

TECNOLOGIA EM INFORMÁTICA

Estrutura de Dados

Prof. Glauco da Silva
contato: glaucoslv@gmail.com

Roteiro

- *Objetivos*
- *Listas encadeadas circulares*
- *Ordenação em listas encadeadas*
- *Classes Java para manipulação de vetores e listas*
- *Exercícios*

Roteiro

- ***Objetivos***
- *Listas encadeadas circulares*
- *Ordenação em listas encadeadas*
- *Classes Java para manipulação de vetores e listas*
- *Exercícios*

Objetivos

- *Estudar as listas encadeadas circulares, mostrando seu funcionamento e como manipulá-las*
- *Apresentar a forma de se ordenar elementos em listas encadeadas*
- *Mostrar classes da API Java que são utilizadas para se trabalhar com vetores e listas*

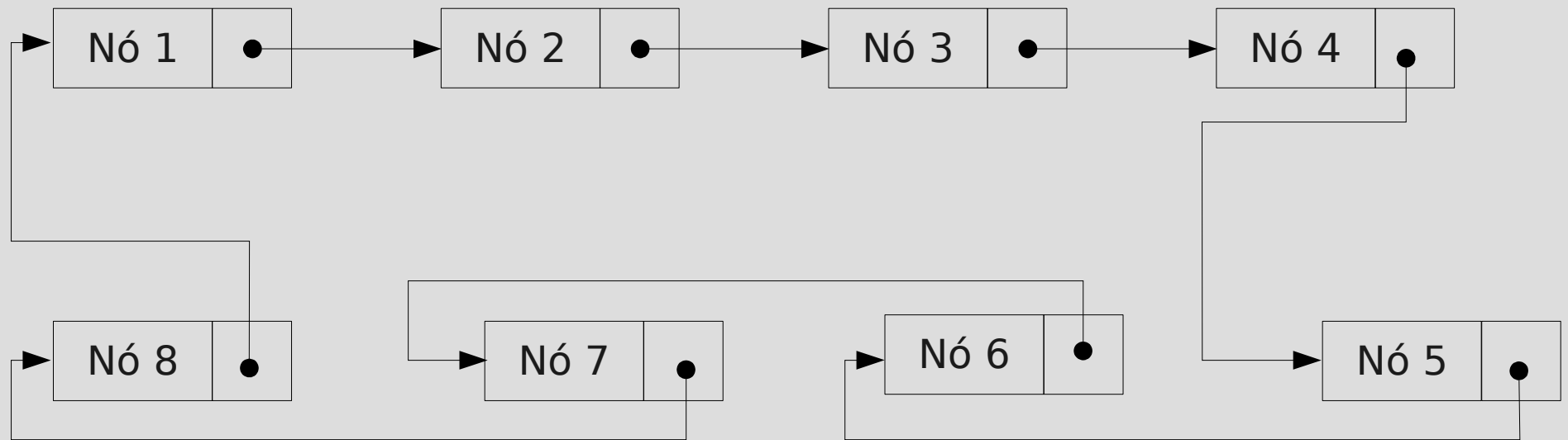
Roteiro

- *Objetivos*
- ***Listas encadeadas circulares***
- *Ordenação em listas encadeadas*
- *Classes Java para manipulação de vetores e listas*
- *Exercícios*

Listas encadeadas circulares

- *Possui o mesmo tipo de nó de uma lista encadeada simples*
- *Não existe cabeça ou cauda, o último nó da lista aponta para o primeiro nó*
- *É possível percorrer toda a lista a partir de qualquer nó da lista*
- *É necessário um nó de controle (cursor)*

Listas encadenadas circulares



Listas encadeadas circulares

- *Métodos de manipulação*

- *adicionar(nodo)* – *Inserere um novo nó, imediatamente após o cursor*

- *remover()* - *remove e retorna o nó que se encontra imediatamente após o cursor*

- *avancar()* - *avança o cursor para o próximo nó da lista*

Listas encadenadas circulares

No.java

```
public class No {
    private int dado;
    private No proximo;

    public No(int d, No nodo){
        dado = d;
        proximo = nodo;
    }

    public int getDado() { return dado; }

    public void setDado(int dado) { this.dado = dado; }

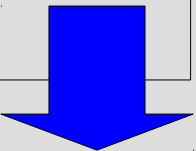
    public No getProximo() { return proximo; }

    public void setProximo(No proximo) { this.proximo = proximo; }
}
```

Listas encadeadas circulares

ListaCircular.java

```
public class ListaCircular {  
    protected No cursor;  
    protected int tamanho;  
  
    public ListaCircular(){  
        cursor = null;  
        tamanho = 0;  
    }  
  
    public No getCursor() { return cursor; }  
  
    public int getTamanho() { return tamanho; }  
  
    public void avancar(){ cursor = cursor.getProximo(); }
```

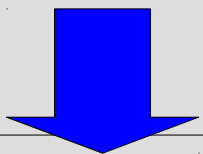


Listas encadeadas circulares

ListaCircular.java

```
public void adicionar(No nodo){
    if (cursor == null){
        nodo.setProximo(nodo);
        cursor = nodo;
    }
    else {
        nodo.setProximo(cursor.getProximo());
        cursor.setProximo(nodo);
    }
    tamanho++;
}

public No remover(){
    No no = cursor.getProximo();
    if (no == cursor) cursor = null;
    else{
        cursor.setProximo(no.getProximo());
        no.setProximo(null);
    }
    tamanho--;
    return no;
}
```



Listas encadeadas circulares

ListaCircular.java

```
public String toString(){
    if (cursor == null)
        return " [ ] ";
    String s = " [..." + cursor.getDado();
    No no = cursor;
    for (avancar(); no != cursor; avancer())
        s += ", " + cursor.getDado();
    return s + "...]";
}
}
```

Roteiro

- *Objetivos*
- *Listas encadeadas circulares*
- ***Ordenação em listas encadeadas***
- *Classes Java para manipulação de vetores e listas*
- *Exercícios*

Ordenação em listas encadeadas

- *Mesma idéia da ordenação em vetores*
- *Utiliza-se o algoritmo InsertionSort para ordenar os elementos de uma lista duplamente encadeada*

Ordenação em listas encadeadas

Algoritmo InsertionSort(L)

Entrada: uma lista duplamente encadeada L com elementos comparáveis

Saída: a lista L com os elementos ordenados

se $L.size() \leq 1$ então retorna

$fim = L.getPrimeiro()$

enquanto fim não for o último nó de L faça

$pivo = fim.getNext()$

 Remove o pivo de L

$insercao = fim$

 enquanto $insercao$ não for o cabeçalho e o elemento $insercao$ for maior que o pivo faça

$insercao = insercao.getAnterior()$

 Acrescenta o pivo após $insercao$ em L

se $insercao = fim$ então

$fim = fim.getProximo()$

Ordenação em listas encadeadas

```
public static void ordenar(ListaDuplamenteEncadeada lista){
    if (lista.tam <= 1)
        return;
    NoDupla pivo;
    NoDupla insercao;
    NoDupla fim = lista.getPrimeiro();
    while (fim != lista.getUltimo()){
        pivo = fim.getProximo();
        lista.remover(pivo);
        insercao = fim;
        while (lista.existeAnterior(insercao) &&
            insercao.getDado().compareTo(pivo.getDado()) > 0)
            insercao = insercao.getAnterior();
        lista.adicionaDepois(insercao, pivo);
        if (insercao == fim)
            fim = fim.getProximo();
    }
    System.out.println("Lista ordenada com sucesso.");
}
```

Roteiro

- *Objetivos*
- *Listas encadeadas circulares*
- *Ordenação em listas encadeadas*
- ***Classes Java para manipulação de vetores e listas***
- *Exercícios*

Classes Java

- *Java fornece métodos para se trabalhar com vetores*
- *A classe `java.util.Arrays` manipula vetores com métodos pré-definidos*
- *Alguns métodos:*
 - *`equals(A, B)` – verifica se `A` e `B` são iguais*
 - *`fill(A, x)` – preenche todos os campos de `A` com o elemento `x`*
 - *`sort(A)` – ordena o vetor `A`*

Classes Java

```
public class TesteVetor {
    public static void main(String[] args) {
        int vetor[] = new int[10];
        Random rand = new Random();
        for (int i=0; i<vetor.length; i++)
            vetor[i] = rand.nextInt(100);
        int[] aux = vetor.clone();
        System.out.println("Vetores iguais antes da ordenação-> " +
Arrays.equals(aux, vetor));
        Arrays.sort(aux);
        System.out.println("Vetores iguais após a ordenação-> " +
Arrays.equals(aux, vetor));
        System.out.println("Vetor auxiliar: " + Arrays.toString(aux));
        System.out.println("Vetor original: " + Arrays.toString(vetor));
    }
}
```

Classes Java

- *Java fornece métodos para se trabalhar com listas*
- *Existem duas classes para se trabalhar com listas*
 - *ArrayList*
 - *A lista é implementada como um vetor de tamanho variável*
 - *LinkedList*
 - *A lista é implementada com células duplamente encadeadas*

Classes Java

- *Alguns métodos da classe List*
 - *add(int i, Object O) – Insere o objeto O na posição i*
 - *Object get(int i) – Recupera o objeto da posição i*
 - *int indexOf(Object O) – Retorna o índice da primeira ocorrência de O*

Classes Java

- *Alguns métodos da classe List*
 - *void clear() – Remove todos os elementos da lista*
 - *boolean contains(Object O) – Retorna true se a lista contém o objeto O*
 - *ListIterator listIterator() – Retorna um Iterator para a lista*

Roteiro

- *Objetivos*
- *Listas encadeadas circulares*
- *Ordenação em listas encadeadas*
- *Classes Java para manipulação de vetores e listas*
- ***Exercícios***

Exercícios

- *Ler 5 elementos e carregar em um ArrayList, em seguida descobrir qual é o menor e o maior valor e exibí-los na tela*
- *Ver Lista Anexa*