Aula anterior

- Tipos primitivos.

- Estrutura de decisão e repetição.

Objetivos

1 - Operadores do java

2 - Entender a sintaxe de declaração de funções em Java.

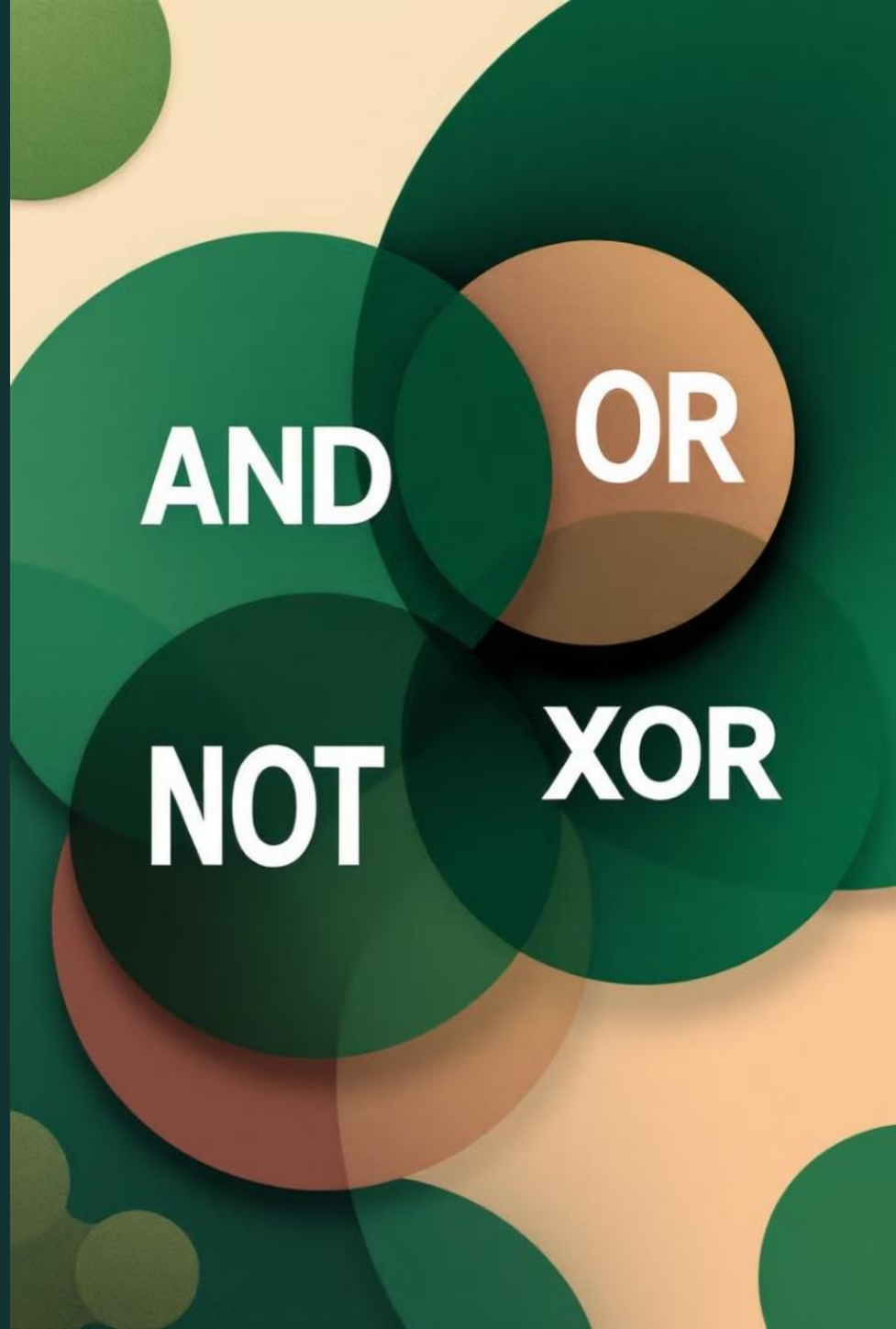
3 - Explorar o retorno de valores de funções.

4 - Compreender como passar parâmetros para funções.

5 - Introduzir operações básicas com objetos String.

Operadores do Java

Operadores Relacionais	Java
Maior	>
Menor	<
Maior igual	>=
Menor igual	<=
Igual	==
Diferente	!=
Não(inversão)	!
E	&&
Ou	



Operadores do java

Operadores Matemáticos	Java
Soma	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da divisão	%
Incremento	++
Decremento	--

Declaração de funções/Passagem de parâmetros

Funções Puras (Pure Functions) Definição:

Funções puras são funções que, dadas as mesmas entradas, sempre produzem as mesmas saídas e não causam efeitos colaterais observáveis. Ou seja, elas não dependem de ou alteram o estado de algo externo à função, como variáveis globais, arquivos, bancos de dados, etc.

Características Principais:

1.Determinismo: A mesma entrada sempre resultará na mesma saída.

2.Imutabilidade: Não alteram variáveis ou estados fora de seu escopo.

3.Sem Efeitos Colaterais: Não causam alterações fora de seu escopo, como modificar um objeto, escrever em um arquivo, ou alterar um banco de dados.

Funções Impuras (Impure Functions) Definição:

Funções impuras são aquelas que podem produzir diferentes resultados com as mesmas entradas ou que causam efeitos colaterais observáveis. Elas podem depender de variáveis ou estados externos e também podem modificar esses estados.

Características Principais:

1.Não-Determinismo: A mesma entrada pode produzir resultados diferentes, dependendo de fatores externos.

2.Dependência de Estado Externo: Podem depender de variáveis globais, bancos de dados, ou outros estados fora de seu escopo.

3.Efeitos Colaterais: Podem causar alterações em variáveis externas, arquivos, ou estados globais.

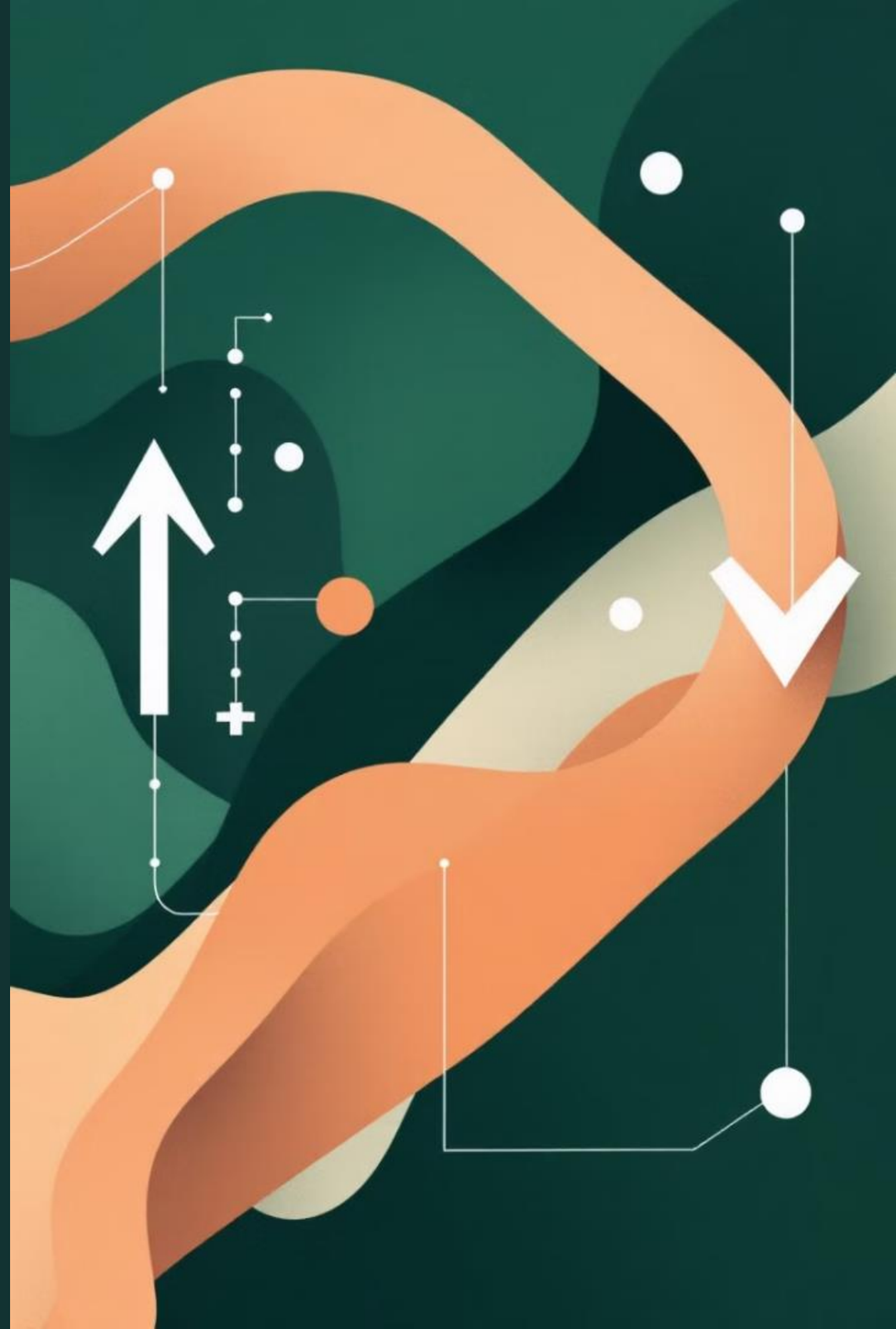
Declaração de funções/Passagem de parâmetros

É composta por tipo de retorno, nome, parâmetros e implementação:

Tipo de Retorno: Indica qual é o tipo de dado que é entregue como resposta pela função. Pode ser qualquer tipo de dado utilizado na declaração de uma variável, salvo o tipo **void**, que é responsável por indicar que a função apenas executa o seu código mas não entrega uma resposta.

Nome: Utilizado para referenciar a função posteriormente.

Parâmetros: Indicam quais informações devem ser entregues à função para que ela execute. **Implementação:** Define a sequência de código que será executada ao se chamar a função. "Executar" uma função significa pedir para que sua implementação seja executada.



Estrutura de funções

```
public tipoRetorno nomeFuncao(tipoParametro parametro) {
    // Corpo da função
    return valorRetorno;
}
```

Exemplos

```java
public class Funcoes {

    // Função pura que soma dois números
    public static int soma(int a, int b) {
        return a + b;
    }

    // Função impura que imprime uma mensagem no console
    public static void imprimirMensagem(String mensagem) {
        System.out.println(mensagem);
    }

    // Função impura que gera um número aleatório
    public static int gerarNumeroAleatorio(int limite) {
        return (int) (Math.random() * limite);
    }

    public static void main(String[] args) {

        // Exemplo de função pura
        int resultadoSoma = soma(2, 3);
        System.out.println("Resultado da soma: " + resultadoSoma);

        // Exemplo de função impura imprimindo uma mensagem
        imprimirMensagem("Olá, mundo!");

        // Exemplo de função impura gerando um número aleatório
        int numeroAleatorio = gerarNumeroAleatorio(10);
        System.out.println("Número aleatório: " + numeroAleatorio);


    }
```

Objetos do Tipo String

Em Java, objetos do tipo String pertencem à classe String, que faz parte do pacote java.lang. Diferente dos tipos primitivos (int, double, char, etc.), a String é um objeto imutável, o que significa que seu valor não pode ser alterado após a criação.

```
package pack;
class ImmutableEx{
     public static void main(String args[]){

       String s="Rahul";
       //concat() method appends the string at the end
       s.concat("Kumar");
            //Will print Rahul becoz Strings are immutable objects
       System.out.println(s);
     }
    }
```

Métodos Úteis da classe String

Concatenação de Strings:

- Usamos o operador + para concatenar duas ou mais strings.
- Exemplo: "João" + " " + "Silva" resulta em "João Silva".

Comprimento da String:

- O método .length() retorna o número de caracteres na string.
- Exemplo: "João Silva".length() retorna 10.

Comparação de Strings:

- O método .equals() compara duas strings para verificar se são exatamente iguais.
- Exemplo: "João".equals("Silva") retorna false.

Métodos Úteis da classe String

Substituição de Caracteres:

- O método .replace(char antigo, char novo) substitui todas as ocorrências de um caractere por outro.
- Exemplo: "João Silva".replace('a', '@') resulta em "João Silv@".

Divisão de String (split):

- O método .split("regex") divide a string em um array com base em um delimitador.
- Exemplo: "João Silva".split(" ") retorna um array ["João", "Silva"].

Conversão para Maiúsculas e Minúsculas:

- .toUpperCase() converte a string para letras maiúsculas.
- .toLowerCase() converte a string para letras minúsculas.
- Exemplo: "João Silva".toUpperCase() retorna "JOÃO SILVA".

Métodos Úteis da classe String

Extração de Substring:

O método .substring(int inicio, int fim) extrai uma parte da string do índice inicio ao índice fim. Exemplo: "João Silva".substring(0, 4) retorna "João".

Verificação de Conteúdo:

- O método .contains(CharSequence s) verifica se a string contém uma sequência de caracteres específica.
- Exemplo: "João Silva".contains("Silva") retorna true.

Remoção de Espaços em Branco:

- O método .trim() remove espaços em branco no início e no final da string.
- Exemplo: " João Silva ".trim() retorna "João Silva".

Referências Bibliográficas

- Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017. 934 p. ISBN: 978-85-430-0479-2.
- PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estrutura de dados com aplicações em Java. 2. ed. São Paulo: Pearson Prentice Hall, 2009. 262 p. ISBN: 978-85-7605-207-4.