==========1-INTRODUÇÃO A LINGUAGEM JAVA (SINTAXE)==========

A - ESTRUTURA FUNDAMENTAL
    //CÓDIGO A0 - Hello World
    //CÓDIGO A1_0 - Tipos primitivos (int, double)
    //CÓDIGO A1_1 - Outros tipos primitivos (...byte, short, long, float)
    //CÓDIGO A1_3 - Constante
    //CÓDIGO A1_4 - Outros tipos primitivos (char)
    //CÓDIGO A1_5 - Outros tipos primitivos (boolean)
    //CÓDIGO A1_6 - String basica
B - ENTRADA E Saída DE DADOS I/O
    //CÓDIGO B1_1 - Entrada e Saída de Dados em Java: Scanner
    //CÓDIGO B1_2- Entrada e Saída de Dados em Java: Métodos (ação) do scanner (objeto)
    //CÓDIGO B1_3 - Aspectos de Entrada e Saídas de Dados (Ex: tipos de dados e cálculo da média)
    //CÓDIGO B1_4 - Aspectos de Entrada e Saídas de Dados (entrada e bases) e Classe Wrapper
    //CÓDIGO B1_5 - Aspectos de Entrada e Saídas de Dados e Classe Wrapper
    //CÓDIGO B1_6 - Aspectos de Entrada e Saídas de Dados e Classe Wrapper (ex: chars)
    //CÓDIGO B1_7 - Aspectos de Entrada e Saídas de Dados e caso String (classe) com seus proprios metodos
C - ESTRUTURA DE DECISÃO (CONDICIONAIS)
    //CÓDIGO C1_1 - IF em Java
    //CÓDIGO C1_2 - IF ELSE em Java
    //CÓDIGO C1_3 - IF ELSE IF em Java
    //CÓDIGO C1_4 - IF dentro de IF em Java
    //CÓDIGO C1_5 - SWITCH CASE DEFAULT em Java (com int)
    //CÓDIGO C1_6 - SWITCH CASE DEFAULT em Java (com char)
D - ESTRUTURA DE REPETIÇÃO
    //CÓDIGO D1_1 - FOR em Java
    //CÓDIGO D1_2 - WHILE em Java
    //CÓDIGO D1_3 - DO WHILE em Java
    //CÓDIGO D1_4 - DO WHILE em Java: (ex. Com o CASE)
E - OPERADORES MATEMÀTICOS E LOGICOS
    //CÓDIGO E1_1 - Operadores aritméticos em Java:
    //CÓDIGO E1_2 - Precedência aritmética em Java:
    //CÓDIGO E1_3 - Operadores Relacionais em Java:
    //CÓDIGO E1_4 - Operadores Lógicos em Java:
    //CÓDIGO E1_5 - Operadores Condicionais Lógicos e Relacionais em Java:
    //CÓDIGO E1_6 - Classe Math (classe utilitaria)
F - ARRAY
    //CÓDIGO F1_1 - Inicializando um array fixo em Java:
    //CÓDIGO F1_2 - Array, propriedade Length e loop para varrer os elementos
    //CÓDIGO F1_3 - Declarando um array para incrementá-lo em Java:
    //CÓDIGO F1_4 - Array multidimensional em Java:
    //CÓDIGO F1_5 - Array multidimensional incremento em Java:
    //CÓDIGO F1_6 - Array multidimensional incremento com busca em Java:
    //CÓDIGO F1_7_A - Classe Array (classe utilitária) - Operações básicas com arrays
    //CÓDIGO F1_7_B - Classe Array (classe utilitária) - Operações avançadas com arrays
G - STRING
    //CÓDIGO G1_1 - String literalmente apresentada

```
//CÓDIGO G1_2 - Manipulacao de String: metodo substring()
//CÓDIGO G1_3 -  Manipulação de String (concatena string): método concat() e
operador +
//CÓDIGO G1_3_A -  Manipulação de String (concatena string): forma de voltar a
variável original
//CÓDIGO G1_3_B -  Manipulação de String (concatena string): alimentar uma
variável vazia
//CÓDIGO G1_4 -  Manipulação de String (concatena com substring): método
concat com substring
//CÓDIGO G1_5 -  Manipulação de String (tamanho string): método .length()
//CÓDIGO G1_6 -  Manipulação de String (comparação tamanho da string): método
compareTo()
//CÓDIGO G1_7 - Comparação de Strings com .equals() e .equalsIgnoreCase()
//CÓDIGO G1_8 - Verificação de substring com contains()
```

==========

—--A - ESTRUTURA FUNDAMENTAL—--

—-----------

//CÓDIGO A0 - Hello World

public: torna o método e a variável acessíveis fora da classe, permitindo que outras classes os chamem diretamente.
static: permite que o método ou variável sejam acessados diretamente pela classe sem precisar criar um objeto, o que é útil aqui, pois tornam o método/variável independentes de qualquer estado de objeto específico.
Void: É um método void, não retorna nenhum valor (sem argumento)
Void apenas executa o que esta dentro

```java
public class A0_HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

—-----------

//CÓDIGO A1_0 - Tipos primitivos (int, double)

```java
public class A1_0Teste {
    public static void main(String[] args) {
        // Declaração de variáveis inteiras e reais
        int numeroInteiro = 10;      // Tipo inteiro (int)
        double numeroReal = 5.75;    // Tipo real (double)

        // Exibindo os valores das variáveis
```

```
        System.out.println("Número inteiro: " + numeroInteiro);
        System.out.println("Número real: " + numeroReal);
    }
}
```

—----------

//CÓDIGO A1_1 - Outros tipos primitivos (...byte, short, long, float)

Tipos Inteiros
byte Quando você precisa de um número pequeno, e deseja economizar espaço (contadores ou informações que não variem, como data)
Tamanho: 8 bits (1 byte)
Intervalo: -128 até 127
short Quando precisa de números um pouco maiores do que byte, mas ainda quer economizar memória (pontuação em jogos)
Tamanho: 16 bits (2 bytes)
Intervalo: -32.768 até 32.767
int Usado quando você precisa de um intervalo de números maiores
Tamanho: 32 bits (4 bytes)
Intervalo: -2^31 até 2^31 - 1 (aproximadamente -2 bilhões até 2 bilhões)
long Quando você precisa trabalhar com números muito grandes, como cálculos financeiros
Tamanho: 64 bits (8 bytes)
Intervalo: -2^63 até 2^63 - 1 (um número extremamente grande)

```
public class A1_1Teste {
    public static void main(String[] args) {
        // Tipos inteiros
        byte numeroByte = 120;          // Tipo byte
        short numeroShort = 30000;      // Tipo short
        int numeroInt = 30000;          // Tipo int
        long numeroLong = 1000000000L;  // Tipo long (necessita do sufixo 'L')

        // Exibindo os valores das variáveis inteiras
        System.out.println("Número byte: " + numeroByte);
        System.out.println("Número short: " + numeroShort);
        System.out.println("Número int: " + numeroInt);
        System.out.println("Número long: " + numeroLong);
    }
}
```

Tipos Reais
float Quando precisa de números decimais, mas não requer muita precisão (ex.: medidas de temperatura, pequenas distâncias, ou cálculos não tão críticos)
Tamanho: 32 bits (4 bytes)
Precisão: Aproximadamente 6 a 7 dígitos decimais
double Quando precisa de maior precisão em cálculos com números decimais, como em cálculos científicos, financeiros detalhados, ou cálculos matemáticos complexos.

Tamanho: 64 bits (8 bytes)
Precisão: Aproximadamente 15 dígitos decimais

```java
public class A1_2Teste {
    public static void main(String[] args) {
        // Tipos reais
        float numeroFloat = 5.75f;      // Tipo float (necessita do sufixo 'f')
        double numeroDouble = 19.99;    // Tipo double

        // Exibindo os valores das variáveis reais
        System.out.println("Número float: " + numeroFloat);
        System.out.println("Número double: " + numeroDouble);
    }
}
```

—----------

//CÓDIGO A1_3 - Constante

static: Quando usamos static, a constante pertence à classe e não a uma instância da classe (não precisa criar um objeto para acessar o valor).
final: Esse é o ponto principal da constante! O final indica que essa variável não pode ser alterada após ser definida.

```java
public class A1_3Teste {
   public static final double PI = 3.1416;

   public static void main(String[] args) {
       float valor = 12.123456789f;

       System.out.printf("Float com 6 casas decimais: %.6f\n", valor);
       System.out.printf("Float em notação científica: %e\n", valor);
       System.out.printf("Valor da constante PI: %.4f\n", PI);
   }
}
```

—----------

//CÓDIGO A1_4 - Outros tipos primitivos (char)

```java
public class A1_4Teste {
    public static void main(String[] args) {
        // Tipo caractere
        char letraA = 'A'; // Tipo char, representa um único caractere

        // Exibindo o valor da variável char
```

```
        System.out.println("Caractere letra A: " + letraA);
    }
}
```

//CÓDIGO A1_5 - Outros tipos primitivos (boolean)

```
public class A1_5Teste {
    public static void main(String[] args) {
        // Tipo booleano, valores true ou false
        boolean ligado = true;   // Tipo boolean, valor true
        boolean desligado = false;  // Tipo boolean, valor false

        // Usando os valores booleanos
        System.out.println("O dispositivo está ligado? " + ligado);
        System.out.println("O dispositivo está desligado? " + desligado);
    }
}
```

—----------

//CÓDIGO A1_6 - String basica

```
public class A1_6Teste {
   public static void main(String[] args) {

       // String (não é um tipo primitivo)
       String texto = "Exemplo de texto";

       // Exibindo os valores
       System.out.println("String: " + texto);
   }
}
```

—----------

—-- B - ENTRADA E Saída DE DADOS I/O —--

—----------

//CÓDIGO B1_1 - Entrada e Saída de Dados em Java: Scanner com nextLine()

Scanner é uma classe
import java.util.Scanner; é a importação de uma classe, a classe Scanner
A classe Scanner faz parte do pacote java.util

linhadigitada é um objeto
Scanner linhadigitada = new Scanner(System.in); é um objeto a partir da classe Scanner
new indica que estamos instanciando (criando) um novo objeto da classe Scanner.
Instância é um objeto específico criado a partir de uma classe.
Linhadigitada é portanto uma instancia da classe B1_1Teste

String entrada = linhadigitada.nextLine();
nextLine(): Esse método (ação) do objeto scanner lê toda a entrada do usuário como texto (String).

```java
import java.util.Scanner;

public class B1_1Teste {
    public static void main(String[] args) {
        Scanner linhadigitada = new Scanner(System.in);

        System.out.print("Digite algo: ");
        String entrada = linhadigitada.nextLine(); // Lê toda a linha de texto

        System.out.println("Você digitou: " + entrada);

        linhadigitada.close();
    }
}
```

—----------

//CÓDIGO B1_2 - Entrada e Saída de Dados em Java: Scanner com next()

String palavra = palavradigitada.nextLine();
next(): Esse método (ação) do objeto scanner lê toda a entrada do usuário como texto (String).

```java
import java.util.Scanner;

public class B1_2Teste {
    public static void main(String[] args) {
```

```
        Scanner palavradigitada = new Scanner(System.in);

        System.out.print("Digite uma palavra: ");
        String entrada = palavradigitada.next(); // Lê apenas uma palavra
        System.out.println("Você digitou: " + entrada);

        palavradigitada.close();
    }
}
```

—----------

//CÓDIGO B1_3 - Entrada e Saída de Dados em Java: Métodos (ação) do scanner (objeto)

O objeto scanner não expõe atributos públicos diretamente manipuláveis pelo programador. Em vez disso, a classe Scanner oferece uma série de métodos para realizar diferentes tipos de leitura de dados.

Dependendo do tipo de dado que você deseja ler do usuário, você pode usar diferentes métodos do Scanner:
nextInt(): Lê um valor do tipo int (número inteiro).
nextDouble(): Lê um valor do tipo double (real com maior precisão).

```
import java.util.Scanner;

public class B1_3Teste {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);

        System.out.print("Digite um número inteiro: ");
        int numeroInt = entrada.nextInt(); // Lê um número inteiro
        System.out.println("Você digitou o inteiro: " + numeroInt);

        System.out.print("Digite um número decimal (double): "); double
numeroDouble = entrada.nextDouble(); // Lê um número double
System.out.println("Você digitou o double: " + numeroDouble);

        entrada.close();
    }
}
```

outros:
nextFloat(): Lê um valor do tipo float (real com precisão menor).
nextBoolean(): Lê um valor booleano (true ou false).
nextLine(): Lê uma linha completa de texto (string).
next(): Lê uma única palavra (até o próximo espaço ou enter).
nextLong(): Lê um valor do tipo long.

—----------

//CÓDIGO B1_4 - Aspectos de Entrada e Saídas de Dados (Ex: tipos de dados e calculo da media)

```
import java.util.Scanner;

public class B1_4Teste {
   public static void main(String[] args) {
       Scanner entrada = new Scanner(System.in);
```

```
        System.out.print("Digite seu nome: ");
        String nome = entrada.nextLine();

        System.out.print("Digite a primeira nota: ");
        double nota1 = entrada.nextDouble();

        System.out.print("Digite a segunda nota: ");
        double nota2 = entrada.nextDouble();

        double media = (nota1 + nota2) / 2;

        System.out.printf("Olá, %s! Sua média é: %.2f\n", nome, media);
        System.out.println("Olá,!" + nome + "Sua média é " + media);

        entrada.close();
    }
}
```

—----------

//CÓDIGO B1_5 - Aspectos de Entrada e Saídas de Dados (entrada e bases) e Classe Wrapper (Embrulhadoras)

Uma classe Wrapper (embrulhada/embrulhadora) é uma classe que ´´embrulha´ um tipo primitivo em um objeto.
Permite que trata tipos primitivos como objetos
A ideia é deixar os tipos primitivos mais flexiveis, expandir eles.

Integer é uma Classe Wrapper (embrulhada) (nao é necessário definir ela), faz parte do Java
toBinaryString() toOctalString () e toHexString () são métodos estáticos da classe Integer
Método estático é um método que pertence à classe e não a uma instância/objeto dessa classe.

```java
import java.util.Scanner;

public class B1_5Teste {
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);

       System.out.print("Digite um número inteiro: ");
       int numero = scanner.nextInt();

       // Exibindo os valores em diferentes bases
       System.out.println("Valor em decimal: " + numero);
       System.out.println("Valor em octal: " + Integer.toOctalString(numero));  //
Converte para octal
       System.out.println("Valor em hexadecimal: " + Integer.toHexString(numero));
// Converte para hexadecimal
       System.out.println("Valor em binário: " + Integer.toBinaryString(numero));
// Converte para binário

       scanner.close();
   }
}
```

—----------

//CÓDIGO B1_5 - Aspectos de Entrada e Saídas de Dados e Classe Wrapper

Classes Wraper para tipos numéricos

Integer: Para o tipo primitivo int. (ex: método toBinaryString()
Float: Para o tipo primitivo float. (ex: método toHexString()
Double: Para o tipo primitivo double. (ex: método toHexString()
Long: Para o tipo primitivo long. (ex: método toBinaryString()
char: Para o tipo primitivo short. (ex: isLetter(), toUpperCase())

```
public class B1_5Teste {
    public static void main(String[] args) {
        // Trabalhando com Integer
        int numero = 10;
        System.out.println("Binário: " + Integer.toBinaryString(numero));  //
Binário
        System.out.println("Hexadecimal: " + Integer.toHexString(numero)); //
Hexadecimal

        // Trabalhando com Float
        float numeroFloat = 3.14f;
        System.out.println("Hexadecimal de float: " +
Float.toHexString(numeroFloat));

        // Trabalhando com Character
        char letra = 'b';
        System.out.println("Maiúsculo: " + Character.toUpperCase(letra));

    }
}
```

—----------

//CÓDIGO B1_6 - Aspectos de Entrada e Saídas de Dados e Classe Wrapper (ex: chars)

```
public class B1_6Teste {
    public static void main(String[] args) {
        // Trabalhando com Character
        char letra1 = 'b';
        System.out.println("Maiúsculo: " + Character.toUpperCase(letra1));
        System.out.println("É digito? " + Character.isDigit(letra1));
        System.out.println("É letra? " + Character.isLetter(letra1));

        char letra2 = 'B';
        System.out.println("Maiúsculo: " + Character.toLowerCase(letra2));

    }
```

```
}
```

—----------

//CÓDIGO B1_7 - Aspectos de Entrada e Saídas de Dados e caso String (classe) com seus proprios metodos

String é uma classe que já oferece esses métodos prontos para uso.
o método toUpperCase() pode ser chamado diretamente na instância (objeto criado) palavra da classe String

```
public class B1_7Teste {
    public static void main(String[] args) {
        // Trabalhando com Character
        String palavra = "teste";
        System.out.println("Maiúsculo: " + palavra.toUpperCase());

    }

}
```

—----------

—--C - ESTRUTURA DE DECISÃO (CONDICIONAIS) —--

—----------

//CÓDIGO C1_1 - IF em Java:

Obs operadores, depois veremos: > < >= <= == !=

```
public class C1_1Teste {
   public static void main(String[] args) {
       int a = 10;

       if (a < 20) {
           // SE a condição for verdadeira
           System.out.println("a eh menor que 20");
       }
   }
}
```

—----------

//CÓDIGO C1_2 - IF ELSE em Java

```java
public class C1_2Teste {
   public static void main(String[] args) {
       int a = 100;

       if (a < 20) {
           // SE a for menor
           System.out.println("a menor que 20");
       } else {
           // SENAO
           System.out.println("a nao eh menor que 20");
       }
   }
}
```

—----------

//CÓDIGO C1_3 - IF ELSE em Java (com operador ternário)

```java
public class C1_3Teste {
    public static void main(String[] args) {
        int x = 10;
        String result = (x > 5) ? "maior que 5" : "menor ou igual a 5";
        System.out.println("O valor de x é " + result);
    }
}
```

—----------

//CÓDIGO C1_4 - IF ELSE IF em Java

```java
public class C1_4Teste {
   public static void main(String[] args) {
       int a = 100;

       if (a == 10) {
           // SE for verdadeiro
           System.out.println("Valor de a é 10");
       } else if (a == 20) {
           // SENAO SE for verdadeiro
           System.out.println("Valor de a é  20");
       } else if (a == 30) {
           // SENAO SE for verdadeiro
           System.out.println("Valor de a é 30");
```

```
        } else {
            // SENAO nenhuma é verdadeira
            System.out.println("Nenhuma condicao eh verdadeira");
        }
    }
}
```

—----------

//CÓDIGO C1_5 - IF dentro de IF em Java

```
public class C1_5Teste {
    public static void main(String[] args) {
        int a = 100;
        int b = 200;

        if (a == 100) {
            // SE sim então
            if (b == 200) {
                // SE sim então
                System.out.println("Valor de a eh 100 e b eh 200");
            }
        }
    }
}
```

—----------

//CÓDIGO C1_6 - SWITCH CASE DEFAULT em Java (com int)

Usa um int como variável de controle (lista).
Lê um número inteiro do input do usuário usando scanner.nextInt().

```
import java.util.Scanner;

public class C1_6Teste {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);

        System.out.println("Menu: Digite o item");
        System.out.println("1- Lista 1");
        System.out.println("2- Lista 2");
        System.out.println("3- Lista 3");

        int lista = entrada.nextInt();
```

```
        switch(lista) {
            case 1:
                System.out.println("Voce escolheu 1");
                break;
            case 2:
                System.out.println("Voce escolheu 2");
                break;
            case 3:
                System.out.println("Voce escolheu 3");
                break;
            default:
                System.out.println("Opcao invalida");
        }

        entrada.close();
    }
}
```

—----------

//CÓDIGO C1_7 - SWITCH CASE DEFAULT em Java (com char)

Usa um char como variável de controle.
Character. É uma classe wrapper (embrulhadora) pra expandir o tipo primitivo char
Character.toUpperCase(opcao) método da classe wrapper garantir que a letra digitada seja convertida para maiúscula (tornar insensivel a maiuscula/minuscula)

Em Java, o Scanner não tem uma função direta para ler um único caractere, e por isso geralmente se usa charAt(0) em conjunto com next() para obter o primeiro caractere de uma entrada.

next() do Scanner foi projetado para trabalhar com String

.charAt(0) pertence a classe String e é usado para acessar um caracter específico (nesse caso é o primeiro caracter digitado)

```
import java.util.Scanner;

public class C1_7Teste {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);

        System.out.println("Menu: Digite a letra");
        System.out.println("A - Opção A");
        System.out.println("B - Opção B");
        System.out.println("C - Opção C");

        System.out.print("Escolha: ");
```

```
        char opcao = entrada.next().charAt(0);

        switch(Character.toUpperCase(opcao)) {
            case 'A':
                System.out.println("Você escolheu a Opção A");
                break;
            case 'B':
                System.out.println("Você escolheu a Opção B");
                break;
            case 'C':
                System.out.println("Você escolheu a Opção C");
                break;
            default:
                System.out.println("Opção inválida");
        }

        entrada.close();
    }
}
```

—----------

//CÓDIGO C1_8 - SWITCH CASE DEFAULT em Java (com string)

```
import java.util.Scanner;

public class C1_8Teste {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);

        System.out.print("Digite o dia da semana: ");
        String dia = entrada.nextLine();  // Lê a string digitada pelo usuário

        // Usando switch com string insensível a maiúsculas/minúsculas
        switch(dia.toLowerCase()) {  // Converte a string para minúsculas
            case "segunda":
                System.out.println("Início da semana");
                break;
            case "sexta":
                System.out.println("Quase fim de semana");
                break;
            default:
                System.out.println("Meio da semana");
        }

        entrada.close();
```

```
    }
}
```

—----------

—--D - ESTRUTURA DE REPETIÇÃO —--

—----------

//CÓDIGO D1_1 - FOR em Java:

A sintaxe das estruturas de repetição (while, do-while, for) é praticamente idêntica em Java e C.

```
public class D1_1Teste {
   public static void main(String[] args) {
       for (int a = 10; a < 20; a++) {
           System.out.println("valor de a = " + a);
       }
   }
}
```

—----------

//CÓDIGO D1_2 - WHILE em Java:

```
public class D1_2Teste {
   public static void main(String[] args) {
       int i = 0;

       while (i < 10) {
           System.out.println("valor " + i);
           i++;
       }

       System.out.println("pronto! fim de loop");
   }
}
```

—----------

//CÓDIGO D1_3 - DO WHILE em Java:

```
public class D1_3Teste {
   public static void main(String[] args) {
       int a = 10;
```

```
        do {
            System.out.println("valor de a: " + a);
            a = a + 1;
        } while (a < 20);
    }
}
```

—----------

//CÓDIGO D1_4 - DO WHILE em Java: (ex. Com o CASE)

```
import java.util.Scanner;

public class D1_4Teste {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int lista;


        do {

            System.out.println("Menu: Digite o item");
            System.out.println("1 - Lista 1");
            System.out.println("2 - Lista 2");
            System.out.println("3 - Lista 3");
            System.out.println("4 - Voltar ao menu");
            System.out.println("0 - Sair");


            lista = entrada.nextInt();


            switch (lista) {
                case 1:
                    System.out.println("Você escolheu 1");
                    break;
                case 2:
                    System.out.println("Você escolheu 2");
                    break;
                case 3:
                    System.out.println("Você escolheu 3");
                    break;
                case 4:
                    // Volta ao menu (não precisa de ação específica, o loop já faz
isso)
                    System.out.println("Voltando ao menu...");
```

```
                    break;
                case 0:
                    // Sair do loop
                    System.out.println("Saindo...");
                    break;
                default:
                    System.out.println("Opção inválida! Tente novamente.");
            }

        } while (lista != 0);  // O loop continua enquanto o usuário não digitar 0

        entrada.close();
    }
}
```

—----------

—--E - OPERADORES MATEMÀTICOS E LOGICOS —--

—----------

//CÓDIGO E1_1 - Operadores aritméticos em Java:

```
public class E1_1Teste {
    public static void main(String[] args) {
        int a = 21;
        int b = 10;
        int c;

        c = a + b; // soma
        System.out.println("Valor de c: " + c);
        c = a - b; // subtração
        System.out.println("Valor de c: " + c);
        c = a * b; // multiplicação
        System.out.println("Valor de c: " + c);
        c = a / b; // divisão
        System.out.println("Valor de c: " + c);
        c = a % b; // resto
        System.out.println("Valor de c: " + c);
    }
 }
```

—----------

//CÓDIGO E1_2 - Precedência aritmetica em Java:

```
public class E1_2Teste {
    public static void main(String[] args) {
        int a, b, c, d;
```

```
        a = (2 * 1 + (2 * 1));
        b = (2 * 1) + (2 * 1);
        c = ((2 * (1 + 2) * 1));
        d = (2 * (1 + (2 * 1)));
       System.out.println("valor de a b c d:\n" + a + "\n" + b + "\n" + c + "\n" +
d);
    }
 }
```

—----------

//CÓDIGO E1_3 - Operadores Relacionais em Java:

```
public class E1_3Teste {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        if (a == b) {
            System.out.println("a = b");
        }
        if (a < b) {
            System.out.println("a menor que b");
        }
        if (a > b) {
            System.out.println("a maior que b");
        }
        if (a <= b) {
            System.out.println("a menor igual que b");
        }
        if (a >= b) {
            System.out.println("a maior igual que b");
        }
        if (a != b) {
            System.out.println("a é diferente de b");
        }
    }
 }
```

—----------

//CÓDIGO E1_4 - Operadores Lógicos em Java:

```
public class E1_4Teste {
    public static void main(String[] args) {
        boolean a = true;
        boolean b = false;

        if (a && b) {
```

```
            System.out.println("Linha 1 VERDADE");
        }
        if (a || b) {
            System.out.println("Linha 2 - VERDADE");
        }
        if (!(a && b)) {
            System.out.println("Linha 3 VERDADE");
        }
    }
}
```

—----------

//CÓDIGO E1_5 - Operadores Condicionais Lógicos e Relacionais em Java:

```
public class E1_5Teste {
    public static void main(String[] args) {
        int a = 11;

        if ((a > 10) && (a < 20)) {
            System.out.println("maior que 10 E menor que 20");
        }
        if ((a < 10) || (a < 20)) {
            System.out.println("menor que 10 OU menor que 20");
        }
        if ((a <= 10) && (a <= 20)) {
            System.out.println("menor igual a 10 E menor igual a 20");
        }
    }
 }
```

—----------

//CÓDIGO E1_6 - Classe Math (classe utilitaria)

O math é uma classe interna do Java que faz parte do pacote java.lang.
Ela fornece constantes matemáticas como Math.PI e Math.E.
Inclui métodos para operações trigonométricas, exponenciais, logarítmicas, arredondamento, e outras funções matemáticas.

Todos os métodos da classe Math são estáticos, então pode chamá-los diretamente usando Math.nomeDoMetodo() sem precisar criar uma instância da classe (sem criar objeto).

```
public class E1_6Teste {
    public static void main(String[] args) {
        double raio = 5.0;
        double base = 2.0;
        double expoente = 3.0;
```

```
        int i = -10;  // Variável inteira para usar com Math.abs()
        double d = 4.5;  // Variável double para usar com Math.pow()

        // Calcula e exibe a área de um círculo
        System.out.println("Área do círculo: " + (2 * Math.PI * raio));

        // Limitando o número de casas decimais
        System.out.printf("Área do círculo com precisão: %.5f\n", 2 * Math.PI *
raio);

        // Alguns métodos da classe Math
        System.out.println("Base elevada ao expoente: " + Math.pow(base,
expoente));
        System.out.println("Seno de 45 graus: " + Math.sin(Math.toRadians(45)));
        System.out.println("Raiz quadrada de 2: " + Math.sqrt(2));
        System.out.println("Logaritmo natural (ln) de 10: " + Math.log(10));
        System.out.println("Logaritmo de base 10 de 100: " + Math.log10(100));
        System.out.println("Arredondamento para cima de 3.1: " + Math.ceil(3.1));
        System.out.println("Arredondamento para baixo de 3.9: " + Math.floor(3.9));
        System.out.println("Valor absoluto de " + i + ": " + Math.abs(i));
        System.out.println("Exponenciação de " + d + " ao quadrado: " + Math.pow(d,
2));

        // Ex: Arredondamento
        double numero = 7.85;
        System.out.println("Arredondamento de 7.85: " + Math.round(numero));

        // Ex: de máximo e mínimo
        System.out.println("Máximo entre 10 e 20: " + Math.max(10, 20));
        System.out.println("Mínimo entre 10 e 20: " + Math.min(10, 20));

        // Ex: de número aleatório
        System.out.println("Número aleatório entre 0.0 e 1.0: " + Math.random());
    }
}
```

—----------

—----------

—--F - ARRAY —--

//CÓDIGO F1_1 - Inicializando um array fixo em Java:
//CÓDIGO F1_2 - Array, propriedade Length e loop para varrer os elementos
//CÓDIGO F1_3 - Declarando um array para incrementá-lo em Java:
//CÓDIGO F1_4 - Array multidimensional em Java:
//CÓDIGO F1_5 - Array multidimensional incremento em Java:
//CÓDIGO F1_6 - Array multidimensional incremento com busca em Java:
//CÓDIGO F1_7 - Classe Array (classe utilitaria)

—----------

Declaração de arrays com colchetes após o tipo int[] em vez de antes do nome da variável.
Tipagem forte: Em Java, os arrays são fortemente tipados.
Índice baseado em zero: Como em C, o primeiro elemento do array tem índice 0.
Verificação de limites: Java faz verificação de limites do array em tempo de execução, lançando uma exceção se você tentar acessar um índice fora dos limites.
—----------

//CÓDIGO F1_1 - Inicializando um array fixo em Java:

```java
public class F1_1Teste {
   public static void main(String[] args) {
       float[] n = {12, 3, 4, -3, 9};
       System.out.printf("1o elemento %.2f\n", n[0]);
       System.out.printf("2o elemento %.2f\n", n[1]);
       System.out.printf("3o elemento %.2f\n", n[2]);
       System.out.printf("4o elemento %.2f\n", n[3]);
       System.out.printf("5o elemento %.2f\n", n[4]);
   }
}
```

—----------

//CÓDIGO F1_2 - Array, propriedade Length e loop para varrer os elementos

```java
public class F1_2Teste {
   public static void main(String[] args) {
       // Criando um array de inteiros
       int[] numeros = {1, 2, 3, 4, 5};

       // Obtendo o tamanho do array usando a propriedade length
       int tamanho = numeros.length;

       // Imprimindo o tamanho do array
       System.out.println("O tamanho do array é: " + tamanho);
```

```
        // Usando o tamanho em um loop para imprimir todos os elementos
        System.out.println("Elementos do array:");
        for (int i = 0; i < numeros.length; i++) {
            System.out.println("Elemento " + i + ": " + numeros[i]);
        }
    }
}
```

—----------

//CÓDIGO F1_3 - Declarando um array para incrementá-lo em Java:

```
import java.util.Scanner;

public class F1_3Teste {
   public static void main(String[] args) {
        int[] n = new int[5];
        Scanner entrada = new Scanner(System.in);

        System.out.println("Entre com 5 numeros:");
        for (int i = 0; i < 5; ++i) {
            System.out.printf("Numero %d: ", i+1);
            n[i] = entrada.nextInt();
        }

        System.out.printf("Primeiro numero: %d\n", n[0]);
        System.out.printf("Ultimo: %d\n", n[4]);

        entrada.close();
    }
}
```

—----------

//CÓDIGO F1_4 - Array multidimensional em Java:

```
public class F1_4Teste {
    public static void main(String[] args) {
        // Declaração e inicialização de um array 2D (matriz 2x2)
        float[][] n = {{12, 3}, {-3, 9}};


        // Imprimindo cada elemento da matriz
        System.out.printf("0x0 elemento %.2f\n", n[0][0]); // Imprime 12.00
        System.out.printf("0x1 elemento %.2f\n", n[0][1]); // Imprime 3.00
        System.out.printf("1x0 elemento %.2f\n", n[1][0]); // Imprime -3.00
        System.out.printf("1x1 elemento %.2f\n", n[1][1]); // Imprime 9.00
    }
 }
```

—----------

//CÓDIGO F1_5 - Array multidimensional incremento em Java:

```java
import java.util.Scanner;

public class F1_5Teste {
   public static void main(String[] args) {
       int[][] matriz = new int[2][2];
       Scanner entrada = new Scanner(System.in);

       System.out.println("\nDigite valor para os elementos da matriz\n");

       for (int i = 0; i < 2; i++) {
           for (int j = 0; j < 2; j++) {
               System.out.printf("Elemento[%d][%d] = ", i, j);
               matriz[i][j] = entrada.nextInt();
           }
       }

       System.out.printf("0x0 elemento %d\n", matriz[0][0]);
       System.out.printf("0x1 elemento %d\n", matriz[0][1]);
       System.out.printf("1x0 elemento %d\n", matriz[1][0]);
       System.out.printf("1x1 elemento %d\n", matriz[1][1]);

       entrada.close();
   }
}
```

—----------

//CÓDIGO F1_6 - Array multidimensional incremento com busca em Java:

```java
import java.util.Scanner;

public class F1_6Teste {
   public static void main(String[] args) {
       int[][] matriz = new int[2][2];
       Scanner scanner = new Scanner(System.in);

       System.out.println("\nDigite valor para os elementos da matriz\n");

       // Incremento
       for (int i = 0; i < 2; i++) {
           for (int j = 0; j < 2; j++) {
               System.out.printf("Elemento[%d][%d] = ", i, j);
```

```
                matriz[i][j] = scanner.nextInt();
            }
        }

        System.out.println("\nValores inseridos:");
        System.out.printf("0x0 elemento %d\n", matriz[0][0]);
        System.out.printf("0x1 elemento %d\n", matriz[0][1]);
        System.out.printf("1x0 elemento %d\n", matriz[1][0]);
        System.out.printf("1x1 elemento %d\n", matriz[1][1]);

        // BUSCA varredura
        System.out.println("\nVarredura da matriz:");
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.printf("Elemento[%d][%d] = %d\n", i, j, matriz[i][j]);
            }
        }

        scanner.close();
    }
}
```

—----------

//CÓDIGO F1_7_A - Classe Array (classe utilitaria) - Operacoes basicas com arrays
Classe Arrays
│
├── Método estático: sort()
├── Método estático: binarySearch()
├── Método estático: copyOf()
├── Método estático: fill()
├── Método estático: toString()
└── ... (outros métodos estáticos)

Não criamos instâncias de Arrays, usamos diretamente os métodos estáticos
Métodos estáticos pertencem à classe, não a instâncias (objetos) da classe.
Não precisam de um objeto da classe para serem chamados.

```java
import java.util.Arrays;

public class F1_7_ATeste {
    public static void main(String[] args) {
        // Criando arrays para demonstração
        int[] numeros = {5, 2, 8, 1, 9, 3, 7, 4, 6};
        int[] numerosCopia1 = new int[5];

        // 1. Ordenação //chama o método para ordenar e depois usa para converter
tudo em string
        System.out.println("1. Ordenação:");
        Arrays.sort(numeros);
        System.out.println("Array ordenado: " + Arrays.toString(numeros));

        // 2. Busca binária (requer array ordenado) //Aponta qual é o indice do
numero 7
        System.out.println("\n2. Busca binária:");
        int indice = Arrays.binarySearch(numeros, 7);
        System.out.println("Índice do número 7: " + indice);

        // 3. Preenchimento de array //completa um array com um determinado valor
        System.out.println("\n3. Preenchimento de array:");
        Arrays.fill(numerosCopia1, 42);
        System.out.println("Array preenchido: " + Arrays.toString(numerosCopia1));

        // 4. Cópia de array //copia de um array declarado em um que absorverá os
elementos.
        System.out.println("\n4. Cópia de array:");
        int[] numerosCopia2 = Arrays.copyOf(numeros, numeros.length);
        System.out.println("Cópia do array: " + Arrays.toString(numerosCopia2));

        //4.1 Copia de array usando um array declarado ja: era int[]
```

```
numerosCopiados = new int[5];
        numerosCopia1 = Arrays.copyOf(numeros, 6);
        System.out.println("Array com 6 elementos copiados: " +
Arrays.toString(numerosCopia1));

        // 5. Comparação de arrays
        System.out.println("\n5. Comparação de arrays:");
        System.out.println("Arrays iguais? " + Arrays.equals(numeros,
numerosCopia2));
        System.out.println("Arrays iguais? " + Arrays.equals(numeros,
numerosCopia1));
    }
}
```

—----------

//CÓDIGO F1_7_B - Classe Array (classe utilitária) - Operações avançadas com arrays

```
import java.util.Arrays;

public class F1_7_BTeste {
    public static void main(String[] args) {
        // Criando arrays para demonstração
        int[] numeros = {5, 2, 8, 1, 9, 3, 7, 4, 6};
        String[] frutas = {"maçã", "banana", "laranja", "uva", "pera"};

        // 6. Criação de array com streams
        System.out.println("6. Criação de array com streams:");
        int[] quadrados = Arrays.stream(numeros).map(n -> n * n).toArray();
        System.out.println("Quadrados dos números: " + Arrays.toString(quadrados));

        // 7. Ordenação parcial
        System.out.println("\n7. Ordenação parcial:");
        Arrays.sort(frutas, 0, 3);
        System.out.println("Frutas parcialmente ordenadas: " +
Arrays.toString(frutas));

        // 8. Preenchimento de parte do array
        System.out.println("\n8. Preenchimento de parte do array:");
        Arrays.fill(numeros, 2, 5, 0);
        System.out.println("Array com parte preenchida: " +
Arrays.toString(numeros));

        // 9. Exibição de array multidimensional
        System.out.println("\n9. Exibição de array multidimensional:");
```

```
        int[][] matriz = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        System.out.println("Matriz: " + Arrays.deepToString(matriz));

        // 10. Verificação de igualdade profunda
        System.out.println("\n10. Verificação de igualdade profunda:");
        int[][] matriz2 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        System.out.println("Matrizes iguais? " + Arrays.deepEquals(matriz,
matriz2));
    }
}
```

—----------

—--G - STRING —--

Java não precisa de uma importação de classe específica para manipulação de strings, pois essas funcionalidades são parte da classe String padrão.

Em Java, strings são objetos imutáveis, então operações como concatenação criam novas strings.

`String nome = "teste";` está criando uma String literal. Em Java, quando você atribui um valor de string diretamente entre aspas duplas, você está criando uma String literal e portanto criando um objeto imutável.

As Strings literais em Java são armazenadas em uma área especial de memória chamada "String pool", o que permite otimizações de memória.

Classe String é uma classe final e imutável:
Final porque ela nao pode ser estendida por outras classes.
Imutavel porque nao pode ser alterado (qualquer alteração, cria novos objetos)

Diferente das Classes Math e Array, que sao classes utilitárias (onde os métodos nao precisam de objetos para serem utilizados)

Possui muitos métodos de manipulação de strings úteis, como `concat()`, `substring()`, `indexOf()`, `replace()`

Alguns métodos estão disponíveis na classe String
Java usa o método `concat()` ou o operador `+` para concatenação de strings.
Para obter uma substring, Java usa o método `substring()`.

O comprimento de uma string em Java é obtido com o método `length()`.

Para comparação de strings, Java usa o método `compareTo()`.

—----------

//CÓDIGO G1_1 - String literalmente apresentada

```java
public class G1_1Teste {
    public static void main(String[] args) {
        String nome = "teste";
        System.out.println("String literal = " + nome);
    }
}
```

—----------

//CÓDIGO G1_2 - Manipulacao de String: metodo substring()

```java
public class G1_2Teste {
    public static void main(String[] args) {
        String str1 = "abcdefghij"; //string literal atribuida a str1
        String str2 = str1.substring(0, 5);//
        System.out.println("str2 = " + str2);
    }
}
```

—----------

//CÓDIGO G1_3 Manipulação de String (concatena string): método concat() e operador +

```java
public class G1_3Teste {
    public static void main(String[] args) {
        String str1 = "abc";
        String str2 = "def";
        String str3 = "ghi";

        // Usando método .concat()
        str1 = str1.concat(str2);
        System.out.println("str1 = " + str1);

        // Usando o operador + para concatenar
        str1 = str1 + str3;
        System.out.println("str1 = " + str1);
    }
}
```

—----------

//CÓDIGO G1_3_A Manipulação de String (concatena string): forma de voltar a variavel original

```java
public class G1_3_ATeste {
    public static void main(String[] args) {
        String originalStr1 = "abc";  // Valor original
```

```
        String str1 = originalStr1;   // Variável para manipulação
        String str2 = "def";

        // Usando método .concat()
        str1 = str1.concat(str2);
        System.out.println("str1 = " + str1);  // Saída: abcdef

        // Retornando ao valor original
        str1 = originalStr1;
        System.out.println("str1 original = " + str1);  // Saída: abc
    }
}
```

—----------

//CÓDIGO G1_3_B Manipulação de String (concatena string): alimentar uma variavel vazia

```
public class G1_3_BTeste {
    public static void main(String[] args) {
        String str1 = "abc";
        String str2 = "def";
        String str3 = "ghi";
        String resultado = "";  // String vazia para acumular concatenações

        // Alimentando a string vazia
        resultado = resultado.concat(str1);  // Adiciona "abc"
        resultado = resultado.concat(str2);  // Adiciona "def"
        resultado = resultado.concat(str3);  // Adiciona "ghi"

        System.out.println("Resultado acumulado = " + resultado);  // Saída:
abcdefghi
        System.out.println("str1 original = " + str1);  // Saída: abc (preservado)
    }
}
```

—----------

//CÓDIGO G1_4 Manipulação de String (concatena com substring): método concat com substring

O método concat() é usado para juntar strings.
substring(0, 2) extrai os primeiros 2 caracteres de str2.

O resultado é atribuído de volta a str1, modificando seu valor original.

```
public class G1_4Teste {
    public static void main(String[] args) {
        String str1 = "abc";
        String str2 = "def";
        str1 = str1.concat(str2.substring(0, 2));
        System.out.println("str1 = " + str1);
    }
}
```

—----------

//CÓDIGO G1_5 Manipulação de String (tamanho string): método .length()

```
public class G1_5Teste {
    public static void main(String[] args) {
        String str = "abcdefghi";
        int tamanho = str.length();
        System.out.println("O tamanho da string " + str + " vale " + tamanho);
    }
}
```

—----------

//CÓDIGO G1_6 Manipulação de String (comparação tamanho da string): metodo compareTo()

```
public class G1_6Teste {
    public static void main(String[] args) {
        String str1 = "ac";
        String str2 = "ae";
        int retorno = str1.compareTo(str2);
        System.out.println("retorno = " + retorno);

        // Explicação dos possíveis valores de retorno:
        // 0: conteúdo das strings são iguais
        // < 0: str1 é lexicograficamente (ordem alfabética) anterior a str2
        // > 0: str1 é lexicograficamente (ordem alfabética) posterior a str2
    }
}
```

—----------

//CÓDIGO G1_7 - Comparação de Strings com .equals() e .equalsIgnoreCase()

```java
public class G1_7Teste {
    public static void main(String[] args) {
        String str1 = "Java";
        String str2 = "java";

        // Verificação de igualdade exata
        if (str1.equals(str2)) {
            System.out.println("As strings são exatamente iguais.");
        } else {
            System.out.println("As strings são diferentes.");
        }

        // Verificação ignorando maiúsculas/minúsculas
        if (str1.equalsIgnoreCase(str2)) {
            System.out.println("As strings são iguais ignorando
maiúsculas/minúsculas.");
        } else {
            System.out.println("As strings são diferentes mesmo ignorando
maiúsculas/minúsculas.");
        }
    }
}
```

—----------

//CÓDIGO G1_8 - Verificação de substring com contains()

```java
public class G1_8Teste {
    public static void main(String[] args) {
        String frase = "Aprender Java aqui";
        String palavra = "Java";

        if (frase.contains(palavra)) {
            System.out.println("A palavra \"" + palavra + "\" está presente na
frase.");
        } else {
            System.out.println("A palavra \"" + palavra + "\" não está presente na
frase.");
        }
    }
}
```

—----------

-string
-list
-collection
-arquivos