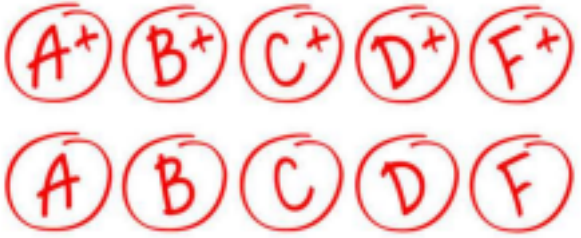
Na aula de hoje

- Conceito de Arrays;
- Conceito de Matrizes;
- Matrizes Unidimensionais e Multidimensionais; - Criação e declaração de Matrizes; - Loop foreach;
- Exemplos de código;
- Exercícios

Arrays e Matrizes
Introdução

Imagine que você precisa guardar as notas de todos os alunos da sua turma. Ao invés de criar várias variáveis separadas, podemos usar um array para



armazenar todas as notas em uma única estrutura.

Ou então, você tem uma loja de roupas e precisa controlar o estoque de cada item. Criar uma variável para cada item seria cansativo e pouco prático, não é mesmo? Arrays nesse caso podem ser úteis para armazenar peças de uma mesma categoria em uma única estrutura.

Arrays e Matrizes Conceito de Array

Exemplos do dia a dia

Um array (ou vetor) é uma coleção de elementos do mesmo tipo de dado, como números, letras ou até mesmo objetos complexos.

Pense em um array como uma lista organizada, onde cada item tem um número de índice que indica sua posição.

Podemos usar esse índice para acessar e modificar cada elemento individualmente.

Importante: alguns critérios de tipo de dado e organização de um vetor podem mudar de uma linguagem para outra.

Lista de compras

Cada item da lista é um elemento do array.

Notas de alunos

As notas de cada aluno são armazenadas em um

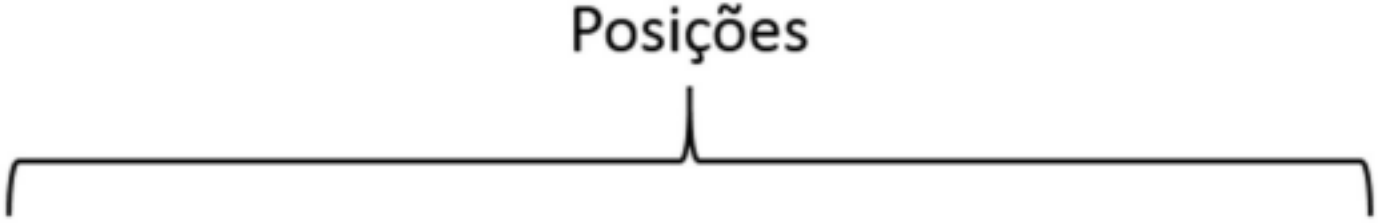
array.

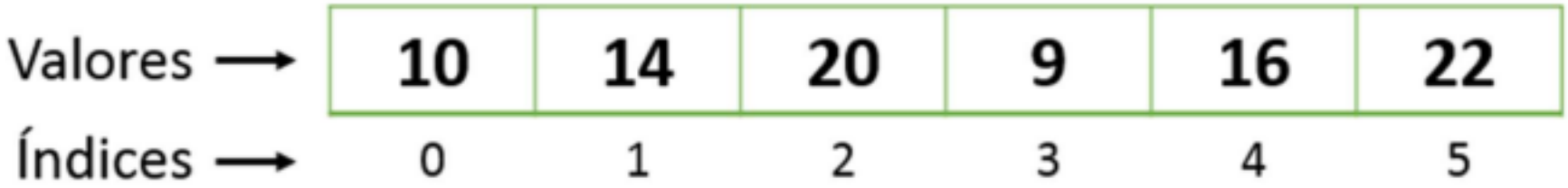
Playlist de músicas
Cada música é um elemento, e podemos acessá-las

pela ordem.

Arrays e Matrizes

Índices e Posições





Arrays e Matrizes
Índices e Posições

Arrays e Matrizes
Arrays

Síntaxe de Declaração
tipo[] nomeDoArray;

Inicialização

Existem duas maneiras principais de inicializar um array: diretamente com valores ou definindo o tamanho do array e atribuindo valores posteriormente.

Inicialização com valores

```
int[] numeros = {1, 2, 3, 4, 5};
```

Inicialização com tamanho:

```
int[] numeros = new int[5]; // Cria um array de inteiros com 5 posições
```

Revisão de Conceitos
Arrays

Acessando e Modificando Elementos do Array

Os elementos de um array são acessados através de seus índices. Por exemplo, o primeiro elemento de um array números pode ser acessado com numeros[0]. Para modificar um elemento, basta atribuir um novo valor ao índice desejado.

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

// Acessar um valor
```

```
System.out.println(numeros[2]); // Saída: 30

// Modificar um valor
numeros[2] = 100;
System.out.println(numeros[2]); // Saída: 100
```

Revisão de Conceitos
Arrays

Iterando sobre Arrays
Uma das operações mais comuns com arrays é iterar (percorrer) seus elementos. Existem várias maneiras de fazer isso em Java:

For loop tradicional

```
int[] numeros = {10, 20, 30, 40, 50};
for (int i = 0; i < numeros.length; i++) {
        System.out.println("Elemento na posição " + i + ": " + numeros[i]); }
```

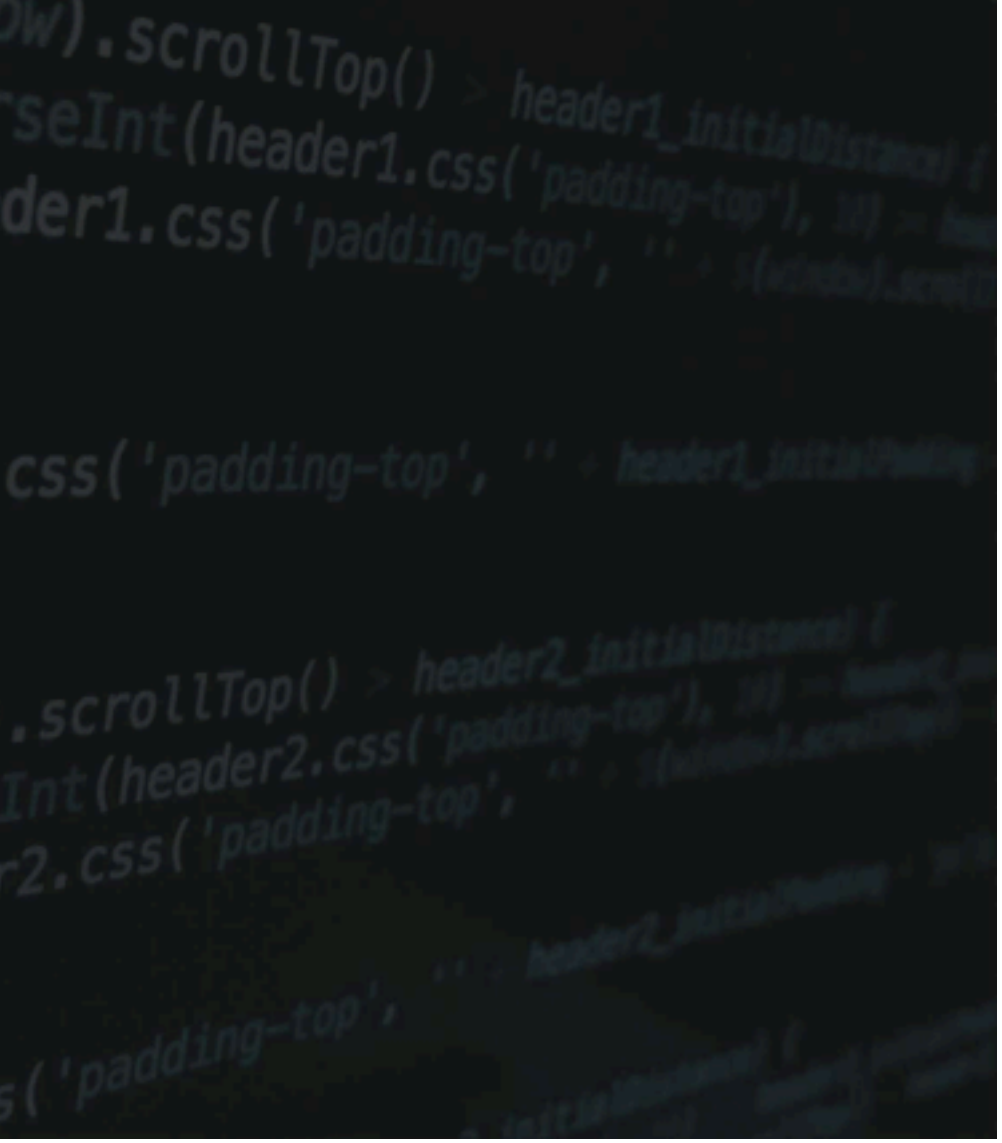
Enhanced for loop (for-each)

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

Revisão de Conceitos Arrays

Calculando a média de uma lista de números

```
import java.util.Scanner; public class MediaArray {

// Lendo os números double soma = 0; for (int i = 0; i <
        public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
soma += numeros[i];}
            // Definindo o tamanho do array

            System.out.print("Quantos números
você quer inserir? ");
            int tamanho = scanner.nextInt();

            // Criando o array double[] numeros = new
double[tamanho];
                                                tamanho; i++) {
                                                    System.out.print("Digite o
                                        número " + (i + 1) + ": "); numeros[i] =
                                        scanner.nextDouble();
                                        // Calculando a média
                                                    double media = soma / tamanho;

                                                System.out.println("A média é: "+ media);

                                                scanner.close(); }

                                        }
```



Arrays e Matrizes Conceito de Matriz

Uma matriz é um tipo especial de array multidimensional com duas dimensões, comumente representada como uma tabela de linhas e colunas.

Existem dois tipos de matrizes:

- Unidimensional: Uma única linha de elementos, como uma lista.

Cada elemento em uma matriz é acessado por um par de índices, representando sua posição na linha e coluna.

- Multidimensional: Várias linhas e colunas, como uma tabela.

Podemos usar matrizes para representar dados mais complexos, como uma tabela de notas de alunos com várias matérias.

	0	1	2	3
0	"K"	"S"	"A"	"P"
1	"M"	"Q"	"C"	"Z"
2	"W"	"G"	"L"	"N"
3	"X"	"S"	"J"	"R"

Revisão de Conceitos

Arrays Síntaxe de Declaração

Acessando elementos de uma matriz:

```
System.out.println(matriz[1][2]);
```

Iterando sobre uma matriz - Repetição aninhada

```
int[][] matriz = new int[3][3];
// Matriz 3x3
```

Inicialização de uma matriz int[][] matriz = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
//Saída: 6

matriz[i].length; j++) {System.out.print(matriz[i][j] + " ");}
System.out.println(); //Quebra de linha para exibir a matriz formatada}

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

Revisão de Conceitos Arrays

Controle de Estoque em Lojas

```
import java.util.Scanner;

public class ControleEstoque {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		int[][] estoque = new int[3][4];
```

```
System.out.println("Digite a quantidade de estoque para cada produto em cada loja:");
for (int i = 0; i < 3; i++) {
        System.out.println("Loja " + (i + 1) + ":"); for (int j = 0; j < 4; j++) {
                System.out.print("Produto " + (j + 1) + ": "); estoque[i][j] = scanner.nextInt();
        }
}
```

Revisão de Conceitos Arrays

Controle de Estoque em Lojas

```
// Exibindo a matriz de estoque
System.out.println("\nEstoque das lojas:"); for (int i = 0; i < 3; i++) {
        System.out.print("Loja " + (i + 1) + ": "); for (int j = 0; j < 4; j++) {
                System.out.print(estoque[i][j] + " "); }
        System.out.println(); }

          int[] totalPorProduto = new int[4];
for (int j = 0; j < 4; j++) {
        for (int i = 0; i < 3; i++) {
                totalPorProduto[j] += estoque[i][j]; }
}
```

Revisão de Conceitos Arrays

Controle de Estoque em Lojas

```
            // Exibindo o total de cada produto
            System.out.println("\nTotal de cada produto em todas as lojas:"); for (int j = 0; j < 4; j++) {
                                        System.out.println("Produto " + (j + 1) + ": " + totalPorProduto[j]);
            }

            scanner.close();
        }
}
```

Exercícios - Array

Arrays e Matrizes

Exercícios

1. Lista de Compras

- Crie um programa que permita ao usuário inserir o nome de 5 itens que deseja comprar no supermercado.
- Armazene os nomes dos itens em um array de strings.
- Após a entrada dos itens, exiba a lista de compras na tela, um item por linha.

2. Controle de Gastos Diários

- Crie um programa que registra os gastos diários do usuário durante uma semana.
- Utilize um array de double para armazenar os gastos de cada dia. - Peça ao usuário que insira o gasto de cada dia da semana.
- Ao final, calcule e exiba o gasto total da semana e a média de gastos por dia.

Exercício - Matrizes

Arrays e Matrizes

Exercícios

3. Controle de estoque

- Crie um programa para controlar o estoque de uma loja com 5 produtos e 3 filiais.
- Utilize uma matriz de inteiros para representar o estoque, onde as linhas representam os produtos e as colunas representam as filiais.
- Peça ao usuário que insira a quantidade em estoque de cada produto em cada filial.
- Exiba o estoque de cada produto, mostrando a quantidade disponível em cada filial.
- Calcule e exiba o estoque total de cada produto (soma das quantidades em todas as filiais).

Tratamento de Exceções Introdução

Try-Catch

Quando programamos, é comum que erros ocorram durante a execução de um programa.

Esses erros são chamados de exceções e, se não forem tratados, podem fazer com que o programa falhe ou exiba resultados inesperados.

Por isso, usamos os blocos try e catch para tratar exceções e garantir que o programa continue executando mesmo diante de erros.

Tratamento de Exceções

Exeções

O que são Exceções?

Exceções são eventos que ocorrem durante a execução de um programa e que interrompem o fluxo normal da execução. As exceções geralmente surgem de situações inesperadas, como:

- Divisão por zero.
- Acesso a um índice inexistente de um array.

- Tentativa de abrir um arquivo que não existe.

Exceções em Java são objetos que representam esses eventos de erro. Quando uma exceção ocorre, ela é "lançada" pelo programa e pode ser capturada e tratada para que o programa não seja interrompido.

Tratamento de Exceções

Escopo de um Bloco

EXEMPLO

```
public class EscopoDeBloco {
    public static void main(String[] args) {
                                        int x = 10; // Variável x declarada no escopo da função main

        if (x > 5) {
            int y = 20; // Variável y declarada no escopo do bloco if System.out.println(x + y); // 30
        }

        // System.out.println(y); // Erro: y não está definido fora do bloco if

        for (int i = 0; i < 5; i++) { // Variável i declarada no escopo do loop forSystem.out.println(i);
        }

          System.out.println(i); // Erro: i não está definido fora do loop for}
}
```

Tratamento de Exceções
Try e catch

O Bloco try e catch
Para tratar exceções, usamos os blocos try e catch.

Bloco try: O código que pode gerar uma exceção é colocado dentro do bloco try. Se uma exceção ocorrer nesse bloco, ela será capturada.

Bloco catch: O código dentro do bloco catch é executado se uma exceção ocorrer no bloco try. Ele serve para tratar a exceção e evitar que o programa pare.

Sintaxe básica

```
try {
        // Código que pode gerar uma exceção
} catch (TipoDaExcecao e) {
        // Código para tratar a exceção
}
```

Exemplo de código - Divisão por zero Tratamento de Exceções

Divisão por zero

```
import java.util.Scanner;

public class ExemploDivisaoPorZero {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

try {
System.out.print("Digite o numerador: ");
int numerador = scanner.nextInt();

System.out.print("Digite o denominador: "); int denominador = scanner.nextInt();
```

Exemplo de código - Divisão por zero Tratamento de Exceções

Divisão por zero

```
// Este código pode lançar uma ArithmeticException se o denominador for zero
int resultado = numerador / denominador;
System.out.println("Resultado: " + resultado);

} catch (ArithmeticException e) {
// Este código é executado se uma exceção ocorrer System.out.println("Erro: Não é possível
dividir por zero!");}
```

scanner.close();
}
}

O Bloco Finally

// Código que pode gerar exceções

O bloco finally é uma parte opcional da estrutura try-catch e é usado para definir um código que deve ser executado sempre, independentemente de uma exceção ter sido lançada ou não. Ele é comumente utilizado para liberar recursos (como

Tratamento de Exceções

Finally

fechar arquivos ou conexões com o banco de dados).

// Código que sempre será executado
System.out.println("Este código sempre será executado, com ou sem

Exemplo:

```
try {

} catch (Exception e) {
        System.out.println("Erro: " + e.getMessage()); } finally {
```

```
exceção.");
}
```

Acesse para conhecer a documentação de Exceções ~~do Java~~

Tratamento de Exceções

Documentação

https://docs.oracle.com/javase/8/docs/api/index.html?java/lang/Exception.html

SENAI

Acesse a documentação oficial de Java ou realize as quatro operações uma inteligência artificial e

Desenvolva uma calcu~~ladora em Java~~ que

Tratamento de Exceções pesquise sobre o conceito e a

Exercícios

aplicação das seguintes exceções: Documentação

Utilize tratamento de exceções para capturar os erros a seguir:

- Divisão por zero;
- Entrada inválida (ex.: usuário digita uma letra em vez de número).

básicas.

Calculadora

- NullPointerException
- ArrayIndexOutOfBoundsException
- ArithmeticException
- FileNotFoundException