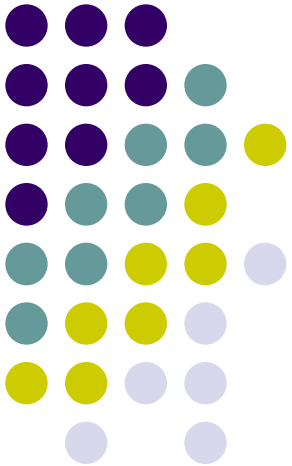
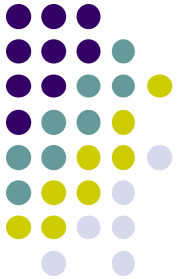
# Programação OO

## Bibliotecas Java



Profª. Roberta B Torres ( rbtorresiff@gmail.com )

# Bibliotecas Java



https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

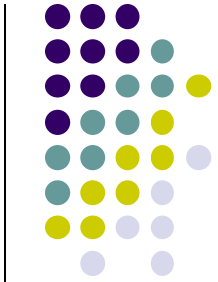
## AS BIBLIOTECAS DE JAVA

Cleyton Caetano de Souza

IFPB-Campus Monteiro

cleyton.caetano.souza@gmail.com

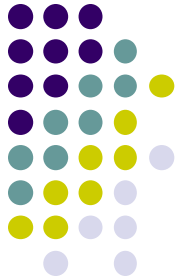
# Bibliotecas Java



- JAVA é uma linguagem OO, isso significa que ao programarmos pensamos nos problemas como uma sequência de interações entre objetos de diferentes Classes
  - Eu vou imaginar as entidades que serão necessárias
    - Num sistema acadêmico
    - Num sistema de compras online
- Muitos projetos tem funcionalidades em comum!
- Não precisamos sempre programar tudo!
  - Já existe muita coisa pronta!
  - Precisamos aprender a usar!

https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

# Bibliotecas Java



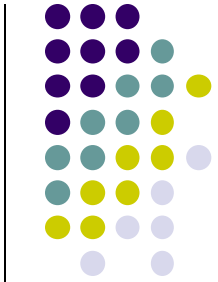
- Java tem uma ampla gama de recursos já pré-definidos.

  ➔ *Desde tocar um som até enviar um e-mail.*

- Esses recursos fazem parte da API.

  Application Programming Interface
  (Interface de Programação de Aplicativos)

# Bibliotecas Java

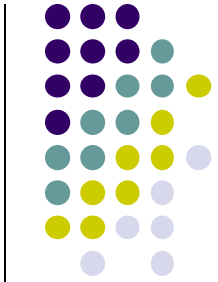


## APIs

- API **é um conjunto de rotinas disponibilizados** por um software **para a utilização das suas funcionalidades** por aplicativos que não pretendem envolver-se nos seus detalhes implementação, **mas apenas usar seus serviços**.

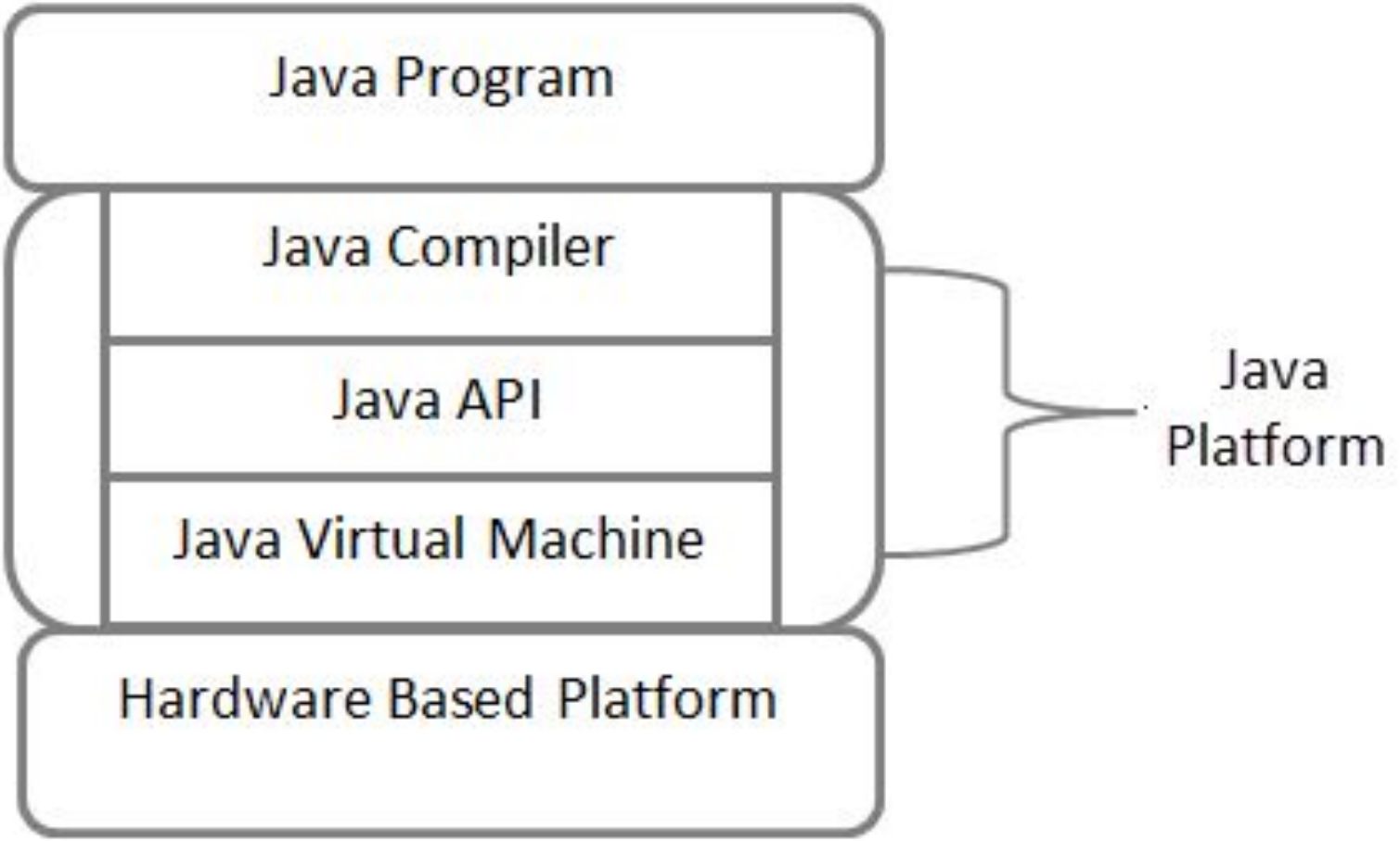
https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

# Bibliotecas Java



- A API é um grupo de programas de suporte destinados a cumprir funções específicas
  - Essas funções estão divididas em diferentes partes (pacotes) relacionadas as suas funcionalidades (o que nós denominamos bibliotecas).
- Para usá-las não precisamos saber como elas realizam tal tarefa, mas apenas como usar.
  - As vezes, descobrir como usar uma classe é intuitivo e simples, outras precisamos consultar a documentação

# Plataforma Java x API




https://www.educba.com/what-is-api-in-java/

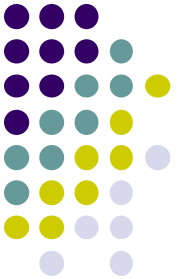
# Classes da API Java

https://docente.ifrn.edu.br/albalopes/semestres-anteriores/2012.1/disciplinas/programacao-orientada-a-objetos-eja/Aula12ClassesdaAPIJava.pdf

- As classes predefinidas em Java são agrupadas em diretórios chamados de pacotes
- Coletivamente, esses pacotes são referidos como Java API
- Para utilizar classes já definidas e disponíveis em pacotes, utiliza-se a instrução **import**.
  - Nessa instrução, especifica-se a localização das classes que são usadas
  - **Exemplo:** ao utilizarmos a classe **Scanner** para ler valores digitados pelo usuário, escrevemos a instrução **import**:

```
import java.util.Scanner;
```

# Bibliotecas Java

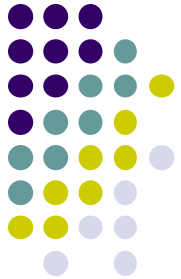


## Pacotes mais usados da API Java

| Pacote API | Recurso |
|---|---|
| java.awt | Recursos Gráficos |
| java.io | Entrada e Saída de Dados |
| java.lang | Recursos Fundamentais da Linguagem Java |
| java.math | Operações Matemáticas |
| java.util | Miscelânea de Recursos Utilitários |

https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

# Bibliotecas Java



**Acesse o link abaixo para visualizar as API´s do Java:**

https://docs.oracle.com/javase/8/docs/api/index.html?overview-summary.html

**Vamos entender como usar a documentação nos próximos slides.**

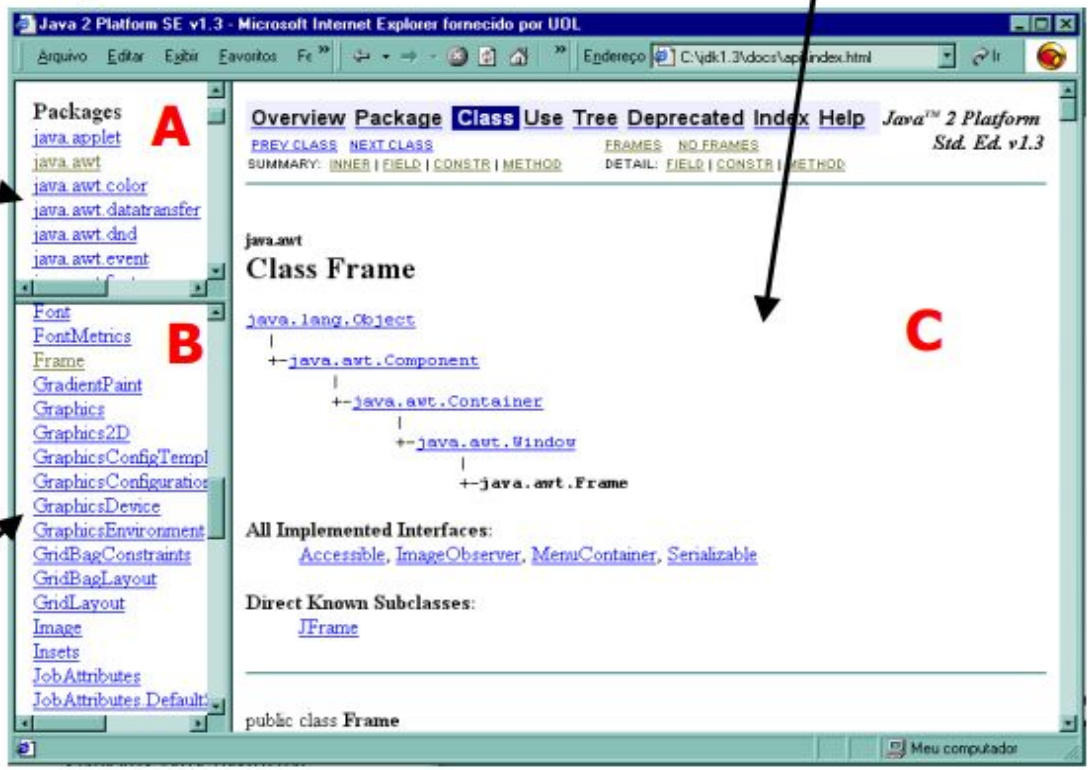
# Bibliotecas Java

*Descrição da classe (escolhida na janela B)*
*- hierarquia*
*- documentação detalhada, métodos, variáveis, etc.*

*Se um método não for encontrado na classe mostrada, procure nas superclasses (use as referências cruzadas)*

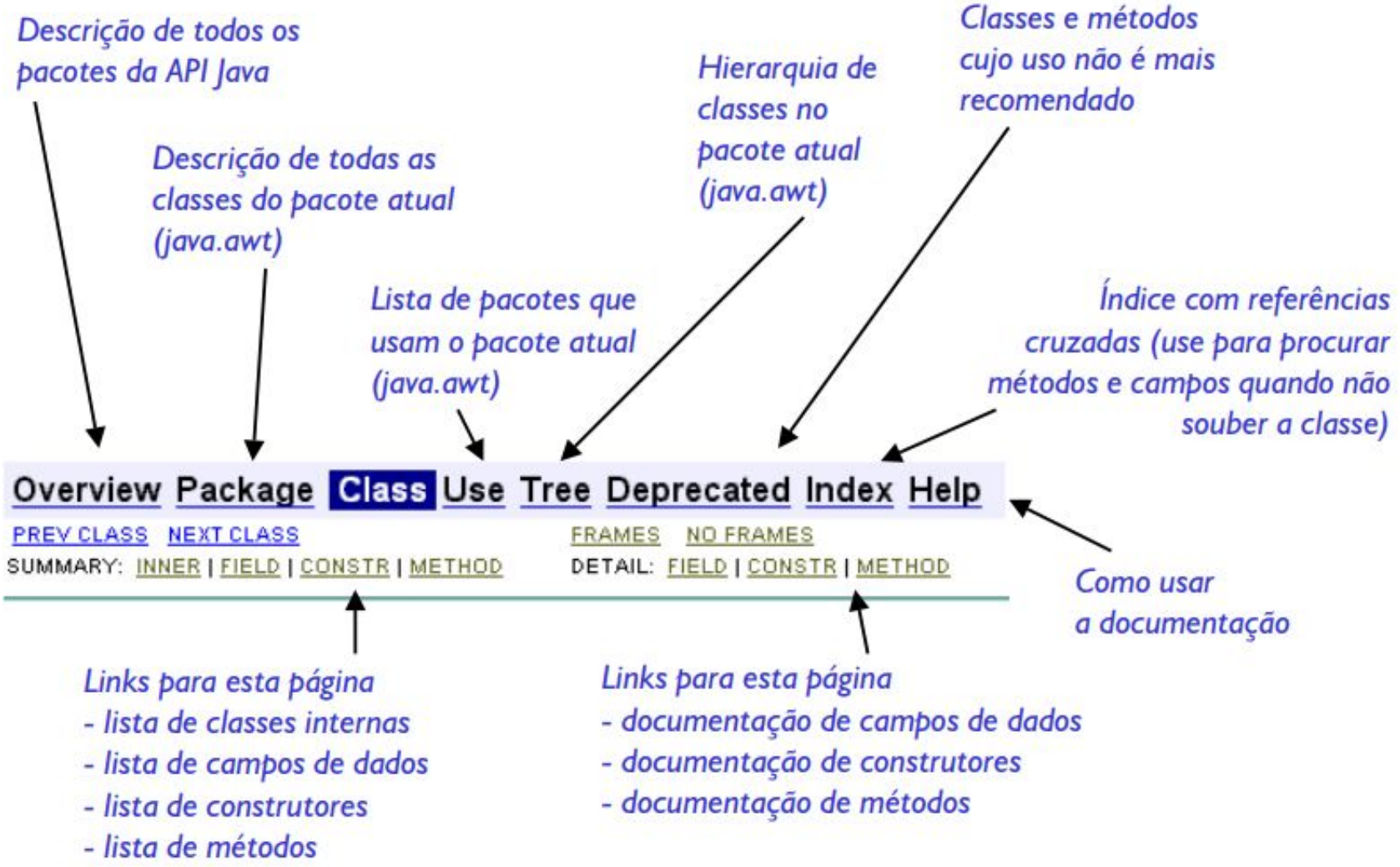
*Lista de pacotes*

*Lista de classes e interfaces do pacote escolhido na janela (A)*



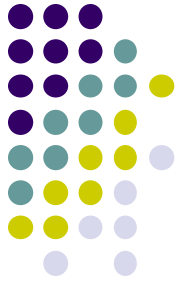

http://www.argonavis.com.br/cursos/java/j100/java_04.pdf

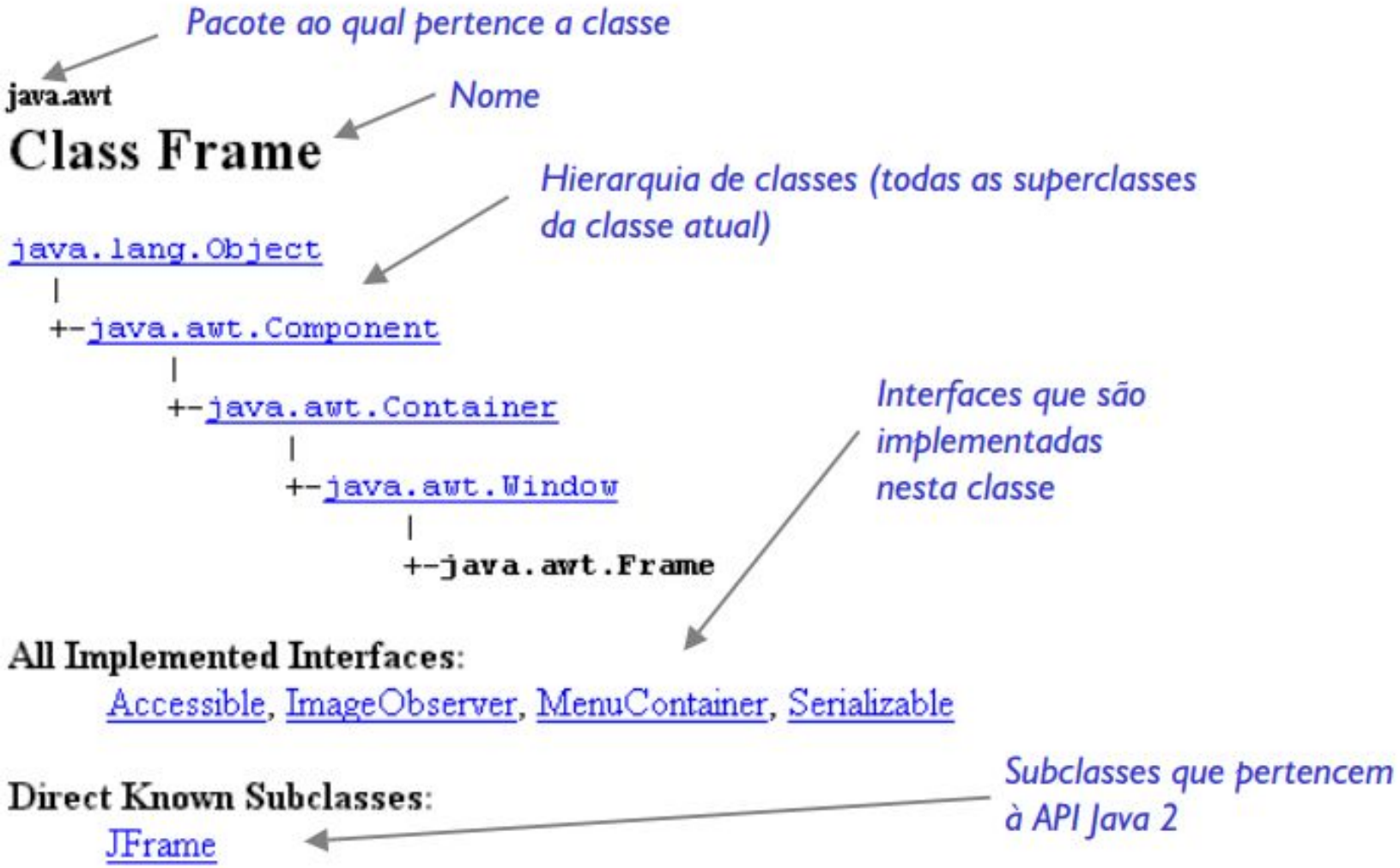
# Bibliotecas Java




http://www.argonavis.com.br/cursos/java/j100/java_04.pdf

# Bibliotecas Java






http://www.argonavis.com.br/cursos/java/j100/java_04.pdf

# Bibliotecas Java

http://www.argonavis.com.br/cursos/java/j100/java_04.pdf

- *Listas de classes internas, campos de dados, métodos e construtores*

# Bibliotecas Java

Vamos conhecer um pouco mais sobre as seguintes API´s:

**MATH**

**STRING**

**DATE**

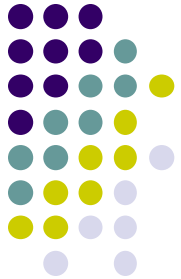
**SimpleDateFormat**

**Calendar**

# Bibliotecas Java

## Classe

# Classe Math



- A classe Math do pacote java.lang contém uma série de métodos matemáticos bastante úteis
  - Ex. Gerar Número Aleatório, Arredondar um número, Calcular a Raiz Quadrada, etc.
- Possui definida duas constantes
  - public final static double PI
  - public final static double E
- **Todos os métodos da classe Math são static**

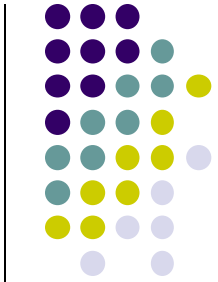
# Classe Math

- Os métodos da classe Math permitem ao programador realizar certos cálculos matemáticos comuns
- Eles não precisam instanciar nenhum objeto antes da utilização, pois são métodos **estáticos** da classe.
- Para utilizá-los, escreve-se:

```
<nome_da_classe>.<nome_do_metodo>(<argumentos>)
```

https://docente.ifrn.edu.br/albalopes/semestres-anteriores/2012.1/disciplinas/programacao-orientada-a-objetos-eja/Aula12ClassesdaAPIJava.pdf

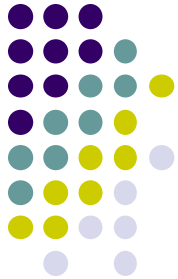
# Classe Math
# Métodos Pré-definidos



https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

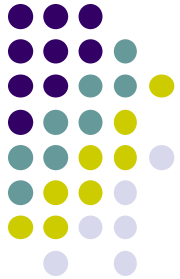
| Método | O que faz | Exemplo de utilização | Resultado |
|---|---|---|---|
| Math.max | Retorna o maior valor entre os valores fornecidos | Math.max(145, 159) | 159 |
| Math.min | Retorna o menor valor entre os valores fornecidos | Math.min(2, 3) | 2 |
| Math.abs | Retorna o módulo do valor passado como parâmetro | Math.abs(-154) | 154 |
| Math.ceil | Arredonda o parâmetro para cima | Math.ceil(8.02) | 9.0 |
| Math.floor | Arredonda para baixo | Math.floor(8.8) | 8.0 |
| Math.round | Arredonda para o valor mais próximo | Math.round(1.5) e Math.round(1.4) | 2 e 1 |

# Classe Math
# Métodos Pré-definidos



| Método | O que faz | Exemplo de utilização | Resultado |
|---|---|---|---|
| Math.random | Retorna um número aleatório no intervalo [0, 1[ | Math.random() | 0.88 |
| Math.sqrt | Retorna a raiz quadrada do número passado como parâmetro | Math.sqrt(4) | 2 |
| Math.pow | Retorna a potência do primeiro parâmetro elevado ao segundo parâmetro | Math.pow(2,3) | 8 |

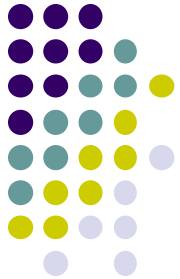
# Classe Math
# Métodos Pré-definidos



| | | | |
|---|---|---|---|
| PI | Constante π | Math.PI | 3.1415. |
| pow | Exponenciação | Math.pow(5,2) | 25 |
| sqrt | Raiz Quadrada | Math.sqrt(25) | 5 |
| cbrt | Raiz Cúbica | Math.cbrt(27) | 3 |

FONTE: https://www.youtube.com/watch?v=W9V5wtOOZHs

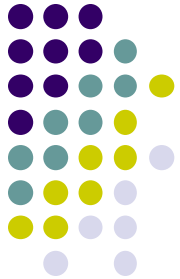
# Classe Math
# Métodos Pré-definidos



| | | | |
|---|---|---|---|
| abs | Valor Absoluto | `Math.abs(-10)` | 10 |
| floor | Arredonda para Baixo | `Math.floor(3.9)` | 3 |
| ceil | Arredonda para Cima | `Math.ceil(4.2)` | 5 |
| round | Arredonda Aritmeticamente | `Math.round(5.6)` | 6 |

FONTE: https://www.youtube.com/watch?v=W9V5wtOOZHs

# Classe Math

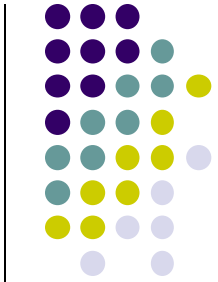


**Exemplo**: Calcular a raiz quadrada de um número (método **.sqrt(double d)** da API Math).

```
public static void main(String [] args){
    double x = 9.0;
    double raiz = Math.sqrt(x);
    System.out.println("A raiz de " + x + " é: " + raiz);
}
```

https://docente.ifrn.edu.br/albalopes/semestres-anteriores/2012.1/disciplinas/programacao-orientada-a-objetos-eja/Aula12ClassesdaAPIJava.pdf

# Classe Math



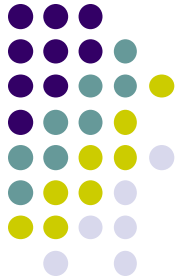
- **Videoaula: Classe Math (9 min) - Canal de Loiane Groner**

  https://www.youtube.com/watch?v=341K_YKI6KQ&list=PLGxZ4Rq3BOBoqYyFWOV_YbfBW80YGAGEI&index=36

- **Exemplo Prático usando a API Math:**

  https://www.youtube.com/watch?v=QjI9I2gUyEs - (9min)

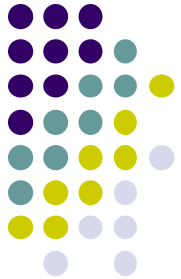
# PRATIQUE: Classe Math



- Escreva um programa que leia dois números, calcule a potência do primeiro elevado ao segundo, e imprima a raiz quadrada desse número arredondando para cima.
- Escreva um programa que gere dois números aleatórios, o primeiro entre 0 e 100 e o segundo entre 1 e 100, em seguida, imprima o maior desses dois números elevado ao cubo.
- Escreva um programa que calcule a raiz cúbica de um número digitado pelo usuário.

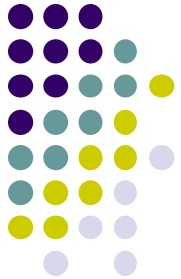
https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

# Bibliotecas Java



# Classe STRING

# Classe String

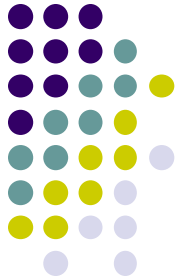


Material Consultado como referência:

- https://sites.google.com/site/cleytoncaetanodesouza/disciplinas/programacao-orientada-a-objetos-poo

- https://docente.ifrn.edu.br/albalopes/semestres-anteriores/2012.1/disciplinas/programacao-orientada-a-objetos-eja/Aula12ClassesdaAPIJava.pdf

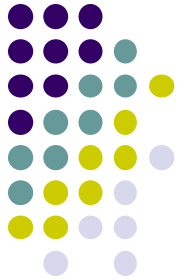
# Classe String



- String é uma classe JAVA que faz parte do pacote java.lang.String
- Os objetos da classe String são tratados como se fossem tipos primitivos (como **int**, **float**, **boolean**)
  - Por esse motivo não é necessário realizar o import quando se utiliza objetos do tipo String

- String é uma cadeia de caracteres
- Em Java
  - Não há um tipo primitivo para String
  - Strings são objetos

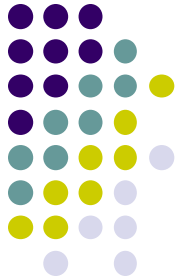
# Classe String
# Métodos de Comparação



- Em Java, String são comparadas através do método **.equals(String s)**. O método **equals** requer que a String que se deseja comparar seja passada por parâmetro (argumento):

```
public static void main(String[] args) {
    String a = "teste";
    String b = "teste";

    if (a.equals(b)){
        System.out.println("String iguais!");
    }else{
        System.out.println("String diferentes!");
    }
}
```

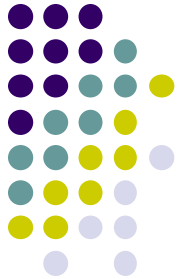
# Classe String
# Métodos de Comparação



- public **boolean** equals(String other)
  - Compara se duas String são iguais
- public **boolean** equalsIgnoreCase(String other)
  - Compara se duas Strings são iguais considerando letras maiúsculas iguais a minúsculas

```
String nome = "teste";
System.out.println(nome.equals("Teste"));
System.out.println(nome.equalsIgnoreCase("Teste"));
```

# Classe String
# Método indexOf



- Método para retornar caractere em determinado índice
  - Retorna o índice do caractere **c** passado por parâmetro: **indexOf(char c)**
  - As strings começam a contar do caractere 0:

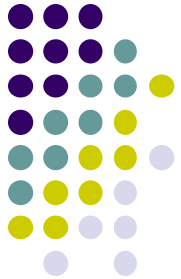
| m | o | r | a | d | i | a |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

```
String palavra = "moradia";
int posicao = palavra.indexOf('r');
System.out.println("O caractere está na posição: " + posicao);
```

  - Resultado

```
Saída - Exemplos String (run)
run:
O caractere está na posição: 2
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

# Classe String
# Método indexOf



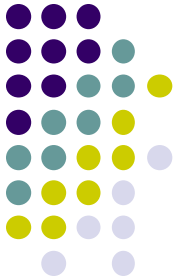
- Se o caractere buscado não existir na String, o valor -1 é retornado

| m | o | r | a | d | i | a |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

```
String palavra = "moradia";
int posicao = palavra.indexOf('x');
System.out.println("O caractere está na posição: " + posicao);
```

```
Saída - Exemplos String (run)
run:
O caractere está na posição: -1
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

# Classe String
# Método indexOf



- É possível utilizar o método indexOf para procurar não apenas caracteres, mas uma string também.

| m | o | r | a | d | i | a |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

```
String palavra = "moradia";
int posicao = palavra.indexOf("dia");
System.out.println("A string está na posição: " + posicao);
```

```
Saída - ExemplosString (run)
run:
A string está na posição: 4
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

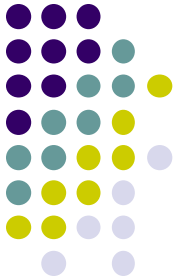
# Classe String
# Método length

- Retorna o comprimento(tamanho) da String. Não necessita de nenhum argumento.

```
public static void main(String[] args) {
    String palavra = "moradia";
    int tamanho;
    tamanho = palavra.length();
    System.out.println("A quantidade de caracteres da palavra é: " + tamanho);
}
```

```
Saída - Exemplos String (run)
run:
A quantidade de caracteres da palavra é: 7
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

# Classe String
# Método charAt

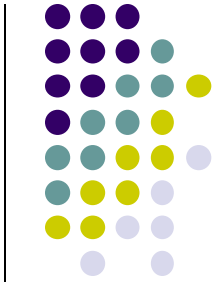


- Retorna o caracter de uma posição específica da palavra.

```
14  public static void main(String[] args) {
15      String palavra = "moradia";
16      char caractere = palavra.charAt(4);
17      System.out.println("O caractere na posição 4 é: " + caractere);
18
```

```
Saída - ExemplosString (run)                                   Tarefas
run:
O caractere na posição 4 é: d
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

# Classe String



- public **String** toLowerCase()
  - Retorna a String em caixa baixa

```
String nome = "Teste";
String nomeCaixaBaixa = nome.toLowerCase();
```

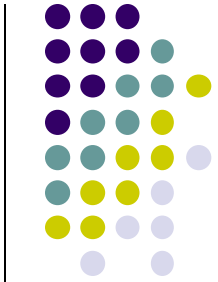
- public **String** toUpperCase()
  - Retorna a String em caixa alta

```
String nome = "Teste";
String nomeCaixaAlta = nome.toUpperCase();
```

- public **String** replace(char old, char new)
  - Retorna uma String com os caracteres substituídos

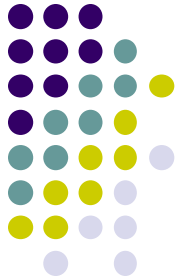
```
String nome = "AbAcAxi";
System.out.println(nome.replace('A', 'a'));
```

# Classe String



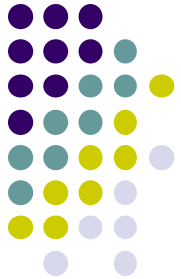
- Outros métodos
  - **toLowerCase()**
    - Retorna nova String toda minúscula. Não necessita de argumento
  - **toUpperCase()**
    - Retorna nova String toda maiúscula. Não necessita de argumento.
  - **compareTo(String s)**
    - Compara duas strings lexográficamente (em ordem alfabética). Retorna: 0 se as strings forem iguais; valor maior do que 0 se a string for maior; valor menor que 0 a string for menor.
  - **compareToIgnoreCase(String s)**
    - Compara duas strings alfabeticamente ignorando maiúsculas e minúsculas.

# Classe String



- Outros métodos
  - **replace(char** caractere_antigo, **char** novo_caractere)
    - Retorna uma nova string substituindo todas as ocorrências do caractere_antigo pelo caractere_novo
  - **substring(int** inicio, **int** fim)
    - Retorna uma nova string que é parte da string original, delimitada pelos índices passados como parâmetro.

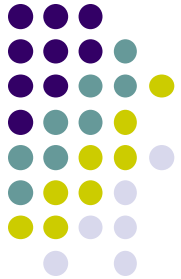
# Classe String
# Método de Concatenação



- Método para **concatenar**
  - String podem ser concatenadas (juntar uma com a outra) através do método **.concat(String s)**

```
public static void main(String[] args) {
        String silaba1 = "ca";
        String silaba2 = "sa";
        String palavra = silaba1.concat(silaba2);
        System.out.println(palavra);

}
```

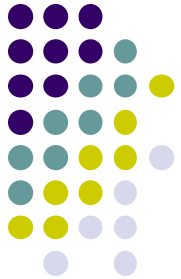
# Classe String
# Método de Concatenação



- Método para **concatenar**
  - O operador + pode ser utilizado ao invés do método **concat** para facilitar a construção dos programas.

```
public static void main(String[] args) {
        String silaba1 = "ca";
        String silaba2 = "sa";
        String palavra = silaba1 + silaba2;
        System.out.println(palavra);

}
```

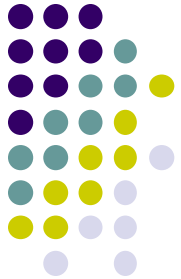
# PRATIQUE: Classe String



- Escreva um programa que leia uma String e imprima na tela a String invertida.

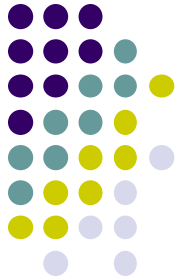
- Escreva um programa que leia uma String e imprima na tela a String com as consoantes em caixa baixa e as vogais em caixa alta.

# PRATIQUE: Classe String



1. Crie um programa em Java que leia o login e a senha de um usuário. Caso o login seja igual a "ifrn" e a senha "escola", deverá ser exibida a mensagem "Usuário autenticado no sistema". Caso contrário, deverá ser exibida a mensagem "Falha na autenticação". Ignore maiúsculas e minúsculas
2. Crie um programa em Java que leia uma frase e substitua todas os letras "a" por "b".
3. Crie um programa em Java que leia uma frase e remova todos os caracteres de espaços da frase. Ex: a frase "O livro está em cima da mesa" deverá ficar como: "Olivroestáemcimadamesa"
4. Criar um algoritmo que, dado o nome de uma pessoa (Nome + Sobrenome), escreva apenas o sobrenome. Ex: "Alba Lopes" , escreve somente "Lopes"
5. Crie um programa em Java que leia duas palavras do usuário e, em seguida, e escreva-as em ordem alfabéticas.
6. Crie um programa em Java que simule a criação de um cadastro. Deverá ser informado o nome e o e-mail. O sistema só deve permitir realizar cadastro caso o e-mail digitado seja válido. Um e-mail é considerado válido caso possua os caracteres arroba (@) e ponto (.). Além disso, o nome anterior ao @ deve possuir no mínimo 2 caracteres.

# Bibliotecas Java



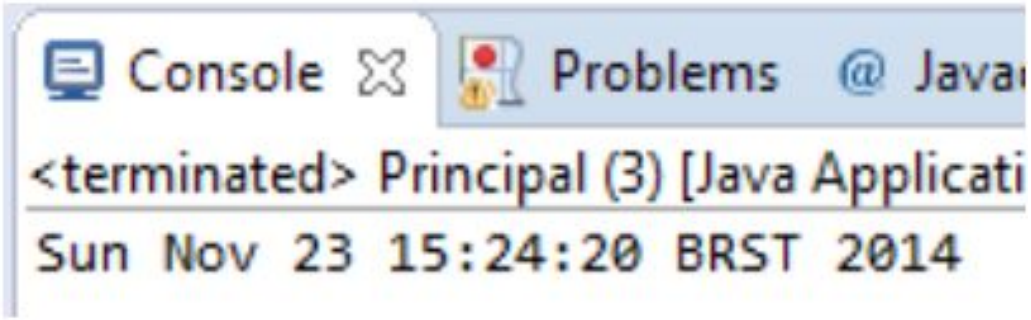
## Classe DATE

# Classe Date

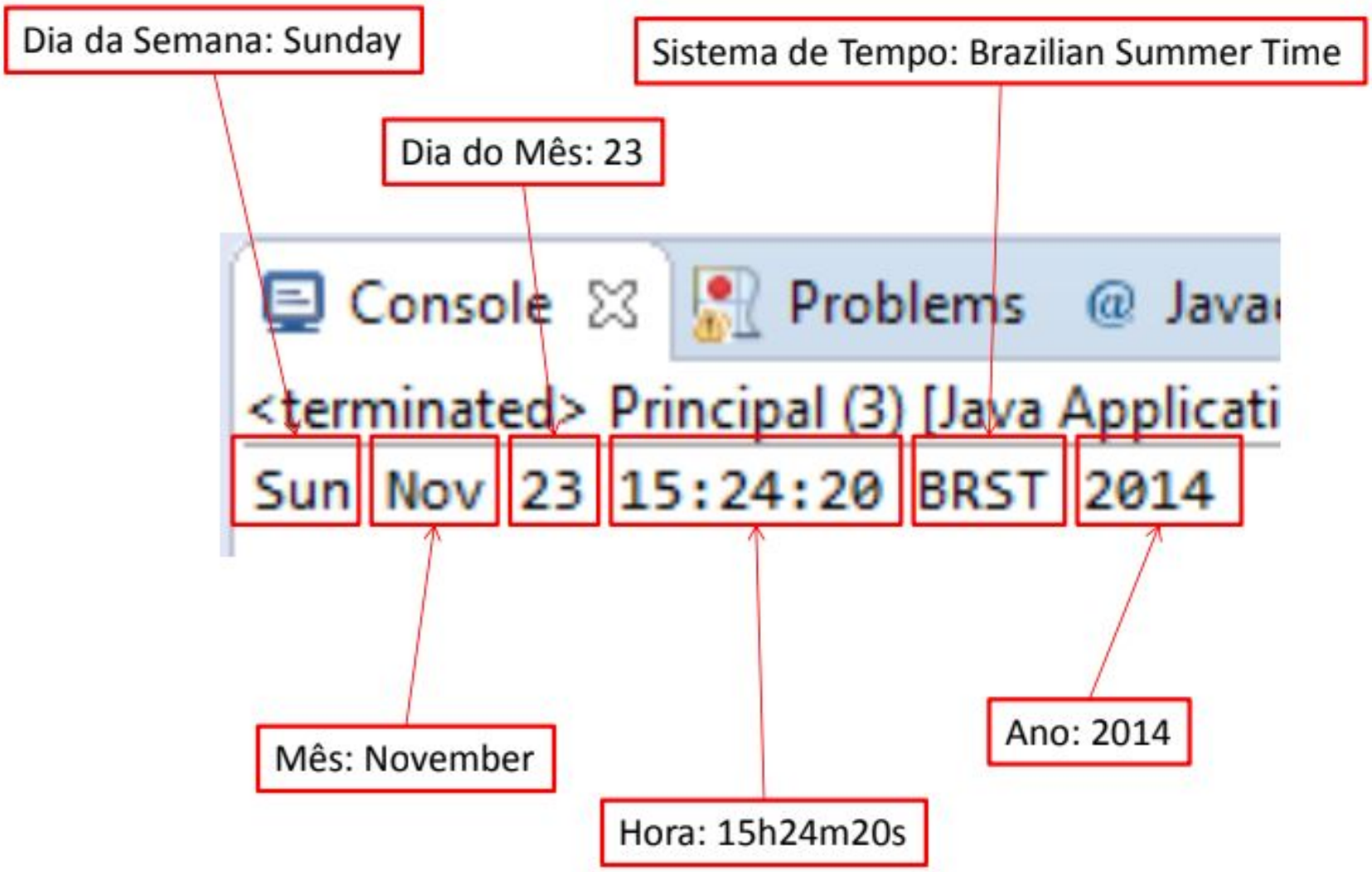
- Em Java, as Datas também são representadas por objetos **Date** do pacote java.util
- Como criar uma data que representa o instante atual?

```
Date date = new Date();
System.out.println(date.toString());
```

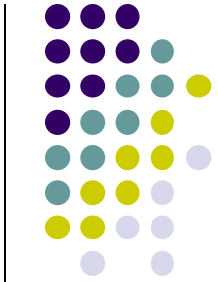
- Qual será a saída?
  - O construtor default cria uma data que representa o instante atual



# Classe Date

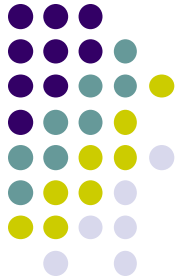

Dia da Semana: Sunday
Sistema de Tempo: Brazilian Summer Time
Dia do Mês: 23
Console
Problems
@ Java
<terminated> Principal (3) [Java Applicati
Sun
Nov
23
15:24:20
BRST
2014
Mês: November
Hora: 15h24m20s
Ano: 2014

# Classe Date



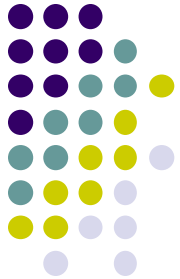
- A classe Date é muito limitada, ela serve para representar uma data como um objeto
- Ela possui poucos métodos que são verdadeiramente úteis, por exemplo:
  - after
  - before
  - compareTo
- Se eu quiser manipular a data ou exibir ela formatada, será mais fácil utilizar a “ajuda” de outros objetos

# Classe Date: Métodos



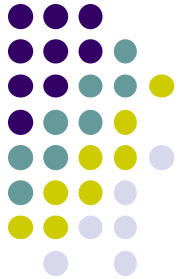
- public **boolean** after(Date outraData)
  - Retorna true se a data que chamou o método é posterior a data passada como parâmetro
- public **boolean** before(Date outraData)
  - Retorna true se a data que chamou o método é anterior a data passada como parâmetro
- public **int** compareTo(Date outraData)
  - Retorna o valor 0 se os duas datas são iguais, -1 se a data for anterior que a outraData e +1 se a data for posterior a outraData

# Bibliotecas Java



# Classe
# SIMPLEDATEFORMAT

# Classe SimpleDateFormat
## (pacote java.text)



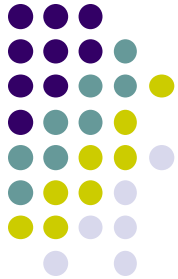
- Se eu quero exibir a data com uma certa formatação eu preciso usar a classe SimpleDateFormat
- Principais métodos que vocês usarão
  - public **String** format(Date data)
  - public **Date** parse(String data)

# Classe SimpleDateFormat

Máscara que será usada para formatar a String

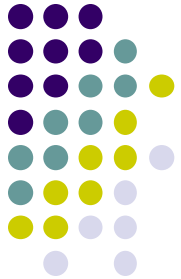
```
Date date = new Date();
SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
System.out.println(df.format(date));
```

# Classe SimpleDateFormat



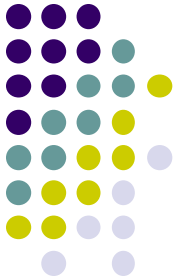
| Símbolo | Significado | Símbolo | Siginificado |
|---|---|---|---|
| d | Dia do mês | D | Dia do ano |
| E | Dia da semana | a | Marcador de am/pm |
| W | Semana do ano | W | Semana do mês |
| m | Minuto | M | Mês |
| y | Ano | G | Designador de era (AC/DC) |
| H | Hora em am/pm (1 a 12) | H | Hora do dia (0 a 23) |
| k | Hora do dia (1 a 24) | K | Hora em am/pm (0 a 11) |
| s | Segundos | S | Milissegundos |
| z | Sistema de Tempo (general time zone) | Z | Sistema de Tempo (RFC 822 time zone) |

# PRATIQUE:
# Classe SimpleDateFormat



- Qual a máscara que deve ser utilizada para exibir a data nesse formato?
  - 1/1/99 17:00
  - 03-7-2014 - 8:8 pm
  - 31/Dez/77, 19:40
  - 0:59:00
  - Terça-feira, 22 de janeiro de 14
- Teste cada uma das máscaras para uma data que representa o instante atual!

# Método *parse*: Transforma String em Date



Transformando a String "30/01/1989" em um objeto Date que representa a data 30/01/1989

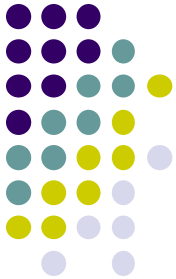
```
SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
try {
    Date date = df.parse("30/01/1989");
    System.out.println(date);
} catch (ParseException e) {
    e.printStackTrace();
}
```

Console  Problems  @ Javadoc

<terminated> Principal (3) [Java Application] (

Mon Jan 30 00:00:00 BRT 1989

# Tratamento de Exceções: try...catch



O método parse pode resultar em um erro! Quando?

```
SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
try {
    Date date = df.parse("30/01/1989");
    System.out.println(date);
} catch (ParseException e) {
    e.printStackTrace();
}
```

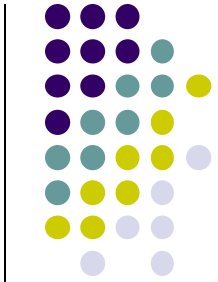
Quando esse erro acontece, o método lança uma exceção chamada ParseException!

Console | Problems | @ Javadoc

<terminated> Principal (3) [Java Application] (

Mon Jan 30 00:00:00 BRT 1989

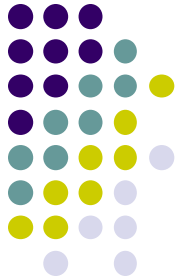
# Tratamento de Exceções: try...catch



- O código tem que estar preparado para tratar esse erro, por isso precisamos do “try catch”
- Algumas das classes que vocês vão utilizar lançam exceções que devem ser tratadas de forma semelhante!

```
try {
    //fazer isso
} catch (Exception e) {
    //se der errado venha para cá
}
```

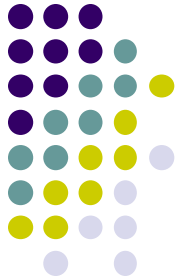
# Métodos Depreciados



Atualmente a maioria dos métodos da classe **Date** estão classificados como deprecated (depreciados), ou seja, são métodos que não são mais utilizados, por isso essa classe foi substituída pela classe Calendar, para haver suporte correto à internacionalização do sistema de datas.

Um método **deprecated** ainda está disponível, mas não é recomendado utilizá-lo por diversas razões: performance, flexibilidade, projeto, etc.

# Métodos depreciados



Métodos deprecated aparecerão com esse traço no código

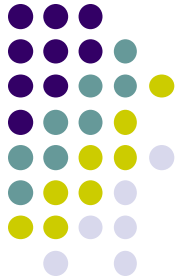
```
public static void main(String[] args) {

    int dia = 30;
    int mes = 0;
    int ano = 1989 - 1900;
    Date date = new Date (ano, mes, dia);
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MMMM/yyyy EEEE");
    System.out.println(sdf.format(date));
}
```

Console | Problems | Javadoc | Declara

```
<terminated> Principal (4) [Java Application] C:\Program
30/Janeiro/1989 Segunda-feira
```

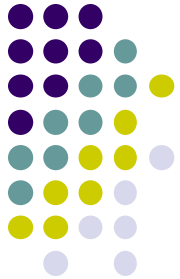
Os meses de Janeiro a Dezembro são numerados de 0 a 11.

# Bibliotecas Java



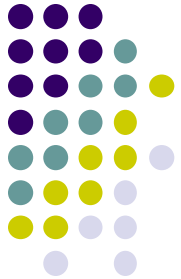
# Classe CALENDAR

# Classe Calendar
## (pacote java.util)



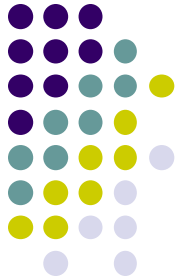
- A classe Calendar foi criada para substituir e/ou complementar a classe Date
  - A classe Date permite representar um instante no tempo como um objeto
  - A classe Calendar permite, além disso, fazer diversas operações com datas
    - Adicionar uma unidade de tempo a uma data
    - Reduzir uma data em alguma unidade de tempo

# Classe Calendar
## (pacote java.util)



- Como utilizar a classe Calendar?
  - A classe Calendar não pode ser instanciada utilizando o operador new e seu construtor
    - Isso não é possível Calendar c = new Calendar();
  - Nem todos os seus métodos são estáticos, logo não posso chamá-los com o nome da classe
    - Isso não é possível Calendar.set(1989, 0, 30)
  - Como obter um objeto do tipo Calendar?

# Classe Calendar
## (pacote java.util)



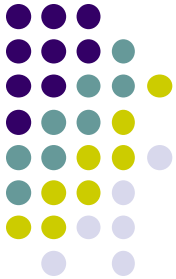
Calendar é uma **classe abstrata,** não pode ser instanciada. Logo para obter um objeto calendário é necessário usar o método estático getInstance().

- Calendar.getInstance() retorna um objeto do tipo Calendar

```
Calendar calendar = Calendar.getInstance();
```

- Esse método retornará um objeto calendar que representará o instante atual

# Classe Calendar
## (pacote java.util)



Obtendo uma instância de Calendar

```
public static void main(String[] args) {

    Calendar calendar = Calendar.getInstance();
    calendar.set(1989, 0, 30);
    Date date = calendar.getTime();
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MMMM/yyyy EEEE");
    System.out.println(sdf.format(date));

}
```

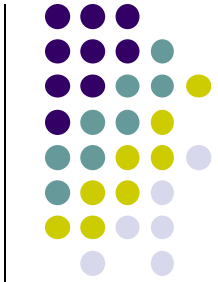
Alterando a data padrão para uma data específica

Criando o objeto Date utilizando o objeto Calendar. Esse objeto representa a data que configurei na linha anterior.

Console | Problems | Javadoc | Declaration

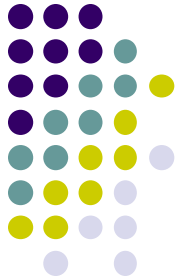
```
<terminated> Principal (4) [Java Application] C:\Program Files (x86)
30/Janeiro/1989 Segunda-feira
```

# Classe Calendar - Método add (pacote java.util)



```
public static void main(String[] args) {

    Calendar calendar = Calendar.getInstance();
    calendar.set(1989, 0, 30);

    Date date = calendar.getTime();
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MMMM/yyyy EEEE");

    System.out.println(sdf.format(date));//30/Janeiro/1989 Segunda-feira

    calendar.add(Calendar.DATE, +2);
    date = calendar.getTime();
    System.out.println(sdf.format(date));//01/Fevereiro/1989 Quarta-feira

    calendar.add(Calendar.MONTH, +1);
    date = calendar.getTime();
    System.out.println(sdf.format(date));//01/Março/1989 Quarta-feira

    calendar.add(Calendar.YEAR, -1);
    date = calendar.getTime();
    System.out.println(sdf.format(date));////01/Março/1988 Terça-feira

}
```

# Classe Calendar
## (pacote java.util)

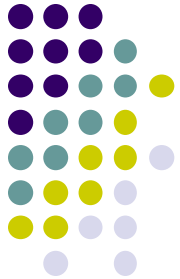


```java
import java.util.Calendar;

public class Testa_Metodo_Get_Calendar{

    public static void main(String[] args) {
        Calendar c = Calendar.getInstance();

        System.out.println("Data/Hora atual: "+c.getTime());
        System.out.println("Ano: "+c.get(Calendar.YEAR));
        System.out.println("Mês: "+c.get(Calendar.MONTH));
        System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));
    }
}
```

Mostra o dia da semana, mês e ano

https://www.devmedia.com.br/trabalhando-com-as-classes-date-calendar-e-simpledateformat-em-java/27401

# Classe Calendar
## (pacote java.util)

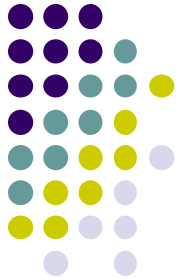


```java
import java.util.Calendar;

public class Testa_Metodos_Set_Get_Calendar{

    public static void main(String[] args) {
        Calendar c = Calendar.getInstance();
        c.set(Calendar.YEAR, 1995);
        c.set(Calendar.MONTH, Calendar.MARCH);
        c.set(Calendar.DAY_OF_MONTH, 20);

        System.out.println("Data/Hora atual: "+c.getTime());
        System.out.println("Ano: "+c.get(Calendar.YEAR));
        System.out.println("Mês: "+c.get(Calendar.MONTH));
        System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));
    }

}
```

Alterando a data/hora com método set

https://www.devmedia.com.br/trabalhando-com-as-classes-date-calendar-e-simpledateformat-em-java/27401

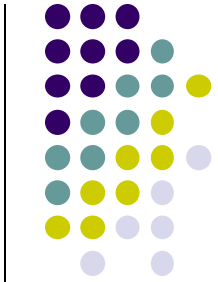
# Classe Calendar
## (pacote java.util)



**Material Complementar sobre a Classe Calendar - Leitura Obrigatória**

https://www.devmedia.com.br/trabalhando-com-as-classes-date-calendar-e-simpledateformat-em-java/27401

# Material de Consulta Geral



- https://www.educba.com/what-is-api-in-java/ - Em inglês (excelente material)