




CURSO DE PROGRAMAÇÃO EM JAVA

Aula 1 
Apresentação do
Curso e Introdução a
Java

Boa noite!

Yasmin Wassef

Você pode me contatar em:

ywassef@hotmail.com

1. EQUIPE

**Equipe do Programa de Aprendizagem em
Pensamento Computacional**

Programa de Aprendizagem em Pensamento Computacional

INSTRUTORA DO CURSO

Yasmin Wassef

PROFESSORES PARTICIPANTES

Jurandy Gomes de Almeida Júnior

MONITORES

André Lucas Maegima

Kevin Costa Scaccia

Igor Luppi de Oliveira

Ricardo Elizeu Neto

Slides e atividades disponíveis no Google Classroom: [arjvv3w](#)

O curso contará com certificado de participação, o qual pode ser creditado como Atividades Complementares.

2.

CONTEÚDO PROGRAMÁTICO E ESTRUTURA DO CURSO

Conteúdo Programático



Introdução aos Aplicativos Java

- ▷ O que é Java
- ▷ Características da Linguagem
- ▷ Ambiente de desenvolvimento Java típico (editor, compilador, carregador de classe, verificador de bytecode, Java Virtual Machine, Garbage Collection)
- ▷ IDE Eclipse



Introdução a Classes e Objetos

- ▷ Noções de classes, objetos, métodos, variáveis de instância, estado e comportamento
- ▷ Entrada e Saída de Dados
- ▷ Métodos Set e Get



Tipos de Dados

- ▷ Palavras-chave, Identificadores e Literais
- ▷ Tipos de Dados: lógicos, textuais e numéricos
- ▷ Tipos primitivos e tipos por referência
- ▷ Construtores



Operadores

- ▷ Palavras chave e operadores, identificadores
- ▷ Operadores de atribuição
- ▷ Operadores aritméticos, relacionais e lógicos
- ▷ Incremento e decremento



Expressões

- ▷ Aritméticas, Relacionais, Lógicas
- ▷ Precedência de Expressões
- ▷ Conversão de tipos em expressões
- ▷ Abreviação de expressões



Estruturas de Seleção

- ▷ Estruturas de única seleção
- ▷ Estruturas de seleção dupla
- ▷ Estruturas de seleção aninhadas
- ▷ Estrutura de seleção múltipla: switch
- ▷ Operador Ternário

Conteúdo Programático



Estruturas de Repetição

- ▷ Operadores de atribuição compostos
- ▷ Operadores de incremento e decremento
- ▷ A instrução de repetição while



Estruturas de Repetição

- ▷ A estrutura de repetição do...while
- ▷ Comandos de desvio (break, continue...)



Estruturas de Repetição

- ▷ Princípios básicos de repetição controlada por contador
- ▷ A instrução de repetição for



Módulo de Programas em Java

- ▷ Definição: métodos, classes e pacotes
- ▷ Métodos *static*, campos *static* e classe *Math*
- ▷ Declaração e utilização de métodos



Arrays

- ▷ Introdução a arrays
- ▷ Declarando e criando arrays
- ▷ Passando arrays para métodos
- ▷ Arrays multidimensionais



Recursividade

- ▷ Conceitos de recursão
- ▷ Aplicação e exemplos

Conteúdo Programático



Classes e Objetos

- ▷ Estudo de casos
- ▷ Escopo de classes
- ▷ Referenciando membros do objeto atual com a referência `this`
- ▷ Construtores
- ▷ Encapsulamento
- ▷ Acesso ao Pacote



Herança

- ▷ Superclasses e subclasses
- ▷ Membros `protected`
- ▷ Relacionamento entre superclasses e subclasses
- ▷ Construtores em subclasses
- ▷ Classe `Object`



Polimorfismo

- ▷ Conceito e definição
- ▷ Demonstrando um comportamento polimórfico
- ▷ Classes e métodos abstratos
- ▷ Métodos e classes `Final`



Exceções

- ▷ Tratamento de exceções
- ▷ Hierarquia de exceções em Java
- ▷ Exceções encadeadas
- ▷ Declarando novos tipos de exceções



Arquivos e Fluxos

- ▷ Introdução
- ▷ Hierarquia de dados
- ▷ Arquivos e fluxos
- ▷ Classe `File`
- ▷ Arquivos de texto de acesso sequencial
- ▷ Arquivos de texto de acesso aleatório



Coleções

- ▷ Visão Geral
- ▷ Classe `Arrays`
- ▷ Classe `Collections`
- ▷ Listas
- ▷ Algoritmos de coleções
- ▷ Classe `Stack` do pacote `java.util`
- ▷ Classe `PriorityQueue`
- ▷ Conjuntos
- ▷ Mapas
- ▷ Classes `Properties`



Strings

- ▷ Fundamentos de caracteres e Strings
- ▷ Classe `String`
- ▷ Classe `StringBuffer`
- ▷ Classe `Character`
- ▷ Classe `StringTokenizer`
- ▷ Classe `Pattern`
- ▷ Classe `Matcher`

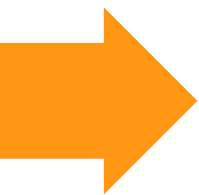
3.

PROGRAMAÇÃO EM JAVA

Por que Java?

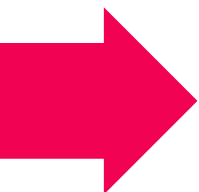


Por que programar?



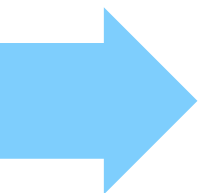
Por que surgiu?

Devido à necessidade de dizer ao computador o que fazer.



Software Prontos

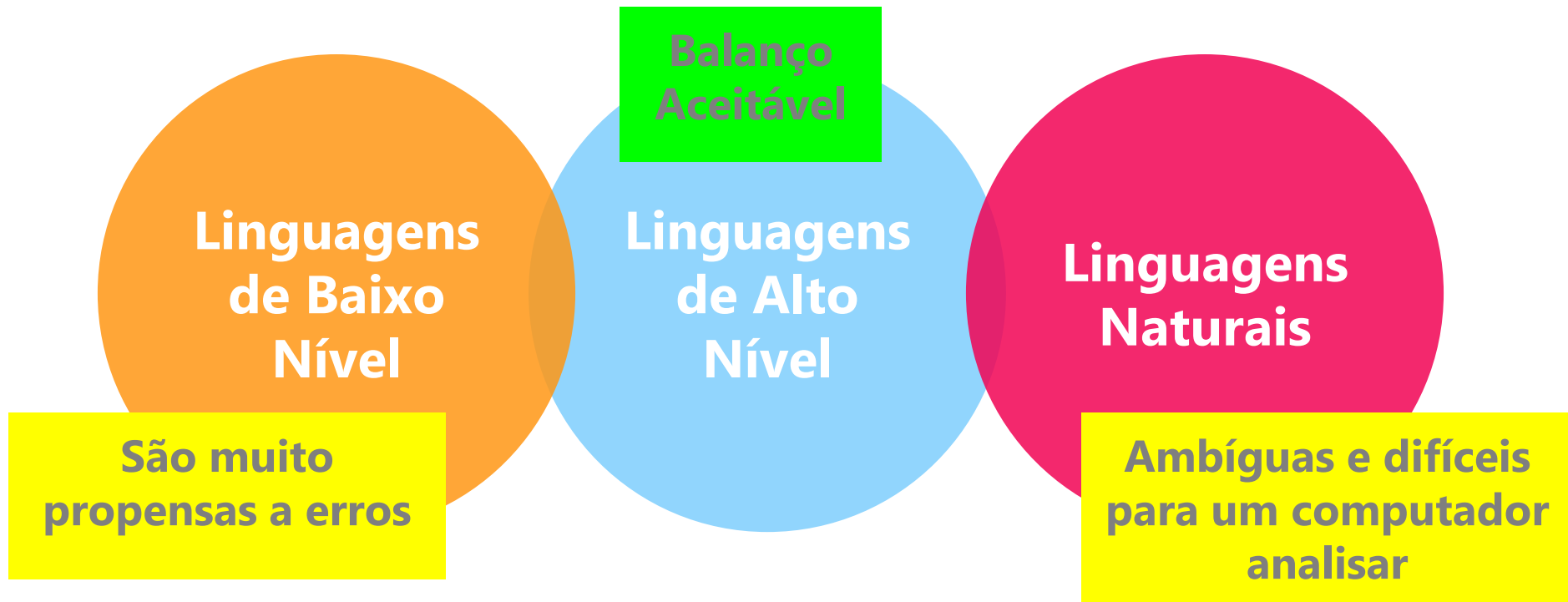
Os programas que possuímos em nossos computadores permitem que façamos exatamente o que queremos.



Programação

Permite que façamos o computador fazer **qualquer coisa** que desejarmos.

Linguagens de Programação







- ▷ Amplamente utilizada
- ▷ Amplamente disponível
- ▷ Compreende um conjunto completo de abstrações modernas
- ▷ Possui uma variedade de verificadores de erros nos programas



- ▷ Linguagem mais utilizada no mundo
- ▷ Não é somente uma linguagem, e sim uma plataforma de desenvolvimento
- ▷ Aprendendo Java é possível construir aplicações para desktop, celular, cartão, web, televisão digital, etc.
- ▷ Os grupos de usuários Java são muito fortes em todo o mundo



▷ Com Java é possível desenvolver em qualquer sistema operacional e para qualquer sistema operacional;

Java é:

- ▷ Uma linguagem de programação
- ▷ Um ambiente de desenvolvimento
- ▷ Um ambiente de aplicação



Objetivos em se desenvolver Java:

- ▷ Criar uma linguagem orientada a objetos
- ▷ Prover velocidade de desenvolvimento e **portabilidade** (execução do código compilado pela JVM)
- ▷ Eliminar exigências de programação que afetam a robustez de um código (ponteiros, alocação de memória)



Objetivos em se desenvolver Java:

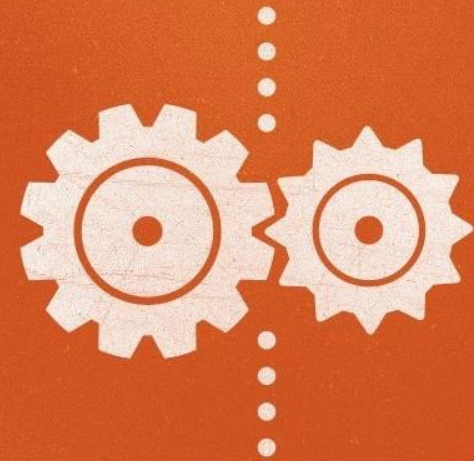
- ▷ Programação multitarefa, mesmo em sistemas operacionais que não deem suporte nativo a Threads
- ▷ Permitir que um programa seja dinamicamente modificado através da carga de componentes via redes (Internet)
- ▷ Checagem de integridade de um programa, garantindo a segurança do sistema operacional e de dados durante a sua execução



Principais características de Java

- ▷ **Concisa e simples:** é uma evolução de C++: não suporta aritmética de ponteiros, registros, etc.
- ▷ **Orientada a objetos:** suporta os principais conceitos de orientação a objetos. Favorece extensibilidade e reusabilidade
- ▷ **Provê acesso a Internet/WWW:** contém bibliotecas especiais que possibilitam o trabalho com protocolos TCP/IP como HTTP e FTP. Permite acesso a URLs
- ▷ **Robusta:** fortemente tipada. Programas são confiáveis. Reduz imprevistos em tempo de execução

PROGRAMMER



</CODE>

Principais características de Java

- ▷ **Portável:** aplicações funcionam do mesmo jeito em qualquer ambiente.
- ▷ **Segura:** restrições de acesso a arquivos (applets), manipulação de ponteiros, etc.
- ▷ **Concorrente:** suporta aplicações concorrentes (multithreads e monitores)

PROGRAMMER



</CODE>

3.

AMBIENTE DE DESENVOLVIMENTO

**Editor, Compilador, Carregador de Classe,
Verificador de ByteCode, Java Virtual Machine,
Garbage Collection**

Java o levará a
novas fronteiras...

Em seu humilde lançamento
para o público, Java seduziu
programadores com sua
sintaxe amigável, recursos
orientados a objetos,
gerenciamento de memória
e, o melhor de tudo – a
promessa da **portabilidade**.



Java™

Portabilidade

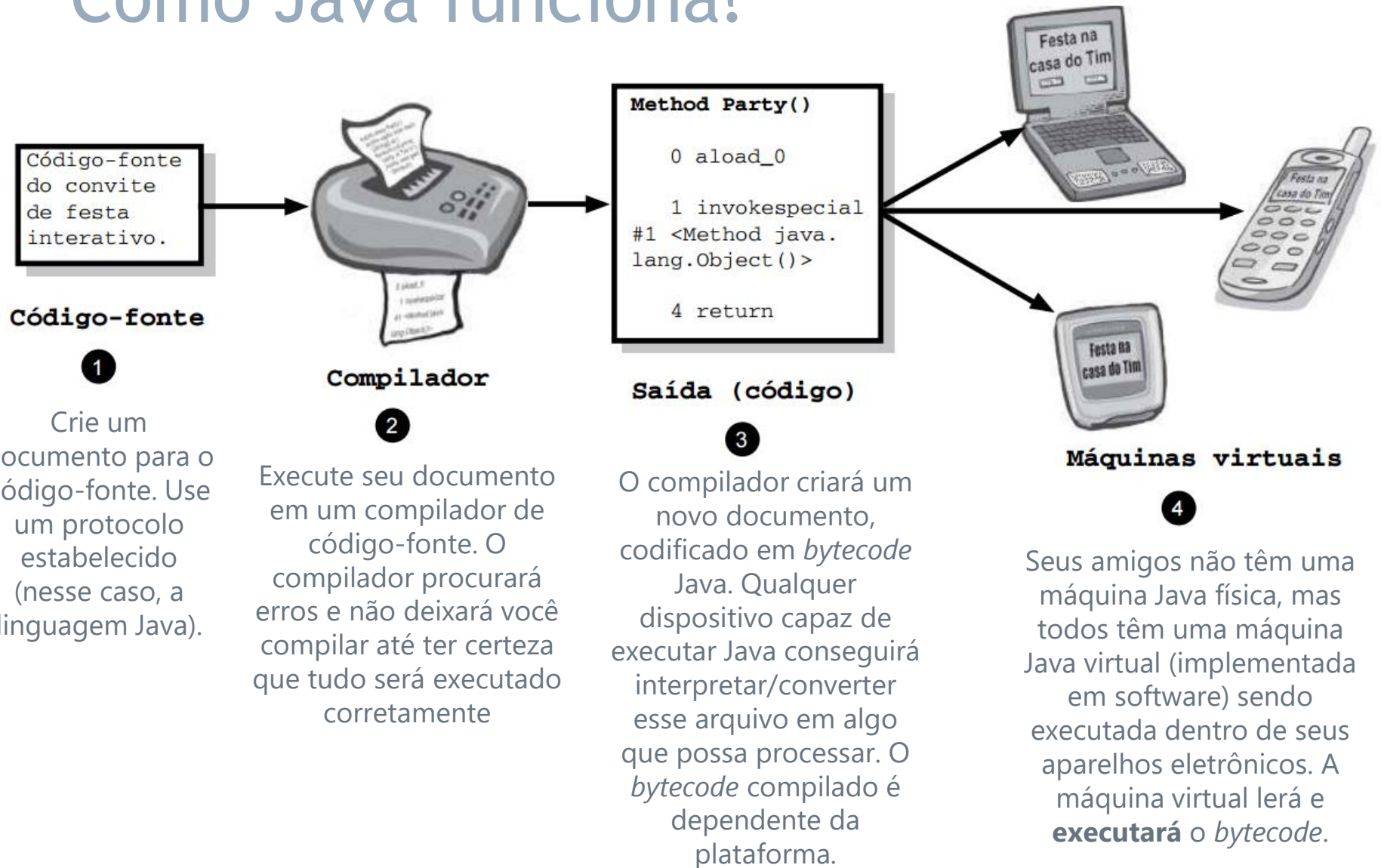


**Idealize seu
programa**

**Execute em
qualquer
lugar**

**Escreva
uma vez**

Como Java funciona?



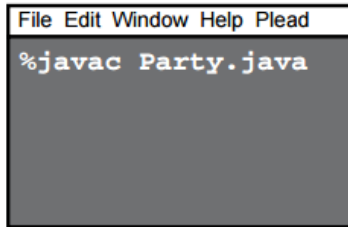
O que você fará em Java?

```
import java.awt.*;
import java.awt.event.*;
class Party {
    public void buildInvite() {
        Frame f = new Frame();
        Label l = new
Label("Party at Tim's");
        Button b = new
Button("You bet");
        Button c = new
Button("Shoot me");
        Panel p = new Panel();
        p.add(l);
    } // mais código aqui...
}
```

Código-fonte

1

Digite seu código-fonte. Salve como: **Party.java**



Compilador

2

Compile o arquivo **Party.java** usando o `javac` (aplicativo do compilador). Se não houver erros, você terá um documento chamado **Party.class**. O arquivo gerado pelo compilador é composto de *bytecodes*.

```
Method Party()
  0 aload_0
  1 invokespecial #1 <Method java.
lang.Object()>
  4 return
Method void buildInvite()
  0 new #2 <Class java.awt.Frame>
  3 dup
  4 invokespecial #3 <Method java.
  7 <init>()V>
  8 awt.Frame()>
```

Saída (código)

3

Código compilado: **Party.class**



Máquinas virtuais

4

Execute o programa iniciando a Java Virtual Machine (JVM) com o arquivo **Party.class**. A JVM converterá o bytecode em algo que a plataforma subjacente entenda e executará seu programa.



E se fosse em C?

Nesse caso, o código deveria ser compilado várias vezes – uma para cada sistema operacional desejado. No caso de Java, o código é compilado apenas uma vez, gerando o bytecode que poderá ser interpretado por qualquer outra máquina virtual Java, rodando em Linux, Windows, ou qualquer outro sistema operacional desejado.

A arquitetura Java

A arquitetura Java é formada pelos seguintes componentes:

- ▷ **Máquina virtual Java** (JVM)
- ▷ **Gerenciador de alocação/liberação de memória** (Garbage Collection)
- ▷ **Sand box** – módulo de segurança de um código (é impossível criar um vírus em Java)





Como funciona a arquitetura Java?



Arquitetura Java

Garbage Collection

Para facilitar a vida dos programadores e evitar os erros comuns associados à alocação de memória, a linguagem Java introduziu um novo conceito: o garbage-collection. Garbage-collection é um mecanismo de controle automático de alocação e liberação de memória.

Máquina Virtual

Possui definições concretas para a implementação dos seguintes itens:

- **Conjunto de instruções**
(equivalentes às instruções da CPU)
- **Conjunto de registradores**
- **Formato padrão de classes**
 - **Pilha de memória**
- **Pilha de objetos coletados pelo garbage-collector**
 - **Área de memória**

Sand Box

Refere-se ao nível de acesso a arquivos do sistema, tais como unidades de disco rígido e à rede. O *sandbox* garante que uma aplicação não confiável e provavelmente mal-intencionada não tenha acesso a recursos do sistema.

Funcionamento JVM

A JVM não permite que um programa Java acesse recursos de hardware diretamente, protegendo o computador de operações perigosas, como acesso às regiões protegidas da memória ou formatação física do disco rígido. Um programa Java só é executado caso passe pela verificação de segurança da JVM, que consiste em dizer que:

- ▷ O programa foi escrito utilizando-se a sintaxe e semântica da linguagem Java
- ▷ Não existem violações de áreas restritas de memória no código



Funcionamento JVM

- ▷ O código não gera **Stack Overflow**
- ▷ Os tipos de parâmetros dos métodos são corretos
- ▷ Não existe nenhuma conversão ilegal entre dados do programa, como a tentativa de conversão de inteiros em ponteiros
- ▷ O acesso a objetos está corretamente declarado

Caso alguma das condições acima não seja satisfeita, a máquina virtual Java causará um erro de execução (*runtime error*).



Funcionamento Garbage Collection

Quando uma variável é declarada em um código de computador, a JVM cria um ponteiro para uma área de memória equivalente ao tamanho do tipo de dado utilizado por essa variável. Quando essa variável é associada a outra região, a JVM coloca o espaço alocado anteriormente em uma pilha de objetos em desuso. Caso o computador fique com pouca memória disponível, a JVM remove objetos dessa pilha, permitindo que esse espaço de memória seja realocado.



4. ECLIPSE

IDE-Eclipse, Primeiro Programa em Java

Primeiro programa em Java

```
/*
```

```
Primeiro programa em Java: HelloWorld.java
```

```
*/
```

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

Obrigado!

Alguma pergunta?

Você pode me contatar em:
ywassef@hotmail.com