Coleções de Objetos
Interface Set e
Classes HashSet e TreeSet

Profa. Karen Selbach Borges

Última Atualização: Jun/2025

Interface Set

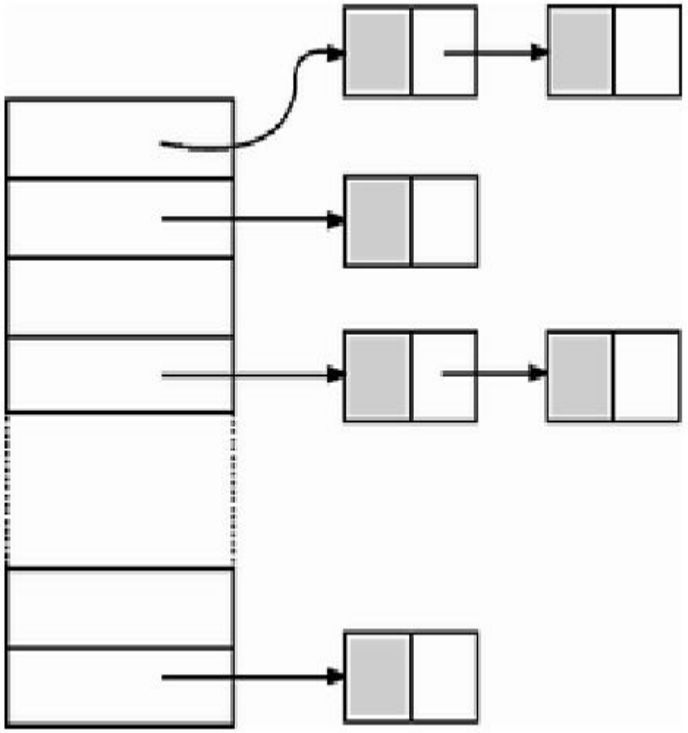
- É uma especialização de Collection
- Implementa coleções que não podem conter objetos duplicados.
- A duplicidade é verificada através da implementação de equals.
- Exemplos :
  - conjunto de disciplinas cursadas por um aluno
  - conjunto de cartas em um jogo de poker

Métodos da Interface Set

- Possui os métodos definidos em Collection.
- Especial atenção ao método add:
  - boolean **add**(E e) : retorna false se tentar adicionar um elemento que já existe.

Classe HashSet

- Implementa uma estrutura do tipo Tabela de Hash:
  - É um array de listas encadeadas
  - O elemento é inserido em uma posição calculada (*hash code*)



Classe HashSet

- Construtores:
  - **HashSet**( ): 16 posições iniciais
  - **HashSet**(Collection c)
  - **HashSet**(int initialCapacity)

Exemplo

- Dada uma classe Pessoa

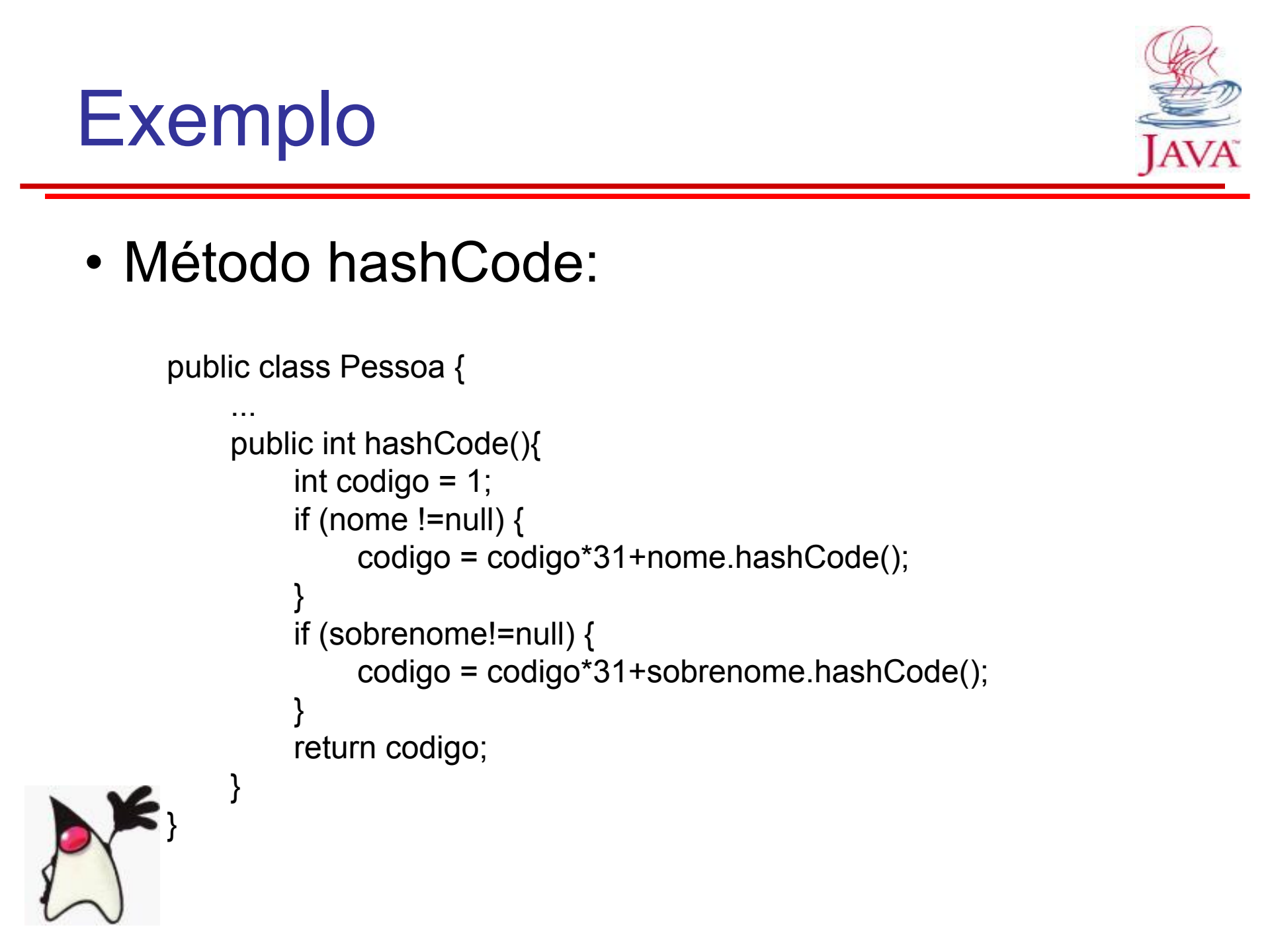
```
public class Pessoa {
        private String nome;
        private String sobrenome;

        public Pessoa(String nome, String sobrenome) {
                this.nome = nome;
                this.sobrenome = sobrenome;
        }

        public String toString(){
                return (nome + " " + sobrenome);
        }
}
```

Exemplo

- Método equals:

```
public class Pessoa {
        ...
        public boolean equals(Object obj) {
                Pessoa outro = (Pessoa) obj;
                if (nome.equals(outro.nome)){
                        if (sobrenome.equals(outro.sobrenome)){
                                return true;
                        }
                }
                return false;
        }
}
```

Exemplo



- Método hashCode:

```
public class Pessoa {
    ...
    public int hashCode(){
        int codigo = 1;
        if (nome !=null) {
            codigo = codigo*31+nome.hashCode();
        }
        if (sobrenome!=null) {
            codigo = codigo*31+sobrenome.hashCode();
        }
        return codigo;
    }
}
```

Exemplo

```
Pessoa fulano = new Pessoa(“Pedro”, “Rocha”);
Pessoa sicrano = new Pessoa(“Pedro”, “Rocha”);
Pessoa beltrano = new Pessoa(“Paulo”, “Silva”);

HashSet set = new HashSet();
set.add(fulano);
set.add(beltrano);
set.add(sicrano); // Não insere porque já existe Pedro Rocha

System.out.println("\nConteúdo da Lista");
Iterator it = set.iterator();
while(it.hasNext()){
      System.out.println(it.next());
}
```

Classe TreeSet

- Implementa uma coleção, onde os elementos estão ordenados (não importa em que ordem foram adicionados).
- Se os elementos forem tipo primitivos, ele ordenará pelo valor, sem necessidade de cast.

Classe TreeSet

```
TreeSet<Integer> tree = new TreeSet<Integer>();
    tree.add(12);
    tree.add(63);
    tree.add(34);
    tree.add(45);

    Iterator<Integer> iterator = tree.iterator();
    System.out.print("Tree set data: ");
    while (iterator.hasNext()) {
        System.out.print(iterator.next() + " ");
```

Classe TreeSet

- Se os elementos forem objetos, a classe que define esses objetos deve implementar a interface Comparable
- OU os objetos devem ser aceitos por um Comparator
- Exemplo de uso :
  - Lista telefônica

Classe TreeSet

- Construtores:
  - **TreeSet**() : árvore vazia, onde os elementos são ordenados conforme a sua ordem natural (uso de Comparable)
  - **TreeSet**(Comparator c) : árvore vazia, onde os elementos são ordenados conforme o objeto Comparator

Classe TreeSet

- Os dados inseridos na árvore são automaticamente classificados
- Construtores
  - **TreeSet**(Collection c) : árvore inicializada com os elementos da coleção, ordenados conforme a sua ordem natural
  - **TreeSet**(SortedSet s) : árvore inicializada com os elementos do set, ordenados conforme a ordem da TreeSet

Exemplo

- Dada uma classe Pessoa

```
public class Pessoa implements Comparable{
    private String nome;
    private String sobrenome;
    private int idade;

    public Pessoa(String nome, String sobrenome) {
        this.nome = nome;
        this.sobrenome = sobrenome;
    }

    public int compareTo(Object obj){
        Pessoa p = (Pessoa)obj;
        int teste = this.getNome().compareTo(p.getNome());
        if (teste==0) teste = this.getSobrenome().compareTo(p.getSobrenome());
        return (teste);
    }
}
```

Exemplo

```
import java.util.*;
public class TestaTreeSet {
	public static void main(String[ ] args) {
		TreeSet sorter = new TreeSet();
		Pessoa fulano = new Pessoa("Rafael", "Quadros");
		sorter.add(fulano);
		Pessoa sicrano = new Pessoa("Pedro", "Rocha");
		sorter.add(sicrano);
		Pessoa beltrano = new Pessoa ("Paulo", "Silva");
		sorter.add(beltrano);

		System.out.println("\nPessoas ordenadas pelo nome");
		Iterator it = sorter.iterator();
		while (it.hasNext()){
			Pessoa p = (Pessoa) it.next();
			System.out.println(p);
		}
	}
}
```

Classe TreeSet

Alguns outros métodos interessantes:

- public SortedSet<E> subSet(E fromElement, E toElement): retorna os elementos que vão de fromElement (inclusive) até toElement (exclusive)
- public SortedSet<E> headSet(E toElement): retorna os elementos menores do que toElement
- public SortedSet<E> tailSet(E fromElement): retorna os elementos maiores ou iguais a fromElement

Comparação

Feature	HashSet	TreeSet
Data Structure	Hash Table	Red-Black Tree
Ordering	Unordered	Naturally Ordered/Custom Ordering via Comparator
Null Handling	Supports one null	Does not support null
Performance (basic operations)	O(1) (average case)	O(log n)
Initial Capacity & Load Factor	Supported	Not Supported
Custom Ordering	Not Supported	Supported