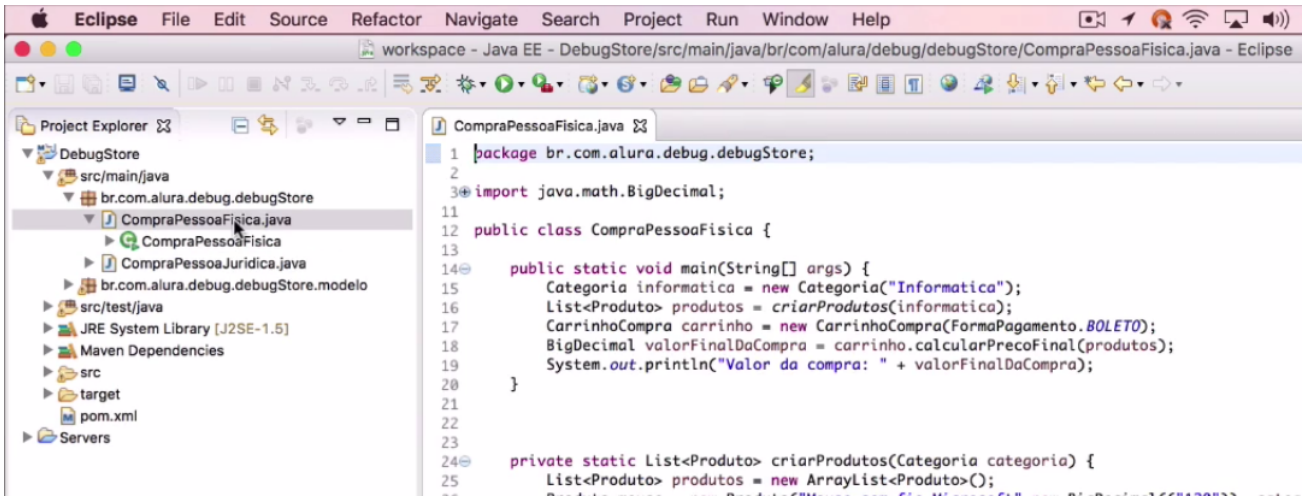
02

# Nosso primeiro breakpoint a gente nunca esquece

## Transcrição

A partir de uma aplicação em Java, entenderemos o processo de depuração (ou ***debug***, em inglês). Temos esta app simulando um e-commerce com várias categorias: informática, perfumaria, entre outras, como o Americanas (http://www.americanas.com.br/) e o Walmart (https://www.walmart.com.br/). Simularemos uma compra de uma pessoa física e de uma pessoa jurídica.

No primeiro pacote, temos duas classes: pessoas físicas e jurídicas. Começaremos pela pessoa física, clicando no arquivo correspondente que se encontra do lado esquerdo ( `CompraPessoaFisica.java` ).




Trata-se de uma aplicação que está rodando a partir de um método `main` , não de uma aplicação web. Digamos que esteja mais para desktop, porém sem interface, a parte gráfica.

Há o método `main` , principal, que roda toda a aplicação, e aquele que cria os produtos listados para a compra de um PC *gamer* (teclado, mouse, monitor, e demais produtos), e seus preços: o mouse custa `R$120` , o teclado, `R$350` , o monitor, `R$250` , e assim sucessivamente. As categorias em que eles se encontram são passadas como argumento, e a categoria é criada no `main` , " `Informática` ".

Vamos rodar esta aplicação para ver qual o valor total desta compra. Para isto, basta clicar em cima da classe com o lado direito do mouse, depois em "Run as > Java Application". Feito o processo, o resultado foi retornado ( `R$268,11` ), um valor estranho, levando-se em consideração todos os valores dos produtos individualmente. Então, tem algo errado nessa aplicação.

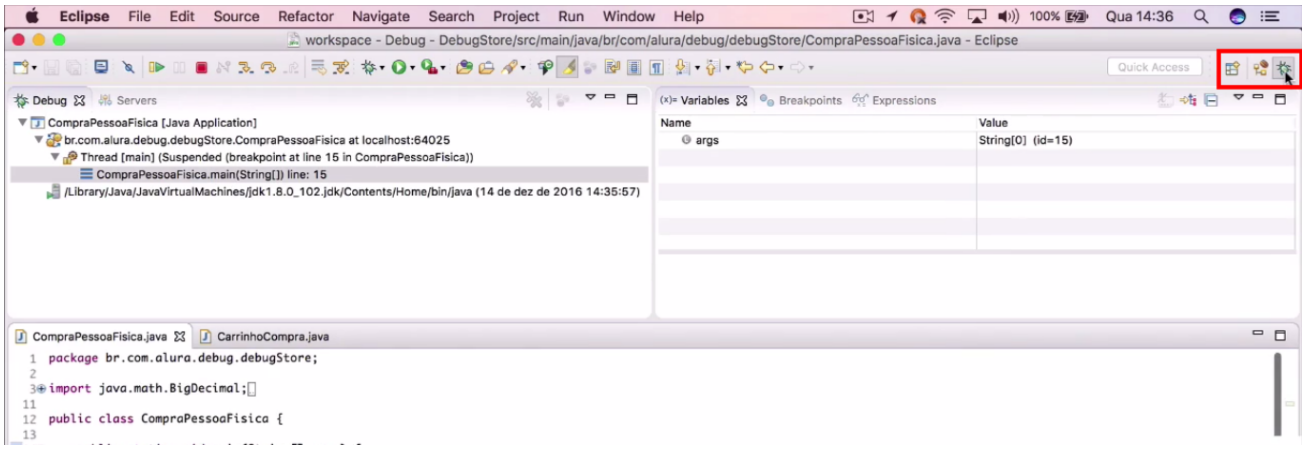
Quando se utiliza um código legado, de outra pessoa, e ele está incompreensível, mas é preciso arrumar algum problema, é muito comum "debugarmos" a aplicação para sabermos onde se encontra o erro exatamente.

O modo *debug* possibilita pausarmos a aplicação em um determinado ponto. Faremos isto, clicando em cima da classe com o botão direito do mouse, indo à opção "Debug As > Java Application". Feito isto, a aplicação não parou em nenhum momento.

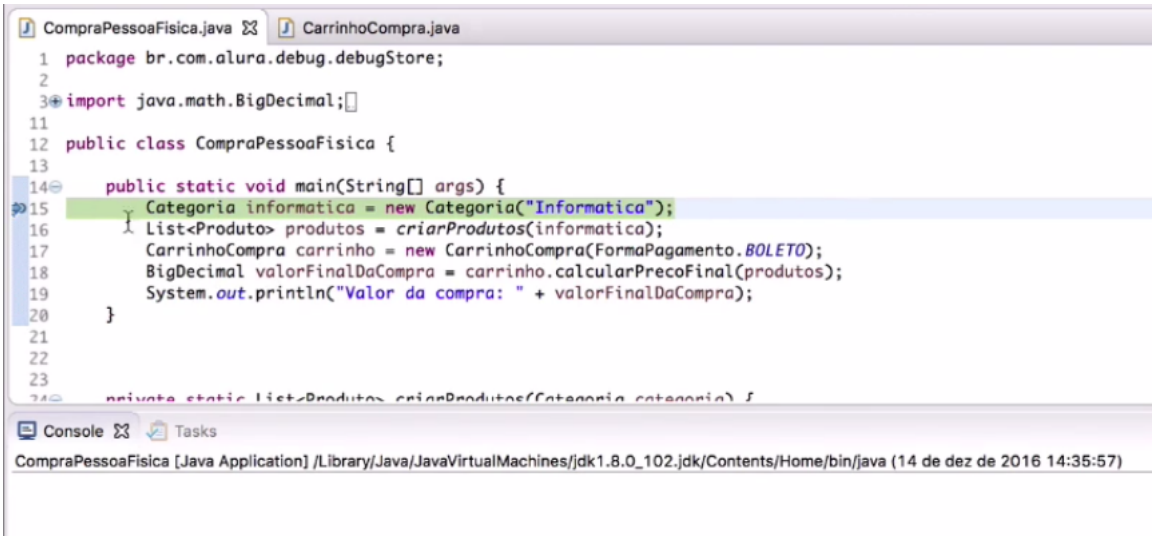
Não informamos ao nosso IDE em qual ponto gostaríamos que a aplicação fosse pausada, ou seja, não determinamos o *breakpoint*. Há duas maneiras de se fazer isto: clicando duas vezes ao lado da linha, criando uma bolinha, ou usando o atalho de teclado "Cmd + Shift + B" (no MAC) ou "Ctrl + Shift + B" (no Windows ou Linux), com a linha desejada selecionada.

A bolinha sinaliza o exato momento em que a depuração deve parar. O ponto de partida de um *debug* é saber onde colocar o *breakpoint* para que na hora em que aplicação rodar, parar ali. Como sabemos que a porta de entrada é o método `main`, vamos rodar novamente a aplicação em modo *debug*.

Desta vez, aparece uma mensagem perguntando se queremos trocar a perspectiva Java, que está sendo utilizada no momento, para a Debug. Clicaremos em "Yes" para isso, e o layout é alterado, disponibilizando-se outras opções e ferramentas. As perspectivas são configuradas no canto superior direito:




A aplicação, após a pausa, fica aguardando nova ação de nossa parte, mantendo selecionada a linha em que está:




Veremos melhor sobre isso durante o curso, agora, nosso objetivo foi atendido, de parar a aplicação em determinado ponto ou *breakpoint*. Até a próxima!