ESTRUTURA BÁSICA

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Olá, mundo!");
    }
}
```

public class Main

- **public**: quer dizer que essa "classe" pode ser acessada por qualquer parte do programa. É como dizer "isso está visível para todo mundo".
- **class**: essa palavra-chave diz ao Java que estamos criando uma **classe**, que é como uma "caixa" ou estrutura que guarda o código.
- **Main**: é o **nome da classe**. Pode ser qualquer nome (seguindo algumas regras), mas o nome do arquivo .java precisa ser igual ao nome da classe. Por exemplo: se a classe se chama Main, o arquivo tem que se chamar Main.java.

public static void main(String[] args) {

Essa linha é obrigatória em todo programa Java que queremos executar. É o ponto de entrada do programa, ou seja, o Java começa a executar o código a partir daqui.

- **public**: de novo, quer dizer que esse método pode ser acessado por qualquer parte do programa.
- **static**: significa que esse método pode ser usado sem precisar criar um objeto da classe. (Entendesse melhor quando chegarmos em objetos java POO).
- **void**: significa que esse método não retorna nenhum valor. Ele só executa o que está dentro dele.
- **main**: é o nome do método principal. Esse nome é obrigatório porque o Java procura esse nome para começar a rodar o programa.

System.out.println("Olá, mundo!");

É o comando de saída do Java, assim como o `escreva` Visualg ou o `printf(“”)` do C. Falemos mais sobre ele no próximo capítulo

SAÍDAS EM JAVA

Como já bem sabemos todas linguagens de programação têm um recurso de saída, ou seja uma forma de imprimir uma mensagem na tela. Que guia o usuário sobre o que está acontecer e sobre o que o mesmo deve fazer.

Em Java, a forma mais comum de exibir algo na tela é usando:

```
1. System.out.println();

2. System.out.print();
```

Esses comandos fazem parte de uma biblioteca padrão do Java, que já vem embutida no sistema. Isso significa que não precisamos importar nenhuma biblioteca extra para usá-los. Estão disponíveis por padrão na linguagem, dentro do pacote **java.lang**, que o Java importa automaticamente.

Existe apenas uma pequena **Diferença** entre as duas opções, que está no "ln" adicional na primeira instrução que significa basicamente "*Próxima Linha*" (Não literalmente). Ou seja na primeira opção após mostrar a mensagem o **console Java** passa apara a próxima linha.

- **System**: classe do pacote **java.lang**. Representa o sistema onde o programa está rodando (o computador, basicamente).
- **Out**: é um objeto que representa a saída padrão (normalmente, a tela/console). O tipo dele é PrintStream, que tem métodos para imprimir dados.
- **println() e print()**: Métodos do objeto out, usados para imprimir dados. Aceitam vários tipos de dados: texto, números, variáveis, etc.

Exemplos de uso:

```
System.out.println("Texto");

System.out.println(123);

System.out.println(3.14);

System.out.println(true);
```

Concatenação (+)

```
int idade = 25; (declaração e atribuição de valores à variável)

System.out.println("Idade: " + idade);
```

ENTRADAS EM JAVA

Assim como todas as linguagens de programação têm recursos de saída, elas também possuem formas de entrada de dados, ou seja, maneiras de receber informações digitadas pelo usuário. Essas informações podem ser números, textos, etc., e são fundamentais para que o programa interaja com quem o está usando.

Em Java, a forma mais comum de capturar uma entrada do usuário é usando:

Scanner: Esse recurso pertence a uma biblioteca externa ao núcleo do Java, chamada java.util. Isso significa que precisamos importar essa biblioteca manualmente antes de usá-la.

Para usar o Scanner, é necessário colocar a seguinte linha no início do

```
import java.util.Scanner;
```

Depois de importar a biblioteca, criamos um objeto da classe Scanner, que vai ser responsável por ler as entradas. Isso é feito da seguinte forma:

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

Scanner: é a classe usada para fazer a leitura.

entrada: é o nome do objeto que estamos criando (você pode dar outro nome, mas esse é comum).

new: palavra-chave usada para criar um novo objeto.

System.in: representa a entrada padrão, ou seja, o teclado.

Métodos para ler diferentes tipos de dados

Depois que temos o objeto Scanner, usamos métodos diferentes para ler o tipo de dado correto:

`entrada.nextLine()`; – lê uma linha completa de texto (String).

`entrada.next();` – lê uma palavra (também String, mas para entradas sem espaço).

`entrada.nextInt();` – lê um número inteiro (int).

`entrada.nextDouble();` – lê um número decimal (double).

`entrada.nextBoolean();` – lê um valor verdadeiro/falso (boolean).

```
import java.util.Scanner;

public class ExemploEntrada {

    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 sua idade: ");

        int idade = entrada.nextInt();


        System.out.println("Olá " + nome + ", você tem " + idade + "
anos.");

    }

}
```

Concatenação (+)

Assim como nas saídas, também podemos juntar textos e variáveis para exibir mensagens mais completas:

```
String cidade = "Maputo";

System.out.println("Você está em " + cidade);
```

Observações importantes:

O método nextLine() lê a linha toda, incluindo espaços. Já o next() lê apenas até o primeiro espaço.

Sempre importe java.util.Scanner no início do seu programa para conseguir usar a classe Scanner.

Ao final do uso, pode-se (opcionalmente) fechar o Scanner com entrada.close();, mas isso não é obrigatório em programas simples.