**ISUTC** INSTITUTO **SUPERIOR** DE
**TRANSPORTES** E **COMUNICAÇÕES**

# Programação 1

**sumário:**

- Sintaxe,
- Dados variáveis vs dados constantes
- Tipos de dados
- Expressões
- Operadores
- Entrada e saída de dados

**Erros**

- Exemplo1:

compilação: **javac Exemplo.java**

**'javac' is not recognized as an internal or external command, operable program or batch file.**

- Exemplo2:

execução: java Exemplo

O bytecode para execução não foi encontrado.

```
C:\ISUTC\Semestre II\2022\Programacao 1\Exemplos>java Exemplo
Error: Could not find or load main class Exemplo
Caused by: java.lang.ClassNotFoundException: Exemplo
```

## Erros

- Exemplo3:

erro de sintaxe, faltou acrescentar o “;” na linha 6.

```
2  public class Exemplo{
3      public static void main(String args []){
4          // toda a logica do programa vem aqui
5          int cont = 1;
6          cont++
7          System.out.println(cont);
8      }
9  }
```

```
C:\ISUTC\Semestre II\2022\Programacao 1\Exemplos>javac Exemplo.java
Exemplo.java:6: error: ';' expected
                cont++
                      ^
1 error

C:\ISUTC\Semestre II\2022\Programacao 1\Exemplos>
```

## Erros

- Exemplo4:

erro de semântica, a variável que sofre uma operação não foi inicializada.

```
2  public class Exemplo{
3      public static void main(String args []){
4          // toda a logica do programa vem aqui
5          int cont;
6          cont++;
7          System.out.println(cont);
8      }
9  }
```

```
C:\ISUTC\Semestre II\2022\Programacao 1\Exemplos>javac Exemplo.java
Exemplo.java:6: error: variable cont might not have been initialized
                cont++;
                ^
1 error

C:\ISUTC\Semestre II\2022\Programacao 1\Exemplos>
```

**Sintaxe**

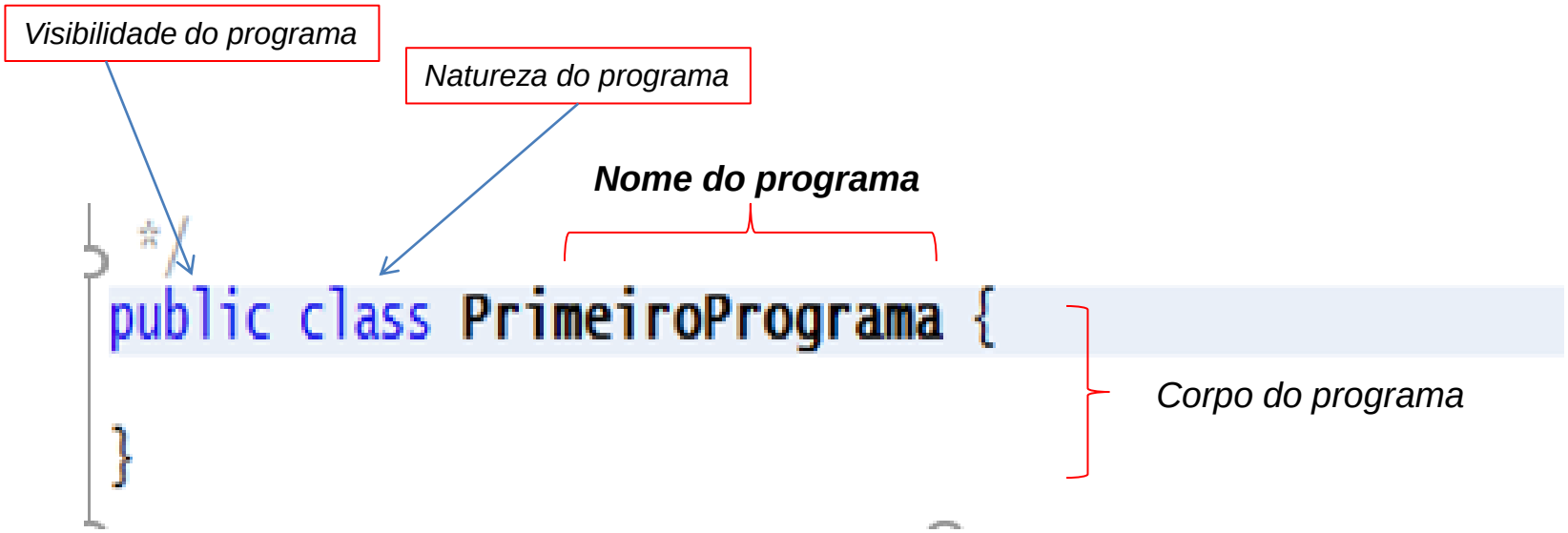
Sintaxe é o conjunto de regras que definem uma linguagem.

- Geralmente a sintaxe de uma linguagem formal é **extremamente rígida** e deve ser seguida **perfeitamente**.
- Como programadores devemos estar atentos a sintaxe do código;

Por exemplo:

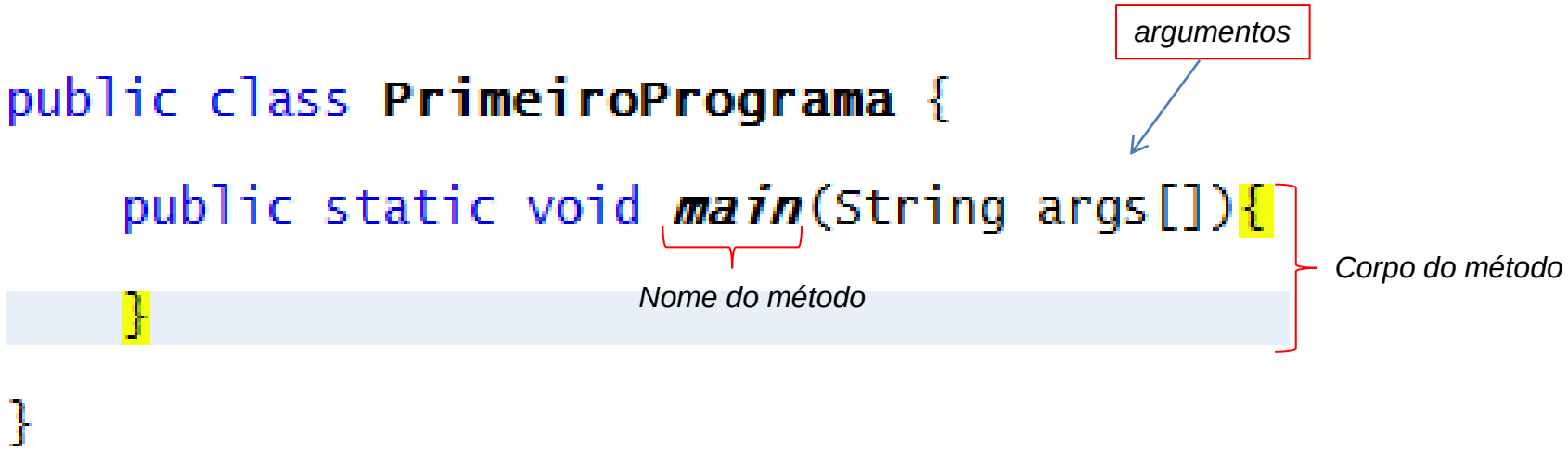
- valorTotal
- valortotal
- valor_total

# Sintaxe


Visibilidade do programa
Natureza do programa
Nome do programa
*/
public class PrimeiroPrograma {
}
Corpo do programa

## Sintaxe

Método **main –** Onde iniciam as aplicações *stand-alone*, o processo de execução e/ou leitura de instruções em java.



**Dados**

É uma porção de memória utilizada para armazenar informações em um programa escrito em qualquer linguagem de programação.

**[tipo] [nome] [;]**

- **Tipo**: determina o espaço de memoria que deve ser reservado para armazenar o valor correspondente e ainda a forma de representação,
- **Nome:** identifica a variável permitindo a sua manipulação por parte do programador, sem que necessite saber a sua localização na memoria.

**Dados variável**

- O seu valor pode mudar ao longo da execução do programa;
- A cada momento, apenas armazena um único valor;
- A atribuição de um valor destrói o valor anterior;

NB: Para usar a variável é necessário declará-la previamente

Estrutura de declaração: **[tipo] [nome] [;]**

Exemplo:

```
float raio;
int miniTeste;
```

## Dados constantes

- Armazena valores que se mantém inalterados ao longo da execução do programa;
- É útil quando temos valores que se vão manter inalterados durante a execução do programa;
- Tal como as variáveis, as constantes devem ser declaradas e *imediatamente inicializadas.*

Estrutura de declaração: **final [tipo] [nome] = [valor] [;]**
A palavra **final** previne com que o valor não seja modificado.

**Exemplo:** final double PI = 3.14;

## Tipos de Dados: inteiros

Existem 4 tipos de dados que podem ser utilizados para armazenar números inteiros. Cada uma delas difere no tamanho da variável [capacidade máxima de armazenamento de dados].

Portanto a escolha do tipo de dados dependera do dado/valor a ser armazenado.

| DATA TYPE | DESCRIPTION |
|---|---|
| `byte` | Variables of this type can have values from −128 to +127 and occupy 1 byte (8 bits) in memory |
| `short` | Variables of this type can have values from −32768 to 32767 and occupy 2 bytes (16 bits) in memory |
| `int` | Variables of this type can have values from −2147483648 to 2147483647 and occupy 4 bytes (32 bits) in memory |
| `long` | Variables of this type can have values from −9223372036854775808 to 9223372036854775807 and occupy 8 bytes (64 bits) in memory |

## Tipos de Dados

```
byte temperatura;
temperatura = 34;

short kilometros;
kilometros = 32000;

int temperaturaSolar;
temperaturaSolar = 15600000;

long anosLuz;
anosLuz = 9460700000000000;
```

## Tipos de Dados: reais

É uma estrutura que permite armazenar valores decimais e de grande dimensão.

A sua *decimal part* [componente decimal] é considerada flutuante ex.0.005 pode ser representar em 5x10E-3

Existem dois [2] tipos:

| DATA TYPE | DESCRIPTION |
|---|---|
| `float` | Variables of this type can have values from −3.4E38 ($-3.4 \times 10^{38}$) to +3.4E38 ($+3.4 \times 10^{38}$) and occupy 4 bytes in memory. Values are represented with approximately 7 decimal digits accuracy. The smallest non-zero `float` value that you can have is approximately $1.4 \times 10^{-45}$. |
| `double` | Variables of this type can have values from −1.7E308 ($-1.7 \times 10^{308}$) to +1.7E308 ($+1.7 \times 10^{308}$) and occupy 8 bytes in memory. Values are represented with approximately 17 decimal digits accuracy. The smallest non-zero `double` value that you can have is approximately $4.9 \times 10^{-324}$. |

## Tipos de Dados: reais

```
float pi;
pi = 3.141592653f;

double divisao;
divisao = 0.33333333333333333333333334;
```

## Tipos de Dados: caracteres

É uma estrutura que permite possuir um carácter entre plicas ‘ ’.

char caracter;

caracter = ‘A’;

| tipo | Memória ocupada | valores |
|---|---|---|
| char | 16bits | Qualquer carácter pertencente ao UNICODE. |

## Tipos de Dados: valores lógicos

É uma estrutura que possui dois tipos de valores: verdadeiro ou falso.

boolean pergunta;

pergunta = true;

| tipo | Memória ocupada | valores |
|---|---|---|
| boolean | 1bit | true ou false |

**ATENÇÃO**

- Antes de utilizar uma variável precisa-se primeiro de identificar o nome desta;
- Java é uma linguagem *case sensitive* Turma e turma, são nomes de variáveis diferentes;
- Não se deve incluir espaços em branco para nomes compostos ex. total alunos *[errado]*, mais sim total_alunos| totalAlunos *[correcto]*;
- Não deve-se incluir números no inicio da variáveis ex. 5Elements *[errado]* mas sim ao longo do seu texto/nome ex. fiveElements | five_elements*[correcto];*
- A declaração de variável apenas reserva um espaço na memória, enquanto não for inicializada, a variável tem um valor indefinido;
- Usar uma variável não inicializada resulta em um erro detectado pelo compilador;

**Expressões**

No quotidiano, muitas vezes usamos expressões sem perceber que as mesmas representam expressões algébricas ou numéricas.

- **Expressões aritméticas:** são expressões matemáticas que envolvem operações com números.

Por exemplo:

a=7+5+4; b=5+20-87; c=(6+8)-10; d=(5×4)+15

- **Expressões algébricas**: são expressões matemáticas que apresentam letras e podem conter números. São também denominadas expressões literais. Por exemplo:

A=2a+7b; B=(3c+4)-5; C=23c+4

**Operadores**

Operadores são símbolos que representam **atribuições, cálculos e ordem dos dados**.

As *operações* seguem uma ordem de prioridade, ou seja, alguns cálculos(ou outros) são processados antes de outros.

Estão divididos em *3 tipos* em relação à quantidade de operandos no qual operam:

- Unário: **a - -**
- Binário: **c = a * b**
- Ternário: **c > 0 ? a : b**

Ternário e um operador semelhante a if..else veja a seguir com mais explicação sobre o operador ternário

# Operador ternário

## if..else

```
if (a > 0) {
    b = 1;
}else{
    b = 2;
}
```

## ternário será

```
b = (a > 0) ? 1 : 2;
```

## Operadores

| Operador | Operação | Prioridade |
|---|---|---|
| + | Soma | 4 |
| - | Subtração | 4 |
| / | Divisão | 3 |
| * | Multiplicação | 3 |
| $\sqrt{x}$ | Raiz quadrada | 2 |
| $e^x$ | Exponenciação | 2 |
| ( ) | parênteses | 1 |

A divisão retorna um inteiro se os argumentos forem inteiros, mas se o numerador for de outro tipo retorna um ponto flutuante

15 / 2 = 7

15.0 / 2 = 7.5

# Operadores relacionais

| Operador | Operação |
|---|---|
| > | Maior que |
| < | Menor que |
| >= | Maior ou igual que |
| <= | Menor ou igual que |
| == | Igual a |
| != | Diferente de |

| Operador | Operação |
|---|---|
| && | Conjunção |
| \|\| | Disjunção |
| ! | Negação |

| Operador | Operação |
|---|---|
| var = var + 1;<br>var++; | Soma 1 ao operando |
| var = var - 1;<br>var--; | Decrementa 1 ao operando |
| ++var | Soma 1 ao operando |
| --var | Decrementa 1 ao operando |
| var = var + valor<br>var += valor | Soma o valor ao operando |
| var = var - valor<br>var -= valor | Decrementa o valor ao operando |
| var = var * valor<br>var *= valor | Multiplica o valor ao operando |
| var = var / valor<br>var /= valor | Divide o valor ao operando |

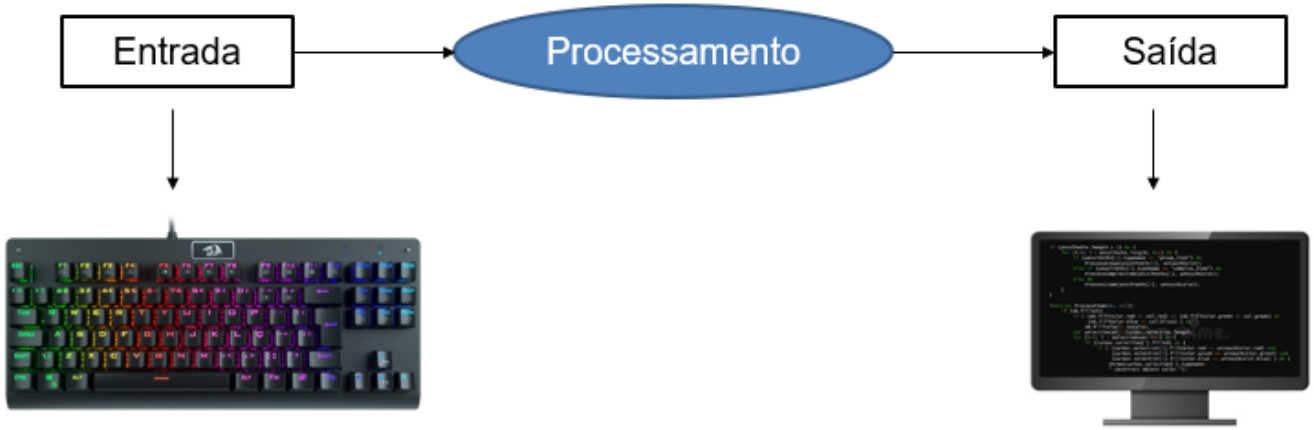
```
int a = 7;
int b = 7;
int x = 2 * ++a;   //primeiro realiza o incremento e depois multiplica
int y = 2 * b++; //primeiro multiplica e depois incrementa
```

x = 16
y = 14
a = 8
b = 8

## Entrada e saída de dados

Ciclo de um programa:

- Entrada
- Processamento
- Saída



## Entrada de dados

É a recepção de dados de forma bruta através de um dispositivo de entrada, como por exemplo o teclado. A linguagem Java, através do pacote **java.util** disponibilizou a **classe Scanner**, que implementa operações de entrada de dados pelo teclado.

Métodos da classe Scanner: para as utilizar deve importar a classe com o comando **import java.util.Scanner;**

## Métodos da classe Scanner:

| Tipo | Função | Designação |
|---|---|---|
| Byte | nextByte() | Retorna um byte |
| Short | nextShort() | Retorna um short |
| Int | nextInt() | Retorna um inteiro |
| Long | nextLong() | Retorna um long |
| Float | nextFloat() | Retorna um float |
| Double | nextDouble() | Retorna um double |
| Char | next().chatAt(0) | Retorna um caracter |
| String | next() | Retorna uma String simples sem espaço em branco |
| String | nextLine() | Retorna uma String que possui espaços em branco |

# Métodos da classe Scanner:

```
1  import java.util.Scanner;
2
3  public class Exemplo{
4      public static void main(String args []){
5          // toda a logica do programa vem aqui
6          Scanner ler = new Scanner(System.in);
7
8          float val1 = ler.nextFloat();
9          int val2 = ler.nextInt();
10         byte val3 = ler.nextByte();
11         long val4 = ler.nextLong();
12         boolean val5 = ler.nextBoolean();
13         double val6 = ler.nextDouble();
14         char val7 = ler.next().charAt(0);
15         String nomeCompleto = ler.nextLine();
16         String nome = ler.next();
17     }
18 }
19
```

## Saída de dados

O objecto **System.out** é a saída padrão, dentro desse objecto temos funções que geram saídas de Strings.

| função | Designação |
|---|---|
| System.out.print(“msg”) | Mantém o cursor na mesma linha. Geralmente são utilizadas sequências de escape para pular uma linha. |
| System.out.println(“msg”) | Insere uma nova linha, deixando o marcador posicionado na linha abaixo. |
| System.out.printf(“msg”) | Especifica o formato da entrada do tipo de valor, que deve ser o mesmo tipo de dados apontado na instrução. |

# Saída de dado

```
import java.util.Scanner;

public class Exemplo{
    public static void main(String args []){
        // toda a logica do programa vem aqui
        Scanner ler = new Scanner(System.in);
        int val1 = ler.nextInt();
        int val2 = ler.nextInt();

        System.out.print("Val1 = "+val1);
        System.out.println("Val2 = "+val2);
        System.out.printf("Soma das variáveis num1 e num 2 = %d",(val1 + val2));
    }
}
```

**GARANTE O TEU FUTURO**

**COM UMA FORMAÇÃO SÓLIDA**

ISUTC

Prolong. da Av. Kim Il Sung (IFT/TDM) Edifício D1
Maputo, Moçambique

**www.facebook.com/isutc**

**www.transcom.co.mz/isutc**