Array e Vetor nível 2 (Avançado)

📘 Aula de 1 Hora – Arrays em Java (Intermediário e Avançado)

A proposta desta aula é elevar o aluno do nível básico ao avançado no uso de arrays, abordando **busca**, **ordenação**, **matrizes**, **operações funcionais**, **arrays multidimensionais irregulares**, **algoritmos clássicos**, **tratamento de exceções**, **cópias**, **análise de desempenho**, entre outros.

✅ EXEMPLO 1 — Encontrar o segundo maior número do array

Problema

Dado um array de inteiros, encontre o **segundo maior valor** sem ordenar o array.

Código

java Copiar código

```java
public class SegundoMaior {
    public static void main(String[] args) {
        int[] numeros = {10, 50, 20, 5, 80, 60};

        int maior = Integer.MIN_VALUE;
        int segundoMaior = Integer.MIN_VALUE;
```

```
        for (int n : numeros) {
            if (n > maior) {
                segundoMaior = maior;
                maior = n;
            } else if (n > segundoMaior && n != maior) {
                segundoMaior = n;
            }
        }

        System.out.println("Segundo maior número: " + segundoMaior);
    }
}
```

Explicação

- Fazemos apenas **uma passada** pelo array.
- Mantemos duas variáveis: `maior` e `segundoMaior`.
- Complexidade: **O(n)** (ideal para grandes volumes).

✅ EXEMPLO 2 — Remover valores duplicados (sem usar Set)

Problema

Remover duplicados de um array mantendo apenas a primeira ocorrência.

Código

java Copiar código

```
import java.util.Arrays;

public class RemoveDuplicados {
    public static void main(String[] args) {
        int[] numeros = {2, 5, 2, 8, 9, 9, 9, 1};
        int[] temp = new int[numeros.length];
        int index = 0;

        for (int i = 0; i < numeros.length; i++) {
            boolean existe = false;
```

```
            for (int j = 0; j < index; j++) {
                if (numeros[i] == temp[j]) {
                    existe = true;
                    break;
                }
            }

            if (!existe) {
                temp[index++] = numeros[i];
            }
        }

        int[] resultado = Arrays.copyOf(temp, index);

        System.out.println(Arrays.toString(resultado));
    }
}
```

Explicação

- Usamos um array temporário que acumula os valores únicos.
- Trabalhamos apenas com arrays (sem Collections).
- Complexidade: **O(n²)** devido aos loops.

✅ EXEMPLO 3 — Rotacionar um array para a direita

Problema

Rotacionar o array N vezes, movendo o último elemento para o início.

Código

java Copiar código

```
import java.util.Arrays;

public class RotacionarArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int rotacoes = 2;

        for (int r = 0; r < rotacoes; r++) {
            int ultimo = arr[arr.length - 1];
```

```
            for (int i = arr.length - 1; i > 0; i--) {
                arr[i] = arr[i - 1];
            }

            arr[0] = ultimo;
        }

        System.out.println(Arrays.toString(arr));
    }
}
```

Explicação

- Rotacionamos manualmente, ideal para ensinar lógica de movimentação.
- Não usa Collections ou streams.

✅ EXEMPLO 4 — Buscar o valor mais próximo de um número alvo

Problema

Dado um array e um número alvo, retornar o número mais próximo.

Código

java Copiar código

```
public class ValorMaisProximo {
    public static void main(String[] args) {
        int[] arr = {3, 8, 15, 17, 22, 27};
        int alvo = 20;

        int maisProximo = arr[0];
        for (int n : arr) {
            if (Math.abs(n - alvo) < Math.abs(maisProximo - alvo)) {
                maisProximo = n;
            }
        }

        System.out.println("Mais próximo de " + alvo + " é: " + maisProximo
```

```
    }
}
```

Explicação

- Comparamos diferenças absolutas.
- Útil para problemas matemáticos e sensores.

✅ EXEMPLO 5 — Ordenação Bubble Sort (manual)

Problema

Ordenar um array **sem usar Arrays.sort()** para mostrar o algoritmo.

Código

java Copiar código

```java
import java.util.Arrays;

public class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {8, 2, 5, 1, 9};

        for (int i = 0; i < arr.length - 1; i++) {
            for (int j = 0; j < arr.length - 1 - i; j++) {
                if (arr[j] > arr[j+1]) {
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }

        System.out.println(Arrays.toString(arr));
    }
}
```

Explicação

- Ensina lógica de comparação e troca.
- Base para entender algoritmos reais como quick e merge sort.

✅ EXEMPLO 6 — Matriz: encontrar a soma das diagonais

Problema

Calcular a soma das diagonais principal e secundária de uma matriz NxN.

Código

java Copiar código

```java
public class SomaDiagonais {
    public static void main(String[] args) {
        int[][] m = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int somaPrincipal = 0;
        int somaSecundaria = 0;

        for (int i = 0; i < m.length; i++) {
            somaPrincipal += m[i][i];
            somaSecundaria += m[i][m.length - 1 - i];
        }

        System.out.println("Diagonal principal: " + somaPrincipal);
        System.out.println("Diagonal secundária: " + somaSecundaria);
    }
}
```

Explicação

- Mostra domínio de arrays bidimensionais.
- Útil para exercícios matemáticos.

✅ EXEMPLO 7 — Matriz irregular (jagged array)

Problema

Criar uma matriz com linhas de tamanhos diferentes.

Código

java Copiar código

```java
public class MatrizIrregular {
    public static void main(String[] args) {
        int[][] matriz = new int[3][];

        matriz[0] = new int[]{1, 2};
        matriz[1] = new int[]{3, 4, 5};
        matriz[2] = new int[]{6};

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

Explicação

- Arrays podem ter linhas de tamanhos diferentes.
- Muito útil em tabelas dinâmicas.

✅ EXEMPLO 8 — Filtrar números pares usando Streams (nível avançado)

Problema

Filtrar apenas números pares de um array.

Código

java Copiar código

```java
import java.util.Arrays;

public class FilterStream {
    public static void main(String[] args) {
        int[] arr = {10, 15, 22, 33, 40};
```

```
        int[] pares = Arrays.stream(arr)
                .filter(n -> n % 2 == 0)
                .toArray();

        System.out.println(Arrays.toString(pares));
    }
}
```

Explicação

- Aborda programação funcional.
- Introduz streams e lambda expressions.
- Código extremamente limpo.

✅ EXEMPLO 9 — Expandir array manualmente (simular ArrayList)

Problema

Adicionar um elemento em um array redimensionando manualmente.

Código

java Copiar código

```java
import java.util.Arrays;

public class ExpandirArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3};

        int[] novoArr = Arrays.copyOf(arr, arr.length + 1);
        novoArr[novoArr.length - 1] = 99;

        System.out.println(Arrays.toString(novoArr));
    }
}
```

Explicação

- Demonstra como o `ArrayList` funciona internamente.
- Ensina cópias e manipulação manual.

✅ EXEMPLO 10 — Verificar se o array está ordenado (ascendente)

Problema

Criar uma função que identifica se o array está ordenado.

Código

java Copiar código

```java
public class VerificarOrdenacao {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 8, 9};

        boolean ordenado = true;

        for (int i = 0; i < arr.length - 1; i++) {
            if (arr[i] > arr[i + 1]) {
                ordenado = false;
                break;
            }
        }

        System.out.println("Está ordenado? " + ordenado);
    }
}
```

Explicação

- Útil para validação e testes.
- Mostra lógica de comparação encadeada.