

Cap 06  
Collections

# Parte 2 – Entendendo Collections Básicas

Linguagem de Programação III  
Prof. Otacílio José Pereira

# Objetivos e Tópicos

- Compreender as collections básicas ou mais tradicionais

## Tópicos

- *Como compreender as collections*
- *List*
  - *Operações básicas*
  - *Aplicações*
  - *Implementações: ArrayList e LinkedList*
  - *Implementações: ArrayList e Vector*
- *Queue*
- *Stack*
- *Compreendendo o iterator*





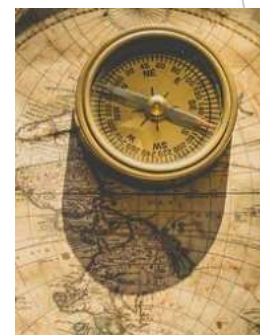
# Contexto

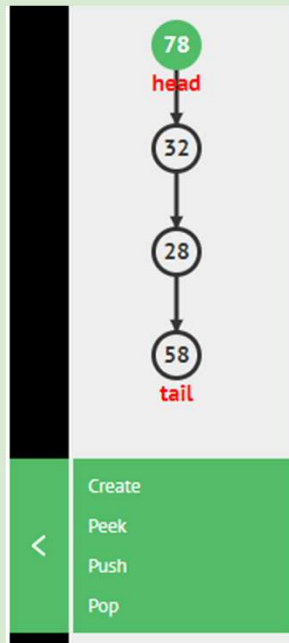
- Onde estamos?

# Onde estamos?

- ▶ *Considerando nosso planejamento inicial*
- ▶ Capítulo 1 – Introdução
- ▶ Capítulo 2 - Conceitos básicos de Orientação a Objetos
  - ▶ OO, classes e objetos
  - ▶ Atributos, métodos, encapsulamento, getters, setters
  - ▶ Outros tópicos: Atributos de classe (static), tipos primitivos, tipos por referência
- ▶ Capítulo 3 – Herança e Polimorfismo
  - ▶ Parte 1 – Herança e breve introdução a Polimorfismo
  - ▶ Parte 2 – Aplicação dos conceitos em um exemplo de UI
- ▶ Capítulo 4 – Classes abstratas e interfaces
- ▶ Capítulo 5 – Collections
- ▶ Capítulo 6 – Outros tópicos

Foco





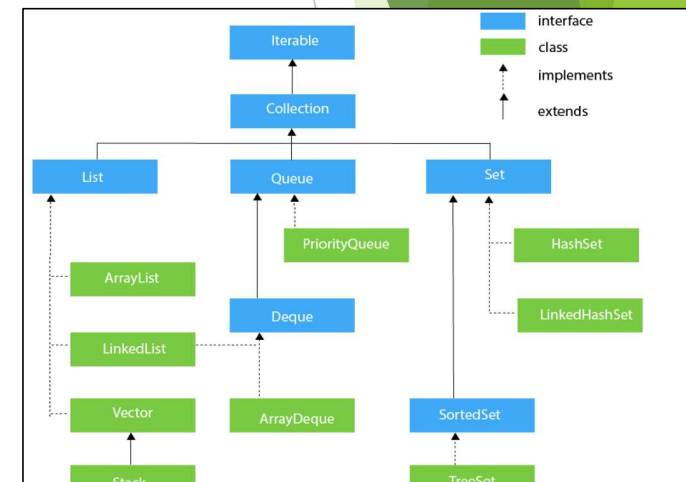
# Como compreender as collections?

- Tópicos relevantes

# O que vimos até agora sobre collections?

- Soluções computacionais manipulam dados
- Além dos recursos básicos (tipos, variáveis, vetores e classes)
- **Estes dados podem requerer estruturas mais avançadas**
- Collections é uma framework que fornece estes recursos para manipular os dados destas soluções
  - Framework: conjunto de elementos (classes, interfaces) e outros para dar suporte no desenvolvimento
- A Framework possui um projeto de interfaces mais altas (Collection) e de interfaces (abstração) de List, Set, Queue e outras e com as implementações concretas

```
List<String> lst = new ArrayList();           // List (Interface)   ArrayList (Classe concreta)
```



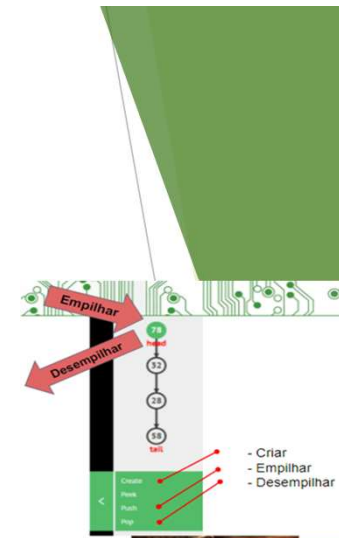
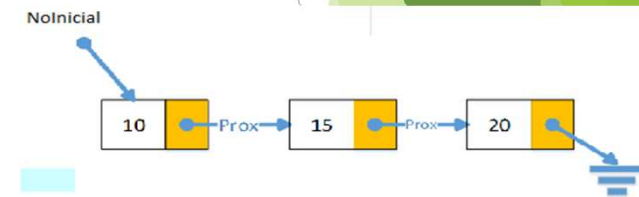
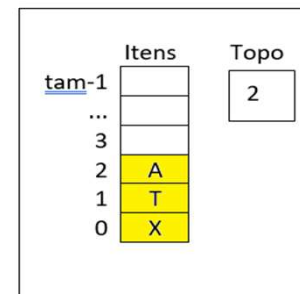
# Como compreender

- Ainda na aula passada, vimos brevemente um exemplo com ArrayList
- O foco agora é passar por cada uma das collections para discutir o seu funcionamento
- Para organizar este estudo vale a pena definir quais os tópicos para explorar

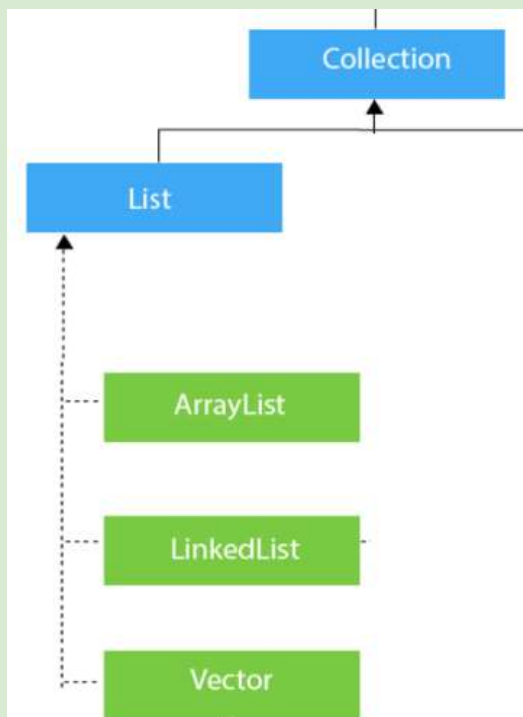
```
public static void main(String args[])
{
    ArrayList<String> l = new ArrayList();
    l.add("Fabiana");
    l.add("Marcelo");
    l.add("Cristina");
    for (int i=0; i<l.size(); i++)
    {
        String aux = l.get(i);
        System.out.println(" Nome : " + aux);
    }
}
```

# Como compreender as collections?

- Uma collection funciona como uma estrutura de dados
  - Encapsula uma forma de guardar os dados
  - Com as operações para sua manipulação
- Para compreender então
  - Qual a forma da estrutura?
  - Quais operações?
  - Exemplo de uso simples
  - Exemplos de aplicações curiosas
  - Implementações internas/Concretas
  - Variações e características
- Para ilustrar, observe os esquemas ao lado
  - O tipo abstrato (collection) pilha tem uma forma de elementos em sequência
  - E pode ser codificada internamente como um vetor ou um encadeamento





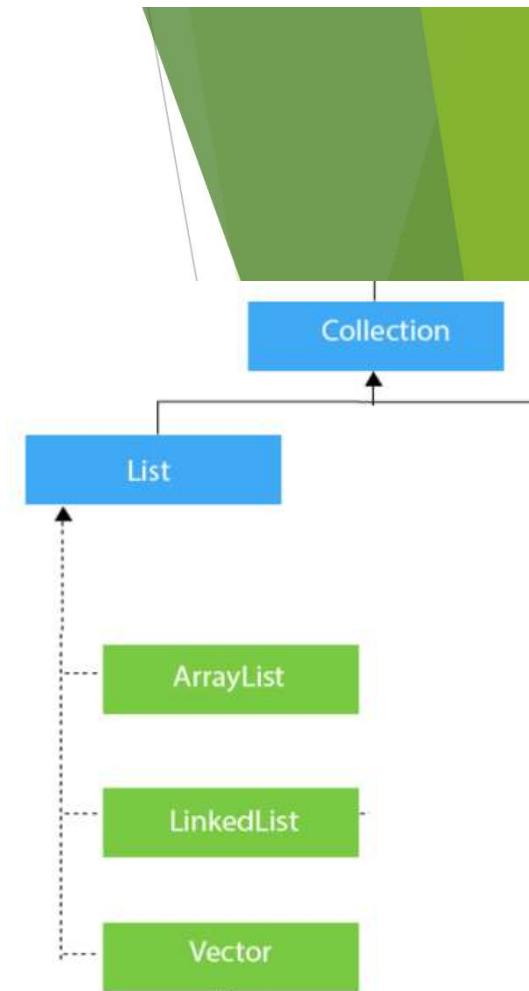


## Interface List e Arraylist, Linkedlist e Vector

- Estrutura e operações
- Exemplos simples
- Variações
- Exemplos curiosos

# Interface List

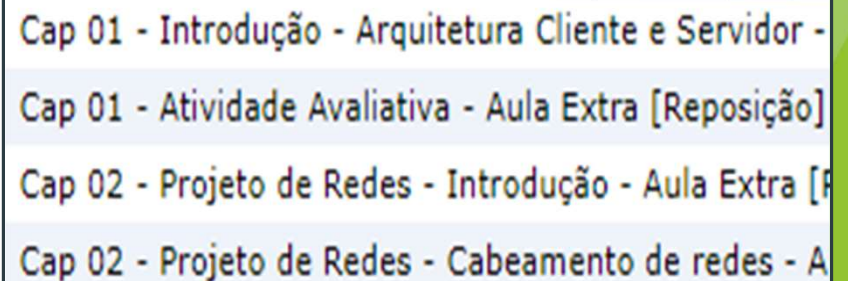
- Pensando de maneira abstrata, qual a “jeito” de uma estrutura em lista
  - São elementos em sequência
  - Permitem operações das diversas formas nas diversas posições
- Se o papel é “acomodar” dados, quais operações básicas devem ser previstas
  - Adição de elementos
  - Remoção
  - Recuperação
  - Pesquisa
  - Ordenação



# Tipo Lista (Exemplos de aplicações)

- **Exemplos didáticos**

- Cadastros em geral funcionam como lista
- (Estrutura)
  - Os tópicos de disciplinas, são uma lista
- (Lógica / “Funcionamento”)
  - **Tópicos podem ser inseridos no início, ou no meio ou no fim**
  - **E da mesma forma, removidos**



Cap 01 - Introdução - Arquitetura Cliente e Servidor -  
Cap 01 - Atividade Avaliativa - Aula Extra [Reposição]  
Cap 02 - Projeto de Redes - Introdução - Aula Extra [P  
Cap 02 - Projeto de Redes - Cabeamento de redes - A

# Tipo Lista: Outros exemplos

- Exemplo 1 - Contatos no Whatsapp
  - Lista de contatos de whatsapp, telefone, cadastro de funcionários, de clientes enfim tudo oferecem mais liberdade na manipulação adicionados



## Exemplo 2 -

### Time ou Elenco de Futebol

- Só para ilustrar mesmo, em um time de futebol as alterações dos jogadores ocorrem em qualquer posição

-

# Interface List

## - Adição de elementos

Operação	Descrição
add (E)	Adiciona um elemento ao final
add (índice, E)	Adiciona o elemento em determinado índice
addAll (Collection)	Adiciona uma collection de elementos em outra
addAll (índice, Collection)	Adiciona uma collection a partir de determinado índice

## - Remoção de elementos

Operação	Descrição
remove (index)	Remove elemento em certo índice
remove (Object)	
removeAll (Collection)	

# Interface List

- Outras operações

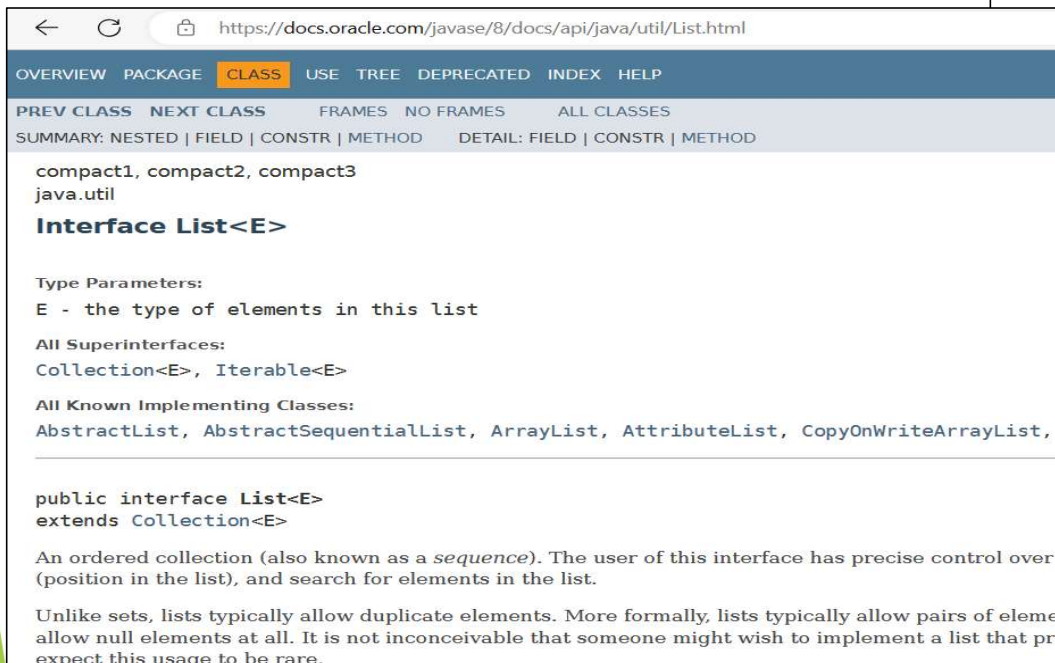
Operação	Descrição
set (index, E)	Modifica o elemento de certo índice por um novo elemento
get (index)	Recupera o elemento em determinado índice
indexOf (Object)	Pesquisa pelo elemento retornando o índice

- Outras operações

Operação	Descrição
isEmpty ()	Verifica se lista está vazia
clear ()	Limpa a lista
sort ()	Ordena a lista
Size ()	Verifica o tamanho

# Dica

- Este é um resumo das operações, vale dar uma olhada na especificação da linguagem contida no site da Oracle (proprietária)



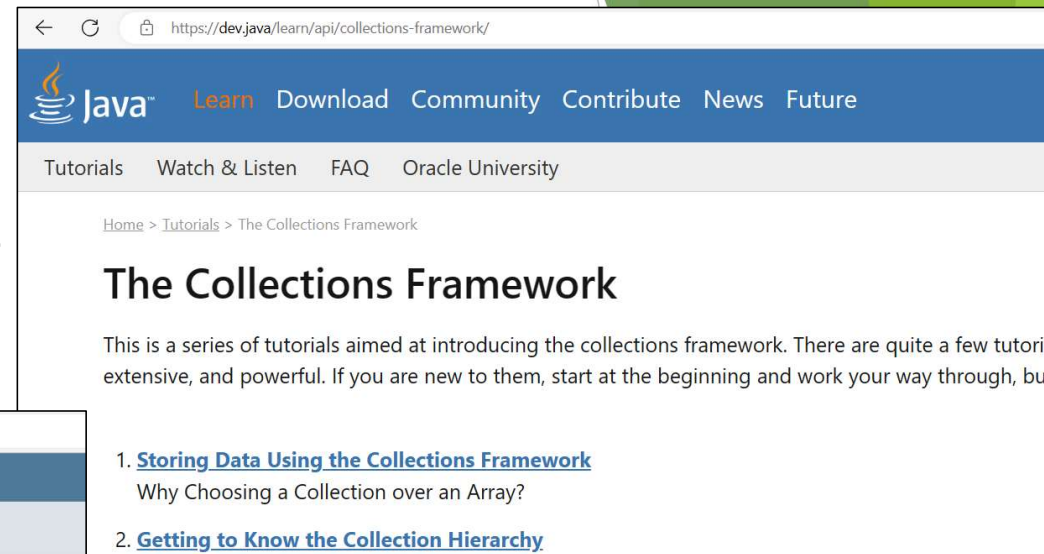
The screenshot shows the Oracle Java API documentation for the `List` interface. The browser address bar displays `https://docs.oracle.com/javase/8/docs/api/java/util/List.html`. The page has a navigation bar with tabs for OVERVIEW, PACKAGE, CLASS (selected), USE, TREE, DEPRECATED, INDEX, and HELP. Below the navigation bar, there are links for PREV CLASS, NEXT CLASS, FRAMES, NO FRAMES, and ALL CLASSES. The main content area shows the package `compact1, compact2, compact3` and `java.util`. The **Interface List<E>** is defined with the following details:

- Type Parameters:** E - the type of elements in this list
- All Superinterfaces:** `Collection<E>`, `Iterable<E>`
- All Known Implementing Classes:** `AbstractList`, `AbstractSequentialList`, `ArrayList`, `AttributeList`, `CopyOnWriteArrayList`, etc.

```
public interface List<E>
    extends Collection<E>
```

An ordered collection (also known as a *sequence*). The user of this interface has precise control over (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements that are equal, but not necessarily identical. It is not inconceivable that someone might wish to implement a list that prevents this usage to be rare.



The screenshot shows the Java Collections Framework tutorial page. The browser address bar displays `https://dev.java/learn/api/collections-framework/`. The page has a navigation bar with links for Java, Learn, Download, Community, Contribute, News, and Future. Below the navigation bar, there are links for Tutorials, Watch & Listen, FAQ, and Oracle University. The main content area is titled **The Collections Framework** and includes a description: "This is a series of tutorials aimed at introducing the collections framework. There are quite a few tutorials, extensive, and powerful. If you are new to them, start at the beginning and work your way through, but..."

1. [Storing Data Using the Collections Framework](#)  
Why Choosing a Collection over an Array?
2. [Getting to Know the Collection Hierarchy](#)

Acima está uma página com tutoriais oficiais. Ao lado a especificação técnica da interface `List`, assim como existem outras especificações dos outros elementos de Collections

# Exemplo simples 1

- O que será impresso pelo programa a seguir?

```
public static void main(String args[])
{
    List<String> lista = new ArrayList();

    lista.add("Adriana");
    lista.add("Fábio");
    lista.add("Cristiane");
    lista.add(1, "Marcelo");
    lista.add(1, "Zenildo");
    lista.remove(2);
    System.out.println("-- Resultado --");
    for (String s : lista)
    {
        System.out.println(s);
    }
}
```



# Exemplo simples 2

- O que será impresso pelo programa a seguir?

```
public static void main(String args[])
{
    List<String> lst_base = new ArrayList();
    List<String> lst_adic = new ArrayList();

    lst_base.add("Adriana");
    lst_base.add("Carlos");
    lst_base.add("Márcio");

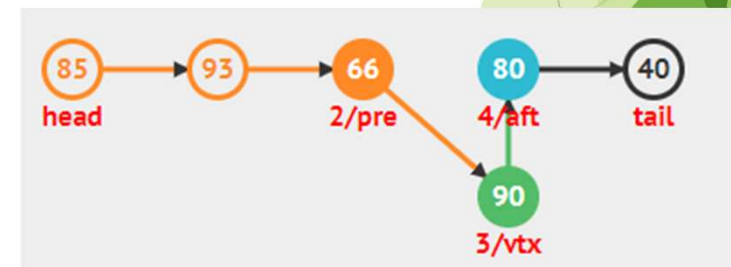
    lst_adic.add("João");
    lst_adic.add("Ana");

    lst_base.addAll(2, lst_adic);

    System.out.println("-- Resultado --");
    for (String s : lst_base.subList(1, 4))
    {
        System.out.println(s);
    }
}
```

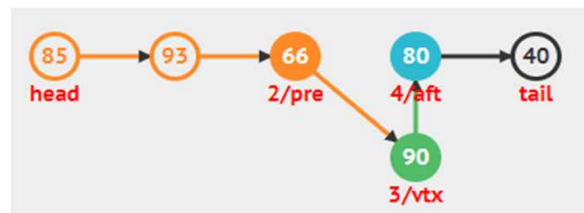
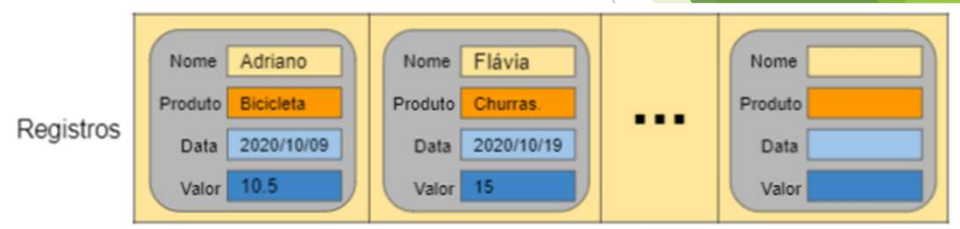
# Implementações: Arraylist e Linkedlist

- Algumas formas de implementação concreta da interface List ocorre com as classes Arraylist e Linkedlist
- A diferença entre elas é normalmente discutida em disciplinas de estruturas de dados
- A base da implementação de um Arraylist é um array com os elementos armazenados de forma contígua
- A LinkedList é a implementação de lista encadeada



# Implementações: Arraylist e Linkedlist

- Arraylist é mais apropriada em situações que não mudam muito de tamanho e sobretudo que as operações ocorrem apenas ao final do conjunto de elementos
  - Operações ocorridas no meio da estrutura requerem muito deslocamento de registros para manter a alocação contígua
- A linkedlist é mais dinâmica e adequada para muitas alterações sobretudo no meio da lista



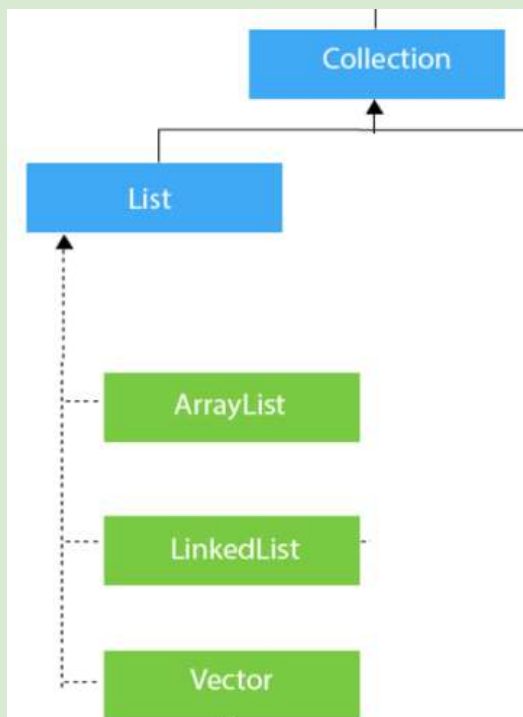
# Arraylist e Vector

- São muito semelhantes
- Apenas algumas poucas diferenças
  - Estas estruturas possuem um caráter dinâmico, isto é, conforme seu preenchimento total ela cresce automaticamente
  - No caso do ArrayList o aumento automático é de 50%
    - Um arraylist com 100 elementos ao encher passaria para 150
  - Já no caso do Vector o aumento é de 100%
    - Neste caso o Vector é dobrado ao se totalmente preenchido
- Além disso, existe um detalhe quanto ao uso por diversas threads, em soluções concorrentes. O Vector é automaticamente sincronizado.

Registros

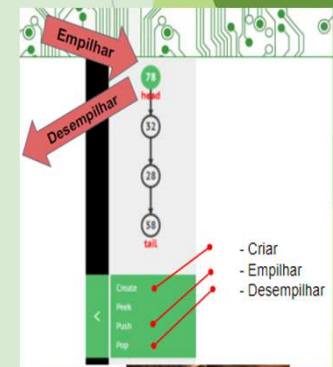
Nome Adriano	Nome Flávia	...	Nome
Produto Bicicleta	Produto Churras.		Produto
Data 2020/10/09	Data 2020/10/19		Data
Valor 10.5	Valor 15		Valor

+



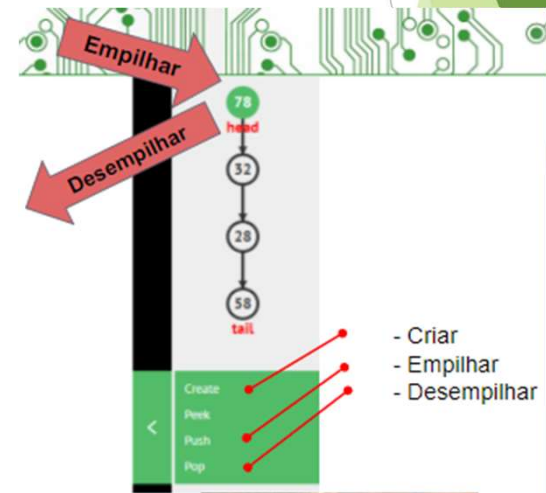
## Tipo Pilha : Stack

- Estrutura e operações
- Exemplos simples
- Variações
- Exemplos curiosos



# Tipo Pilha (Stack)

- Três tipos de dados (collections) possuem uma forte relação entre elas: Lista, Pilha e Fila
- Todas elas apresentam uma estruturação linear, com elementos em sequência e a diferença está na forma como os itens são inseridos e retirados da estrutura
- No tipo Pilha, os itens são inseridos e retirados pelo topo da pilha



# Tipo Pilha (Stack) - Operações

- **Push (Empilhar)**
  - Adiciona um elemento no topo
- **Pop (Desempilhar)**
  - Retira o elemento do topo

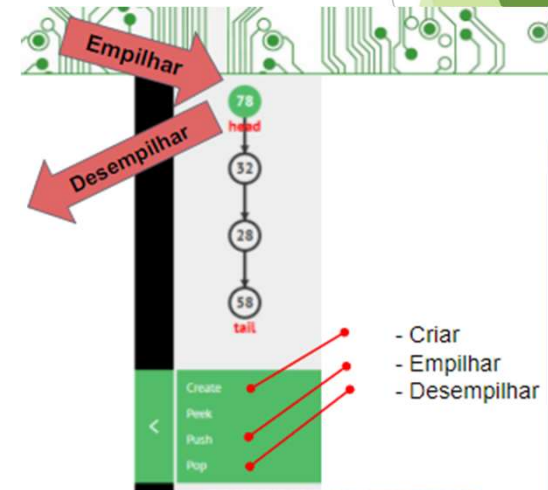
```
public static void main(String args[])
{
    Stack<String> pilha = new Stack();

    pilha.push("Este é um ");      pilha.push("teste com ");
    pilha.push("edição de ");      pilha.push("texto como ");
    pilha.push("exemplo de ");     pilha.push("erro 1 ");
    pilha.push("erro 2.");

    System.out.println("-- Resultado 1 --");
    System.out.println(pilha.pop());
    System.out.println(pilha.pop());

    pilha.push("fazer e ");
    pilha.push("desfazer.");

    System.out.println("-- Resultado 2 --");
    for (String s : pilha)
    {
        System.out.println(s);
    }
}
```





# Exemplo 1 (Stack)

- O que será impresso pelo programa a seguir?

```
public static void main(String args[])
{
    Stack<String> pilha = new Stack();

    pilha.push("Este é um ");      pilha.push("teste com ");
    pilha.push("edição de ");      pilha.push("texto como ");
    pilha.push("exemplo de ");      pilha.push("erro 1 ");
    pilha.push("erro 2.");

    System.out.println("-- Resultado 1 --");
    System.out.println(pilha.pop());
    System.out.println(pilha.pop());

    pilha.push("fazer e ");
    pilha.push("desfazer.");

    System.out.println("-- Resultado 2 --");
    for (String s : pilha)
    {
        System.out.println(s);
    }
}
```



# Pilha (Stack): Exemplos curiosos

## - Exemplo 1 - Tipos de estoque

- Nos portos, os containers são organizados em pilhas
- Apenas os containers do topo podem ser removidos ou adicionados
- Cargas e descargas

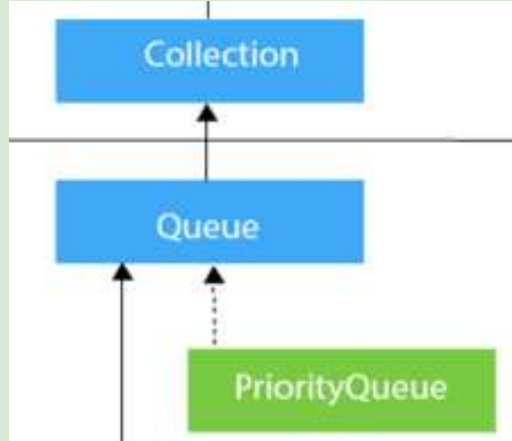


## Exemplo 2 -

### Comandos de Desfazer e Refazer

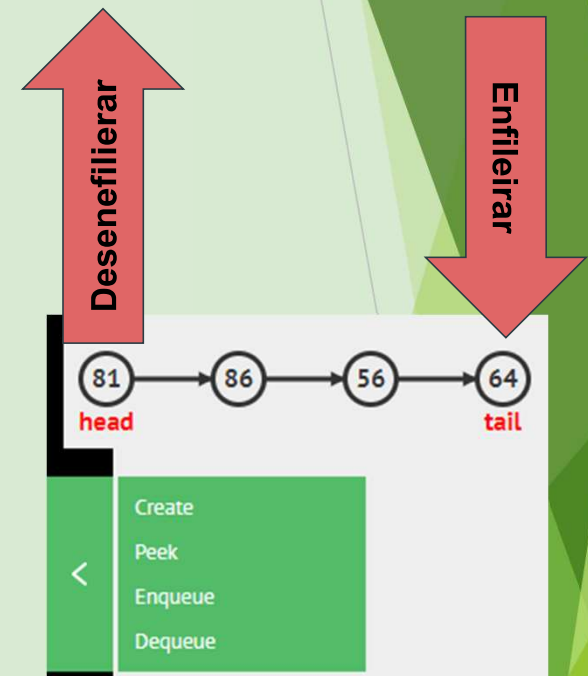
- Perceba que quando você solicita “Desfazer”
- Em algum software (word por exemplo)
- A última alteração é desfeita
- As edições “empilhadas” uma após a outra, possibilitando desfazer a última

- Perceba, são dois cenários extremamente diferentes, **comandos do word** e **containers em um porto** mas a **estruturação é a mesma**



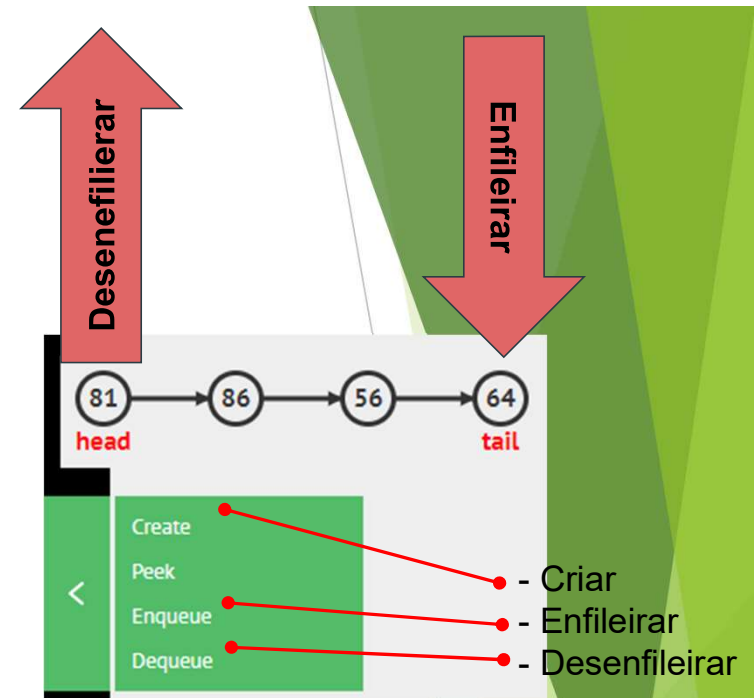
## Tipo Fila : Queue

- Estrutura e operações
- Exemplos simples
- Variações
- Exemplos curiosos



# Tipo Fila (Queue)

- Outro tipo (collection) muito comum dentre as tipos que são sequências (Lista, Pilha e Fila) é a Fila
- Neste tipo, os itens são inseridos em uma extremidade e retirados de outra extremidade
- Termos muito comuns usados nestes tipos
  - Pilha – LIFO –
    - Last in First Out
    - Último a chegar, primeiro a sair
  - Fila – FIFO –
    - First in First Out
    - Primeiro a chegar, primeiro a sair



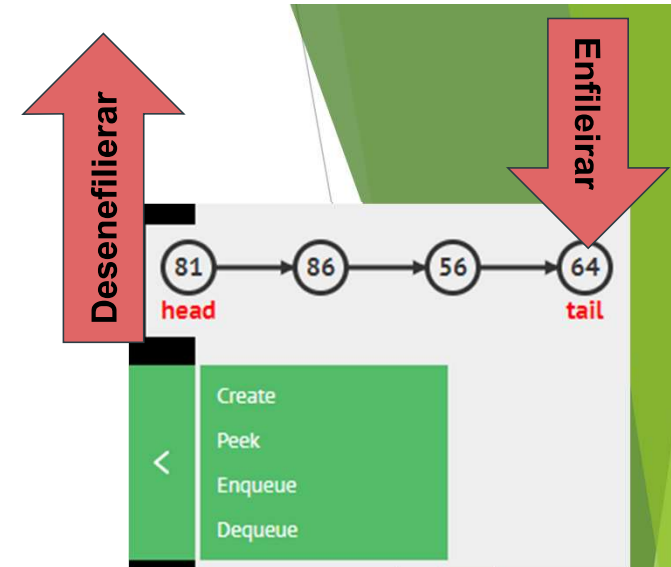
# Tipo Fila (Queue) - Operações

## - Add / Offer (Enfileirar)

- Adicionam um elemento ao final da fila
- A diferença entre elas está no tratamento de erros
  - Add : retorna uma exceção
  - Offer : apenas indica false quando não insere, usada quando o erro é “normal!”

## - Remove / poll (Desenfileirar)

- Removem os elementos da fila
- Da mesma forma a diferença está no tratamento de erro com “poll” sendo menos crítico, apenas retorna NULL



```
public static void main(String args[])
{
    Queue<String> fila = new PriorityQueue();

    fila.add("A");          fila.offer("B");
    fila.add("C");          fila.offer("D");

    System.out.println("  Itens retirados --");
    System.out.println(fila.poll());
    System.out.println(fila.poll());
}
```

## Exemplo 1 (Fila)

- O que será impresso pelo programa a seguir?

```
public static void main(String args[])
{
    Queue<String> fila = new PriorityQueue();

    fila.add("A");        fila.offer("B");
    fila.add("C");        fila.offer("D");

    System.out.println("-- Resultado 1 --");
    for (String s : fila)
    {
        System.out.println(s);
    }

    System.out.println("-- Itens retirados --");
    System.out.println(fila.poll());
    System.out.println(fila.poll());

    System.out.println("-- Resultado 2 --");
    for (String s : fila)
    {
        System.out.println(s);
    }
}
```

# Tipo Fila: Exemplos curiosos

## Exemplo 1 - Ingressos pela Internet

- Quando você se cadastra, você tem uma ordem na fila!
- Quando os ingressos são vendidos, a fila é obedecida.



- **Exemplo 2 - Fila de espera para aluguel**
  - Lembra do exemplo dos interessados em alugar um apartamento
  - Eles aguardam em uma fila, o primeiro que chegou tem prioridade

# Iterator

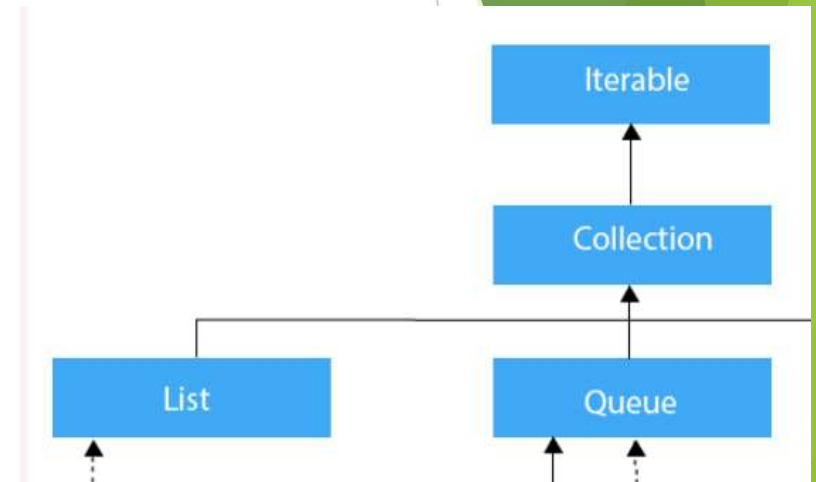
- Explorando conceito de interface
- Importância de conceito
- Percorrendo Collections com Iterator



# Percorrendo collections

## Questão chave!

- Ao manipular um conjunto de coisas, você concorda que uma operação comum é a de percorrer o conjunto de coisas?
- Lembre-se da manipulação de vetores, era comum um laço de repetição do primeiro ao último índices
- Por esta natureza comum ao lidar com conjunto ou coleção de coisas, perceba que na modelagem da framework a Interface collections herda/implementa a interface Iterable, para justamente para ser possível percorrer as coleções





# Iterator

- Cada collection pode retornar um elemento Iterator que permite percorrer a collection.
- Iterator possui dois métodos para realizar este percorrer pela collection
- Para usar o Iterator
  - **A) Recuperar este objeto a partir da collection**
  - **B) hasNext()**
    - Verifica se existe algum próximo elemento
    - Comum uso no While
  - **C) next()**
    - Recupera o próximo elemento

```
System.out.println("-- Resultado com iterator --");
Iterator<String> it = aux.iterator();
while(it.hasNext())
{
    System.out.print(it.next());
}
```

# Conclusões

- Collections mais tradicionais
- Proposta de exercícios

# Conseguimos atingir os objetivos?

- Compreender as collections básicas ou mais tradicionais

## Tópicos

- *Como compreender as collections*
- *List*
  - *Operações básicas*
  - *Aplicações*
  - *Implementações: ArrayList e LinkedList*
  - *Implementações: ArrayList e Vector*
- *Queue*
- *Stack*
- *Compreendendo o iterator*

