

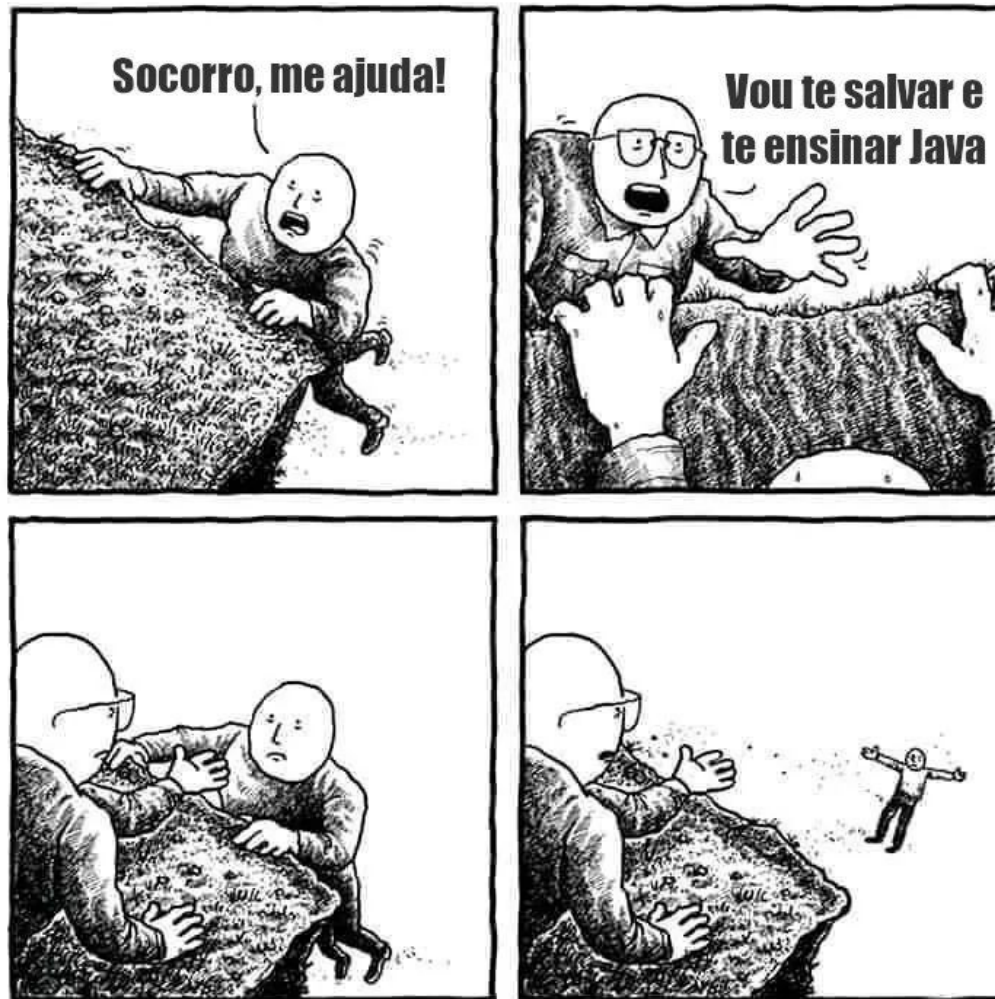
# 01-Fundamentos

Conteúdo: Introdução - História e Características. Ambiente de Desenvolvimento. Estrutura Básica do Código. Alô Mundo. Identificadores e Palavras Reservadas. Comentários. Instruções. Tipos de Dados: primitivos e por referência. Variáveis. Operadores. String em Java. Estruturas de Controle e Repetição. Vetores em Java.

# Bibliografia Recomendada

- Use a Cabeça! Java. 2ª edição. 2005. Kathy Sierra, Bert Bates. Editora Alta Books.
- SCJP – Certificação Sun para Programador Java 6 – Guia de Estudo. 2009. Kathy Sierra, Bert Bates. Editora Alta Books.





# Introdução – História (1)

- Apresentada pela Sun Microsystems em agosto de 1995, iniciou uma subsidiária
- Nasceu em ambientes UNIX
- Fortalecida pelo conceito de Network Computer
- 2009: Oracle compra a Sun (?)



# Introdução - Características (1)

- A tecnologia Java pode ser entendida como:
  - Uma linguagem de programação
  - Um ambiente de execução de aplicações (JRE)
  - Um ambiente de desenvolvimento (SDK)
- Orientada a Objetos
- Similar ao C++, mas simplificada
  - Não implementa herança múltipla
  - Abstrai o conceito de ponteiro

# Características (2)

- Enorme biblioteca de classes/funções
- Voltada para a criação de aplicações distribuídas (biblioteca TCP/IP)
- Permite executar algoritmos em concomitância ou simultaneamente
- Execução dinâmica e sob demanda
- Reflexiva (RTTI – Runtime Type Identification)
- Maior segurança
  - Não permite manipulação direta de ponteiros
  - Coleta automática de lixo

# Características (3)

- Portabilidade
- Compilador Java gera código para um processador virtual (máquina Java), o chamado Bytecode.
- Cada plataforma onde o aplicativo Java roda, deve ter um ambiente de execução (JRE - Java Runtime Environment) que interpreta cada bytecode e gera instruções da máquina em questão.
- Existem otimizadores para evitar a perda de performance pela interpretação, os chamados compiladores Just-In-Time.

# Teste Rápido

- O que é Java? Aponte todas que se aplicam.
  - (a) uma linguagem de programação
  - (b) um gerador de código de programação
  - (c) um ambiente de desenvolvimento
  - (d) uma ferramenta de execução de programas
  - (e) um substituto do C++



# Resposta do Teste Rápido

- O que é Java?

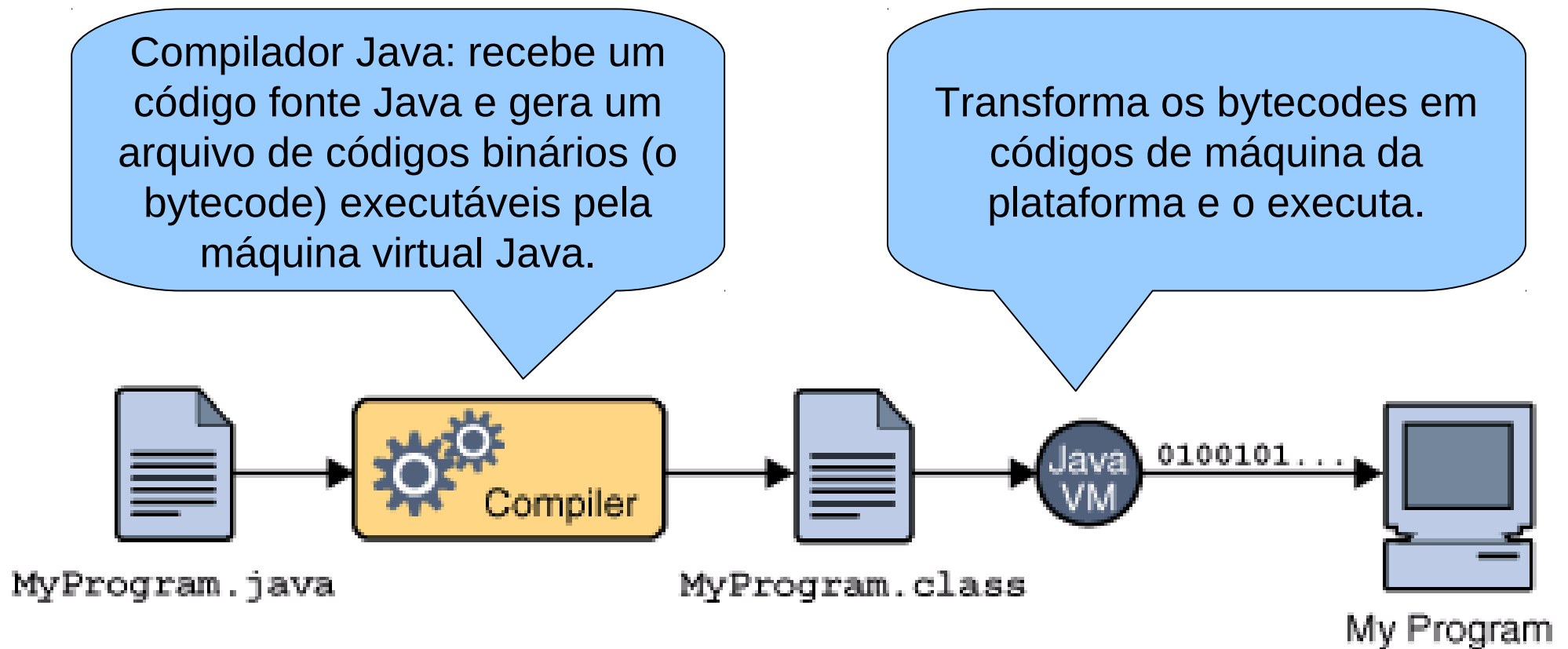
- (a) uma linguagem de programação
  - A linguagem em que se escreve os programas é chamada assim
- (b) um gerador de código de programação
  - Não! Embora a plataforma java gere códigos, eles são voltados para execução do programa sobre uma máquina virtual, não para programação.
- (c) um ambiente de desenvolvimento
  - Correto. Várias ferramentas acompanham o SDK para permitir o desenvolvimento de aplicações.
- (d) uma ferramenta de execução de programas
  - Sim. Java embute uma VM para execução de programas. Inclusive, atualmente, existem programas em outras linguagens que rodam sobre a JVM.
- (e) um substituto do C++
  - Não. Java herda características da linguagem C e C++, mas não foi feita para ser um substituto de C++, embora sua característica de alto nível e maior produtividade para algumas categorias de aplicações possam estimular essa substituição.

# Ambiente Desenvolvimento Java (1)

- JVM (Java Virtual Machine)
  - ♦ Carrega, verifica e executa *byte code* Java numa plataforma (hardware + sistema operacional)
  - ♦ Existem JVMs para várias plataformas
- Coletor de Lixo (Garbage collection)
  - ♦ Encarrega-se de liberar automaticamente memória alocada pelo programa que não é mais necessária
- Especificações: J2ME, J2SE e J2EE

# Ambiente Desenvolvimento Java (2)

- Processo de Compilação e Execução



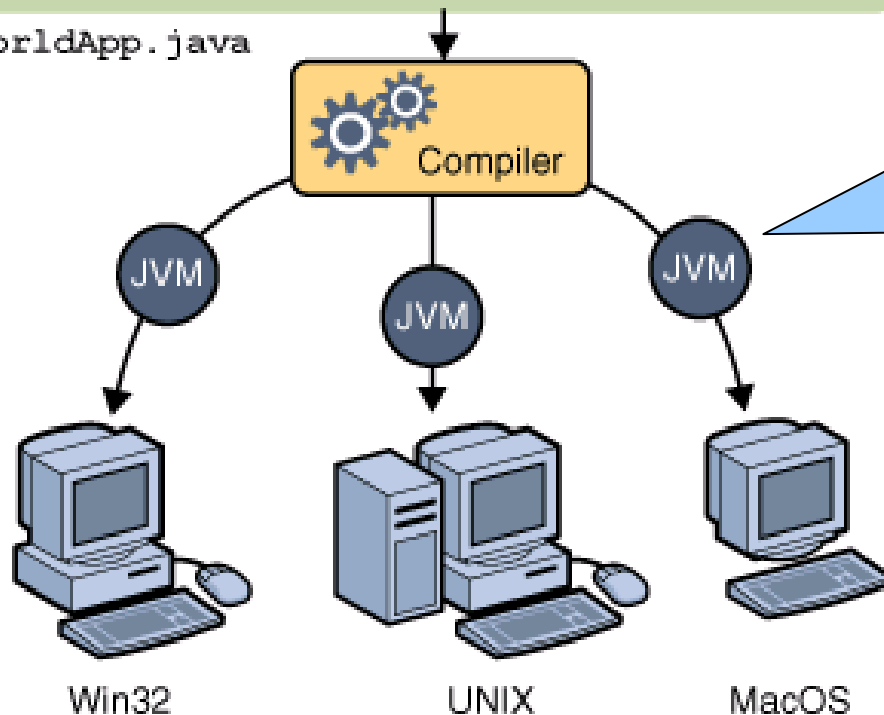
# Ambiente Desenvolvimento Java (3)

- Escreva uma vez, rode em qualquer lugar

Java Program

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



O mesmo bytecode é submetido para as JVM em cada uma das plataformas, executando da mesma maneira.

# Ambiente Desenvolvimento Java (4)

- Instalar o SDK (Software Development Kit) do Java, edição Standard
  - Sítio <http://www.oracle.com/technetwork/java>
  - Baixar o instalador Java 6 SE (cerca de 53.16 MB)
- Para ambiente de execução apenas, basta o JRE. O SDK já inclui o JRE.
- Utilizar um editor de texto qualquer
- Adicionalmente, pode-se usar uma ferramenta de edição de código: Eclipse, NetBeans, etc.

# Estrutura Básica do Código



- Salvar o código com o nome AloMundo.java

```
public class AloMundo {  
    public static void main(String[] args) {  
        System.out.println("Alô Mundo!");  
    }  
}
```

Exibe no console!

- Compilação: `javac AloMundo.java`
  - É gerado arquivo `AloMundo.class`
- Execução: `$ java AloMundo`

Não pode colocar a extensão!

- Todo o código Java deve estar dentro de uma classe.
- O nome do arquivo (sem extensão) deve ser o mesmo nome da classe pública.
- A rotina `main` é obrigatória em todo programa Java (ponto de início) e deve ter sempre a declaração exatamente como mostrado.

# Teste Rápido

- Qual são as ferramentas para: (1) gerar um bytecode a partir de um código Java e; (2) para subir uma máquina virtual e executar uma classe que contém um método main, respectivamente?
  - (a) notepad e dir
  - (b) javac e java
  - (c) JDK e JRE
  - (d) java e javac
  - (e) Eclipse e Netbeans

# Resposta do Teste Rápido

- (b) é a correta.
  - JDK, de Java Development Kit, é a SDK do Java (kit de desenvolvimento). Ele contém tanto o compilador (javac) quanto a VM (java). JRE é Java Runtime Environment, e contém somente o necessário para carregar a máquina virtual Java e rodar programas. Tanto Eclipse quanto Netbeans compilam e rodam – dentro da própria IDE – aplicativos Java, mas fazem isso por intermédio das ferramentas do Java (JRE e/ou JDK).



# Identificadores e Palavras Reservadas

- Identificador: qualquer nome dado pelo programador dentro de um programa
  - Exemplos: variável, classe, funções, etc.
- Java é sensível à caixa (*case sensitive*)
- 1º caractere deve ser letra, \_ ou \$
- Não pode ter espaço em branco
- Não pode ser igual a palavra reservada:

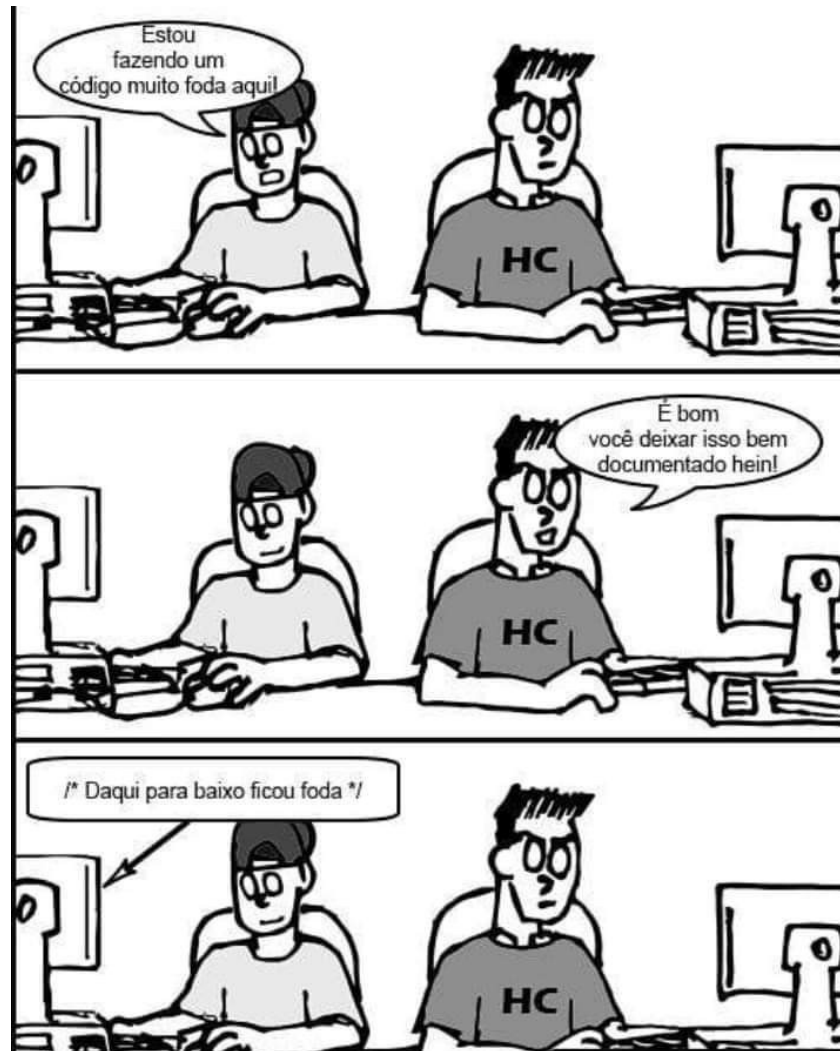
abstract, boolean, break, byte, case, catch, char, class, public, continue, default, do, double, else, enum, extends, false, final, finally, float, for, private, if, implements, import, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, const,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

# Comentários em Código

- Comentário de uma linha com `//` comentário
  - Todo resto da linha é ignorado
- Comentário de bloco com `/*` comentário `*/`
- JavaDoc: permite gerar documentação a partir dos comentários do código fonte

```
// Este programa é o AloMundo
public class AloMundo {
    /* Esse é o método
    main() */
    public static void main(Strings[] args) {
        System.out.println("Alô Mundo!"); // igual printf
    }
} // fim do programa
```

# Para rir e refletir



# Instrução e Bloco de Instruções

- Instrução tem seu fim determinado pelo ;

```
int x;  
int y;  
x=x+i;
```

```
int x; int y; x=x+i;
```

Bloco de instruções: { ... }

- Instruções são executadas na ordem definida
- Podem haver blocos aninhados
- Serve para definir o escopo de uma variável

# Teste Rápido

- Considere que o código abaixo está num arquivo chamado AloMundo.java. Aponte todos os erros encontrados.

```
class Alo Mundo {  
static void main(String params) {  
int final == 30  
system.out.println "fui!!!"  
}
```

# Resposta do Teste Rápido

```
public class AloMundo {  
    public static void main(String[] params) {  
        int x = 30;  
        System.out.println("Fui!");  
    }  
}
```

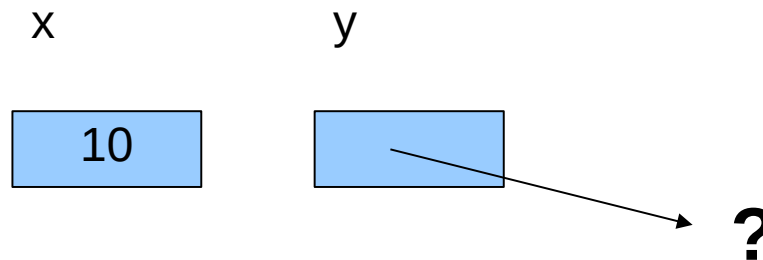
1. não pode haver espaço no nome da classe
2. o método main e a classe devem ter o modificador public
3. o tipo do parâmetro de main deve ser String[]
4. o identificador final para a variável é inválido
5. cada instrução deve terminar com ponto-e-vírgula
6. o operador == é incorreto
7. o s de System deve ser maiúsculo
8. o parâmetro do método deve estar entre parênteses
9. não havia o fechamento do bloco da classe. Indentação é fundamental!!!

# Tipos de Dados

- Duas categorias: primitivos e de referência
- Tipos primitivos: dados simples ou escalares
- Tipos de referência: guardam, ao invés do dado propriamente dito, um “ponteiro” para onde o objeto está supostamente instanciado (heap)

`int x=10;`

`MinhaClasse y;`



# Tipos Primitivos (1)

- Compatíveis entre plataformas (diferente do C)
- **boolean**: valor lógico true ou false
- **char**: caractere em notação unicode (16 bits). Pode ser usado com inteiro entre 0 e 65535.
- **byte**: inteiro de 8 bits. Assume valores entre -128 e 127
- **short**: inteiro de 16 bits. Assume valores entre -32768 e 32767
- **int**: inteiro de 32 bits. Valores entre -2.147.483.648 e 2.147.483.647



# Tipos Primitivos (2)

- **long**: inteiro 64 bits. Valores entre  $-2^{63}$  e  $2^{63}-1$ .
- **float**: representa números em 32 bits na notação de ponto flutuante. Ver *Java Specification Language* para detalhes sobre precisão e capacidade de representação.
- **double**: representa números em 64 bits na notação de ponto flutuante. Da mesma forma que o float, não deve ser usado para representar valores precisos (usar BigDecimal).  
Ver documentação para detalhes.

# Tipos Primitivos (3)

- Para todo tipo primitivo existe uma classe (Wrapper) correspondente
  - Classe Integer para o tipo int , Character para o tipo char, Double para double, etc.
- Permite o programador Java trabalhar apenas com variáveis do tipo referência
- Boxing e Unboxing permitem uso combinado
  - `int y = 4; Integer x = 5 + y;`
- Tipos primitivos:
  - Mais rápidos, pois podem ser manipulados diretamente por instruções elementares de hardware

# Variáveis (1)

- Sintaxe: <tipo-dado> <nome>;

`int idade; // valor inteiro`

`float preço; // valor decimal`

`int numeroSerie,ano; // declaração dupla`

`String nomePai,nomeMae; // String é uma classe`

- Atribuição de valor com operador =

`idade=23; // muda o valor durante a execução`

- Declarar e atribuir o valor

`int idade=23; // cria a variável inteira com valor 23`

# Variáveis (2)

- Tipo booleano
  - Valores possíveis: true ou false
  - Pode armazenar o resultado de uma expressão lógica, como  $(a \leq b)$

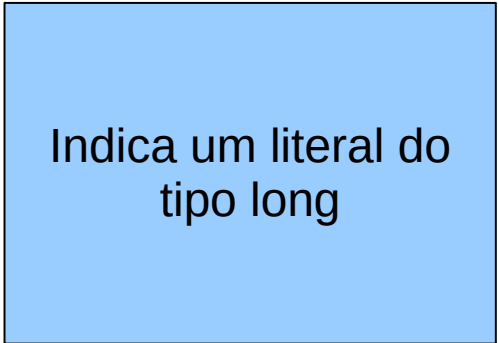
```
boolean fim=false;
```

- Tipos Inteiros: byte, int, char, short, long
  - Não ocorre exceção por overflow. Exemplo: um valor byte 127 somado a 1 resulta em -128
  - Ocorre exceção de divisão por zero (ArithmeticException)

# Variáveis (3)

- Exemplo

```
public class Arit {  
    public static void main(String args[]) {  
        byte a=127;  
        short b=32767;  
        int c=2147483647;  
        long d=9223372036854775807L;  
        System.out.println("Valor de a=" + a);  
        System.out.println("Valor de b=" + b);  
        System.out.println("Valor de c=" + c);  
        System.out.println("Valor de d=" + d);  
        d = d / 0; /* Causa ArithmeticException */  
    }  
}
```



Indica um literal do  
tipo long

# Teste Rápido

\_\_\_\_ v = 256;

Quais os tipos podem ser usados? Marque todos que se aplicam.

- (a) int
- (b) short
- (c) float
- (d) byte
- (e) char

# Resposta Teste Rápido

- Somente (d) não pode ser usado. O tipo primitivo byte, como o próprio nome diz, só pode conter um byte, o que permite representar um número decimal até 127 positivo.

# Operadores em Java

OPERADORES	DESCRIÇÃO
<code>expr++ expr--</code>	Pós incremento
<code>++expr --expr +expr -expr ~ !</code>	Unários
<code>* / %</code>	multiplicativos
<code>+, -</code>	aditivos
<code>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</code>	Deslocamento de bits
<code>&lt; &gt; &lt;= &gt;= instanceof</code>	Relacionais
<code>== !=</code>	Igualdade
<code>&amp;</code>	AND binário
<code>^</code>	XOR binário
<code> </code>	OR binário
<code>&amp;&amp;</code>	AND lógico
<code>  </code>	OR lógico
<code>? :</code>	ternário
<code>= += -= *= /= %= &amp;= ^=  = &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;=</code>	atribuição



# Variáveis (4)

```
byte b=3;
short s=2;
int i=10;
long m=7L;
boolean v=true;
//ERRO: s = s + s; (cannot convert int to short)
s = (short)(s+s); // agora OK s recebe 4... ou s += s;
m -= i; // m = -3 (7 menos 10)
// ERRO: m = i / 3.0; // cannot convert double to long
m = i / 4; // (10 dividido por 4 = 2) quociente
m = 7 % b; // m = 1 (resto da divisão de 7 por 3)
v = (v || false); // v recebe true (ou com curto-circuito)
v = (v & false); // v recebe false (e sem curto-circuito)
v = !v; // inversão lógica, v recebe true
```

Operadores aritméticos  
retornam int

# Variáveis (5)

- Tipo caractere
  - Armazena um caractere unicode de 16 bits
  - De 0 a 255 obedece tabela ASCII estendido
  - Constante representada entre apóstrofes ou pelo valor numérico
  - Exemplos:

```
char c='a';
```

```
char b='C';
```

```
char quebra='\n'; // escape newline
```

```
char a=65; // letra A maiúscula unicode
```

# Variáveis (6)

- Tipos de ponto flutuante
  - Valores podem ser representados na notação decimal (-24.321) ou científica (2.52E-31)
  - A maioria dos operadores usados nos inteiros funcionam com os de ponto flutuante, exceto % e os que operam bit a bit (deslocamento e lógicos)

- Exemplos:

```
float f=-23.3113F;
```

```
double d=2.52;
```

```
double c=2.52E-31;
```

```
d=f++; // Qual será o valor de d?
```



# Exercícios de Fixação

- Faça o exercício Primitivos (60 min)



# String em Java (1)

- **String** é uma classe, não um tipo primitivo
- Mesmo sendo uma classe, tem um tratamento especial dentro do código Java
- Possui operações de concatenação, extração de substring, procura de trecho, etc.

```
String nome="Ana";  
nome=nome+" Beatriz";  
int x = nome.length();  
char c= nome.charAt(1);  
"hamburger".substring(4, 8); // "urge"
```

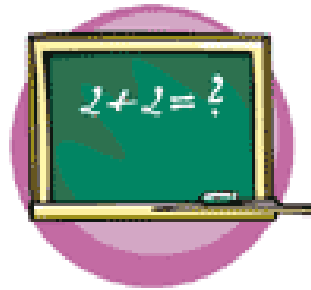
índice inicial,  
inclusivo

índice final,  
exclusivo

# String em Java (2)

- É imutável: métodos criam novo objeto String e retornam referência para o recém criado objeto

```
String s = "java";  
String s2 = "Dude";  
s.toUpperCase();  
s.replace('a', 'x');  
s.trim();  
s2.concat(" rapa");  
s2 = s2.toLowerCase();  
System.out.println(s+" "+s2);  
String s3=new String("java");  
if(s==s3) System.out.println("iguais");  
else System.out.println("diferentes");  
if(s2.equals("dude")) System.out.print("OK");
```



Qual o resultado?  
Quantos objetos foram criados?

# Estruturas de Controle

## Instrução if

- Sintaxe: if(<condição>) <instruções>

- Uma única instrução

```
if( idade > 18 ) System.out.println("Maior");  
System.out.println("Executa incondicionalmente");
```

- Bloco de instruções

```
if( precoTotal > 200.49 ) {  
    desconto=20.00;  
    System.out.println("Desconto 10%");  
}  
System.out.println("Executa após bloco do if");
```

# Estruturas de Controle

## Instrução if/else

- Sintaxe: `if(<condição>) <instruções_V>`  
`else <instruções_F>`

- Uma única instrução

```
if( temProduto ) System.out.println("Ok");  
else System.out.println("Em Falta");
```

- Bloco de instruções

```
if( qtdeEmEstoque >= qtdeVenda ) {  
    continua=true;  
    System.out.println("Disponível");  
} else {  
    continua=false;  
    System.out.println("Em falta");  
}
```

Da mesma forma, caso houvesse apenas uma instrução a executar, poderíamos omitir a demarcação de bloco.



# Estruturas de Controle

## Múltiplos if

- Sintaxe: `if(<condição>) <instruções>`  
`else if(<outra_condição>) <instruções>`

...

`else <instruções>;`

```
if( nota >= 9.5 ) System.out.println("CDF");  
else if( nota > 7 ) System.out.println("Bom");  
else if( nota >= 5 ) System.out.println("Regular");  
else if( nota >= 3 ) System.out.println("Fraco");  
else System.out.println("Burrinho");
```

# Estruturas de Controle switch

- Sintaxe:

Só pode tipos  
inteiros byte, char,  
short, int; ou enum.

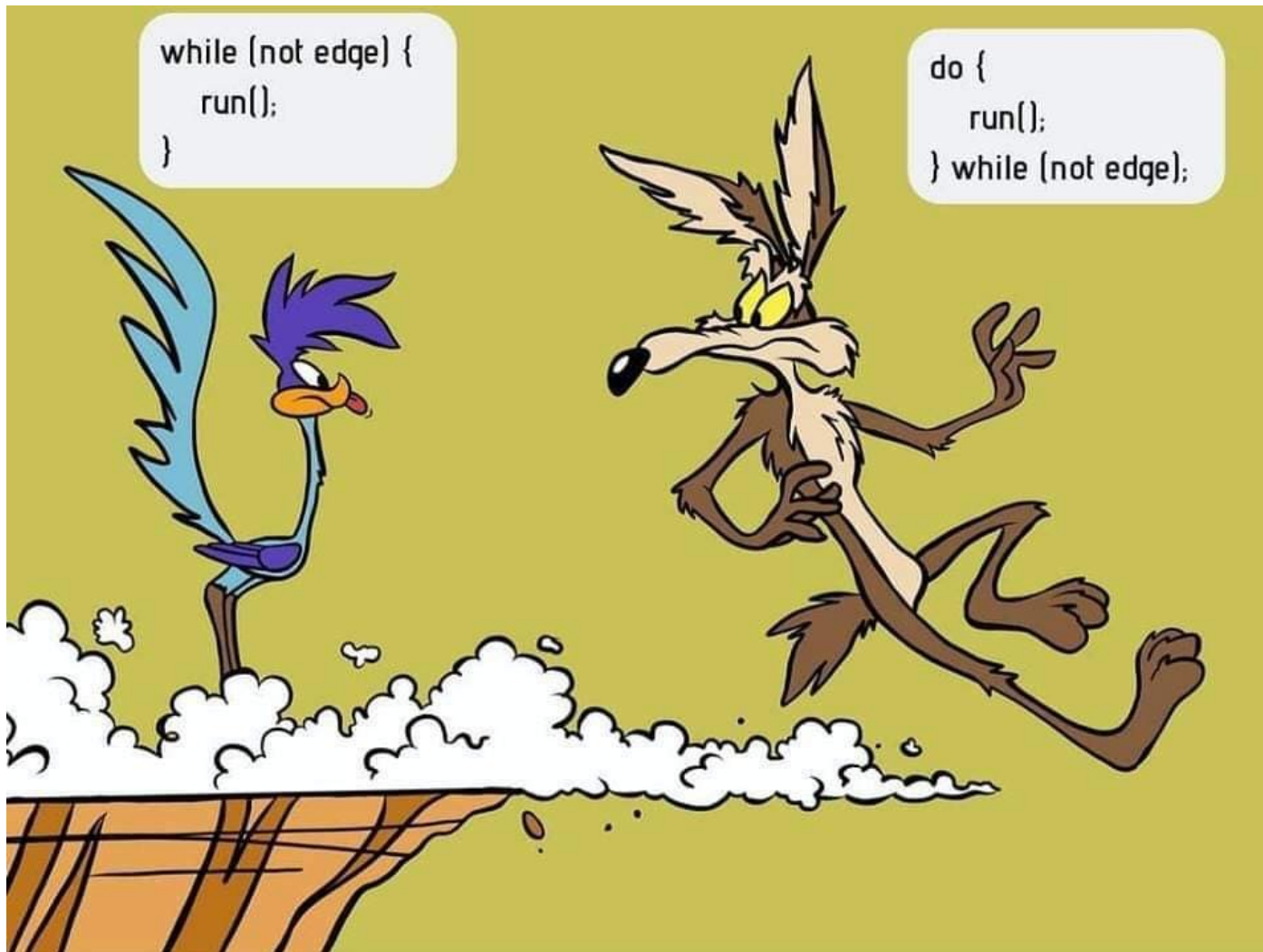
```
switch(<valor_teste>) {  
    case <valor_caso1> :  
        <instruções_caso1>  
        break;  
    case <valor_caso2> :  
        <instruções_caso2>  
        break;  
    default:  
        <instruções>  
}
```

- Exemplo

```
char op='E';  
switch(op) {  
    case 'N':  
        System.out.println("Novo");  
        break;  
    case 'E':  
        System.out.println("Excluir");  
    case 'A':  
        System.out.println("Alterar");  
        break;  
    default:  
        System.out.println("Erro");  
        break;  
}
```



# Estruturas de Repetição while/do-while



# Estruturas de Repetição while/do-while

- Sintaxe: `while(<condição>) <instruções>;`

```
int x=0;
while( x< 100) {
    if((x % 3)==0) System.out.println(x+" é múltiplo");
    x++;
}
```

- Sintaxe: `do <instrs.> while(<condição>);`

```
int x=0;
do {
    if((x % 3)==0) System.out.println(x+" é múltiplo");
} while( ++x < 100);
```

# Estruturas de Repetição for

- Sintaxe:

for(<inicialização>;<condição>;<incremento>)  
<instruções>

```
for( int i=0 ; i<10 ; i++ ) {
```

```
    /* atenção: i é local ao bloco.
```

```
    Ao terminar o for, i deixa de existir */
```

```
    System.out.println("Numero: "+i);
```

```
}
```

# Estruturas de Repetição

## break/continue

- **break**: finaliza incondicionalmente a execução de um bloco de repetição ou switch, mesmo que a condição ainda seja verdadeira.
- **continue**: força a avaliação da expressão contida na condição da estrutura de repetição.

```
int x=0;
while(x<150) {
    if(x==10) {
        x+=10;
        continue;
    }
    if( x == 100 ) break;
    else x++;
}
```

# Vetor (1)

- Declaração: **<tipo>[] <nome>;**

incorreto

```
int[] vetor = new int[10]; // int[10] x;
vetor[5]=4; // coloca valor no 6o.elemento
/* define e inicializa valores */
float[] nota={7.8F,8.4F,4.2F,1.8F,6.4F};
float[] nota2=new float[]{7.8F,8.4F,4.2F,1.8F,6.4F};
/* vetor multidimensional */
char[][] tela = new char[25][80];
tela[10][35]='a';
double[][][] jogo=new double[10][10][20];
int ind[]; // declara, também aceito, mas fora padrão
ind = new int[10]; // aloca espaço 10 ints.para ind
```

# Vetor (2)

```
/* define e inicializa */  
long fibonacci[]={1,2,3,5,8,13,34,55,89,144};  
float seno[]={0.000F,0.500F,0.866F,1.000F,0.500F};  
int A[][]={ {1,2,3}, {0,1,3}, {0,0,-1} };  
  
short []nota;  
nota = new short[55];  
nota[55]=4; // lança ArrayIndexOutOfBoundsException  
  
// O vetor é um objeto em Java  
System.out.println( nota.length ); // exibe 55
```



# Complementação

- Complementação
  - E/S usando classe Console
  - Diálogos modais do Swing (showMessageDialog e showInputDialog do javax.swing.JOptionPane)
  - Funções de Conversão String-Números
  - Fazer exercício EntradaConsole da parte 1
- Mão na Massa
  - Compilar por linha de comando
  - Ambiente IDE NetBeans ou Eclipse



# Atividades Extraclasses

- Questionário 'Desvios'
- Questionário 'Laços'
- Questionário 'Vetores'
- Questionário 'Console'
- T1 : Trabalhos extraclasses grupo Console



# Humor



# Exercícios Suplementares

- Exercício Suplementar ConsoleInput