GUIA DE REFERÊNCIA · LINGUAGEM DE PROGRAMAÇÃO I

Formatação de Saída com printf()

Como controlar a exibição de textos, números e datas no console em Java

Java

1. O que é o printf()

O comando System.out.printf() serve para imprimir textos formatados no console, permitindo controlar como números, textos e datas aparecem na tela. Ele funciona de maneira parecida com o printf da linguagem C.

A ideia é montar uma string de formatação com "marcadores" especiais chamados especificadores de formato. Cada marcador começa com o símbolo % seguido de uma letra que define o tipo de dado que será exibido no lugar dele.

> 💡 **Por que usar**
>
> O printf() evita aquelas concatenações gigantes de texto com o símbolo +, como em "Nome: " + nome + " | Idade: " + idade.

Sintaxe básica:

```
System.out.printf("Texto com %especificador", variavel);
```

2. Principais especificadores de formato

Os marcadores começam com % e definem o tipo de dado que será exibido:

Especificador	Usado para
%s	Texto (String)
%d	Números inteiros (int, long, byte, etc.)
%f	Números com ponto flutuante (double, float)
%b	Valores booleanos (true / false)
%c	Um único caractere (char)
%n	Pular uma linha, adaptada ao sistema operacional (melhor que \n)

Exemplo — misturando texto, strings e inteiros:

```
String nome = "Ana";
int idade = 25;

System.out.printf("Olá, meu nome é %s e eu tenho %d anos.%n", nome, idade);
// Saída: Olá, meu nome é Ana e eu tenho 25 anos.
```

3. Formatando casas decimais (%.Xf)

Por padrão, o %f exibe muitas casas decimais. Você pode limitar isso usando %.Xf, onde X é o número de casas desejado:

```
double preco = 1543.2876;

System.out.printf("Valor padrão: %f%n", preco);    // Saída: 1543.287600
System.out.printf("Valor limitado: %.2f%n", preco); // Saída: 1543,29 (arredondado)
```

4. Preenchendo com zeros à esquerda

Se você precisa que um número tenha sempre a mesma quantidade de dígitos (como códigos de barras, IDs ou horas), adicione um 0 antes da largura desejada:

```
int id1 = 5;
int id2 = 142;

// %04d significa: mostre um inteiro com no mínimo 4 dígitos, preenchendo com zero
System.out.printf("Código do produto 1: ID_%04d%n", id1);
System.out.printf("Código do produto 2: ID_%04d%n", id2);

/* Saída:
Código do produto 1: ID_0005
Código do produto 2: ID_0142
*/
```

5. Moeda e números grandes

Para exibir valores monetários ou números com separador de milhar (como 1.000,00), adicione uma vírgula logo após o %:

```
double salario = 5250.75;
int populacao = 1500300;

System.out.printf("Salário: R$ %,.2f%n", salario);
System.out.printf("População da cidade: %,d habitantes.%n", populacao);

/* Saída (em sistemas configurados em português):
Salário: R$ 5.250,75
População da cidade: 1.500.300 habitantes.
*/
```

> ⚠️ **Atenção**
> A formatação de milhar e de moeda depende da configuração regional (locale) do sistema onde o programa roda. Teste sempre no ambiente de destino.

6. Alinhamento e espaçamento — criando tabelas no console

Você pode definir uma largura mínima para o campo usando um número após o %. Números negativos alinham o texto à esquerda; números positivos alinham à direita.

```
String item1 = "Hambúrguer";
String item2 = "Batata Frita";
String item3 = "Refrigerante";

System.out.printf("====================================%n");
System.out.printf("%-20s | %10s%n", "ITEM", "PREÇO");
System.out.printf("====================================%n");
System.out.printf("%-20s | R$%9.2f%n", item1, 28.50);
System.out.printf("%-20s | R$%9.2f%n", item2, 12.00);
System.out.printf("%-20s | R$%9.2f%n", item3, 6.50);
System.out.printf("====================================%n");

/* Saída:
====================================
ITEM                 |      PREÇO
====================================
Hambúrguer           | R$    28,50
Batata Frita         | R$    12,00
Refrigerante         | R$     6,50
====================================
*/
```

7. Formatando datas e horas (sufixo t)

O printf() também extrai partes de objetos de data (como LocalDateTime) usando a letra t (ou T para maiúsculas):

```
import java.time.LocalDateTime;

LocalDateTime agora = LocalDateTime.now();

// %1$t indica que todas as formatações usam o primeiro argumento (agora)
// d = dia, m = mês, Y = ano com 4 dígitos, H = hora, M = minuto
System.out.printf("Data: %1$td/%1$tm/%1$tY - Horário: %1$tH:%1$tM%n", agora);

// Saída: Data: 15/09/2026 - Horário: 09:24
```

8. Dica bônus: String.format()

Se você não quiser imprimir o texto direto na tela, mas sim guardá-lo formatado dentro de uma variável, use exatamente a mesma lógica com String.format():

```
String mensagem = String.format("O valor final é R$ %.2f", 99.90);
// Agora a variável 'mensagem' guarda o texto "O valor final é R$ 99,90"
```

9. Resumo: print() vs println() vs printf()

Método	O que ele faz	Pula linha automaticamente?
System.out.print()	Imprime o texto exatamente como enviado.	Não
System.out.println()	Imprime o texto e pula para a próxima linha.	Sim
System.out.printf()	Imprime o texto formatado usando especificadores (%s, %d, %f...).	Não (use %n)