Alocação Dinâmica de Memória em Java

Prof. Dr. Marcelo Fernando Rauber

Alocação Dinâmica de Memória em Java

- Alocação Dinâmica de Memória significa que o programa pode definir a quantidade de memória que utiliza durante a sua execução.
- Em Java, desde a sua concepção é todo orientado a Objetos. Agora, não nos aprofundaremos neste tema.
- A exemplo dos Tipos Astratos de Dados, trabalharemos com class e o operador new.

Alocação Dinâmica de Memória em Java

- Java Garbage Collection:
  - É um processo automático da JVM
  - Responsável por **liberar**/apagar da memória os objetos que não são referenciados, isto é, não estão sendo usados;
- Outras linguagens, como C, C++, a liberação da memória é “manual” pelo programador, com comandos específicos para esse fim.
  - É um trabalho a mais a ser realizado;
  - Em Java a JVM automatiza esse processo.
- É um ponto de discussão, já que o controle manual da liberação de memória, poderia, em tese, trazer melhoria de controle ou desempenho.

Alocação Dinâmica de Memória em Java

- Para facilitar o entendimento, passaremos a apresentar alguns códigos e procurar entender o que está acontecendo na memória RAM do computador. Para isso os Slides estarão divididos em duas partes:
  - Do Lado esquerdo, serão apresentados os códigos fonte em Java;
  - Do lado direito, será representa a memória do computador, como se os códigos fonte do lado esquerdo já tivessem sido executados.

Alocação Dinâmica de Memória em Java

Código fonte em Java:

RAM:

Alocação Dinâmica de Memória em Java

- Primeiramente, passemos a analisar uma variável simples, de tipo primitivo da linguagem.
  - Nesses caso a variável aponta diretamente para o espaço de memória.

Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
// Programa principal
/* Declaração de variável de um
tipo primitivo */

 double Salario;
```

RAM:

Alocação Dinâmica de Memória em Java

Código fonte em Java:

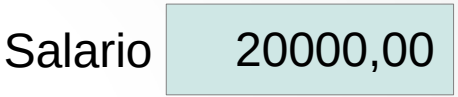
```
// Programa principal
/* Declaração de variável de um
tipo primitivo */

 double Salario;


//Definindo um valor

Salario = 20000;
```

RAM:



Alocação Dinâmica de Memória em Java

- Para fins didáticos, para trabalhar com alocação dinâmica, nós devemos:
  - Definir um TAD (veja aulas anteriores);
  - Criar uma nova variável desse TAD, com o new;
    - Neste caso, a variável não faz acesso direto a memória, mas sim, tem uma referência de onde estão dos dados, isto é, guarda o endereço de memória de onde estão os dados.
- Nos slides seguintes vamos aprofundar nossos estudos de TAD.

Alocação Dinâmica de Memória em Java

- Para fins didáticos, vamos assumir que foi definida a seguinte classe ou TAD dentro do package:

```
class Carro {
   String Placa;
   int Ano;
}
```

Alocação Dinâmica de Memória em Java

Código fonte em Java:

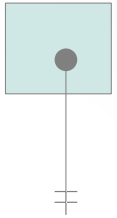
```
// Programa principal
/* Declaração de variável da
classe anterior */

Carro c1;


/* Não cria toda a estrutura
na memória, apenas um espaço
para apontar para algum
lugar, por enquanto, aponta
para null */
/* Neste caso, usamos para
representação usamos o
símbolo de terra */
```

RAM:

C1



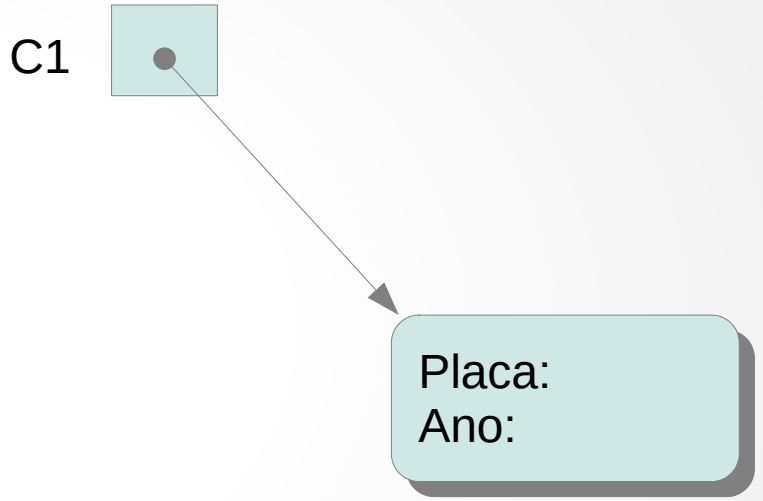
Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
Carro c1;
c1 = new Carro();


/* O comando new aloca a
estrutura na memória, e C1
faz uma referencia a essa
estrutura, isto é, aponta
para essa estrutura */
```

RAM:



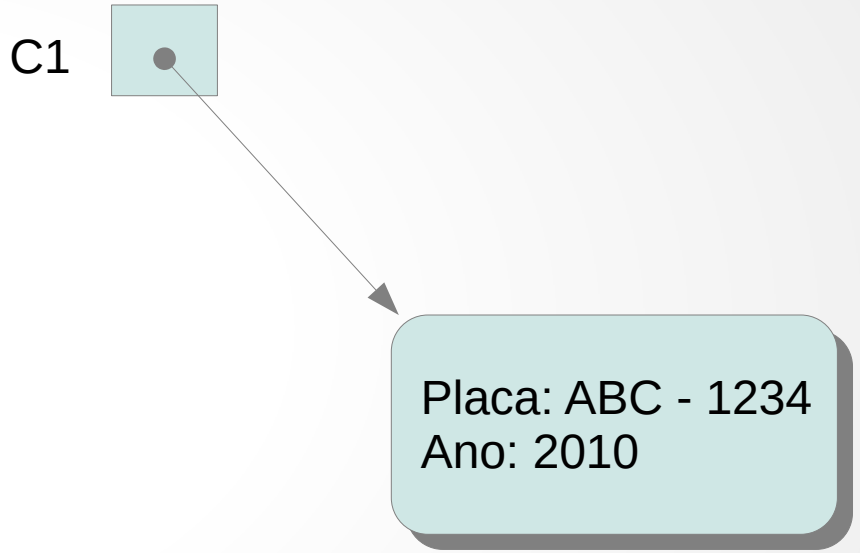
Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
Carro c1;
c1 = new Carro();
//Definindo valores
c1.Placa = "ABC - 1234";
c1.Ano = 2010;


/* O comando new aloca a
estrutura na memória, e C1
faz uma referencia a essa
estrutura, isto é, aponta
para essa estrutura */
```

RAM:



Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
Carro c1;
Carro c2;
c1 = new Carro();
c1.Placa = "ABC - 1234";
c1.Ano = 2010;
c2 = c1;
c2.Placa = "DFG - 9876";
c2.Ano = 1980;

System.out.println("c1: " +
c1.Placa);
System.out.println("c2: " +
c2.Placa );
```

RAM:

Consegues determinar o que aparecerá na tela???

Tende determinar antes de seguir.
Resposta no próximo Slide.

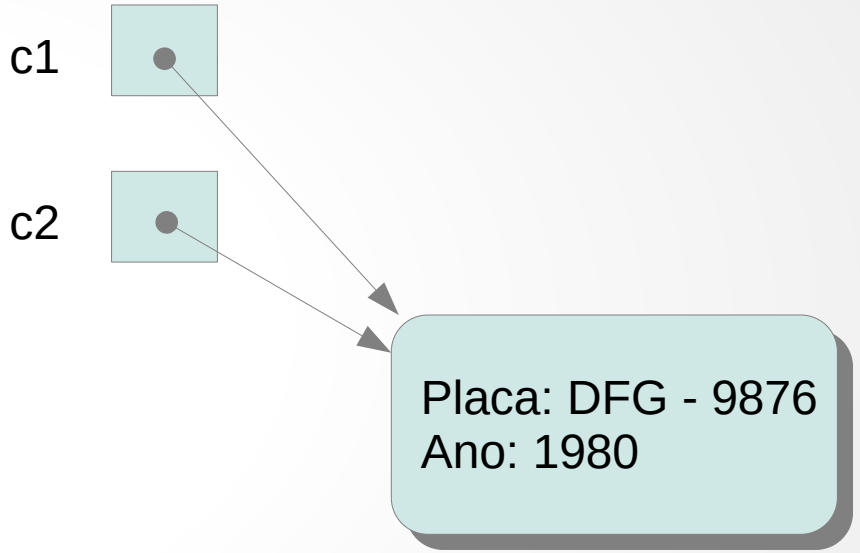
Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
Carro c1;
Carro c2;
c1 = new Carro();
c1.Placa = "ABC - 1234";
c1.Ano = 2010;
c2 = c1;
c2.Placa = "DFG - 9876";
c2.Ano = 1980;

System.out.println("c1: " +
c1.Placa);
System.out.println("c2: " +
c2.Placa );
```

RAM:



Tanto c1 quanto c2 apontam para a mesma estrutura, portando, ao definir valores para c2, estará substituído o que tinha em c1.
Portanto, na tela aparecerá:
c1: DFG - 9876
c2: DFG - 9876

Alocação Dinâmica de Memória em Java

Listas Encadeadas SIMPLES

Listas Encadeadas Simples

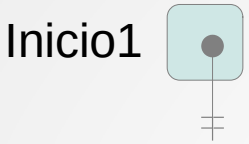
- Como já vimos, lista é um conjunto de elementos;
- É um conjunto de elementos, de mesmo tipo, conectados.
- Cada elemento dessa lista é chamado de Nó;
- Cada Nó aponta para o próximo Nó da lista;
- É chamada de “simples” pois aponta para um único Nó;
- Em tempo de execução, podemos alocar na memória tantos Nós quantos forem necessários.

Listas Encadeadas Simples

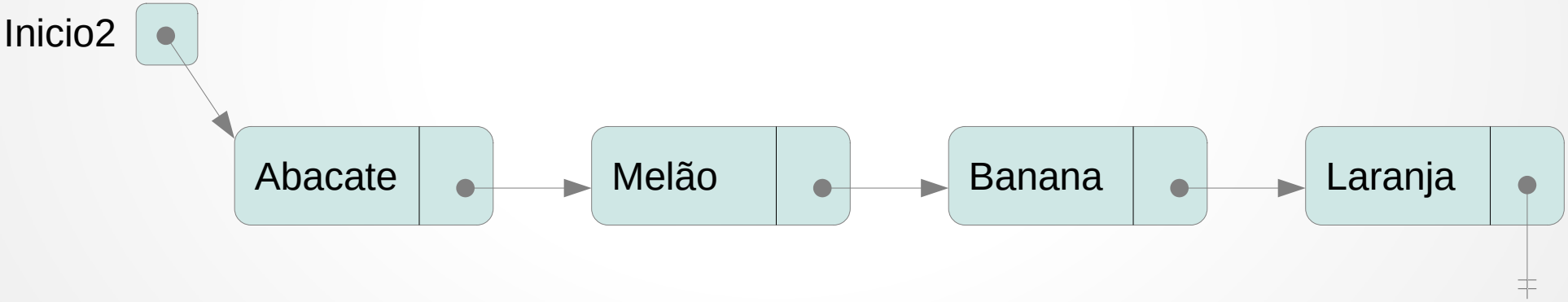
- Para fins didáticos, podemos dizer que a ideia é construir estruturas que lembram vetores, mas sem tamanho definido.
- Também não teremos índices, sendo necessário percorrer a lista desde o seu início.

Listas Encadeadas Simples

Um exemplo de uma lista vazia



Um exemplo de uma lista com vários Nós



O último Nó apontará para null

Listas Encadeadas Simples

- A primeira tarefa é definir a classe com os dados que queremos guardar na lista.
- Como cada Nó aponta para o próximo, o último campo da classe sempre será uma referencia para ela mesma.
  - Pode parecer estranho, mas funciona. Veja um exemplo no próximo slide.

Listas Encadeadas Simples

- Exemplo de definição de uma classe para trabalhar com listas simplesmente encadeadas, para armazenar dados de pessoas:

```
class NoPessoa {
    String CPF;
    String nome;
    double salario;
    NoPessoa proximo;
}
```

Listas Encadeadas Simples

- Definida a estrutura de dados a serem guardados, no programa principal devemos criar:
  - Uma variável para guardar o início da lista;
  - Uma ou mais variáveis auxiliares para percorrer a lista ou fazer outros processamentos.

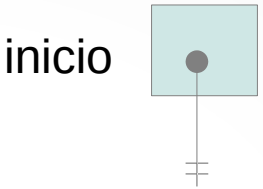
- Vejamos alguns exemplos de operações e sua representação na memória. Utilizaremos como base a classe `NoPessoa` do Slide anterior.

Alocação Dinâmica de Memória em Java

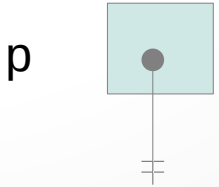
Código fonte em Java:

```
NoPessoa inicio, p;
```

RAM:



Ambos apontam para null

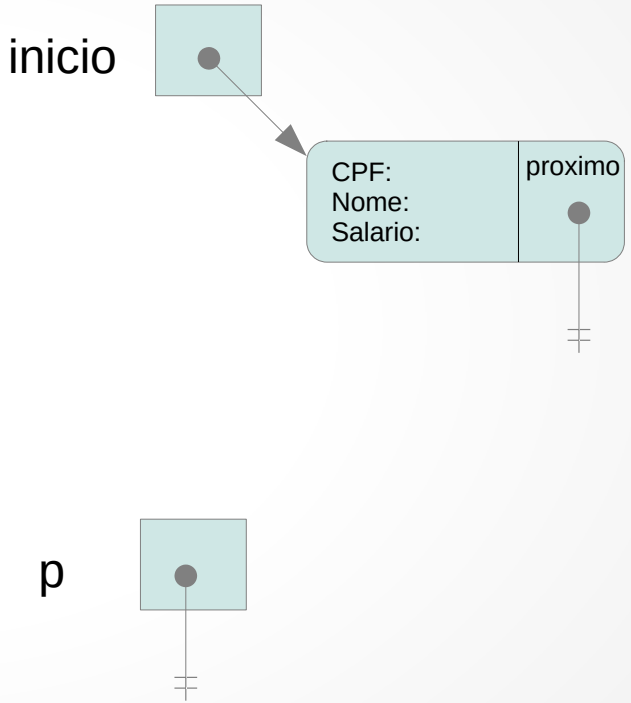


Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
NoPessoa inicio, p;
inicio = new NoPessoa();
```

RAM:



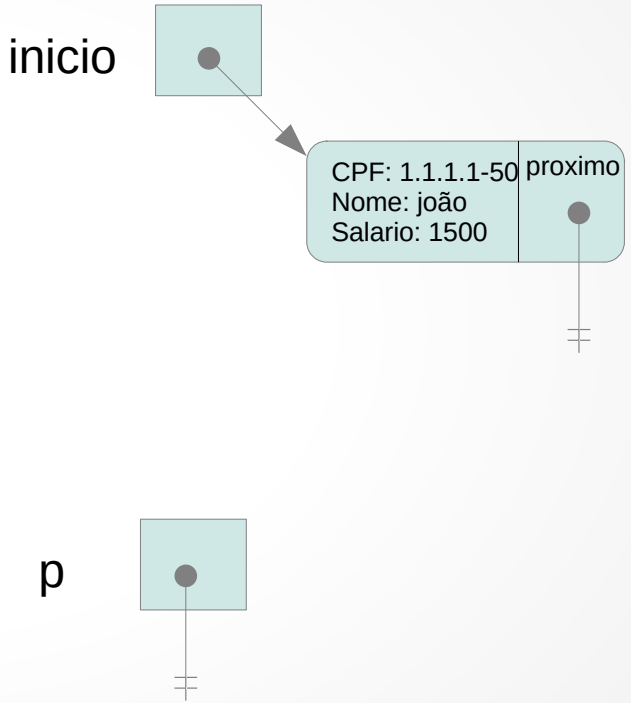
Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
NoPessoa inicio, p;
inicio = new NoPessoa();

inicio.CPF = "1.1.1.1-50";
inicio.nome = "joão";
inicio.salario = 1500;
```

RAM:



Alocação Dinâmica de Memória em Java

Código fonte em Java:

```
NoPessoa inicio, p;
inicio = new NoPessoa();

inicio.CPF = "1.1.1.1-50";
inicio.nome = "joão";
inicio.salario = 1500;

p = new NoPessoa();
p.CPF = "2.2.2.2-55";
p.nome = "maria";
p.salario = 3000;
p.proximo = inicio;
inicio = p;
```

RAM:

Consegues determinar o que acontece na memória RAM???

DESENHE NO SEU CADERNO.

Tende DESENHAR antes de seguir.

Vá passo a passo, refazendo os desenhos que eu já passei.

É importante desenhar para compreender o que acontece. Lembre de ir fazendo e desenhando na mesma ordem que está o código fonte.

Resposta no próximo Slide.

Alocação Dinâmica de Memória em Java

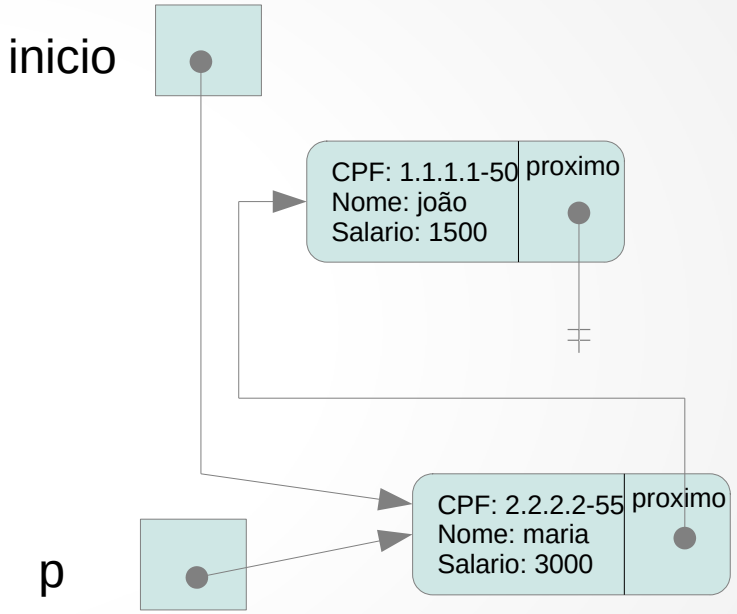
Código fonte em Java:

```
NoPessoa inicio, p;
inicio = new NoPessoa();

inicio.CPF = "1.1.1.1-50";
inicio.nome = "joão";
inicio.salario = 1500;

p = new NoPessoa();
p.CPF = "2.2.2.2-55";
p.nome = "maria";
p.salario = 3000;
p.proximo = inicio;
inicio = p;
```

RAM:



Foi feita uma inserção no início da lista.

Alocação Dinâmica de Memória em Java

- Exercício / Atividade
  - Utilizando o projeto “ExemploListaEncadeada” fornecido pelo professor (em Java, no NetBeans), crie duas novas sub-rotinas:
    1) Inserir no Final: sub-rotina para inserir elementos no final da lista;
    2) Procurar: sub-rotina para procurar se determinado nome aparece na lista.

ExemploListaEncadeada

```
package exemplolistaencadeada;

import java.util.Scanner;

/**
 *
 * @author Prof_Marcelo_Studier
 */
public class NoPessoa {

    String cpf;

    String nome;

    double salario;

    NoPessoa proximo;
}


public class ExemploListaEncadeada {

    static NoPessoa inserirNoInicio(NoPessoa no, String cpf, String nome, double sal ){

        NoPessoa p = new NoPessoa();

        p.cpf = cpf;

        p.nome = nome;

        p.salario = sal;

        p.proximo = no;

        return p;
    }



    static void imprimir(NoPessoa no){

        NoPessoa p;

        p = no;

        while (p.proximo != null) {

            System.out.println("CPF = " + p.cpf + " - p.nome = " + p.nome);

            p = p.proximo;

        }
    }


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        // TODO code application logic here

        NoPessoa lista, p;

        lista = new NoPessoa();
```