

AULA 3 – USANDO COLLECTIONS

GS1020 – Programação Orientada a Objetos II

Prof. Dr. Murillo G. Carneiro
mgcarneiro@ufu.br



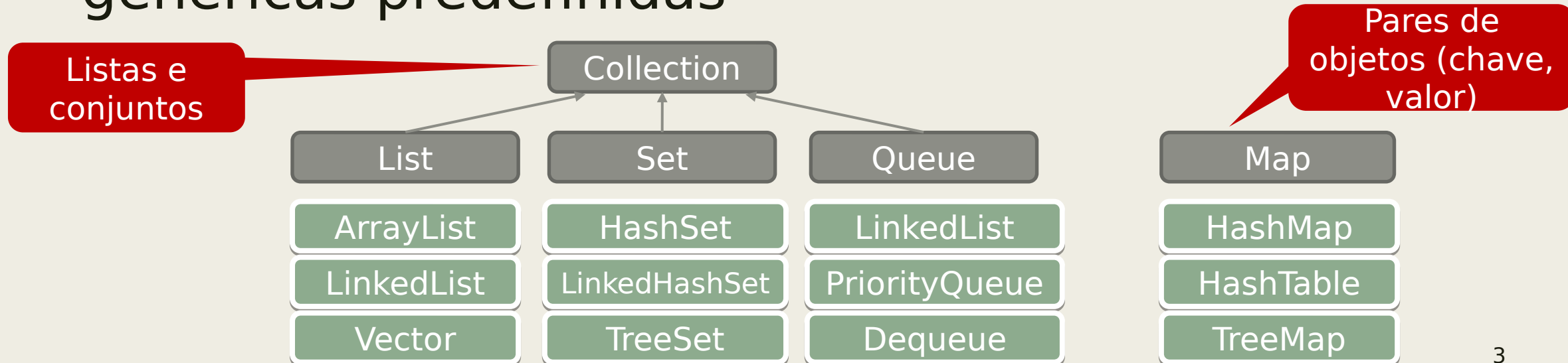
Material baseado nos slides cedidos pelo Prof. Rafael D. Araújo (FACOM/UFU)

Objetivo da aula

- Entender e implementar classes e métodos com collections em Java.

Collections

- Uma coleção é uma estrutura de dados — na realidade, um objeto — que pode armazenar referências a outros objetos
- O Java provê um framework de coleções (java.util) que contém estruturas de dados genéricas predefinidas



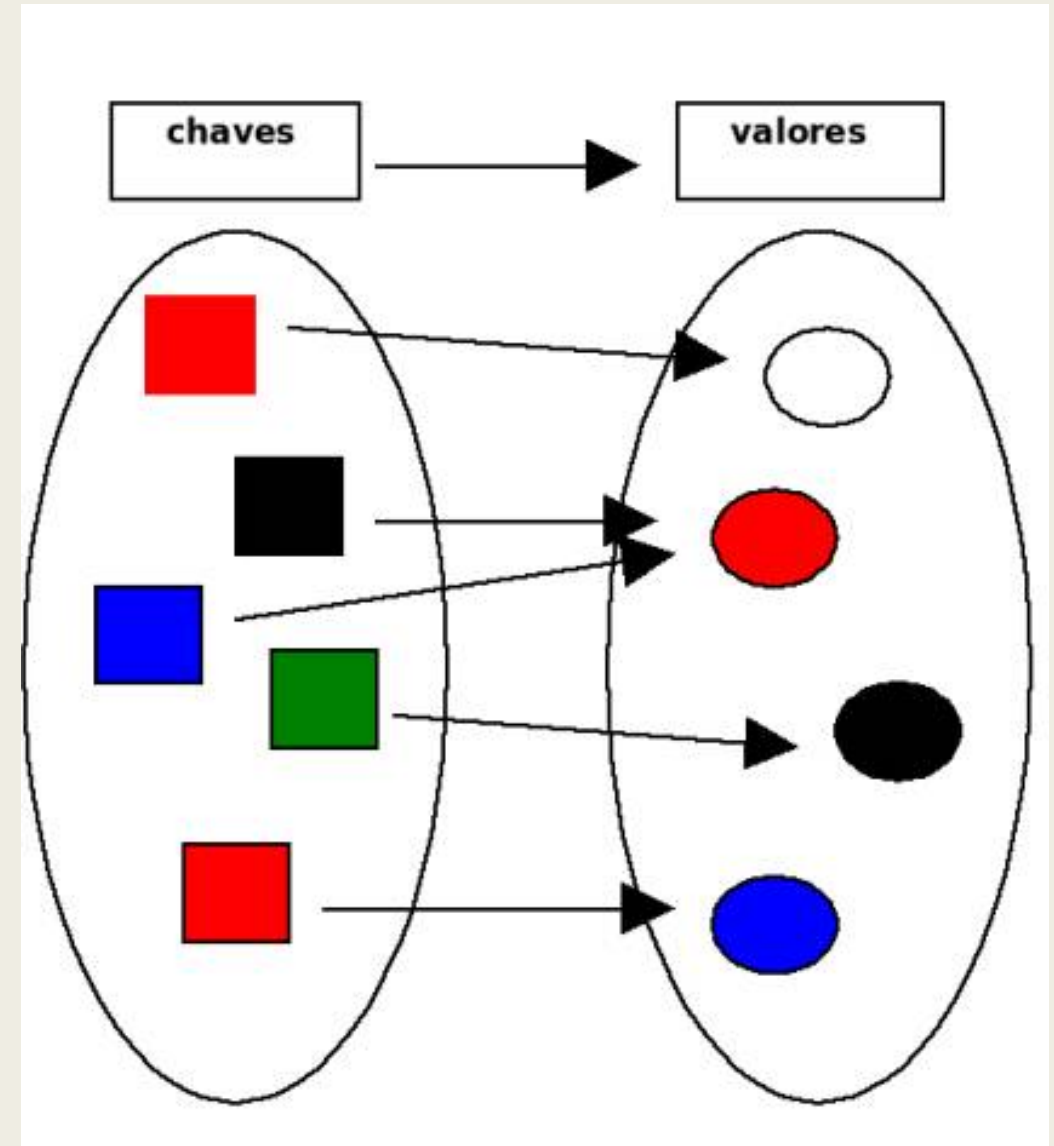
Collections

■ Diferença entre as estruturas de dados genéricas

| Interface | Descrição |
|-------------------|---|
| Collection | A interface-raiz na hierarquia de coleções a partir da qual as interfaces Set, Queue e List são derivadas. |
| Set | Uma coleção que <i>não</i> contém duplicatas. |
| List | Uma coleção ordenada que <i>pode</i> conter elementos duplicados. |
| Map | Uma coleção que associa chaves a valores e que <i>não pode</i> conter chaves duplicadas. Map não deriva de Collection. |
| Fila | Em geral, uma coleção <i>primeiro a entrar, primeiro a sair</i> que modela uma <i>fila de espera</i> ; outras ordens podem ser especificadas. |

Map

- Um mapa é um conjunto de associações de um objeto chave a um objeto valor.
- As chaves nunca podem se repetir.



Collections - Exemplo

```
List<String> listaNomes = new ArrayList<String>();  
listaNomes.add("Manoel");  
listaNomes.add("Joaquim");  
listaNomes.add("Maria");
```

```
List<Animal> listaNomes = new ArrayList<Animal>();  
listaNomes.add(new Gato());  
listaNomes.add(new Cachorro());
```

```
Map<String,Integer> exemploMap = new HashMap<String,Integer>();  
exemploMap.put("Maria", 20);  
exemploMap.put("João", 22);
```

Exercício 1

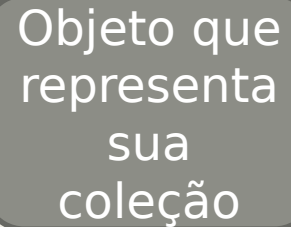
- Crie uma lista de nomes de pessoas.
- Utilize o método `sort()` para ordená-la.
- Imprima a lista.

Interface Iterator

- A interface Iterator define métodos para percorrer Collections
 - *iterator(): método que retorna um objeto Iterator para a coleção*
 - *hasNext(): retorna o valor true se ainda existir objeto a ser percorrido na coleção*
 - *next(): devolve o próximo objeto da coleção*

Interface Iterator

■ Exemplo:



Objeto que
representa
sua
coleção

```
Iterator it = lista.iterator();  
while(it.hasNext()){  
    System.out.println("=>" + (String)it.next());  
}
```

Exercício 2

- Implemente um método que recebe duas listas (lista1 e lista2) e imprime os elementos que existem tanto na lista1 quanto na lista 2.
 - *Utilize os métodos da interface Iterator para percorrer as listas.*
 - *Dica 1: o método “contains(Object)” das coleções verifica se um objeto existe na coleção.*
 - *Dica 2: também é possível utilizar um objeto do tipo HashSet, pois ele não permite inserir itens repetidos.*

Exercício 3

- Implemente um método que recebe um texto e conta a quantidade de cada palavra (considerar texto com letras minúsculas).
 - *Dica: Utilize um objeto do tipo HashMap*
 - *Key: cada palavra do texto*
 - *Value: quantidade de ocorrências da palavra*

Exemplo de entrada: Teste do teste para teste

Saída:
teste, 3
do, 1
para, 1

Exercício 4

- Crie uma classe Pessoa que contém os atributos nome (String) e endereço (String).
- Crie um HashMap para armazenar placas de carros (String) e seu respectivo dono (Pessoa).
- Adicione valores de teste.
- Imprima o HashMap apresentando a placa com o nome e o endereço do dono.

Exercício 5

- Implemente um método que recebe uma lista de nomes de cidades (sem repetição) e o método deve imprimir os nomes agrupados pela quantidade de caracteres do nome.
 - *agrupaCidades(ArrayList<String> cidades)*
 - *Utilize um objeto do tipo HashMap para agrupar*

Exemplo de entrada: Luz, Bauru, Monte Carmelo, Belém, Itu

Saída:

3 [Luz, Itu]

5 [Bauru, Belém]

13 [Monte Carmelo]

Referências

- DEITEL, H. M. Java: como programar, 8. ed. São Paulo: Prentice Hall, 2010. Capítulo 16 e 20.