

Debug no Eclipse

MC302 EF

Programação Orientada a Objetos

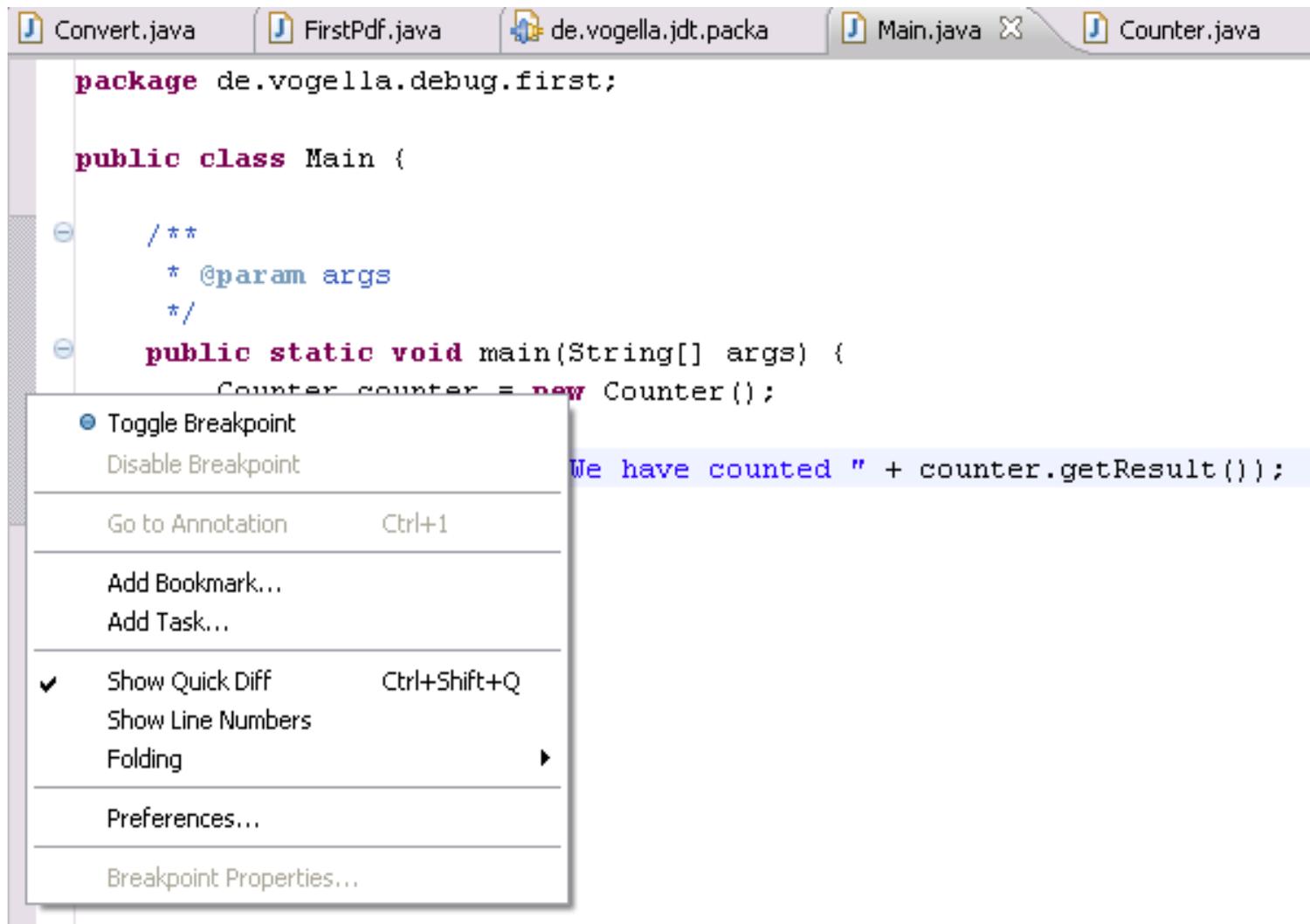
PED: Lucas Augusto Carvalho
lucas.carvalho@ic.unicamp.br

Prof. Fernando Vanini

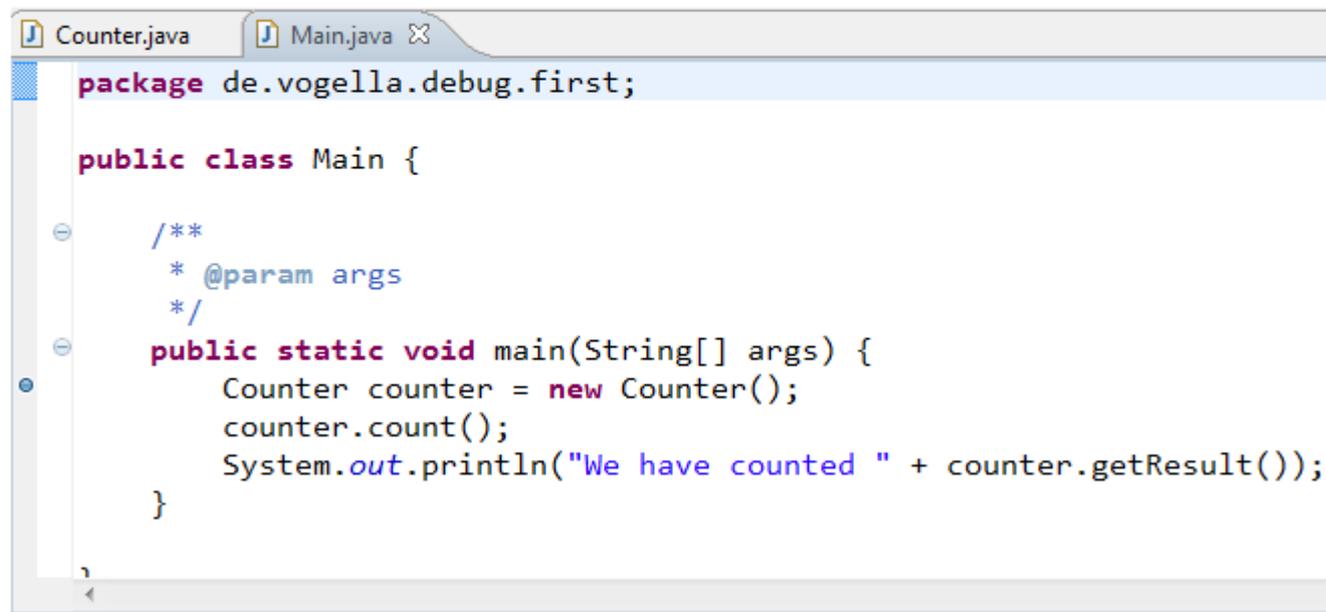
O que é Depuração?

- Permite a execução iterativamente de programas enquanto você observa o código fonte e as variáveis durante a execução.
- Breakpoint: ponto de parada na execução

Definindo Breakpoints



Definindo Breakpoints



```
Counter.java Main.java X
package de.vogella.debug.first;

public class Main {
    /**
     * @param args
     */
    public static void main(String[] args) {
        Counter counter = new Counter();
        counter.count();
        System.out.println("We have counted " + counter.getResult());
    }
}
```

Iniciando a Depuração

The screenshot displays an IDE interface with the following components:

- Package Explorer:** Shows a project structure with a context menu open over a file. The menu items include: New, Open (F3), Open With, Open Type Hierarchy (F4), Show In (Alt+Shift+W), Copy (Ctrl+C), Copy Qualified Name, Paste (Ctrl+V), Delete, Remove from Context (Ctrl+Alt+Shift+Down), Build Path, Source (Alt+Shift+S), Refactor (Alt+Shift+T), Import..., Export..., References, Declarations, Refresh (F5), Assign Working Sets..., Run As, Debug As (highlighted), Profile As, Validate, Team, Compare With, Replace With, Restore from Local History..., Web Services, Add Default Package, and Properties (Alt+Enter).
- Run and Debug Toolbar:** Located at the bottom, it contains icons for Run (a green play button) and Debug (a blue play button with a gear icon). The Debug icon is highlighted with a red box.
- Main Editor:** Displays the source code of a Java class named `Main` in the package `de.vogella.debug`. The code includes a comment `/** * @param args */` and a `public static void main` method that creates a `Counter` object and prints its count.
- Run and Debug Console:** A sub-menu is open over the `Debug As` option, showing two configurations: `1 Debug on Server` (Alt+Shift+D, R) and `2 Java Application` (Alt+Shift+D, J). A `Debug Configurations...` option is also visible.

File Edit Source Navigate Search Project Git Run Window Help

Perspectiva de Depuração

The screenshot displays the Eclipse IDE in a debug perspective. The main window is titled "Debug - de.vogella.debug.first/src/de/vogella/debug/first/Main.java - Eclipse". The menu bar includes File, Edit, Refactor, Run, Source, Navigate, Search, Project, Window, and Help. The toolbar contains various icons for debugging and development.

The Debug console (top left) shows the execution stack:

- Main (1) [Java Application]
- de.vogella.debug.first.Main at localhost:2038
- Thread [main] (Suspended (breakpoint at line 9 in Main))
- Main.main(String[]) line: 9
- C:\Program Files\Java\jre6\bin\javaw.exe (06.07.2009 11:30:57)

The Variables view (top right) shows a table with the following data:

| Name | Value |
|------|-------------------|
| args | String[0] (id=16) |

The code editor (bottom left) shows the source code of Main.java with a breakpoint at line 9:

```
package de.vogella.debug.first;

public class Main {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Counter counter = new Counter();
        counter.count();
        System.out.println("We have counted " + counter.getResult());
    }
}
```

The Outline view (bottom right) shows the class structure:

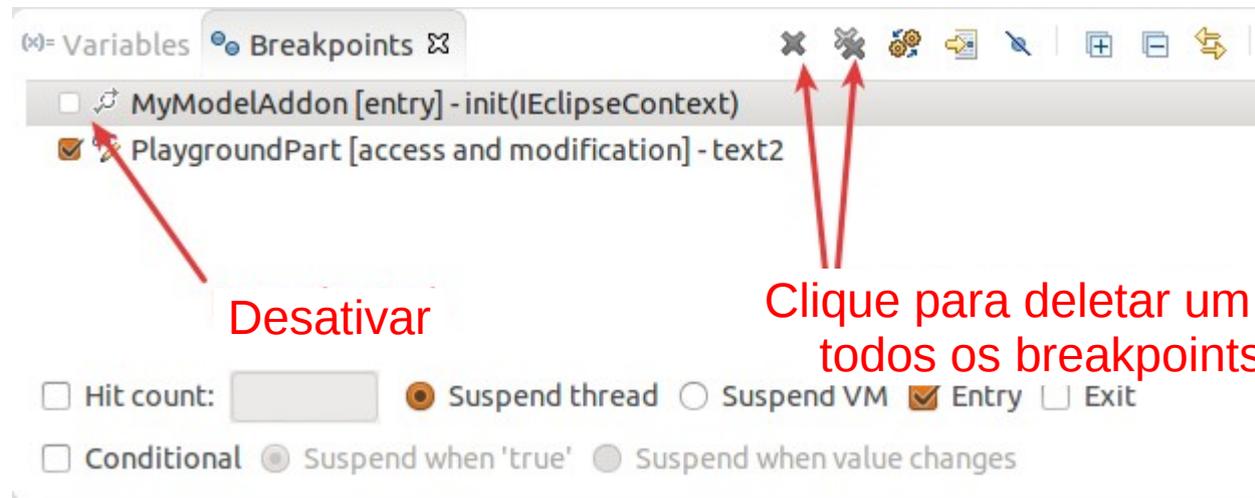
- de.vogella.debug.first
- Main
- main(String[]): void

The Console view (bottom) shows the output of the application:

```
Main (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (06.07.2009 11:30:57)
```

The status bar at the bottom indicates "Writable", "Smart Insert", and "9 : 1".

Gerenciando os Breakpoints



Execução da Depuração

F5 – executa a linha corrente e passa para a próxima linha. Se a linha selecionada é uma chamada de método, o depurador executa o código associado.

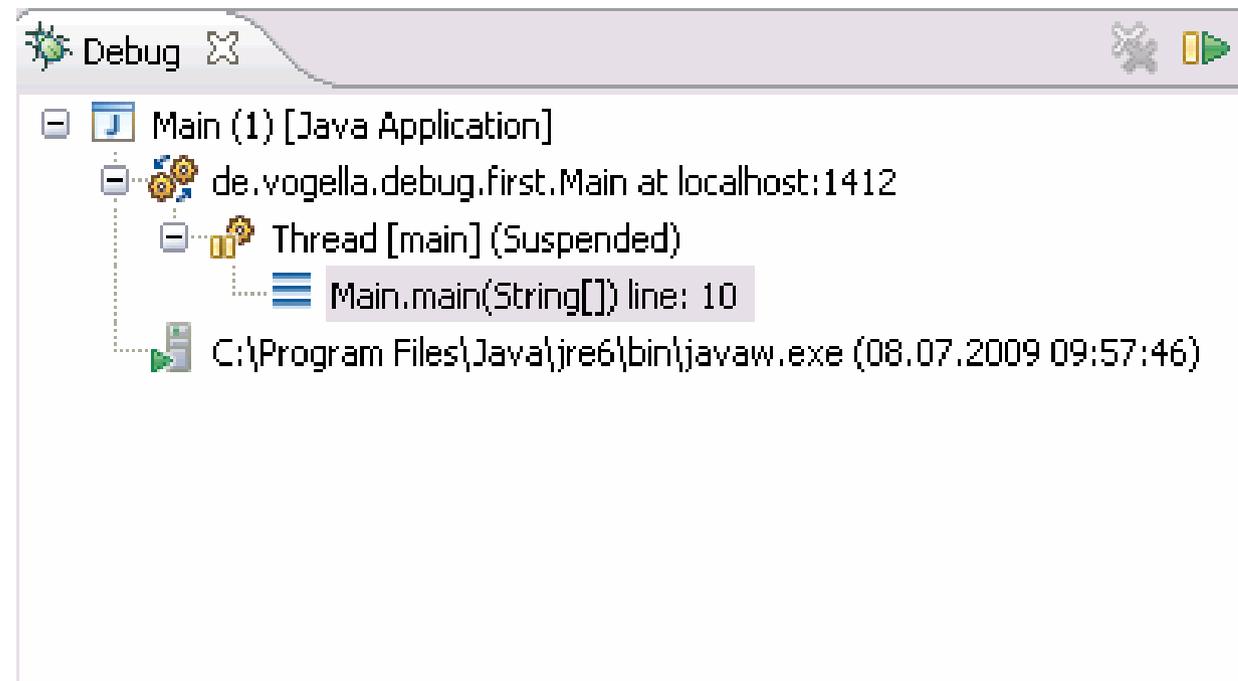
F6 – prossegue sobre uma chamada de método, ou seja, executa um método sem passar o depurador por cada linha do seu código.

F7 – prossegue para o invocador do método atualmente em execução. Isto finaliza a execução do método corrente e retorna para o invocador deste método.

F8 – informa ao depurador para continuar a execução do código do programa até que alcance o próximo breakpoint ou watchpoint.



Pilha de Chamadas



Variáveis

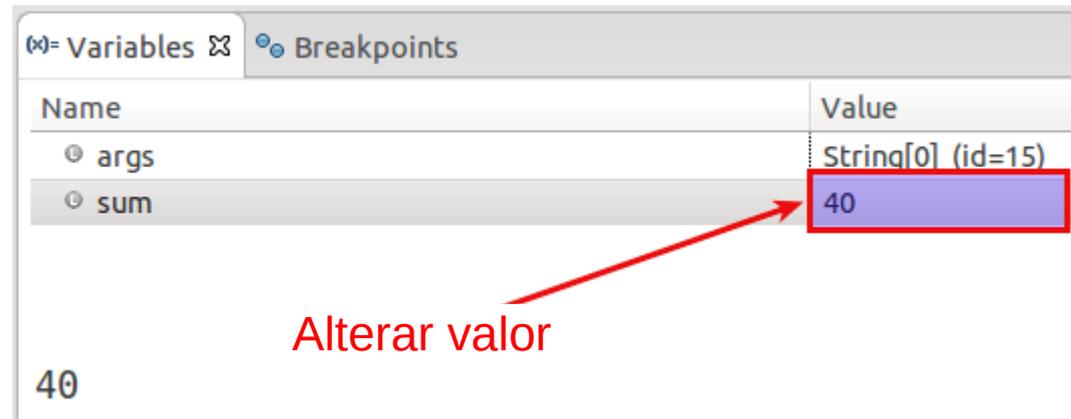
| Name | Value |
|---------|-------------------|
| args | String[0] (id=16) |
| counter | Counter (id=20) |
| result | 0 |

de.vogella.debug.first.Counter@587c94

Vertical View Orientation
Horizontal View Orientation
Variables View Only
Show Columns
Select Columns...

Layout
Java

Modificação de Variáveis



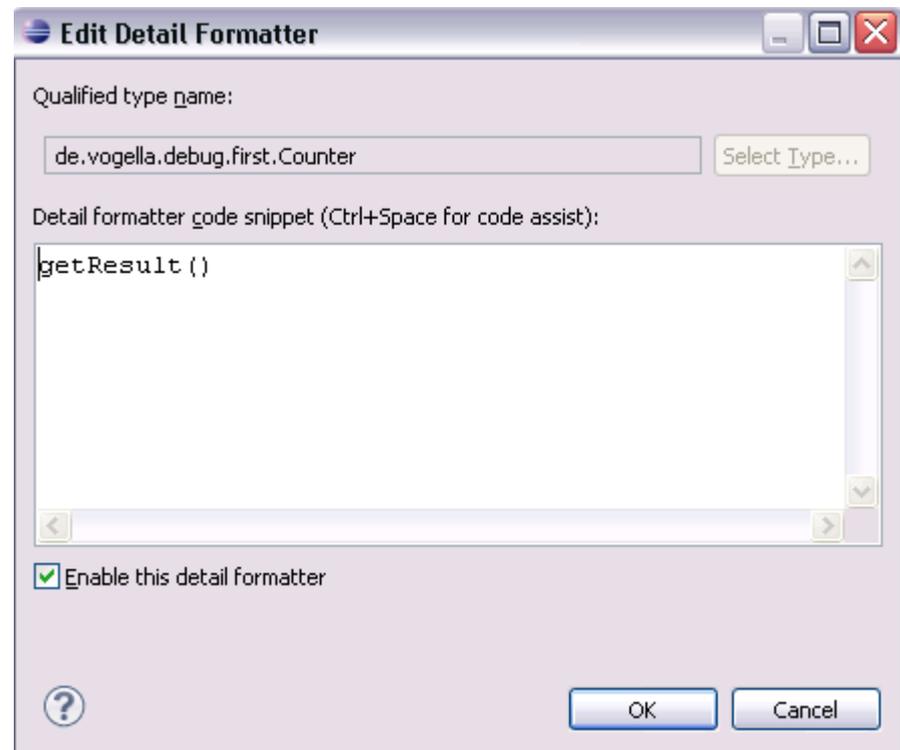
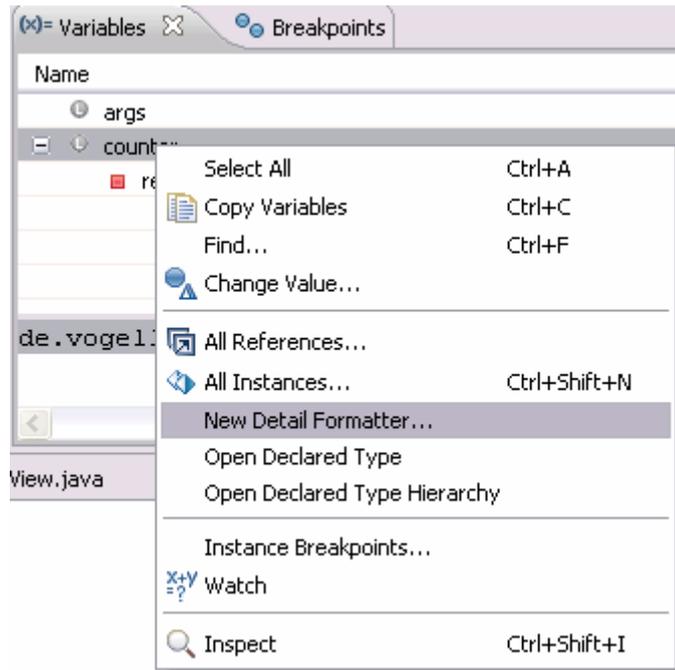
The image shows a debugger's Variables window. The window has two tabs: 'Variables' and 'Breakpoints'. The 'Variables' tab is active, displaying a table with two columns: 'Name' and 'Value'. The table contains two rows: 'args' with value 'String[0] (id=15)' and 'sum' with value '40'. The 'sum' row is highlighted, and its value '40' is enclosed in a red rectangular box. A red arrow points from the text 'Alterar valor' to the '40' in the box. Below the table, the value '40' is also displayed in a separate box.

| Name | Value |
|------|-------------------|
| args | String[0] (id=15) |
| sum | 40 |

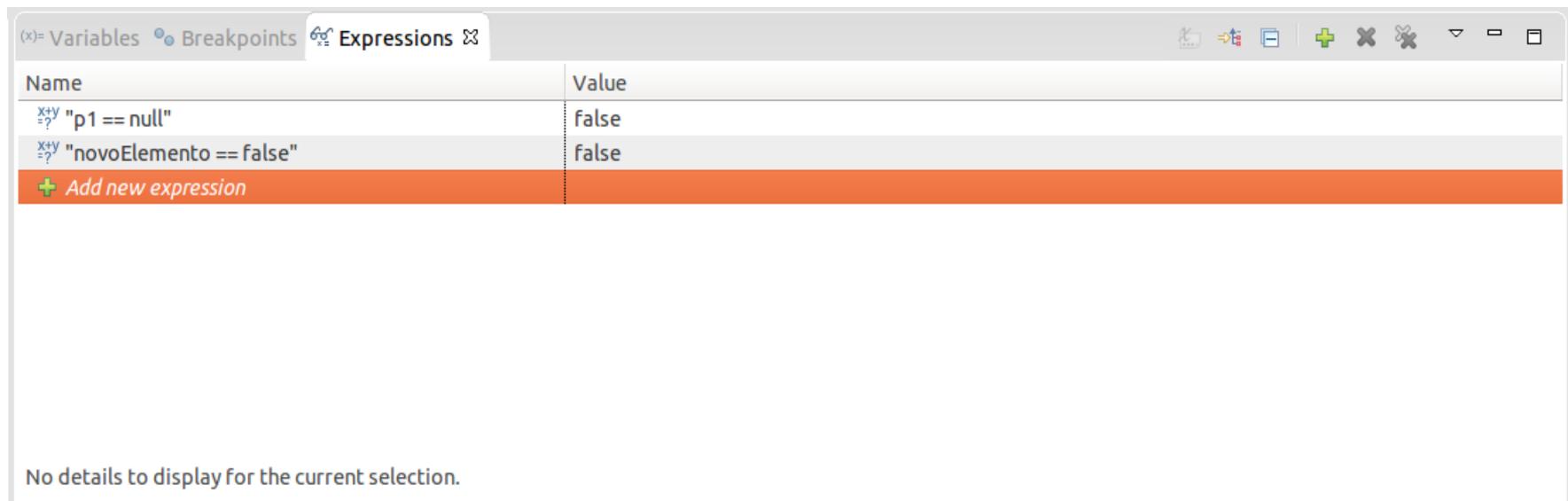
Alterar valor

40

Visualização de Variáveis



Expressões



The screenshot shows a debugger window with the 'Expressions' tab selected. The window title is '(x) Variables Breakpoints Expressions'. The main area contains a table with two columns: 'Name' and 'Value'. The first row shows the expression '"p1 == null"' with a value of 'false'. The second row shows the expression '"novoElemento == false"' with a value of 'false'. Below the table is an orange button with a plus sign and the text 'Add new expression'. At the bottom of the window, it says 'No details to display for the current selection.'

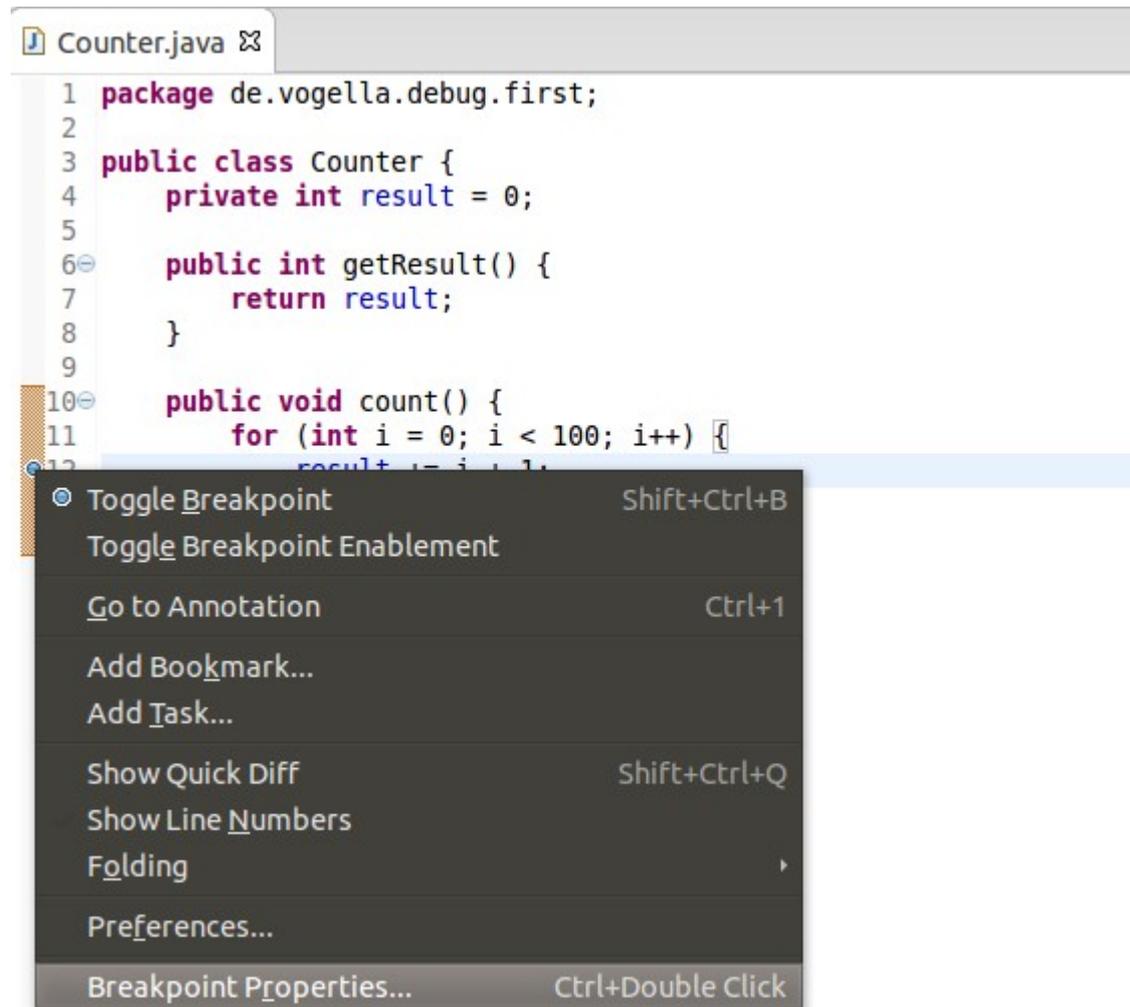
| Name | Value |
|--------------------------------------|-------|
| <code>"p1 == null"</code> | false |
| <code>"novoElemento == false"</code> | false |

[+ Add new expression](#)

No details to display for the current selection.

Propriedades do Breakpoint

- Condições



The screenshot shows an IDE window titled "Counter.java" containing the following Java code:

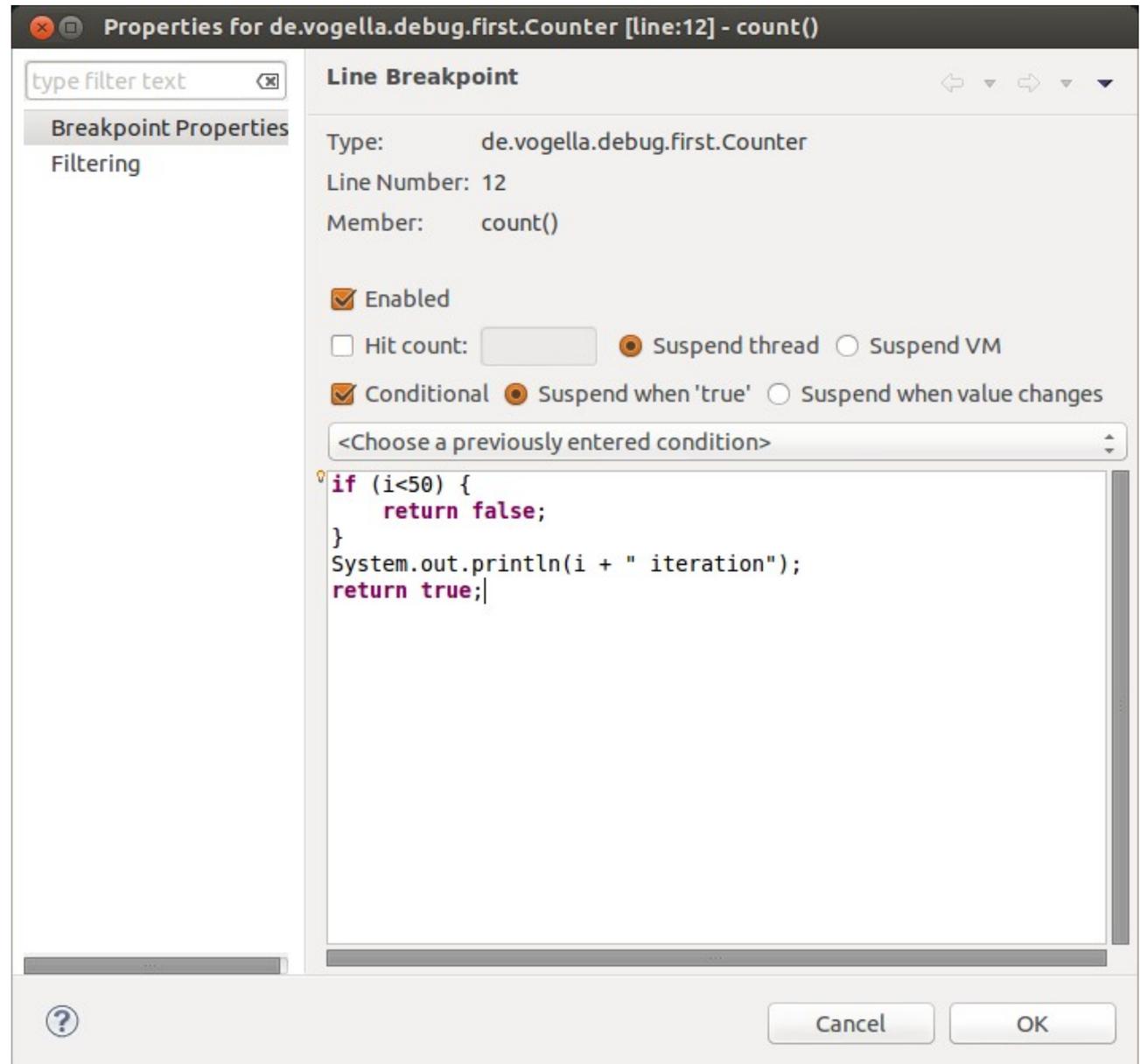
```
1 package de.vogella.debug.first;
2
3 public class Counter {
4     private int result = 0;
5
6     public int getResult() {
7         return result;
8     }
9
10    public void count() {
11        for (int i = 0; i < 100; i++) {
12            result = i + 1;
```

A breakpoint is set on line 12. A context menu is open over this line, listing the following options:

- Toggle Breakpoint (Shift+Ctrl+B)
- Toggle Breakpoint Enablement
- Go to Annotation (Ctrl+1)
- Add Bookmark...
- Add Task...
- Show Quick Diff (Shift+Ctrl+Q)
- Show Line Numbers
- Folding
- Preferences...
- Breakpoint Properties... (Ctrl+Double Click)

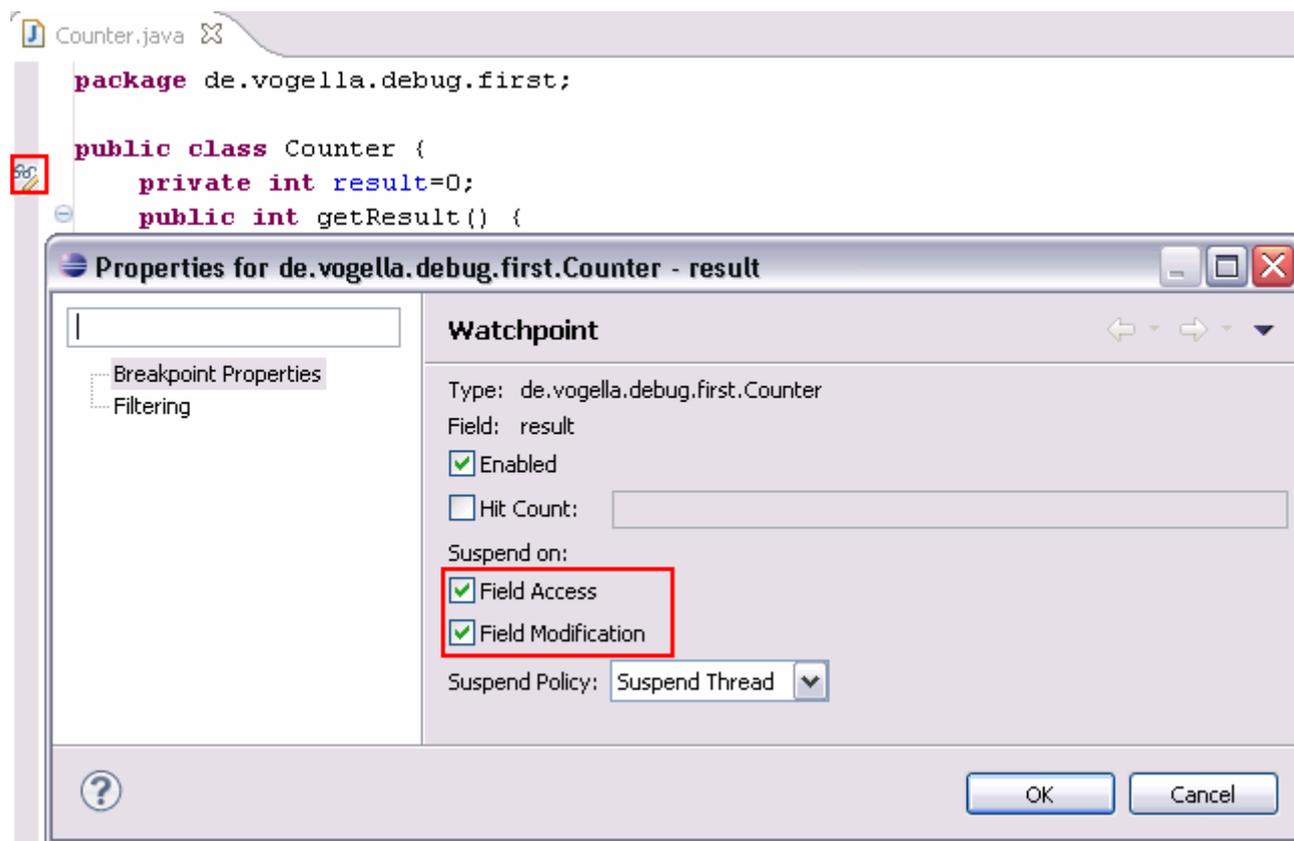
Propriedades do Breakpoint

- Condições

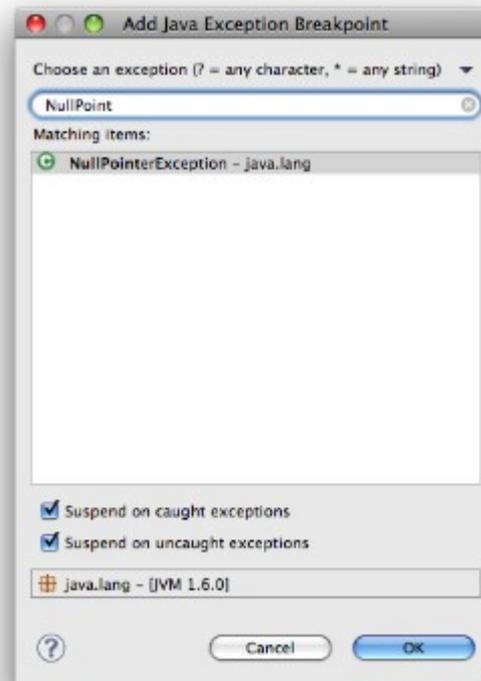


Propriedades do Watchpoint

- Watchpoint: um breakpoint definido em um campo



Exception Breakpoints



Method Breakpoints

The screenshot shows an IDE window with a Java code snippet and a dialog box for configuring a method breakpoint. The code is as follows:

```
public void count() {  
    for (int i = 0; i < 100; i++) {  
        result += i + 1;  
    }  
}
```

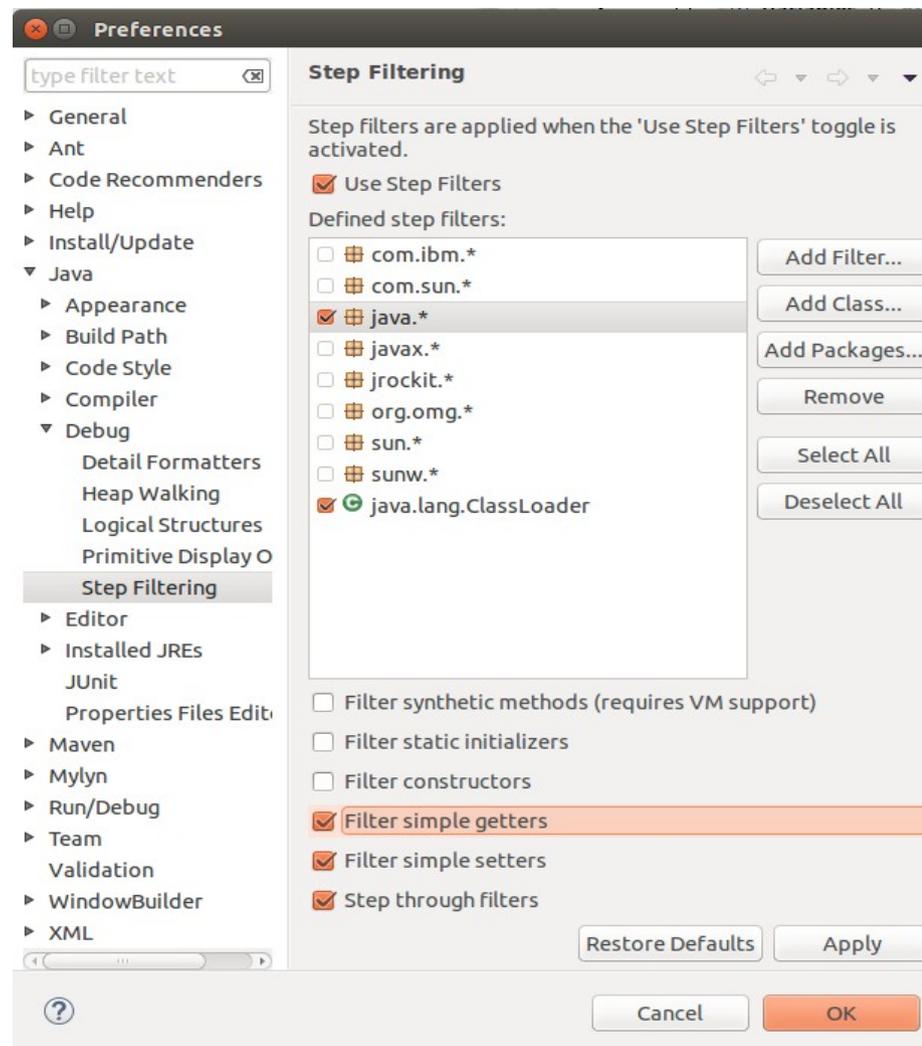
The dialog box, titled "Properties for de.vogella.debug.first.Counter - count()", has the following settings:

- Method Breakpoint** section:
 - Type: de.vogella.debug.first.Counter
 - Method: count()
 - Enabled
 - Hit Count: [empty text box]
 - Enable Condition (Ctrl+Space for code assist)
 - Condition editor: [empty text area]
 - Suspend when:
 - condition is 'true'
 - value of condition changes
 - Suspend on:
 - Method Entry
 - Method Exit
 - Suspend Policy: Suspend Thread

The "Suspend on:" section and the "Method Entry" checkbox are highlighted with a red box in the original image.

Step Filter

- Window → Preferences → Java → Debug → Step Filtering



Referência

Java Debugging with Eclipse – Tutorial

<http://www.vogella.com/tutorials/EclipseDebugging/article.html>

Apostila Java e Orientação a Objetos. Apêndice - Debugging.

<http://www.caelum.com.br/apostila-java-orientacao-objetos/apendice-debugging/>