

# ORIENTAÇÃO A OBJETOS

## AULA 1

### Planejamento e Introdução a Orientação a Objeto

Vandor Roberto Vilardi Rissoli



# APRESENTAÇÃO

- Síntese do Plano de Ensino
- Tecnologia Java
- Histórico
- Fundamentos da Programação Java
- Referências



# Conteúdo Programático

- Linguagem de Programação Java
- Ambiente de Programação
- Instruções de Controle do Fluxo de Dados
- Estruturas de Dados Homogêneas
- Fundamentos de Orientação a Objeto
  - Classes e Objetos
  - Encapsulamento
  - Herança
  - Polimorfismo
- Classes Abstratas e Interfaces
- Tratamento de Exceções
- Programação Gráfica (GUI)
- Coleções e Comparadores
- *Generics*
- Introdução a Programação Web



# Recursos e Metodologia de ensino

- Aulas expositivas teóricas
- Aulas práticas em laboratório
- Exercícios de fixação usando exemplos reais
- Desenvolvimento de atividades **extraclasse**
- Uso de **recursos virtuais** de apoio à aprendizagem
- Diversificação dos métodos de exploração do conteúdo disciplinar de acordo com a necessidade da abordagem instrucional



# Bibliografia

- **Básica**

BJARNE Stroustrup, The C++ Programming Language, 4th Edition, Addison-Wesley Professional, 2013.

HORSTMANN, C. S., CORNELL, G. Core Java, Volume I - Fundamentals, 8th Edition, Prentice Hall, 2016.

LANO, K. UML 2 Semantics and Applications, Wiley, 2009.  
[EBRARY]

- **Complementar**

HORSTMANN, C. S., CORNELL, G. Core Java2 , volume 1 e volume 2, Makron Books, 2001.

DEITEL, H. M. e DEITEL, P. J. Java: Como Programar, 8 edição, Pearson, 2010.

BARCLAY, K.; Savage, J. Object-Oriented Design with UML and Java, Elsevier, 2003. [EBRARY]



# Cronograma das Atividades

- O período letivo da turma de terça e quinta-feira é de **10/03** até **07/07/2020**;
- Todas as aulas utilizaram o laboratório em período integral;
- Existem **18 semanas** com **33 aulas** previstas para este semestre, com diversas atividades;
- A **última aula** corresponde a aula de encerramento e apresentação de resultados finais da turma;
- No calendário oficial da UnB existem alguns dias para possível **reposição de aulas**, se for necessário.



# AVALIAÇÃO

- Consiste em 7 atividades avaliativas (**V1,V2,P1,V3,V4, TF e V5**) mais uma prova de reposição (**PR**), em caráter de substituição à ausência justificável na **P1**, de acordo com a comprovação coerente com a legislação nacional.
- A realização e entrega de exercícios, tarefas e trabalhos solicitados pelo docente ainda consiste em outra nota (**E**).
- Respeitando o peso de cada avaliação é calculada a Média Final (**MF**) entre as notas obtidas no semestre, conforme é indicado no cálculo da **MF**:

$$\begin{aligned} \mathbf{MF} = & ( (V1 \times 0,03) + (V2 \times 0,045) + (P1 \times 0,225) + \\ & + (V3 \times 0,09) + (V4 \times 0,12) + (TF \times 0,27) + \\ & + (V5 \times 0,12) + (E \times 0,1) ) \end{aligned}$$



# AVALIAÇÃO

- A atividade de Reforço é OBRIGATÓRIA e será realizada semanalmente, até a próxima avaliação (prova), em que o estudante que obtiver nota superior a 3,0 não terá mais a obrigatoriedade. Caso a sua nota seja igual ou inferior a 3,0 ele entra ou permanece em **Reforço** até a nova prova;
- A falta na atividade de **reforço** só será admitida mediante justificativa direta ao professor da disciplina que replanejará a atividade como for mais adequado ao aluno;
- Quando o estudante realizar a **PR**, ela substituirá a nota da **P1** que o aluno esteve ausente e foi aceita sua justificativa, sendo feito o cálculo regular para obtenção da Média Final (**MF**). As demais provas não tem **PR**;





# AVALIAÇÃO

- A contabilização das atividades avaliativas identificadas com V (V1,V2,V3,V4 e V5) só serão usadas no cálculo da **MF** se todos os conteúdos que fizerem parte da respectiva avaliação estiverem na situação SATISFATÓRIA, antes da realização da avaliação V correspondente, caso contrário o valor a ser agregado ao cálculo da **MF** será **ZERO** na respectiva avaliação V.

**Para ser APROVADO na disciplina o estudante deverá:**

- Obter pelo menos **75%** de frequência nas aulas;
- Possuir nota igual ou superior a **4,0** pontos em **TF**;
- Atingir Média Final (**MF**) maior ou igual a **5,0** pontos.



# Considerações Finais



*"... o êxito na educação é consequência de três elementos indissociáveis: o Trabalho, a Solidariedade e a Perseverança."*

**PESTALOZZI**



# Introdução

## JAVA

- Linguagem de Programação Computacional
  - pertencia a *Sun Microsystems* => **ORACLE**
  - desenvolvida por *James Gosling*

Linguagens Convencionais



processo tradução



**binário**

(executado pela máquina)

Linguagem Java



processo tradução

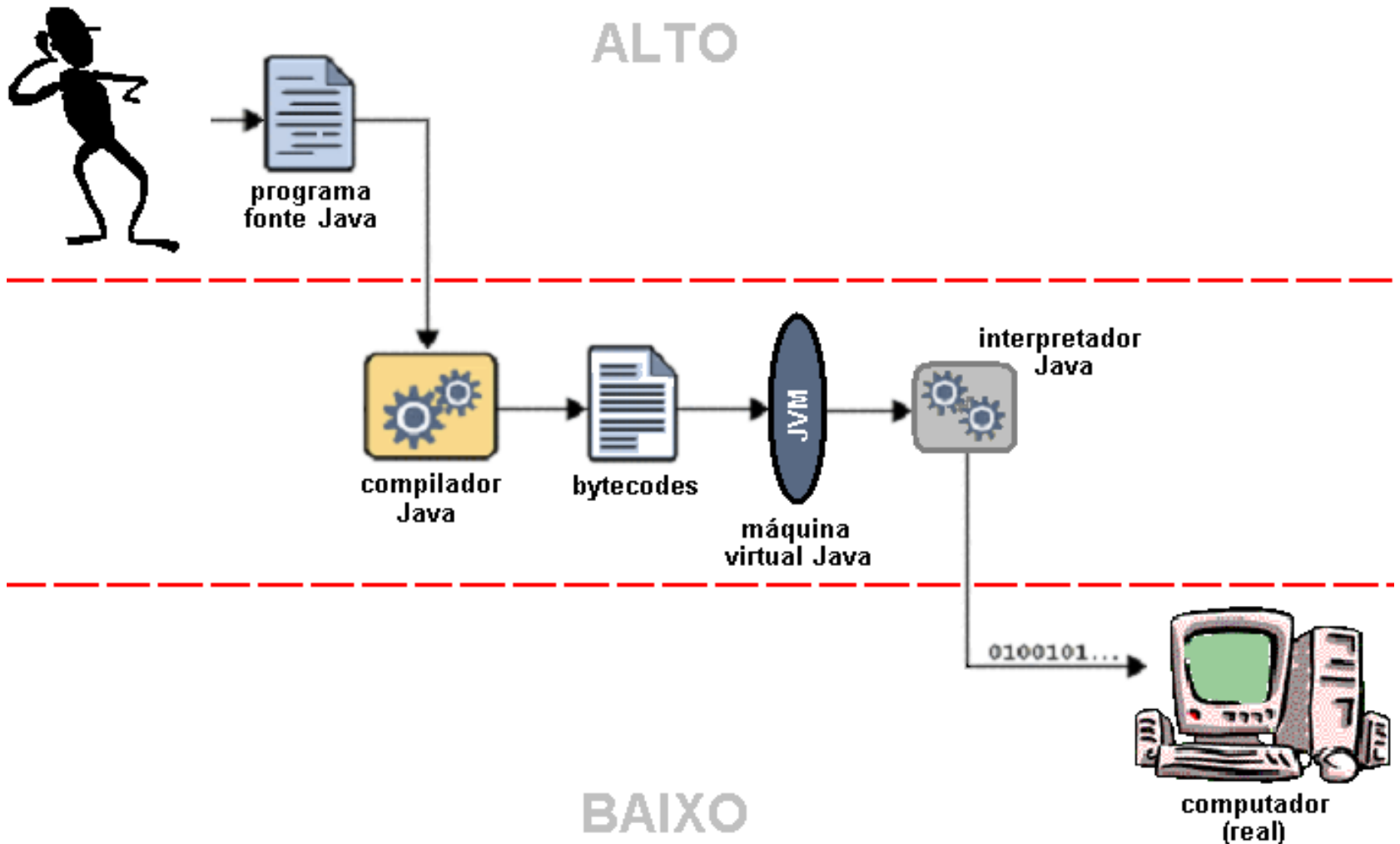


**bytecode**

(executado pela máquina virtual)



# Programação Java



# Histórico

- **1991 – Engenheiros da Sun Microsystems**
  - pequena linguagem para equipamentos de consumo eletrônicos
  - independência de arquitetura com segurança
  - projeto Green
- **1992 – Produto \*7 (StarSeven - controle remoto)**
- **1993 – Novo nome projeto: *First Person Inc.***
- **1994 – Elaboração do *browser* para Internet (1993)**
- **1995 – SunWorld apresenta esta tecnologia Java**
  - projeção dos atuais *applets*
- **1996 – Lançamento da primeira versão do Java**
  - Grande difusão do Java com Netscape 2.0
  - Esboço de futuro para linguagem Java no JavaOne/96
- **1998 – Conferência JavaOne – Java 1.2 (Java 2)...**

# Tecnologia Java

Basicamente, existem três tipos de programação Java:

- **Aplicações** – sistemas computacionais convencionais
- **Applets** – programa executado em browser (cliente)
- **Servlets** – programa executado no servidor Web



# Tecnologia Java

- **Java Standard Edition (JSE - ou J2SE de Java 2)**
  - Para computadores pessoais e notebooks, sendo por ela construídas a maioria das aplicações Java
  - Divide-se em:
    - Java Development Kit (JDK) ou Software/Standard Development Kit (SDK): ambiente desenvolvimento
    - Java Runtime Edition (JRE): ambiente execução
- **Java Micro Edition (JME - ou J2ME de Java 2)**
  - Para dispositivos móveis, sendo dividida em: CLDC (celular, smartfone,...) e CDC (palmtops, pocket,...)
- **Java Enterprise Edition (JEE - ou J2EE de Java 2)**
  - Para aplicações corporativas, integração de sistemas ou distribuição de serviços para terceiros

# Tecnologia Java

## JSE

JDK	Java Language	Java Language											
	Tools & Tool APIs	java	javac	javadoc	apt	jar	javap	JPDA		JConsole			
		Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot		Scripting	JVM TI		
	Deployment Technologies	Deployment			Java Web Start				Java Plug-in				
		AWT				Swing			Java 2D				
	User Interface Toolkits	Accessibility		Drag n Drop		Input Methods		Image I/O		Print Service		Sound	
	Integration Libraries	IDL	JDBC		JNDI		RMI		RMI-IIOP				
	Other Base Libraries	Beans	Intl Support		Input/Output		JMX		JNI		Math		
		Networking	Override Mechanism		Security		Serialization		Extension Mechanism		XML JAXP		
	lang and util Base Libraries	lang and util		Collections		Concurrency Utilities		JAR		Logging		Management	
		Preferences API		Ref Objects		Reflection		Regular Expressions		Versioning		Zip	Instrumentation
Java Virtual Machine													
Platforms													

Java SE API											
-------------	--	--	--	--	--	--	--	--	--	--	--





# Tecnologia Java

Algumas outras siglas e expressões importantes:

- **JVM** – Java Virtual Machine
- **API** – Application Programming Interface
- **IDE** – Integrated Development Environment
- **JSP** – Java Server Pages
- **GUI** – Graphic User Interface
- **CGI** – Common Gateway Interface



# Tecnologia Java

## PLATAFORMA

A maioria das plataformas é formada pelo conjunto de hardware e software (S.O.) que trabalham juntos.

A plataforma Java é diferente, por ser formada somente pelo software que opera sobre uma outra plataforma qualquer.

### Convencional

**Software**  
(sistema operacional)

+

**Hardware**

### JAVA

**Software**

**JVM + Bibliotecas**



# Tecnologia Java

## PLATAFORMA PADRÃO

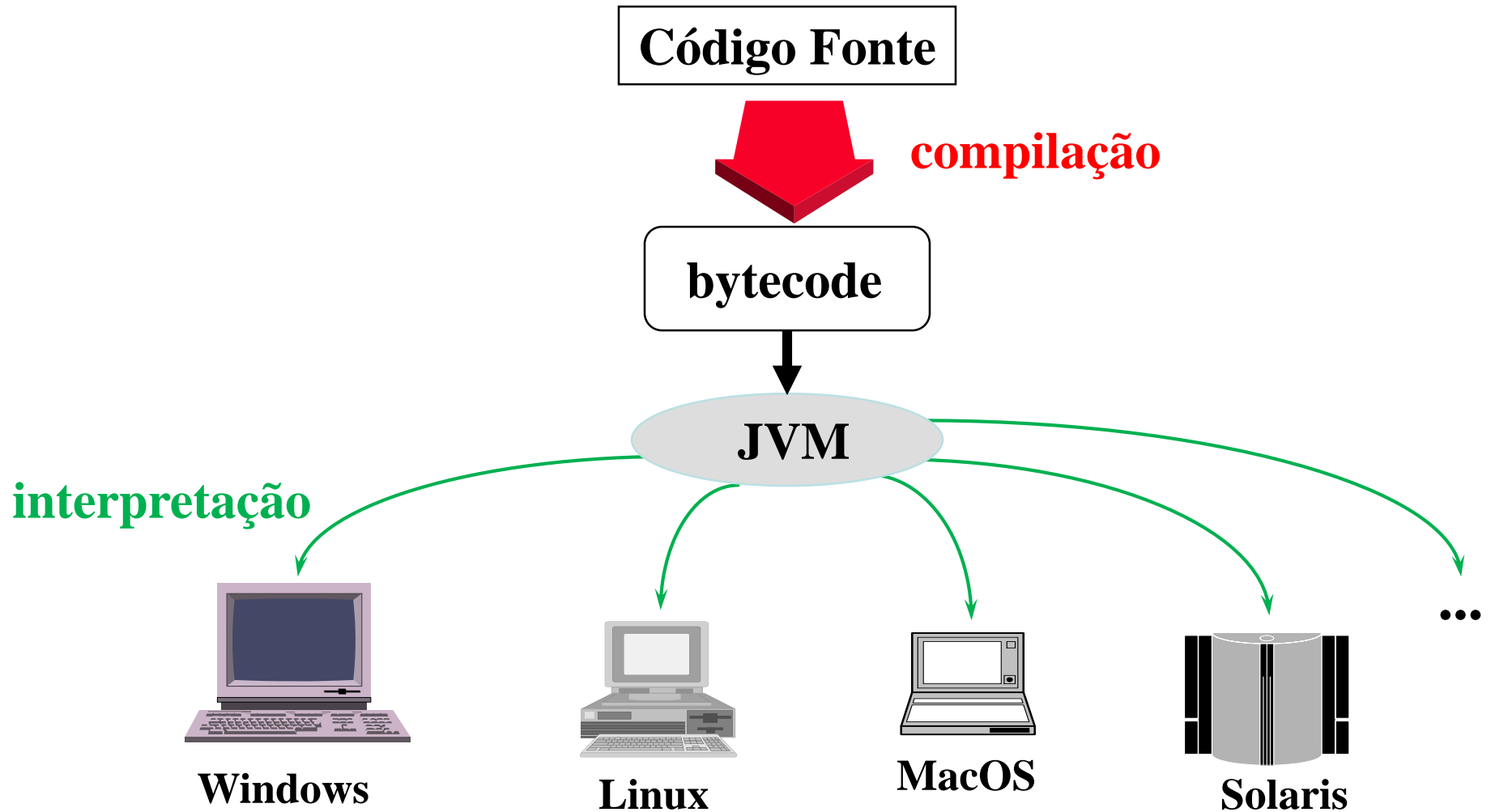
A plataforma padrão Java é composta por:

- **javac** – compilador Java
- **javadoc** – gerador de documentação
- **java** – executor de programa Java
- **appletviewer** – visualizador de applets Java

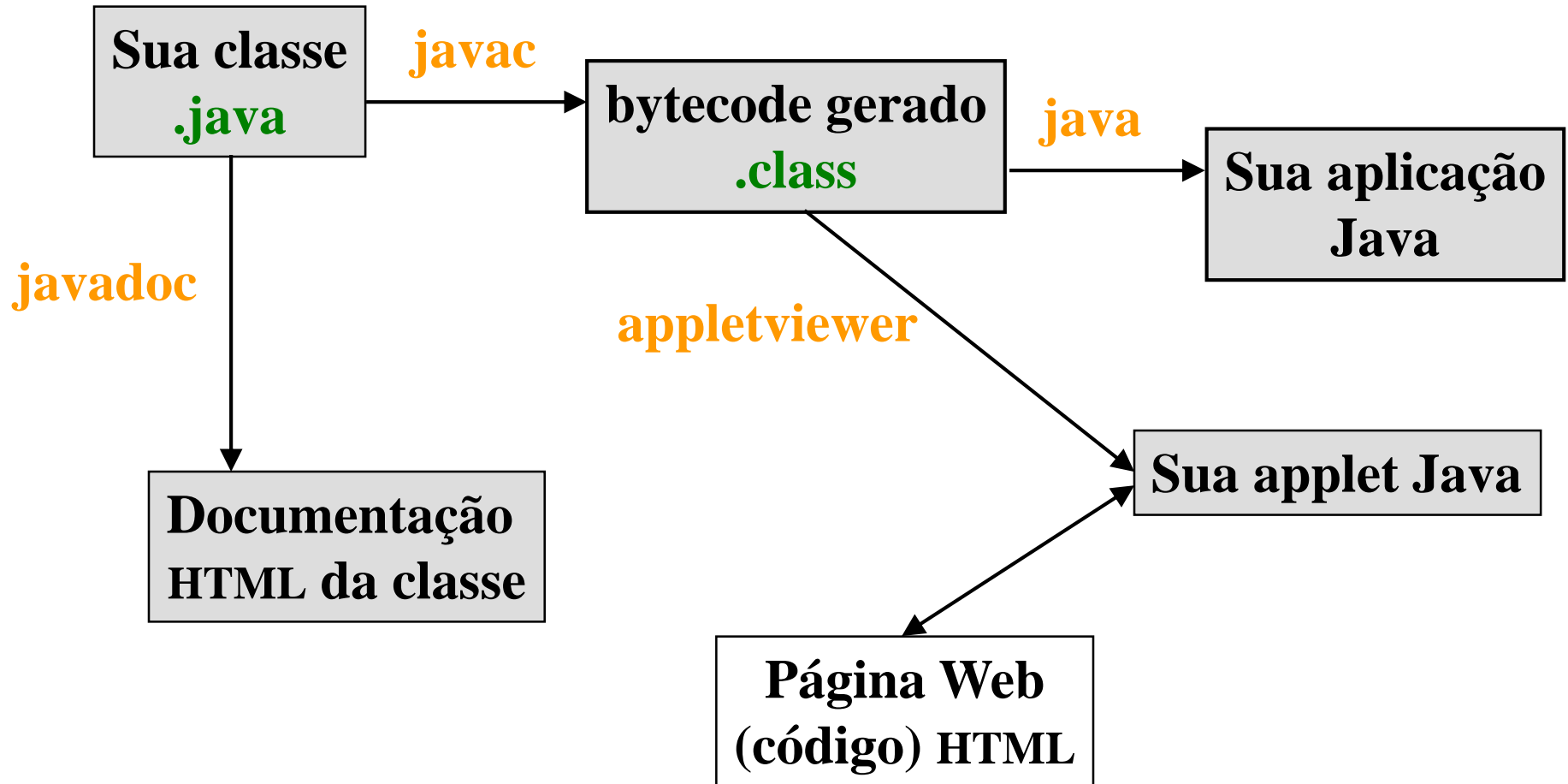


# Tecnologia Java

## MULTIPLATAFORMA



# Tecnologia Java



# Características da Linguagem Java

- **Simples:** sintaxe semelhante a Linguagem C++, sem recursos avançados na manipulação de memória pelo programador, além de ser destinadas a criação de “pequenos programas”
  - **Orientação a Objeto (OO):** técnica de programação computacional (paradigma) centrada nos dados (objetos) e em suas interfaces para com estes objetos, tendo Java nascido orientado a objeto (não foi adaptada)
  - **Portável:** suas aplicações podem ser executadas em diferentes plataformas sem adaptações, fornecendo as mesmas funcionalidades em redes heterogêneas (Internet)
    - Concepção da Sun Microsystems
- “Write once, run anywhere.” ⇒ Uma vez escrito, executa em qualquer lugar.

# Características da Linguagem Java

- **Robusta:** enfatiza a verificação antecipada de possíveis problemas na verificação dinâmica posterior e na eliminação de situações sujeitas a erros de programação (elimina a sobrescrita de memória e consequentemente a destruição de dados nela armazenados)
- **Segura:** elaboração de programas que protegem o S.O., além do ambiente de execução do próprio programa; considerada uma das linguagens mais segura para Programação
- **Distribuída:** componentes Java podem estar em uma máquina e serem acessados por outra a distância, com a mesma facilidade de acesso a um arquivo local, sendo esta tecnologia nascida para World Wide Web (sem adaptações)

# Características da Linguagem Java

- **Desempenho:** apesar de interpretar os bytecodes, existem alternativas para compilação (JIT- *just in time*, entre outros) com desempenho melhorado de 10 a 20 vezes em sua velocidade
- **Múltiplas Linhas de Execução (*multithreading*):** a capacidade de múltiplas execuções dos processos envolvidos, simultaneamente, promove melhor interatividade e comportamento em tempo de execução
- **Arquitetura Neutra:** geração de bytecodes independentes da arquitetura específica de execução (código neutro), sendo estes interpretados pelo sistema Java em tempo de execução da JVM





# Arquitetura Java

## Máquina Virtual Java

- *Java Virtual Machine*
- Máquina imaginária implementada por meio de software na emulação de uma máquina real
- Prover a especificação da plataforma no qual os códigos Java serão executados, através da interpretação na máquina imaginária

## Coletor de Lixo Java

- *Garbage Collection*
- Contínuo processo executado em *background* sobre o S.O. que gerencia toda memória alocada sobre a máquina real
- Em cada ciclo da CPU da JVM é analisado o uso da memória e desalocado o que não está sendo usado



# Iniciando a Programação

## Programa (ou código)

- ➔ Conjunto de instruções sequenciais que solicita que o computador execute alguma ação (ou atividade) por meio de uma comunicação, no nosso caso, usando uma linguagem de programação

Fonte - escrito na linguagem desejada, no caso - `‘.java’`

Objeto (compilado) - fonte traduzido em bytecodes - `‘.class’`



# Iniciando a Programação

## Estrutura de programa na linguagem Java

- Programas em Java são construídos a partir de classes
- Com a definição de uma classe se pode criar qualquer número de objetos, conhecidos como modelos daquela classe
- Uma classe pode ser formada por 2 membros:
  - **Campos (ou atributos)**: dados que pertencem a classe e seus objetos, compondo seu **estado**
  - **Métodos**: conjunto de instruções (subprogramas) que operam os campos para manipular o **estado** do objeto

# Iniciando a Programação

## Estrutura de programa na linguagem Java

- O método *main* é obrigatório nos aplicativos Java, pois é a partir dele que se inicia sua execução, com exceção para os *applets* (web)
- Está presente em **Java** o conceito de bloco de instruções indicadas pelos marcadores { e } para cada bloco elaborado
- Toda instrução deve estar entre as chaves '{', '}' e ser encerrada com ';'
- Esta linguagem é *case sensitive* (maiúsc./minúscula)
- Utilização de **texto estruturado** na construção de programas adequados e com qualidade

# Iniciando a Programação

## Estrutura de programa na linguagem Java

- Valores do tipo **String** são entre aspas ("), enquanto que um único caractere estará entre apóstrofes ('')
- Identificadores usam letras, números e sublinha '\_'
- 3 tipos de comentários no código fonte em Java:
  - // comentário no resto linha, após esta simbologia
  - /\* inicia um bloco de comentário até \*/ que encerra
  - /\*\* inicia o bloco de comentário que fará parte da documentação gerada para este programa, sendo encerrado em \*/



# Iniciando a Programação

## IDENTIFICADORES

- Recursos definidos pelo programador que recebem seus nomes (identificadores), como: campos, métodos, ...
- Criação de nomes para estes recursos usam letras alfabéticas, números, sublinha ('\_') e \$, facilitando a programação, convencionalmente
- Nomes em Java são traduzidos em **Unicode**, podendo fazer uso de muitos caracteres especiais como  $\pi$  como identificador válido
- Seu primeiro caracter deve ser letra, se evitando o uso de caracteres especiais na programação convencional
- Letras maiúsculas são diferentes de minúsculas
- Não pode ser igual a uma palavra reservada em Java
- Devem possuir fácil reconhecimento, sendo significativos

# Iniciando a Programação

## **IDENTIFICADORES** (convenção)

- Campos (variáveis) e métodos iniciam com letras minúsculas, porém, quando o identificador é um nome composto (mais que uma palavra) a primeira letra de cada palavra, após a primeira, é sempre maiúscula

**Exemplos:** contador, idadeMedia ou indiceTaxaAumento

- Classes iniciam com letras maiúsculas e para os nomes compostos segue a mesma norma descrita anteriormente

**Exemplos:** Pessoa ou DispositivoMovel

- Constantes são definidas com caracteres todos em maiúsculos

**Exemplos:** TAMANHO ou VALORMAXIMO

# Iniciando a Programação

Não usar palavras reservadas como identificadores.

## **Palavras Reservadas (50)**

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while





# Iniciando a Programação

## Tipo de Dados Primitivos (escalares)

- Todos os outros tipos de dados em **Java** são baseados em um desses **oito tipos**
- A definição do tipo de dados permite que o computador aloque e mantenha livre um espaço exato de memória que será utilizado pelo “programa” elaborado

<u>Tipo</u>	<u>Descrição do tipo</u>	<u>Quantidade</u>
inteiro	byte, short, int, long	4
real (ponto flutuante)	float, double	2
lógico	boolean	1
caracter	char	1



# Iniciando a Programação

## Tipo de Dados Primitivos

Tipo		Tamanho em bits	Faixa
byte	(inteiro)	08	-128 a 127
short	(inteiro)	16	-32768 a 32767
int	(inteiro)	32	-2147483648 a 2147483647
long	(inteiro)	64	-9223372036854775808 a 9...
float	(real)	32	3.4E-38 a 3.4E38
double	(real)	64	1.7E-308 a 1.7E308
boolean	(lógico)	8	true ou false
char	(caracter)	16	0 a 65535

⇒ Por padrão, valor real (ponto flutuante) em Java é **double**.



# Iniciando a Programação

## VARIÁVEL

- Posição nomeada de memória usada para guardar um valor que pode ser modificado pelo programa
- As variáveis em Java devem ser declaradas e iniciadas antes de serem usadas dentro de seu escopo (método ou bloco)
- Forma geral da definição ou declaração de variável  
`<tipo de dado> <identificador>;`

onde

`<tipo de dado>` é qualquer tipo de dado válido

`<identificador>` um ou mais nomes de identificadores separados por ‘,’

→ Exemplos: `byte valor;`  
`double totalPeso, salarios;`

# Iniciando a Programação

## OPERADOR DE ATRIBUIÇÃO

Realiza o armazenamento de um determinado valor, representado a direita do símbolo de igual (=), a um local de armazenamento na memória, representado por um identificador sempre à esquerda do símbolo (=).

Este valor a ser armazenado no identificador à esquerda do símbolo poder ser:

- um valor único → `numero = 5; ou sexo = 'M';`
- o resultado de uma expressão → `total = 10 + 20;`

Forma Geral

à esquerda ← símbolo → à direita

<identificador> = <expressão>;

Em Java é possível múltiplas atribuições, exemplo:

```
aux = valor = contador = 21;
```

# Iniciando a Programação

## CONSTANTE

- Posição nomeada de memória usada para guardar um único valor que não pode ser modificado pelo programa
- Todas constantes em Java devem ser declaradas antes de serem usadas
- Forma geral de definição de constate

**final** <tipo de dado> <identificador> = <valor>;

onde

<tipo de dado> - qualquer tipo de dado válido em Java

<identificador> - nome da constante (seu identificador)

**final** -palavra reservada que identifica a criação da constante

<valor> - valor atribuído a constante, sem poder alterar

→ Exemplo: **final** int MAXIMO = 30;

# Iniciando a Programação

## OPERADORES ARITMÉTICOS

### Operador Unário

- menos (troca de sinal)

### Operadores Binários

- + adição
- subtração
- \* multiplicação
- / divisão convencional (envolve ao menos um valor real)
- / divisão inteira (envolve valores inteiros somente - *div*)
- % *mod* - resto da divisão inteira

### Incremento e Decremento

++ incremento

-- decremento

Prefixo    a++    ou    a--

Sufixo      ++a    ou    --a

### Precedência

- |   |       |
|---|-------|
| 1 | ++ -- |
| 2 | * / % |
| 3 | + -   |
| 4 | =     |



# Iniciando a Programação

## EXPRESSÕES - Operadores Aritméticos de Atribuição

Estes operadores ( $+=$  ,  $-=$  ,  $*=$  ,  $/=$  ,  $\%=$  ) são usados com uma variável a sua esquerda e uma expressão a sua direita. A operação consiste em atribuir um novo valor à variável que dependerá do operador e da expressão à direita

$\langle \text{variável} \rangle \langle \text{operador} \rangle = \langle \text{expressão} \rangle ;$

### Exemplos:

$i += 2;$

equivale a

$i = i + 2;$

$x *= y + 1;$

equivale a

$x = x * (y + 1);$

$t /= 2.5;$

equivale a

$t = t / 2.5;$

$p \% = 5;$

equivale a

$p = p \% 5;$

$d -= 3;$

equivale a

$d = d - 3;$

:

:

:



# Iniciando a Programação

## CUIDADOS COM ALGUMAS ATRIBUIÇÕES

- O tipo **char** corresponde a um inteiro especial, sendo exclusivamente positivo, representando um único Unicode (mais combinações que o padrão ASCII)
- As operações sobre **inteiros** resultará sempre em no mínimo um valor **int**, por exemplo:  
    byte x = 1;  
    byte y = 2;  
    byte total = x + y; // erro de compilação, sendo  
                          // correto   int total = x + y;
- A atribuição de **long** e **float** deve ser feita com:  
    long x = 10L   // ou L em minúsculo  
    float y = 2.4F // ou F em minúsculo
- Tipo **boolean** só recebe **true** ou **false** e não **1** ou **0** (zero)



# Iniciando a Programação

## OPERADORES RELACIONAIS

Todas as expressões relacionais resultam em um valor lógico, ou seja, **true** ou **false** (verdadeiro ou falso respectivamente)

<u>Operadores</u>		<u>Expressão</u>
igualdade	==	$x == y$
diferente	!=	$x != y$
maior que	>	$x > y$
menor que	<	$x < y$
maior ou igual	>=	$x >= y$
menor ou igual	<=	$x <= y$



# Iniciando a Programação

## OPERADORES LÓGICOS

As expressões com operadores lógicos respeitam as definições da lógica convencional e suas propriedades matemáticas estudadas nos conteúdos representados por “tabelas verdades”.

<u>Operadores</u>	<u>Expressão</u>	<u>Realização</u>
E <b>&amp;&amp;</b>	op1 <b>&amp;&amp;</b> op2	só avalia op2 se op1 for <i>true</i>
E <b>&amp;</b>	op1 <b>&amp;</b> op2	sempre avalia op1 e op2
OU <b>  </b>	op1 <b>  </b> op2	só avalia op2 se op1 for <i>false</i>
OU <b> </b>	op1 <b> </b> op2	sempre avalia op1 e op2
NÃO <b>!</b>	<b>!</b> op1	nega ou troca valor de op1



# Iniciando a Programação

## CONVERSÃO OU CASTING

*Casting* é o processo de conversão de um tipo primitivo de dado para outro tipo, sendo comum nos dados numéricos.

Existem dois tipos de conversão: **implícito** e **explícito**.

- Implícito: realização automática quando tamanho do tipo a ser convertido é maior que o tipo original (ou atual)
- Explícito: necessidade de ser explícita porque o tamanho do tipo original (ou atual) é maior que o tipo a ser convertido

Exemplo: `double x = 6.4;`

`int y = 2;`


`int total = (int) (x / y); // converte resultado em int`

Não é possível fazer *casting* para o tipo de dado **boolean**.



# Iniciando a Programação

## PROMOÇÃO E CASTING



<b>PARA:</b> <b>DE:</b>	byte	short	char	int	long	float	double
byte	=	<i>impl.</i>	(char)	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>
short	(byte)	=	(char)	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>
char	(byte)	(short)	=	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>
int	(byte)	(short)	(char)	=	<i>impl.</i>	<i>impl.</i>	<i>impl.</i>
long	(byte)	(short)	(char)	(int)	=	<i>impl.</i>	<i>impl.</i>
float	(byte)	(short)	(char)	(int)	(long)	=	<i>impl.</i>
double	(byte)	(short)	(char)	(int)	(long)	(float)	=

*impl.* - corresponde a conversão implícita



# Iniciando a Programação

## CASTING EM OPERAÇÕES BINÁRIAS

- 4 regras básicas na operação binária de *casting*
  - Se um dos operantes é double, o outro operante é convertido para **double**
  - Se um dos operantes é float, o outro operante é convertido para **float**
  - Se um dos operantes é long, o outro operante é convertido para **long**
  - Senão todos os operantes são convertidos para **int** na resolução da operação



# Iniciando a Programação

## Exemplos:

### Conversão Implícita

```
int idade;  
double medialdades;  
  
idade = 16;  
medialdades = idade;
```

O processamento acima atribui o conteúdo inteiro de idade a variável double medialdades, efetivando um casting implícito.

### Conversão Explícita

```
short ano = 2010;  
byte valor = ano; // erro  
  
byte valor = (byte) ano;
```

O processamento correto acima realiza um casting explícito para armazenar um dado short (maior) em um byte (menor em capacidade de armazenamento), podendo ocorrer **perda de dado**.



# Exercício Proposto

- 1.a) Elabore um PROGRAMA em C que armazene três alturas de pessoas, onde esta quantidade de pessoas deverá estar definida em uma constante. Armazene nestas alturas os valores 1.58, 2.07 e 0.55 em três variáveis diferentes. Por fim, realize o cálculo da média aritmética destas alturas e as apresente ao usuário.

Após o término do PROGRAMA em C, que deverá estar completo e funcionando corretamente encerre este exercício e utilize o ambiente de desenvolvimento em Java indicado por seu professor.

- 1.b) Depois do programa em C, deverá ser elaborada uma nova tradução para esta nova linguagem que se está começando a estudar (**Java**), onde as declarações e expressões de cálculos deverão ser escritos baseados nesta linguagem, mantendo os comandos, por enquanto, com as palavras reservadas usadas na Linguagem C.

# Programando em Java

## Restrições Importantes na Programação Java

- Programas Java são escritos em editores de texto comuns (ASCII e Unicode) e gravados com extensão java (**.java**)
- Estes arquivos (.java) consistem nos códigos fontes da Linguagem Java, sendo denominados classes
- Um programa (ou aplicação) Java é composto por uma ou várias classes
- O nome deste arquivo fonte DEVE ser o mesmo do nome da classe descrita por meio de seu texto
- Toda aplicação Java deve ter ao menos um método, sendo este o principal (**main()**), com exceção dos programas para Web (*applets* por exemplo)





# Programando em Java

## Execução por Linha de Comando

- Após salvar o arquivo fonte (.java), o mesmo deverá ser compilado para geração do arquivo bytecode

- Seguir para opção de Prompt de comando do S.O.
- Executar o compilador Java (javac)

**`javac <NomeArquivoFonte.java>`**

- Geração do arquivo bytecode com o mesmo nome e extensão .class
- Acionar o executor Java de bytecodes para execução do programa Java elaborado
- No Prompt de comando será acionado o executor

**`java <NomeArquivoFonte>`**

- Não é necessária a extensão do arquivo a ser executado (.class)

→ Precedendo a esta execução é necessária a instalação de um JDK

# Programando em Java

## Parâmetros pela Linha de Comando

- É possível receber e manipular parâmetros através da linha de comando em Java, por exemplo:

```
System.out.println("Primeiro Parâmetro= " + args[0]);  
System.out.println("Segundo Parâmetro= " + args[1]);
```

- Estes parâmetros são inseridos na linha de comando que aciona o executor java, após o nome do programa, por exemplo:

```
java c:/temp/TestaArgs Oi 32
```

- Observe que o caminho (*path*) onde um programa chamado **TestaArgs** se encontra no computador pode ser incluído no processo de compilação (javac) e execução (java) do mesmo.
- Todos os parâmetros de linha de comando são String.

# Referência de Criação e Apoio ao Estudo

## Material para Consulta e Apoio ao Conteúdo

- HORSTMANN, C. S., CORNELL, G., Core Java2 , volume 1, Makron Books, 2001.
  - Capítulo 1 e 3
- FURGERI, S., Java 2: Ensino Didático: Desenvolvendo e Implementando Aplicações, São Paulo: Érica, 2002.
  - Capítulo 1 e 2
- ASCENCIO, A. F. G.; CAMPOS, E. A. V., Fundamentos da programação de computadores, 2 ed., São Paulo: Pearson Prentice Hall, 2007.
  - Capítulo 1
- Universidade de Brasília (UnB Gama)
  - <https://cae.ucb.br/conteudo/unbfga>  
(escolha a disciplina **Orientação a Objetos** no menu superior)