

Linguagem de Programação I

Aula 3

Funções Básicas da API Java

`l.bertholdo@ifsp.edu.br`

Conteúdo

- Funções Básicas da API Java
 - Funções Matemáticas
 - Manipulação de Cadeias de Caracteres
 - Manipulação de Data e Hora
 - Formatação de Dados
 - Conversão de Tipos

Funções Básicas da API Java

- Antes de mais nada, o que é uma API?

***API (Application Programming Interface)** é um conjunto de programas de suporte destinados a cumprir funções específicas, os quais podem ser reutilizados em outros programas sempre que necessário. Uma API é dividida em diferentes partes funcionais chamadas “pacotes” ou “bibliotecas”.*

- A **API Java** fornece várias classes que visam simplificar e automatizar tarefas específicas comumente encontradas durante o desenvolvimento de aplicações baseadas nessa linguagem.
- Por exemplo: arredondamento de números, operações matemáticas, formatação de dados, conversão de tipos de dados e manipulação de strings e datas.

Funções Matemáticas

- Métodos definidos na classe **Math**, utilizados em expressões e cálculos matemáticos.
- Principais métodos:
 - round
 - floor
 - ceil
 - sqrt
 - pow

Funções Matemáticas

- Método **round**
 - Arredonda um número real para o número inteiro mais próximo.

Sintaxe: `Math.round(<número real>)`

```
System.out.println(" Valor retornado: " + Math.round(1.37));  
System.out.println(" Valor retornado: " + Math.round(2.8));  
System.out.println(" Valor retornado: " + Math.round(-1.37));  
System.out.println(" Valor retornado: " + Math.round(-2.8));
```

```
Valor retornado: 1  
Valor retornado: 3  
Valor retornado: -1  
Valor retornado: -3
```

```
// Para números com casa decimal ".5", o método arredonda para o número inteiro mais alto.  
System.out.println(" Valor retornado: " + Math.round(3.5));  
System.out.println(" Valor retornado: " + Math.round(4.5));  
System.out.println(" Valor retornado: " + Math.round(-3.5));  
System.out.println(" Valor retornado: " + Math.round(-4.5));
```

```
Valor retornado: 4  
Valor retornado: 5  
Valor retornado: -3  
Valor retornado: -4
```

Funções Matemáticas

- Método **floor**
 - Arredonda um número real para o próximo inteiro **menor** que o número especificado.

Sintaxe: `Math.floor(<número real>)`

```
System.out.println(" Valor retornado: " + Math.floor(1.37));  
System.out.println(" Valor retornado: " + Math.floor(2.8));  
System.out.println(" Valor retornado: " + Math.floor(-1.37));  
System.out.println(" Valor retornado: " + Math.floor(-2.8));
```

```
Valor retornado: 1.0  
Valor retornado: 2.0  
Valor retornado: -2.0  
Valor retornado: -3.0
```

```
// Para números com casa decimal ".5", vale a mesma regra.  
System.out.println(" Valor retornado: " + Math.floor(3.5));  
System.out.println(" Valor retornado: " + Math.floor(4.5));  
System.out.println(" Valor retornado: " + Math.floor(-3.5));  
System.out.println(" Valor retornado: " + Math.floor(-4.5));
```

```
Valor retornado: 3.0  
Valor retornado: 4.0  
Valor retornado: -4.0  
Valor retornado: -5.0
```

Funções Matemáticas

- Método **ceil**
 - Arredonda um número real para o próximo inteiro **maior** que o número especificado.

Sintaxe: `Math.ceil(<número real>)`

```
System.out.println(" Valor retornado: " + Math.ceil(1.37));  
System.out.println(" Valor retornado: " + Math.ceil(2.8));  
System.out.println(" Valor retornado: " + Math.ceil(-1.37));  
System.out.println(" Valor retornado: " + Math.ceil(-2.8));
```

```
Valor retornado: 2.0  
Valor retornado: 3.0  
Valor retornado: -1.0  
Valor retornado: -2.0
```

```
// Para números com casa decimal ".5", vale a mesma regra.  
System.out.println(" Valor retornado: " + Math.ceil(3.5));  
System.out.println(" Valor retornado: " + Math.ceil(4.5));  
System.out.println(" Valor retornado: " + Math.ceil(-3.5));  
System.out.println(" Valor retornado: " + Math.ceil(-4.5));
```

```
Valor retornado: 4.0  
Valor retornado: 5.0  
Valor retornado: -3.0  
Valor retornado: -4.0
```

Funções Matemáticas

- Método **sqrt**
 - Calcula a raiz quadrada de um número maior ou igual a zero.

Sintaxe: `Math.sqrt(<número>)`

```
System.out.println(" Valor retornado: " + Math.sqrt(81));
```

```
Valor retornado: 9.0
```

* Existe também o método **Math.cbrt(<número>)** para calcular raiz cúbica.

Funções Matemáticas

- Método **pow**
 - Realiza operações de pontenciação.

Sintaxe: `Math.pow(<base>, <expoente>)`

```
System.out.println(" Valor retornado: " + Math.pow(5, 3));
```

```
Valor retornado: 125.0
```

Manipulação de Cadeias de Caracteres

- Métodos definidos na classe **String**, utilizados para manipular cadeias de caracteres (*strings*).
- Principais métodos:
 - length
 - toLowerCase e toUpperCase
 - trim
 - substring
 - replace
 - equals
 - contains
 - startsWith e endsWith
 - indexOf e lastIndexOf

Manipulação de Cadeias de Caracteres

- Método **length**
 - Retorna o número de caracteres de uma string.

```
String texto = "Município de São Paulo";  
System.out.println(texto.length());
```

Manipulação de Cadeias de Caracteres

- Métodos **toLowerCase** e **toUpperCase**
 - **toLowerCase** converte todas as letras de uma string para minúsculas.
 - **toUpperCase** converte todas as letras de uma string para maiúsculas.

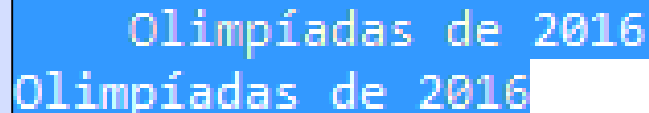
```
String texto = " Olimpíadas de 2016 ";  
System.out.println(texto.toLowerCase());  
System.out.println(texto.toUpperCase());
```

```
olimpíadas de 2016  
OLIMPÍADAS DE 2016
```

Manipulação de Cadeias de Caracteres

- Método **trim**
 - Remove todos os espaços que se encontram no início e no final de uma string.

```
String texto = "    Olimpíadas de 2016    ";  
System.out.println(texto);  
System.out.println(texto.trim());
```



```
Olimpíadas de 2016  
Olimpíadas de 2016
```

** Para remover espaços no meio de uma cadeia de caracteres, deve ser usado o método **replace**.*

Manipulação de Cadeias de Caracteres

- Método **substring**

- Retorna parte de uma string com base em uma posição inicial e uma posição final.

Sintaxe: <string>.substring(<posição inicial>, [<posição final*>])

** Se não informada, retorna até o último caractere presente na string.*

```
String texto = "R$150,00";  
System.out.println(texto.substring(0, 5));  
System.out.println(texto.substring(2, 5));  
System.out.println(texto.substring(2));
```

Observe que o caractere presente na posição final (5) é descartado.

```
R$150  
150  
150,00
```

Manipulação de Cadeias de Caracteres

- Método **replace**

- Substitui parte dos caracteres de uma string por outra.

Sintaxe: <string>.replace(<string a ser substituída>, <string substituta>)

```
String texto = "Apartamento 53";  
texto = texto.replace("Apartamento", "Apto");  
System.out.println(texto);
```

Apto 53

Manipulação de Cadeias de Caracteres

- Método **equals**
 - Retorna se uma cadeia de caracteres é **igual** a outra.
 - O método **equals** é *case-sensitive*, ou seja, faz distinção entre letras maiúsculas e minúsculas.

```
String texto1 = "Livro";  
String texto2 = "Livro";  
String texto3 = "livro";  
String texto4 = "Livro ";  
  
System.out.println(texto1.equals(texto2));  
System.out.println(texto1.equals(texto3));  
System.out.println(texto1.equals(texto4));
```

```
true  
false  
false
```


Manipulação de Cadeias de Caracteres

- Método **equals**

- Ao comparar cadeias de caracteres, deve-se dar preferência ao método **equals**, pois nem sempre o operador “==” apresenta comportamento correto, quando usado para comparar tipos de dados que não são primitivos, como o tipo **String** por exemplo.

Ao verificar que os conteúdos de **txt1** e **txt2** são iguais, o compilador aloca o texto “Azul Claro” em uma mesma área da memória e associa as duas variáveis a esta área.

```
String txt1 = "Azul Claro";  
String txt2 = "Azul Claro";  
String parte1 = "Azul ";  
String parte2 = "Claro";  
String txt3 = parte1 + parte2;
```

Ao atribuir o conteúdo da variável **txt3** em tempo de execução (por meio da variáveis **parte1** e **parte2**), o texto “Azul Claro” desta variável é alocado em outra área de memória.

```
true  
false  
true  
true
```

```
System.out.println(txt1 == txt2); // Verifica se as variáveis ocupam o mesmo local na memória.  
System.out.println(txt1 == txt3); // Verifica se as variáveis ocupam o mesmo local na memória.
```

```
System.out.println(txt1.equals(txt2)); // Compara se os conteúdos das variáveis são iguais.  
System.out.println(txt1.equals(txt3)); // Compara se os conteúdos das variáveis são iguais.
```

Manipulação de Cadeias de Caracteres

- Método **contains**
 - Retorna se uma cadeia de caracteres **contém** com uma determinada string.
 - O método **contains** é *case-sensitive*, ou seja, faz distinção entre letras maiúsculas e minúsculas.

```
String texto = "Quando o Sol bater na janela do teu quarto";  
System.out.println(texto.contains("jan"));  
System.out.println(texto.contains("JAN"));  
System.out.println(texto.contains(" na "));
```

```
true  
false  
true
```

Manipulação de Cadeias de Caracteres

- Métodos **startsWith** e **endsWith**
 - **startsWith** retorna se uma cadeia de caracteres **inicia** com uma determinada string.
 - **endsWith** retorna se uma cadeia de caracteres **termina** com uma determinada string.
 - Ambos métodos são *case-sensitive*. Para que o método **startsWith** procure a string a partir de uma determinada posição, basta incluir um 2º argumento indicando esta posição.

```
String texto = "Quando o Sol bater na janela do teu quarto";  
System.out.println(texto.startsWith("Q"));  
System.out.println(texto.startsWith("S", 9));  
System.out.println(texto.startsWith("q"));  
System.out.println(texto.endsWith("to"));  
System.out.println(texto.endsWith("TO"));
```

```
true  
true  
false  
true  
false
```

Manipulação de Cadeias de Caracteres

- Métodos **indexOf** e **lastIndexOf**
 - **indexOf** retorna a posição da **1ª ocorrência** de uma string em uma cadeia de caracteres. E **lastIndexOf** retorna a posição da **última ocorrência** de uma string em uma cadeia de caracteres.
 - Caso a string não seja encontrada, é retornado o valor “-1”.
 - Ambos métodos são *case-sensitive*. Para que eles procurem a string a partir de uma determinada posição, basta incluir um 2º argumento indicando esta posição.

```
String texto = "azul, verde, amarelo, azul, amarelo";  
System.out.println(texto.indexOf("azul"));  
System.out.println(texto.indexOf("azul", 2));  
System.out.println(texto.indexOf("AZUL"));  
System.out.println(texto.lastIndexOf("amarelo"));
```

0
22
-1
28

Manipulação de Data e Hora

- Métodos definidos na classe **Calendar** utilizados para manipulação de datas e horas.
- Principais métodos:
 - **set** – Define os dados de uma data e uma hora.
 - **get** – Recupera os dados de uma data ou uma hora a partir de constantes da classe **Calendar**.

1 = Domingo
2 = Segunda
3 = Terça
4 = Quarta
5 = Quinta
6 = Sexta
7 = Sábado

8 = Setembro, pois os meses são representados de 0 à 11.

```
Calendar data_hora = Calendar.getInstance();  
// Ano, Mês, Dia, Hora, Minuto, Segundo  
data_hora.set(2017, 8, 5, 15, 38, 22);
```

```
System.out.println("Dia: " + data_hora.get(Calendar.DAY_OF_MONTH));  
System.out.println("Mês: " + data_hora.get(Calendar.MONTH));  
System.out.println("Ano: " + data_hora.get(Calendar.YEAR));  
System.out.println("Dia da Semana: " + data_hora.get(Calendar.DAY_OF_WEEK));  
System.out.println();
```

```
System.out.println("Horas: " + data_hora.get(Calendar.HOUR));  
System.out.println("Horas: " + data_hora.get(Calendar.HOUR_OF_DAY));  
System.out.println("Minutos: " + data_hora.get(Calendar.MINUTE));  
System.out.println("Segundos: " + data_hora.get(Calendar.SECOND));
```

Dia: 5
Mês: 8
Ano: 2017
Dia da Semana: 3

Horas: 3
Horas: 15
Minutos: 38
Segundos: 22

Manipulação de Data e Hora

- O método **get** também pode ser usado para manipular os valores de uma data ou de uma hora.

```
Calendar data_hora = Calendar.getInstance();
// Ano, Mês, Dia, Hora, Minuto, Segundo
data_hora.set(2017, 8, 5, 15, 38, 22);

data_hora.set(Calendar.DAY_OF_MONTH, data_hora.get(Calendar.DAY_OF_MONTH) + 12);
data_hora.set(Calendar.MONTH, data_hora.get(Calendar.MONTH) - 3);
data_hora.set(Calendar.YEAR, data_hora.get(Calendar.YEAR) + 2);

data_hora.set(Calendar.HOUR, data_hora.get(Calendar.HOUR) + 4);
data_hora.set(Calendar.MINUTE, data_hora.get(Calendar.MINUTE) + 18);
data_hora.set(Calendar.SECOND, data_hora.get(Calendar.SECOND) - 8);

System.out.println("Dia: " + data_hora.get(Calendar.DAY_OF_MONTH));
System.out.println("Mês: " + data_hora.get(Calendar.MONTH));
System.out.println("Ano: " + data_hora.get(Calendar.YEAR));

System.out.println("Horas: " + data_hora.get(Calendar.HOUR_OF_DAY));
System.out.println("Minutos: " + data_hora.get(Calendar.MINUTE));
System.out.println("Segundos: " + data_hora.get(Calendar.SECOND));
```

```
Dia: 17
Mês: 5
Ano: 2019
Horas: 19
Minutos: 56
Segundos: 14
```

Formatação de Dados

- As classes **DecimalFormat**, **NumberFormat**, **String**, **SimpleDateFormat** e **DateFormatSymbols** da API Java fornecem vários métodos para formatação de dados.
- Com eles, é possível formatar informações de diversas naturezas como números, valores monetários, percentuais, números de documentos, datas, etc.

Formatação de Dados

- Método **format** da classe **DecimalFormat**.
 - Arredonda um número real para o número mais próximo, de acordo com uma quantidade especificada de casas decimais.

```
DecimalFormat df = new DecimalFormat("##0.00");  
System.out.println(df.format(57.231));  
System.out.println(df.format(57.287));  
System.out.println(df.format(-57.231));  
System.out.println(df.format(-57.287));  
System.out.println(df.format(57.2));
```

57,23
57,29
-57,23
-57,29
57,20

```
DecimalFormat df = new DecimalFormat("##0.0");  
df.setRoundingMode(RoundingMode.UP); // Arredonda para o número mais distante de zero.  
System.out.println(df.format(53.75));  
System.out.println(df.format(-53.75));  
df.setRoundingMode(RoundingMode.DOWN); // Arredonda para o número mais próximo de zero.  
System.out.println(df.format(53.75));  
System.out.println(df.format(-53.75));
```

53,8
-53,8
53,7
-53,7

```
DecimalFormat df = new DecimalFormat("###,##0.00");  
System.out.println(df.format(48500));
```

48.500,00

Formatação de Dados

- Método **format** da classe **NumberFormat**.

```
double valor = 1278.35;
// Cria um objeto NumberFormat com a formatação da moeda local.
NumberFormat nf1 = NumberFormat.getCurrencyInstance();
System.out.println(nf1.format(valor));

double porcentagem = 0.385;
// Cria um objeto NumberFormat com a formatação de porcentagem.
NumberFormat nf2 = NumberFormat.getPercentInstance();
nf2.setMinimumFractionDigits(2); // Número mínimo de casas decimais.
System.out.println(nf2.format(porcentagem));
```

R\$ 1.278,35
38,50%

- Métodos **format** e **replaceAll** da classe **String**.

```
int numero1 = 23;
System.out.println(String.format("%04d", numero1));

String cpf = "12345678900";
System.out.println(cpf.replaceAll("(\\d{3})(\\d{3})(\\d{3})(\\d{2})", "$1.$2.$3-$4"));

String cnpj = "12345678000199";
System.out.println(cnpj.replaceAll("(\\d{2})(\\d{3})(\\d{3})(\\d{4})(\\d{2})", "$1.$2.$3/$4-$5"));
```

0023
123.456.789-00
12.345.678/0001-99

Formatação de Dados

- Método `format` da classe `SimpleDateFormat`.

```
Calendar data_hora = Calendar.getInstance();
data_hora.set(2017, 8, 5, 15, 38, 22);

// Representações de hora
SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("hh:mm:ss");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("hh:mm");
System.out.println(sdf.format(data_hora.getTime()));

System.out.println();

// Representações de dia
sdf = new SimpleDateFormat("d/MM/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("dd/MM/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

System.out.println();
```

15:38:22

03:38:22

03:38

5/09/2017

05/09/2017

```
// Representações de mês
sdf = new SimpleDateFormat("dd/M/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("dd/MM/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("dd/MMM/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("dd/MMMM/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

System.out.println();

// Representações de ano
sdf = new SimpleDateFormat("dd/MM/yy");
System.out.println(sdf.format(data_hora.getTime()));

sdf = new SimpleDateFormat("dd/MM/yyyy");
System.out.println(sdf.format(data_hora.getTime()));

System.out.println();
```

05/9/2017

05/09/2017

05/set/2017

05/Setembro/2017

05/09/17

05/09/2017

Formatação de Dados

- Métodos “get” da classe **DateFormatSymbols**.

```
Calendar data_hora = Calendar.getInstance();
data_hora.set(2017, 8, 5, 15, 38, 22);

System.out.println("Mês: " + new DateFormatSymbols().
    getMonths()[data_hora.get(Calendar.MONTH)]);
System.out.println("Mês: " + new DateFormatSymbols().
    getShortMonths()[data_hora.get(Calendar.MONTH)]);
System.out.println();

System.out.println("Dia da Semana: " + new DateFormatSymbols().
    getWeekdays()[data_hora.get(Calendar.DAY_OF_WEEK)]);
System.out.println("Dia da Semana: " + new DateFormatSymbols().
    getShortWeekdays()[data_hora.get(Calendar.DAY_OF_WEEK)]);
System.out.println();

System.out.println("Hora (AM/PM): " + data_hora.get(Calendar.HOUR) + ":" +
    data_hora.get(Calendar.MINUTE) + " " +
    new DateFormatSymbols().
    getAmPmStrings()[data_hora.get(Calendar.AM_PM)]);
```

Mês: Setembro

Mês: set

Dia da Semana: Terça-feira

Dia da Semana: Ter

Hora (AM/PM): 3:38 PM

Conversão de Tipos

- Métodos utilizados para conversão de tipos de dados.
- Por exemplo: string para números inteiros e decimais, string para data, string para hora, float para inteiro, double para inteiro, char para string, números inteiros e decimais para string, etc.

Conversão de Tipos

```
// Conversão de String para int
String numero1 = "150";
System.out.println("String->int: " + (Integer.parseInt(numero1) + 10));

// Conversão de String para long
String numero2 = "1234567890123456789";
System.out.println("String->long: " + (Long.parseLong(numero2) + 10));

// Conversão de String para float
String numero3 = "23.56";
System.out.println("String->float: " + (Float.parseFloat(numero3) + 10));

// Conversão de String para double
String numero4 = "78359.17829";
System.out.println("String->double: " + (Double.parseDouble(numero4) + 10));

// Conversão de String para Calendar
String data = "10/02/2016";
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
Calendar d = Calendar.getInstance();
d.setTime(sdf.parse(data));
System.out.println("String->Calendar: \n" +
    "Dia: " + d.get(Calendar.DAY_OF_MONTH) + "\n" +
    "Mês: " + d.get(Calendar.MONTH) + "\n" +
    "Ano: " + d.get(Calendar.YEAR));
```

```
String->int: 160
String->long: 1234567890123456799
String->float: 33.559998
String->double: 78369.17829
String->Calendar:
Dia: 10
Mês: 1
Ano: 2016
```

Para este tipo de conversão, é preciso lançar uma exceção do tipo **ParseException** no método que contém a conversão. **Exemplo:** *public static void main(String[] args) throws ParseException*

Conversão de Tipos

```
int num1 = 36;  
float num2 = 20.72f;  
double num3 = 60.87;
```

```
float -> int: 56  
double -> int: 96
```

```
// Conversão de float para int  
int resultado1 = num1 + (int)num2;  
System.out.println("float -> int: " + resultado1);  
  
// Conversão de double para int  
int resultado2 = num1 + (int)num3;  
System.out.println("double -> int: " + resultado2);
```

```
// Conversão de char para String  
char caractere = 'a';  
String letra = String.valueOf(caractere);  
System.out.println("char -> String: " + letra);  
  
// Conversão de int para String  
int n1 = 123;  
String num = String.valueOf(n1);  
System.out.println("int -> String: " + num);  
  
// Conversão de float para String  
float n2 = 157.33f;  
num = String.valueOf(n2);  
System.out.println("float -> String: " + num);  
  
// Conversão de double para String  
double n3 = 128.65;  
num = String.valueOf(n3);  
System.out.println("double -> String: " + num);
```

```
char -> String: a  
int -> String: 123  
float -> String: 157.33  
double -> String: 128.65
```

Referências

- Peter Jandl Junior; Java – Guia do Programador – 3ª Edição. São Paulo: Novatec Editora, 2015.
- Rafael Santos; Introdução à Programação Orientada a Objetos usando Java – 2ª edição. Rio de Janeiro: Elsevier, 2013.
- Caíque Cardoso; Orientação a Objetos na Prática – Aprendendo Orientação a Objetos com Java. Rio de Janeiro: Editora Ciência Moderna, 2006.