

# **AULA**

# **LINGUAGEM DE PROGRAMAÇÃO I**

---

**COLLECTION - CONTINUAÇÃO**

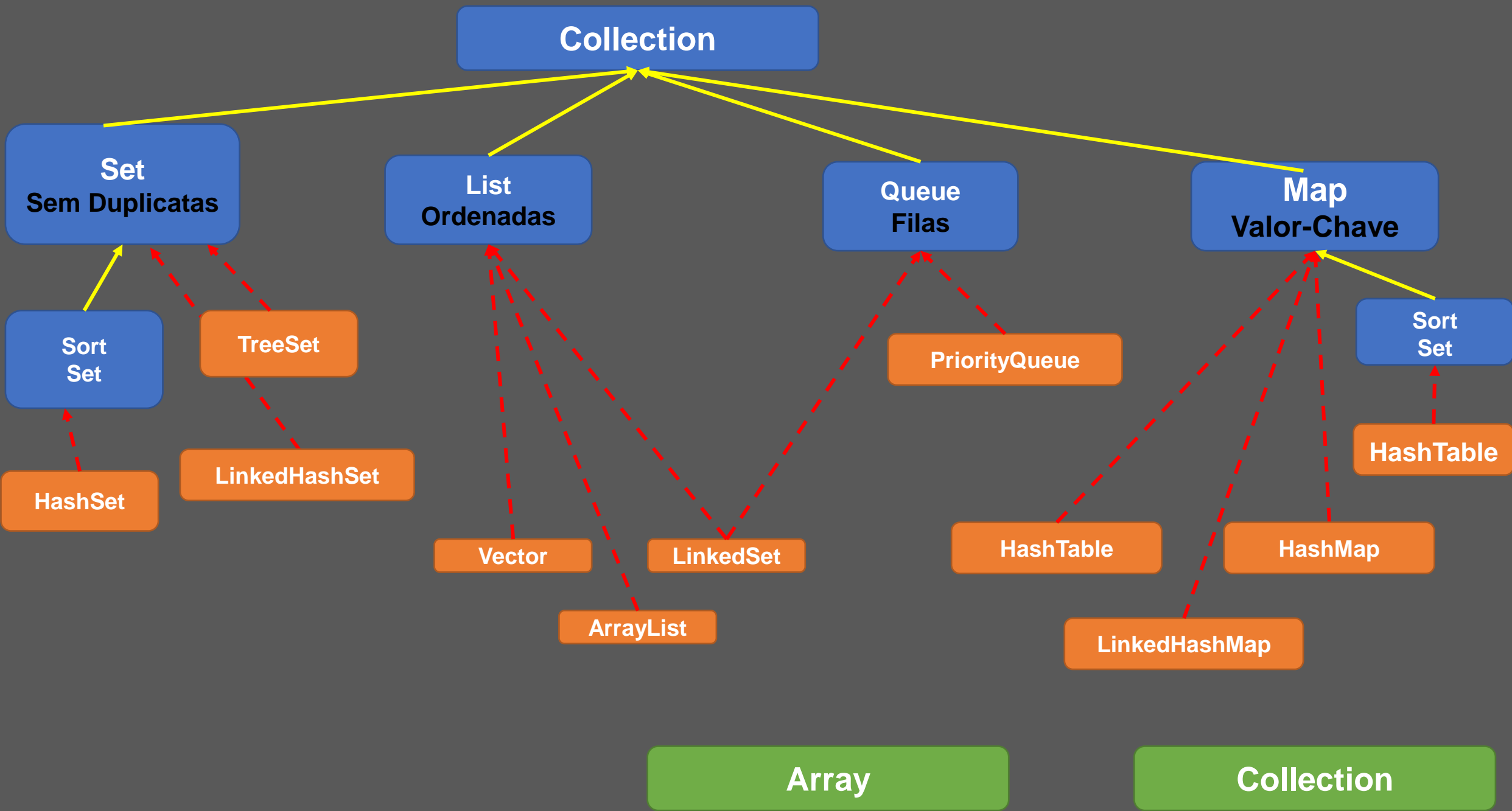
**Prof. Fábio Modesto**

**fabiomodesto@ifsp.edu.br**

**Instituto Federal de Educação, Ciência e Tecnologia  
de São Paulo**

# Agenda

- 1) Set
- 2) HashSet
- 3) TreeSet
- 4) LinkedHashSet
- 5) Teste de Performance no TreeSet, HashSet e LinkedHashSet
- 6) Queue
- 7) Map



# Collection - Set

- **Set**
  - **Coleção que não pode conter elementos duplicados.**
  - **Ele modela a abstração do conjunto matemático.**
  - **A interface Set contém apenas métodos herdados da Collection e adiciona a restrição de que elementos duplicados são proibidos.**

# Set - Métodos

## **add( )**

Adiciona um objeto à coleção.

## **clear( )**

Remove todos os objetos da coleção.

## **contains( )**

Retorna verdadeiro se um objeto especificado for um elemento dentro da coleção.

# Set - Métodos

## **isEmpty( )**

Retorna verdadeiro se a coleção não tiver elementos.

## **iterator( )**

Retorna um objeto Iterator para a coleção, que pode ser usado para recuperar um objeto.

## **remove( )**

Remove um objeto especificado da coleção.

## **size( )**

Retorna o número de elementos da coleção.

```
import java.util.Arrays;  
import java.util.List;  
import java.util.Set;  
import java.util.HashSet;
```

```
public class Colecoes {  
    public static void main(String[] args)  
    {  
        String [] cores = {"branco", "azul", "verde", "vermelho", "amarelo", "amarelo", "verde"};  
        List <String> lista = Arrays.asList(cores);  
        System.out.println(cores);  
    }  
}
```

# Saida:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ AulaCollection ---  
[branco, azul, verde, vermelho, amarelo, amarelo, verde]
```

-----



# Collection - Set

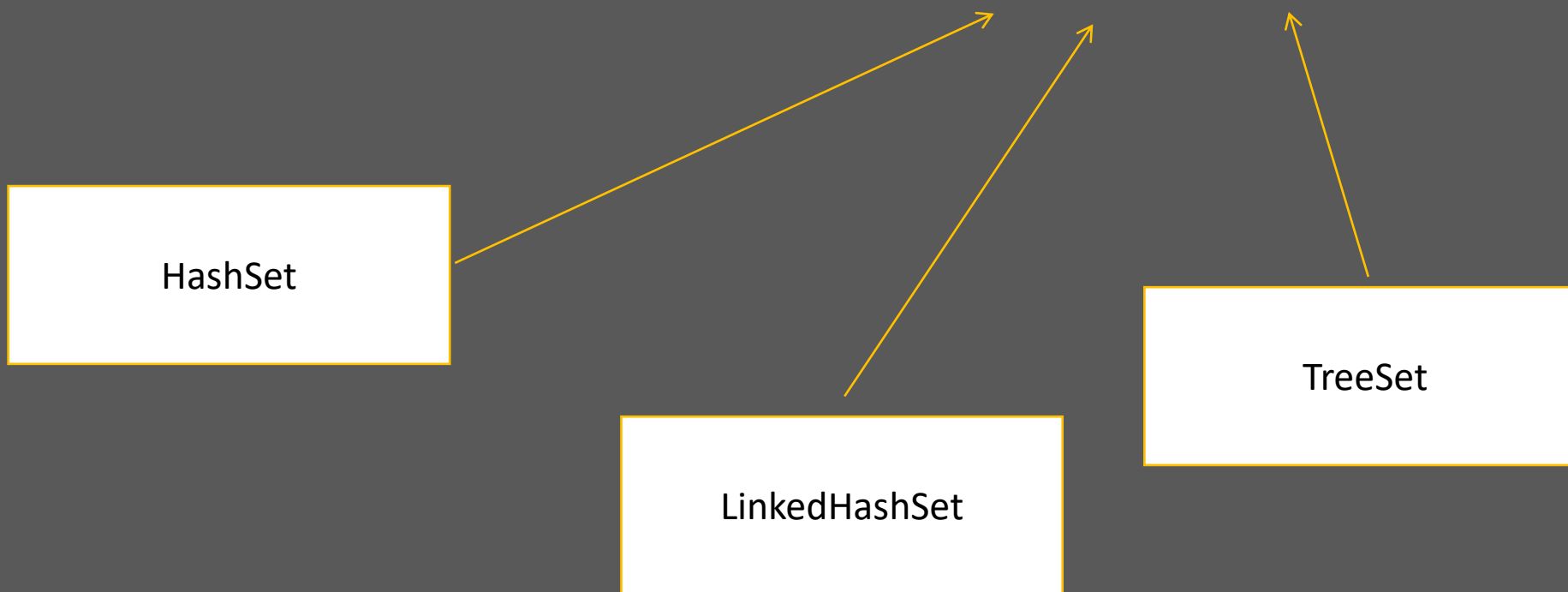
- Sintaxe do set

```
Set <Classe_Tipo> nome_da_coleção = new  
tipo_objeto_coleção<Classe>();
```

# Collection - Set

- Sintaxe do set

```
Set <Classe_Tipo> nome_da_coleção = new tipo_objeto_coleção<>();
```



# Collection - Set

```
public static void main(String[] args)
{
    String [] cores = {"verde", "amarelo", "azul", "branco", "azul", "amarelo", "verde"};
    List <String> lista = Arrays.asList(cores);
    Set <String> lista2 = new HashSet<String>(lista);
    System.out.println(lista);
    System.out.println(lista2);
}
```

# Saída:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ AulaCollection ---  
[branco, azul, verde, vermelho, amarelo, amarelo, verde]  
[vermelho, amarelo, branco, verde, azul]  
-----
```

```
- import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.HashSet;
import java.util.Iterator;

public class Colecoes {
    public static void main(String[] args)
    {
        String [] cores = {"verde", "amarelo", "azul", "branco", "azul", "amarelo", "verde"};
        List <String> lista = Arrays.asList(cores);
        Set <String> lista2 = new HashSet<String>(lista);
        System.out.println(lista);
        System.out.println(lista2);
        Iterator valor = lista2.iterator();
        System.out.println("O valor de interação é>:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```

```
import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.HashSet;
import java.util.Iterator;
```

Interfator Iterator.

```
public class Colecoes {
    public static void main(String[] args)
    {
        String [] cores = {"verde", "amarelo", "azul", "branco", "azul", "amarelo", "verde"};
        List <String> lista = Arrays.asList(cores);
        Set <String> lista2 = new HashSet<String>(lista);
        System.out.println(lista);
        System.out.println(lista2);
        Iterator valor = lista2.iterator();
        System.out.println("O valor de interação é>:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```

```
import java.util.Arrays;  
import java.util.List;  
import java.util.Set;  
import java.util.HashSet;  
import java.util.Iterator;
```

```
public class Colecoes {  
    public static void main(String[] args)  
    {  
        String [] cores = {"verde", "amarelo", "azul", "branco", "azul", "amarelo", "verde"};  
        List <String> lista = Arrays.asList(cores);  
        Set <String> lista2 = new HashSet<String>(lista);  
        System.out.println(lista);  
        System.out.println(lista2);  
        Iterator valor = lista2.iterator();  
        System.out.println("O valor de interação é>:");  
        while (valor.hasNext()) {  
            System.out.println(valor.next());  
        }  
    }  
}
```

Retorna a quantidade  
de iterações de um  
coleção

```
import java.util.Arrays;  
import java.util.List;  
import java.util.Set;  
import java.util.HashSet;  
import java.util.Iterator;
```

```
public class Colecoes {  
    public static void main(String[] args)  
    {  
        String [] cores = {"verde", "amarelo", "azul", "branco", "azul", "amarelo", "verde"};  
        List <String> lista = Arrays.asList(cores);  
        Set <String> lista2 = new HashSet<String>(lista);  
        System.out.println(lista);  
        System.out.println(lista2);  
        Iterator valor = lista2.iterator();  
        System.out.println("O valor de interação é>:");  
        while (valor.hasNext()) {  
            System.out.println(valor.next());  
        }  
    }  
}
```

Verifica se existe um  
próximo elemento em  
valor



```
import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.HashSet;
import java.util.Iterator;
```

```
public class Colecoes {
    public static void main(String[] args)
    {
        String [] cores = {"verde", "amarelo", "azul", "branco", "azul", "amarelo", "verde"};
        List <String> lista = Arrays.asList(cores);
        Set <String> lista2 = new HashSet<String>(lista);
        System.out.println(lista);
        System.out.println(lista2);
        Iterator valor = lista2.iterator();
        System.out.println("O valor de interação é>:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```

Retorna o próximo  
elemento na iteração

# Set - TreeSet

- **TreeSet**
  - Fornece uma implementação da interface Set que usa uma árvore para armazenamento.
  - Os objetos são armazenados em uma ordem ordenada e ascendente.
  -

# TreeSet - Métodos

## **void add(Object o)**

Adiciona o elemento especificado a este conjunto se ele ainda não estiver presente.

## **boolean addAll(Collection c)**

Adiciona todos os elementos da coleção especificada a este conjunto.

## **void clear()**

Remove todos os elementos deste conjunto.

## **Object clone()**

Retorna uma cópia superficial desta instância TreeSet.

## **Comparator comparator()**

Retorna o comparador usado para ordenar este conjunto ordenado, ou nulo se este conjunto de árvore usar seus elementos de ordenação natural.

# TreeSet - Métodos

## **boolean contains(Object o)**

Retorna verdadeiro se este conjunto contiver o elemento especificado.

## **Object first()**

Retorna o primeiro elemento (mais baixo) atualmente neste conjunto ordenado.

## **SortedSet headSet(Object toElement)**

Retorna uma visão da parte deste conjunto cujos elementos são estritamente inferiores ao elemento.

## **boolean isEmpty()**

Retorna verdadeiro se este conjunto não contiver elementos.

## **Iterator iterator()**

Retorna um iterador sobre os elementos neste conjunto

# TreeSet - Métodos

## **object last()**

Retorna o último (mais alto) elemento atualmente neste conjunto ordenado.

## **boolean remove (Objeto o)**

Remove o elemento especificado deste conjunto se estiver presente

## **int size()**

Retorna o número de elementos neste conjunto (sua cardinalidade).

## **SortedSet subSet(Object fromElement, Object toElement)**

Retorna uma visão da parte deste conjunto cujos elementos variam de Reino, inclusive, até ToElement, exclusivo.

## **SortedSet tailSet(Object fromElement)**

Retorna uma visão da parte deste conjunto cujos elementos são maiores ou iguais a de partir de Reino.

```
- import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.Iterator;
import java.util.TreeSet;

public class Colecoes {
    public static void main(String[] args)
    {
        Integer [] numeros = {12, 63, 35, 45};
        List <Integer> lista = Arrays.asList(numeros);
        Set <Integer> lista2 = new TreeSet<Integer>(lista);
        Iterator valor = lista2.iterator();
        System.out.println("Conjunto de Dados tipo árvore:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```

```
- import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.Iterator;
import java.util TreeSet;

public class Colecoes {
    public static void main(String[] args)
    {
        Integer [] numeros = {12, 63, 35, 45};
        List <Integer> lista = Arrays.asList(numeros);
        Set <Integer> lista2 = new TreeSet<Integer>(lista);
        Iterator valor = lista2.iterator();
        System.out.println("Conjunto de Dados tipo árvore:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```

Utilizando a classe  
Integer

```
import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.Iterator;
import java.util.TreeSet;

public class Colecoes {
    public static void main(String[] args)
    {
        Integer [] numeros = {12, 63, 35, 45};
        List<Integer> lista = Arrays.asList(numeros);
        Set<Integer> lista2 = new TreeSet<Integer>(lista);
        Iterator valor = lista2.iterator();
        System.out.println("Conjunto de Dados tipo árvore:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```



```
- import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.Iterator;
import java.util.TreeSet;

public class Colecoes {
    public static void main(String[] args)
    {
        Integer [] numeros = {12, 63, 35, 45};
        List <Integer> lista = Arrays.asList(numeros);
        Set <Integer> lista2 = new TreeSet<Integer>(lista);
        Iterator valor = lista2.iterator();
        System.out.println("Conjunto de Dados tipo árvore:");
        while (valor.hasNext()) {
            System.out.println(valor.next());
        }
    }
}
```

# Saída:

Conjunto de Dados tipo árvore:

12

35

45

63

---

**BUILD SUCCESS**

---

# Set - LinkedHashSet

- **LinkedHashSet**
  - É um meio termo entre HashSet e TreeSet,
  - O LinkedHashSet faz uso também do HashTable com linked list.
    - Os elementos continuam na ordem que são inseridos, diferente do HashSet que “embaralha” tudo.

```
package colecao.aulacollection;
```

```
import java.util.Iterator;  
import java.util.LinkedHashSet;
```

```
public class Colecoes {  
    public static void main(String[] args)  
    {  
        LinkedHashSet <String> linguagens = new LinkedHashSet<String>();  
        linguagens.add("java");  
        linguagens.add("python");  
        linguagens.add("c#");  
        linguagens.add("prolog");  
        linguagens.add("php");  
        System.out.println(linguagens);  
    }  
}
```

# Saída:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ AulaCollection ---  
[java, python, c#, prolog, php]
```

---

```
import java.util.Iterator;
import java.util.LinkedHashSet;

public class Colecoes {
    public static void main(String[] args)
    {
        LinkedHashSet <String> linguagens = new LinkedHashSet<String>();
        linguagens.add("java");
        linguagens.add("python");
        linguagens.add("c#");
        linguagens.add("prolog");
        linguagens.add("php");
        Iterator iterator = linguagens.iterator();
        while(iterator.hasNext())
        {
            System.out.println(iterator.next() + " ");
        }
    }
}
```

# Saída:

```
[...]
--- exec-maven-plugin:1.5.0:exec (default-cli) @ AulaCollection ---
java
python
c#
prolog
php
```

# Teste de Performance no TreeSet, HashSet e LinkedHashSet

```
import java.util.LinkedHashSet;
import java.util.Random;
import java.util.TreeSet;
import java.util.HashSet;

public class Colecoes {
    public static void main(String[] args)
    {
        Random r = new Random();

        HashSet<Integer> hashSet = new HashSet<>();
        TreeSet<Integer> treeSet = new TreeSet<>();
        LinkedHashSet<Integer> linkedSet = new LinkedHashSet<Integer>();
    }
}
```



# Teste de Performance no TreeSet, HashSet e LinkedHashSet

```
// Inicio Tempo hashSet
long startTime = System.nanoTime();

for (int i = 0; i < 1000; i++) {
    int x = r.nextInt(1000 - 10) + 10;
    hashSet.add(i);
}

// Fim Tempo hashSet

long endTime = System.nanoTime();
long duracao = endTime - startTime;
System.out.println("HashSet: " + duracao);
```

# Teste de Performance no TreeSet, HashSet e LinkedHashSet

```
// Inicio Tempo treeSet
    startTime = System.nanoTime();

    for (int i = 0; i < 1000; i++) {
        int x = r.nextInt(1000 - 10) + 10;
        treeSet.add(i);
    }
// Fim Tempo treeSet
endTime = System.nanoTime();
duracao = endTime - startTime;
System.out.println("TreeSet: " + duracao);
```

# Teste de Performance no TreeSet, HashSet e LinkedHashSet

```
// Inicio Tempo linkedSet
startTime = System.nanoTime();

for (int i = 0; i < 1000; i++) {
    int x = r.nextInt(1000 - 10) + 10;
    linkedSet.add(i);
}
// Fim Tempo linkedSet
endTime = System.nanoTime();
duracao = endTime - startTime;
System.out.println("LinkedHashSet: " + duracao);
}
}
```

# Saída:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ AulaCollection ---  
HashSet: 2700200  
TreeSet: 9157100  
LinkedHashSet: 3859400  
-----
```

# Queue

- A interface Queue está disponível no pacote `java.util` e amplia a interface `Collection`.
- Implementa uma lista ordenada de objetos com seu uso limitado para inserir elementos no final da lista e excluir elementos desde o início da lista, ou seja, segue o FIFO ou o princípio First-In-First-Out.

# Queue - Métodos

**add()**- Este método é usado para adicionar elementos na cauda da fila. Mais especificamente, no último da lista de vinculados, se for usado, ou de acordo com a prioridade em caso de implementação de fila prioritária.

**peek()**- Este método é usado para visualizar a cabeça da fila sem removê-la. Ele retorna Null se a fila estiver vazia.

**element()**- Este método é semelhante ao peek(). Ele joga NoSuchElementException quando a fila está vazia.

**remove()**- Este método remove e devolve a cabeça da fila. Ele joga NoSuchElementException quando a fila é impty.

**poll()**- Este método remove e devolve a cabeça da fila. Ele retorna nulo se a fila estiver vazia.

**size()**- Este método devolve o não. de elementos na fila.

```
[- import java.util.Queue;
  import java.util.LinkedList;

  public class ColecaoQueue {

    public static void main (String[] args)
    {
      Queue <String> fila = new LinkedList<>();
      fila.add("Ricardo");
      fila.add("Beatriz");
      fila.add("José");
      fila.add("Wagner");
      System.out.println(fila);
    }
  }
```

# Saída:

```
-----[jar]-----  
[ ] --- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
  [ ] [Ricardo, Beatriz, José, Wagner]  
-----  
BUILD SUCCESS
```



```
package colecao.colecao;
```

```
import java.util.Queue;  
import java.util.LinkedList;
```

```
public class ColecaoQueue {
```

```
    public static void main (String[] args)
```

```
    {
```

```
        Queue <String> fila = new LinkedList<>();
```

```
        fila.add("Ricardo");
```

```
        fila.add("Beatriz");
```

```
        fila.add("José");
```

```
        fila.add("Wagner");
```

```
        System.out.println(fila);
```

```
        fila.add("Angelo");
```

```
        System.out.println("Inserindo Angelo no Fim da Fila:" + fila);
```

```
    }
```

```
}
```

# Saída:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
[Ricardo, Beatriz, José, Wagner]  
Inserindo Angelo no Fim da Fila:[Ricardo, Beatriz, José, Wagner, Angelo]
```

---

**BUILD SUCCESS**

---

```
package colecao.colecao;
```

```
import java.util.Queue;  
import java.util.LinkedList;
```

```
public class ColecaoQueue {
```

```
    public static void main (String[] args)
```

```
    {
```

```
        Queue <String> fila = new LinkedList<>();
```

```
        fila.add("Ricardo");
```

```
        fila.add("Beatriz");
```

```
        fila.add("José");
```

```
        fila.add("Wagner");
```

```
        System.out.println(fila);
```

```
        fila.add("Angelo");
```

```
        System.out.println("Inserindo Angelo no Fim da Fila:" + fila);
```

```
        System.out.println("Primeiro elemento da fila:" + fila.peek());
```

```
    }
```

```
}
```

# Saída:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
[Ricardo, Beatriz, José, Wagner]  
Inserindo Angelo no Fim da Fila:[Ricardo, Beatriz, José, Wagner, Angelo]  
Primeiro elemento da fila:Ricardo
```

---

```
import java.util.Queue;  
import java.util.LinkedList;
```

```
public class ColecaoQueue {
```

```
    public static void main (String[] args)
```

```
    {
```

```
        Queue <String> fila = new LinkedList<>();
```

```
        fila.add("Ricardo");
```

```
        fila.add("Beatriz");
```

```
        fila.add("José");
```

```
        fila.add("Wagner");
```

```
        System.out.println(fila);
```

```
        fila.add("Angelo");
```

```
        System.out.println("Inserindo Angelo no Fim da Fila:" + fila);
```

```
        System.out.println("Primeiro elemento da fila:" + fila.peek());
```

```
        System.out.println("Remove e Retorna o elemento do inicio da fila: " + fila.poll());
```

```
        System.out.println( fila);
```

```
    }
```

```
}
```

# Sáida:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
[Ricardo, Beatriz, José, Wagner]  
Inserindo Angelo no Fim da Fila:[Ricardo, Beatriz, José, Wagner, Angelo]  
Primeiro elemento da fila:Ricardo  
Remove e Retorna o elemento do inicio da fila: Ricardo  
[Beatriz, José, Wagner, Angelo]  
-----
```

```
import java.util.Queue;  
import java.util.LinkedList;
```

```
public class ColecaoQueue {
```

```
    public static void main (String[] args)
```

```
    {
```

```
        Queue <String> fila = new LinkedList<>();
```

```
        fila.add("Ricardo");
```

```
        fila.add("Beatriz");
```

```
        fila.add("José");
```

```
        fila.add("Wagner");
```

```
        System.out.println(fila);
```

```
        fila.add("Angelo");
```

```
        System.out.println("Inserindo Angelo no Fim da Fila:" + fila);
```

```
        System.out.println("Primeiro elemento da fila:" + fila.peek());
```

```
        System.out.println("Remove e Retorna o elemento do inicio da fila: " + fila.poll());
```

```
        System.out.println( fila);
```

```
        System.out.println("Remove e Retorna o elemento do inicio da fila: " + fila.remove());
```

```
        System.out.println( fila);
```

```
    }
```

```
}
```

# Saída:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
[Ricardo, Beatriz, José, Wagner]  
Inserindo Angelo no Fim da Fila:[Ricardo, Beatriz, José, Wagner, Angelo]  
Primeiro elemento da fila:Ricardo  
Remove e Retorna o elemento do inicio da fila: Ricardo  
[Beatriz, José, Wagner, Angelo]  
Remove e Retorna o elemento do inicio da fila: Beatriz  
[José, Wagner, Angelo]
```



# Remove() vs. Poll()

- A diferença entre o Remove e o Poll() está que o primeiro apenas na forma de lançar uma exceção se essa fila estiver vazia.
  - NoSuchElementException
- O Poll se a fila estiver vazia retorna NULL.

# Map

- A interface Map mapeia chaves únicas para valores.
- Uma chave é um objeto que você usa para recuperar um valor em uma data posterior.
-

```
package colecao.colecao;
```

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

```
public class ColecaoMap {  
    public static void main (String[] args)  
    {  
        Map <String, String> pais = new HashMap<>();  
        pais.put("BR", "Brasil");  
        pais.put("RU", "Rússia");  
        pais.put("IN", "Índia");  
        pais.put("CN", "China");  
        System.out.println(pais.containsKey("BR"));  
    }  
}
```

containsKey()  
busca uma  
chave  
cadastrado NA  
hash.  
Retornando **True**  
(existe) ou **False**  
(não existe)

[ ]

--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
true

-----

```
package colecao.colecao;

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

public class ColecaoMap {
    public static void main (String[] args)
    {
        Map <String, String> pais = new HashMap<>();
        pais.put("BR", "Brasil");
        pais.put("RU", "Rússia");
        pais.put("IN", "Ín dia");
        pais.put("CN", "China");
        System.out.println(pais.containsKey("US"));
    }
}
```

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
false
```

---

```
public class ColecaoMap {  
    public static void main (String[] args)  
    {  
        Map <String, String> pais = new HashMap<>();  
        pais.put("BR", "Brasil");  
        pais.put("RU", "Rússia");  
        pais.put("IN", "Índia");  
        pais.put("CN", "China");  
        System.out.println(pais.containsKey("US"));  
        System.out.println("A Hash contém Brasil? " + pais.containsValue("Brasil"));  
    }  
}
```

containsValue()

Retorna True se este mapa mapear uma ou mais chaves do valor especificado.

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---  
false  
A Hash contém Braisl? true  
  
-----  
BUILD SUCCESS  
-----  
  
Total time: 1.973 s  
Finished at: 2021-06-02T17:34:02-03:00  
-----
```



```
public static void main (String[] args)
{
    Map <String, String> pais = new HashMap<>();
    pais.put("BR", "Brasil");
    pais.put("RU", "Rússia");
    pais.put("IN", "Ín dia");
    pais.put("CN", "China");
    System.out.println(pais.containsKey("US"));
    System.out.println("A Hash contém Braisl? " + pais.containsValue("Brasil"));
    System.out.println("Qual o conteúdo da chave RU? " + pais.get("RU"));
}
```

## Get(chave)

Retorna o conteúdo de uma chave passada por parâmetro

```
--- exec-maven-plugin:1.5.0.exec (default-cli) @ Colecao
```

```
false
```

```
A Hash contém Braisl? true
```

```
Qual o conteúdo da chave RU? Rússia
```

```
-----
```

```
BUILD SUCCESS
```

```
-----
```

```
public static void main (String[] args)
{
    Map <String, String> pais = new HashMap<>();
    pais.put("BR", "Brasil");
    pais.put("RU", "Rússia");
    pais.put("IN", "Ín dia");
    pais.put("CN", "China");
    System.out.println(pais.containsKey("US"));
    System.out.println("A Hash contém Braisl? " + pais.containsValue("Brasil"));
    System.out.println("Qual o conteúdo da chave RU? " + pais.get("RU"));
    pais.remove("RU");
    System.out.println(pais);
}
```

Remove(chave)

Remove o conteúdo da chave passa como parâmetro.

```
A Hash contém Braisl? true  
Qual o conteúdo da chave RU? Rússia  
{BR=Brasil, IN=Ín dia, CN=China}
```

---

**BUILD SUCCESS**

---

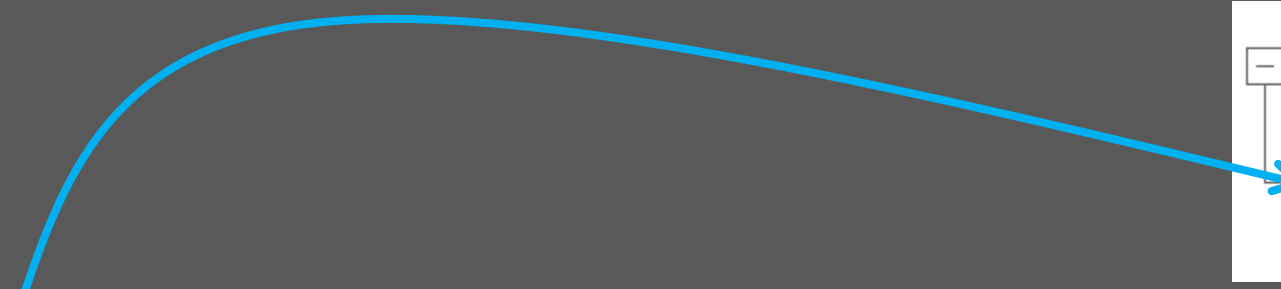
Rússia foi  
removido do  
mapa

# Recuperando as chaves do mapa

```
Set <String> chaves = pais.keySet();  
for(String chave : chaves){  
    System.out.println(chave);}  
  
}
```

# Recuperando as chaves do mapa

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



```
Set<String> chaves = pais.keySet();  
for(String chave : chaves){  
    System.out.println(chave);  
}
```

# Recuperando as chaves do mapa

BR

IN

CN

# Concluindo

**FORAM ABORDADOS NESTA AULA:**

- Collections – parte 2

**ESTE SLIDES ESTÃO BASEADOS NA BIBLIOGRAFIA:**

- DEITEL, Paul; DEITEL, Harvey. Java. Pearson Educação, 2010.

**NA PRÓXIMA AULA:**

- Generics