Arrays em Java

Vetores (Arrays Unidimensionais)

1. O Que é um Array?

Um array (vetor) é uma estrutura de dados que armazena um conjunto de elementos do mesmo tipo, organizados sequencialmente na memória. Em Java, arrays são objetos e possuem tamanho fixo definido no momento da criação.

> 💡 **Conceito-chave**
> Pense num array como uma fileira de caixas numeradas, onde todas as caixas guardam o mesmo tipo de objeto. Cada caixa tem um índice que começa em 0.

1.1 Características Fundamentais

- Tipo homogêneo: todos os elementos são do mesmo tipo de dado.
- Tamanho fixo: definido na criação e não pode ser alterado.
- Acesso por índice: primeiro elemento no índice 0, último em length-1.
- Armazenamento contíguo: os elementos ficam em posições consecutivas de memória.
- São objetos: herdam os métodos da classe Object e possuem o atributo length.

1.2 Diagrama Visual de Memória

Considere a declaração: int[] notas = {85, 92, 78, 95, 60};

	notas[0]	notas[1]	notas[2]	notas[3]	notas[4]
Valor →	**85**	**92**	**78**	**95**	**60**
Índice →	0	1	2	3	4

O array possui length = 5. O índice válido vai de 0 a 4.

2. Declaração, Criação e Inicialização

2.1 Formas de Declarar um Array

Java oferece diferentes sintaxes para trabalhar com arrays:

Declaração e criação separadas

```
// Forma 1: declarar e criar em etapas separadas

int[] notas;              // declara a variável (ainda null)

notas = new int[5];


// Forma 2: declarar e criar ao mesmo tempo

double[] precos = new double[10];


// Forma 3: declarar, criar e inicializar com valores

String[] nomes = {"Ana", "Bruno", "Carlos"};


// Forma 4: inicialização explícita com new

int[] idades = new int[] {20, 25, 30};
```

> 💡 **Valores Padrão**
> Ao criar um array com new sem inicializar, Java preenche com valores padrão: 0 para int/long/short/byte, 0.0 para double/float, false para boolean, caractere nulo ('\0') para char, e null para objetos (String, etc.).

2.2 Acessando e Modificando Elementos

```
int[] notas = new int[5];


// Atribuindo valores individualmente
```

```java
notas[0] = 85;

notas[1] = 92;

notas[2] = 78;


// Lendo um valor

int primeiroNota = notas[0];

System.out.println(primeiroNota);
// Saída: 85


// Usando length para saber o tamanho

System.out.println(notas.length);
// Saída: 5
```

> 💡 **Cuidado: ArrayIndexOutOfBoundsException**
> Tentar acessar um índice fora do intervalo válido (menor que 0 ou maior/igual a length) lança ArrayIndexOutOfBoundsException em tempo de execução. Sempre verifique os limites antes de acessar posições dinâmicas.

3. Percorrendo Arrays

3.1 Com o Laço for Tradicional

O for tradicional é ideal quando precisamos do índice durante a iteração (por exemplo, para acessar posições específicas ou percorrer ao contrário).

```java
int[] notas = {85, 92, 78, 95, 60};


// Percorrendo do início ao fim

for (int i = 0; i < notas.length; i++) {

    System.out.println("Nota[" + i + "] = " + notas[i]);
```

```
}

// Percorrendo ao contrário
for (int i = notas.length - 1; i >= 0; i--) {
    System.out.print(notas[i] + " ");
}
// Saída: 60 95 78 92 85
```

3.2 Com o Enhanced for (for-each)

O for-each é mais limpo e legível quando só precisamos dos valores, sem a necessidade do índice.

```
int[] notas = {85, 92, 78, 95, 60};

for (int nota : notas) {
    System.out.println("Nota: " + nota);
}
```

3.3 Comparativo: for vs for-each

Característica	for tradicional	for-each
Acesso ao índice	✅ Sim (variável i)	❌ Não disponível
Percurso inverso	✅ Sim	❌ Não diretamente

4. Operações Comuns com Arrays

4.1 Cálculo de Soma e Média

```
int[] notas = {85, 92, 78, 95, 60};

int soma = 0;
```

```java
for (int nota : notas) {
    soma += nota;
}

double media = (double) soma / notas.length;
System.out.println("Soma: " + soma);
System.out.println("Média: " + media);
// Saída: Soma: 410 | Média: 82.0
```

4.2 Encontrando Maior e Menor Valor

```java
int[] notas = {85, 92, 78, 95, 60};
int maior = notas[0];
int menor = notas[0];

for (int nota : notas) {
    if (nota > maior) maior = nota;
    if (nota < menor) menor = nota;
}

System.out.println("Maior: " + maior + " | Menor: " + menor);
// Saída: Maior: 95 | Menor: 60
```

4.3 Buscando um Elemento (Busca Linear)

```
String[] nomes = {"Ana", "Bruno", "Carlos", "Diana"};

String busca = "Carlos";

int posicao = -1;


for (int i = 0; i < nomes.length; i++) {

    if (nomes[i].equals(busca)) {

        posicao = i;

        break;

    }

}


if (posicao != -1)

    System.out.println(busca + " encontrado no índice " + posicao);

else
    System.out.println("Não encontrado");

// Saída: Carlos encontrado no índice 2
```

4.4 Usando Arrays.sort() e Arrays.toString()

```
import java.util.Arrays;


int[] nums = {5, 2, 8, 1, 9, 3};


// Exibir array como string

System.out.println(Arrays.toString(nums));

// Saída: [5, 2, 8, 1, 9, 3]
```

```java
// Ordenar o array (modifica o original)

Arrays.sort(nums);

System.out.println(Arrays.toString(nums));

// Saída: [1, 2, 3, 5, 8, 9]
```

5. Passagem de Arrays para Métodos

Em Java, arrays são passados por referência. Isso significa que quando você passa um array para um método, o método recebe o endereço do array original — não uma cópia. Modificações dentro do método afetam o array original.

```java
public class ExemploMetodos {


    // Método que calcula a média de um array

    public static double calcularMedia(int[] valores) {

        double soma = 0;

        for (int v : valores) soma += v;

        return soma / valores.length;

    }


    // Método que dobra todos os valores (modifica o original!)

    public static void dobrarValores(int[] arr) {

        for (int i = 0; i < arr.length; i++)

            arr[i] *= 2;

    }


    public static void main(String[] args) {

        int[] notas = {10, 8, 9, 7};
```

```
        System.out.println(calcularMedia(notas));  // 8.5

        dobrarValores(notas);

        System.out.println(Arrays.toString(notas)); // [20, 16, 18, 14]

    }

}
```

> 💡 **Passagem por Referência vs Cópia**
>
> Se você não quer que um método altere o array original, use Arrays.copyOf() para criar uma cópia antes de passar: int[] copia = Arrays.copyOf(original, original.length);

6. Exercícios Práticos

Resolva os exercícios a seguir para praticar os conceitos apresentados.

✏ Exercício 1: Estatísticas de Turma

Dado um array com as notas de uma turma, implemente um programa completo.

1. Declare e inicialize um array com 6 notas entre 0 e 100.
2. Calcule e exiba: soma, média, maior nota e menor nota.
3. Exiba quantos alunos ficaram acima da média (nota > média calculada).
4. Desafio: exiba as notas em ordem crescente usando Arrays.sort().

✏ Exercício 2: Inversão de Array

Escreva um método que inverte a ordem dos elementos de um array sem usar outro array.

1. Crie um método static void inverter(int[] arr).
2. Use apenas variável auxiliar temporária para trocar elementos.
3. Dica: percorra o array trocando arr[i] com arr[length-1-i] até o meio.
4. Teste com: {1, 2, 3, 4, 5} → esperado: {5, 4, 3, 2, 1}.

✏ Exercício 3: Frequência de Elementos

Dado um array de inteiros entre 1 e 5, conte quantas vezes cada número aparece.

1. Declare int[] valores = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5};
2. Crie um array contagem de tamanho 10 (índice = valor).
3. Percorra valores e incremente contagem[valores[i]].
4. Exiba: 'O número X aparece Y vez(es).' para cada número que aparece.

✏ Exercício 4: Verificação de Palíndromo Numérico

Um array é palíndromo se lido da esquerda para direita é igual ao contrário.

1. Crie um método static boolean ehPalindromo(int[] arr).
2. Compare arr[0] com arr[n-1], arr[1] com arr[n-2], etc.
3. Retorne false assim que encontrar uma diferença; retorne true se checar tudo.
4. Teste com: {1,2,3,2,1} → true | {1,2,3,4,5} → false.

✏ Exercício 5: Rotação de Array (Desafio)

Rotacionar um array à direita em k posições: o último elemento vai para o início.

1. Entrada: {1, 2, 3, 4, 5} com k=2 → Saída: {4, 5, 1, 2, 3}.
2. Implemente usando um array auxiliar para armazenar os últimos k elementos.
3. Depois desloque os demais elementos k posições para a direita.
4. Por fim, coloque os k elementos salvos no início do array.

7. Referência Rápida — Sintaxe de Arrays

Operação	Sintaxe / Exemplo
Declarar e criar	`int[] arr = new int[5];`
Inicializar com valores	`int[] arr = {1, 2, 3, 4, 5};`
Acessar elemento	`arr[2]   // terceiro elemento`
Modificar elemento	`arr[2] = 99;`
Tamanho do array	`arr.length   // retorna int`
Percorrer com for	`for (int i = 0; i < arr.length; i++)`
Percorrer com for-each	`for (int x : arr)`
Exibir como texto	`Arrays.toString(arr)`