

Lógica de Programação

Estruturas de Dados e
Complexidade dos Algoritmos

Pilha e Fila - Sequencial e Encadeada

Germinare Tech

Carlos Santi e Marcelo Modolo

GerminaTECH

Pilha e Fila – Sequencial e Encadeada

Carlos Santi e Marcelo Modolo

Relembrando...

- **Tipo Abstrato de Dados (TAD):** define o conjunto de valores e as operações sobre os valores, mas não define sua implementação.
Exemplos: lista, pilha, fila, árvore
- **Estrutura de Dados (ED):** é uma maneira organizar e representar computacionalmente um Tipo Abstrato de Dados.
Exemplos: sequencial, encadeado

Tipo Abstrato de Dados (TAD)

- **Lista:** conjunto de itens interligados cujas operações de inserção e remoção podem ser feitas em qualquer parte da lista
- **Pilha:** conjunto de itens interligados cujas operações de inserção e remoção só podem ser feitas no topo da pilha (LIFO ou UEPS)
- **Fila:** conjunto de itens interligados cuja operação de inserção é feita no final da fila e a operação de remoção é feita no início da fila (FIFO ou PEPS)

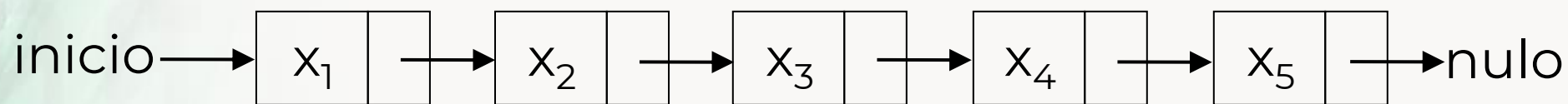
Estrutura Sequencial (ou Estática)

- Os itens são armazenados em posições contíguas da memória
- O tamanho da estrutura deve ser definido na sua criação
- A estrutura pode ser percorrida em qualquer direção
- Qualquer item pode ser acessado diretamente

0	x_1
1	x_2
2	x_3
3	x_4
4	x_5
5	
6	

Estrutura Encadeada (ou Dinâmica)

- Cada item é encadeado por um ponteiro que aponta para o item seguinte da estrutura
- Permite armazenamento em posições não contíguas de memória
- O acesso a um determinado item é feito passando por todos os itens que o precedem



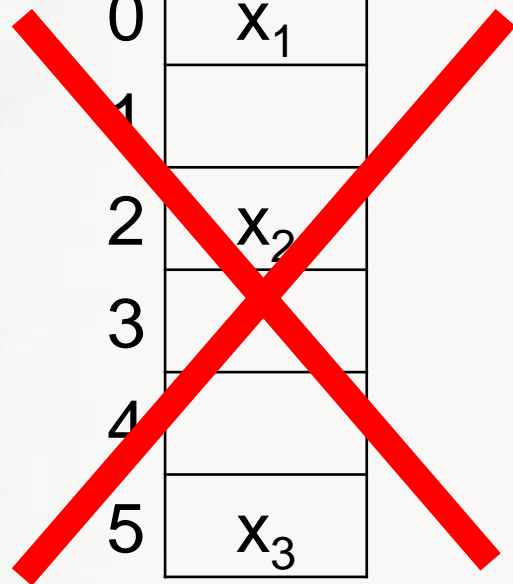
Estrutura Sequencial (ou Estática) em JAVA

Vetor em Java

- O Vetor é uma estrutura adequada em Java para implementar um Estrutura de Dados Sequencial.
- Sintaxe para criação de um vetor:
*<tipo ou classe> [] <nome do vetor> =
new <tipo ou classe> [<quantidade de valores>];*
- Sintaxe para acesso a um item ao vetor:
<nome do vetor> [<posição>];

Manipulação de Vetores

- Armazenamento dos itens poderia ser arbitrário em quaisquer posições do vetor
- Melhor custo-benefício é obtido se armazenar itens compactados no início do vetor



0	x_1
1	
2	x_2
3	
4	
5	x_3

0	x_1
1	x_2
2	x_3
3	
4	
5	

Pilhas e Filas

Sequenciais (ou Estáticas)

- Pilhas e filas são representadas computacionalmente em Java usando vetores de maneira semelhante à lista
- As regras de negócio das pilhas e filas são extremamente rígidas
- Por terem restrições quanto a entrada e saída dos itens, as pilhas e filas têm menos regras de negócio que a lista

Pilha Sequencial (ou Estática)

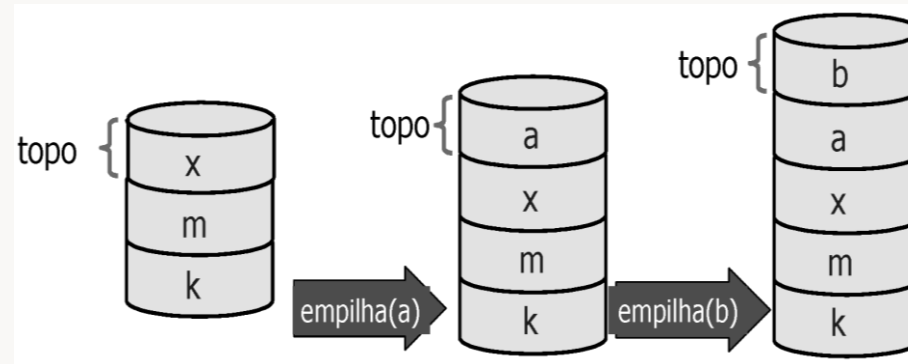
Regras de Negócio da Pilha

1. Empilhar (push)
2. Remover elemento do topo (pop)
3. Retornar topo (peek)
4. Verificar se a pilha está vazia (isEmpty)

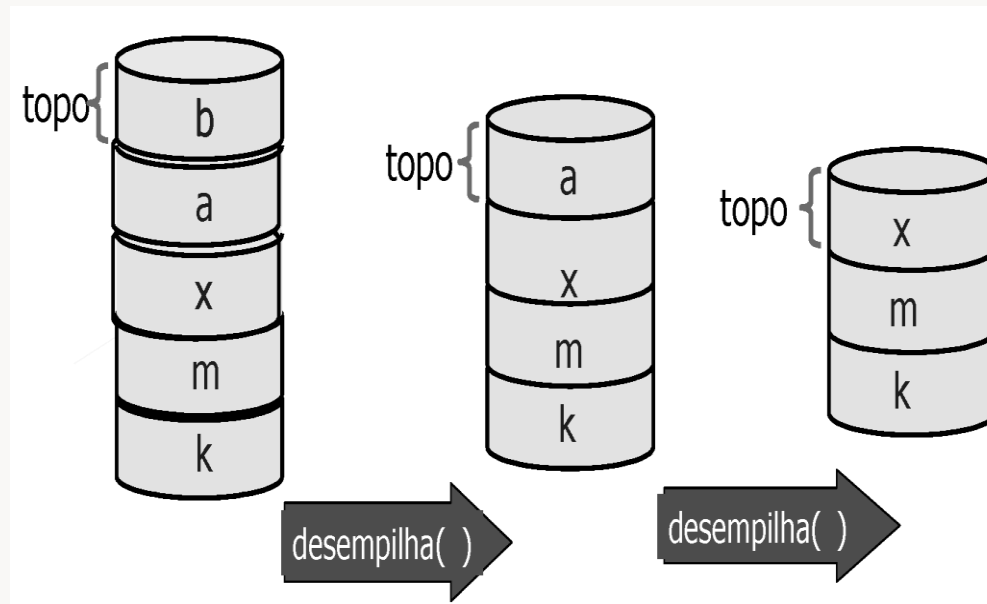
Observação: dentro dos parênteses estão os nomes usados na literatura

Regras de Negócio da Pilha

Empilhar



Remover Topo



Pilha Sequencial (ou Estática)

- Vamos exemplificar como implementar as quatro regras de negócio da pilha usando um vetor para mostrar como é o funcionamento de uma pilha sequencial (ou estática)
- Vamos criar uma classe chamada PilhaEstatica que implementa uma pilha sequencial de números reais

Criar a classe e o construtor

```
public class PilhaEstatica {  
    private int capacidade;  
    private int topo;  
    private double[] container;  
  
    public PilhaEstatica(int capacidade) {  
        this.capacidade = capacidade;  
        this.topo = -1;  
        this.container = new double[capacidade];  
    }  
}
```

Regra 1: Empilhar (push)

```
public boolean push(double item) {  
    if (this.topo + 1 == this.capacidade) {  
        return false;  
    } else {  
        this.topo = this.topo + 1;  
        this.container[this.topo] = item;  
        return true;  
    }  
}
```

Regra 2: Remover elemento do topo (pop)

```
public double pop() {  
    int temp = this.topo;  
    this.topo--;  
    return this.container[temp];  
}
```

Regra 3. Retornar topo (peek)

```
public double peek() {  
    return this.container[this.topo];  
}
```

Regra 4. Verificar se a pilha está vazia (isEmpty)

```
public boolean isEmpty() {  
    if (this.topo == -1) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Mostrar a pilha

```
public String toString() {  
    String s = "";  
    for (int i = 0; i <= this.topo; i++) {  
        s += String.valueOf(this.container[i]) + " ";  
    }  
    return s;  
}  
} // Fim da classe PilhaEstática
```


Classe Principal

```
import java.util.Scanner;

public class Principal {

    public static void main(String[ ] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Criando a pilha:");
        PilhaEstatica pilha = new PilhaEstatica(20);
        System.out.println("Pilha vazia: " + pilha.isEmpty());
    }
}
```

- Saída ao executar o programa:

run:

Criando a pilha:

Pilha vazia: true

Classe Principal

```
System.out.println("Inserindo 10 números na pilha:");  
for (int i = 0; i < 10; i++) {  
    pilha.push(i+1);  
}  
System.out.println("Pilha: " + pilha);
```

- Saída ao executar o programa:

Inserindo 10 números na pilha:

Pilha: 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0

Classe Principal

```
System.out.println("Pilha vazia: " + pilha.isEmpty());  
System.out.println("Topo: " + pilha.peek());
```

- Saída ao executar o programa:

Pilha vazia: false

Topo: 10.0

Classe Principal

```
System.out.println("Removendo o topo:");  
if (!pilha.isEmpty()) {  
    System.out.println("Removido: " + pilha.pop());  
}  
System.out.println("Novo Topo: " + pilha.peek());
```

- Saída ao executar o programa:

Removendo o topo

Removido: 10.0

Novo Topo: 9.0

Classe Principal

```
System.out.println("Digite três números para serem inseridos: ");  
    pilha.push(teclado.nextDouble());  
    pilha.push(teclado.nextDouble());  
    pilha.push(teclado.nextDouble());
```

- Saída ao executar o programa:

Digite três números para serem inseridos:

5,7

9,4

7,2

Classe Principal

```
System.out.println("Novo Topo: " + pilha.peek());  
System.out.println("Pilha: " + pilha);  
}  
}
```

- Saída ao executar o programa:

Novo Topo: 7.2

Pilha: 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 5.7 9.4 7.2

Perguntas???



Exercício 1

- Crie os métodos de manipulação de uma pilha sequencial de Strings: inserir, retirar, retornar topo, verificar se está vazia e mostrar pilha. DESAFIO: não utilize o atributo “capacidade”
- No main crie uma pilha sequencial de Strings com capacidade para 10 elementos
- Crie um menu para o usuário escolher entre inserir, retirar, mostrar topo, verificar se a pilha está vazia e mostrar a pilha

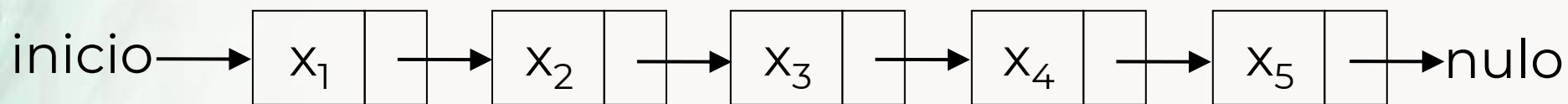
Exercício 2

- Crie a classe Produto com os atributos código, nome e preço, método construtor, getters e setters, e toString
- Crie os métodos de manipulação de uma pilha sequencial de Produto: inserir, retirar, retornar topo, verificar se está vazia e mostrar pilha
- No método main crie uma pilha sequencial com número de objetos informado pelo usuário
- Crie um menu para o usuário escolher entre inserir, retirar, mostrar topo, verificar se a pilha está vazia e mostrar a pilha

Estrutura Encadeada (ou Dinâmica) em JAVA

Estrutura Encadeada (ou Dinâmica)

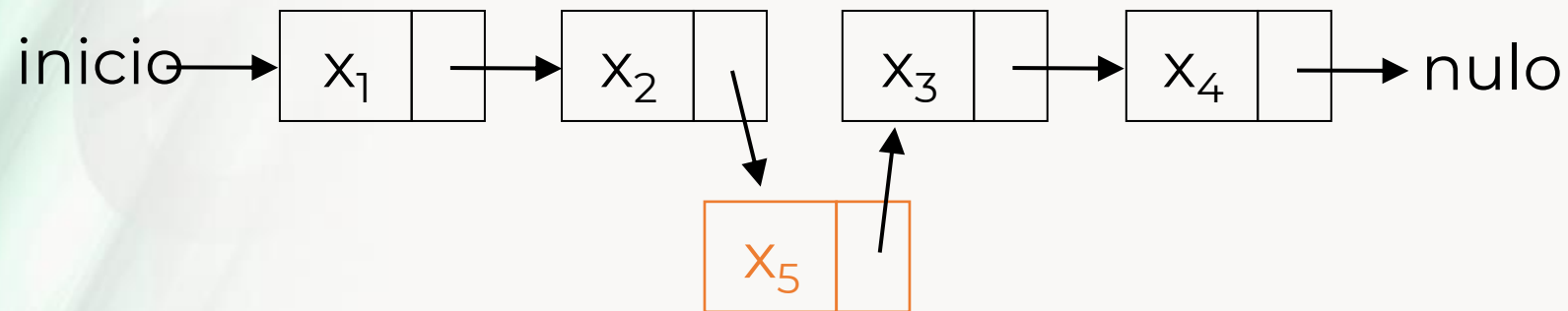
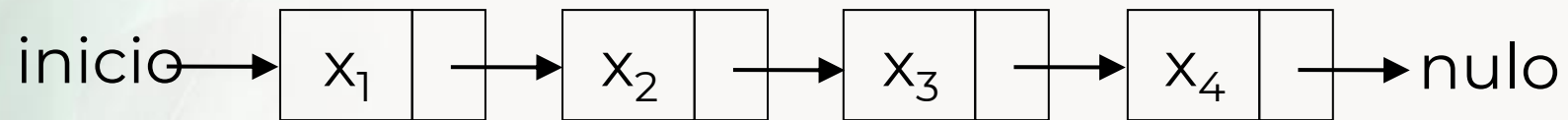
- Cada item é encadeado por um ponteiro que aponta para o item seguinte da estrutura
- Permite armazenamento em posições não contíguas de memória
- O acesso a um determinado item é feito passando por todos os itens que o precedem



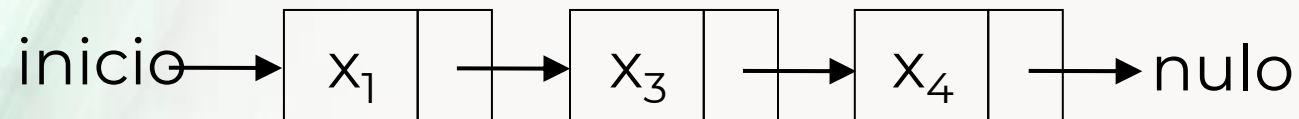
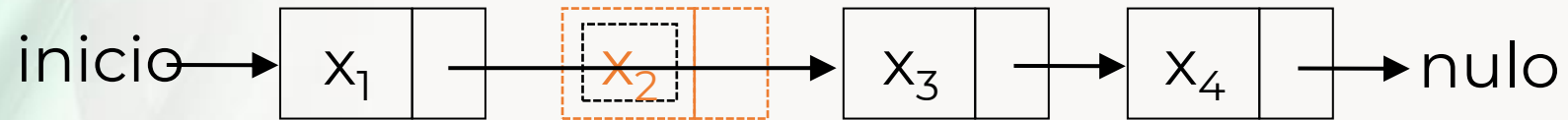
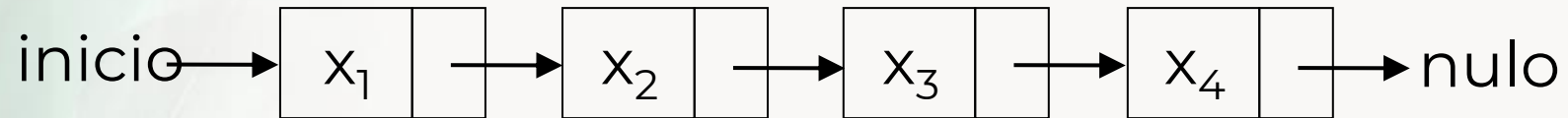
Porque usar Estrutura Encadeada

- A estrutura encadeada resolve o problema de desperdício de memória
- Nessa estrutura não é necessário reservar memória quando a estrutura é criada
- A medida que vão sendo inseridos novos itens a memória vai sendo alocada
- Teoricamente não existe limite para o número de itens dessa estrutura, mas na prática o limite é a capacidade da memória do computador

Inserir um item na lista



Removendo um item da lista



Métodos das Estruturas Encadeadas

- Existem métodos em Java que já foram construídos para as Listas, Pilhas e Filas Encadeadas
- Esses métodos estão em classes e Interfaces que vamos usar para manipular essas estruturas encadeadas:
 - Lista: classe LinkedList
 - Pilha: classe Stack
 - Fila: interface Queue
- API JAVA: <http://java.sun.com/javase/6/docs/api/>

Pilha Encadeada (ou Dinâmica)

Pilha: Métodos da API Stack

- **Stack()**: Cria uma pilha vazia (CONSTRUTOR)
- boolean **isEmpty()**: testa se a pilha está vazia, retorna true se estiver vazia
- Object **push**(Object item): coloca o objeto no topo da pilha
- Object **peek**(): retorna o objeto do topo da pilha sem removê-lo da mesma
- Object **pop**(): remove o objeto do topo da pilha, retornando esse objeto

Exemplo 1: Utilização dos Métodos

```
import java.util.Scanner;  
import java.util.Stack;  
  
public class Principal {  
  
    public static void main(String[] args) {  
        Stack pilha = new Stack();  
        Scanner sc = new Scanner(System.in);  
        pilha.push(5.5);  
        System.out.println("Digite um número real: ");  
        pilha.push(sc.nextDouble());  
        pilha.push(10.0);  
    }  
}
```

Exemplo 1: Utilização dos Métodos

```
System.out.println("Topo: " + pilha.peek());  
System.out.println("Retirado topo: " + pilha.pop());  
System.out.println("Topo: " + pilha.peek());  
}  
}
```

Saída ao executar o programa:

run:

Digite um número real: 12,8

Topo: 10.0

Retirado topo: 10.0

Topo: 12.8

Exemplo 2: Utilização dos Métodos

- Vamos criar uma pilha de Pessoas e usar os métodos da classe Stack para manipular essa pilha.
- Crie a classe Pessoa com os atributos nome e altura, método construtor, getters e setters, e toString
- No método main:
- Crie uma pilha encadeada vazia de objetos Pessoa
- Verifique se a pilha está vazia e mostre uma mensagem
- Inclua cinco objetos pessoa na pilha
- Verifique se a pilha está vazia
- Mostre o topo da pilha
- Mostre a pilha
- Insira um elemento na pilha e mostre o novo topo
- Mostre a pilha

Exemplo 2: Utilização dos Métodos

```
public class Pessoa {  
    private String nome;  
    private double altura;  
    public Pessoa(String nome, double altura) {  
        this.nome = nome;  
        this.altura = altura;  
    }  
    public String toString() {  
        return nome + " tem " + altura + " metros";  
    }  
}
```

Construir a Classe Pessoa

```
public String getNome() {  
    return nome;  
}  
public void setNome(String nome) {  
    this.nome = nome;  
}  
public double getAltura() {  
    return altura;  
}  
public void setAltura(double altura) {  
    this.altura = altura;  
}  
} // Fim da classe
```

Criar a pilha

```
import java.util.Scanner;  
import java.util.Stack;  
  
public class Principal {  
    public static void main(String[ ] args) {  
        Scanner teclado = new Scanner(System.in);  
        System.out.println("Criando a pilha:");  
        Stack <Pessoa> pilha = new Stack();  
        System.out.println("Pilha vazia: " + pilha.isEmpty());  
    }  
}
```

Saída ao executar o programa:

run:

Criando a pilha:

Pilha vazia: true

Inserir elementos na pilha

```
System.out.println("Digite cinco nomes e alturas  
para inserir na pilha:");  
for (int i = 0; i < 5; i++) {  
    pilha.push(new Pessoa(teclado.next(),  
        teclado.nextDouble()));  
}
```

Inserir elementos na pilha

- Saída ao executar o programa:

Digite cinco nomes e alturas para inserir na pilha:

Laura

1,57

Taiz

1,64

Marcia

1,74

Magda

1,77

Percia

1,52

Testar pilha vazia e mostrar topo

```
System.out.println("Pilha: " + pilha);  
System.out.println("Pilha vazia: " + pilha.isEmpty());  
if (!pilha.isEmpty()) {  
    System.out.println("Topo: " + pilha.peek());  
}
```

- Saída ao executar o programa:

Pilha: [Laura tem 1.57 metros, Taiz tem 1.64 metros, Marcia tem 1.74 metros, Magda tem 1.77 metros, Percia tem 1.52 metros]

Pilha vazia: false

Topo: Percia tem 1.52 metros

Remover topo

```
System.out.println("Removendo o topo:");  
if (!pilha.isEmpty()) {  
    System.out.println("Removido: " + pilha.pop());  
}  
if (!pilha.isEmpty()) {  
    System.out.println("Novo Topo: " + pilha.peek());  
}
```

- Saída ao executar o programa:

Removendo o topo:

Removido: Percia tem 1.52 metros

Novo Topo: Magda tem 1.77 metros

Inserir elemento

```
System.out.println("Digite um nome e uma altura para  
serem inseridos: ");  
pilha.push(new Pessoa (teclado.next(), teclado.nextDouble()));  
if (!pilha.isEmpty()) {  
    System.out.println("Novo Topo: " + pilha.peek());  
}
```

- Saída ao executar o programa:

Digite um nome e uma altura para serem inseridos:

Maria

1,66

Novo Topo: Maria tem 1.66 metros

Mostrar pilha

```
        System.out.println("Pilha: " + pilha);  
    }  
}
```

- Saída ao executar o programa:

Pilha: [Laura tem 1.57 metros, Taiz tem 1.64 metros,
Marcia tem 1.74 metros, Magda tem 1.77 metros,
Maria tem 1.66 metros]

CONSTRUÍDO COM SUCESSO (tempo total: 1 minutos
23 segundos)

Perguntas???



Exercício 1

- Usando os métodos da classe Stack construa um programa que cria uma pilha vazia de números naturais e permite ao usuário incluir nessa pilha pelo teclado quantos elementos desejar. A inclusão deve ser encerrada quando o usuário digitar um número negativo.
- Depois o programa deverá criar um menu para o usuário escolher entre inserir, retirar, mostrar topo, verificar se a pilha está vazia e mostrar a pilha.

Exercício 2

- Escreva um programa que lê o nome e a nota de candidatos de um concurso e insere em um vetor. O número máximo de candidatos é 50.
- Em seguida deve criar uma pilha encadeada na qual os candidatos devem ser colocados em ordem decrescente de sua classificação no concurso. Ou seja, o programa deve percorrer o vetor pegando o candidato a partir da maior nota até a menor e colocar nessa ordem na pilha.
- Depois de colocar os candidatos na pilha, o programa deve solicitar que o usuário digite um número que indica a quantidade de classificados. Então o programa divide a pilha lida em duas outras pilhas, uma com os nomes dos candidatos classificados e outra com os nomes dos não classificados. Essas pilhas devem estar ordenadas de forma que no seu topo está o candidato pior classificado entre aqueles da pilha. No final o programa deve mostrar, separadamente, as duas pilhas.

Resumo sobre Pilhas

- Nessas aulas estudamos como manipular Pilhas Sequenciais (ou Estáticas) e Encadeadas (ou Dinâmicas) usando a linguagem Java
- Criamos os métodos para as Pilhas Sequenciais e usamos os métodos já existentes em classe e Interface do Java para as Pilhas Encadeadas

Fila Sequencial (ou Estática)

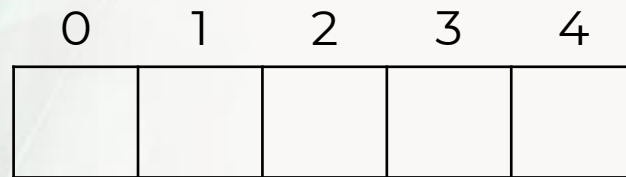
Regras de Negócio da Fila

1. Verificar se a fila está vazia (isEmpty)
2. Inserir um elemento no final da fila (offer)
3. Retirar um elemento do início da fila (poll)
4. Retornar o elemento do início da fila (peek)
5. Verificar o tamanho da fila (size)

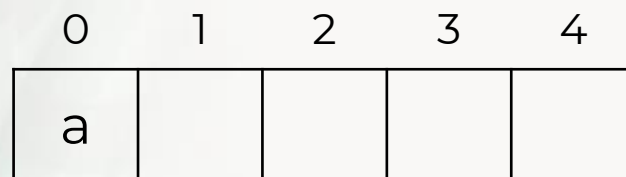
Observação: dentro dos parênteses estão os nomes usados na literatura

Funcionamento da Fila

Fila vazia

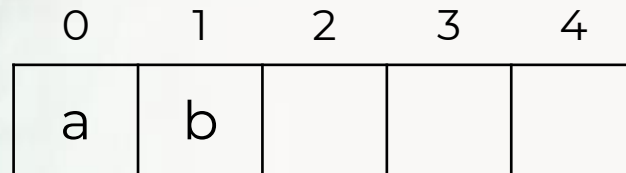


Inserir item a



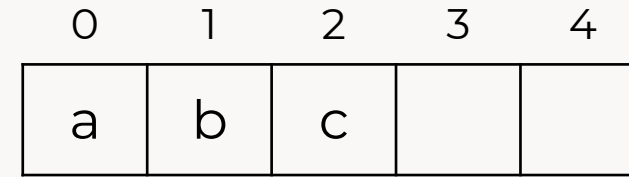
ini
fim

Inserir item b



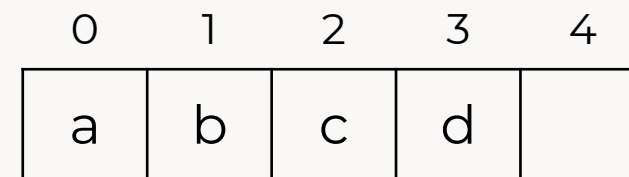
ini fim

Inserir item c



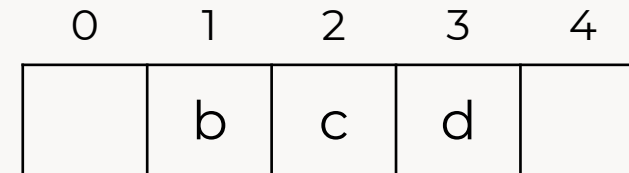
ini fim

Inserir item d



ini fim

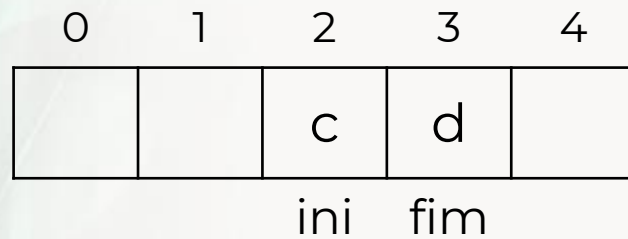
Remover item



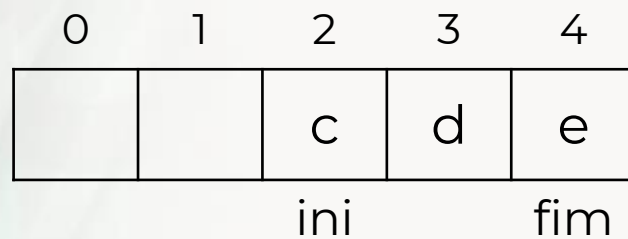
ini fim

Funcionamento da Fila

Remover item

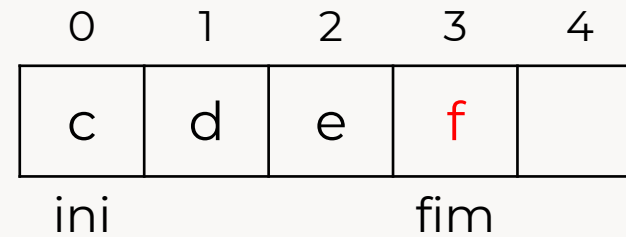


Inserir item e

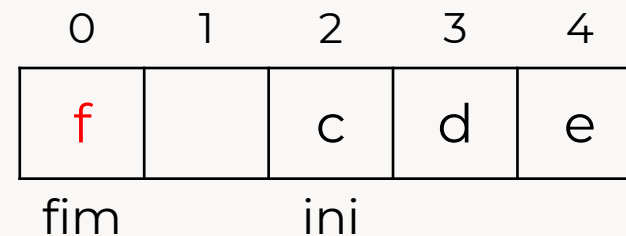


Como inserir o
item **f**?

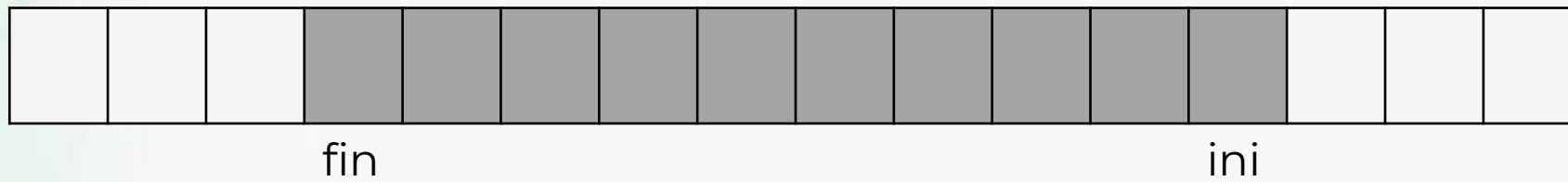
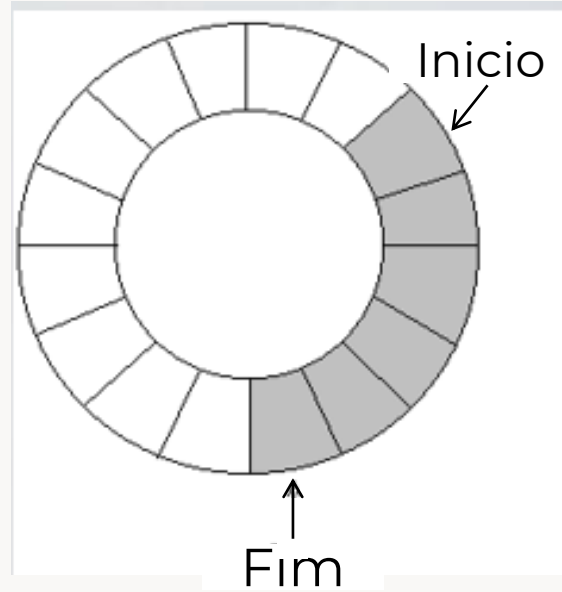
Movendo todos os
elementos



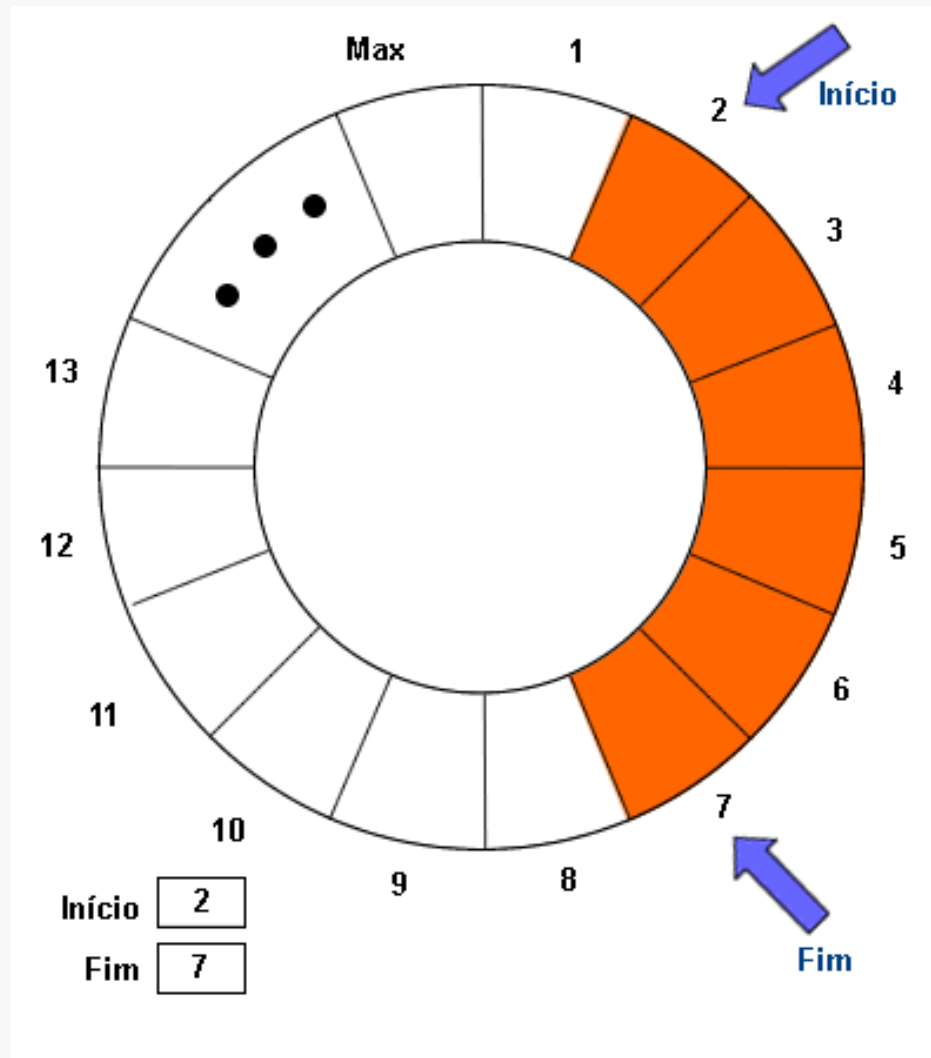
Simulando uma
fila circular



Fila Circular



Fila Circular



Fila Circular Sequencial (ou Estática)

- Vamos exemplificar como implementar as regras de negócio da fila usando um vetor para mostrar como é o funcionamento de uma fila circular sequencial (ou estática)
- Vamos criar uma classe chamada FilaCircularEstática que implementa uma fila sequencial de nomes de pessoas

Regra 1: Iniciar fila

```
public class FilaCircularEstatica {  
    private int capacidade;  
    private int inicio;  
    private int fim;  
    private int quantidadeArmazenada;  
    private String[ ] container;  
  
    public FilaCircularEstatica(int capacidade) {  
        this.capacidade = capacidade;  
        quantidadeArmazenada = 0;  
        container = new String[capacidade];  
    }  
}
```

Regra 2: Verificar se a fila está vazia (isEmpty)

```
public boolean isEmpty() {  
    if (quantidadeArmazenada == 0) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Regra 3. Inserir um elemento no final da fila (offer)

```
public boolean offer(String x) {  
    if (quantidadeArmazenada == capacidade) {  
        return false;  
    } else {  
        if (isEmpty()) {  
            fim = 0;  
            inicio = 0;  
        }  
        quantidadeArmazenada++;  
        container[fim] = x;  
        fim = (fim + 1) % capacidade;  
        return true;  
    }  
}
```

Regra 4. Retirar um elemento do início da fila (poll)

```
public String poll() {  
    String temp = container[inicio];  
    quantidadeArmazenada--;  
    inicio = (inicio + 1) % capacidade;  
    return temp;  
}
```

Regra 5. Copiar o elemento do início da fila (peek)

```
public String peek() {  
    return container[inicio];  
}
```

Regra 6. Verificar o tamanho da fila (size)

```
public int size() {  
    return quantidadeArmazenada;  
}
```

Mostrar a fila

```
public String toString() {  
    String s = "";  
    for (int i = 0; i < size(); i++) {  
        s = s + String.valueOf(container[(inicio + i) % capacidade]) + " ";  
    }  
    return s;  
}
```


Método main

```
public static void main(String[] args) {  
    Scanner teclado = new Scanner(System.in);  
    System.out.println("Criando a fila:");  
    FilaCircularEstatica fila = new FilaCircularEstatica(50);  
    System.out.println("Fila vazia: " + fila.isEmpty());  
}
```

- Saída ao executar o programa:

run:

Criando a fila:

Fila vazia: true

Método main

```
System.out.println("Digite cinco nomes para inserir na fila:");  
    for (int i = 0; i < 5; i++) {  
        fila.offer(teclado.next());  
    }
```

- Saída ao executar o programa:

Digite cinco nomes para inserir na fila:

Lucia

Carlos

Marisa

Guilherme

Joana

Método main

```
System.out.println("Tamanho da fila: " + fila.size());  
System.out.println("Fila: " + fila);  
System.out.println("Fila vazia: " + fila.isEmpty());  
System.out.println("Primeiro elemento: " + fila.peek());
```

- Saída ao executar o programa:

Tamanho da fila: 5

Fila: Lucia Carlos Marisa Guilherme Joana

Fila vazia: false

Primeiro elemento: Lucia

Método main

```
System.out.println("Removendo elemento:");  
    if (!fila.isEmpty()) {  
        System.out.println("Removido: " + fila.poll());  
    } else {  
        System.out.println("Fila vazia");  
    }  
    System.out.println("Primeiro elemento: " + fila.peek());  
    System.out.println("Tamanho da fila: " + fila.size());
```

- Saída ao executar o programa:

Removendo elemento:

Removido: Lucia

Primeiro elemento: Carlos

Tamanho da fila: 4

Método main

```
System.out.println("Digite dois nomes: ");  
    fila.offer(teclado.next());  
    fila.offer(teclado.next());  
    System.out.println("Tamanho da fila: " + fila.size());  
    System.out.println("fila: " + fila);  
}  
} //fim do método main
```

- *Saída ao executar o programa:*

Digite dois nomes:

João

Maria

Tamanho da fila: 6

fila: Carlos Marisa Guilherme Joana João Maria

Perguntas???



Exercício

- Implemente a classe FilaPrioridade com métodos para uma fila sequencial de prioridade usando objetos da classe pessoa que tem como atributos nome, CPF e idade
- Cada elemento da fila tem uma prioridade associada
- A inclusão é feita segundo as regras de negócio da fila, mas a remoção deve ser feita com base nessa prioridade

Fila Encadeada (ou Dinâmica)

Métodos das Estruturas Encadeadas

- Existem métodos em Java que já foram construídos para as Listas, Pilhas e Filas Encadeadas
- Esses métodos estão em classes e Interfaces que vamos usar para manipular essas estruturas encadeadas:
 - Lista: classe LinkedList
 - Pilha: classe Stack
 - Fila: Interface Queue

Fila: Métodos da Interface Queue

- NÃO TEM CONSTRUTOR
- boolean **isEmpty()**: testa se a fila está vazia, retorna true se estiver vazia
- Object **peek()**: copia o primeiro objeto da fila sem removê-lo da mesma
- Object **poll()**: remove o primeiro objeto da fila, retornando seu valor
- boolean **offer**(Object item): armazena item no final da fila

Fila Encadeada: exemplo de utilização dos métodos

- Vamos criar uma lista de Pessoas e usar os métodos da Interface Queue para manipular essa lista

```
public class Pessoa {  
    private String nome;  
    private double altura;  
  
    public Pessoa(String nome, double altura) {  
        this.nome = nome;  
        this.altura = altura;  
    }  
}
```

Criar fila

```
public class Principal {  
    public static void main(String[] args) {  
        Scanner teclado = new Scanner(System.in);  
        System.out.println("Criando a fila:");  
        Queue <Pessoa> fila = new <Pessoa> LinkedList();  
        System.out.println("Fila vazia: " + fila.isEmpty());  
    }  
}
```

- Saída ao executar o programa:

run:

Criando a fila:

Fila vazia: true

Inserir elementos na fila

```
System.out.println("Digite cinco nomes e alturas para  
                                inserir na fila:");  
for (int i = 0; i < 5; i++) {  
    fila.offer(new Pessoa (teclado.next(), teclado.nextDouble()));  
}
```

Inserir elementos na fila

- Saída ao executar o programa:
Digite cinco nomes e alturas para inserir na fila:
Jorge
1,72
Mauricio
1,65
Luiz
1,82
Paulo
1,78
Carlos
1,68

Mostrar tamanho, fila e primeiro elemento

```
System.out.println("Tamanho da fila: " + fila.size());  
System.out.println("Fila: " + fila);  
System.out.println("Fila vazia: " + fila.isEmpty());  
System.out.println("Primeiro elemento: " + fila.peek());
```

- Saída ao executar o programa:

Tamanho da fila: 5

Fila: [Jorge tem 1.72 metros., Mauricio tem 1.65 metros., Luiz tem 1.82 metros., Paulo tem 1.78 metros., Carlos tem 1.68 metros.]

Fila vazia: false

Primeiro elemento: Jorge tem 1.72 metros

Remover elemento da fila

```
System.out.println("Removendo elemento:");  
if (!fila.isEmpty()) {  
    System.out.println("Removido: " + fila.poll());  
} else {  
    System.out.println("Fila vazia");  
}  
System.out.println("Primeiro elemento: " + fila.peek());
```

- Saída ao executar o programa:

Removendo elemento:

Removido: Jorge tem 1.72 metros.

Primeiro elemento: Mauricio tem 1.65 metros.

Inserir um elemento na fila

```
System.out.println("Tamanho da fila: " + fila.size());  
System.out.println("Digite um nome e uma altura para  
serem inseridos: ");  
fila.offer(new Pessoa (teclado.next(), teclado.nextDouble()));  
System.out.println("Tamanho da fila: " + fila.size());
```

- *Saída ao executar o programa:*

Tamanho da fila: 4

Digite um nome e uma altura para serem inseridos:

Jose

1,70

Tamanho da fila: 5

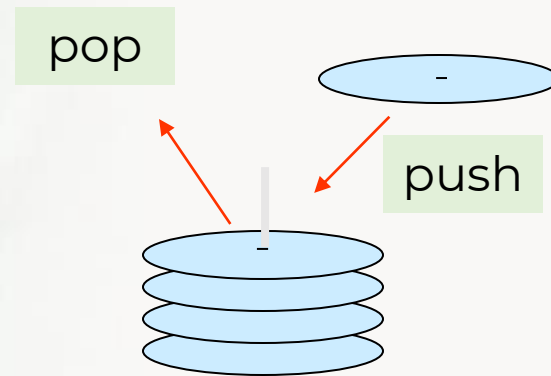
Inserir um elemento na fila

```
System.out.println("fila: " + fila);  
} // fim do método main
```

- *Saída ao executar o programa:*
fila: [Mauricio tem 1.65 metros., Luiz tem 1.82 metros., Paulo tem 1.78 metros., Carlos tem 1.68 metros., Jose tem 1.7 metros.]

Diferenças de Pilha e Fila

Pilha



LIFO (Last-In-First-Out) método de acesso

Fila



FIFO (First-In-First-Out) método de acesso

Qual usar: Fila ou Pilha?

Pilha

- Processar diretório dentro de diretório
- Mecanismo de desfazer/refazer dos editores de texto
- Navegação entre páginas web
- Registrar a sequência de métodos chamados em um programa em Java

Fila

- Processar eventos na sua ordem de chegada
- Troca de mensagens entre computadores conectados em uma rede
- Armazenar quais teclas foram acionadas
- Controle de documentos para impressão

Perguntas???



Exercício

- Em um site de comércio eletrônico, os clientes fazem encomendas de produtos
- A classe Produto tem os atributos código, descrição, preço
- A classe Cliente tem os atributos CPF, nome, endereço
- Implemente um sistema que coloque as encomendas numa fila encadeada, quando compradas, e as retire na ordem correta quando for acontecer o processamento e envio
- Simule o funcionamento do site no método main