04

Percorrendo um laço de repetição

Transcrição

Paramos no método `calcularPrecoFinal()`, porque achamos que o erro está ali. Vamos tentar debugar cada uma das linhas para entendermos exatamente o que está acontecendo neste método, e o que ele nos retorna, repetindo todos os passos vistos no vídeo anterior.

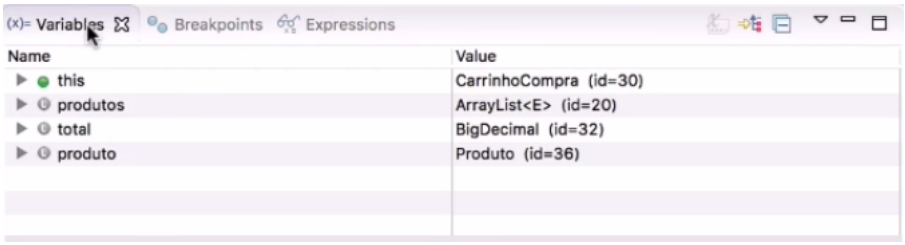
Primeiramente criaremos a variável `BigDecimal total`, que por sua vez criará um `BigDecimal` zerado, apenas para sua inicialização. Seguiremos a partir disso. Há um laço de repetição `for each`, que utiliza cada um dos `produtos` recebidos como argumento, percorrendo-os. Quando estes laços ocorrem, o importante é o escopo, tanto de método quanto de laço.

Como exemplo, aproveitaremos o próximo passo. Quando a aplicação começar a andar no laço, apertaremos o `F6`, nos encontrando no escopo do laço:

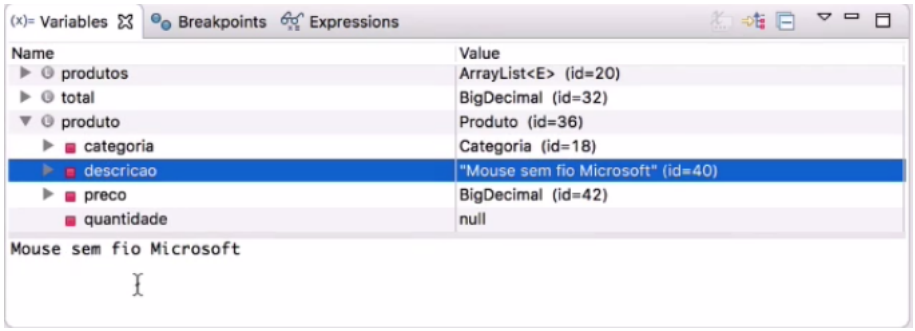
```
BigDecimal preco = produto.getPreco();
```

Após o término dos laços, todas as variáveis criadas serão apagadas e posteriormente reiniciadas (caso o laço se repita). No exemplo acima, a variável `preco` é criada, pegando-se a variável `produto`, para cada item proveniente de `lista`.

A aba chamada "*Variables*" (ou "Variáveis") é muito importante quando se está debugando uma aplicação, sendo localizada ao lado superior direito, contendo todas as variáveis no escopo naquele determinado momento:



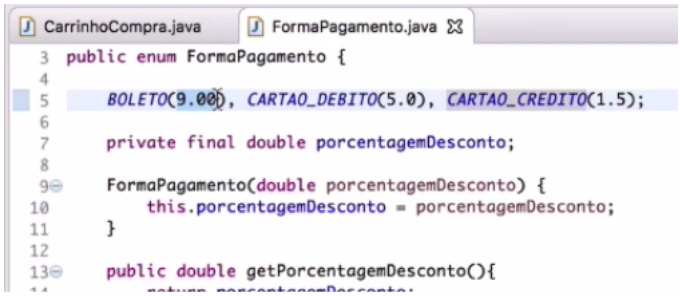
Clicando-se em cada uma delas, obteremos maiores informações, como os objetos que estão dentro deste array. É possível navegar por essas variáveis: o `produto`, por exemplo, é um objeto que contém vários atributos, visíveis quando se clica na seta ao seu lado.



Neste escopo específico, foram criados estes produtos, com atributos navegáveis. Sabendo que este ponto é bastante importante por nos mostrar as variáveis naquele exato momento, voltaremos à navegação. O `preco` foi criado pelo `BigDecimal`, e surge a partir do `produto.getPreco()`. Apertaremos `F6` para que essa variável seja populada.

Executando-se assim o método `getPreco()`, joga-se o resultado para dentro da variável `preco`. Quando consultamos seus detalhes, mostra-se o valor "`120`", que é o "*Value*" `BigDecimal`. O `produto`, neste caso, é o mouse, e ele custa "`120`".

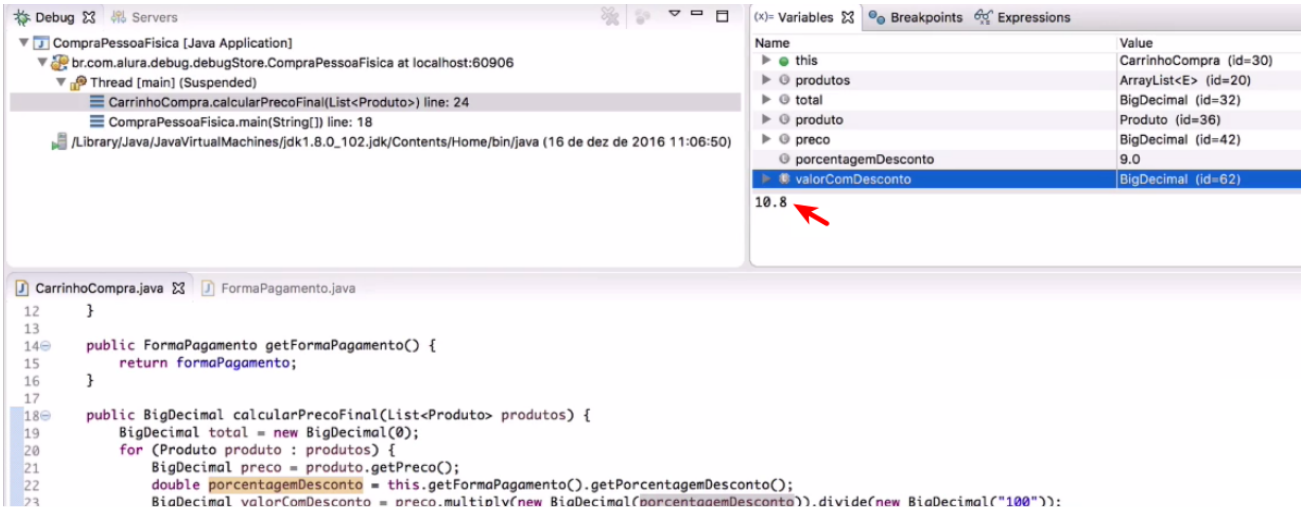
O `porcentagemDesconto` é um `double` que utiliza o `this` do método `getFormaPagamento()`, e a porcentagem (`getPorcentagemDesconto()`). Lembra das três formas de pagamento? Boleto, cartão de crédito e de débito? Cada uma delas possui uma porcentagem de desconto. Vamos entrar nesta classe para vermos mais detalhes, apertando "Cmd" (no Mac), ou "Ctrl" (no Windows ou Linux) selecionando `FormaPagamento`.



Estes números indicados entre parênteses após a forma de pagamento são as porcentagens correspondentes. Pagando-se a compra com boleto, o desconto é de `9%`, por exemplo. Voltaremos à aba do carrinho de compras, acessando o método em que estávamos antes, apertaremos `F6` para pularmos à próxima linha a ser executada, criando-se `porcentagemDesconto`.

Na aba "Variáveis", verificamos as informações referentes a ela, sabendo-se que o valor é de `9.0` e que se trata de um `double`. O `valorComDesconto` é um `BigDecimal`, obtido a partir do `preco` ("`120`"). Multiplicamos este valor por `porcentagemDesconto` e dividimos o resultado por `100`, devido ao percentual de desconto.

Executamos, vamos à próxima linha e o valor de desconto é executado (`10.8`), significa que para aquele produto nesta forma de pagamento, o desconto obtido é de `R$ 10,80`.



O cálculo acima está correto, pois utiliza-se `valorComDesconto` e adiciona-se o resultado obtido a `total`. As variáveis são zeradas, apagadas, porém o valor total não foi apagado pois foi criado fora do escopo. Percorreremos as linhas novamente.

Clicaremos em "*Step over*", indo à próxima linha. Indo a "produto" na aba "*Variables*", temos a "descricao", que é "Teclado de Gamer Alien". Qual seu preço? " `350` ". Trata-se de outro escopo, com outras variáveis. Iremos à próxima linha, em que se pega o preço, a `porcentagemDesconto` (`9.0`), com `valorComDesconto` (`31.5`) resulta em `total` (`10.8`).

Apertando-se `F6` , as variáveis são novamente zeradas, e o resultado total é de `42.3` . Continuando, o preço (`250`), a porcentagem de desconto (`9.0`), e o valor com desconto (`22.5`) são calculados. Rodaremos a aplicação mais uma vez, e temos o produto (processador) com valor `1500` , porcentagem de desconto `9.0` , e valor com desconto `135` .

A cadeira é o último produto desta lista, à qual passaremos linha a linha. O preço dela é `759` , o valor com desconto é `68.31` . Terminados todos os cálculos, a aplicação sai do laço automaticamente, retornando o valor total (`268.11`).

Logo, vê-se que o problema se dá neste momento: a aplicação está somando todos os valores dos produtos com descontos, sendo que o correto seria diminuir o valor com desconto do preço de cada produto, e aí, sim, somar todos os resultados.

Temos aquelas mesmas opções vistas no vídeo anterior, deixar a aplicação correr normalmente, seguindo passo a passo até chegar ao método `main` novamente, mesmo estando errado, ou a aplicação pode ser pausada neste instante, pois já sabemos onde está o erro, então podemos corrigi-la. Faremos isto, apertando o `F2` , e agora precisamos acrescentar duas linhas de código para corrigir este *bug*. É necessário pegar o preço do produto, fazendo-se a subtração deste valor pelo preço com desconto:

```
BigDecimal valorComDesconto = preco.multiply(new BigDecimal(porcentagemDesconto)).divide(new B
BigDecimal valorProdutoComDesconto  = preco.subtract(valorComDesconto);
total = total.add(valorProdutoComDesconto);
```

Rodaremos a aplicação novamente no modo *Debug*, para confirmar que o erro foi solucionado. Irei ao método `main` , clicarei com o botão direito do mouse em cima da classe, e depois em "Debug As > Java Application". Navegarei linha a linha usando o atalho `F6` , chegando à linha desejada de cálculo do valor total da compra. Feito isto, entraremos nela apertando `F5` (ou "*Step into*"), continuando a rodar linha a linha.

No caso do mouse, o desconto é de `10.8` , a partir da qual obteremos o valor final com desconto, `109.2` . Repetiremos estes passos para todos os demais produtos e, caso se pare a aplicação em alguma destas etapas, não teremos o resultado final. Então queremos que a aplicação continue rodando, executando-se os *loops* sem parar, até chegarmos ao resultado final. Quando isto ocorre, clicaremos em "*Resume*" (`F8` no teclado), e o valor final correto é mostrado: `2710.89` .