

Classes JAVA

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)

- **Package**
 - java.util
- **Interesse**
 - Leitura de Dados
 - Fontes
 - Teclado
 - Ficheiro de Texto

- **Realizada**

- Por objeto da classe Scanner // objeto = **instância**

- **Código Necessário**

- 1º Declaração de objeto da classe Scanner para ler do teclado
- 2º Leitura através de métodos de instância aplicados ao objeto declarado

- **Exemplos**

```
import java.util.Scanner;


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

        // Declaração de objeto (ou instância) para ler do teclado
        Scanner ler = new Scanner(System.in);

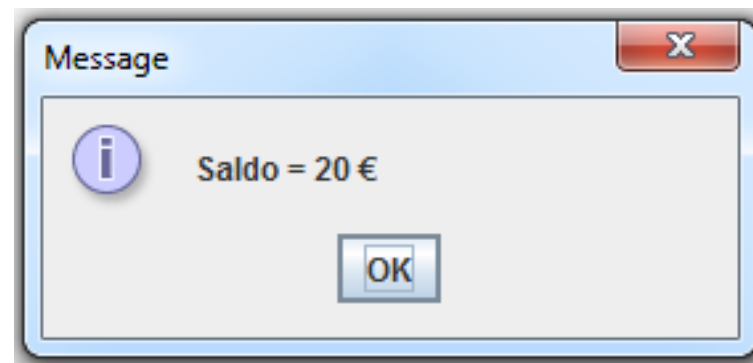
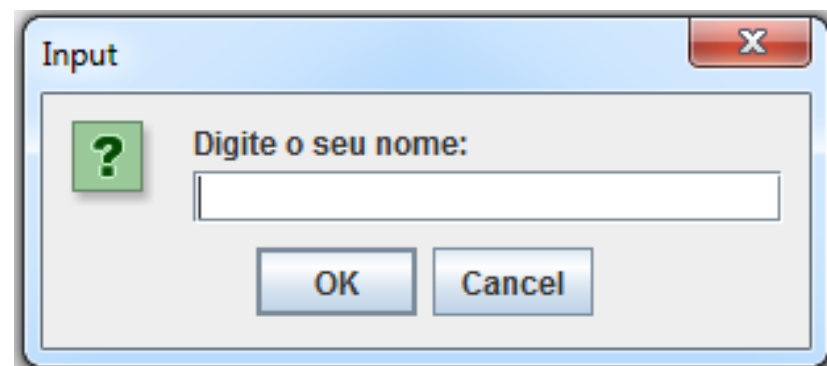
        // Leitura
        String nome = ler.next();           // leitura de nome simples
        String nome = ler.nextLine();       // leitura de nome composto
        int i = ler.nextInt();
        long l = ler.nextLong();
        double d = ler.nextDouble();
        float f = ler.nextFloat();
    }
}
```

■ Pormenores

- `Scanner ler = new Scanner(System.in) ;`
 - Declaração do objeto (ou instância) **ler** da classe `Scanner` para ler do teclado
 - `new` **operador** para criar uma instância
 - `Scanner()` **construtor** de instâncias da classe `Scanner`
 - `System.in` representa **teclado da consola**
- **Métodos de instância** da classe `Scanner` que podem ser **aplicados ao objeto** `ler`
 - `next()` Lê próxima string simples do teclado (i.e., cadeia de caracteres terminada pelo carácter espaço ou newline ('\n'))
 - `nextLine()` Lê próxima linha do teclado (i.e., cadeia de caracteres terminada em \n)
 - `nextInt()` Lê próximo int do teclado
 - `nextLong()` Lê próximo long do teclado
 - `nextFloat()` Lê próximo float do teclado
 - `nextDouble()` Lê próximo double do teclado

- 
- [Scanner](#)
 - [JOptionPane](#)
 - [String](#)
 - [Character](#)
 - [Formatter](#)
 - [Calendar](#)
 - [Math](#)
 - [Integer](#)
 - [Long](#)
 - [Float](#)
 - [System](#)

- **Package**
 - javax.swing
- **Interesse**
 - **Caixas de Diálogo da Interface Gráfica**
 - Tipos
 - **Entrada**
 - Para Leitura de Dados do Teclado
 - **Saída**
 - Para Escrita de Resultados



- Método de Classe `showMessageDialog`
 - Sintaxe
 - `void showMessageDialog(null, "mensagem")`
 - Exemplos

```
import javax.swing.JOptionPane;

public class Demo_JOptionPane {

    public static void main(String[] args) {

        // método showMessageDialog aplicado à classe JOptionPane
        JOptionPane.showMessageDialog(null, "ISEP");

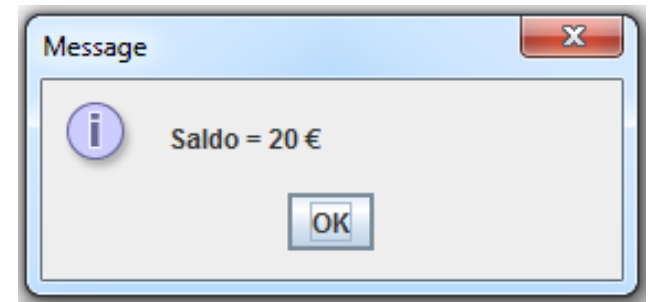
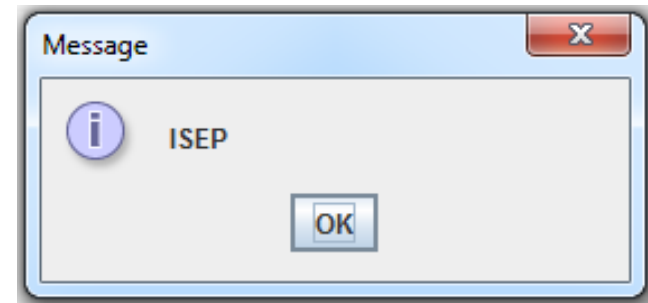
        ...

        ...

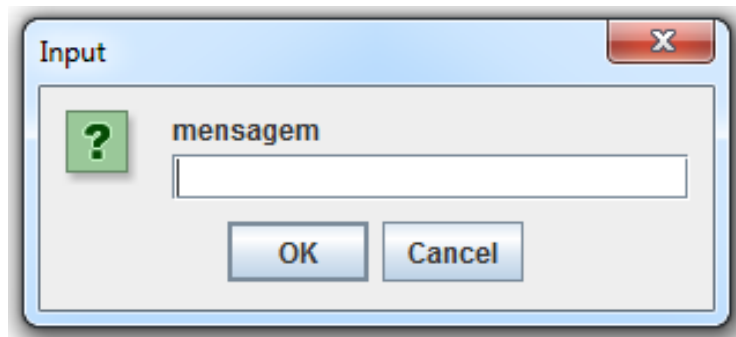
        int x = 20;

        JOptionPane.showMessageDialog(null, "Saldo = " + x + " €");

        ...
    }
}
```



- Método de Classe `showInputDialog` (1/2)
 - Método aplicado à classe `JOptionPane`
 - Sintaxe
 - `String showInputDialog("mensagem")`



- Retorno
 - String contendo dados fornecidos pelo utilizador na caixa de texto
 - `Cancel` retorna string vazia (`""`)

- Método de Classe `showInputDialog`

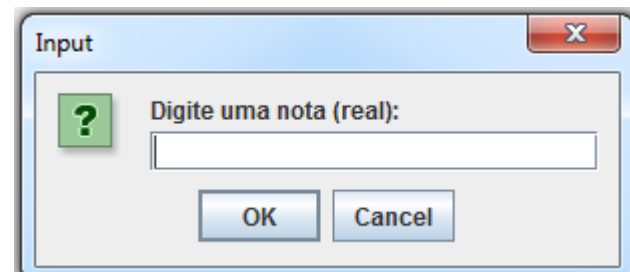
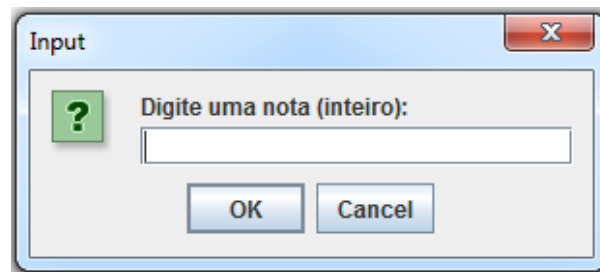
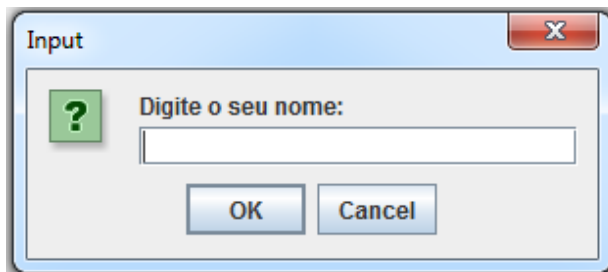
- Exemplos


```
import javax.swing.JOptionPane;

public class Demo_JOptionPane {

    public static void main(String[] args) {

        // método showMessageDialog aplicado à classe JOptionPane
        String nome = JOptionPane.showInputDialog("Digite o seu nome:");
        ...
        // método de classe parseInt da classe Integer converte String em int
        int nota = Integer.parseInt( JOptionPane.showInputDialog("Digite uma nota (inteiro):"));
        ...
        // método de classe parseFloat da classe Float converte String em float
        float nota = Float.parseFloat( JOptionPane.showInputDialog("Digite uma nota (real):") );
    }
}
```



- [Scanner](#)
- [JOptionPane](#)
-  [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)

- [Introdução](#)
- Métodos de **Instância**
 - [length](#)
 - [toLowerCase](#)
 - [toUpperCase](#)
 - [trim](#)
 - [substring](#)
 - [charAt](#)
 - [equalsIgnoreCase](#)
 - [equals](#)
 - [compareTo](#)
 - [compareToIgnoreCase](#)
 - [replace](#)
 - [split](#)
- Método de **Classe**
 - [format](#)

▪ Significado de String

- Cadeia de caracteres = texto
- Exemplos
 - "ISEP"
 - "Algoritmia e Programação"
 - "Nico"

▪ Interesses da Classe String

- Guardar valores do tipo String
 - Cada objeto String guarda uma string
 - Exemplo

```
String s = "ISEP";      // ⇔ String s = new String("ISEP");  
String v = "";          // String vazia
```

- Usar funções sobre strings
 - Exemplo
 - Método de classe [format](#)
 - Permite formatar string passada por parâmetro

- **Aplicáveis**

- Objetos do tipo String

- **Exemplos** // considerando: `String s = "Aprog";`

- `int` **length()**

- Retorna comprimento da string (quantidade de caracteres)

- Exemplo: `int comprimento = s.length(); // ⇒ comprimento = 5`

- `String` **toLowerCase()**

- Retorna a string com todas as letras minúsculas

- Exemplo: `String s2 = s.toLowerCase(); // ⇒ s2 = "aprog"`

- `String` **toUpperCase()**

- Retorna a string com todas as letras maiúsculas

- Exemplo: `String s2 = s.toUpperCase(); // ⇒ s2 = "APROG"`

- `String` **trim()**

- Retorna cópia da string sem espaços brancos iniciais e finais

- Exemplo: `String s = " APROG ";
s2 = s.trim(); // ⇒ s2 = "APROG"`

- **Exemplos** // considerando: `String s = "Aprog";`
 - `String substring(int índiceInício, int índiceFim)`
 - Retorna nova string = substring da string sobre a qual é aplicado este método
 - Substring começa no índiceInício e estende-se até ao índiceFim-1
 - Comprimento da substring = índiceFim - índiceInício
 - Exemplo

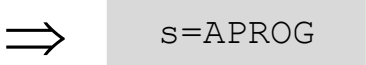
```
String s2 = s.substring(1,3);    // ⇒ s2 = "pr"
```
 - `String substring(int índiceInício)`
 - Retorna nova string = substring da string sobre a qual é aplicado este método
 - Substring começa no índiceInício e estende-se até ao fim da string
 - Exemplo

```
String s2 = s.substring(1);    // ⇒ s2 = "prog"
```
 - `char charAt(int índice)`
 - Retorna carater que se encontra na posição índice
 - Exemplos

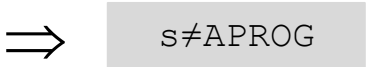
```
char c = s.charAt(0);    // ⇒ c = 'A'
char c = s.charAt(1);    // ⇒ c = 'p'
```

- **Exemplos** // considerando: `String s = "Aprog";`
 - boolean **equalsIgnoreCase**(String outraString)
 - Compara duas strings alfabeticamente e retorna
 - true Ex: `s.equalsIgnoreCase("APROG")` // compara strings s e "APROG"
 - false Ex: `s.equalsIgnoreCase("ALIN")`
 - Não distingue maiúsculas de minúsculas
 - Exemplo:

```
if ( s.equalsIgnoreCase("APROG") )
    System.out.println(" s=APROG ");
else
    System.out.println(" s≠APROG ");
```


 - boolean **equals**(String outraString)
 - Diferença do método anterior
 - Distingue maiúsculas de minúsculas
 - Exemplo:

```
if ( s.equals("APROG") )
    System.out.println(" s=APROG ");
else
    System.out.println(" s≠APROG ");
```



- **Exemplos** // considerando: `String s = "Aprog";`

- `int compareTo(String outraString)`

- Compara duas strings alfabeticamente e retorna um número inteiro

- Negativo Ex: `s.compareTo("PPROG")` `s` é anterior a "PPROG"
- Positivo Ex: `s.compareTo("ALIN")` `s` é posterior a "ALIN"
- Zero Ex: `s.compareTo("Aprog")` `s` e "Aprog" são iguais

Interesse

- Ordenação de strings

- Distingue maiúsculas de minúsculas

- Exemplos

```
if ( s.compareTo("APROG")==0 )
    System.out.println(" s=APROG ");
else
    System.out.println(" s≠APROG ");
```



`s≠APROG`

```
String s1 = ler.next(); // ler é instância de Scanner para ler teclado
String s2 = ler.next();
if ( s1.compareTo(s2)<0 )
    System.out.println("Strings ordenadas:" + s1 + "," + s2);
else
    System.out.println("Strings ordenadas:" + s2 + "," + s1);
```

- `int compareToIgnoreCase(String outraString)`

- Diferença do método anterior
 - Não distingue maiúsculas de minúsculas

- Exemplos

- String **replace**(char antigo , char novo)
 - Retorna nova string com carater antigo substituído pelo carater novo (em todas as ocorrências)
 - Exemplo

```
String s = "solo";  
String s2 = s.replace('o','a');           // ⇒ s2 = "sala"
```

- Interesse

- Decompor uma string em partes separadas por delimitadores (iguais ou diferentes)

- Exemplo: Delimitadores Iguais

```
String registo = "joão/1961/10/1"; // campos do registo separados por /
```



- Obtenção dos campos do registo

```
String[] campos; // vetor para guardar partes da string registo (campos)
campos = registo.split("/"); // 1º Decompõe registo pelo delimitador /
// 2º Cria vetor de strings (comprimento = nº partes)
// 3º Guarda as partes em elementos diferentes
```

- Resultado

campos	"joão"	"1961"	"10"	"1"
	0	1	2	3

- Algoritmia

```
ED TEXTO registo, campos[ ]
INÍCIO
    registo ← "joão/1961/10/1"
    campos ← separar(registo, "/")
    ...
FIM
```

- Exemplo: Delimitadores Diferentes

```
String s = "4-5+6-7+8+9";  
String[] c = s.split("[+-]");
```

- Resultado

c	"4"	"5"	"6"	"7"	"8"	"9"
	0	1	2	3	4	5

- `String format(String formato [, parametro1, parametro2, ..., parametroN])` // [...] significa opcional

- Semelhante ao método `System.out.printf`
- **Retorna** string formatada
- Tem um ou mais parâmetros
 - **formato** especifica formato da string retornada
define texto e especificadores de formato da lista de parâmetros (1, 2, ... N)

- Exemplo 1

```
double x = 12.346267;
String mensagem = String.format("x=%8.2f", x);
```

⇒

mensagem: X= 12,35

2
8

formato inclui

Texto
(retorno literal)

+

Especificador de formato
(representa parâmetro x e especifica o
seu formato: 8.2)

3 espaços = 8-5

- Exemplo 2

```
double media = 12.346267;
String nome = "Ana";
String mensagem = String.format("A média das notas da aluna %s é %.2f", nome, media);
```

// **String retornada** mensagem = "A média das notas da aluna Ana é 12,35"

- **Parâmetro formato**

- É uma **string**
- Especifica formato da string retornada pelo método
- Pode incluir
 - Texto // retornado literalmente
 - Carateres especiais // Ex: **\n** (newline) **\t** (tab)
 - Especificadores de formato
 - Tipos
 - Especiais // Ex: **%n** (newline) e **%%** (percentagem)
 - Normais // especificam formato da lista de parâmetros (1, 2, ..., N)
 - Correspondência entre especificador (E) e parâmetro (P)
 - Por omissão
 - Estabelecida pela ordem dos parâmetros ($E1 \Leftrightarrow P1$, $E2 \Leftrightarrow P2$,...)
 - Exemplo

```
String s = String.format("Nota de %s é %d", nome, nota);
```
- Método **substitui** especificadores pelos parâmetros correspondentes
 - Ex: `s = "Nota de Nico é 20"` // nome=Nico e nota=20

▪ Especificador de formato do parâmetro formato

- Sintaxe: **% [índice_parametro\$] [flags] [dimensão] [.decimal] conversão** // [...] = opcional
 - índice_par. : índice do parâmetro de substituição (inicial é 1) Ex: %1\$d (" %1\$d %1\$d",i)
 - flags:
 - // alinhamento à esquerda (direita por omissão) Ex: %-4d, %-10s
 - + // adiciona sinais para números positivos e negativos Ex: %+d +345 ou -498
 - 0 // adiciona zeros à frente até preencher campo Ex: %04d 0045
 - , // adiciona separador de grupos Ex: %,2f 3.456.789,23
 - dimensão : tamanho total reservado ao campo
 - decimal : número de casas decimais
 - conversão: tipo de parâmetro
 - d inteiro
 - f floating-point (double ou float)
 - s string
 - c carater
 - b booleano
 - o octal
 - h hexadecimal
- Especiais: não são substituídos por parâmetros
 - n newline (evita problemas do \n porque é newline da plataforma)
 - % percentagem

▪ Exemplos

```
int nota=20; String nome="Nico";
String s = String.format("Nota de %s é %d %n", nome, nota); //s= Nota de Nico é 20
double taxa=12.34345;
String s = String.format("Taxa de reprovados: %.2f%%",taxa); //s= Taxa de reprovados: 12.34%
```


- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)

- Métodos para
 - Determinar categoria de carater
 - Letra
 - Minúscula
 - Maiúscula
 - Dígito
 - Converter caracteres
 - Minúsculas para maiúsculas
 - Vice-versa
 - Determinar valor numérico
 - Carateres do tipo dígito

- boolean **isLetter**(char **c**) // Verifica se carater **c** é letra
- boolean **isDigit**(char **c**) // Verifica se carater **c** é dígito
- boolean **isSpaceChar**(char **c**) // Verifica se carater **c** é carater espaço
- boolean **isUpperCase**(char **c**) // Verifica se carater **c** é maiúsculo
- boolean **isLowerCase**(char **c**) // Verifica se carater **c** é minúsculo
- char **toUpperCase**(char **c**) // Converte carater **c** num carater maiúsculo
- char **toLowerCase**(char **c**) // Converte carater **c** num carater minúsculo
- int **getNumericValue**(char **c**) // Retorna valor int que o carater **c** representa
// Retorna -1 se carater **c** não representa um int

▪ Exemplos

```
char c = 'A';  
if ( Character.isLetter(c) )  
    System.out.println(c + " é uma letra");  
  
char c2 = Character.toLowerCase(c);  
  
String s = "1T2X";  
int digito = Character.getNumericValue(s.charAt(0)); // digito = 1  
int codigo = s.charAt(0); // codigo = 49
```

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
-  [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)

- **Package**

- java.util

- **Interesse**

- Escrita Formatada
 - Ficheiro de texto
 - Saída da consola // Exemplo: Janela Linha de Comandos

- **Exemplos**

- **Escrita** de números reais com apenas n **casas decimais**
 - Exemplo: 12.5 % // com apenas 1 casa decimal
 - **Escrita** de **tabela** (colunas de largura fixa e alinhamentos específicos)

NOME	IDADE
João	10

// alinhamentos à direita

1. Declaração de objeto da classe Formatter

- Exemplo

```
Formatter out = new Formatter(System.out) ;
```

- `System.out` representa a saída da consola
- Operador `new` contrói um objeto da classe `Formatter` para escrever na saída da consola
- `out` é uma instância (objeto) da classe `Formatter`

2. Escrita através do método format

- Semelhante aos métodos

- `format` da classe `String` `// String.format(...)`
- `printf` de `System.out` `// System.out.printf(...)`

- Exemplo

```
String nome = "Nico";  
int idade = 23;  
out.format("Nome:%s %n Idade:%d %n", nome, idade);
```



```
Nome: Nico  
Idade: 23
```

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)



- **Package**

- java.util

- **Interesse**

- Processar Tempos e Datas

- **Exemplos**

```
Calendar hoje = Calendar.getInstance();           // hoje: guarda instante de tempo actual
int dia = hoje.get(Calendar.DAY_OF_MONTH);
int mes = hoje.get(Calendar.MONTH);
int ano = hoje.get(Calendar.YEAR);
int hora = hoje.get(Calendar.HOUR);
int minuto = hoje.get(Calendar.MINUTE);
int segundo = hoje.get(Calendar.SECOND);
```

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)



▪ Interesse

- Constantes Matemáticas // **atributos** da classe
- Operações Numéricas Básicas e Funções Trigonométricas // **métodos** da classe

▪ Métodos de Classe

- `double sqrt(double a)` // Retorna raiz quadrada de **a**; Exemplo: `d=Math.sqrt(3);`
- `tipo abs(tipo a)` // **tipo** = double, float, int ou long
// Retorna valor absoluto de **a**. Ex: `Math.abs(-5)`
- `double pow(double a, double b)` // Retorna **a^b**; Exemplo: $2^5 \Leftrightarrow \text{Math.pow}(2,5)$
- `double exp(double a)` // Retorna **e^a**; Exemplo: $e^5 \Leftrightarrow \text{Math.exp}(5)$
- `double random()` // Retorna nº real aleatório do intervalo [0,1[
- `double cos(double angulo)` // Retorna coseno do **angulo** em radianos
- `double sin(double angulo)` // Retorna seno do **angulo** em radianos
- `double acos(double a)` // Retorna arco coseno de **a** em radianos
- `double toDegrees(double angulo)` // Converte **angulo** em radianos para graus
- `double toRadians(double angulo)` // Converte **angulo** em graus para radianos

▪ Atributos

- `final double PI` // 3.14... Exemplo: `Math.PI`
- `final double E` // 2.71... Exemplo: `Math.E`


- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)



- **Método parseInt**

- Método de Classe
 - Aplicável à classe
- Interesse
 - Converter uma string em int
 - Requisito
 - Carateres da string devem ser dígitos decimais
 - Exceção
 - 1º carater
 - Sinal menos ... para indicar nº negativo
- Sintaxe
 - `int parseInt(String s)`
- Exemplo

```
String x = "-12";  
int y = Integer.parseInt(x);  
  
String x2 = "-12";  
int y2 = Integer.parseInt(x2);
```

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- ▪ [Long](#)
- [Float](#)
- [System](#)

- **Método parseLong**

- Método de Classe
 - Aplicável à classe
- Interesse
 - Converter uma string em long
 - Requisito
 - Carateres da string devem ser dígitos decimais
 - Exceção
 - 1º carater
 - Sinal menos ... para indicar nº negativo
- Sintaxe
 - `long parseLong(String s)`
- Exemplos

```
String x = "129876506543";
```

```
long y = Long.parseLong(x);
```

```
String x2 = "-129876506543";
```

```
long y2 = Long.parseLong(x2);
```

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)



- **Método parseFloat**

- Método de Classe
 - Aplicável à classe
- Interesse
 - Converter uma string em float
- Sintaxe
 - `float parseFloat(String s)`

- **Exemplo**

```
String x = "12.98";  
float y = Float.parseFloat(x);  
  
String x2 = "-12.98";  
float y2 = Float.parseFloat(x2);
```

- [Scanner](#)
- [JOptionPane](#)
- [String](#)
- [Character](#)
- [Formatter](#)
- [Calendar](#)
- [Math](#)
- [Integer](#)
- [Long](#)
- [Float](#)
- [System](#)



▪ Semântica

- Escreve **string formatada** ... na saída do computador

▪ Sintaxe

- **System.out.printf**(String **formato** [, parametro1, parametro2, ..., parametroN])
- Tem **um ou mais** parâmetros
 - [...] significa opcional
 - **formato** especifica formato da string escrita
define texto e especificadores de formato da lista de parâmetros (1,2, ... N)

▪ Exemplo 1

```
double x = 12.346267;  
System.out.printf("X=%8.2f", x);
```

⇒

Saída: X= 12,35

formato incluiTexto
(escrita literal)

+

Especificador de formato
(representa parâmetro x e
especifica o seu formato: 8.2)

3 espaços = 8-5

▪ Exemplo 2


```
double media = 12.346267;  
String nome = "Ana";  
System.out.printf("A média das notas da aluna %s é %.2f", nome, media);
```

// **Saída formatada**: A média das notas da aluna Ana é 12,35

- **Parâmetro formato**

- É uma **string**
- Especifica formato da string que o método escreve
- Pode incluir
 - Texto // escrito literalmente
 - Carateres especiais // Ex: **\n** (newline) **\t** (tab)
 - Especificadores de formato
 - Tipos
 - Especiais // Ex: **%n** (newline) e **%%** (percentagem)
 - Normais // especificam formato da lista de parâmetros (1, 2, ..., N)
 - Correspondência entre especificador (E) e parâmetro (P)
 - Por omissão
 - Estabelecida pela ordem dos parâmetros ($E1 \Leftrightarrow P1$, $E2 \Leftrightarrow P2$,...)
 - Exemplo

```
System.out.printf("Nota de %s é %d", nome, nota);
```


- Método **substitui** especificadores pelos parâmetros correspondentes
 - Ex: `s = "Nota de Nico é 20"` // nome=Nico e nota=20

▪ Especificador de formato do parâmetro formato

- Sintaxe: **% [índice_parametro\$] [flags] [dimensão] [.decimal] conversão** // [...] = opcional
 - índice_par. : índice do parâmetro de substituição (inicial é 1) Ex: %1\$d (" %1\$d %1\$d",i)
 - flags:
 - // alinhamento à esquerda (direita por omissão) Ex: %-4d, %-10s
 - + // adiciona sinais para números positivos e negativos Ex: %+d +345 ou -498
 - 0 // adiciona zeros à frente até preencher campo Ex: %04d 0045
 - , // adiciona separador de grupos Ex: %,2f 3.456.789,23
 - dimensão : tamanho total reservado ao campo
 - decimal : número de casas decimais
 - conversão: tipo de parâmetro
 - d inteiro
 - f floating-point (double ou float)
 - s string
 - c carater
 - b booleano
 - o octal
 - h hexadecimal
- Especiais: **não** são **substituídos** por parâmetros
 - n newline (evita problemas do \n porque é newline da plataforma)
 - % percentagem

▪ Exemplos

```
int nota=20; String nome="Nico";
System.out.printf("Nota de %s é %d %n", nome, nota); //saída: Nota de Nico é 20
double taxa=12.34345;
System.out.printf("Taxa de reprovados: %.2f%%",taxa); //saída: Taxa de reprovados: 12.34%
```