



Introdução ao Java

Prof. Ms. Peter Jandl Junior
Prof. Ms. Têlvio Orru
Prof. Nathan Silva
J12B

Linguagem de Programação Orientada a Objetos
Ciência da Computação - UNIP – Jundiaí

|| Linguagem Java :: conceitos básicos

Esta apresentação mostra as características da plataforma de programação Java e os recursos básicos disponíveis na linguagem de mesmo nome.

História e ideias

Ambiente de programação

Aplicações

Sintaxe



JAVA: HISTÓRIA

O que é Java?

É realmente uma ilha localizada no Oceano Pacífico, que pertence à Indonésia, onde fica a cidade de Jakarta.

A região produz uma variedade de café conhecida como Java.



|| O que é Java?



- Uma plataforma de desenvolvimento de software,
- que contém uma linguagem de mesmo nome e
- uma extensa e flexível API.

Um pouco de história



- Codinome: **Green**
- Onde: **Sun Microsystems**
- Quando: 1991
- Quem: James Gosling, Patrick Naughton e Mike Sheridan
- O que: desenvolver uma nova plataforma de equipamentos portáteis inteligentes
- Por que: o futuro (da Sun)

Um pouco de história



- C/C++ se mostravam inadequados.
- Gosling decidiu por uma nova linguagem de programação baseada em C e C++: **Oak**.
- Equipe desenvolve novo hardware e um mini sistema operacional para dar suporte ao Oak.

Um pouco de história



- 07/Setembro/1992
*7 (Star Seven) está pronto!
- O que era o *7?
- Segundo James Gosling:
"In Classic Sun Form, Everything..."

Um pouco de história

A new SPARC based, handheld wireless PDA, with a 5" touchscreen color LCD with input, a new 16 bit color hardware double buffered NTSC framebuffer, 900MHz wireless networking, PCMCIA bus, multi-media audio codec, a new power supply/battery interface, radical industrial design and packaging/process technology, a version of Unix that runs in under a megabyte, including drivers for PCMCIA, radio networking, touchscreen, display, flash RAM file system, execute-in-place, split I/D cache, with cached framebuffer support, **a new small, safe, secure, distributed, robust, interpreted, garbage collected, multi-threaded, architecture neutral, high performance, dynamic programming language**, a new small, fast, true-color alpha channel compositing, sprite graphics library, a set of classes that implement a spatial user interface metaphor, a user interface methodology which uses animation, audio, spatial cues, gestures, agency, color, and fun, a set of applications which show all of the features of the *7 hardware and software combination, including a TV guide, a fully functioning television remote control, a ShowMe style distributed whiteboard which allows active objects to be transmitted over a wireless network, and an on-screen agent which makes the whole experience fun and engaging.

Um pouco de história



- Forte crescimento da WWW inspirou outro uso: substituir programas **CGI**.
- Desenvolveu-se o *browser* **WebRunner**, que executava código Oak através da Web.
- Em 1995, no **Sun World'95**, apresentou-se oficialmente o **Java** e *browser* **HotJava**.

Um pouco de história

- ❑ Lançamento provoca grande movimentação no mercado.
- ❑ Poucos meses depois a **Netscape** incorpora capacidades semelhantes em seu *browser* **Navigator**.
- ❑ Microsoft lança seu *browser*, o **Internet Explorer**, que também possuía suporte ao **Java**.



Um pouco de história



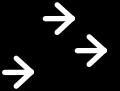
Java

ORACLE

- 26 anos de existência
- 15 versões
- Linguagem de programação mais usada da atualidade
- Milhões de *downloads* do JDK
- 3+ bilhões de dispositivos rodando Java!

Um pouco de história

Desde seu lançamento, a plataforma tem evoluído continuamente, tanto por meio do aprimoramento da linguagem, ampliação da sua API, incorporação de novas tecnologias e operação em novos ambientes computacionais.





JAVA: IDEIAS



Ideias do Java

- Linguagem puramente orientada a objetos.
- Sintaxe simples, semelhante a C++, mas de elevada consistência.
- Verificação forte de tipos.
- Independente de plataforma: é interpretada (bytecodes).
- Desempenho adequado: pode ser compilada no instante da execução.
- Dinâmica e extensível.



Ideias do Java

- Permite desenvolver aplicações distribuídas.
- Suporta múltiplas linhas de execução (multithreaded).
- Projetada com fortes mecanismos de segurança.
- Ausência de ponteiros (não permite manipular endereços de memória).
- Coleta automática de “lixo” (automatic garbage collection).



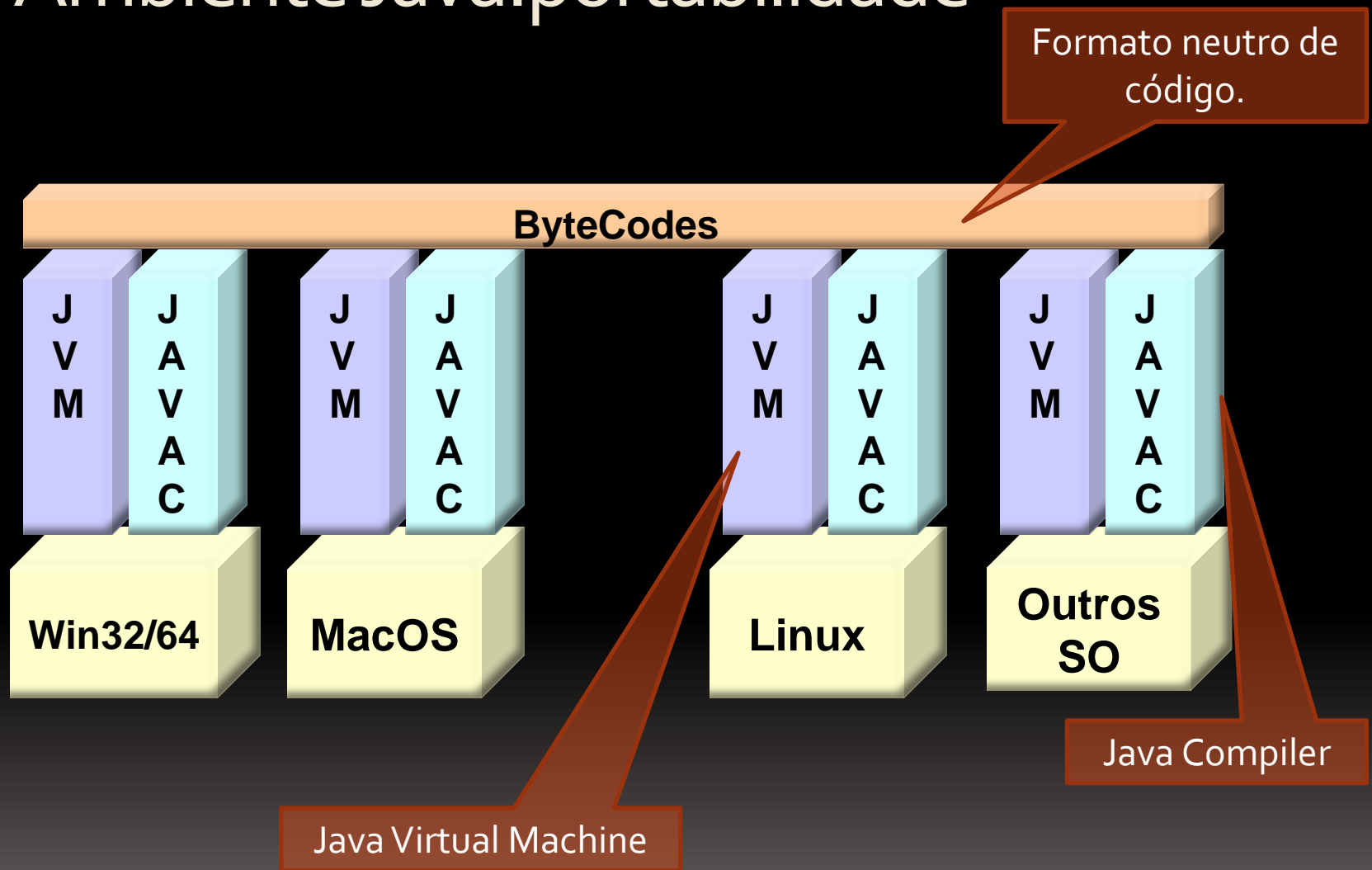
Ideias do Java

- Suporta definição de Interfaces.
- Não oferece herança múltipla (multiple inheritance).
- Oferece os genéricos (semelhante aos templates).
- Acesso padronizado para bancos de dados (JDBC - Java DataBase Connectivity).
- Permite integração com outras linguagens (JNI - Java Native Interface).
- Oferece integração com CORBA (Common Object Request Broker Architecture).



JAVA: AMBIENTE DE PROGRAMAÇÃO

Ambiente Java:portabilidade



Ambiente Java::ciclo simplificado

Editor

- Texto simples ASCII
- Arquivos com extensão *.java*

Compilador

- Transforma programa em *bytecodes*
- Gera arquivos com extensão *.class*

Máquina Virtual

- Carrega e verifica código
- Transforma em código nativo e executa

Ambiente Java::componentes

- **JavaRE** (Java Runtime Environment) é o ambiente de execução padrão da plataforma.
- Versão Atual: JRE 15
- Conjunto mínimo requerido para execução de aplicações Java, constituído de:
 - máquina virtual para ambiente selecionado
 - API (Application Programming Interface)
- Permite execução de quaisquer programas desenvolvidos para plataforma Standard Edition.

Ambiente Java::componentes

- **JavaSDK** (Java Standard Edition Software Development Kit) é o ambiente padrão.
- Versão Atual: JDK 15
- Conjunto de ferramentas para desenvolvimento de programas que inclui:
 - javac (compilador java);
 - java (interpretador e ambiente runtime - JVM);
 - E outras ferramentas.
- Não inclui ambiente visual para desenvolvimento.
- IDE consagrados: Projeto Eclipse & Oracle Netbeans.

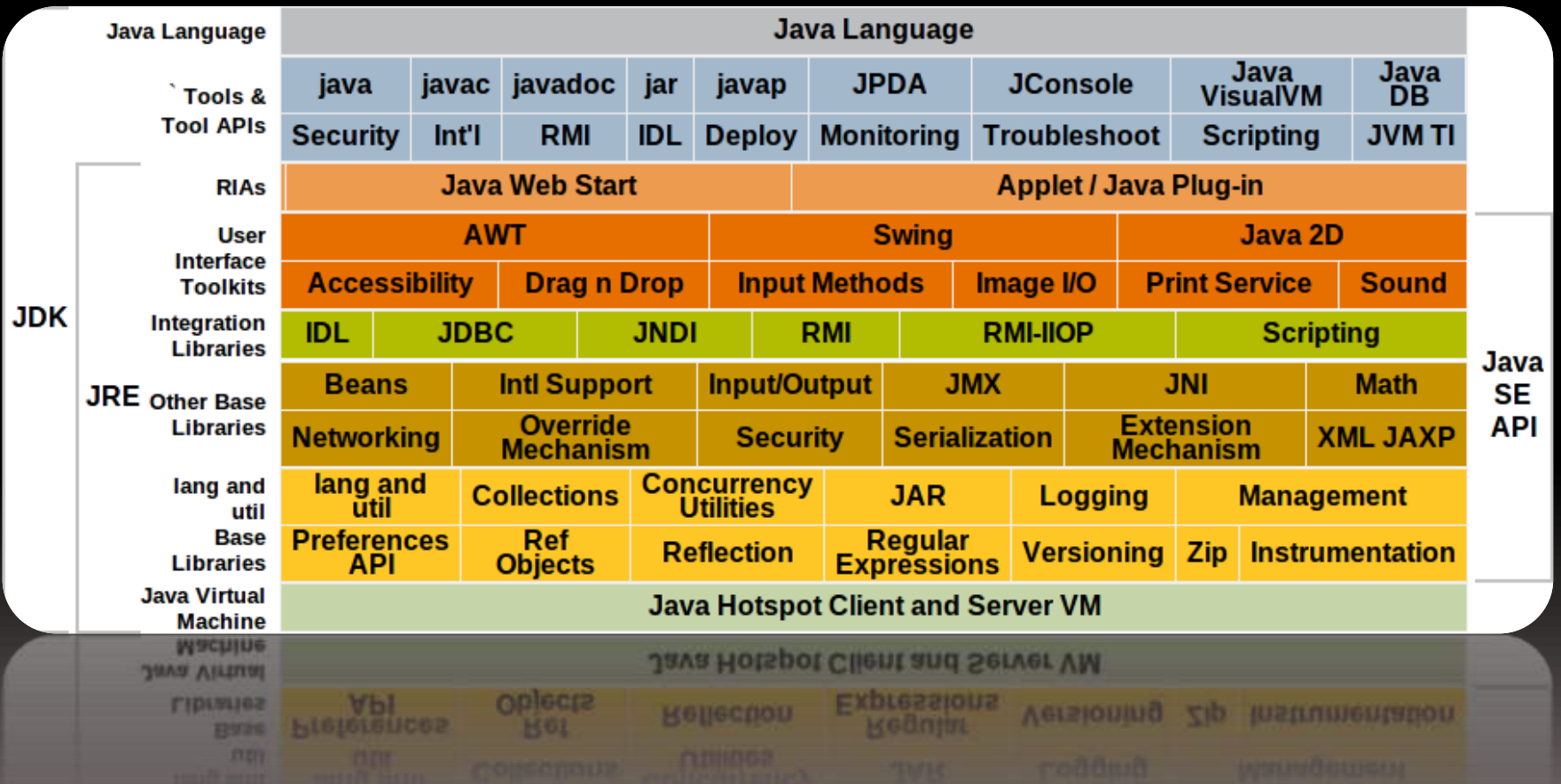


Programas Java

- São semelhantes a programas Windows ou Unix. Possuem as mesmas características e funcionalidades.
- Para executar um programa Java (um arquivo .class) é necessário uma JVM.
- Podem ser simples como aplicações de console, possuir GUI sofisticadas, usar BDs, efetuar comunicação em rede, etc.

API

Application Programming Interface

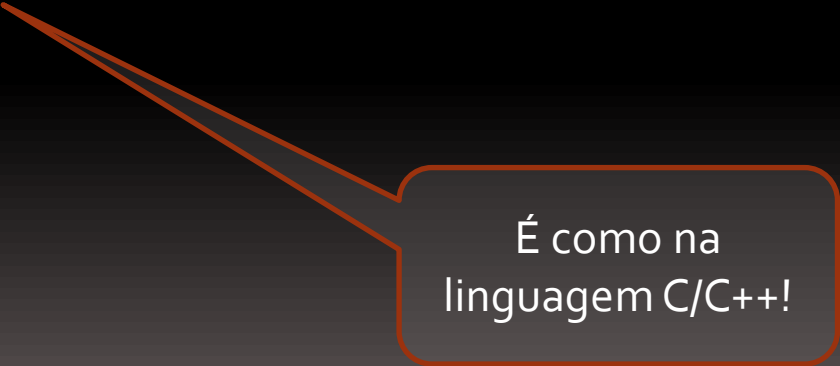


Estrutura dos Programas

- Programas Java são compostos de:
 - ▣ Uma declaração de pacote;
 - ▣ Uma ou mais diretivas de importação;
 - ▣ Uma ou mais declaração de classes (apenas uma pode ser pública);
 - ▣ Uma ou mais declaração de interfaces (apenas uma pode ser pública).
- Os arquivos fonte, de extensão *.java*, **devem** possuir o mesmo nome do elemento público (classe ou interface).

Programa Mínimo

- Composto de:
 - Uma classe contendo o método:
 - `public static void main(String [])`
 - que é o início convencional dos programas.



É como na
linguagem C/C++!

Programa-Exemplo

```
// declaração da classe
public class ED {
    // declaração do início
    public static void main (String a[]){
        // código
        for(int i=0; i<10; i++) {
            System.out.println("Java!");
        }
    }
}
```

Não existe código
fora da classe!

Programas sempre
tem um main().

No main() é disposto
o código do
programa ou seu
início.

Como Editar um Programa Java?

- Utilizando qualquer editor que salve arquivos de texto sem formatação.
- Ideal que seja capaz de:
 - tratar nomes longos;
 - exibir contagem de linhas;
 - fazer destaque de sintaxe; e
 - possibilitar a criação de macros.
- Melhor é usar um IDE (*Integrated Development Environment*) para Java, como o Eclipse!

Como Compilar um Programa Java?

- Utilizando um kit de desenvolvimento padrão:
- Deve-se dispor dos arquivos fonte (*.java)
 - ▣ `javac NomeDaClasse.java`
 - ▣ Atenção ao NomeDaClasse e a extensão .java!



Como Executar um Programa Java?

- Considerando um kit de desenvolvimento padrão.
- Deve-se dispor dos arquivos compilados (*.class)
 - ▣ Acionar arquivo principal da aplicação:
 - ▣ `java NomeDaClasse`
 - ▣ Na execução não se fornece a extensão .class!



JAVA: APLICAÇÕES

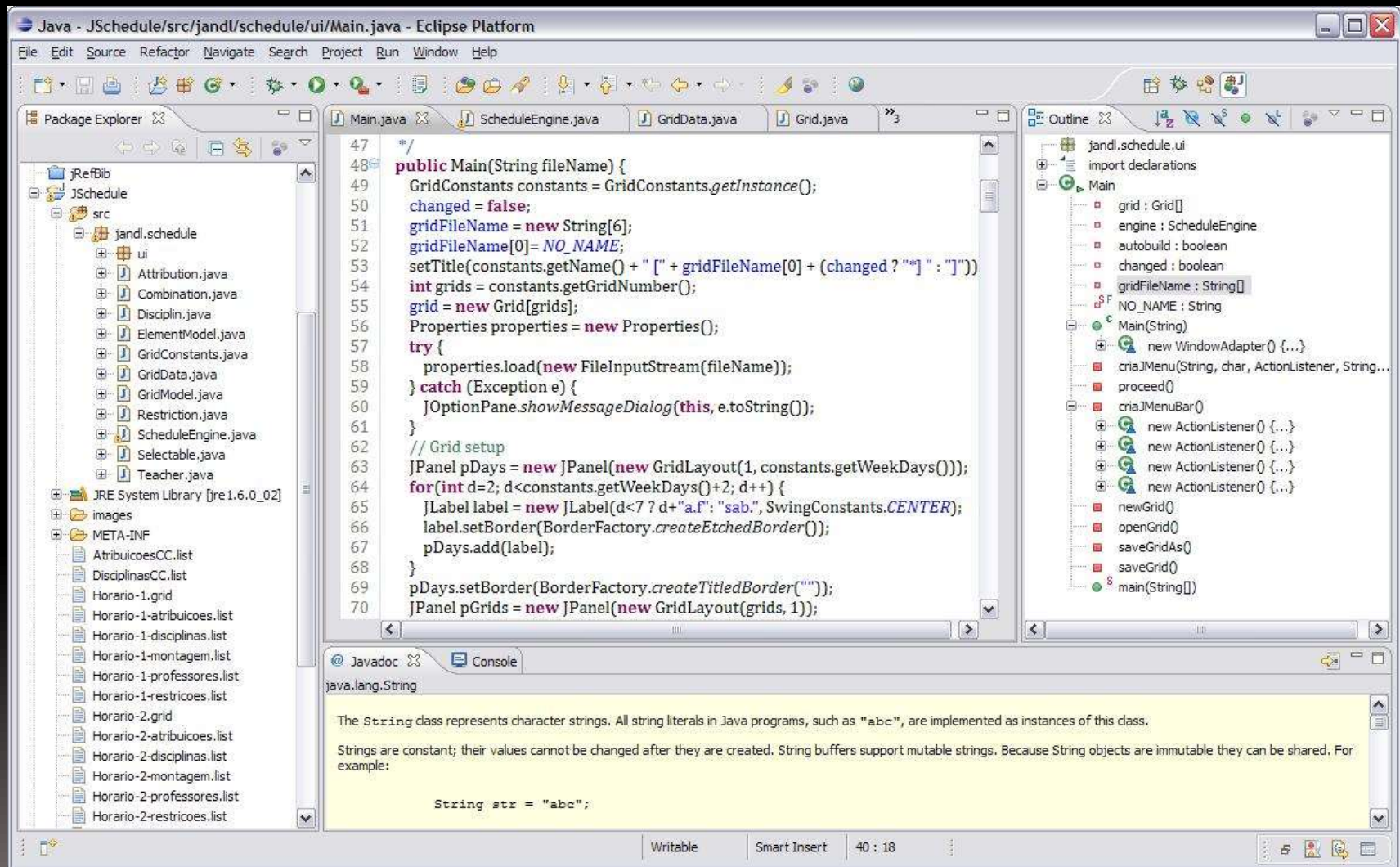
Uma aplicação



LibreOffice
The Document Foundation



Outra aplicação



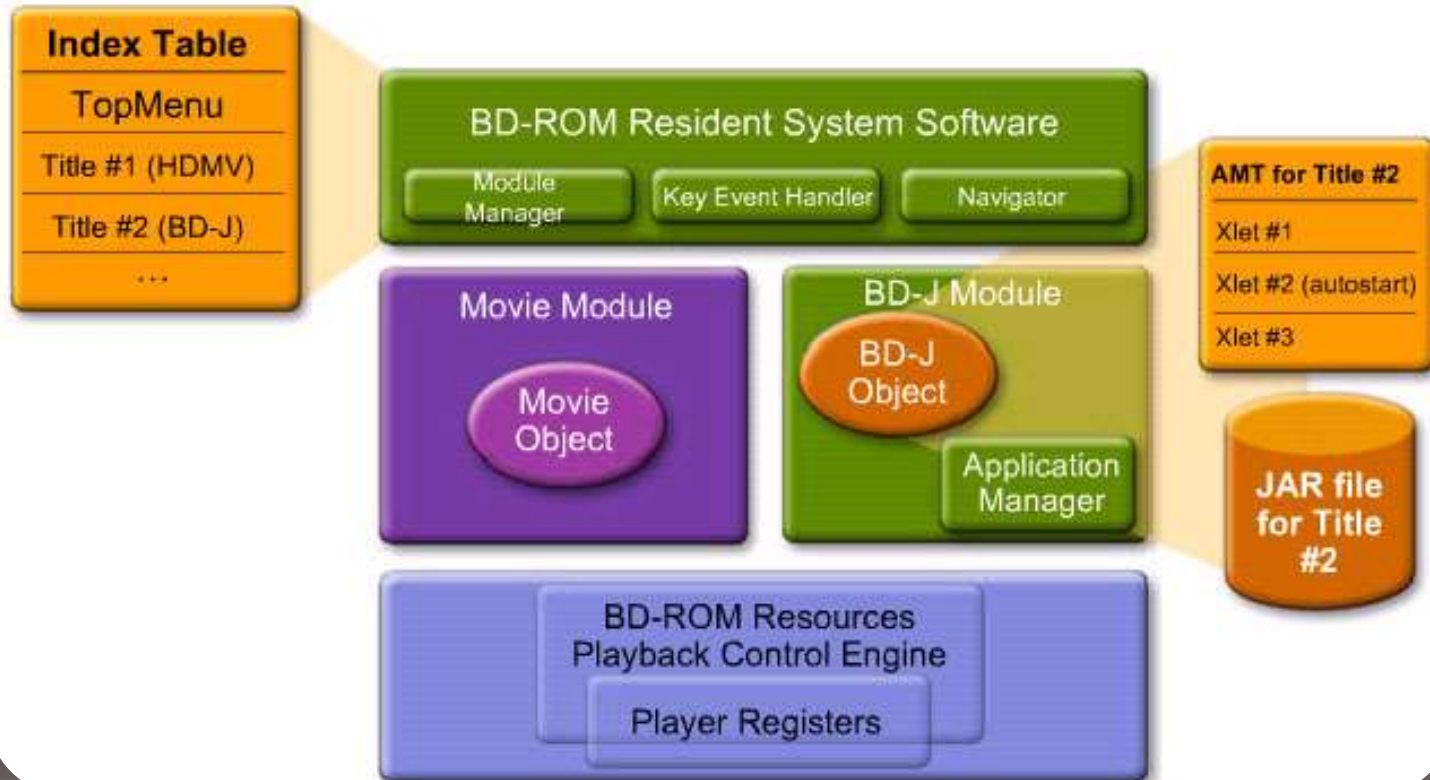
E uma outra aplicação



E mais outra aplicação



BD-ROM Application Layer Structure



E temos muitas mais!







JAVA:SINTAXE

Elementos básicos



- Tipos Primitivos
- Variáveis
- Comentários
- Entrada e Saída
- Operadores e Precedência
- Estruturas de Controle
- *Arrays*
- Controle de Erros

Tipos Primitivos

- Inteiros

- ▣ byte
- ▣ short
- ▣ int
- ▣ long

- O tipo inteiro preferencial é **int**, enquanto o real preferencial é **double**.

- Ponto Flutuante

- ▣ float
- ▣ double

- Caractere

- ▣ char

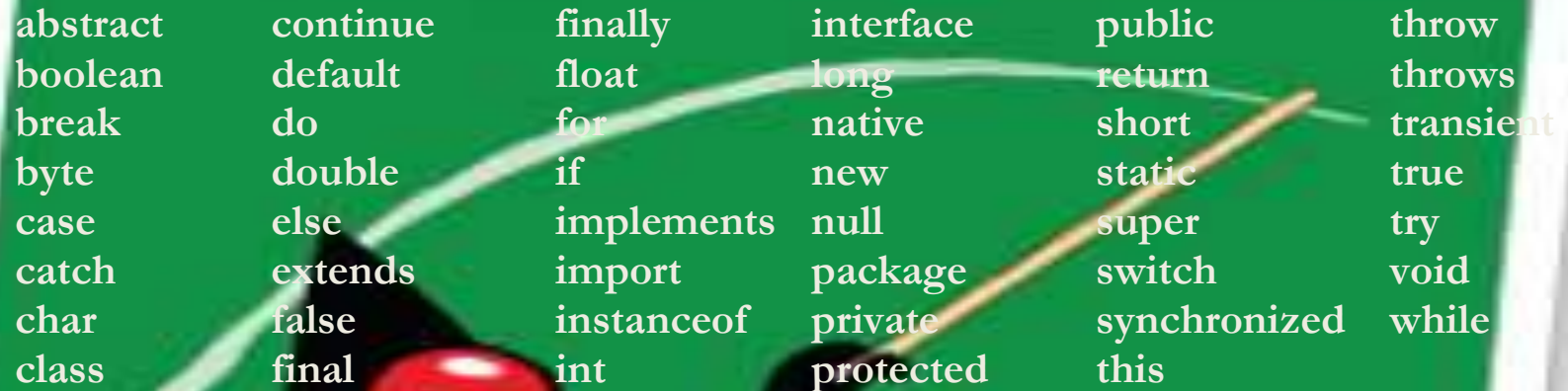
- Lógico

- ▣ boolean

Declaração de Variáveis

- Sintaxe:
- Tipo nome1 [, nome2 [, nome3 [..., nomeN]]];
- Exemplos:
 - ▣ `int i;`
 - ▣ `float total, preco;`
 - ▣ `byte mediaGrupoTarefa2;`
 - ▣ `double valorMedio;`

Palavras Reservadas



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

Das 47 (quarenta e sete), 18 (dezoito) são idênticas a linguagem C e outras 16 (dezesesseis) são como em C++!

Comentários

- `//` comentário de uma linha
- `/*` comentário
 - de múltiplas linhas `*/`
- `/**` comentário de documentação
 - `*` que também pode
 - `*` ter múltiplas linhas
 - `*/`

Método main

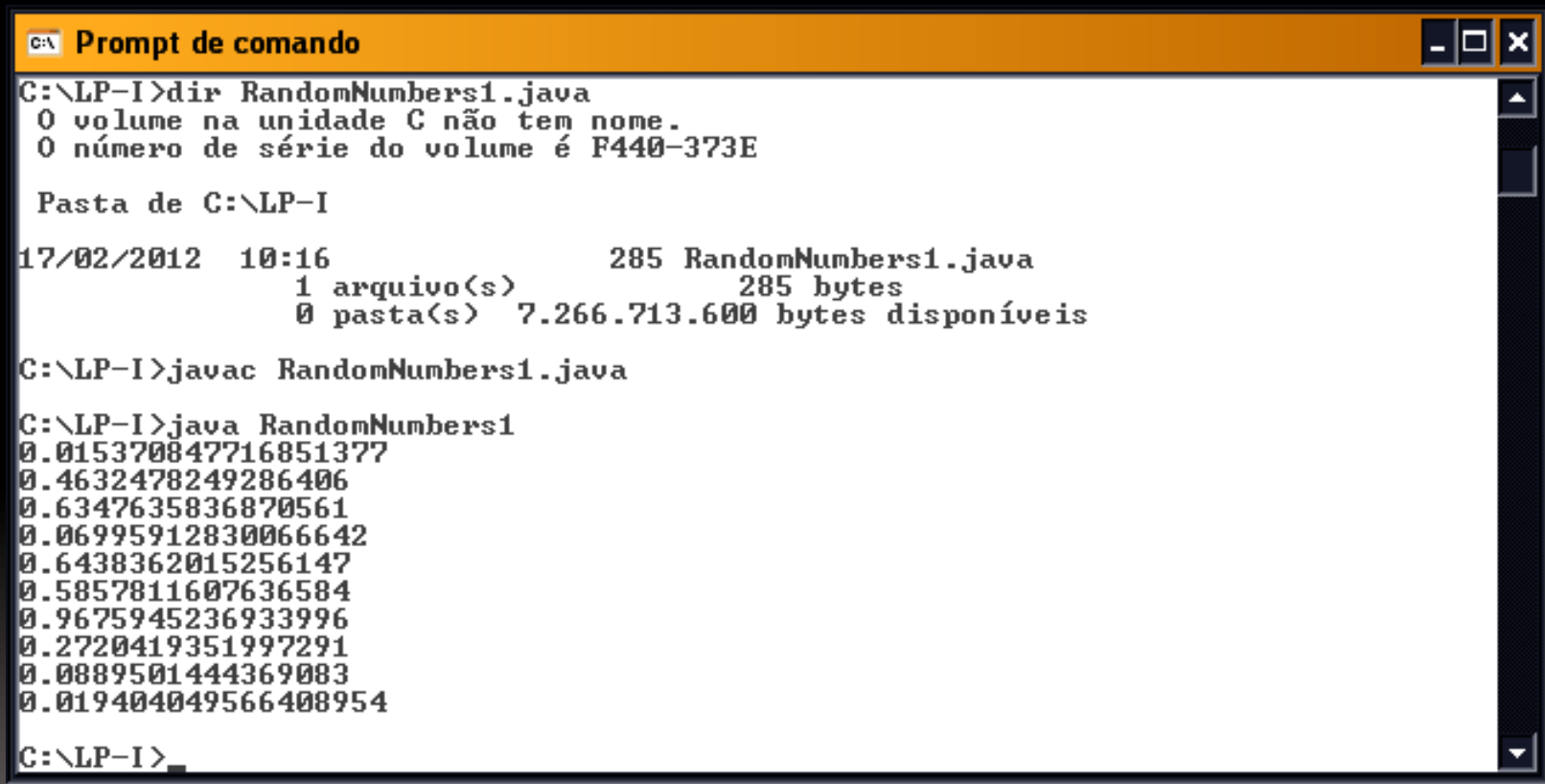
- É o primeiro a ser executado em qualquer aplicação.
- Tem obrigatoriamente:
 - acesso: `public`
 - tipo: `static`
 - retorno: `void`
 - nome: `main`
 - argumentos: `String args[]`

Um exemplo

```
// RandomNumbers.java
public class RandomNumbers {

    public static void main (String args[]) {
        // laço executado 10 vezes
        for (int i=0; i<10; i++) {
            // a cada iteração é impresso um número aleatório
            System.out.println("" + Math.random());
        }
    }
}
```

Um exemplo compilação e execução no console



```
C:\> Prompt de comando

C:\LP-I>dir RandomNumbers1.java
O volume na unidade C não tem nome.
O número de série do volume é F440-373E

Pasta de C:\LP-I

17/02/2012  10:16                285 RandomNumbers1.java
               1 arquivo(s)                285 bytes
               0 pasta(s) 7.266.713.600 bytes disponíveis

C:\LP-I>javac RandomNumbers1.java

C:\LP-I>java RandomNumbers1
0.015370847716851377
0.4632478249286406
0.6347635836870561
0.06995912830066642
0.6438362015256147
0.5857811607636584
0.9675945236933996
0.2720419351997291
0.0889501444369083
0.019404049566408954

C:\LP-I>
```



Métodos

- São subprogramas, i.e., trechos de código que executam tarefas específicas.
- Podem receber UM ou MAIS parâmetros.
- Podem retornar UM resultado (de tipo específico) ou não retorna resultado (void). O retorno de tipo específico exige uso da diretiva *return*.
- Representam técnica de modularização de programas.

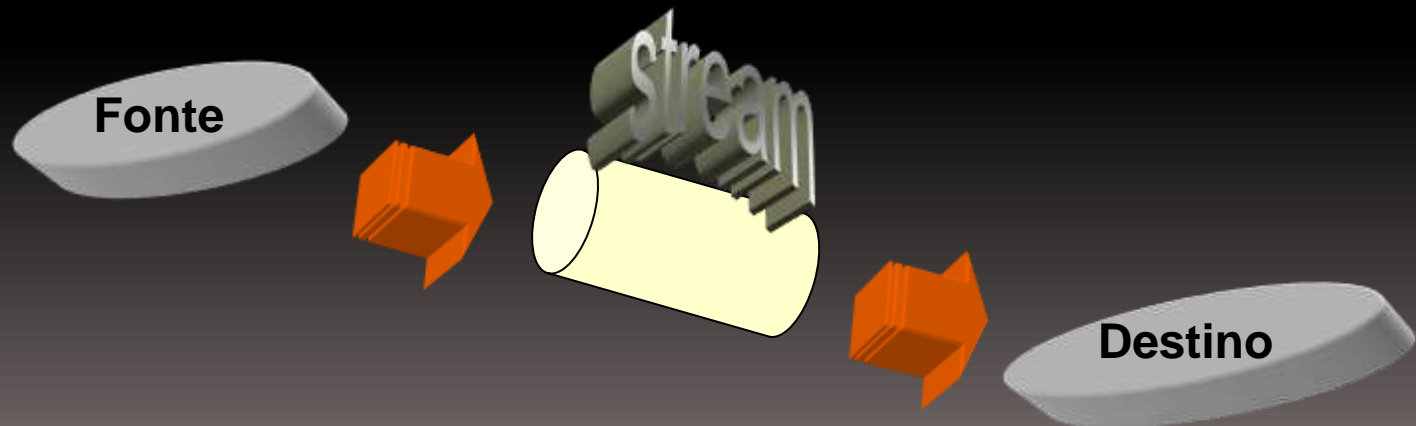
Saída de Dados

- A saída padrão de dados (default) é o próprio console.
- O acesso ao console se faz através da classe `java.lang.System` que nos oferece o objeto `out`.
- `out` é uma stream de dados que leva dados da aplicação para o console (i.e. “imprime na tela”).

Streams

(uma rápida definição)

- São dutos de dados que ligam uma origem (uma fonte) a um destino (um consumidor).
- O fluxo é sempre unidirecional.
- Podem conectar memória, arquivos e dispositivos entre si.



System.out

- Objeto do tipo `java.io.PrintStream`.
- Métodos importantes:
 - `print(argumento);`
 - `println(argumento);`
 - `printf("formato", argumentos);` // ling C
- onde argumento pode ser: inteiro, String, real, char etc.

System.in

- Objeto do tipo `java.io.InputStream`.
- Métodos importantes:
 - `available();`
 - `read();`
 - `skip(int);`
- Suporte oferecido por este objeto é precário:
 - Leitura é orientada a byte.
 - Exige tratamento específico para ler valores inteiros, reais e Strings.
- É sugerido o uso da classe `java.util.Scanner`.

java.util.Scanner

- Nova opção (a partir da versão 5).
- Realiza entrada simples e eficiente.
- Método next() ou nextLine() lêem String (palavra ou linha).
- Métodos nextByte(), nextInt(), nextLong(), nextFloat(), nextDouble(), nextShort() fazem o mesmo para respectivos tipos.

java.util.Scanner

- A declaração e criação de um objeto Scanner é feita assim:

```
Scanner sc = new Scanner(System.in);
```

- A leitura de um valor inteiro é feita com:

```
int i = sc.nextInt();           // preferencial
```

```
long l = sc.nextLong();
```

- A leitura de um valor real é feita com:

```
float f = sc.nextFloat();
```

```
double d = sc.nextDouble();    // preferencial
```

Operadores

- Aritméticos

- $+$, $-$, $*$, $/$ (aritmética simples)
- $\%$ (resto da divisão inteira)
- $-$ e $+$ (sinal)
- $++$ (incremento)
- $--$ (decremento)

- Atribuição

- $=$

Operadores

```
public class Aritmetica {  
    static public void main (String args[]) {  
        int a = 5, b = 2; // Decl de 2 variaveis  
        // Exemplos de operacoes sobre variaveis  
        System.out.println("a = " + a);  
        System.out.println("-b = " + (-b));  
        System.out.println("a + b = " + (a + b));  
        System.out.println("a * b = " + (a * b));  
        System.out.println("a / b = " + (a / b));  
        System.out.println("a % b = " + (a % b));  
        System.out.println("a++ = " + (a++));  
        System.out.println("--b = " + (--b));  
    }  
}
```

Operadores

■ Relacionais

□ >

□ <

□ >=

□ <=

□ ==

□ !=

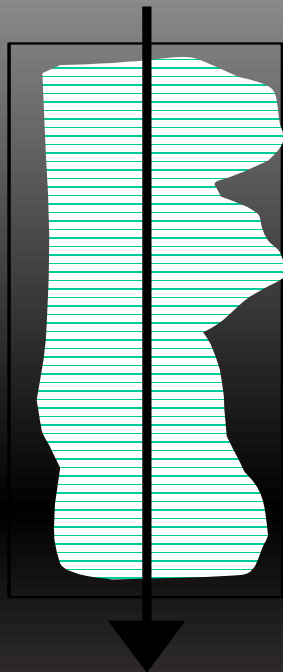
Operadores

```
public class Relacional {  
    static public void main (String args[]) {  
        int a = 15; int b = 12;  
        System.out.println("a = " + a);  
        System.out.println("b = " + b);  
        System.out.println("a == b -> " + (a == b));  
        System.out.println("a != b -> " + (a != b));  
        System.out.println("a < b -> " + (a < b));  
        System.out.println("a > b -> " + (a > b));  
        System.out.println("a <= b -> " + (a <= b));  
        System.out.println("a >= b -> " + (a >= b));  
    }  
}
```

Precedência dos Operadores

- . [] ()
- ++ -- ~ instanceof clone new -
- * / %
- + -
- << >> >>>
- < <= > >=
- == !=
- &
- ^
- |
- &&
- ||
- ?:
- = op=
- ,

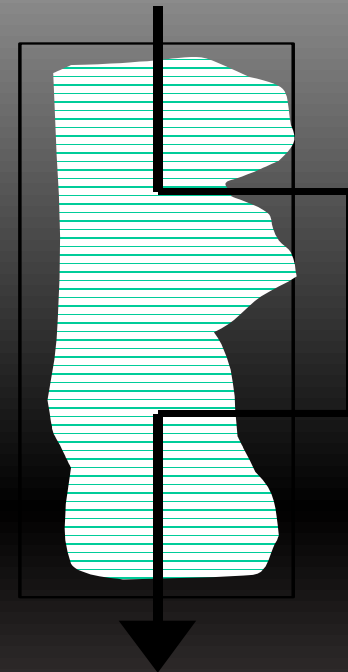
Fluxo de Execução



Fluxo
Sequencial de
Execução



Fluxo
Repetitivo de
Execução



Desvio do
Fluxo de
Execução

■ Diretivas

Isoladas

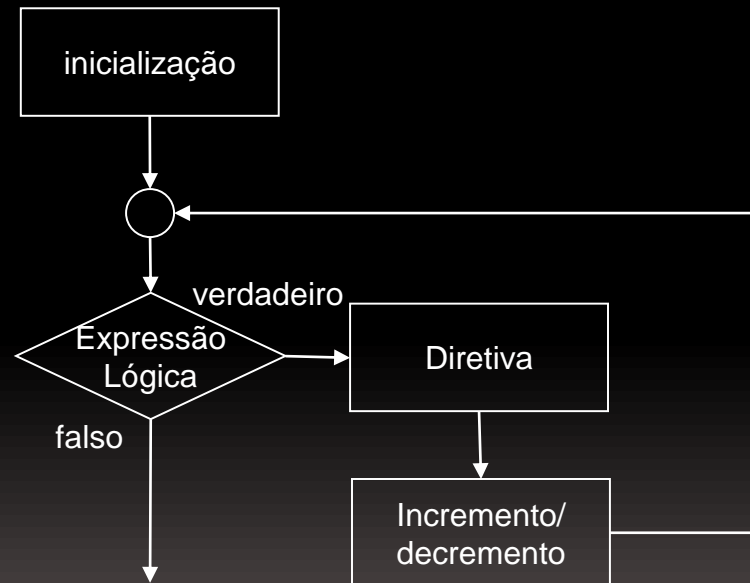
diretiva1;
diretiva2;
diretiva3;
:
diretivaN;

Bloco

```
{  
    diretiva1;  
    diretiva2;  
    diretiva3;  
    :  
    diretivaN;  
}
```

Repetição Simples

- for (inic; condição; incr/decr)
- diretiva;



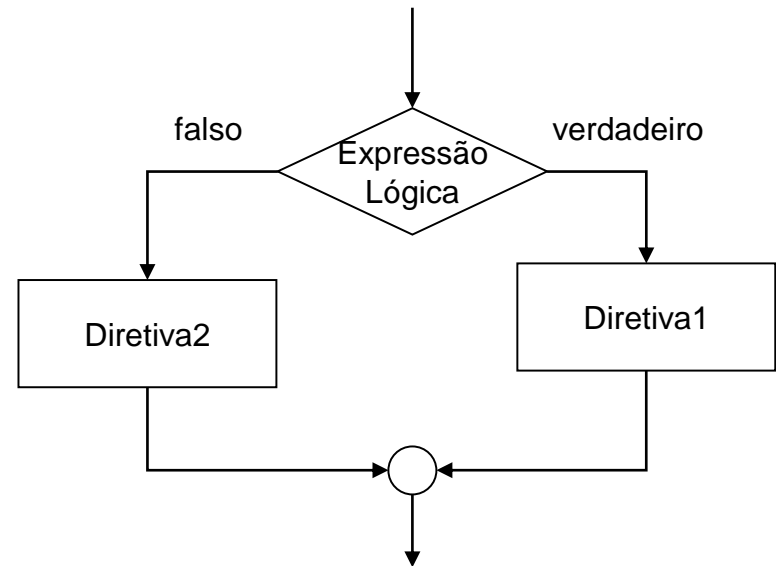
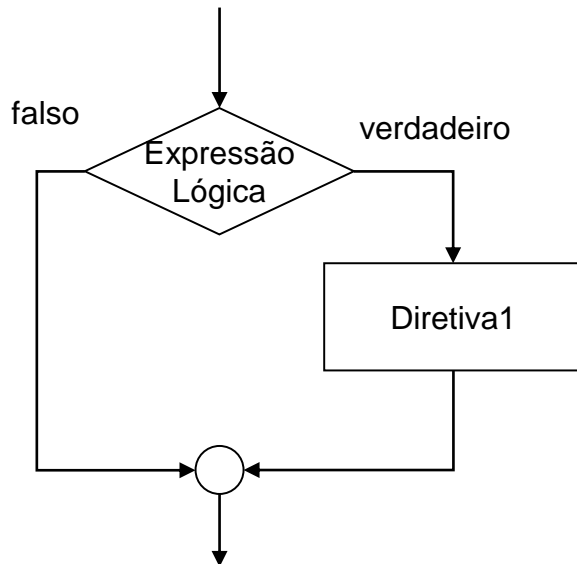


Repetição Simples

```
public class ExemploFor {  
    public static void main (String args[]) {  
        int j;  
        for (j=0; j<10; j++) {  
            System.out.println(""+j);  
        }  
    }  
}
```

Desvio de Fluxo

- if (condição) {
- diretiva1;
- }



- if (condição) {
- diretiva1;
- } else {
- diretiva2;
- }

Desvio de Fluxo

```
public class ExemploIf {  
    public static void main (String args[]) {  
        if (args.length > 0) {  
            for (int j=0;j<Integer.parseInt(args[0]);j++) {  
                System.out.print(" " + j + " ");  
            }  
            System.out.println("\nFim da Contagem");  
        }  
        System.out.println("Fim do Programa");  
    }  
}
```




Repetição Condicional

```
public class ExemploWhile {  
  
    public static void main (String args[]) {  
        int j = 10;  
        while (j > 0) {  
            System.out.println("j="+j);  
            j--;  
        }  
    }  
}
```

Repetição Condicional

```
public class ExemploDoWhile {  
  
    public static void main (String args[]) {  
        int j = 10;  
        do {  
            System.out.println("j="+j);  
            j--;  
        } while (j > 0);  
    }  
}
```

Recomendações de Estudo



- Resolver a Lista I.
- Complementar estudo com:
 - ▣ JANDL JR, Peter.
Java – Guia do Programador, 3ª Ed.
São Paulo: Novatec,
2015.