Programação OO I

Parte I

É importante termos uma linguagem para chamarmos de “minha”….

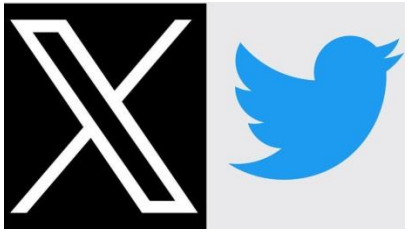
Por que Java?

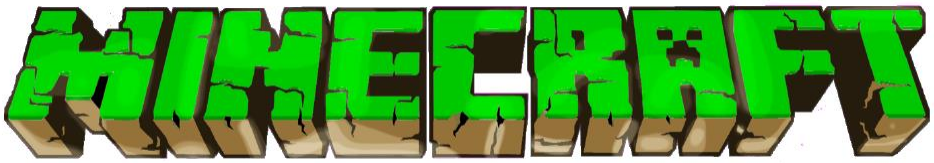
Write once, run everywhere!

Java abstrai alguns conceitos “baixo níveis”

Por que Java?

Grandes empresas....





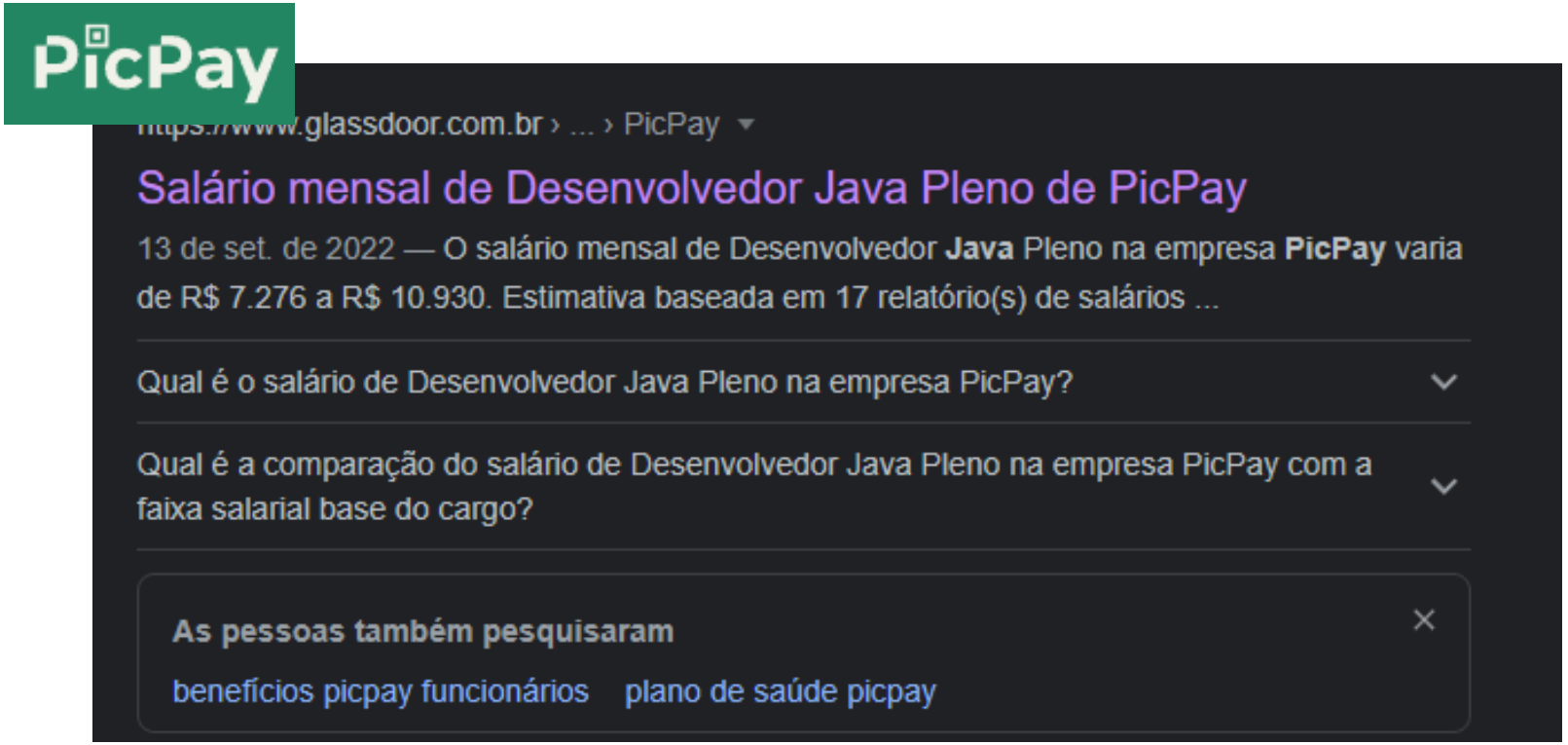
Por que Java?

Aonde é usado?...

- Web-Based Applications
- Building Android Applications
- Embedded Systems
- Big Data Technologies
- Cloud-based Applications
- Gaming Applications
- Software Tools
- Internet Of Things

Por que Java?

Oportunidades...



Por que Java?

Top 50+ Java companies

Find a Java agency to get even the most complex work done fast.

Hire Agencies See how it works

DigiMantra Labs

Orlando United States

4.8/5 (533 jobs)

TOP RATED

Member since Jul 14, 2009

Sign up to contact

Post a job & invite

DigiMantra Labs is one of the word's leading technology solutions providers that engineers modern businesses. We help companies to scale by modernising technology, reimagining processes and transforming experiences so that they can stay

Read more

Hourly rate
$18.00 - $75.00

Minimum project size
$1,000+

Xicom Technologies Ltd.

New Delhi India

4.9/5 (836 jobs)

Member since Jan 31, 2002

Sign up to contact

Post a job & invite

Xicom is an ISO:9001 certified software company with 18+ years in operation and having offices in India, Dubai and USA. We have a strong team of 300+ highly skilled IT/Mobile experts with a strong portfolio of delivering over 1500+ websites and

Read more

Hourly rate
$25.00 - $50.00

Minimum project size
$10,000+

Por que Java?


catho
Explore o site de vagas de emprego mais completo do Brasil!
Faça sua pesquisa e veja as melhores opções de vagas do site de empregos que é o preferido dos recrutadores.
Desenvolvedor Java Pleno
Empresa Confidencial Por que?
De R$ 9.001,00 a R$ 10.000,00
1 vaga: Rio de Janeiro - RJ (1)
Publicada ontem
Operadora de saúde há 50 anos no mercado, está em busca de profissionais especializados em Java, para compor o time.- Conhecimento em Java - Graduação na área - Disponibilidade de... continuar lendo
Quero me candidatar


LinkedIn
https://br.linkedin.com › jobs › java-vagas
29.000 vagas de Java em: Brasil (4.696 nova(s)) - LinkedIn
29.000 principais vagas de Java hoje em Brasil. Aproveite sua rede profissional e seja contratado. Novas vagas de Java adicionadas diariamente.

Por que Java?

Porque vários concursos pedem Java!

TRT-13R-Tec-Jud-Programacao-CE

MODELO - Caderno de Prova, Cargo I, Tipo 001

ESTUDO DE CASO

1. Identificar os três tipos distintos de laços, ou estruturas de repetição da linguagem Java. Para cada tipo de laço, descrever a função específica da estrutura e a sintaxe.

Por que Java?

The *TIOBE* Programming Community *index* is an indicator of the popularity of programming languages.

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	20	20	-	-
C++	4	3	3	3	2	1	2	9
C#								
JavaScr								
PHP								
SQL								
Swift								
R								

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2022	2017	2012	2007	2002
Python	1	5	8	7	12
C	2	2	2	2	2
Java	3	1	1	1	1
C++	4	3	3	3	3
C#	5	4	4	8	15

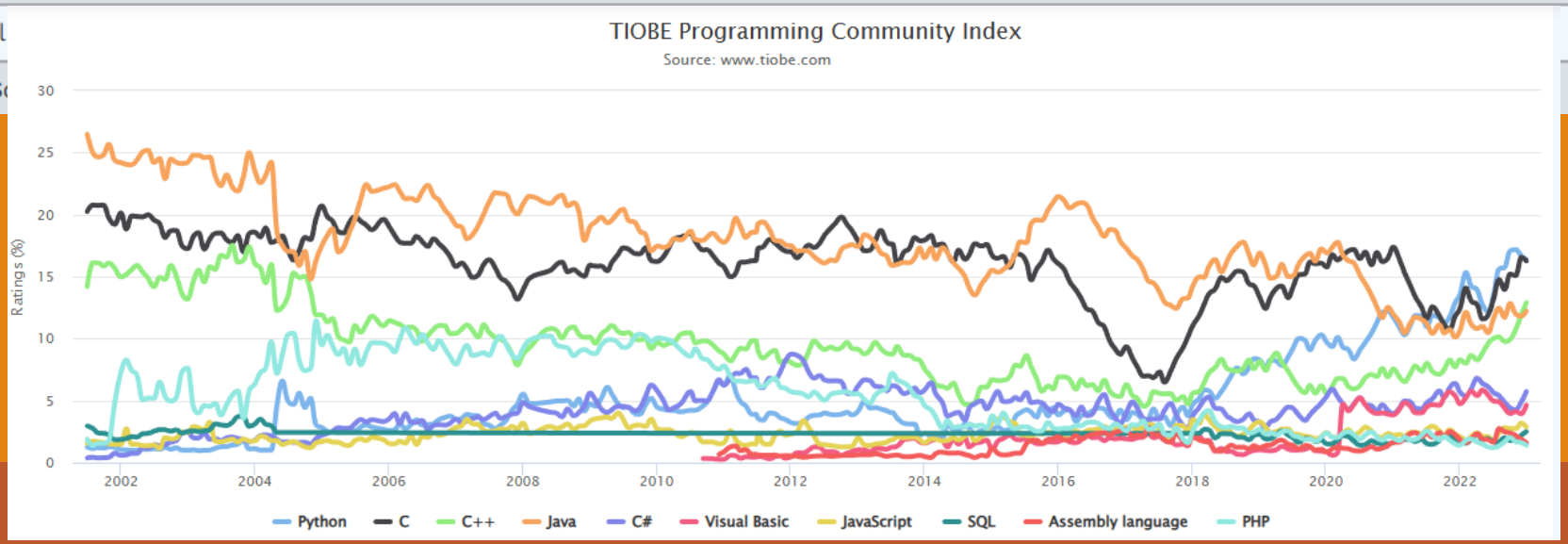
Por que Java?

The *TIOBE* Programming Community *index* is an indicator of the popularity of programming languages.

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2023	2018	2013	2008	2003	1998	1993	1988
Python	1	5	8	7	13	28	17	-
C	2	2	1	2	2	1	1	1
Java	3	1	2	1	1	17	-	-
C++	4	3	4	3	3	2	2	6
C#	5	4	5	8	12	-	-	-
Visual								-
JavaSc								-



Um pouco de História

A Oracle comprou a Sun Microsystems e com ela a plataforma Java.

Java - Hoje

Java faz 28 anos em 2023.

A plataforma Java evoluiu bastante.

- Versões atuais do JDK é o Java 17 e 19 -> 20 (Mar/23)

> There are many useful features introduced from Java 8 to Java 17 like lambda expressions, Stream API, New Date, and Time API, Creating Immutable Lists, Records, Sealed Classes, var for storing local variables without types, String in switch case, Text Block, and many more.

Java - Hoje

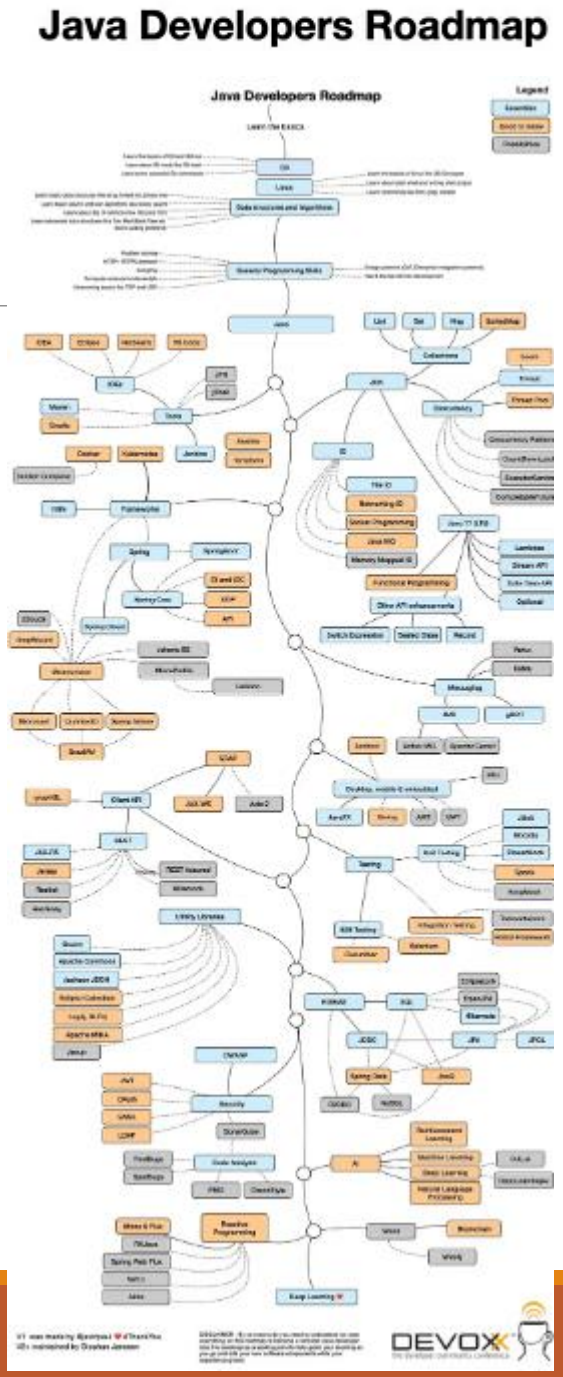
Há diversas ferramentas visuais (IDEs) para desenvolvimento em Java.

- Eclipse
- NetBeans
- IntelliJ
- Visual Code
- Etc...

Java - Hoje

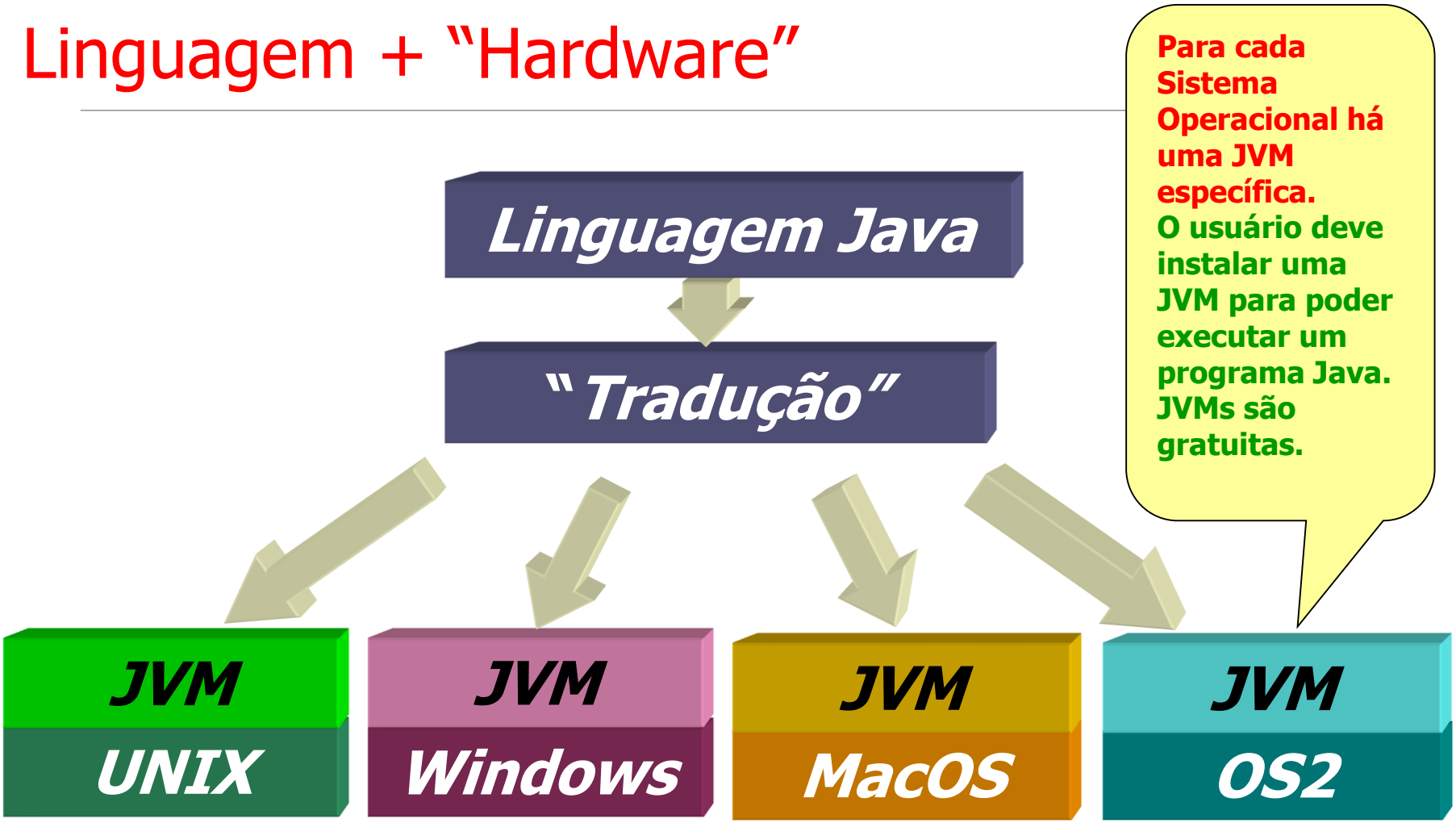
Recursos para aprendizado da plataforma Java

- Comunidades virtuais
- Cursos Online
- Livros
- etc..



A Plataforma Java

- Linguagem + "Hardware"



- Surge dois conceitos importantes...

Java: Visão Geral

Conceitos importantes da linguagem

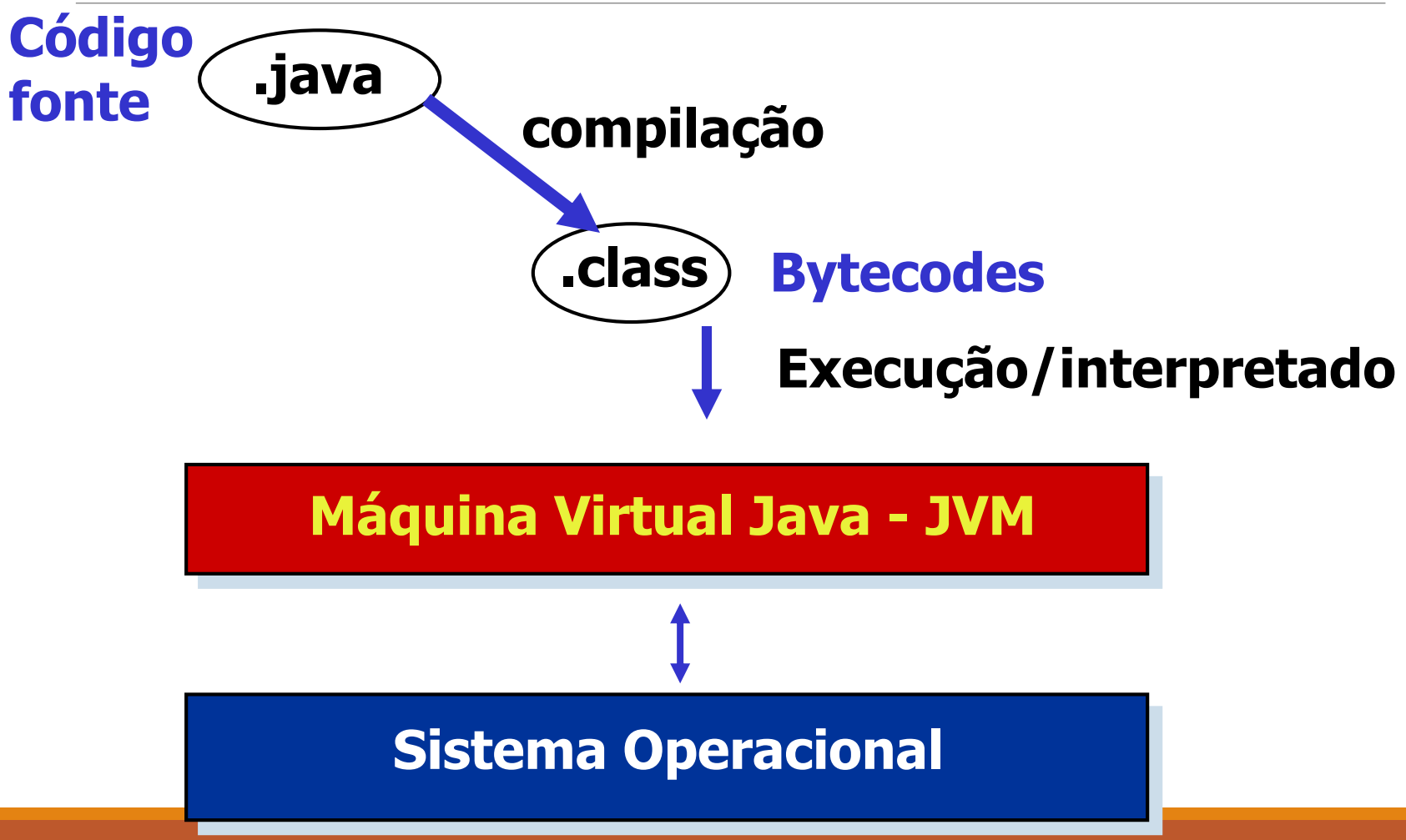
- *Bytecode*
  - É o código de máquina da Máquina Virtual Java
    - *Bytecode* é interpretado: Código passado para máquina *on-the-fly*
  - Formato *bytecodes* foi feito pensando na geração de código de máquina, de maneira otimizado
    - Lembra das opções de otimização de código em C???
  - Desempenho do código final (*bytecodes* transformados em binário) é praticamente o mesmo de código feito em C/C++.
    - Uso da *HotSpot*, nova tecnologia de JVM: aos poucos um *bytecode* Java vai virando código em linguagem de máquina
    - Em última instância, integra-se com códigos em C/C++.

Java: Visão Geral

Conceitos importantes da linguagem

- *Java Virtual Machine* (JVM)
  - É o programa que executa/interpreta os *bytecodes* Java.
  - Elemento que permite a portabilidade
    - Esconde as diferenças entre os diversos sistemas operacionais que executam código Java.
  - Um mesmo código compilado (*bytecode*) pode ser executado em diferentes JVMs sem qualquer modificação.
    - "Arquiteturalmente Neutra"
  - Para cada Sistema Operacional há uma JVM específica.
    - O usuário deve instalar uma JVM para poder executar um programa Java. JVMs são gratuitas.

Diagrama de criação e execução de código Java



Linguagem Java

Blz,

- Entendido a estrutura da plataforma Java…

- Podemos começar agora a conversar sobre características da linguagem Java

Questões Léxicas

- Durante a compilação, os caracteres no código Java são reduzidos a uma série de *tokens*.
- O compilador Java reconhece cinco tipos de *tokens* :

  Identificadores

  Palavras Reservadas

  Literais

  Separadores

  Operadores

  Comentários

- Comentários e espaços em branco, como tabulação, são usados para separar os *tokens*.

Vamos falar um pouco sobre cada uma destas questões....

É importante para entende uma nova linguagem!!!

Questões Léxicas

Identificadores

- Identificadores são nomes usados para nomear variáveis, atributos, métodos, classes
- As regras para declarar identificadores são:
  - Maiúsculas e minúsculas são diferentes
    - Java é sensível ao caso
      - int A é diferente de int a
  - Palavras reservadas não podem ser usadas como identificadores
  - Pode conter letras, números, o caractere $ (cifrão) ou _ (*underscore*)
  - Não pode começar com número
  - É possível utilizar acentuação nas variáveis porque Java usa Unicode como código de caracteres

Questões Léxicas

Palavras Reservadas

- É importante sabermos quais são as palavras reservadas da linguagem Java,
  - Primeiro pra não ocorrer o erro de tentar criar variáveis com estes nomes,
  - Segundo porque a certificação pede esse conhecimento...

–Tabela: Tipos, modificadores e mecanismo de controle de fluxo.

abstract boolean break byte case

catch char class const continue

default do double else extends

false final finallly float for

goto if implements import instanceof

int interface long native new

null package private protected public

return short static super synchronized

this throw throws transient true

try void volatile while

Questões Léxicas

Declaração de Variáveis

- Formato
  - Tipo seguido por lista de identificadores
    - *tipo identificador [= valor] [, identificador [= valor ...];*
      - Ex.: int b, c;

- Inicialização
  - Pode ser feita na declaração;
    - Ex.: int a=10;
  - É interessante inicializar a variável local antes de usar o valor, a não ser que seja utilizado o valor padrão do tipo.
  - Variáveis de classe e instância são inicializadas automaticamente
  - Pode ser feita em qualquer ponto do programa

Tipos em Java

Em Java temos os tipos primitivos e compostos:

- Os tipos primitivos em Java são:
  - boolean, char
  - byte, short, int, long
  - float, double

- Os tipos compostos em Java são:
  - Objetos .... Daqui a pouco a gente fala sobre eles.

Tipos Primitivos

Tipos Inteiros

Tipo	Tamanho	Alcance
byte	1 byte	-128 a 127
short	2 bytes	-32.768 a 32.767
int	4 bytes	-2.147.483.648 a 2.147.483.647
long	8 bytes	-9223372036854775808 a 9223372036854775807

- Todos tipos possuem sinal;
- Cuidado com valores atribuídos
  - Pode ocorrer sobreposição, ou seja, alcançar o limite superior e recomeçar no inferior (*overflow*)

Tipos Primitivos

Tipos Ponto Flutuante

Tipo	Tamanho	Alcance
float	4 bytes	aprox. ±3.402823 E+38F
double	8 bytes	aprox. ±1.7976931348623 1 E+308

- Seguem padrão IEEE754
- Notação Científica

  `1.44E6` (= 1.44 x 10^6 = 1,440,000) ou

  `3.4254e-2` (= 3.4254 x 10^{-2} = 0.034254)

- Precisão: A precisão se refere ao número de casas decimais, não ao tamanho
  - float: até 6 casas decimais (depois da ,)
  - double: até 12 casas decimais (depois da ,)

Tipos Primitivos

Tipo Caractere

- Representação de Caracteres Individuais
- Tamanho de 16 bits

- Tabela Unicode
  - Código numérico sem sinal (até 32.768 caracteres)
  - Internacionalização
  - Compatível com a tabela ASCII
  - Valor literal de um caractere
    - limitado por aspas simples: ‘a’
  - Caracteres especiais e sem representação visual
    - precedidos por barra invertida: ‘\n’, ‘\’’
  - Uso de escapes
    - \u0061 = ‘a’
  - Uso como identificador (só escape \uxxxx)
    - int b\u0061 = 10; // int ba = 10;

Tipos Primitivos

Tipo Booleano

- Valores
  - *true*
  - *false*
- Condições devem ser tipos booleanos
- Não há equivalência com inteiros
  - Em C, 0 é false e qualquer outro valor é true.
  - Declaração: *boolean bool = true;*

```
boolean b = true;
if (b) System.out.println("OK!"); // OK!

int i = (int)b; // Erro de compilação!
i = 1;
if (i) System.out.println("??"); // Id
```

Erro compilação!

Tipos Primitivos

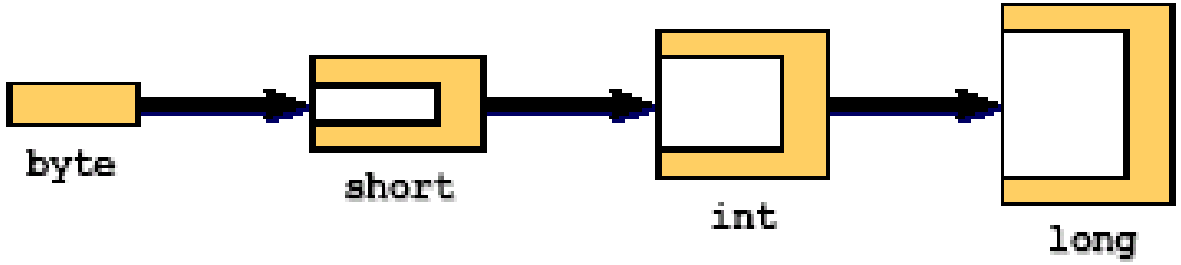
Importante: Conversões entre Tipos Numéricos

- Pode misturar tipos
  - Se algum dos operandos for do tipo *double*, então o outro operando será convertido em um *double*
  - Caso contrário, se algum dos operandos for do tipo *float*, o outro operando será convertido em um *float*
  - Caso contrário, se algum dos operandos for do tipo *long*, o outro operando será convertido em um *long*
- Forma análoga para os tipos inteiros: *int*, *short* e *byte*
- Conversões onde pode haver perda de informação devem ser feitas explicitamente através do operador de coerção (*cast)*
  - Ocorre, por exemplo, quando são feitas operações com tipos inteiros (*byte*, *short* , *char*) menores que *int*
    - Resultados das operações são do tipo *int*
      - Racional: maior probabilidade de ocorrência de *overflow* nestes tipos, uma vez que o intervalo de valores é pequeno

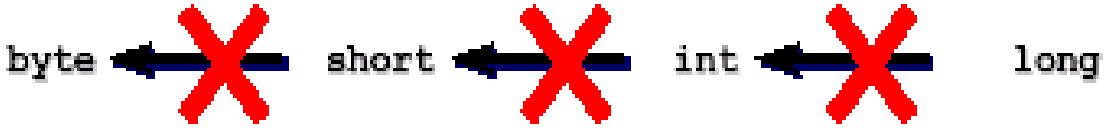
Tipos Primitivos

Resumindo.....

- O Java converte automaticamente valores de um tipo numérico para outro tipo maior



- O Java não faz automaticamente o "*downcast*."



Tipos Primitivos

Conversões entre Tipos Numéricos

- Necessário para atribuir um tipo maior a um menor

(<tipo>) <expressão>

- Converte a expressão para o tipo indicado entre parênteses

```
int a = 1234;
long b = a;

int c = (int) b;
short c = (short) a;
```

conversão implícita

conversão explícita (cast)

Promoções em expressões aritméticas

Conversões entre Tipos Numéricos – Outro ex.

- Tipos de menor precisão são automaticamente convertidos para tipos de maior precisão (promoção aritmética)

```
int + short*byte - double
int + short*short - double
int + short - double
int + int - double
int - double
double
```

Entrada e saída de dados básica

Toda linguagem de programação deve prover um meio de interação com o usuário;

- O meio mais básico é o uso do console, com entrada de dados pelo teclado e saída em texto;

Outros meios são: interface gráfica (janelas), pela *Web*, comandos de voz, etc.;

- Inicialmente aprenderemos agora a forma de interação básica, pelo console:
  - O “shell” do Linux;
  - O “prompt de comando” do Windows

Entrada e saída de dados básica

Saída de dados pelo console

- Java usa o conceito de *stream*: um duto capaz de transportar dados de um lugar a outro;
  - Vamos ver com mais detalhes mais pra frente no curso
- A classe java.lang.System oferece um stream padrão de saída chamado out;
  - É um objeto da classe java.io.PrintStream, aberto e mantido automaticamente pela JVM;
- Oferece vários métodos para impressão de dados:
  - print(), println() e printf().
- Podemos trocar o dispositivo padrão de saída:
  - System.setOut(novaStream).

Entrada e saída de dados básica

Saída de dados pelo console

- Exemplo

```
// 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
for (i = 1; i < 10; i++) {
   System.out.print(i + ", ");
}
System.out.println(10);

String s = "Olá, Java!";
float valor = 45.67;
boolean teste = true;

System.out.println(s);    // Olá, Java!
System.out.print(valor);  // 45.67 (sem quebra)
System.out.println();      // Quebra de linha
System.out.println(teste); // true
```

Entrada e saída de dados básica

Entrada de dados pelo console

- A classe java.lang.System oferece um *stream* padrão de entrada chamado in;
  - É um objeto da classe java.io.InputStream, aberto e mantido automaticamente pela JVM;

- Seus métodos de leitura são muito primitivos e não são simples de serem utilizados
  - Precisamos de outras classes que auxiliem na leitura.
    - Classe Scanner

Entrada e saída de dados básica

Entrada de dados pelo console

- Classe Scanner
  - A ideia é bastante simples....
    - Basta aplicar esse comando:
      - Scanner sc = new Scanner(System.in);
    - E depois chamar o método o nome do tipo de dados que se deseja capturar
      - int i = sc.nextInt();
      - double d = sc.nextDouble();
      - String s = sc.next();
    - Também é necessário escrever no início do código o seguinte comando:
      - import java.util.Scanner;

LP Java

Blz,

- O que falta agora pra gente fechar essa parte.....
  - Operadores
  - Estruturas de fluxos de controle

Operadores

Símbolos especiais que recebem um ou mais argumentos e produzem um resultado;

Temos 5 tipos de operadores:

- Atribuição
- Aritméticos
- Manipulação de bits
- Relacionais
- Booleanos

Operadores

Operadores Aritméticos

- Os operadores aritméticos mais utilizados em Java são:

Operator	Use	Description
+	op1 + op2	Adds op1 and op2
-	op1 - op2	Subtracts op2 from op1
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2

Operator	Use	Description
++	op++	Increments op by 1; evaluates to the value of op before it was incremented
++	++op	Increments op by 1; evaluates to the value of op after it was incremented
--	op--	Decrements op by 1; evaluates to the value of op before it was decremented
--	--op	Decrements op by 1; evaluates to the value of op after it was decremented

Operadores

Operadores Relacionais e Condicionais

- Os operadores mais utilizados em Java são:

Operator	Use	Returns true if
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

Operator	Use	Returns true if
&&	op1 && op2	op1 and op2 are both true, conditionally evaluates op2
||	op1 || op2	either op1 or op2 is true, conditionally evaluates op2
!	! op	op is false
&	op1 & op2	op1 and op2 are both true, always evaluates op1 and op2
|	op1 | op2	either op1 or op2 is true, always evaluates op1 and op2
^	op1 ^ op2	if op1 and op2 are different--that is if one or the other of the operands is true but not both

Operadores

Operadores de Atribuição

- Básico

```
int x = 5;          // inicialização
x = x + 2;          // atribuição
x = y = z = 7;              // encadeada
```

- Compostos

Expressão	Equivale a	Expressão	Equivale a
x += y	x = x + y	x &= y	x = x & y
x -= y	x = x - y	x |= y	x = x | y
x *= y	x = x * y	x ^= y	x = x ^ y
x /= y	x = x / y	x >>= y	x = x >> y
x %= y	x = x % y	x <<= y	x = x << y

Questões Léxicas

Comentários

// texto

Todos os caracteres de // para o final da linha são ignorados.

/ texto */*

Todos os caracteres de /* para */ são ignorados.

```
/** <i>Documentação da classe</i>.
 * @author Fulano da Silva
 * @see java.io.File
 */
public class FileData extends File {
  /** Documentação de atributo. */
  private double tamanho;

  /* Comentário
     de múltiplas linhas. */

  public void excluir() {
    int x = 1; // Comentário de uma linha.
  }
}
```

Questões Léxicas

Comentário de documentação

*/** texto*

* @param nome   nome do cliente

/

- É uma forma especial de comentário usado pela ferramenta *javadoc*.
  - Deve ser usado antes da classe pública, do método e das declarações de variáveis.
  - O *javadoc* reconhece diversas variáveis especiais, as quais são denotadas por @ (sinal de arroba) dentro desses comentários. (usado para identificar parâmetros, por ex.)

Questões Léxicas

Comentário de documentação

- Geração de documentação
  - O utilitário javadoc analisa arquivos fonte procurando por classes, métodos e comentários /**...*/.
  - Ele gera um arquivo HTML no mesmo formato que a documentação da API.

`javadoc FileData.java`

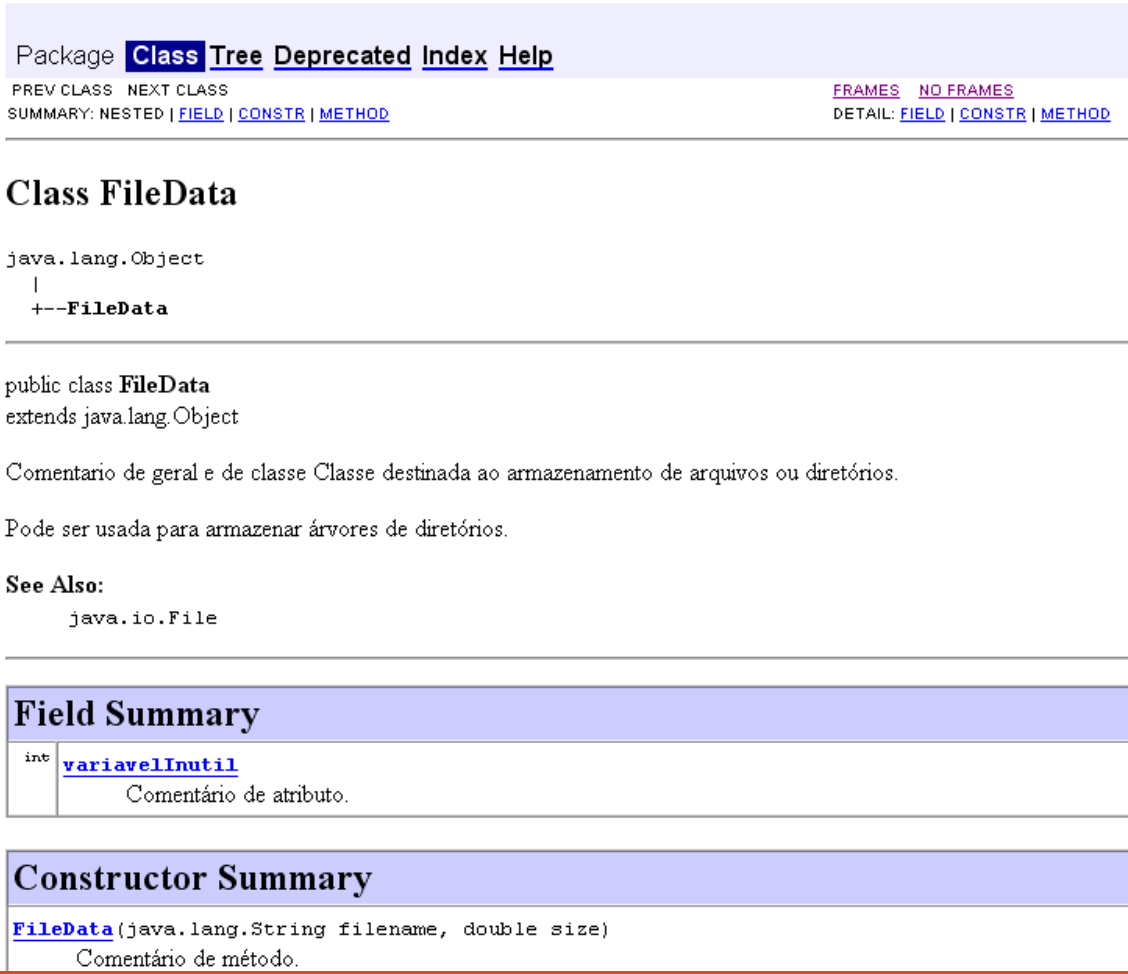
  - Espaços e * 's iniciais são descartados
  - Também é possível passar vários argumentos...

```java
/** Comentario de geral e de classe
 * Classe destinada ao armazenamento de arquivos ou diretórios.
 * <p> Pode ser usada para armazenar árvores de diretórios.
 * @author Vinicius Rosalen
 * @see java.io.File
 */

public class FileData {
    /** Comentário de atributo.
    * Essa variável e inútil
    */
     public int variavelInutil = 10; // inútil

    /** Comentário de método.
    * Construtor
    * @param filename nome do arquivo
    * @param size tamanho do arquivo
    */
    public FileData(String filename, double size) {
    }

    /** Compara se dois nomes de arquivos são iguais
    * @param filename1        um nome de arquivo
    * @param filename2        um nome de arquivo
    * @return         true se os dois nomes são os mesmos ou se os dois forem
    *                 nulos, caso contrário retornará falso.
    */
    public boolean equals(String filename1, String filename2) {
        return true;
    }

    /* comentário de múltiplas linhas
    não farão parte da documentação gerada pelo
    javadoc */
}
```

Questões Léxicas

Comentário de documentação

- Documentação gerada a partir do código anterior



Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS — FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

Class FileData

```
java.lang.Object
  |
  +--FileData
```

public class **FileData**
extends java.lang.Object

Comentario de geral e de classe Classe destinada ao armazenamento de arquivos ou diretórios.

Pode ser usada para armazenar árvores de diretórios.

See Also:

`java.io.File`

Field Summary

int	**variavelInutil** Comentário de atributo.

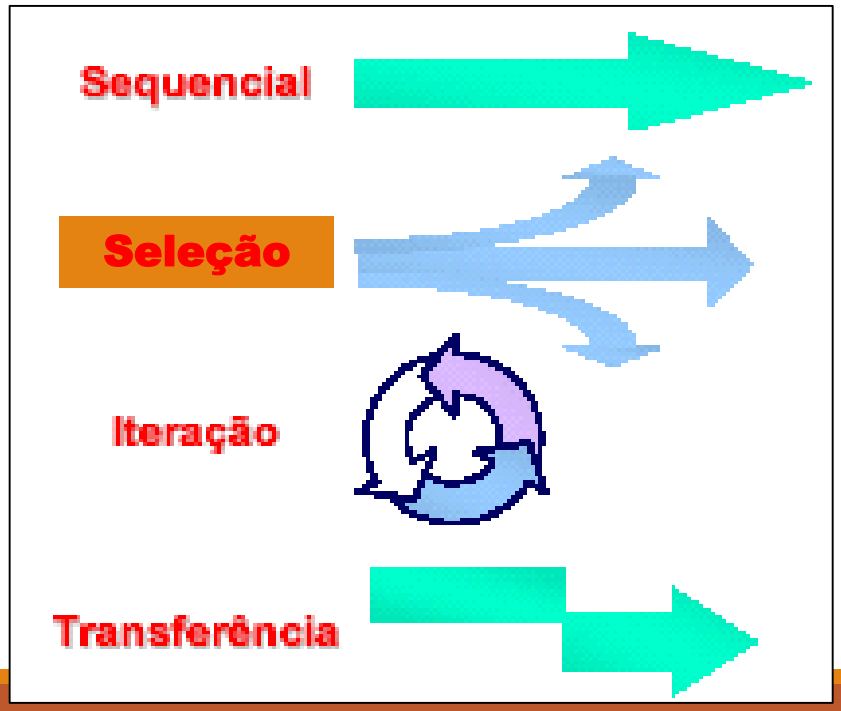
Constructor Summary

FileData(java.lang.String filename, double size)
Comentário de método.

Fluxo de Controle

Pra fechar só falta conversar agora sobre as estruturas de controle que Java oferece....

- Pra quem esqueceu... o controle de fluxo pode ser categorizado em quatro tipos:



Fluxo de Controle

Sintaxe para *Loops/Repetições*

```
while (boolean expression) {
    statement(s)
}
```

```
do {
    statement(s)
} while (expression);
```

```
for (initialization ; termination ; increment) {
    statement(s)
}
```

Fluxo de Controle

Sintaxe para Condicionais

```
if (boolean expression) {
    statement(s)
} else {
    statement(s)
}
```

```
switch (        expression) {
    case integer expression:
        statement(s)
        break;
    ...
    default:
        statement(s)
        break;
}
```

- uso de *break*
- *default* opcional

Botando na prática

Blz,

- Chega de conversa, vamos para prática…

Exercício Junto!!!

"brincar" com o debug tb

Vamos fazer o jogo do adivinha número inteiro positivo secreto…

- Vamos inclusive publicar no PSN….

Regras:

- O número secreto é 10 (mas não conta pra ninguém….)
- O jogador tem até 16 tentativas (inclusive) para acertar
- Existem dicas:
  - 3 <= tentativa <= 5
    - Dica 1: Chuta um numero entre 1 e 1000
  - 8 <= tentativa <= 10
    - Dica 2: Chuta um numero entre 1 e 100
  - tentativa >= 13
    - Dica 3: Chuta um numero entre 1 e 10
- No final diz se ganhou ou perdeu…

Exercício

Segundo a OMS (organização Mundial de Saúde), o IMC normal está entre 18.5 e 25. Faça um programa que calcule o IMC para imprimir na tela se para os valores configurados a pessoa está "Abaixo do Peso", "Normal" ou "Acima do Peso".

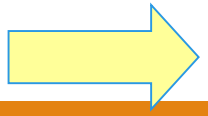
- A fórmula para calcular o Índice de Massa Corporal é: IMC = peso / (altura)^2
- Proíba a entrada de números negativos do usuário (evitar que a entrada seja um número negativo)

Exercícios

Blz... Agora é hora de exercitar.....

Tente resolver os seguintes problemas...

- Em dupla
- Apresentar ao professor no final da aula
- Pontuação em Atividades em sala de aula...



Exercícios

1. Crie um arquivo para a classe abaixo e o compile.
   - Explique o que ocorreu.
   - Faça a correção devida e execute-o.

```
1  class byteExplo {
2      public static void main ( String[] args ) {
3          byte a, b, c;
4          b = 10;
5          c = 15;
6          a = b + c;
7          System.out.println("Um byte: " +  a );
8      }
9  }
```

Exercícios

2. Escreva um programa que calcule a conversão de um temperatura em *Fahrenheit* para *Celsius* e vice versa
   - Dica: fah – 32 / 9 * 5;
   - O que acontece se definir o tipo *fah* ou *cel* como inteiro

3. Faça um algoritmo que receba o salário de um funcionário, calcule e imprima o valor do imposto de renda a ser pago, sabendo que o imposto equivale a 5% do salário.

4. Fazer um algoritmo que calcule o volume de uma esfera em função do raio R.

$$V = \frac{4}{3}\pi r^3$$

Exercícios

5. Faça um algoritmo que receba o valor do salário de um funcionário e o valor do salário mínimo. Calcule e imprima quantos salários mínimos ganha esse funcionário.

6. Faça um algoritmo que receba o salário de um funcionário, calcule e imprima o novo salário sabendo-se que este sofreu um aumento de 25%.

7. Faça um programa que calcula o Índice de Massa Corporal (IMC).
   - O índice de Massa Corporal (IMC) é uma fórmula que indica se um adulto está acima do peso, se está obeso ou abaixo do peso ideal considerado saudável. A fórmula para calcular o Índice de Massa Corporal é: IMC = peso / (altura)^2
   - Em Java, a classe que tem funções matemáticas tais como potência está na classe Math.

Exercícios

8. Faça um algoritmo que receba duas notas de um aluno e seus respectivos pesos, calcule e imprima a média ponderada dessas notas.

9. Determine qual é a idade que o usuário faz no ano atual. Para isso peça o ano de nascimento e o ano atual.

10. Solicite a quantidade de homens e de mulheres de uma turma qualquer da faculdade. Em seguida calcule e exiba o percentual (separadamente) de homens e mulheres desta turma.