

Tipos de dados Primitivos

Tipo	Tamanho	Intervalo/Valores	Exemplo de utilização
byte	1 byte	-128 até 127	byte meuByte = 127;
short	2 bytes	-32.768 até 32.767	short meuShort = 4;
int	4 bytes	-2.147.483.648 até 2.147.483.647	int meuInt = 999999999;
long	8 bytes	-9.223.372.036.854.775.808 até 9.223.372.036.854.775.807	long meuLong = 999999 999 9999;
float	4 bytes	Armazena números fracionários com 6 ou 7 dígitos de precisão. 3,4e ^{-0.38} até 3,4e ^{+0.38}	float meuFloat = 3.1415 926 535 897 93238f;
double	8 bytes	Armazena números fracionários com 15 dígitos de precisão. 1,7e ⁻³⁰⁸ até 1,7e ⁺³⁰⁸	double meuDouble = 3.1415 926 535 897 932 384 6264 33;
char	2 bytes	Conjunto de caracteres Unicode	char meuCar actere = '#';
boolean	1 bit	true ou false	boolean meuBoo leano = true;

Tipos de dados Não-Primitivos

Tipo	Uso	Exemplo
String	Armazena textos.	String texto = "Um exemplo de texto." ;
Array	Armazena um conjunto de elementos do mesmo tipo . O tamanho de um <i>array</i> é fixo e especificado na sua criação. Cada elemento no <i>array</i> pode ser acessado por um índice numérico que indica a sua posição. A primeira posição de um <i>array</i> é sempre 0.	int[] numeros = {10, 20, 30, 40}; //numeros[X] acessa o valor na posição X System.out.println(numeros[0]); // imprime o valor 10 num = numeros[2]; //copia o valor 30 para a variável num numeros[3] = 50; //altera a quarta posição de 40 para 50
Class	Permite modelar e representar um conceito lógico de forma estruturada e reutilizável, com seus atributos e funcionalidades. Os atributos representam as características da entidade modelada e os métodos definem o comportamento ou ações podem ser realizadas por essa entidade.	class MinhaClasse{ //Atributos .. //Métodos .. }

Tipos de dados Não-Primitivos (cont)

Interface

Declara comportamentos que as classes devem implementar.

```
interface MinhaInterface{
    public void metodo1();
    public void metodo2();
}
```

Enum

Representa um conjunto fixo de constantes.

```
public enum MeuEnum {
    VALOR1,
    VALOR2,
    VALOR3
}
```

Quebra de página

Comandos de Seleção

Comandos ou estruturas de seleção ou condicionais são estruturas de controle que possibilitam ao programador tomar decisões com base em condições específicas. Basicamente, este tipo de comando permite definir se um ou mais conjuntos de instruções serão executados ou não.

if

O comando **if** (**se** em português) é a estrutura básica de seleção existente em qualquer linguagem de programação. Ele avalia uma condição e executa um bloco de código se essa condição for verdadeira.

Sintaxe:

```
if (condicao) {
    //exec utado se a condição for verdadeira
}
```

Exemplo:

```
if (idade >= 18) {
    System.out.println ("É maior de idade.");
}
```

if / else

switch / case (cont)

O comando **if** em conjunto com o **else** (**se / senão** em português) permite avaliar uma condição e executar um bloco de código se essa condição for **verdadeira** ou outro bloco de código se a mesma for **falsa**.

Sintaxe:

```
if (condicao) {
    // executado se a condição for verdadeira
}
else {
    // executado se a condição for falsa
}
```

Exemplo 1:

```
if (idade < 18) {
    System.out.println ("É menor de idade.");
}
else {
    System.out.println ("É maior de idade." );
}
```

O comando **else** é opcional. Podem ser concatenados vários **if / else** em conjunto, mas somente o **if** pode conter expressões e todo **else** deve ter um **if** correspondente.

Exemplo 2:

```
if (condicao1) {
    // executado se a condicao1 for verdadeira
}
else if (condicao2) {
    // executado se a condicao1 for falsa
    // e a condicao2 for verdadeira
}
else {
    // executado se nenhuma das condições
    // anteriores for verdadeira
}
```

switch / case

Permite avaliar uma variável e escolher diferentes caminhos de execução com base nos possíveis valores dessa variável. A variável avaliada pode ser do tipo **boolean**, **byte**, **short**, **int**, **enum**, **char** ou **String**. Os outros tipos de dados não são suportados. A opção **default** permite executar um bloco de código caso nenhum dos valores declarados anteriormente corresponda ao conteúdo atual da variável, mas sua definição é opcional.

No **switch/case** não existem blocos definidos por chaves (**{ }**). Para definir o ponto de parada das instruções de um caso, é utilizada a palavra-chave **break**. É possível executar o mesmo bloco de código para diferentes casos, se eles estiverem em sequência, como no **Exemplo 2**.

Sintaxe:

```
switch (variavel) {
    case valor1:
        // executado se a expressao for igual a valor1
        break;
    case valor2:
        // executado se a expressao for igual a valor2
        break;
    // outros cases e blocos de código
    default:
        // opcional
        // executado se nenhum case anterior for válido
}
```

Exemplo 1 (inteiro):

```
int opcao = 2;
switch (opcao) {
    case 1:
        System.out.println ("Opção 1 selecionada");
        break;
    case 2:
        System.out.println ("Opção 2 selecionada");
        break;
    default:
        System.out.println ("Opção inválida.");
}
```

Exemplo 2 (String):

```
String diaSemana = " quarta ";
switch (diaSemana) {
    case " segunda ":
    case " terça ":
    case " quarta ":
    case " quinta ":
    case " sexta ":
        System.out.println ("Dia útil selecionado");
        break;
    case " sábado ":
    case " domingo ":
        System.out.println ("Fim de semana selecionado.");
        break;
    default:
        System.out.println ("Dia inválido.");
}
```

Comandos de Repetição

Os comandos de repetição, estruturas de repetição, laços de repetição ou *loops* são utilizados na programação para executar um bloco de código várias vezes de forma automatizada, com base em uma condição de controle.

while

O **while** (**enquanto** em português) é utilizado quando não sabemos previamente quantas vezes a repetição deve ocorrer, mas temos uma condição de parada.

Sintaxe:

```
while (condicao) {  
    /* executado enquanto a condição  
    for verdadeira */  
}
```

Exemplo 1 (expressão):

```
int i = 1;  
while (i <= 5) {  
    System.out.println(i);  
    i++;  
}
```

Exemplo 2 (boolean):

```
boolean ligado = true;  
while (ligado) {  
    System.out.println("Está ligado");  
    System.out.println("De sligan do...");  
    ligado = false;  
}
```



By **fmarson**
cheatography.com/fmarson/

Not published yet.
Last updated 7th February, 2024.
Page 3 of 6.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>

do/while

O **do/while** (**faça enquanto** em português) é semelhante ao **while**, mas é utilizado quando queremos executar o código **pelo menos uma vez**.

Sintaxe:

```
do{
    /*exec utado pelo menos uma vez e
    enquanto a condição for verdadeira */
} while (condi cao);
```

Exemplo 1:

```
// imprime na tela os números de 1 até 5
int i = 1;
do{
    System.out.println(i);
    i++;
}while (i <= 5);
```

Exemplo 2:

```
// imprime apenas o número 10 na tela
int i = 10;
do{
    System.out.println(i);
    i++;
}while (i <= 5);
```

for

Utilizado quando se conhece antecipadamente quantas vezes o bloco de código deve ser repetido. A sua estrutura contém de forma agrupada uma inicialização, uma condição e uma expressão de iteração.

Sintaxe:

```
for (inici ali zacao; condicao; iteracao) {  
    // bloco de código a ser repetido  
}
```

Exemplo 1:

```
// imprime na tela os números de 1 até 5  
for (int i = 1; i <= 5; i++) {  
    System.out.println(i);  
}
```

Exemplo 2:

```
// imprime na tela os números de 10 até 1  
for (int i = 10; i >= 1; i--) {  
    System.out.println(i);  
}
```

Exemplo 3:

```
/* imprime na tela o conteúdo de um array  
de float com 3 posições */  
float[] array = {1.0f, 3.2f, 9.55f};  
for (int p = 0; p < 3; p++) {  
    System.out.println(array[p]);  
}
```

for each

Este tipo de **for** é utilizado quando se deseja percorrer uma coleção de dados.

Formato:

```
for (tipo elemento : colecao) {  
    /* código a ser repetido  
    para cada elemento */  
}
```

Exemplo (array):

```
int[] numeros = {1, 4, 7, 12};  
  
//imprime os números do array na tela  
for (int numero : numeros) {  
    System.out.println(numero);  
}
```

