

# Máquinas Virtuais

Danilo Silva Marshall  
Érika R. C. de Almeida



# Tópicos abordados

---

- Introdução
- Tipos de máquinas virtuais
- Máquina virtual Java (Java Virtual Machine)
- Máquina virtual .NET
- Referências

# Tópicos abordados

---

- **Introdução**
- Tipos de máquinas virtuais
- Máquina virtual Java (Java Virtual Machine)
- Máquina virtual .NET
- Referências

# Introdução: O que é máquina virtual?

- É o nome que se dá a um ambiente, como um programa ou sistema operacional, que não existe fisicamente, mas sim é criado dentro de outro ambiente
  - Máquinas virtuais: hóspede (guest)
  - Ambiente: hospedeiro (host)

## Introdução: Para que é usada uma máquina virtual?

- Criadas para possibilitar a execução de um conjunto de instruções (*instruction set*) que difere do conjunto de instruções do ambiente hospedeiro
- Com frequência os ambientes hospedeiros rodam várias máquinas virtuais simultaneamente, exatamente como fazem com programas comuns
  - Recursos virtuais independem dos recursos físicos

# Tópicos abordados

---

- Introdução
- Tipos de máquinas virtuais
- Máquina virtual Java (Java Virtual Machine)
- Máquina virtual .NET
- Referências

# Tipos de máquinas virtuais

---

- Hardware
- Linguagens de Alto Nível

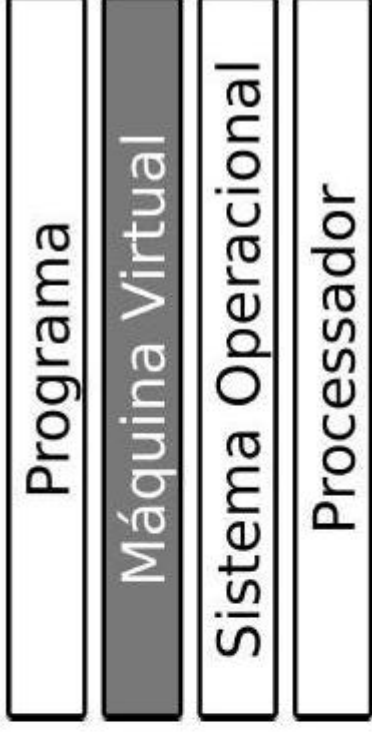
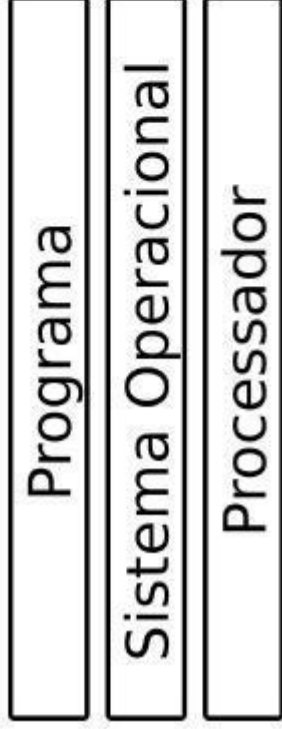
# Máquina virtual de Hardware

- Camada de virtualização fica no topo do hardware, exportando a abstração da máquina a ser virtualizada, como a CPU e a memória, para as camadas superiores
- Exemplo:
  - *VMware*

## Máquina virtual de Linguagem de Alto Nível

- Problemas de compilação com diferentes sistemas operacionais e arquitetura
- A máquina virtual cria uma camada de software entre o programa e o sistema operacional: agora, o programa funciona sobre a máquina virtual e esta se encarrega dos detalhes específicos de relacionamento com o sistema operacional e da arquitetura da máquina real

# Máquina virtual de Linguagem de Alto Nível



# Máquina virtual de Linguagem de Alto Nível

- Exemplos:
  - Máquina virtual Java
    - *Java Virtual Machine (JVM)*
  - Máquina virtual .NET

# Tópicos abordados

---

- Introdução
- Tipos de máquinas virtuais
- Máquina virtual Java (Java Virtual Machine)
- Máquina virtual .NET
- Referências

# Máquina virtual Java

- A linguagem Java é uma linguagem de alto nível orientada a objetos, que foi concebida com a portabilidade sendo um de seus pilares
- Objetivo de que os desenvolvedores se preocupem mais com o sistema a ser desenvolvido e não propriamente com o ambiente em que ele será executado, obtendo isso através do desenvolvimento baseado em uma máquina virtual



# Máquina virtual Java: Características

- Responsável por:
  - Independência de hardware e sistema operacional
  - Pequeno tamanho do código compilado
  - Proteger usuários de programas maliciosos (tem um controle muito fino sobre as ações liberadas para o código que está rodando na máquina virtual)

# Máquina virtual Java: Linguagens de Programação

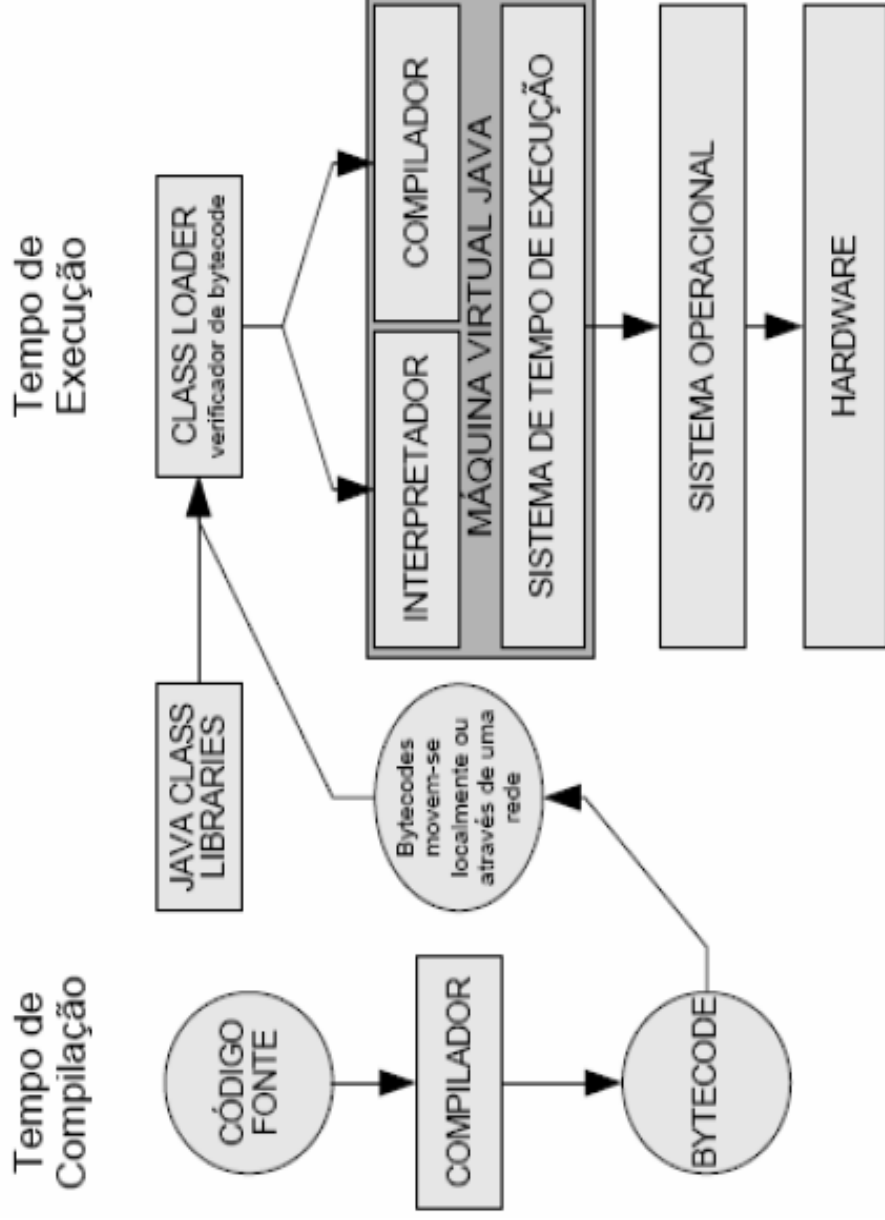
---

- Além da linguagem Java, pode ser usadas:
  - Linguagens de macro e correspondentes pré-processadores (Ex: JPP)
  - Linguagens funcionais (Ex: Lisp, Scheme)
  - Linguagens Lógicas (Ex: Prolog)
  - Linguagens de Script (Ex: Jython)

# Máquina virtual Java: Funcionamento

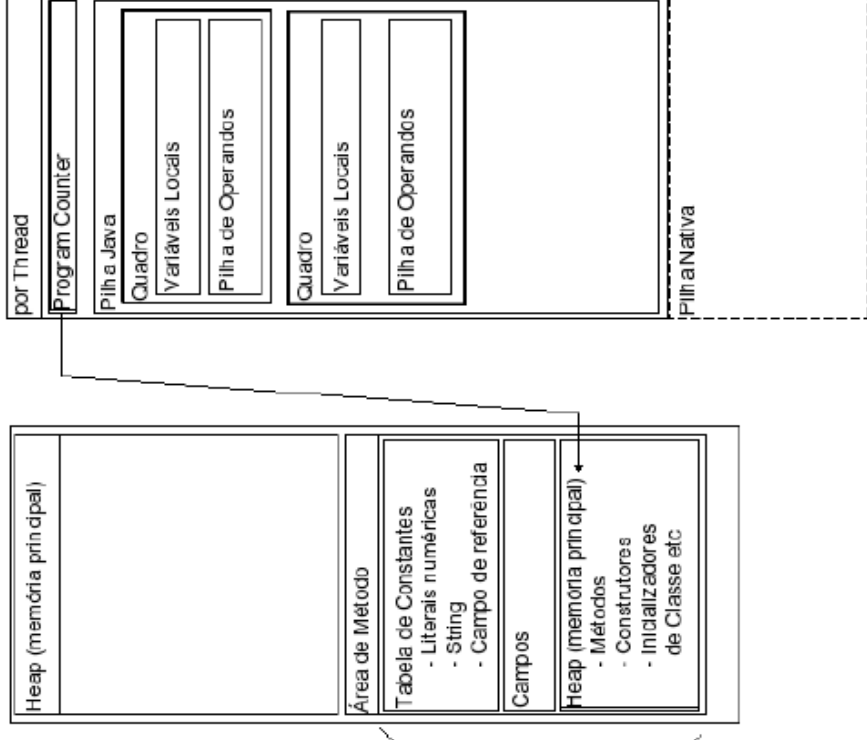
- Os programas escritos em Java são compilados para uma linguagem intermediária chamada *bytecode* através do *javac* (compilador Java)
- O interpretador Java então traduz o *bytecode* gerado para o conjunto de instruções que a máquina hospedeira pode entender
  - A máquina virtual Java nada sabe sobre a linguagem de programação Java, entende somente o arquivo *.class*, que contém os *bytecodes*, e uma tabela de símbolos
    - Conjunto de arquivos *.class: .jar* (Java archive)

# Máquina virtual Java: Funcionamento



# Máquina virtual Java: *Garbage-collected heap*

- Cada programa rodado na *JVM* possui uma quantidade de memória reservada em forma de *heap*
- Instâncias de classes, pilhas e vetores são alocados
- Gerenciamento de memória automático:
  - *Garbage-collector* aloca espaço no endereçamento de memória
  - Cria grafo de memória não utilizado
  - Coleta de lixo é feita com *background thread*



# Máquina virtual Java: Registadores

- Semelhantes ao de uma máquina real
- Argumentos são passados pela pilha
- Função: manter o estado da máquina, a cada linha de *bytecode* lido, eles são atualizados
  - Registrador PC (Um para cada thread da máquina virtual Java):
    - Método não-nativo (i.e., implementado em *bytecodes*): o pc conterá o endereço da instrução sendo executada
    - Método nativo: o valor deste registrador pc será indefinido

# Máquina virtual Java: Pilha

---

- Esquema de *last-in first-out* (LIFO)
- Cada *thread* possui sua pilha, que é criada juntada com a *thread*
- O elemento armazenado é chamado *frame* (quadro)
- Armazena variáveis locais e resultados parciais, além de lidar com a invocação e retorno de métodos
- Pode ser de tamanho fixo ou ser dinamicamente expandida ou reduzida de acordo com as necessidades do programa

# Máquina virtual Java: Pilha

- Exceções:
  - Pilha requer tamanho maior do que o permitido: *StackOverflowError*
  - Precisa ser dinamicamente expandida, mas não existe memória suficiente para tal ou ainda não há memória suficiente para criar uma nova pilha: *OutOfMemoryError*
- Registradores:
  - *Optop*: aponta para o topo da pilha de execução de operandos
  - *Vars*: aponta para a seção de variáveis locais
  - *Frame*: aponta para a seção de ambiente de execução, usada para gerenciar as operações da própria pilha

# Máquina virtual Java: Frames

- Conjunto de dados mantido dentro da pilha
  - Acoplamento dinâmico de métodos e objetos (*dynamic linking*)
  - Retorno de métodos (restaurar registradores)
  - Geração de exceções (em acoplamento dinâmico ou em tempo de execução)
- *Dynamic Linking*: cada quadro possui uma referência para a tabela de símbolos, de forma a realizar a ligação simbólica aos métodos e suas referências em tempos de execução, carregando classes sob demanda e resolvendo símbolos na medida da necessidade
  - Polimorfismo

# Máquina virtual Java: *Runtime Constant Pool*

- Tabela de Símbolos
- Contém vários tipos de constantes:
  - Valores numéricos conhecidos em tempo de compilação
  - Referências a métodos e campos a serem resolvidos em tempo de execução
- Está alocada na área de métodos
- Construída toda vez que uma classe ou interface é criada pela máquina virtual Java

# Máquina virtual Java:

## Área de métodos

---

- Compartilhada por todas as *threads*
- Guarda estruturas das classes, como:
  - Conjunto de constantes em tempo de execução
  - Campos e dados de métodos
  - Código de métodos e construtores
- Quaisquer informações de *debug* do método também

# Máquina virtual Java: Conjunto de instruções

- Mínimo de instruções para máximo de eficiência

```
System.out.println("Hello world!"); <operação> <operador(es)>
```

**Figura 1: Programa Hello World em Java**

**Figura 2: Formato básico de uma instrução**

```
0 getstatic #6 <Field java.lang.System.out Ljava/io/PrintStream;>  
3 ldc #1 <String "Hello world!">  
5 invokevirtual #7 <Method  
java.io.PrintStream.println(Ljava/lang/String;)V>  
8 return
```

**Figura 3: Bytecode do programa Hello World**

# Tópicos abordados

---

- Introdução
- Tipos de máquinas virtuais
- Máquina virtual Java (Java Virtual Machine)
- **Máquina virtual .NET**
- Referências

# Máquina virtual .NET

---

- Idealizado em meados da década de 90 sob o nome *Next Generation Windows Services (NGWS)*
- Lançado no final de 2000 sob o nome **.NET FRAMEWORK**

Microsoft®  
**.net**<sup>™</sup>

# Máquinas virtuais .NET: Linguagens de Programação

- Visual Basic
- .NET
- C#
- J#
- Outros exemplos: C(LCC), Lisp(L#), Perl(Active Perl), Ruby(Ruby .NET)

# Máquina virtual .NET: Características

- Interoperabilidade
- Independência da Linguagem
- *Base Class Library (BCL)*
- Segurança e facilidade
- *Common Runtime Engine (CRE)*
  - *Common Intermediate Language (CIL)*
  - *Common Language Infrastructure (CLI)*
  - *Common Language Runtime (CLR)*

# Máquina virtual .NET: Linguagem Intermediária

- *Common Intermediate Language (CIL)*:
  - Linguagem intermediária que abstrai os detalhes do *hardware* e sistema operacional
  - Suporta tipos, ponteiros, referências a objetos e tipos numéricos
  - Também conhecido como *Microsoft Intermediate Language (MSIL)*

# Máquina virtual .NET: *Metadata*

---

- Informações geradas pelo compilador, independentes da linguagem usada, sobre a estrutura do programa, tornando mais fácil e organizado interpretar o código escrito
- O fato de o *metadata* descrever uma linguagem em uma outra linguagem intermediária (CIL) neutra de maneira transparente torna o desenvolvimento mais fácil e produtivo
- Vantagem:
  - Possibilidade de as informações serem referenciadas entre várias linguagens e ferramentas

# Máquina virtual .NET: Carregador de classes

---

- Carrega um arquivo do tipo *Portable executable* (PE) que contém todos os *metadatas* e o código intermediário CIL necessário para a execução da aplicação
- Cada método carregado é referenciado e submetido à compilação em tempo de execução. Uma vez compilado, o código nativo é gerado e estará pronto para executar, sempre que necessário

# Máquina virtual .NET: Compilador JIT (*Just In Time*)

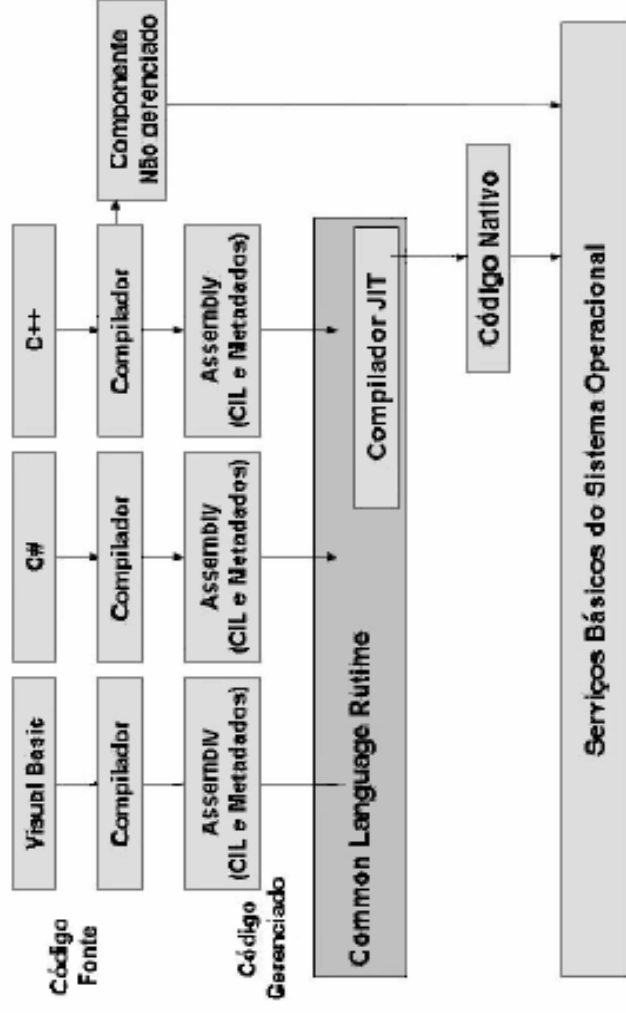
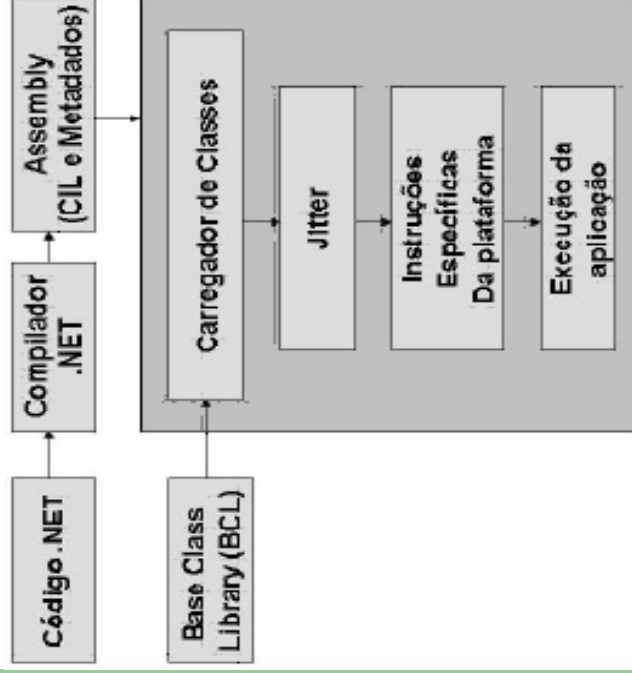
- Termo *Just In Time* se deve à compilação do código por demanda, ou seja, quando ocorre pela primeira vez
- Responsabilidade do compilador transformar o CIL e os metadatas em arquivos executáveis “úteis” ao hardware e sistema operacional do usuário
- Vantagens:
  - Eficiência: métodos são compilados uma única vez e somente quando requisitados
  - Portabilidade: código não é compilado para uma plataforma específica antes de ser distribuído

# Máquina virtual .NET: Execução

- ***Virtual Execution System (VES)***

- Responsável por carregar e executar programas compatíveis com o *Common Language Infrastructure (CLI)*, usando as informações dos *metadatas* e do código fonte para gerar um único arquivo em tempo de execução
- Uma vez que o código é executado, o VES específico da plataforma compila o CIL gerado para a linguagem nativa (linguagem de máquina) através do JIT, agora sim, obedecendo à arquitetura do hardware

# Máquina virtual .NET: Estrutura



# Máquina virtual .NET: *Garbage Collector*

- Obedece o mesmo princípio básico seguido pelo JVM, com pequenas diferenças de política de otimização:
  - Ao passo que no JVM podemos configurar o tamanho máximo da pilha e, conseqüentemente, a agressividade do algoritmo de coleta, no gerenciamento de memória do .NET, a pilha é dividida em 3 partes: 0, 1 e 2. Inicialmente a memória é alocada na região 0 e, conforme a informação vai envelhecendo, ou seja, não é descartada durante as coletas, os dados vão sendo movidos para os níveis 1 e 2, respectivamente

# Máquina virtual .NET: Segurança

---

- Verificação e validação, método no qual executa-se testes que verificam se o programa executa alguma operação maliciosa ao carregar um *assembly* ou se os *metadatas* e CIL são válidos
- *Code Access Security (CAS)*, método que tem a função de evitar que códigos não confiáveis executem ações privilegiadas na máquina

# Tópicos abordados

---

- Introdução
- Tipos de máquinas virtuais
- Máquina virtual Java (Java Virtual Machine)
- Máquina virtual .NET
- **Referências**

# Referências

---

- Gomes, C. S.; Jin, N. K.; Crepaldi, T. F. (2007) Máquinas virtuais Java e .NET
- Scorciapino, K. G. E.; Nascimento, E. R. (2006) Máquinas virtuais Java e .NET