

AULA 9

LINGUAGEM DE PROGRAMAÇÃO 1

COLLECTION – PARTE 1

Prof. Fábio Modesto

fabiomodesto@ifsp.edu.br

Instituto Federal de Educação, Ciência e Tecnologia
de São Paulo

AULA PASSADA

Array e String

AULA DE HOJE

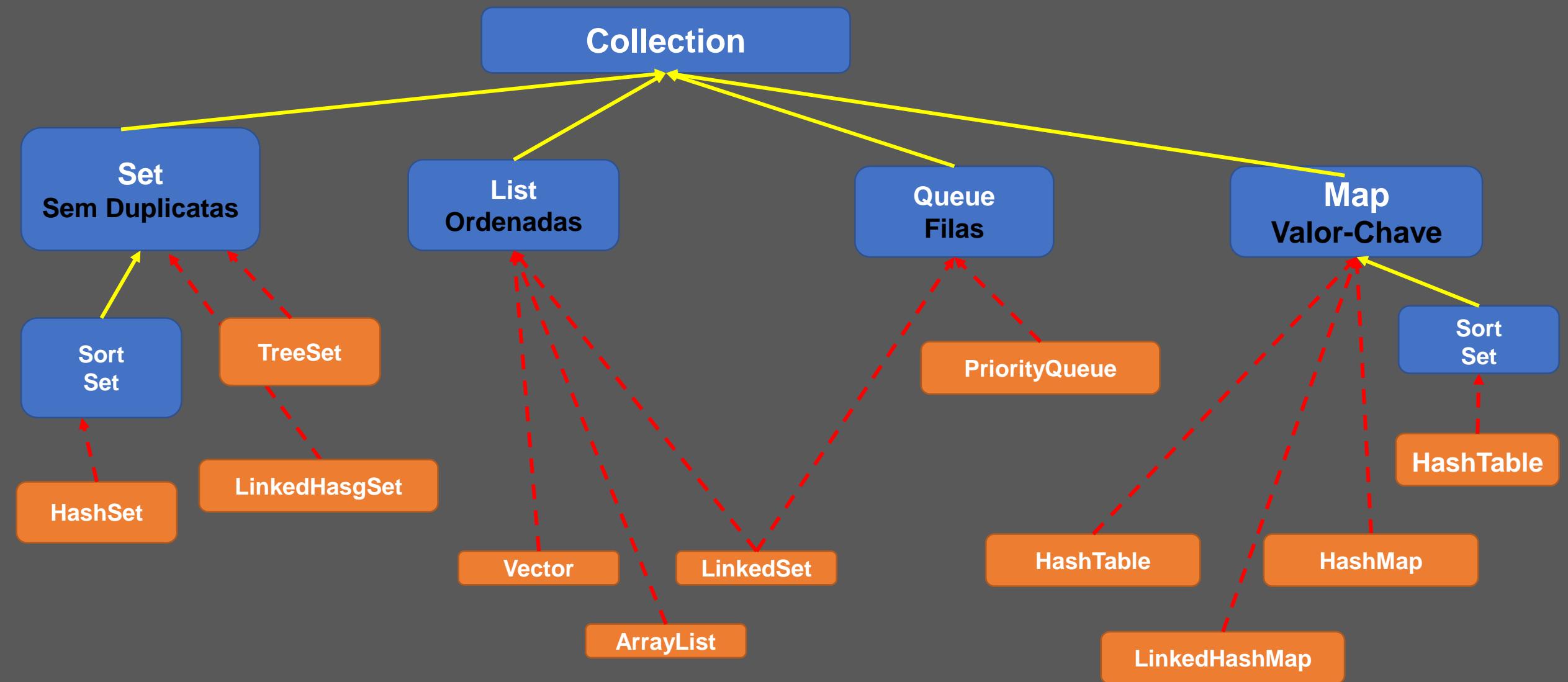
Collection – Parte 1

Agenda

- 1) Introdução de Collection**
- 2) Exemplo de uso do ArrayList**
- 3) Interface e Classes de de Collection**
- 4) Algoritmos de Collection**
- 5) Exemplos práticos do uso métodos gerais e específicos**
 - 1) ArrayList**
 - 2) Collections**
 - 3) List**

Collection

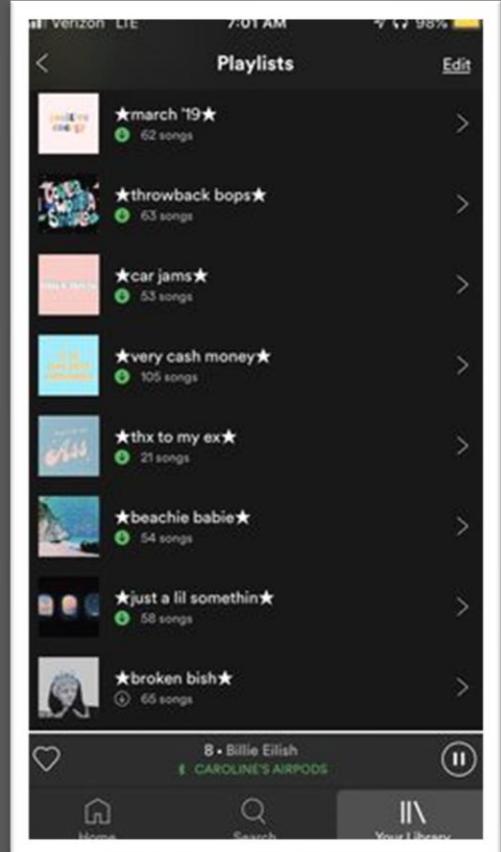
- Uma coleção em Java nada mais é do que um objeto que agrupa vários elementos
- O *framework collections* da API Java oferece
 - Interfaces: definições abstratas de coleções
 - Implementações: objetos concretos de coleções
 - Algoritmos: vários métodos para trabalhar com
 - coleções
 - Ex: busca e ordenação
- O *framework collections* utiliza *generics*



Array

Collection

Exemplos de coleções



Collection

- Collection
 - Raíz da hierarquia de coleções
 - Não determina nenhuma restrição para coleções (ordem, duplicação de elementos, etc)
 - Na API, nenhuma classe concreta implementa diretamente esta interface

Interface Collection

- Set
 - Coleção que não permite elementos duplicados
 - Representa a definição de conjuntos em Matemática
- SortedSet
 - Igual a Set
 - Porém, mantém os elementos ordenados (ascendente)
- List
 - Coleção ordenada de elementos (sequências)
 - Pode conter elementos duplicados
 - Há controle preciso sobre a posição de cada elemento (indexação)

‘Interface Collection

- **Queue**
 - Coleção que representa uma fila genérica
 - Em geral, utiliza a regra FIFO
- **Deque**
 - Semelhante a Queue
 - Porém, inserções e remoções podem ser feitas em
 - qualquer extremidade
 - Também pode ser usada para representar uma pilha (LIFO)

Interface Collection

- Map
 - Coleção de pares chave-valor
 - Cada chave mapeia apenas um valor
 - Chaves não podem ser duplicadas
 - Representa a definição de função em Matemática
- SortedMap
 - Igual a Map
 - Porém, mantém os elementos ordenados pela chave, em ordem ascendente
 - Ex: dicionário, lista telefônica

Classes Concretas *Collections*

- Set
 - HashSet
 - Armazena os elementos em uma tabela hash
 - Melhor performance
 - Não garante nenhum tipo de ordem dos elementos
 - TreeSet
 - Usa uma estrutura de árvore para armazenar os elementos
 - Mantém a ordem dos elementos
 - Bem mais lenta que a HashSet
 - LinkedHashSet
 - Usa tabela hash com lista ligada
 - Mantém a ordem de inserção dos elementos
 - Um pouco menos eficiente que HashSet

Classes Concretas *Collections*

- List
 - ArrayList
 - Implementação diretamente indexada de elementos
 - (arrays)
 - De maneira geral, implementação mais eficiente
 - LinkedList
 - Usa lista duplamente encadeada para armazenar os elementos
 - Acesso aleatório (indexado) exige percorrer a lista

Classes Concretas *Collections*

- **Queue**
 - **add** e **offer** inserem um elemento na fila
 - **remove** e **poll** removem e retornam o elemento do
 - **início**
 - **element** e **peek** retornam, mas não removem, o elemento do **início**
- **LinkedList**
- **PriorityQueue**
 - Prioridade é determinada pelo próprio valor do elemento
 - Elementos precisam ser do tipo **Comparable**

Classes Concretas *Collections*

- Deque
 - Inserção
 - addFirst, offerFirst, addLast, offerLast
 - Remoção
 - removeFirst, pollFirst, removeLast, pollLast
 - ArrayDeque: array
 - LinkedList: lista ligada
- Note que LinkedList implementa as interfaces List, Queue e Deque

Classes Concretas *Collections*

- **Map**
 - **HashMap:** tabela hash
 - **TreeMap:** árvore
 - **LinkedHashMap:** tabela hash e lista ligada
 - Comportamento e performance similar ao visto para Set
- Por armazenar um par chave-valor, há dois parâmetros genéricos

Algoritmos Sobre Collections

- A classe Collections provê alguns métodos estáticos para manipular coleções
- Ordenação
 - <T extends Comparable<? super T>> void sort(List<T>)
 - Note que T precisa ser do tipo Comparable
 - <T> void sort(List<T>, Comparator<? super T>)
 - Usa o objeto Comparator passado para ordenar
 - Permite ordenar utilizando um critério diferente do que foi definido por ordem natural (Comparable)
- Utiliza uma versão otimizada do merge sort
 - Garante performance $n \log n$ e estabilidade

Algoritmos Sobre Collections

- Embaralhamento (shuffling)
 - `void shuffle(List<T>)`
- Métodos de propósito geral
 - **reverse**: inverte os elementos de uma lista
 - **fill**: substitui todos os elementos de uma lista
 - **copy**: copia os elementos de uma lista para outra lista (sobrescreve)
 - **swap**: troca a posição de dois elementos de uma lista
 - **addAll**: adiciona elementos em lote em uma coleção

Algoritmos Sobre Collections

- Busca binária
 - <T> int **binarySearch(List<? extends Comparable<?**
 - **super T>> list, T key)**
 - Lembre-se que a lista precisa estar ordenada para aplicar busca binária
 - Retorna a posição do elemento na lista ou o negativo de onde o elemento deveria estar

Algoritmos Sobre *Collections*

- Composição
 - int frequency(Collection<?> c, Object o)
 - Quantidade de elementos iguais ao que foi passado
 - boolean disjoint(Collection<?> c1, Collection<?> c2)
 - Retorna true se as coleções não tem elementos em comum
- Máximo e mínimo
 - Métodos que determinam o maior e menor elemento em uma coleção
 - Possuem duas versões: uma considera a ordem natural (**Comparable**) e outra uma ordem específica (**Comparator**)

Algoritmos Sobre Collections

- Classe Arrays
 - static <T> List<T> **asList**(T... a)
 - Retorna uma **List** (view) do array passado
 - Não é possível adicionar ou remover elementos
 - Mudanças no array (lista), afeta a lista (array)

Exemplo usando ArrayList

- Antes de começarmos falar de coleções propriamente dita vamos fazer um exemplo de aquecimento usando `ArrayList`.

```
public class ArraySimples {  
    public static void main(String[] args)  
    {  
        String[] nipes={"ouros", "paus", "copas", "espadas"};  
        System.out.println(nipes[0]);  
    }  
}
```

O código mostra a criação de um vetor simples de String,
como visto na aula passada.

Agora e se fizermos essa mudança....

```
public class ArraySimples {  
    public static void main(String[] args)  
    {  
        String[] nipes={"ouros", "paus", "copas", "espadas"};  
        nipes[10]= "sabre de luz";  
        System.out.println(nipes[10]);  
    }  
}
```

O tamanho já havia sido definido no vetor!!!

```
--- exec-maven-plugin:1.5.0:exec (default cli) @ Colecao  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10  
        at colecao.colecao.ArraySimples.main(ArraySimples.java:8)  
Command execution failed.  
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)  
        at org.apache.commons.exec.DefaultExecutor.executeInternal (DefaultExecutor.java:404)  
        at org.apache.commons.exec.DefaultExecutor.execute (DefaultExecutor.java:166)
```

Como resolver?

```
import java.util.ArrayList;

public class ArraySimples {
    public static void main(String[] args)
    {
        ArrayList <String> nipes = new ArrayList();
        nipes.add("ouros");
        nipes.add("copas");
        nipes.add("paus");
        nipes.add("espadas");
        nipes.add("sabre de luz");
        System.out.println(nipes.toString());
    }
}
```

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---
[ouros, copas, paus, espadas, sabre de luz]
BUILD SUCCESS
-----
Total time: 3.598 s
Finished at: 2021-05-27T09:35:10-03:00
-----
```

```
nipes.add("paus");
nipes.add("espadas");
nipes.add("sabre de luz");
System.out.println(nipes.toString());
System.out.println(nipes.size());
}

}
```

O método **size()**
retorna o tamanho
do **ArrayList**

```
[ouros, copas, paus, espadas, sabre de luz]
5
```

```
BUILD SUCCESS
```

```
nipes.add("espadas");
nipes.add("sabre de luz");
System.out.println(nipes.toString());
System.out.println(nipes.size());
    System.out.println("Imprimindo o elemento com indice 3: " + nipes.get(3));
}

}
```

O método `get()`
retorna o elemento
que se encontra
em dada posição

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---
[ouros, copas, paus, espadas, sabre de luz]
5
Imprimindo o elemento com indice 3: espadas
-----
BUILD SUCCESS
```

```
System.out.println("Imprimindo o elemento com indice 3: " + nipes.get(3));
nipes.remove("paus");
System.out.println(nipes.toString());
}
```

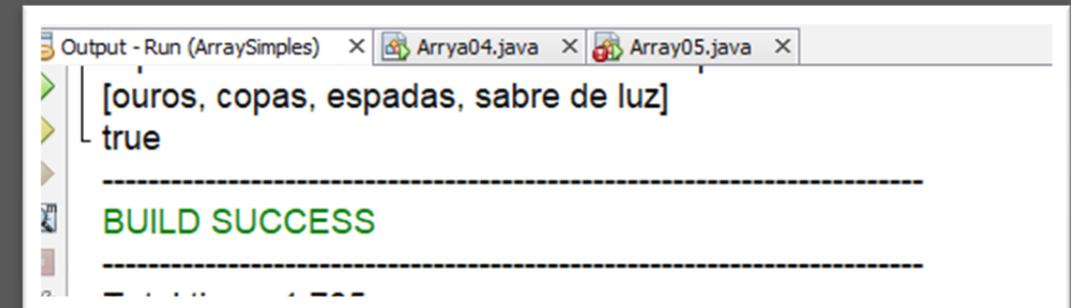
O método
remove(elemento)
remove o
elemento passado
do ArrayList

```
Bauding Colecao 1.0-SNAPSHOT
----- [ jar ] -----
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---
[ouros, copas, paus, espadas, sabre de luz]
5
Imprimindo o elemento com indice 3: espadas
[ouros, copas, espadas, sabre de luz]
```

```
System.out.println(nipes.toString());
System.out.println(nipes.size());
System.out.println("Imprimindo o elemento com indice 3: " + nipes.get(3));
nipes.remove("paus");
System.out.println(nipes.toString());
System.out.println(nipes.contains("sabre de luz"));
}

}
```

O método **contains(elemento)** busca o elemento passado no **ArrayList** e retorna um valor boolean: **true(encontrou)** e **false (não encontrou)**



Coleção

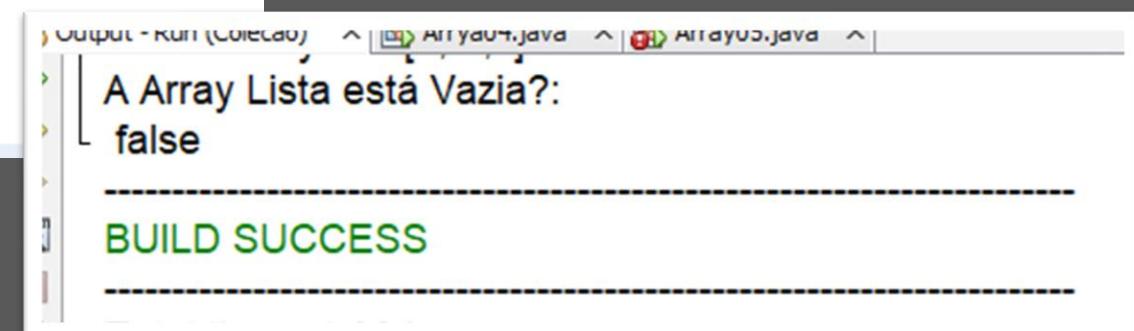
```
[-] import java.util.ArrayList;
[-] import java.util.Collection;

public class Colecao {
    public static void main(String[] args)
    {
        Collection <String> c = new ArrayList();
        c.add("a");
        c.add("e");
        c.add("i");
        System.out.println("minha ArrayList: "+c.toString());
    }
}
```

```
[-] --- exec-maven-plugin:1.5.0:exec (default-cli) @ Colecao ---
[-] minha ArrayList: [a, e, i]
-----
[BUILD SUCCESS]
-----
```

```
import java.util.ArrayList;
import java.util.Collection;

public class Colecao {
    public static void main(String[] args)
    {
        Collection <String> c = new ArrayList();
        c.add("a");
        c.add("e");
        c.add("i");
        System.out.println("minha ArrayList: "+c.toString());
        System.out.println("A Array Lista está Vazia?:\n' "+c.isEmpty());
    }
}
```



O método isEmpty verifica se o ArrayList é vazio, retornando true ou false

```
import java.util.ArrayList;
import java.util.Collection;

public class Colecao {
    public static void main(String[] args)
    {
        Collection <String> c = new ArrayList();
        c.add("a");
        c.add("e");
        c.add("i");
        System.out.println("minha ArrayList: "+c.toString());
        System.out.println("A Array Lista está Vazia?:\n' "+c.isEmpty());
        System.out.println("Existe o elemento a nessa coleção?:\n' "+c.contains("a"));
    }
}
```

Também pode ser usado o método **contains()**

```
Existe o elemento a nessa coleção?:  
' true  
-----  
BUILD SUCCESS  
-----  
Total time: 2.971 s  
Finished at: 2021-05-27T10:11:21-03:00  
-----
```

```
import java.util.ArrayList;
import java.util.Collection;

public class Colecao {
    public static void main(String[] args)
    {
        Collection<String> c = new ArrayList();
        c.add("a");
        c.add("e");
        c.add("i");
        System.out.println("minha ArrayList: "+c.toString());
        System.out.println("A Array Lista está Vazia?:\n' "+c.isEmpty());
        System.out.println("Existe o elemento a nessa coleção?:\n' "+c.contains("a"));
        c.remove("a");
        System.out.println("A coleção depois do remove: "+c.toString());
    }
}
```

Método `remove()` remove um elemento da coleção

A coleção depois do remove: [e, i]

Collection

- Esse métodos são os métodos fundamentais para se trabalhar com as Coleções
- Agora veremos como adicionar grupos de elementos na Coleção

```
System.out.println("minha ArrayList: "+c.toString());
System.out.println("A Array Lista está Vazia?:\n' "+c.isEmpty());
System.out.println("Existe o elemento a nessa coleção?:\n' "+c.contains("a"));
c.remove("a");
System.out.println("A coleção depois do remove: "+c.toString());

=====grupos=====
Collection<String> c2 = Arrays.asList("o", "u");
c.addAll(c2);
System.out.println("Nova Coleção: "+c.toString());

}
}
```

Arrays.asList() é uma classe utilitária de Arrays, que passa um argumento como lista

```
System.out.println("minha ArrayList: "+c.toString());
System.out.println("A Array Lista está Vazia?:\n' "+c.isEmpty());
System.out.println("Existe o elemento a nessa coleção?:\n' "+c.contains("a"));
c.remove("a");
System.out.println("A coleção depois do remove: "+c.toString());

=====grupos=====
Collection<String> c2 = Arrays.asList("o", "u");
c.addAll(c2);
System.out.println("Nova Coleção: "+c.toString());
}

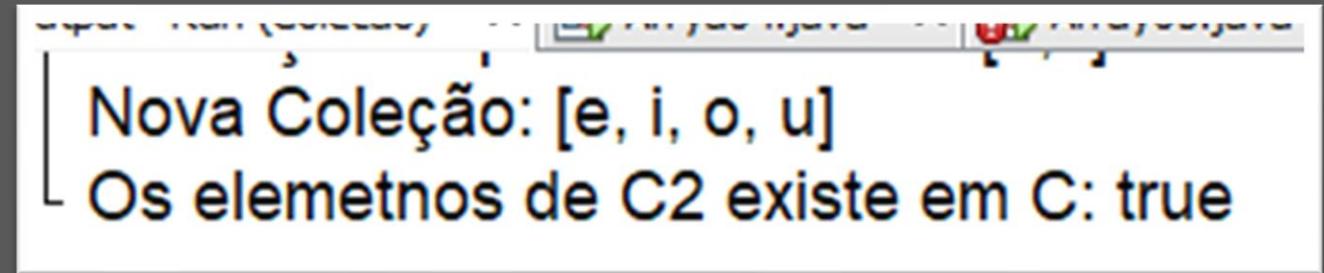
}
```

A coleção depois do remove: [e, i]
Nova Coleção: [e, i, o, u]

BUILD SUCCESS

**addAll(), agrega o conteúdo da Coleção C2
em C.**

```
21 //=====grupos=====
22 Collection<String> c2 = Arrays.asList("o", "u");
23 c.addAll(c2);
24 System.out.println("Nova Coleção: "+c.toString());
25 System.out.println("Os elemetnos de C2 existe em C: "+c.containsAll(c2));
26 }
27 }
```



O método **containsAll()** irá verificar se conteúdo da coleção C2(o e u) estão presentes na Coleção C

Também podemos remover o conteúdo de uma coleção n que foi inserida em uma coleção M.

```
//=====grupos=====
Collection<String> c2 = Arrays.asList("o", "u");
c.addAll(c2);
System.out.println("Nova Coleção: "+c.toString());
System.out.println("Os elemetnos de C2 existe em C: "+ c.containsAll(c2));
c.removeAll(c2);
System.out.println("Nova Coleção depois do removeALL(C2): "+c.toString());
}
```

Para tal usamos o métodos
removeAll

Os elememtos de C2 existe em C. true
Nova Coleção depois do removeALL(C2): [e, i]

```
c.removeAll(c2);
System.out.println("Nova Coleção depois do removeAll(C2): "+c.toString())

=====percorrer os elementos de uma coleção=====
for(String string: c){
    System.out.println(string);
}
```

Nova Coleção depois do removeAll(C2): [e, i]
e
i

Convertendo uma coleção em um Array

```
}

//==Converntendo coleção em Array=====
String [] s = c.toArray(new String[c.size()]);
System.out.println("Array convertido: " + Arrays.toString(s))

}
```

toArray retorna todos os
elementos da coleção no formato
de Array

Convertendo uma coleção em um Array

```
}

//==Converntendo coleção em Array=====
String [] s = c.toArray(new String[c.size()]);
System.out.println("Array convertido: " + Arrays.toString(s))

}
```

Array é convertido para um string
para exibição

Tirando a Prova

```
for(int i=0; i<s.length; i++)  
    System.out.println("elemento["+i+"] = "+ s[i]);  
  
}
```

elemento[0] = e
elemento[1] = i

Limpando toda a Coleção

```
c.clear();
System.out.println(c);

}
}
```

Depois da chamada do método **clear()**, a coleção c está limpa de todos seus elementos


```
[-] import java.util.ArrayList;
[-] import java.util.List;

public class ColecaoLista {
    public static void main(String[] args)
    {
        List<String> lista = new ArrayList();
        lista.add("futebol");
        lista.add("basquete");
        lista.add("tenis");
        lista.add("volei");
        lista.add("natação");
        lista.add("hickey");
        lista.add("corrida");
        System.out.println(lista);
    }
}
```

Percorrendo elemento List

```
public class ColecaoLista {  
    public static void main(String[] args)  
    {  
        List<String> lista = new ArrayList();  
        lista.add("futebol");  
        lista.add("basquete");  
        lista.add("tenis");  
        lista.add("volei");  
        lista.add("natação");  
        lista.add("hickey");  
        lista.add("corrida");  
        System.out.println(lista);  
        for(int i=0; i<lista.size(); i++)  
        {  
            String esporte = lista.get(i);  
            lista.set(i, esporte.toUpperCase());  
        }  
        System.out.println(lista);  
    }  
}
```

Localizar um dado Elemento na Lista - indexOf

```
for(int i=0; i<lista.size(); i++)
{
    String esporte = lista.get(i);
    lista.set(i, esporte.toUpperCase());
}
System.out.println(lista);
System.out.println("Posição: "+lista.indexOf("CORRIDA"));
}
```

Retornado uma sublinstagem

```
    }
    System.out.println(lista);
    System.out.println("Posição: "+lista.indexOf("CORRIDA"));
    System.out.println(" sublista dos elementos que estão entre a posição 2 e 4-->"+lista.subList(2, 4));
}
}
```

sublista dos elementos que estão entre a posição 2 e 4-->[TENIS, VOLEI]

Interface Collection

```
import java.util.List;
import java.util.ArrayList;
public class ColecaoUtilario {
    public static void main(String[] args)
    {
        List <String> linguagem = new ArrayList();
        linguagem.add("Java");
        linguagem.add("Phyton");
        linguagem.add("PHP");
        linguagem.add("C#");
        linguagem.add("Prolog");
        System.out.println(linguagem);
    }
}
```

[Java, Phyton, PHP, C#, Prolog]

Definimos uma coleção com uma
lista com Linguagens de
Programação

```
import java.util.List;
import java.util.ArrayList;
import java.util.Collections;
public class ColecaoUtilitario {
    public static void main(String[] args)
    {
        List <String> linguagem = new ArrayList();
        linguagem.add("Java");
        linguagem.add("Phyton");
        linguagem.add("PHP");
        linguagem.add("C#");
        linguagem.add("Prolog");
        System.out.println(linguagem);
        Collections.sort(linguagem);
        System.out.println("Lista Ordenada em Ordem Alfabética: "+linguagem);
    }
}
```

[Java, Phyton, PHP, C#, Prolog]

Lista Ordenada em Ordem Alfabética: [C#, Java, PHP, Phyton, Prolog]

Collections.sort() é utilizando para ordenar uma lista em ordem alfabética os da da lista

```
public static void main(String[] args)
{
    List <String> linguagem = new ArrayList();
    linguagem.add("Java");
    linguagem.add("Phyton");
    linguagem.add("PHP");
    linguagem.add("C#");
    linguagem.add("Prolog");
    System.out.println(linguagem);
    Collections.sort(linguagem);
    System.out.println("Lista Ordenada em Ordem Alfabética: "+linguagem);
    Collections.reverse(linguagem);
    System.out.println("Ordem da lista invertida: "+linguagem);
```

Collections.reverse() inverte da ordem da Lista

| Lista Ordenada em Ordem Alfabética: [C#, Java, PHP, Phyton, Prolog]
| Ordem da lista invertida: [Prolog, Phyton, PHP, Java, C#]

```
Collections.reverse(linguagem);
System.out.println("Ordem da lista invertida: "+linguagem);
Collections.shuffle(linguagem);
System.out.println("Lista em uma Ordem Aleatória: "+linguagem);

}
```

Collections.shuffle() ordena a lista em ordem aleatória

Lista em uma Ordem Aleatória: [PHP, Java, Prolog, C#, Phyton]

Existe a possibilidade de Inserção de elementos extras me uma lista

```
System.out.println("Lista em uma Ordem Aleatória: "+linguagem);
Collections.addAll(linguagem, "Lisp", "Fortran", "Assembly");
System.out.println(linguagem);
```

Lista Autalizada:[C#, Phyton, PHP, Java, Prolog, Lisp, Fortran, Assembly]

Observe que a lista voltou para ordem original de
isenção

Utilizando o método Collections.frequency() é possível verificar quantas ocorrências de um dado existem na lista

```
Collections.addAll(linguagem, "Lisp", "Fortran", "Assembly");
System.out.println("Lista Autalizada:" + linguagem);
System.out.println("Na lista a Linguagem Priolog apareceu: " +
    Collections.frequency(linguagem, "Prolog")+" vezes");
```

Na lista a Linguagem Priolog apareceu: 1 vezes

Collections permite que você busque o índice de um elemento chave da lista, utilizando Busca Binária

```
List<String> linguagem = Arrays.asList("Java", "C#");
Collections.sort(linguagem);
int indice = Collections.binarySearch(linguagem, "Java");
System.out.println ("o Indice buscado é: " + indice + " e o elemento recuperado é " + linguagem.get(indice));
```

Para usar a busca binária se deve ordenar a lista primeiro

Collections permite que você busque o índice de um elemento chave da lista, utilizando Busca Binária

```
List<String> linguagem = Arrays.asList("Java", "C#");
Collections.sort(linguagem);
int indice = Collections.binarySearch(linguagem, "Java");
System.out.println ("o Indice buscado é: " + indice + " e o elemento recuperado é " + linguagem.get(indice));
```

Recupera-se o índice do elemento chave buscado e se atribui a variável índice

Collections permite que você busque o índice de um elemento chave da lista, utilizando Busca Binária

```
List<String> linguagem = Arrays.asList("Java", "C#");
Collections.sort(linguagem);
int indice = Collections.binarySearch(linguagem, "Java");
System.out.println ("o Indice buscado é: " + indice + " e o elemento recuperado é " + linguagem.get(indice));
```

o Indice buscado é: 3 e o elemento recuperado é Java

Concluindo

FORAM ABORDADOS NESTA AULA:

- Collection

ESTE SLIDES ESTÃO BASEADOS NA BIBLIOGRAFIA:

- Bibliografia

NA PRÓXIMA AULA:

- Mais Collection

