04

Variáveis guardam valores

Transcrição

Um último detalhe muito interessante sobre estas variáveis do tipo primitivo - todas aquelas que vimos exceto a `String` - é seu funcionamento interno. O que são guardadas na memória delas?

Vamos criar mais uma classe, o `TestaValores` . E para não ficarmos digitando `public static void main(String[] args) {}` à mão o tempo todo, aprenderemos um atalho. Digitaremos "main" e apertaremos "Ctrl + barra de espaço" que, assim como em outros editores, tem a ver com o *autocomplete*. No Eclipse, também envolve ***templates***.

Por meio deste atalho, aparecerão algumas opções, apertaremos a tecla "Enter", e o código aparece pronto no editor de texto. Isso passará a ser frequente para vocês.

Para entendermos como é guardado o valor de uma variável no Java, a **passagem por valor**, vamos fazer um desafio:

```
public class TestaValores {

    public static void main(String[] args) {
        int primeiro = 5;
        int segundo = 7;

        System.out.println(segundo);
```

Ao rodarmos o código, obteremos `7` .

```
public class TestaValores {

    public static void main(String[] args) {
        int primeiro = 5;
        int segundo = 7;
        segundo = primeiro;

        System.out.println(segundo);
```

Salvando e rodando este código, obteremos `5` !

```
int primeiro = 5;
int segundo = 7;
segundo = primeiro;
primeiro = 10;

// quanto vale o segundo?

System.out.println(segundo);
```

No `segundo` , tínhamos guardado o `primeiro` , mas agora `primeiro` vale `10` . Quanto vale `segundo` ?

As linguagens de programação trabalham de formas diferentes dependendo do uso de um símbolo específico, ou da existência de alguma referência, e por aí vai. Estas variáveis do tipo primitivo são trabalhadas com o valor do conteúdo, da variável, então, quando copiamos `5` para dentro de `segundo`, e depois copiamos `10` para `primeiro`, a linha `segundo = primeiro;` não diz nada.

Quando se faz uma atribuição no Java, não se diz que uma variável **sempre** segue a outra, e sim que estamos **copiando e colando valores**. Deste modo, `primeiro = 10;` não surtirá efeito para `segundo`. Confirmaremos isto rodando a aplicação, pois continuaremos recebendo `5`.

Isso significa que a variável guarda um valor, e não uma referência, e este exemplo dará base para as entendermos melhor.

Estamos prontos para o próximo passo, que consiste em finalmente começarmos com controle de fluxos, com `if`, `while` e `for`, para estruturarmos nossos primeiros programas! E então veremos a orientação a objetos (O.O.) de maneira contra procedural. Vamos lá?