Módulo IV

Características da Linguagem Java

Assuntos

Histórico

Características da Linguaguem

Garbage Collection

Créditos

Autor

Prof. Alessandro Cerqueira

(alessandro.cerqueira@hotmail.com)

Histórico da Linguagem Java

- Java é um projeto da Sun Microsystems (1994) que foi chefiado por James Gosling e concebido inicialmente para aparelhos que possuíssem processadores (TVs, microondas, etc).
  - Linguagem Oak
- Com o advento da Web, o projeto teve seu desenvolvimento reorientado.
- Java apresenta características adequadas para a criação de aplicações para ambientes corporativos.
- Site oficial: java.sun.com

Características da Linguagem Java

- *"Simples"*
- *Orientada a Objetos*
- *Distribuída*
- *Interpretada*
- *Robusta*
- *Neutra*
- *Segura*
- *"Alto desempenho"*
- *Portátil*
- *Paralelizável*
- *Dinâmica*

Características da Linguagem Java

"Simplicidade"

- Aprender Java não é tarefa simples. Porém torna-se mais fácil para quem domina Orientação a Objetos e/ou já programa com C ou C++, já que sua sintaxe possui alguns pontos em comum com estas linguagens.
- Em geral, desenvolver um sistema em Java torna-se mais simples do que desenvolver em C++ pois não apresenta alguns de seus recursos "desaconselháveis" como:
  - Aritmética de ponteiros (ex: *--pt = vet+5)
  - Estruturas (structs)
  - Definição de tipos (typedef)
  - Macros (#define)
  - Liberação explicita de memória (free)
- Com isto, pode-se eliminar boa parte dos erros mais comuns e simplifica a codificação.

Características da Linguagem Java
Baseada em OO, Distribuída, Robusta

- (OO) Utiliza conceitos do Modelo Orientado a Objetos.
- (Distribuída) Aplicações que envolvem a comunicação entre computadores através da rede são chamadas de Aplicações distribuídas.

  Java apresenta recursos para programação com TCP/IP (sockets), além disponibilizar outras tecnologias para criação de aplicações distribuídas como: RMI, CORBA, **WebServices**, etc.
- (Robusta) Possui verificação em tempo de compilação e execução com o objetivo de gerar códigos confiáveis.
- Gerenciamento automático de memória c/ **Garbage Collection**

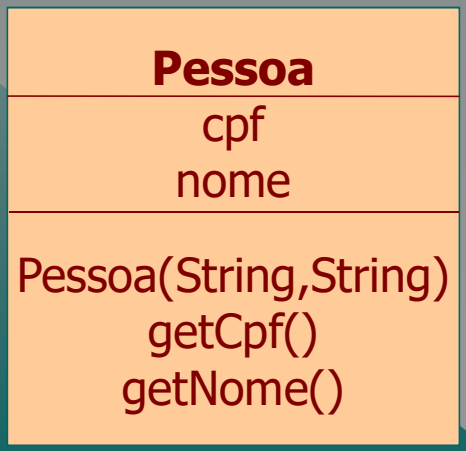
Características da Linguagem Java

Trecho de Código em Java (Exemplo de Garbage Collection)

- Considere o seguinte trecho de código:

```
(1) Pessoa p1, p2, p3;
(2) p1 = new Pessoa("123456789-00","José da Silva");
(3) p2 = new Pessoa("098765432-10","Maria de Souza");
(4) p3 = p1;
```

- Também considere a classe Pessoa com a seguinte interface:

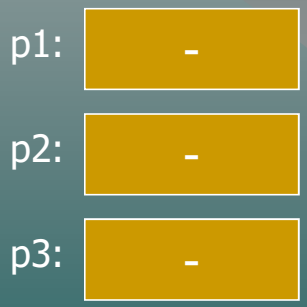


Características da Linguagem Java

Declaração de Variáveis (Exemplo de Garbage Collection)

```
(linha 1) Pessoa p1, p2, p3;
```

- Na primeira linha estamos declarando três variáveis locais *p1*, *p2* e *p3* que são ponteiros para *Pessoa*. Como são variáveis locais, não podemos determinar o valor presente dentro delas. Entretanto, veremos à frente que em Java tudo que for alocado na Heap terá uma inicialização default; ou seja, tudo recebe automaticamente um valor padrão*.



Toda variável/atributo/parâmetro cujo tipo não for um tipo primitivo Na realidade é um ponteiro em Java

Obs*: Para as variáveis locais, não podemos contar que a inicialização default (null) será aplicada.

Características da Linguagem Java

Operadores (Exemplo de Garbage Collection)

```
(linha 2) p1 = new Pessoa("123456789-09","José da Silva");
```

- Na segunda linha temos dois operadores = e *new*. Pela tabela de precedência, o primeiro a ser executado é o *new*. Este é responsável pela instanciação de um novo objeto da classe Pessoa. Para isto, o operador realiza duas operações:

  (1) O operador *new* aloca memória na Heap para o novo objeto, promovendo a inicialização default para os seus atributos.

Observe que foi criado na memória (Heap) um novo objeto da classe Pessoa e que seus atributos foram inicializados com "null".
Considere que o endereço de memória onde este objeto foi criado é o 82AE4C

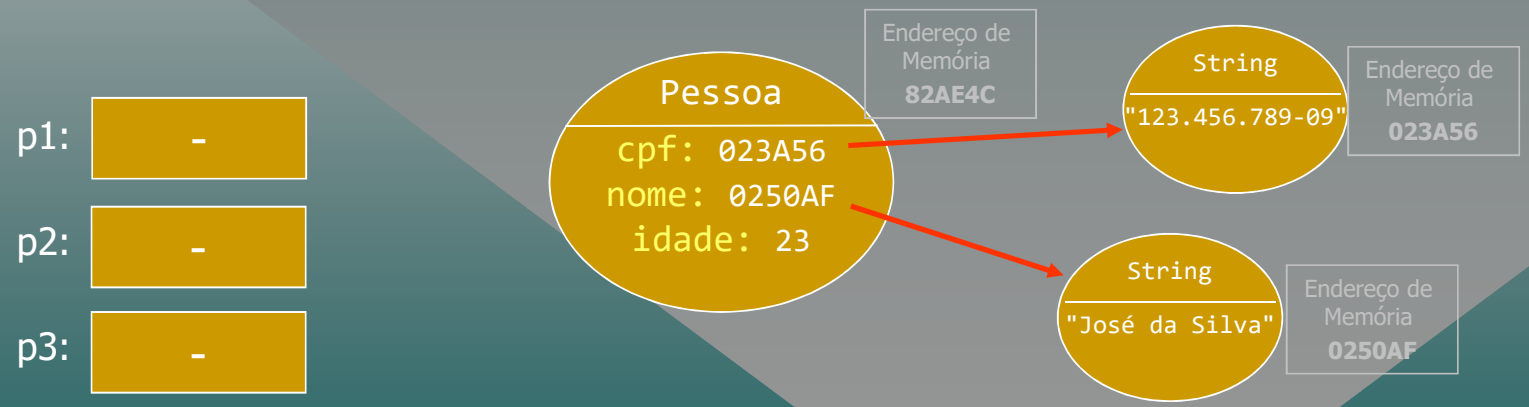
Características da Linguagem Java

Operadores (Exemplo de Garbage Collection)

```
(linha 2) p1 = new Pessoa("123456789-09","José da Silva");
```

(2) O *new* envia uma mensagem para o novo objeto solicitando a execução do método construtor da classe Pessoa.

- Método construtor é um método que o programador coloca na classe cuja responsabilidade é inicializar o novo objeto. (deveria se chamar método inicializador)
- Quem executa o método construtor é o **objeto recém-criado**.
- Em Java, o método construtor é aquele que possui o mesmo nome da classe.



Observe que o objeto Pessoa agora apresenta os seus atributos com valores.
Quem promoveu esta alteração foi o método construtor através dos parâmetros enviados pelo *new*

Características da Linguagem Java Operadores (Exemplo de Garbage Collection)

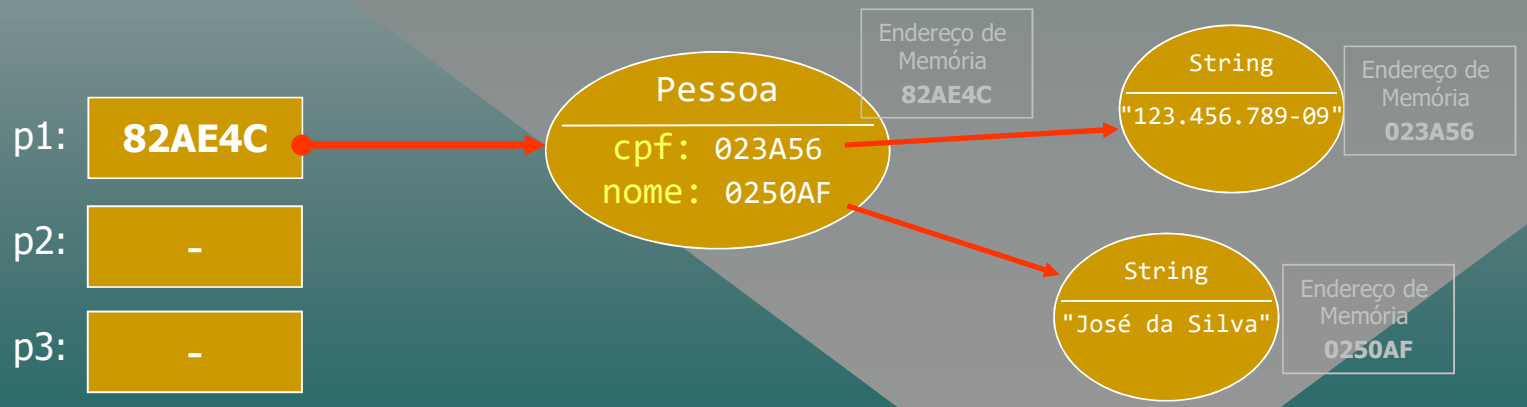
- **Dicas Importantes:**
  - Imagine que cada objeto possui um processador capaz de executar os métodos definidos em sua classe. Quando ele recebe uma mensagem, o objeto pega o código do método associado à mensagem e o executa como se fosse um "script".
  - Um método deve ser escrito de tal forma que qualquer objeto da classe possa executá-lo. Veremos à frente que para escrevermos métodos é necessário uma forma de referenciar os atributos do objeto que estiver executando o método num determinado instante.
  - Devemos considerar que o operador **new** ***retorna ao final da sua execução o endereço de memória*** onde foi alocado o novo objeto.

Características da Linguagem Java

Operadores (Exemplo de Garbage Collection)

```
(linha 2) p1 = new Pessoa("123456789-09","José da Silva");
```

- Depois de instanciado, o operador de atribuição (=) será executado. Este fará com que p1 passe a apontar para o novo objeto.
  - Sempre que fizermos uma atribuição a um ponteiro (ou seja, quando o ponteiro estiver do lado esquerdo da atribuição), queremos que esse **passe a apontar** para o objeto indicado ao lado direito (ou seja, vamos pegar o endereço fornecido do lado direito da atribuição e o armazenamos na variável)
  - Uma variável ponteiro é semelhante a uma variável inteira. Porém o número guardado faz referência a um endereço de memória e as operações com este tipo de variável sempre dizem respeito ao objeto apontado.

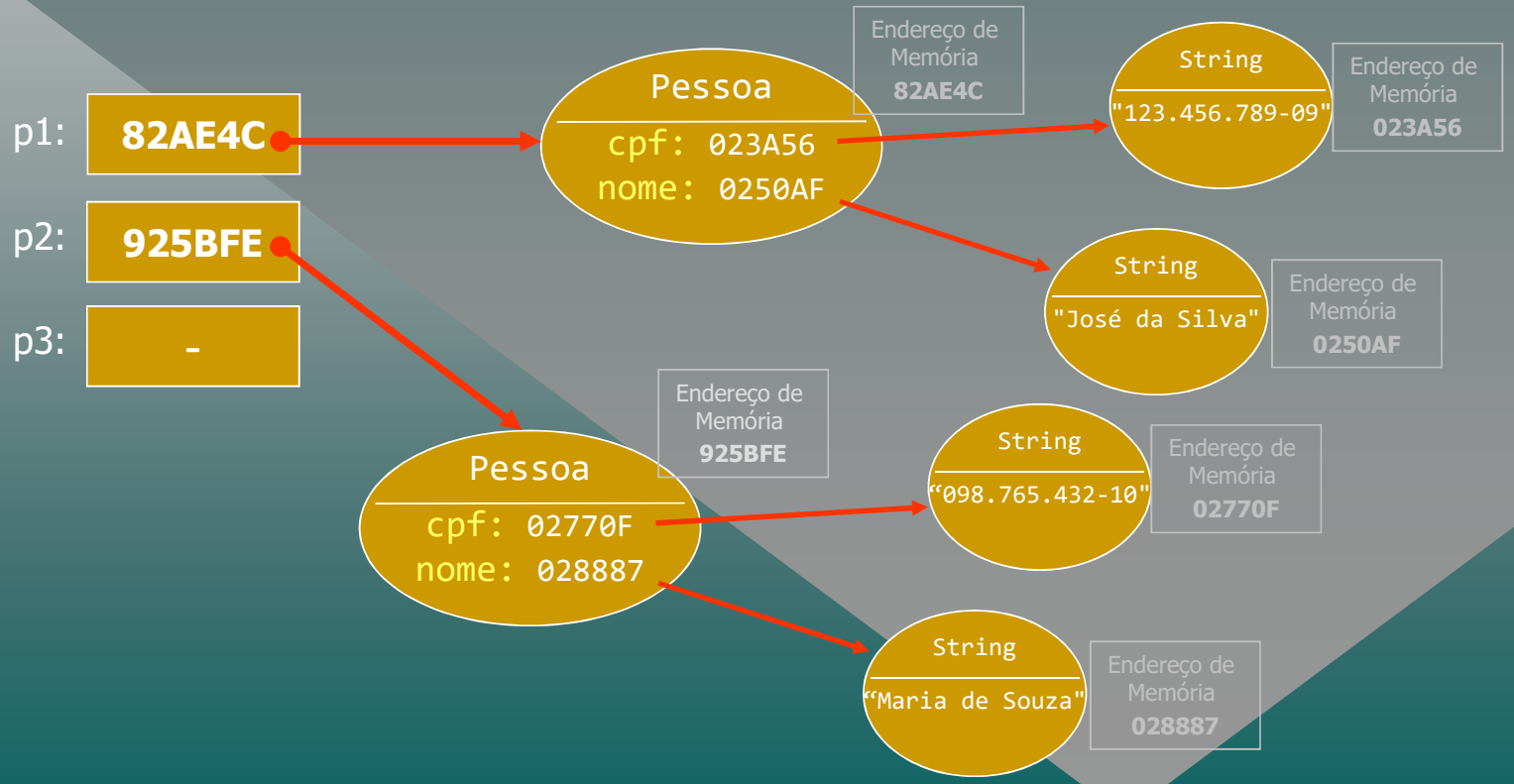


Características da Linguagem Java

Operadores (Exemplo de Garbage Collection)

```
(3) p2 = new Pessoa("098765432-10","Maria de Souza");
```

- Na terceira linha ocorre a mesma coisa que ocorreu na segunda linha, porém *p2* passa a apontar para o novo objeto.

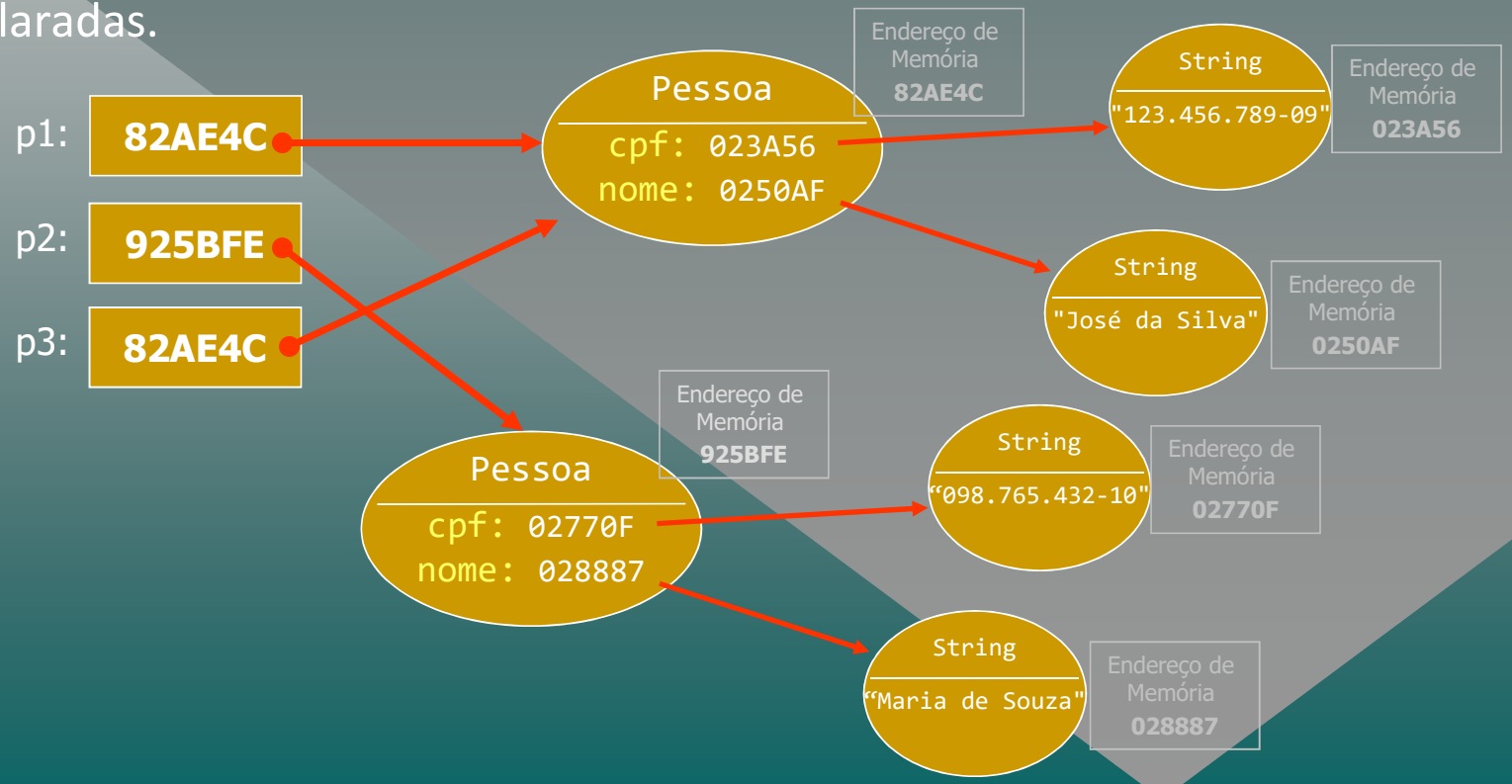


Características da Linguagem Java

Operadores (Exemplo de Garbage Collection)

```
(linha 4) p3 = p1;
```

- Na quarta linha, *p3* passa a apontar para o objeto apontado por p1.
  - Observe que todos os objetos alocados podem ser acessados através das variáveis declaradas.



Características da Linguagem Java

Operadores (Exemplo de Garbage Collection)

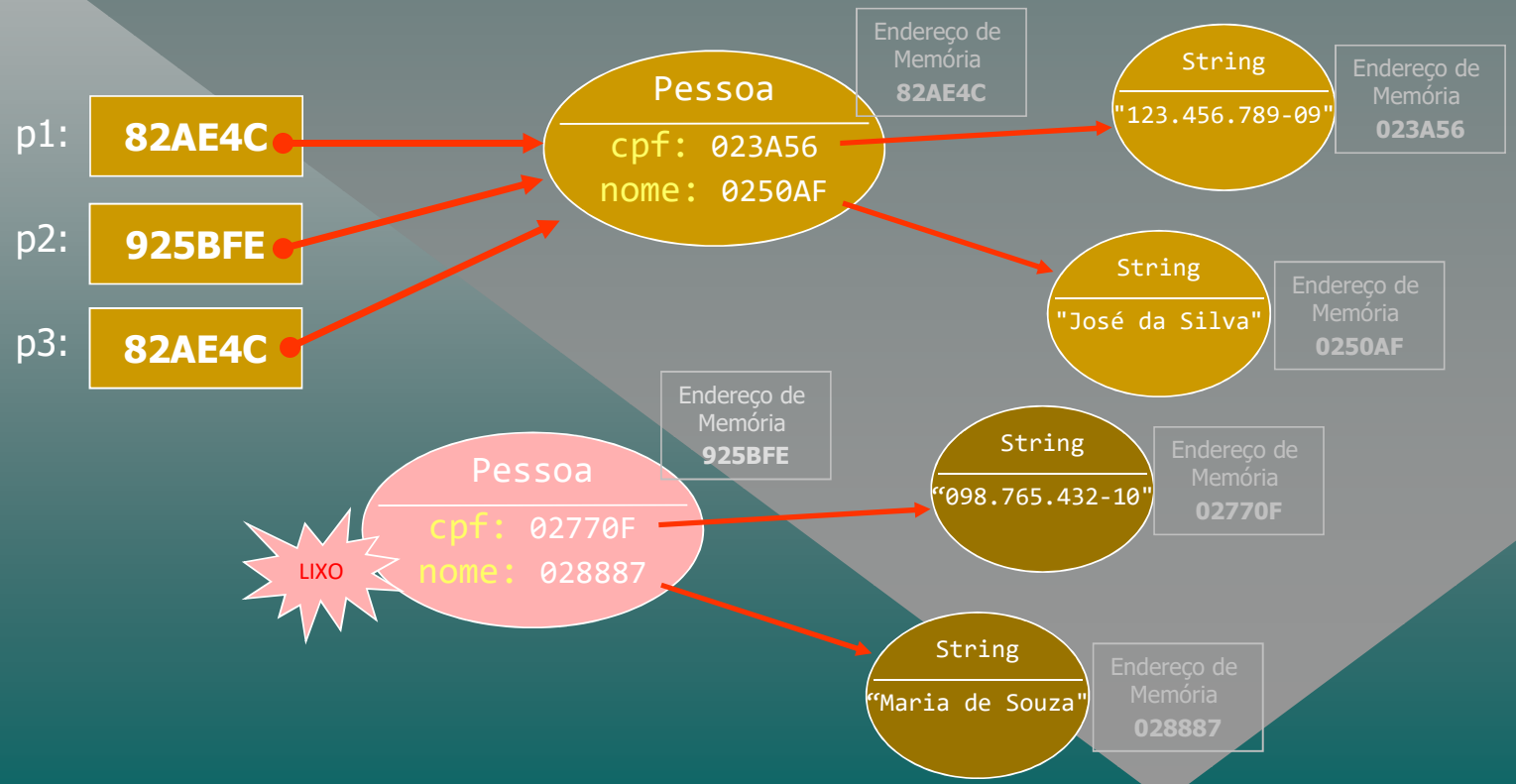
- Sempre que encontrarmos um ponteiro em um código Java, a semântica de seu uso será: **"para o objeto apontado por *<ponteiro>*"**.
- A única exceção é quando o ponteiro está do lado esquerdo da atribuição, cuja semântica é: **"*<ponteiro>* passa a apontar"**.
- Ex:

```
p3 = p1;
```

p3 passa a apontar | para o objeto apontado por **p1**

Características da Linguagem Java
Garbage Colletion (Coletor de Lixo)

- Se adicionarmos uma quinta linha que faça: **(linha 5)** `p2 = p1;`
  Veremos que estamos perdendo a referência que tínhamos para o segundo objeto que foi alocado, e nunca mais poderemos ter acesso a esse objeto **(LIXO)**

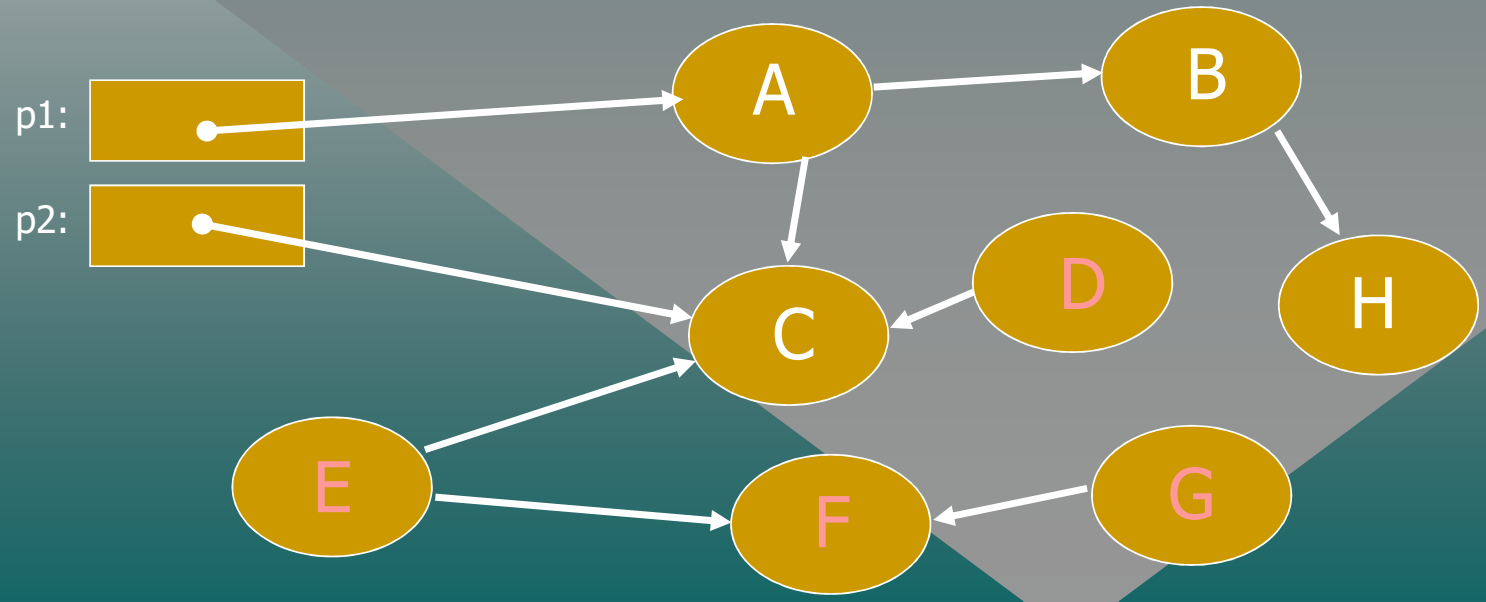


Características da Linguagem Java
Garbage Colletion (Coletor de Lixo)

- Em linguagens como C++, é dever do programador codificar os procedimentos para desalocação de memória. Já em Java, não há esta necessidade pois esta tarefa é feita automáticamente pelo *Garbage Colletion*.
- Este processo é executado por uma Thread que roda em paralelo ao programa recolhendo os objetos "perdidos" e desalocando a memória destinada a eles.
- Não sabemos quando exatamente o coletor irá rodar. Porém quando for executado, o processo removerá todos os objetos considerados lixo.

Características da Linguagem Java
Garbage Collection (Coletor de Lixo)

- Consideramos um objeto perdido se não for possível recuperá-lo direta ou indiretamente a partir das variáveis
  - Ou seja, não há um caminho que ligue alguma variável ao objeto.
- Observe a situação abaixo. Após a execução do garbage collection, os objetos D, E, F e G seriam removidos.



Características da Linguagem Java
Considerações

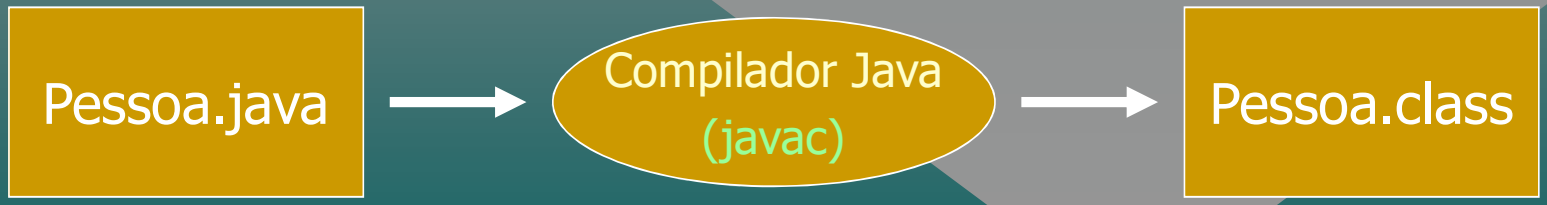
- Apesar do exemplo ter sugerido os endereços de memória, em Java não é possível saber qual é o endereço de memória onde um objeto está (aspecto de segurança).
  - O método **`hashCode()`** presente na classe Object e, por consequência, em todas as classes pode sinalizar qual é o endereço de memória (absoluto ou relativo) onde o objeto está.

Características da Linguagem Java
Neutra, Compilada e Interpretada

- Cada classe escrita em Java deve ficar em um arquivo cujo nome é formado pelo nome da classe + extensão *.java*

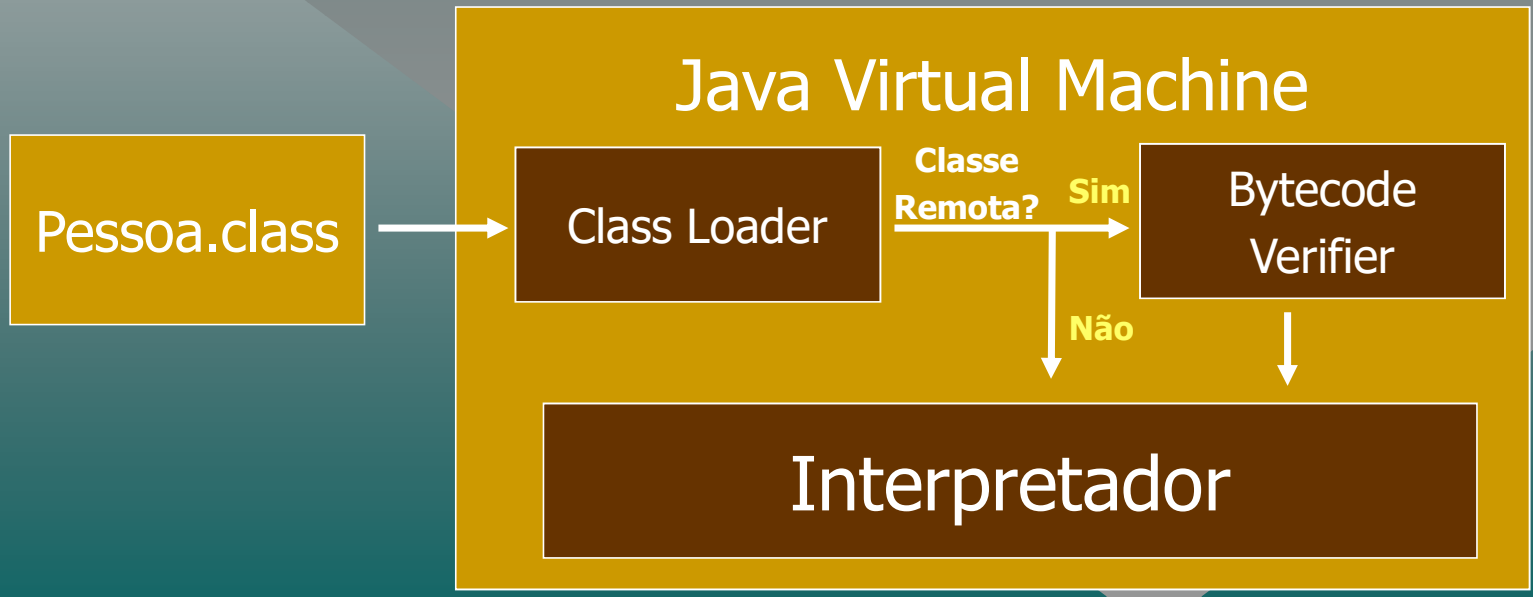
(ex: Classe Pessoa → Pessoa.java)

- Ao submetermos este código ao compilador, ele não gera código de máquina, mas gera um código chamado de *Bytecode*, que é o código de máquina que só pode ser executado por um processador virtual (programa) chamado JVM (Java Virtual Machine).
- Para cada classe compilada é gerado um arquivo com o bytecode produzido pelo compilador. O seu nome é formado pelo nome da classe + extensão *.class*.



Características da Linguagem Java
Neutra, Compilada e Interpretada

- **Para executarmos o programa necessitamos de uma JVM (software) que:**
  - Carrega o bytecode com o *Class Loader*;
  - Se o bytecode veio de fonte externa (ex.Applet), ele passa por um Verificador de ByteCode (segurança!)
  - O interpretador repetidamente lê uma instrução e a executa.



Características da Linguagem Java
Neutra, Compilada e Interpretada

- Genericamente, um Interpretador é um programa capaz de ler e executar um programa; para isto, ele continuamente
  - **(a)** Lê uma instrução,
  - **(b)** Verifica aspectos léxicos/sintáticos/semânticos da instrução,
  - **(c)** Examina quais operações estão associadas à instrução e
  - **(c)** Executa as operações.
  - Ex: JavaScript (Cada navegador web possui um interpretador desta linguagem capaz de ler e executar as funções embutidas na página HTML)
- Em Java, o processo de interpretação é **muito mais simples** que o relatado acima, pois muitas das etapas já foram feitas pelo compilador Java (ex. verificação da sintaxe) e as instruções presentes no bytecode são de fácil entendimento.
- Podemos afirmar que o processo de execução de programas realizado pelo Interpretador Java é muito semelhante ao que uma CPU faz para executar um programa .EXE em memória.

Características da Linguagem Java
(Curiosidade) Tecnologias das JVMs

- Em JVMs com tecnologia HotSpot (mais comum), há a presença do interpretador.
- Entretanto há uma exceção que são as JVMs com tecnologia JIT (Just in Time), o bytecode é convertido para código nativo em tempo de execução objetivando-se melhorar o desempenho do programa. Somente neste caso não há processo de interpretação.
- Estudos mostram que o desempenho de JVMs JIT são praticamente similares ao das JVMs HotSpot.
- Podemos dizer que Java é uma linguagem **neutra** pois o bytecode produzido não está vinculado a nenhum processador/sistema operacional específico.

Características da Linguagem Java
Neutra, Compilada e Interpretada

- Sumarizando:
  - Java é uma linguagem tanto compilada quanto interpretada pois cada classe em Java deve ser submetida ao Compilador Java que, ao invés de gerar um código "assembler" para um processador específico, gera um código chamado bytecode que não está vinculado a nenhum processador ou SO (por isso dizemos que Java é uma linguagem neutra).
  - Para executarmos um programa em Java, necessitamos um Interpretador Java que leia cada código do bytecode e o execute.
  - A grande vantagem de Java apresentar estas características é a PORTABILIDADE

Características da Linguagem Java
Segura, Portátil e "Alto Desempenho"

- (Segura) Apesar de termos variáveis que são ponteiros, Java não nos deixa manipular (ler ou alterar) com o valor do endereço.
  - Extensiva verificação do bytecode (presença do mecanismo CRC).
- (Portátil) Roda em qualquer plataforma.
  - Além de neutra, implementa tipos de dados com tamanho fixos (ex. *Int* 's têm 32 bits).
- ("Alto desempenho") Apesar de Java ser interpretada, os programas rodam rapidamente pois o processo de interpretação é simples.

Características da Linguagem Java
Paralelizável e Dinâmica

- (Paralelizável) Suporta programação de *threads* (subprocessos que rodam "paralelamente" ou "concorrentemente", dependendo da plataforma).
- (Dinâmica) As classes são carregadas em tempo de execução, de acordo com a necessidade (ou seja, a JVM só carrega uma classe quando houver necessidade).
  - Estas classes podem ser atualizadas separadamente (alta coesão e baixo acoplamento)
- Além destas características, encontramos em Java uma série de tecnologias que permitem criar sistemas para uma série de aplicações.

Nomenclaturas

- JDK – Java Development Kit
  - Conjunto de ferramentas (ex. Compilador, Interpretador, Jar, Javadoc, etc.), classes (Java Platform Core Classes) e documentção para fazer o desenvolvimento em Java.
- JRE – Java Runtime Environment
  - Ferramentas para execução. Vem com a JVM (Interpretador simplificado (ex. sem apoio à depuração) para fazer implantação de aplicações) + Java Core Libraries.
- Java Core Libraries
  - Conjunto de classes que compõem a base de uma versão de Java.
- JVM – Java Virtual Machine
  - Máquina Virtual Java (“Interpretador” e demais itens)
- Versão LTS (Long Term Support)
  - Indica que a versão receberá manutenção e suporte, mesmo com o lançamento de versões mais novas. São versões LTS: 8, 11 e 17.
- Java 5 (2004)
  - Nome dado à versão 1.5 (agora chamada de 5.0) do JDK e referência a um padrão de Core Classes. Nessa versão foram acrescentados novos recursos à linguagem.
- Java 8 (2014)
  - Última versão LTS antes do JPMS (módulos) serem introduzidos no Java 9. A partir dessa versão, a cada seis meses há uma nova release com mudanças na linguagem ou nas API’s.

Nomenclaturas

- JAR – Java Archive
  - É um arquivo que armazena dentro de si vários arquivos .class (assim como um arquivo .zip). Serve para empacotar todos os bytecodes que formam um sistema desenvolvido com Java.
- Java SE – Java Platform Standard Edition
  - Centro da tecnologia Java
  - Conjunto básico para o desenvolvimento de aplicações Java.
- Java EE – Java Platform Enterprise Edition
  - Conjunto de especificações e práticas que apoiam o desenvolvimento e implantação de aplicações “multi-tier”
  - Construído sob a Java SE (Standard Edition);
  - Tecnologias: EJB, Servlets, JAXR (XML), Corba, etc.