



Desenvolvimento de Sistemas

Ferramentas de depuração passo a passo

No contexto de testes, ferramentas que permitam a análise da execução de um código são bem-vindas, especialmente como testes de caixa-branca. Elas permitem a análise do que está acontecendo no código em tempo real.

O processo de depuração complementa o processo de testes. Enquanto o teste identifica os erros que acontecem no programa e os efeitos de uma falha, a depuração (ou *debugging*) serve para localizar onde o erro se origina e planejar como corrigi-lo.

Felizmente, os IDEs (*integrated development environment*, ou ambiente de desenvolvimento integrado), desde 1983, com o lançamento do Borland Turbo Pascal, trazem ferramentas que permitem, baseando-se no ambiente de desenvolvimento, um processo de depuração, incluindo pontos de parada no código, executando passo a passos as instruções programadas e permitindo a visualização dos valores guardados em memória. Esses recursos estão presentes e aprimorados em IDEs modernos, como o NetBeans.

A seguir, veja os comandos presentes para a depuração no NetBeans e suas utilizações.

Recursos de IDE

De maneira sucinta, é assim que se realiza a depuração automatizada com NetBeans IDE:

1. Marcar um *breakpoint* (ponto de parada) na linha de código desejada. Para fazer isso, deve-se clicar, na barra lateral onde ficam os números das linhas, na linha em que se deseja que a execução pause (ou seja, o programa executará todas as instruções que acontecerem antes da marcada com *breakpoint*).

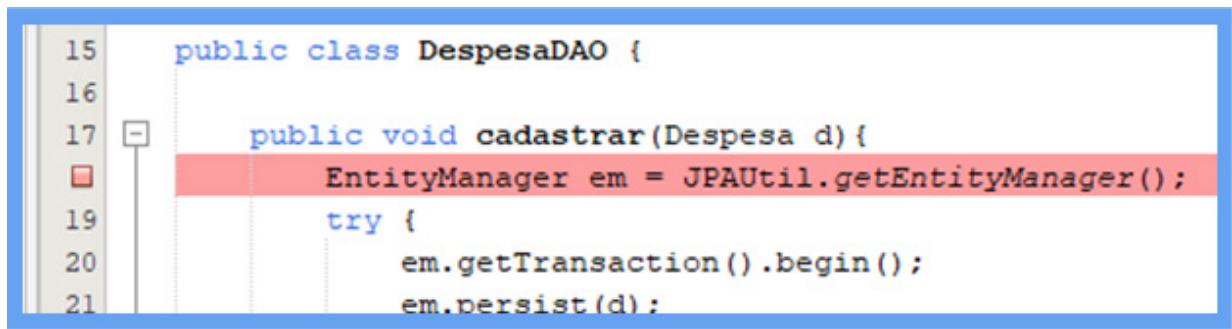


Figura 1 – *Breakpoint* posicionado na linha 18 do código

Fonte: Senac EAD (2023)

2. Acionar a depuração pelo ícone **Debug Project** na barra principal de ferramentas.

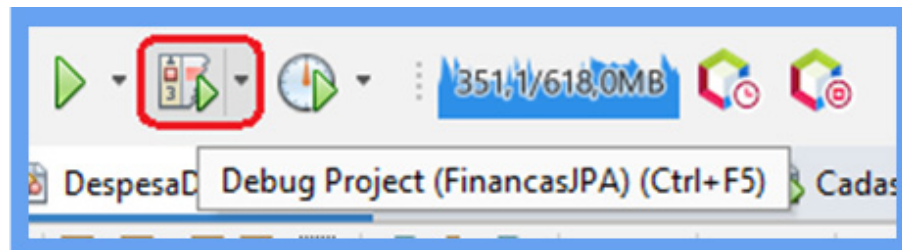


Figura 2 – Ícone **Debug Project**

Fonte: Senac EAD (2023)

3. Acionar a barra de ferramentas de *debug*, o que acontece quando se aciona o *debug*. Em breve, cada um desses recursos será lembrado.

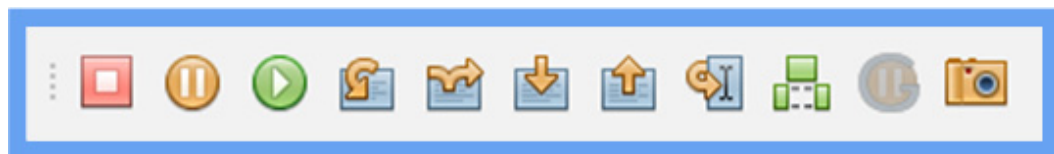


Figura 3 – Ferramentas de *debug* do NetBeans

Fonte: Senac EAD (2023)

4. Avançar linha a linha por meio das ferramentas de *debug*.

No momento em que está pausada a execução em um *breakpoint*, é possível utilizar um dos recursos essenciais da depuração: a visualização dos valores em memória. Para isso, pode-se passar com o *mouse* sobre o nome de uma variável ou um objeto, e um painel *pop-up* abrirá com os valores.

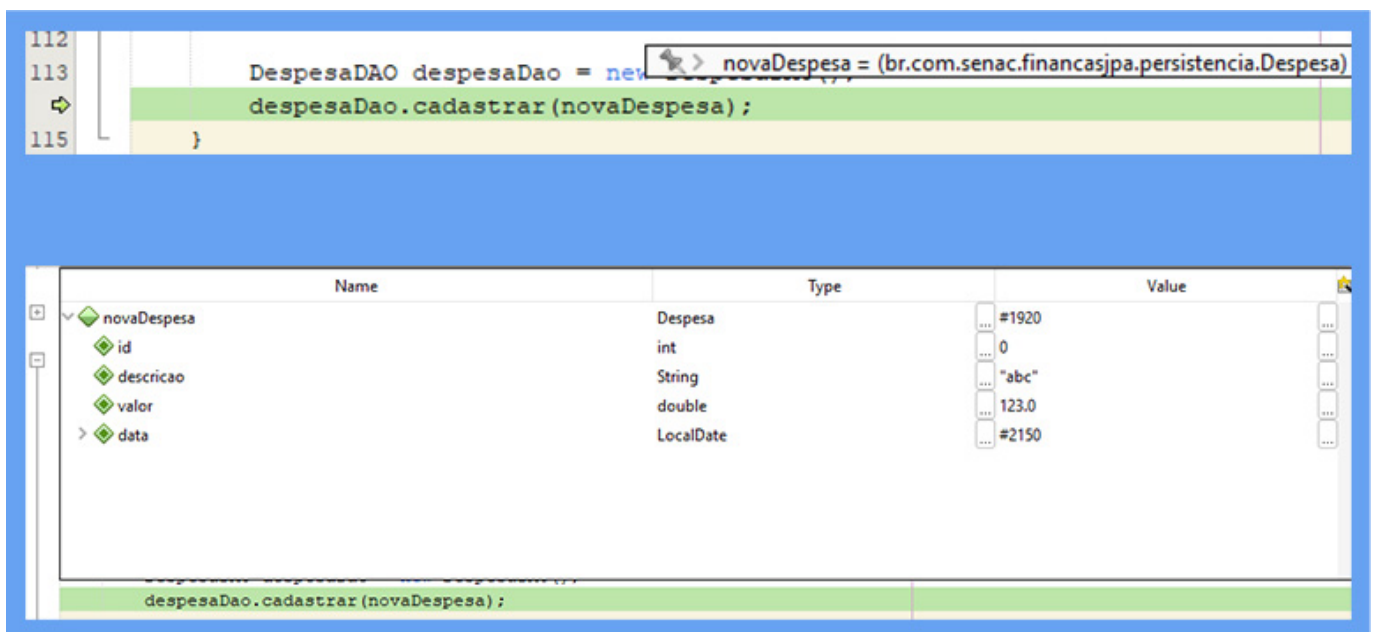


Figura 4 – Visualização de valores no editor do NetBeans. Abaixo, a visão expandida, mostrando os valores dos atributos do objeto **novaDespesa**

Fonte: Senac EAD (2023)

As variáveis podem ser observadas também no painel **Variables**, no qual é possível cadastrar um observador (ou *watch*) para uma variável ou um objeto. Por padrão, o painel, que fica na parte inferior da janela do NetBeans, mostra as variáveis presentes no escopo atual da execução.

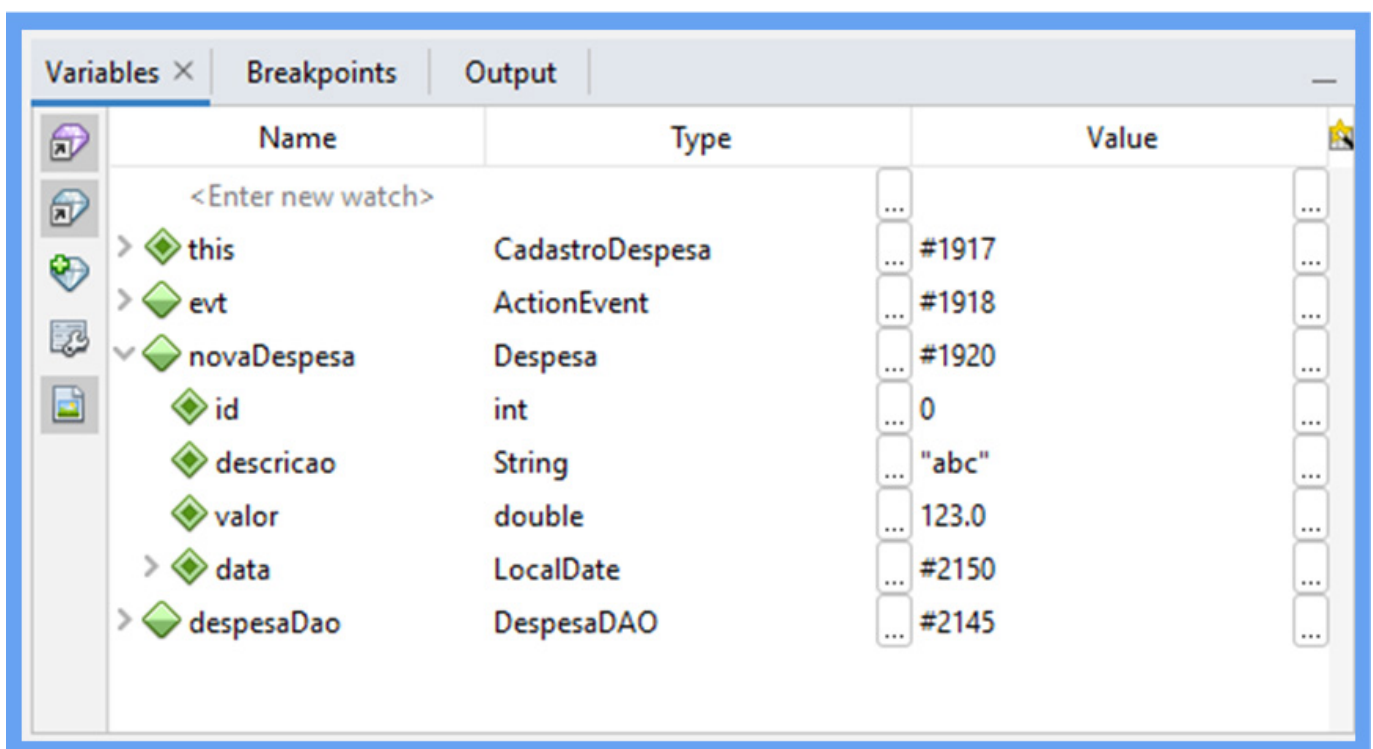


Figura 5 – Aba **Variables** no NetBeans

Fonte: Senac EAD (2023)

Contudo, também é possível incluir um novo *watch*, ou seja, incluir um monitoramento para alguma variável, algum atributo ou um objeto qualquer. No campo **<Enter new watch>**, basta incluir a expressão que se quer monitorar. Isso também pode ser feito por meio do ícone de diamante com um sinal de + (**Create new watch**).

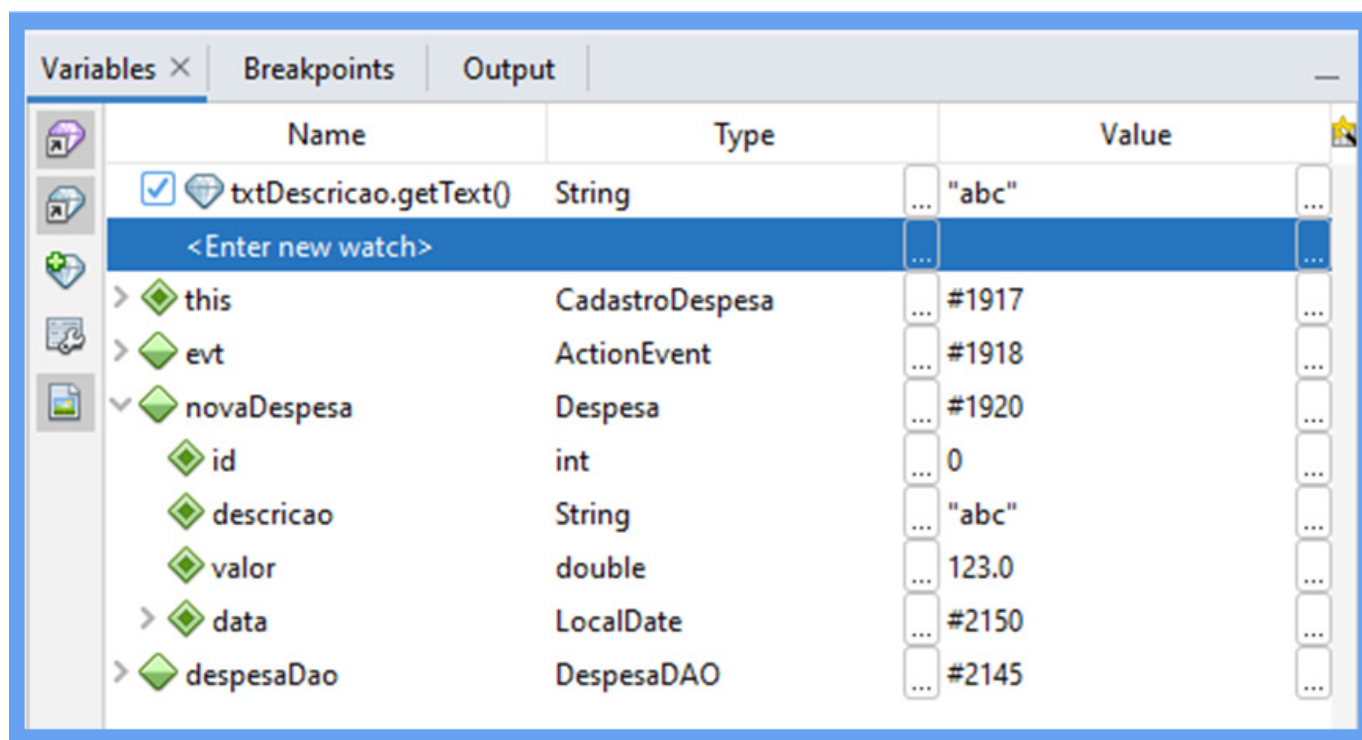


Figura 6 – Incluindo novo *watch* para a expressão **txtDescricao.getText()**

Fonte: Senac EAD (2023)

A expressão não precisa ser necessariamente um nome de variável. Pode ser a invocação de um método, por exemplo.

Note, ainda, que na região inferior da tela há uma aba **Breakpoints** em que se pode monitorar todos os pontos de parada incluídos no projeto.

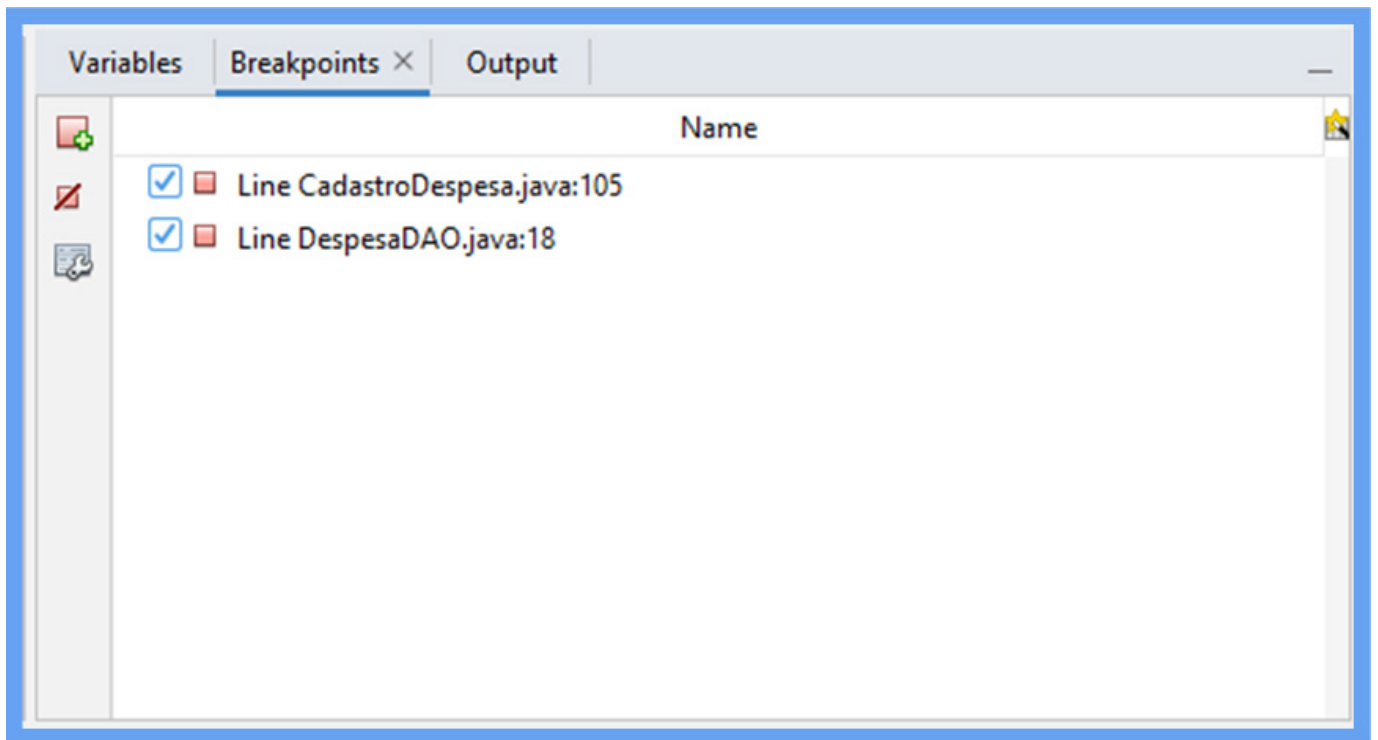


Figura 7 – Aba **Breakpoints** no NetBeans

Fonte: Senac EAD (2023)

Comandos e utilização

A tabela a seguir resume os comandos presentes na barra de ferramentas de depuração e quando eles devem ser usados.












Botão	Descrição	Quando usar
 Finish Debugger Session	Encerra a sessão de depuração	Ao encontrar uma falha, permite continuar a execução até o fim ou simplesmente encerrar o programa usando esse botão.
 Pause	Pausa a execução	Se um programa está executando, mas ainda não atingiu um ponto de parada, esse recurso pode ser usado para pausar sua execução.
 Continue	Segue a execução	Usado para, caso seja necessário, seguir a execução sem ir passo a passo até atingir outro <i>breakpoint</i> posterior ou o fim do programa.
 Step Over	Pular para a próxima linha	Segue a execução para a linha imediatamente posterior à atual.
 Step Over Expression	Pular expressão	Executa expressão a expressão de uma linha. Por exemplo: em.getTransaction().begin(); O comando passaria primeiro pelo getTransaction() e depois pelo begin() .
 Step Into	Entrar na execução de uma expressão	Executa o passo a passo de um método invocado pela linha atualmente em execução.
 Step Out	Sair da execução de uma expressão	Segue com a execução de um método sem ir passo a passo e retornar à linha onde esse método foi invocado.
 Run to Cursor	Pular para o cursor	Sem necessitar criar um novo <i>breakpoint</i> , pula a execução direto para a linha onde o cursor está posicionado.
 Apply Code Changes	Aplicar mudanças em código	Normalmente, é preciso encerrar a depuração para fazer uma alteração no código e depois reexecutar o <i>debug</i> para usar as alterações executadas. Esse recurso aplica as alterações de código sem parar a execução do <i>debug</i> .
 Toggle Pause in GraalVM Script	Pausar GraalVM	Usado para linguagens Ruby e JavaScript.
 Take GUI Snapshot	<i>Print</i> da tela do programa	Tira um <i>print</i> da tela em execução no projeto (JFrame).

Tabela 1 – Comandos presentes na barra de ferramentas de depuração

Fonte: Senac EAD (2023)

Encerramento



O recurso de depuração, presente hoje na maioria dos IDEs, é indispensável na rotina de um programador, pois ajuda a localizar exatamente o que causa uma falha no programa.

As ferramentas disponíveis no NetBeans são bastante abrangentes e permitem um processo bem detalhado de depuração, ajudando o desenvolvedor nessa missão.