

# Linguagem de Programação Orientada a Objetos I

Coleções de Objetos – Sets e Maps

Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>

The bottom of the slide features a decorative graphic consisting of several overlapping, wavy, horizontal bands. The colors transition from a light blue at the top, through a dark grey, to a light grey with a fine diagonal line pattern at the bottom.

# Introdução

- ▶ Implementação das Coleções
  - Três interfaces básicas na organização das coleções:
    - ▢ Conjuntos (sets)
    - ▢ Listas (Lists)
    - ▢ Mapas (Maps)

Implementations						
		Hash Table	Resizable Array	Balanced Tree	Linked List	Hash Table + Linked List
Interfaces	Set	<a href="#">HashSet</a>		<a href="#">TreeSet</a>		<a href="#">LinkedHashSet</a>
	List		<a href="#">ArrayList</a>		<a href="#">LinkedList</a>	
	Map	<a href="#">HashMap</a>		<a href="#">TreeMap</a>		<a href="#">LinkedHashMap</a>

# Conjuntos de Objetos (Sets)

- ▶ Conjuntos (Sets)
  - São coleções onde objetos duplicados não são permitidos
  - Tamanho dos conjuntos pode variar dinamicamente

# Conjuntos de Objetos (Sets)

- ▶ Conjuntos ou Sets
  - São encapsulados por uma instância de uma das classes que implementam a interface **Set**
    - ▢ **HashSet**
    - ▢ **TreeSet**
  - Pacote *java.util*
    - ▢ `import java.util.HashSet;`
    - ▢ `import java.util.TreeSet;`
  - Consulte
    - ▢ <http://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html>
    - ▢ <http://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html>

# Conjuntos de Objetos (Sets)

- HashSet

- ▮ Usa um mecanismo interno de representação dos objetos que é mais rápido para as operações de modificação do conjunto, mas não estabelece nenhuma ordem em particular entre os objetos

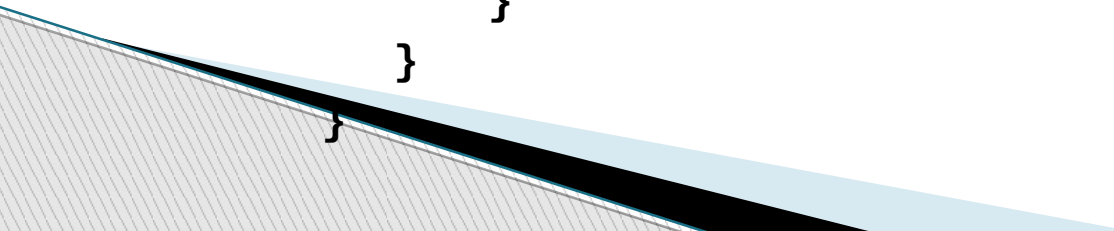
- TreeSet

- ▮ Preserva a ordem natural dos objetos, mas à custa da perda de desempenho na inserção e remoção

# HashSet - Exemplo (1)

```
import java.util.HashSet;
import java.util.Iterator;
public class HashSetTest {
    public static void main(String[] args) {
        HashSet<String> paises = new HashSet<>();
        paises.add("Japao");
        paises.add("Brasil");
        paises.add("Estados Unidos");
        paises.add("Brasil");
        paises.add("Japao");
        System.out.println("Impressao da HashSet:");

        for(String nomePais : paises){
            System.out.println(nomePais);
        }
    }
}
```



# HashSet - Exemplo (2)

```
import java.util.HashSet;
import java.util.Iterator;
public class HashSetTest2 {
    public static void main(String[] args) {
        HashSet<String> paises = new HashSet<String>();
        paises.add("Japao");
        paises.add("Brasil");
        paises.add("Estados Unidos");
        paises.add("Brasil");
        paises.add("Japao");
        if(paises.contains("Brasil")){
            System.out.println("O conjunto contem Brasil");
        }
    }
}
```

Método  
contains



# Mapas de Objetos (Maps)

- ▶ Mapas de Objetos (Maps)
  - Mapas são conjunto de pares de objetos – um chamado *chave* e o outro, chamado *valor*
  - Mapas não permitem chaves repetidas, mas permitem valores repetidos (chaves diferentes podem estar associadas a valores iguais)
  - Duas classes que implementam **Map**
    - ▢ HashMap
    - ▢ TreeMap



# Mapas de Objetos (Maps)

## ► Mapas de Objetos (Maps)

- Pacote *java.util*

- ▮ `import java.util.HashMap;`
  - ▮ `import java.util.TreeMap;`

- Consulte

- ▮ <http://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>
  - ▮ <http://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html>

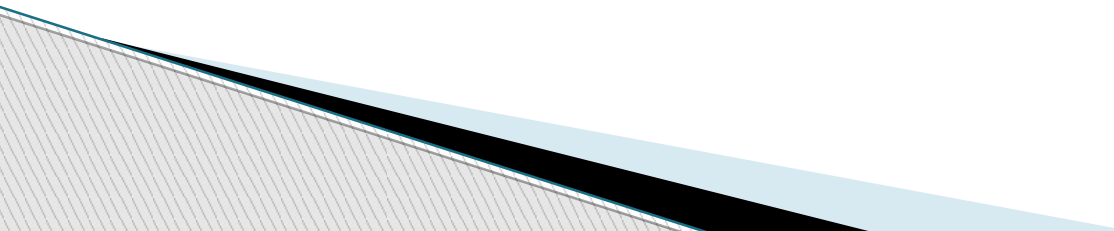
# HashMap - Exemplo

```
import java.util.HashMap;

public class HashMapTest {
    public static void main(String[] args) {
        HashMap<String,String> notasReal = new HashMap<>();
        notasReal.put("1", "Beija-Flor");
        notasReal.put("2", "Tartaruga ");
        notasReal.put("5", "Garca");
        notasReal.put("10", "Arara");
        notasReal.put("20", "Mico-leão-dourado");
        notasReal.put("50", "Onça Pintada");
        notasReal.put("100", "Garoupa");

        System.out.println("Impressao da HashMap:");

        for(String chave : notasReal.keySet()){
            String valor = notasReal.get(chave);
            System.out.println("Chave: " + chave + " Valor: " + valor);
        }
    }
}
```

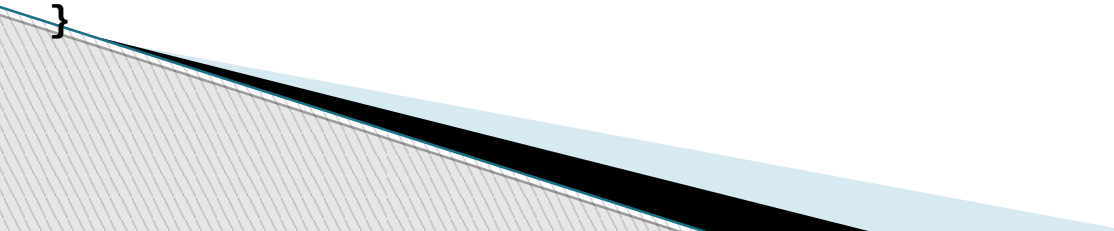


# TreeMap - Exemplo

```
import java.util.TreeMap;

public class TreeMapTest {
    public static void main(String[] args) {
        TreeMap<String, String> notasReal = new TreeMap<>();
        notasReal.put("1", "Beija-Flor");
        notasReal.put("2", "Tartaruga ");
        notasReal.put("5", "Garca");
        notasReal.put("10", "Arara");
        notasReal.put("20", "Mico-leão-dourado");
        notasReal.put("50", "Onça Pintada");
        notasReal.put("100", "Garoupa");
        System.out.println("Impressao da TreeMap:");

        for (String chave : notasReal.keySet()) {
            String valor = notasReal.get(chave);
            System.out.println("Chave: " + chave +
                               " Valor: " + valor);
        }
    }
}
```



# Wrappers e TreeMap

## ► Coleções

- As coleções armazenam objetos e não dados primitivos
- Assim, devemos usar Wrapper Classes
- Exemplo do TreeMap:
  - ❑ `notasReal.put("1","Beija-Flor");`
  - ❑ O valor da nota está sendo inserindo no mapa como uma String
- Solução:
  - ❑ Usar a classe wrapper Integer
  - ❑ `TreeMap<Integer, String> = new TreeMap<>();`
  - ❑ `notasReal.put(1,"Beija-Flor");`