UFMS DIGITAL

Programação Orientada a Objetos

Prof. Dr. Anderson V. de Araujo

UFMS DIGITAL

Módulo II - Aprofundando na Orientação a Objetos

Unidade II - Métodos, Classes e objetos: uma visão mais aprofundada

Declaração de Variáveis

- Variáveis servem para armazenar valores
- Existem dois tipos de variáveis:
  - Tipos Primitivos:
    - `char, boolean, byte, short, int, long, double ou float.`
  - Variáveis de Referência:
    - Usadas para se referir e acessar um objeto.

Variáveis

- Tanto tipos primitivos quanto variáveis de referência podem ser declaradas como Atributos, Variáveis locais e Parâmetros.
- Atributos:
  - São variáveis declaradas dentro da classe, mas fora de qualquer método. Possuem valores padrão de inicialização;
  - Podem ser Atributo da Classe ou Atributo da Instância (do Objeto).
- Variáveis locais:
  - Variáveis declaradas dentro de um método/bloco de código;
  - É obrigatório inicializá-las antes de usar.
- Parâmetros:
  - São variáveis declaradas dentro da assinatura de um método;

Variáveis Locais

- São aquelas declaradas e inicializadas dentro de um método/escopo;
  - E destruídas dentro do método/escopo;
- Com isso, não podem ser acessadas em um código fora de onde foram declaradas;
- Embora o valor da variável local possa ser passado para outro método, armazenando seu valor em uma variável de instância, a variável propriamente dita só vive dentro do escopo do método.

Variável Local - Exemplos

```
class Teste {
   int valor;

   void foo() {
      int contador = 10;
        foo2(contador);
   }
   void foo2(int c) {
        valor = c;
  }
   void doo(int i) {
        contador = i;
     /* não vai funcionar, pois não é possível acessar
contador de fora do método foo() */
   }
}
```

Valores Padrão para Variáveis de Instância

Tipo	Valor Inicial/Padrão
`boolean`	`false`
`byte`	`0`
`short`	`0`
`int`	`0`
`long`	`0l`
`char`	`\u0000`
`float`	`0f`
`double`	`0d`
`Object`	`null`

Variáveis de Classe

- São definidas dentro da classe, mas fora de qualquer método;
- Devem ser precedidas pelo modificador **`static;`**
- Pertencem à classe onde foram declaradas, consequentemente a todos os objetos instanciados a partir daquela classe:
  - Podem ser utilizadas sem ter que instanciar a classe.

Métodos de Classe

- São métodos que estão associados diretamente com a classe e não com cada instância:
  - Podem ser utilizados sem ter que instanciar a classe.
- Devem ser precedidos pelo modificador **`static`**
- E o método **`main`**??
- IMPORTANTE: Métodos e atributos da classe (**`static`**) não podem acessar métodos e atributos da instância

Escopo de uma Variável

- Por quanto tempo uma variável existe?
  - As variáveis `static` têm vida mais longa. São criadas quando a classe é carregada;
  - As variáveis de instância são criadas quando uma nova instância é criada;
  - As variáveis locais existem durante o tempo em que o seu método permanecer na pilha de chamadas;
  - As variáveis de bloco só existem durante o tempo em que o bloco de código está sendo executado;

Método Construtor

- Toda vez que a palavra reservada new for usada, um método construtor vai ser chamado:
  - Ele não “constrói” o objeto;
  - Serve como um método qualquer, mas em geral, serve para inicializar os atributos do objeto recém-criado;
    - Mas ele não inicializa os atributos automaticamente, você tem que fazer
  - Pode ter parâmetros
- Se uma classe não possuir um construtor, a JVM cria um construtor padrão (sem parâmetros) para a classe automaticamente

Método Construtor - Exemplo

```
class Pessoa {
 String nome;
 int idade;

 Pessoa() {//Construtor "Padrão"
 }

 Pessoa(String novoNome, int novaIdade) {
    nome = novoNome;
    idade = novaIdade;
      //Qualquer outro código
 }
}
```

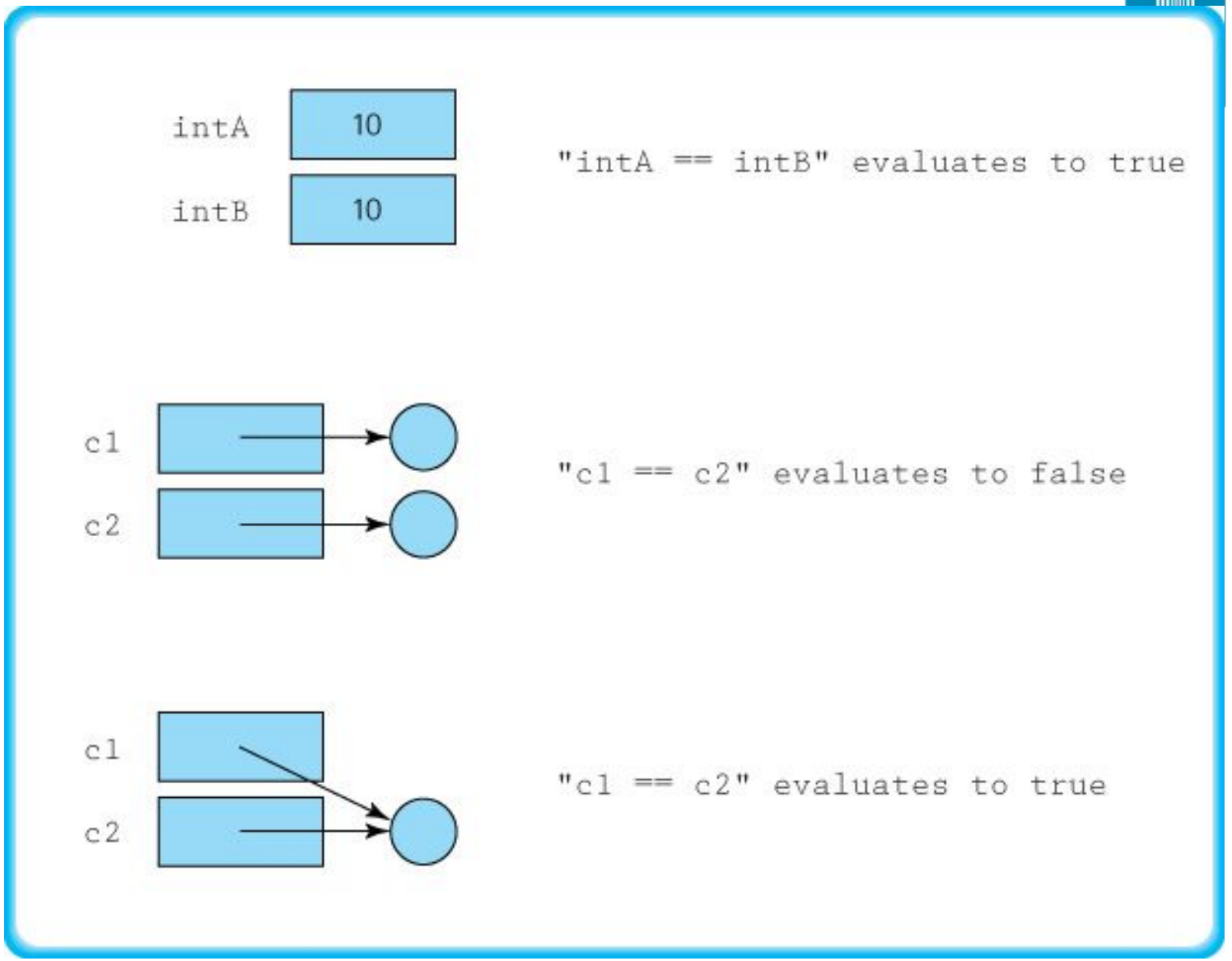
Método Construtor – Exemplo 2

```
class Pessoa2 {
      String nome = "Fulano";
      int idade = 15;

      Pessoa2(String novoNome, int novaIdade) {
            nome = novoNome;
            idade = novaIdade;
      }

    public static void main(String[] args) {

        Pessoa2 p = new Pessoa2("Sicrano", 32);

        System.out.println(p.nome + p.idade);

    }
}
```

Executado antes do construtor!

O que é impresso?

Como Comparar Elementos?



Comparando

Operador ==

- Compara os valores em variáveis de tipos primitivos
- Em objetos, serve para comparar referências, inclusive com null
  - Retorna true se as variáveis de referências apontam para o mesmo objeto

MÉTODO equals

- É um método da superclasse Object
- Deve-se sobrescrever o método equals para proceder de acordo com a regra de igualdade
  - Por exemplo, na classe String, o equals é sobrescrito para comparar caractere por caractere, verificando se ambas as strings têm todos caracteres iguais
- Se não for definido pela classe, funciona exatamente igual o operador ==

Licenciamento