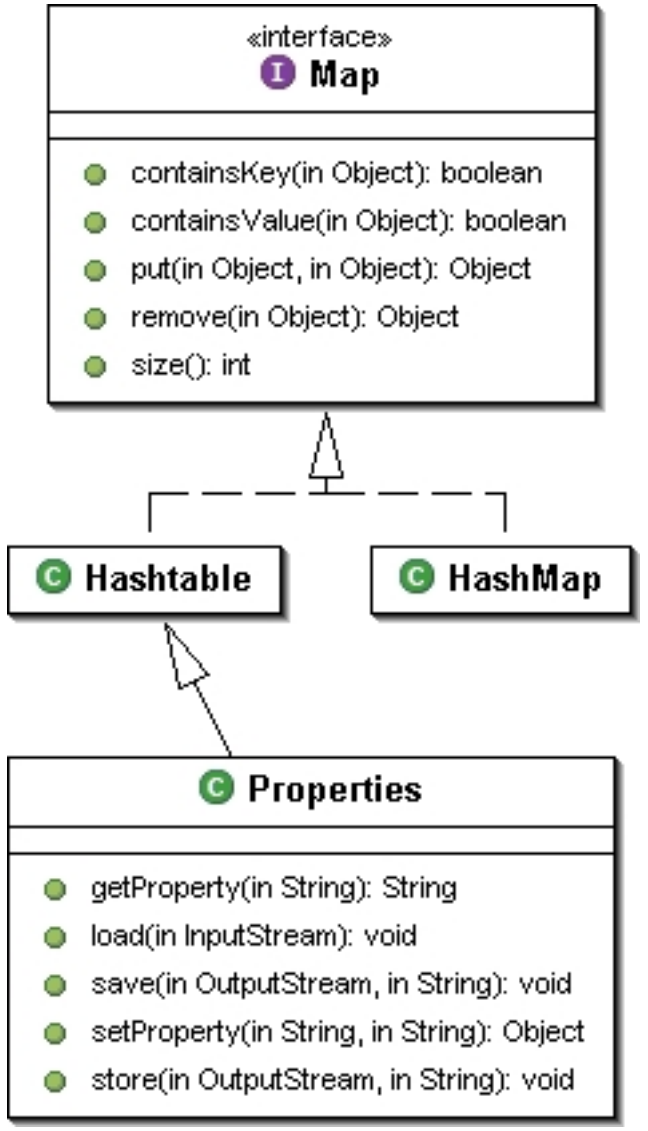


Mapas

- **Um mapa é muito usado para "indexar" objetos de acordo com determinado critério, para podermos buscar esse objetos rapidamente;**
- **Um mapa costuma aparecer juntamente com outras coleções, para poder realizar essas buscas!**
- **Uma mapa, assim como as *listas*, trabalha diretamente com *Objects* (tanto na chave quanto no valor)**

Mapas

- Suas principais implementações são o HashMap, o TreeMap e o Hashtable;
- Apesar do mapa fazer parte do framework, ele não estende a interface Collection, por ter um comportamento bem diferente.



Mapas

- **Criação de um mapa em Java**

```
Map mapa = new HashMap();
```

ou

```
HashMap mapa = new HashMap();
```

```
Map<Object, Object> lista = new HashMap<Object, Object>();
```

Programação Genérica (Generics)

Mapas

- **Exemplo de criação de um mapa em Java.**

```
Map<String, Carro> mapa = new HashMap<String, Carro>();

mapa.put("XYZ-1234",   new Carro("Gol","Ano 2010","48000km");
mapa.put("XYZ-1235",   new Carro("Corsa","Ano 2015","3000km");

Carro c1 = mapa.get("XYZ-1234"); // Obtém obj. Carro dada a placa
Carro c2 = mapa.get("XYZ-1235"); // Obtém obj. Carro dada a placa

System.out.println(c1.getNomeCarro());
System.out.println(c1.getAnoCarro());
```

Mapas

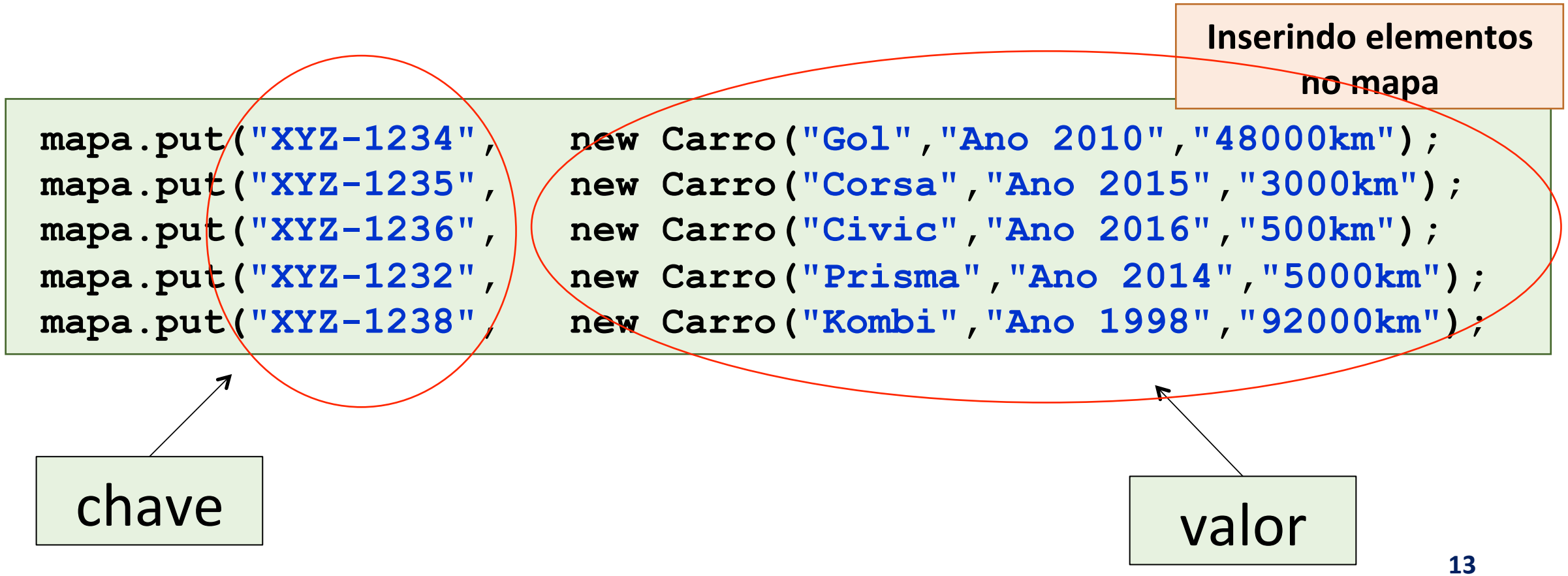
- **Manipulando um mapa em Java**

Inserindo elementos no mapa

```
mapa.put("XYZ-1234",   new Carro("Gol","Ano 2010","48000km");
mapa.put("XYZ-1235",   new Carro("Corsa","Ano 2015","3000km");
mapa.put("XYZ-1236",   new Carro("Civic","Ano 2016","500km");
mapa.put("XYZ-1232",   new Carro("Prisma","Ano 2014","5000km");
mapa.put("XYZ-1238",   new Carro("Kombi","Ano 1998","92000km");
```

Mapas

- **Manipulando um mapa em Java**



Mapas

- **Manipulando um mapa em Java**

```
// Sem uso de Casting!
Carro c1 = mapa.get("XYZ-1234");
Carro c2 = mapa.get("XYZ-1235");

// Será impresso o nome "Gol"
System.out.println(c1.getNomeCarro());
```

Obtendo elementos do mapa

Mapas

- **Manipulando um mapa em Java**

```
// Sem uso de Casting!
Carro c1 = mapa.get("XYZ-1234");
Carro c2 = mapa.get("XYZ-1235");

// Será impresso o nome "Gol"
System.out.println(c1.getNomeCarro());
```

Obtendo elementos do mapa

chave

Mapas

- **Manipulando um mapa em Java**

Verificando a existência de uma chave

```
mapa.containsKey("XYZ-1234"); // true
mapa.containsKey("XYZ-1235"); // true
mapa.containsKey("XYZ-1122"); // false
```

```
int tamanhoMapa = mapa.size(); // 5
```

Obtendo o tamanho de um mapa

Mapas

- **Manipulando um mapa em Java**

```
mapa.clear(); // limpa o mapa
```

Removendo todos os elementos do mapa

```
mapa.isEmpty(); // mapa está vazio?
```

Verificando se mapa está vazio

Mapas

- **Manipulando um mapa em Java**

```
mapa.remove("XYZ-1238");
```

Removendo um elemento específico referente a uma chave

```
mapa.containsValue(c1);
```

Verificando se mapa contém um valor específico

Mapas

- **Manipulando um mapa em Java**

```
mapa.keySet(); // todas as chaves
```

Retornando todas as chaves de um mapa (um conjunto)

```
mapa.values(); // todas os valores
```

Retornando todos os valores de um maa (uma Coleção)

Mapas

- **Percorrendo todos os elementos de um mapa.**

```
Map<String, Carro> mapa = new HashMap<String, Carro>();
mapa.put("XYZ-1234",   new Carro("Gol","Ano 2010","48000km");
mapa.put("XYZ-1235",   new Carro("Corsa","Ano 2015","3000km");
mapa.put("XYZ-1232",   new Carro("Corsa","Ano 2015","3000km");
mapa.put("XYZ-1238",   new Carro("Corsa","Ano 2015","3000km");

for(Map.Entry<String, Carro> umElemento : mapa.entrySet()){
  Carro c = umElemento.getValue(); // obtém o valor
  System.out.println(umElemento.getKey()); // imprime a chave
  System.out.println(c.getNomeCarro()); // imprime o nome do carro
}
```

Exercícios

1. Crie um programa em Java para executar o trecho de código do slide 20 (isto inclui você criar a classe Carro).
2. Crie a classe `Pessoa`, que deve possuir um construtor que receba como parâmetro o nome, a idade e o CPF. OBS: Crie getters e setters.
3. Crie a classe `GerenciaPessoa`, que deve possuir um mapa como variável de instância. Esta classe deverá possuir um método de adicionar uma pessoa (para inserir uma pessoa no mapa) e um método para obter uma pessoa (para obter uma pessoa do mapa, dado um CPF). OBS.: O CPF deve ser a chave.
4. Você deverá criar uma classe principal que insere no objeto `GerenciaPessoa` cinco pessoas. Teste também a recuperação de uma pessoa, dado um CPF que você previamente inseriu.

Dicas de Leitura & Referências Utilizadas

- Apostila Java e Orientação a Objetos – Capítulo 16 Collections Framework

- Javadoc do Collection Framework