UniEinstein
CENTRO UNIVERSITÁRIO | COLÉGIO

Listas Estáticas

Docente: Leandro Lopes Taveira
Coordenador: Pedro Ivo Garcia Nunes

Revisão Rápida

O que vimos na Aula 03?

- Dados, Informação, Conhecimento
- Estruturas de Dados e TADs
- Análise de Complexidade (Big O)

Principais Complexidades

- O(1), O(log n), O(n), O(n^2)

Listas Estáticas vs Dinâmicas

Listas Estáticas

- **Tamanho fixo** definido na criação
- **Não pode ser redimensionada**
- **Exemplo:** Arrays em Java

Listas Dinâmicas

- **Tamanho variável** durante a execução
- **Pode crescer ou diminuir**
- **Exemplo:** Listas Encadeadas (próxima aula)

O que são Arrays?

Definição

- **Estrutura de dados** que armazena uma **coleção de elementos** do mesmo tipo.
- **Posições de memória contíguas** (uma ao lado da outra).
- **Acesso por índice** (posição numérica)

Sobre os Arrays

Características

- **Índices começam em 0** (zero-based)
- **Tamanho fixo** após a criação
- **Elementos do mesmo tipo**

Declaração e Inicialização

Declaração

```
int[] numeros;          // Preferido
int numeros[];          // Alternativo
```

Inicialização

```
// Método 1: Definir tamanho
numeros = new int[5];    // Array de 5 inteiros (valores padrão: 0)

// Método 2: Inicializar com valores
int[] numeros = {10, 20, 30, 40, 50};

// Método 3: Combinado
int[] numeros = new int[]{10, 20, 30, 40, 50};
```

Acesso a Elementos

Sintaxe

```
nomeArray[indice]
```

Exemplo

```
int[] numeros = {10, 20, 30, 40, 50};

System.out.println(numeros[0]);  // 10 (primeiro elemento)
System.out.println(numeros[2]);  // 30 (terceiro elemento)
System.out.println(numeros[4]);  // 50 (último elemento)

// Modificar elemento
numeros[1] = 25;
System.out.println(numeros[1]);  // 25
```

Percorrendo Arrays

For Tradicional

```java
int[] numeros = {10, 20, 30, 40, 50};

for (int i = 0; i < numeros.length; i++) {
    System.out.println("Posição " + i + ": " + numeros[i]);
}
```

Enhanced For (For-Each)

```java
int[] numeros = {10, 20, 30, 40, 50};

for (int numero : numeros) {
    System.out.println(numero);
}
```

Tradicional vs Enhanced For (For-Each)

Quando usar cada um?

- **For tradicional**: Quando precisa do índice
- **For-each**: Quando só precisa do valor

Operações

Vamos criar métodos para realizar as seguintes ações em um array.

- Inserção
- Remoção
- Busca

Os métodos serão criados diretamente na classe Main e usaremos a palavra reservada ‘static’ neles.

Operações - Inserção

Problema

Arrays têm tamanho fixo. Como "inserir" um elemento?

```
public static int[] inserir(int[] array, int elemento, int posicao) {  /**code**/   }
```

Solução

1. Criar um novo array com um tamanho maior que o passado por parâmetro
2. Copiar elementos antes da posição
3. Inserir novo elemento
4. Copiar elementos após a posição

Operações - Inserção - Solução

```java
public static int[] inserir(int[] array, int elemento, int posicao) {  1 usage
    int[] novoArray = new int[array.length + 1];
    for (int i = 0; i < posicao; i++) {// Copiar elementos antes da posição
        novoArray[i] = array[i];
    }
    novoArray[posicao] = elemento;// Inserir novo elemento
    for (int i = posicao; i < array.length; i++) {// Copiar elementos após a posição
        novoArray[i + 1] = array[i];
    }
    return novoArray;
}
```

Operações - Remoção

Problema

Arrays têm tamanho fixo. Como "remover" um elemento sem deixar espaço em branco?

```
public static int[] remover(int[] array, int posicao) {  /**code**/   }
```

Solução

1. Criar um novo array com um tamanho menor que o passado por parâmetro
2. Copiar elementos antes da posição
3. Copiar elementos após a posição (deslocando)

Operações - Remoção - Solução

```java
public static int[] remover(int[] array, int posicao) {  no usages
    int[] novoArray = new int[array.length - 1];
    for (int i = 0; i < posicao; i++) {// Copiar elementos antes da posição
        novoArray[i] = array[i];
    }
    for (int i = posicao + 1; i < array.length; i++) {// Copiar elementos após a posição
        novoArray[i - 1] = array[i];
    }
    return novoArray;
}
```

Operações - Busca

Problema

Existe uma maneira rápida de procurar um elemento por um array?

```
public static int buscar(int[] array, int elemento) {  /**code**/   }
```

Solução

1. Percorrer o array.
2. Comparar com o número fornecido
3. Se encontrar retorna.
4. Se não encontrar retorna -1

Operações - Busca - Solução

```java
public static int buscar(int[] array, int elemento) {  1 usage
    for (int i = 0; i < array.length; i++) {
        if (array[i] == elemento) {
            return i; // Retorna o índice
        }
    }
    return -1; // Não encontrado
}
```

Exercício 1: Removendo a Primeira Ocorrência

Objetivo: Combinar a busca com a remoção, um cenário de uso muito comum.

Enunciado: Usando os métodos `buscar` e `remover` que você já implementou nos slides, crie um programa principal (`main`) que faça o seguinte:

1. Declare um array de `String` com alguns nomes, incluindo nomes repetidos. Ex: `{"Ana", "Pedro", "Luiza", "Pedro", "Carlos"}`.
2. Peça ao usuário para digitar um nome que ele deseja remover.
3. Use o método `buscar` para encontrar o índice da primeira ocorrência do nome digitado.
4. Se o nome for encontrado, use o método `remover` para criar um novo array sem esse nome.
5. Imprima o array resultante. Se o nome não for encontrado, informe o usuário.

Exercício 2: Inserção Ordenada

Objetivo: Desafiar a lógica, pensar sobre a posição correta para inserir um elemento a fim de manter a ordem.
Enunciado: Crie uma função estática chamada `inserirOrdenado` que recebe um array de inteiros já ordenado e um novo número para inserir. O método deve retornar um novo array que também esteja ordenado.

Regras:

- Você não pode simplesmente inserir o elemento e depois ordenar o array.
- Você deve primeiro encontrar a posição correta onde o novo elemento se encaixa e então usar uma lógica de inserção (similar à do slide) para colocá-lo lá.

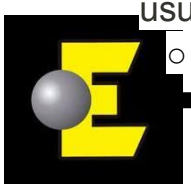
Exercício de Menu 1: Gerenciador de Lista de Tarefas (String)

Enunciado:Crie um programa que gerencia uma lista de tarefas. O programa deve exibir um menu e permitir que o usuário execute as seguintes ações até que ele decida sair:

1. Adicionar Tarefa: Pede ao usuário uma nova tarefa e a insere no final da lista.
2. Concluir Tarefa: Pede ao usuário o nome da tarefa a ser concluída e a remove da lista.
3. Listar Tarefas: Exibe todas as tarefas pendentes, numeradas.
4. Buscar Tarefa: Pede ao usuário um termo e verifica se alguma tarefa com esse nome existe na lista.
5. Sair: Encerra o programa.

Requisitos Técnicos:

- Use um `array` de `String` para armazenar as tarefas. Comece com um array vazio ou com algumas tarefas de exemplo.
- Reutilize os métodos estáticos `inserir`, `remover` e `buscar` adaptados para `String`.
- Use um loop `while` para manter o menu ativo e um `switch` ou `if-else if` para tratar a escolha do usuário.
    - Utilize a classe `Scanner` para ler a entrada do usuário.



Arrays Multidimensionais

Conceito

- Arrays de arrays (matrizes)
- Úteis para representar tabelas, grades, etc.

Declaração

```
int[][] matriz;                    // Declaração
matriz = new int[3][4];           // 3 linhas, 4 colunas

// Ou diretamente
int[][] matriz = new int[3][4];

// Com inicialização
int[][] matriz = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

Manipulando Matrizes

Acesso a Elementos

```java
int[][] matriz = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

System.out.println(matriz[0][0]);  // 1 (primeira linha, primeira coluna)
System.out.println(matriz[1][2]);  // 6 (segunda linha, terceira coluna)

// Modificar elemento
matriz[2][1] = 99;
System.out.println(matriz[2][1]);  // 99
```

Percorrendo Matriz

```
for (int i = 0; i < matriz.length; i++) {           // Linhas
    for (int j = 0; j < matriz[i].length; j++) {    // Colunas
        System.out.print(matriz[i][j] + " ");
    }
    System.out.println(); // Nova linha
}
```

Exercício 1: Encontrar o Maior Valor e sua Posição

Enunciado: Crie uma função que receba uma matriz de inteiros e encontre o maior valor presente nela. O programa deve imprimir não apenas o maior valor, mas também a sua posição (índice da linha e da coluna).

Exemplo:

Plain Text

```
10   5    3

25   32   7

12   8    1
```

Saída Esperada: "O maior valor é 32 e está na posição 1x1."

Exercício 2: Média de Cada Linha

Enunciado: Crie um programa que declare uma matriz 4x3 (4 alunos, 3 notas). Preencha a matriz com notas. Em seguida, calcule e imprima a média das notas de cada aluno (cada linha da matriz).

Exemplo: Para a matriz de notas

```
7.0  8.0  9.0   // Aluno 1

5.0  6.5  7.5   // Aluno 2

9.5  9.0  10.0  // Aluno 3

4.0  5.0  6.0   // Aluno 4
```

Saída Esperada:

```
Média do Aluno 1: 8.0

Média do Aluno 2: 6.5

Média do Aluno 3: 9.5

Média do Aluno 4: 5.0
```

Exercício 3: Jogo da Velha - Verificação de Estado

Enunciado: Crie um programa que represente o tabuleiro de um Jogo da Velha usando uma matriz 3x3 de `char`.

A matriz pode conter 'X', 'O' ou ' ' (espaço para vazio). Crie uma função chamada `verificarGanhador` que recebe o tabuleiro e determina o estado do jogo:

- Retorna 'X' se o jogador X venceu.
- Retorna 'O' se o jogador O venceu.
- Retorna 'E' se deu empate (tabuleiro cheio sem vencedor).
- Retorna ' ' se o jogo ainda está em andamento.

Requisitos:

- A função deve verificar todas as 8 possibilidades de vitória (3 linhas, 3 colunas, 2 diagonais).
- Um empate ocorre se não houver vencedor e não houver mais espaços vazios.

Exemplo de Tabuleiro (vitória de 'X' na primeira linha):

```
'X'  'X'  'X'

'O'  ' '  'O'

' '  ' '  ' '
```

Saída Esperada (teste 1): "Resultado Tabuleiro: E"

Saída Esperada (teste 2): "Resultado Tabuleiro: X"

Vantagens dos Arrays

Acesso Rápido

- O(1) para acessar qualquer elemento por índice
- Exemplo: `array[100]` é tão rápido quanto `array[0]`

Simplicidade

- Fácil de usar para coleções de tamanho conhecido
- Sintaxe simples e intuitiva

Eficiência de Cache

- Elementos contíguos na memória
- Melhor performance para percorrer sequencialmente

Desvantagens dos Arrays

Tamanho Fixo

- Não pode ser redimensionado após a criação
- Problema: E se precisar de mais espaço?

Inserção/Remoção Lenta

- O(n) para inserir/remover no meio
- Motivo: Precisa deslocar elementos

Desperdício de Memória

- Pode alocar mais espaço do que necessário
- Exemplo: Array de 1000 elementos, mas só usa 10

Resumo de Complexidades

Operação	Complexidade	Explicação
Acesso por índice	O(1)	Acesso direto
Busca	O(n)	Pode precisar percorrer todo o array
Inserção no final	O(1)*	*Se houver espaço
Inserção no meio	O(n)	Precisa deslocar elementos
Remoção no final	O(1)	Apenas "remove" o último
Remoção no meio	O(n)	Precisa deslocar elementos

Dúvidas?


TEMPO SOBRANDO
Uhhhhhhhhhhhuuuuuu!!!!!!
Será?

Exercício de Menu 2: Controle de Estoque de Produtos (int)

Enunciado: Desenvolva um sistema de console para controlar o estoque de produtos por código. O sistema deve manter uma lista de códigos de produtos (números inteiros) e oferecer as seguintes opções:

1. Cadastrar Produto: Pede um código de produto. Antes de adicionar, verifica se o código já não existe na lista (usando o método `buscar`). Se não existir, adiciona. Se já existir, informa o usuário.
2. Vender Produto: Pede o código de um produto para "vender" e o remove da lista. Se o produto não for encontrado, informa o usuário.
3. Verificar Estoque: Pede o código de um produto e informa se ele está ou não no estoque (usando o método `buscar`).
4. Listar Produtos: Exibe todos os códigos de produtos atualmente no estoque.
5. Sair: Encerra o programa.

Requisitos Técnicos:

- Use um `array` de `int` para armazenar os códigos dos produtos.
- Reutilize os métodos estáticos `inserir`, `remover` e `buscar` para `int`.
- A lógica de "Cadastrar Produto" é o ponto-chave, pois exige uma busca *antes* da inserção para evitar duplicatas.
    a. Use um loop `while`, um `switch` (ou `if-else if`) e a classe `Scanner`.

Exercício de Menu 3: Gerenciador de Vagas de Estacionamento

Enunciado: Desenvolva um sistema para gerenciar as 10 vagas de um estacionamento. As vagas são numeradas de 0 a 9. O sistema deve usar um array de `int` onde o valor `0` significa "vaga livre" e um número de placa (um `int`, ex: 4567) significa "vaga ocupada".

1. Estacionar: Pede ao usuário o número da vaga (0-9) e o número da placa do carro. Se a vaga estiver livre (`0`), o sistema a ocupa com o número da placa. Se estiver ocupada, informa o usuário.
2. Liberar Vaga: Pede o número da vaga. Se estiver ocupada, libera a vaga (coloca o valor `0` nela). Se já estiver livre, informa o usuário.
3. Buscar Carro: Pede o número da placa e informa em qual vaga (índice) o carro está. Se não encontrar, informa que o carro não está no estacionamento.
4. Listar Vagas: Exibe o estado de todas as 10 vagas, mostrando o índice e se está "Livre" ou "Ocupada pela placa [número]".
5. Sair: Encerra o programa.

Requisitos Técnicos:

- Inicie um array de `int` de tamanho 10, com todos os elementos valendo `0`.
- A operação "Estacionar" não usa o método `inserir` tradicional. Ela acessa e modifica um índice diretamente: `vagas[numeroDaVaga] = placa;`.
- A operação "Liberar Vaga" também modifica diretamente: `vagas[numeroDaVaga] = 0;`.
- A operação "Buscar Carro" é uma implementação clássica do método `buscar`.

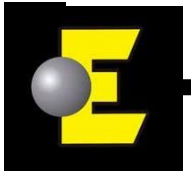
Exercício de Menu 4: Controle de Amigo Secreto

Enunciado: Crie um sistema para gerenciar um amigo secreto. O programa deve manter duas listas (dois arrays de `String`): uma para os participantes e outra para quem cada participante tirou.

1. Adicionar Participante: Pede um nome e o adiciona à lista de participantes. A lista de quem ele tirou deve ser preenchida com um valor padrão (ex: `"[PENDENTE]"`).
2. Realizar Sorteio: Esta é a função principal. Ela deve embaralhar a lista de participantes e atribuir a cada um uma pessoa diferente para presentear. Um participante não pode tirar a si mesmo.
3. Consultar Amigo Secreto: Pede o nome de um participante e, se o sorteio já foi realizado, revela secretamente quem ele tirou.
4. Listar Participantes: Mostra todos que estão na brincadeira.
5. Sair: Encerra o programa.

Requisitos Técnicos:

- Use dois arrays: `String[] participantes` e `String[] amigoSorteado`.
- Sorteio: Esta é a parte desafiadora. Uma abordagem: a. Crie uma cópia do array de participantes. b. Embaralhe essa cópia usando `Collections.shuffle()` (requer converter para `List` e depois de volta para array) ou um algoritmo de Fisher-Yates. c. Percorra a lista de participantes original. Para cada participante `i`, verifique se o amigo sorteado na cópia embaralhada na posição `i` é ele mesmo. Se for, troque com o próximo da lista (`i+1`) para evitar que alguém tire a si mesmo. d. Atribua a lista embaralhada e corrigida ao array `amigoSorteado`.
- Mantenha um `boolean sorteioRealizado = false;` para controlar o estado do programa.



Carpe Diem