Linguagem de Programação Orientada a Objetos (LPOO) Aula 1

Prof. Samuel Ferraz

UFMS

24 de Abril de 2017

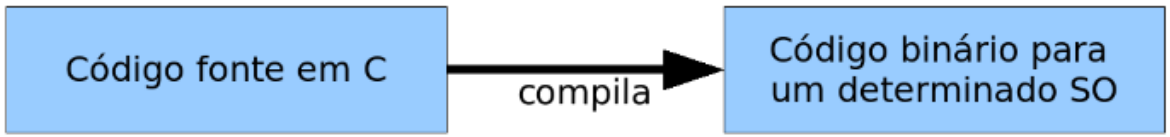
Tópicos

Breve História do Java

- Criado pela antiga *Sun Microsystems*, que foi comprada pela *Oracle*;
- Motivações para sua criação:
  - Uso em pequenos dispositivos;
  - Rodar pequenas aplicações (*applets*) em browser;
- Maior uso da atualidade: no lado do servidor.

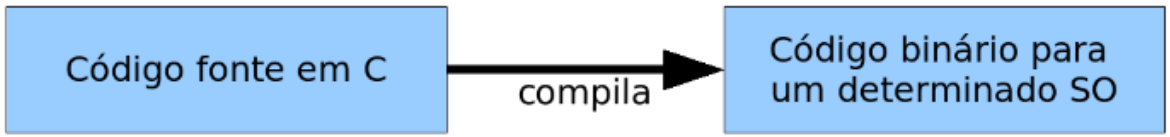
Compilação Clássica

- Compilação na linguagem C:



Compilação Clássica

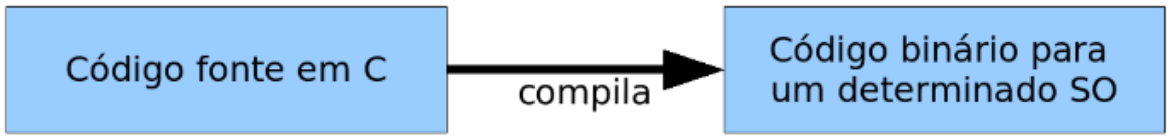
- Compilação na linguagem C:



- O SO alvo deve conseguir entender o código em questão;

Compilação Clássica

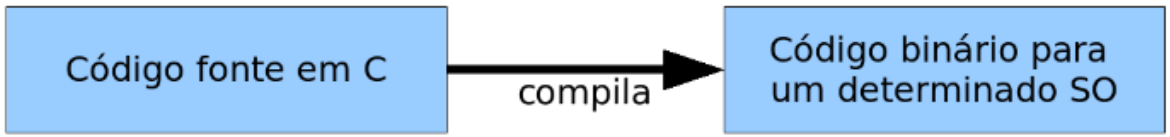
- Compilação na linguagem C:



- O SO alvo deve conseguir entender o código em questão;
- Um código executável para cada sistema operacional;

Compilação Clássica

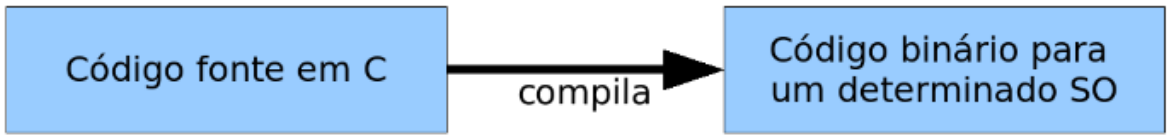
- Compilação na linguagem C:



- O SO alvo deve conseguir entender o código em questão;
- Um código executável para cada sistema operacional;
- Exemplos: Microsoft Office, Google Chrome, etc.

Compilação Clássica

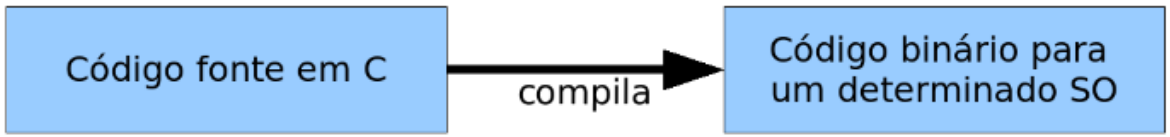
- Compilação na linguagem C:



- O SO alvo deve conseguir entender o código em questão;
- Um código executável para cada sistema operacional;
- Exemplos: Microsoft Office, Google Chrome, etc.
- Problema: como aproveitar bibliotecas de um SO?

Compilação Clássica

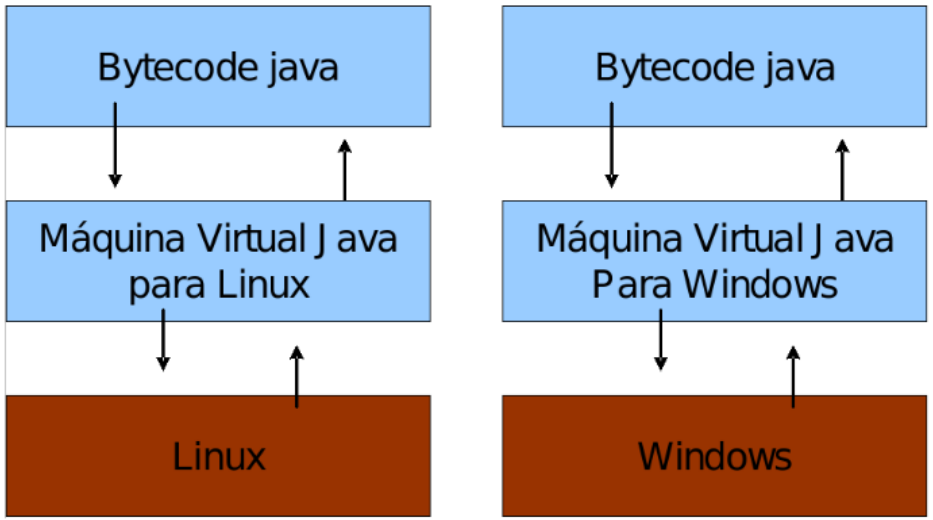
- Compilação na linguagem C:



- O SO alvo deve conseguir entender o código em questão;
- Um código executável para cada sistema operacional;
- Exemplos: Microsoft Office, Google Chrome, etc.
- Problema: como aproveitar bibliotecas de um SO?
- Reescrita de pedaços da aplicação.

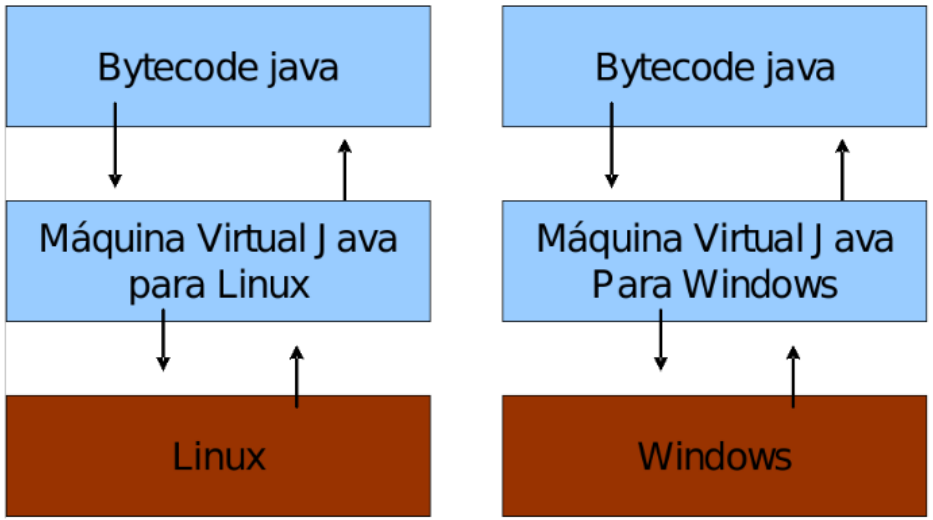
Máquina Virtual

- Java utiliza o conceito de **máquina virtual**:



Máquina Virtual

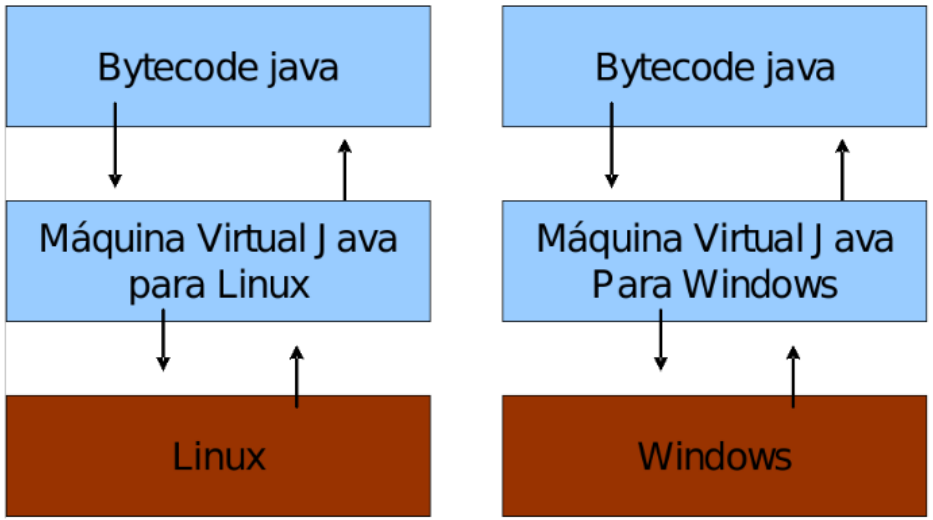
- Java utiliza o conceito de **máquina virtual**:



- Máquina virtual intermedia uso de bibliotecas nativas;

Máquina Virtual

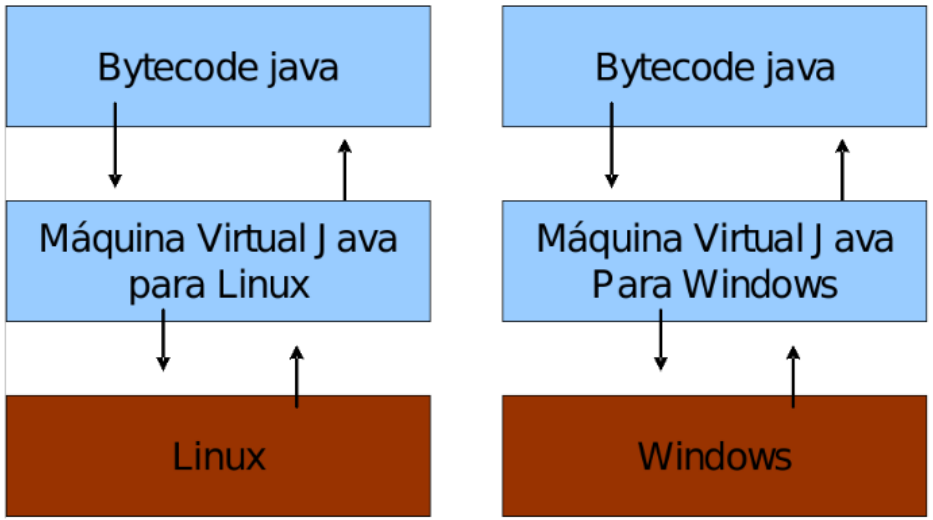
- Java utiliza o conceito de **máquina virtual**:



- Máquina virtual intermedia uso de bibliotecas nativas;
- Independência de SO e de plataforma.

Máquina Virtual

- Java utiliza o conceito de **máquina virtual**:



- Máquina virtual intermedia uso de bibliotecas nativas;
- Independência de SO e de plataforma.
- Máquina virtual vs. interpretador.

Máquina Virtual

- **JVM = Java Virtual Machine**;

Máquina Virtual

- **JVM = Java Virtual Machine**;
- Não entende código Java, entende *bytecodes*;

Máquina Virtual

- **JVM = Java Virtual Machine**;
- Não entende código Java, entende *bytecodes*;
- "*Write once, run anywhere*";

Máquina Virtual

- **JVM = Java Virtual Machine**;
- Não entende código Java, entende *bytecodes*;
- "*Write once, run anywhere*";
- Técnicas de otimização: *hotspot* e *JIT*.

Pré-requisitos para compilação e execução

- **JRE**(*Java Runtime Environment*): componentes necessários para execução de código Java (JVM + bibliotecas);

Pré-requisitos para compilação e execução

- **JRE**(*Java Runtime Environment*): componentes necessários para execução de código Java (JVM + bibliotecas);
- **JDK**(*Java Development Kit*): componentes necessários para execução e compilação de código Java (JVM + bibliotecas + compilador);

Pré-requisitos para compilação e execução

- **JRE**(*Java Runtime Environment*): componentes necessários para execução de código Java (JVM + bibliotecas);
- **JDK**(*Java Development Kit*): componentes necessários para execução e compilação de código Java (JVM + bibliotecas + compilador);
- Podem ser baixados em http://www.oracle.com/technetwork/java/.

- Java vs. produtividade;
- Aplicações de pequeno porte vs. aplicações de médio/grande porte;
- Maturidade e suporte da plataforma;
- Aplicações que necessitam de escalabilidade e heterogeneidade.


KEEP
CALM
AND
CHEGA DE
BLA BLA BLA


Let's Do This!