What is debugging? (1)

- *Debugging* allows you to run a program interactively while watching the source code and the variables during the execution.

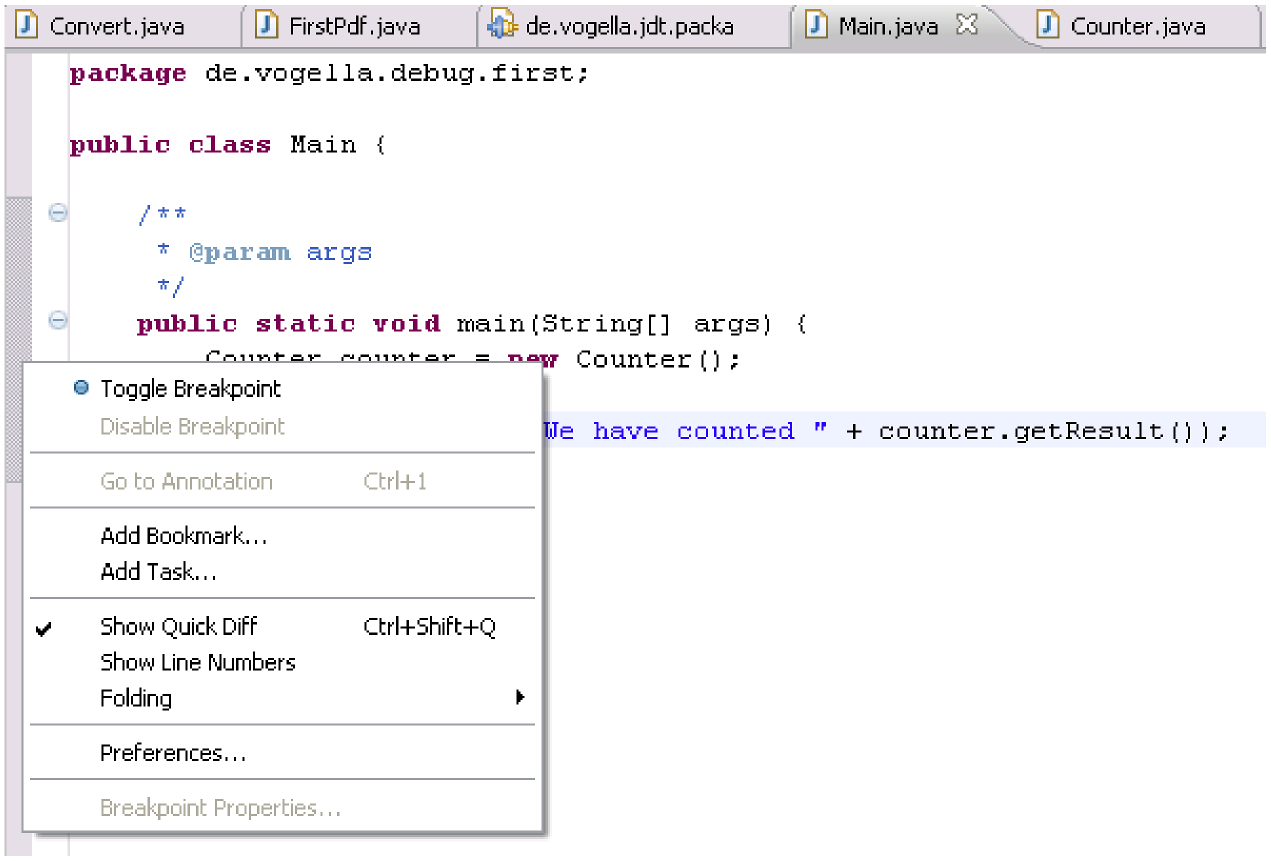
- By *breakpoints* in the source code you specify where the execution of the program should stop. To stop the execution only if a field is read or modified, you can specify *watchpoints* .

What is debugging? (2)

- *Breakpoints* and *watchpoints* can be summarized as *stop points*.
- Once the program is stopped you can investigate variables, change their content, etc.

Setting Breakpoints (1)

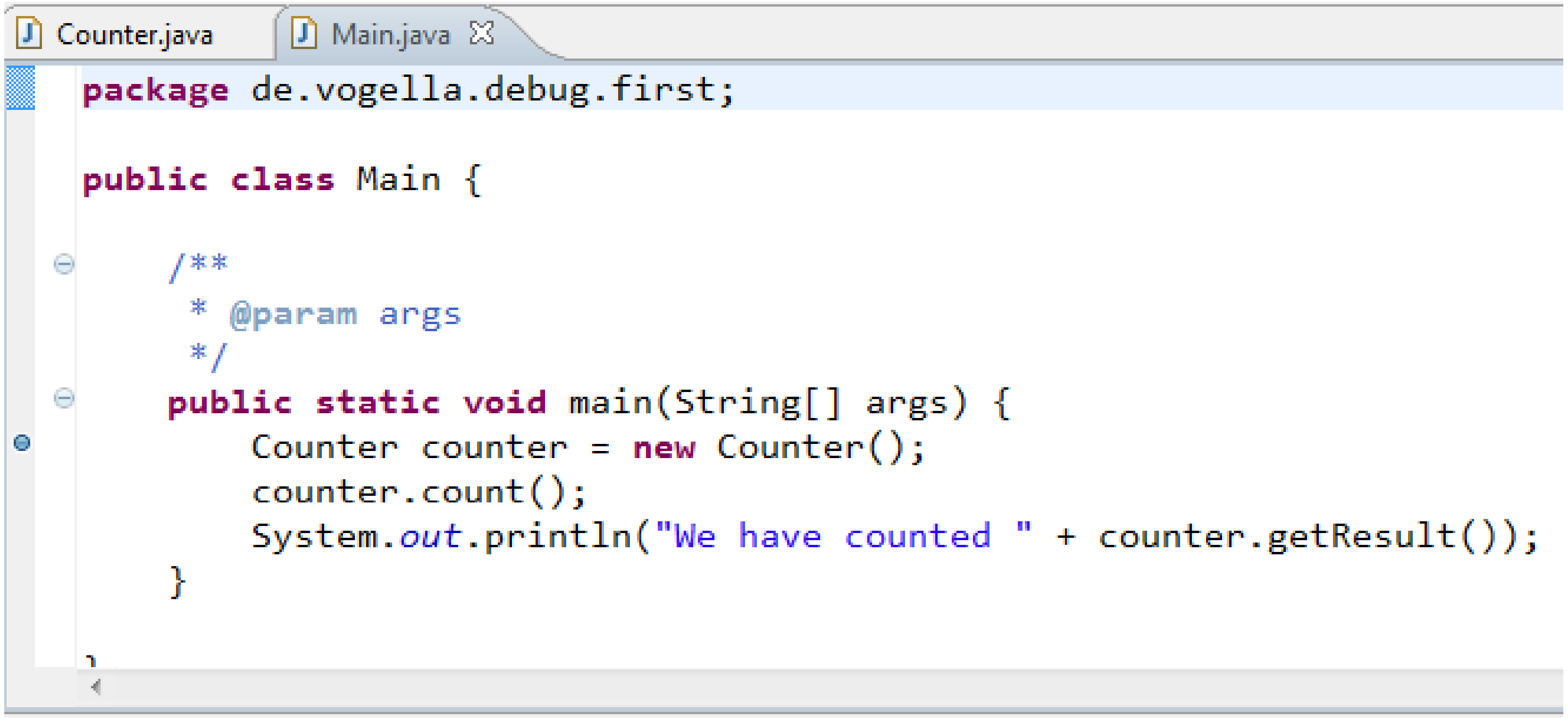
- To set breakpoints in your source code right-click in the small left margin in your source code editor and select Toggle Breakpoint. Alternatively you can double-click on this position.



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

public class Main {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Counter counter = new Counter();

We have counted " + counter.getResult());
```

Setting Breakpoints (2)

- For example in the following screenshot we set a breakpoint on the line Counter counter = new Counter();.

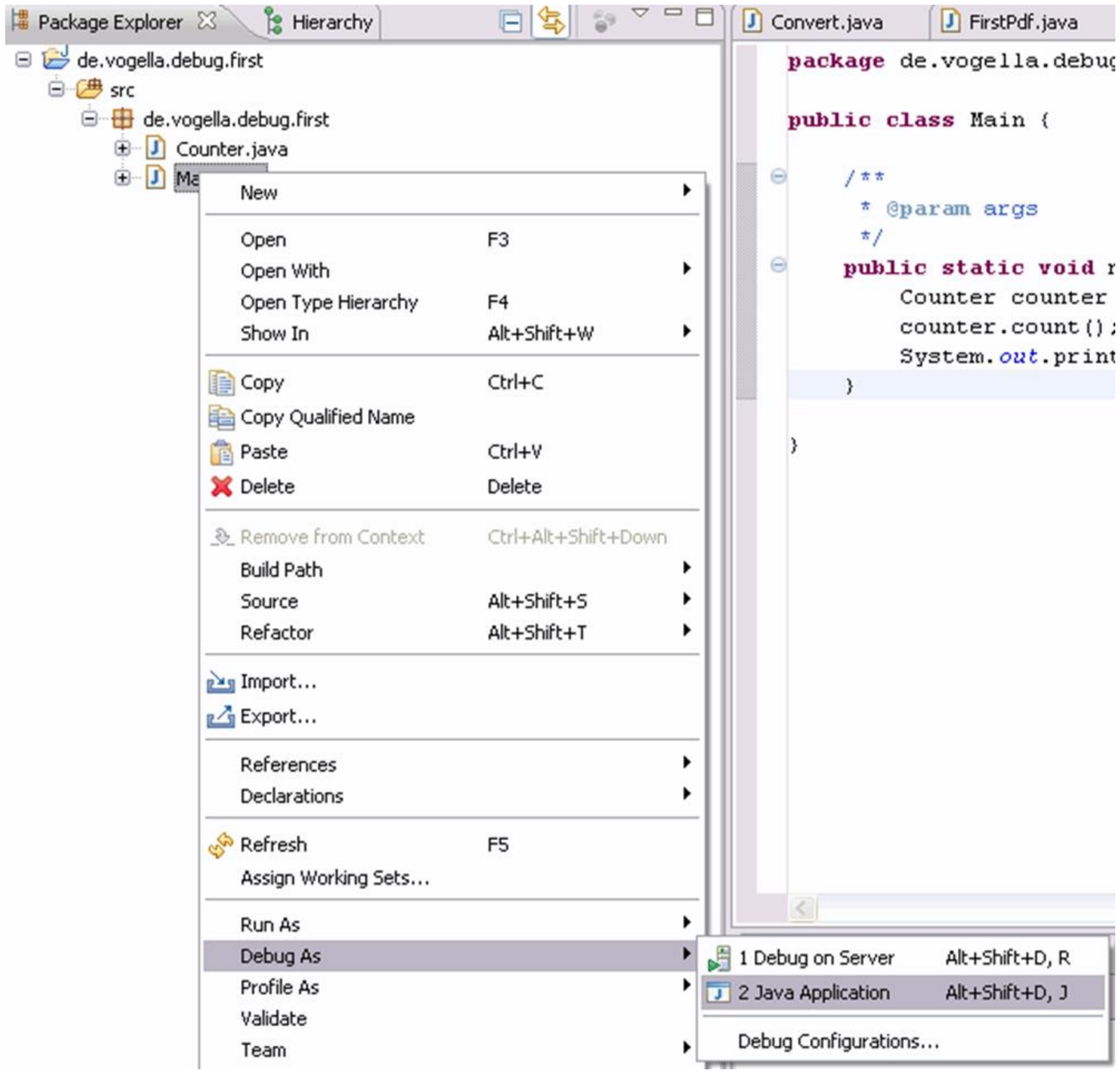


```
package de.vogella.debug.first;

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());
    }

}
```

Starting the Debugger (1)

- To debug your application, select a Java file which can be executed, right-click on it and select:

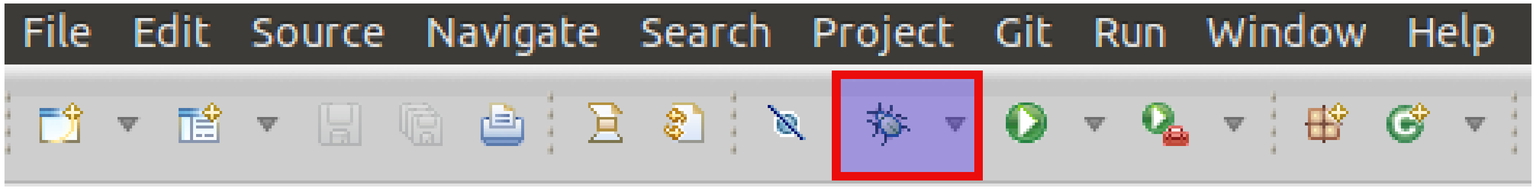
"Debug As" → "Java Application".



```
package de.vogella.debug

public class Main {

    /**
     * @param args
     */
    public static void r
        Counter counter
        counter.count();
        System.out.print
    }

}
```

Starting the Debugger (2)

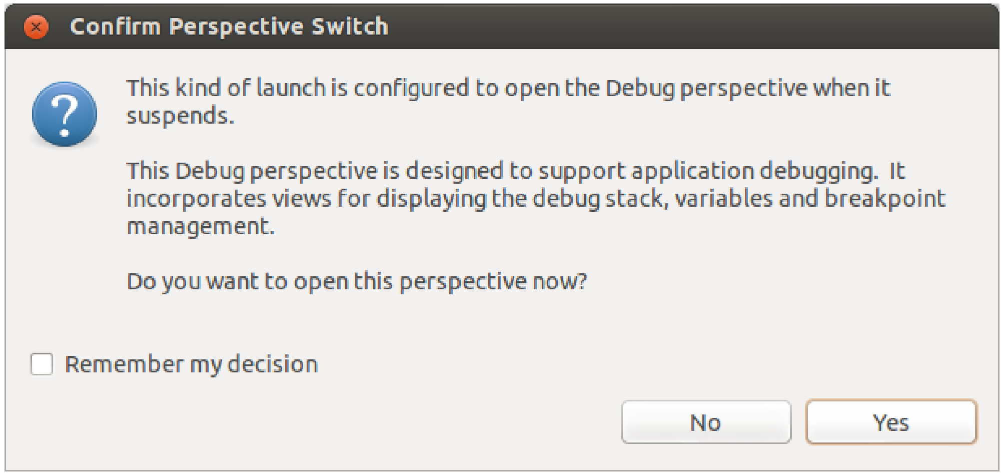
- After you have started the application once via the context menu, you can use the created launch configuration again via the Debug button in the Eclipse toolbar.



- If you have not defined any breakpoints, this will run your program as normal. To debug the program you need to define breakpoints.

Starting the Debugger (3)

- If you start the debugger, Eclipse asks you if you want to switch to the Debug *perspective* once a stop point is reached. Answer Yes in the corresponding dialog.



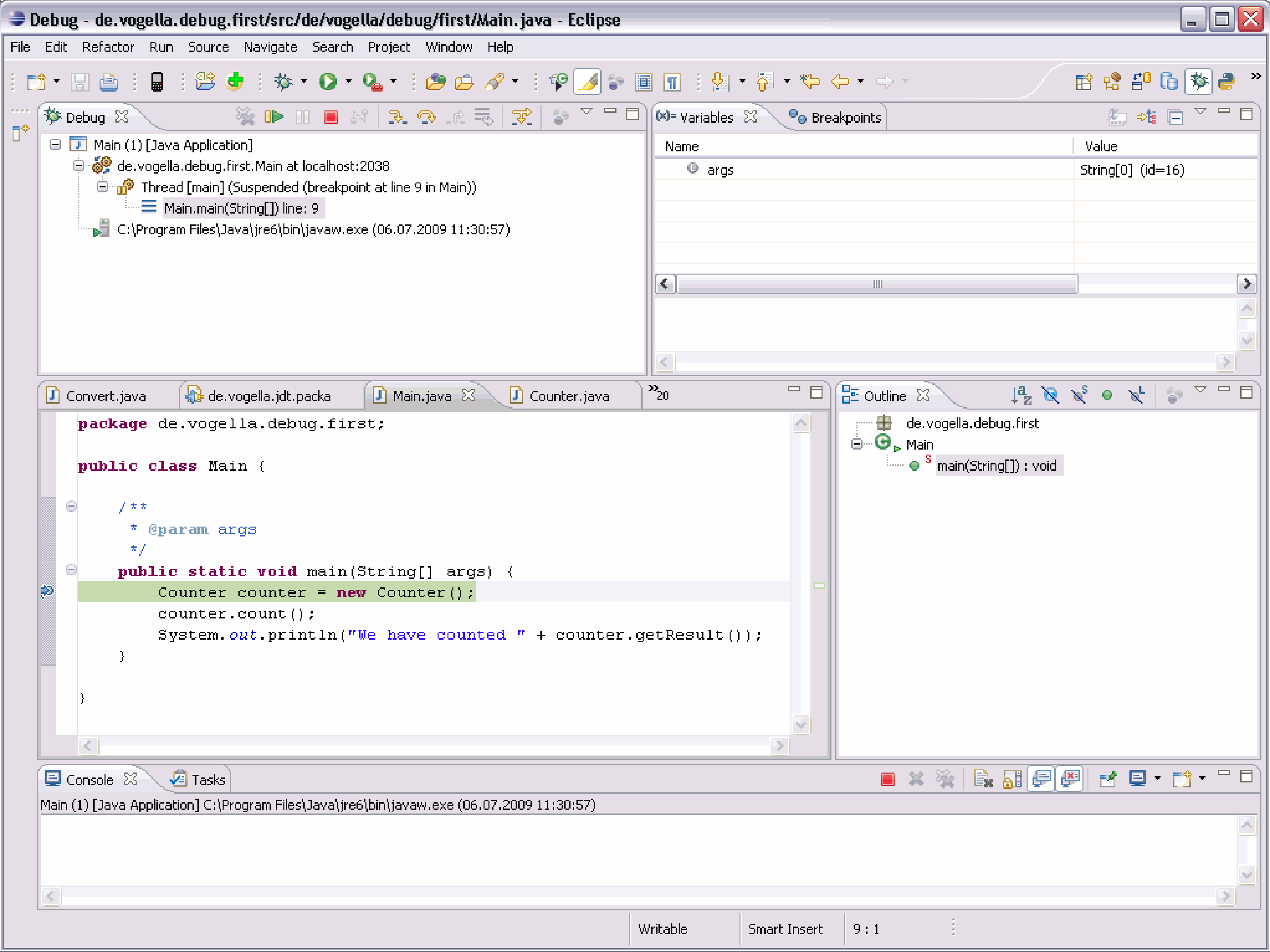
Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

- Afterwards Eclipse opens this *perspective*, which looks similar to the following screenshot.


Debug - de.vogella.debug.first/src/de/vogella/debug/first/Main.java - Eclipse
File Edit Refactor Run Source Navigate Search Project Window Help
Debug
Variables
Breakpoints
Convert.java
de.vogella.jdt.packa
Main.java
Counter.java
20
Outline
Console
Tasks
Writable
Smart Insert
9 : 1

```
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)
```

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

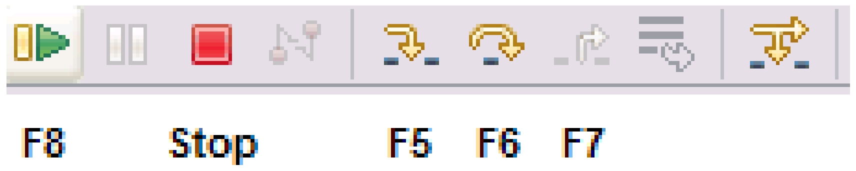
```java
package de.vogella.debug.first;

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());
	}

}
```

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

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

Controlling the program execution (1)

- Eclipse provides buttons in the toolbar for controlling the execution of the program you are debugging. Typically it is easier to use the corresponding keys to control this execution.
- You can use the F5, F6, F7 and F8 key to step through your coding.
- The following picture displays the buttons and their related keyboard shortcuts.

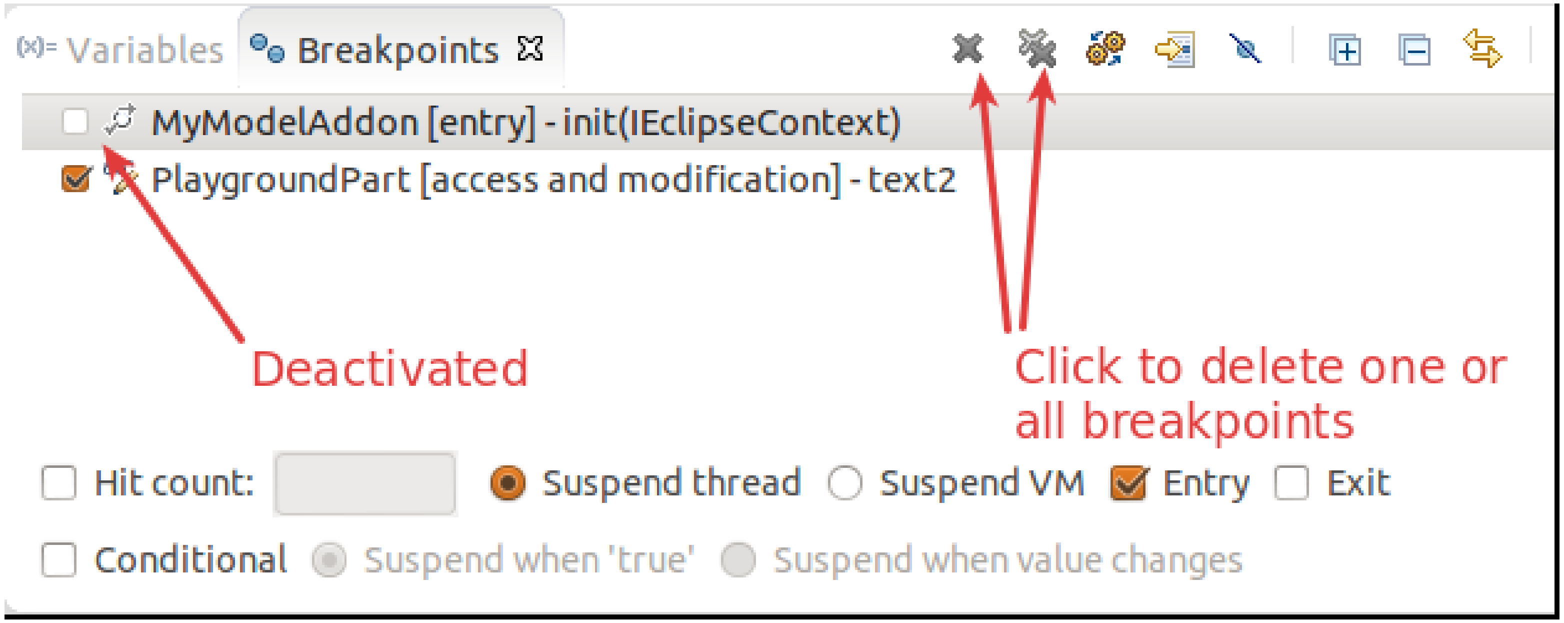


Controlling the program execution (2)

Key	Description
F5	Executes the currently selected line and goes to the next line in your program. If the selected line is a method call the debugger steps into the associated code.
F6	F6 steps over the call, i.e. it executes a method without stepping into it in the debugger.
F7	F7 steps out to the caller of the currently executed method. This finishes the execution of the current method and returns to the caller of this method.
F8	F8 tells the Eclipse debugger to resume the execution of the program code until is reaches the next breakpoint or watchpoint.

Breakpoints view and deactivating breakpoints (1)

- The Breakpoints view allows you to delete and deactivate stop points, i.e. breakpoints and watchpoints and to modify their properties.
- To deactivate a breakpoint, remove the corresponding checkbox in the Breakpoints view . To delete it you can use the corresponding buttons in the view toolbar. These options are depicted in the following screenshot.

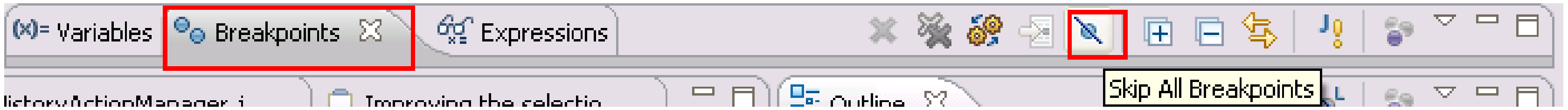


☐ MyModelAddon [entry] - init(IEclipseContext)
☑ PlaygroundPart [access and modification] - text2

☐ Hit count: (●) Suspend thread ( ) Suspend VM ☑ Entry ☐ Exit
☐ Conditional ( ) Suspend when 'true' ( ) Suspend when value changes

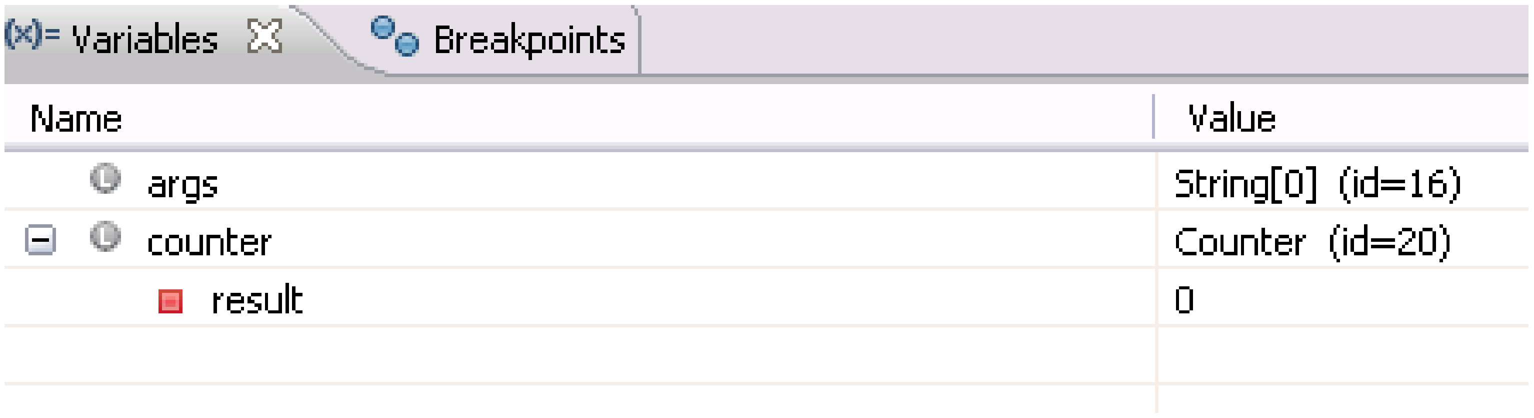
Breakpoints view and deactivating breakpoints (2)

- If you want to deactivate all your breakpoints you can press the Skip all breakpoints button. If you press it again, your breakpoints are reactivated. This button is highlighted in the following screenshot.



Evaluating variables in the debugger (1)

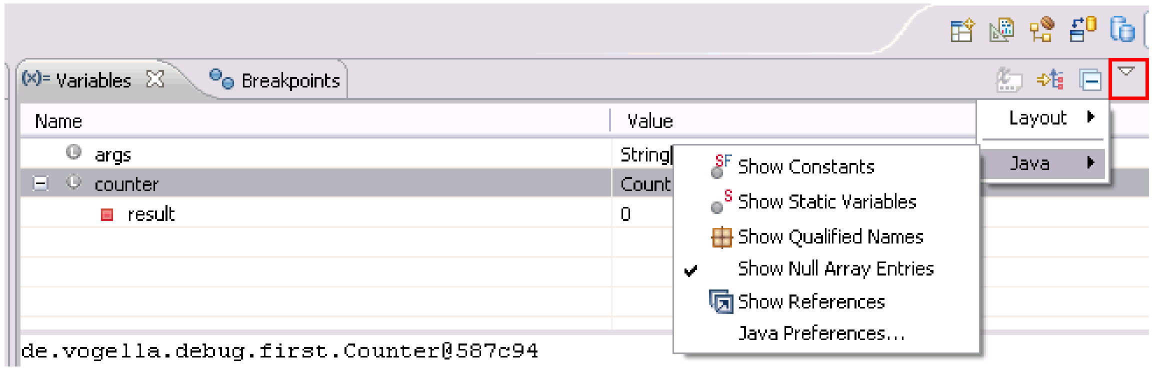
- The Variables view displays fields and local variables from the current executing stack. Please note you need to run the debugger to see the variables in this view .



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

Evaluating variables in the debugger (2)

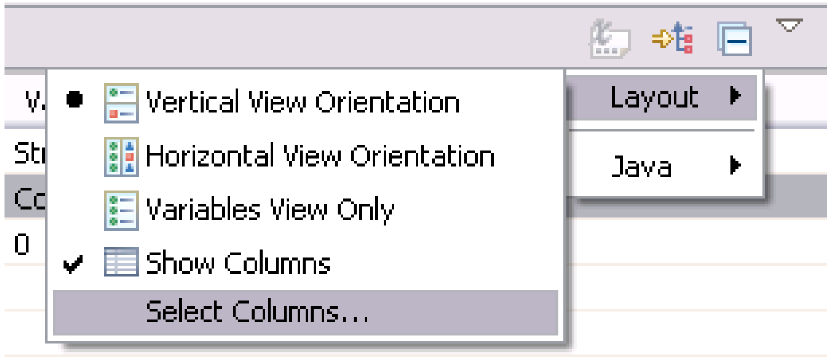
- Use the drop-down menu to display static variables.



Name	Value
args	String
counter	Count
result	0

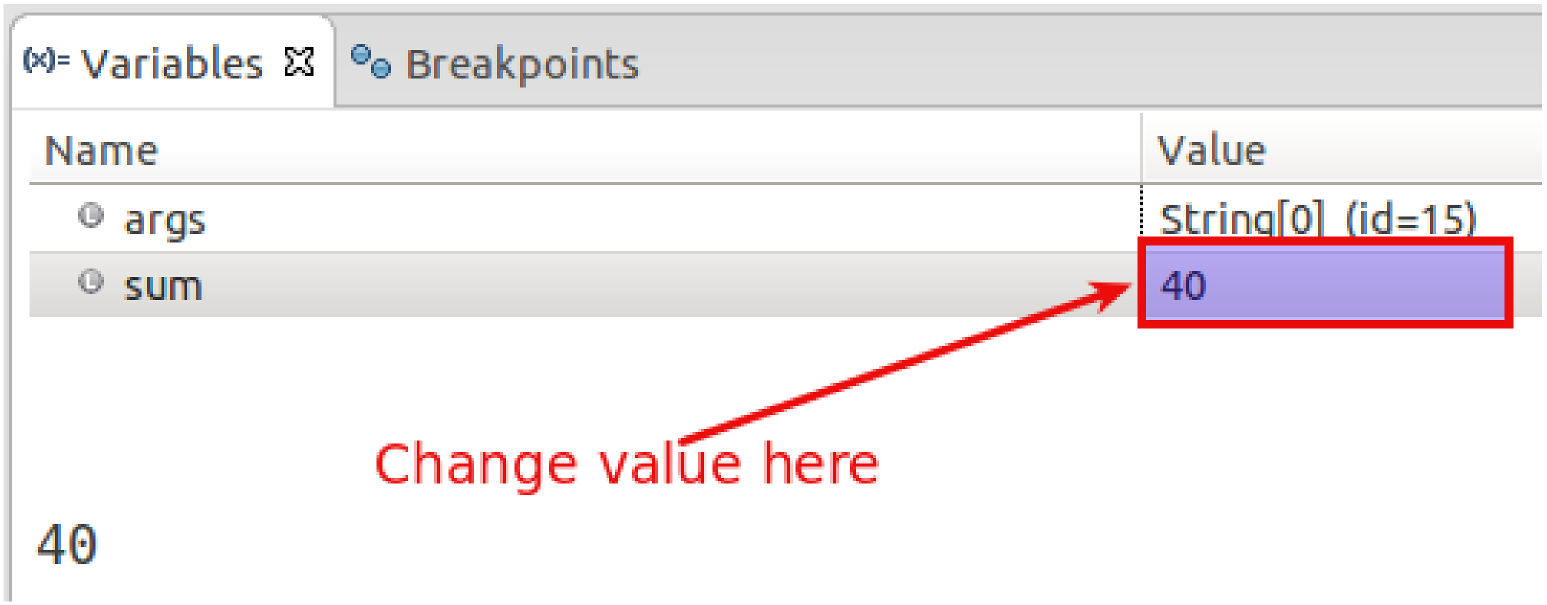
```
de.vogella.debug.first.Counter@587c94
```

- Via the drop-down menu of the Variables view you can customize the displayed columns. For example, you can show the actual type of each variable declaration. For this select Layout → Select Columns... → Type.



Changing variable assignments in the debugger

- The Variables view allows you to change the values assigned to your variable at runtime.



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

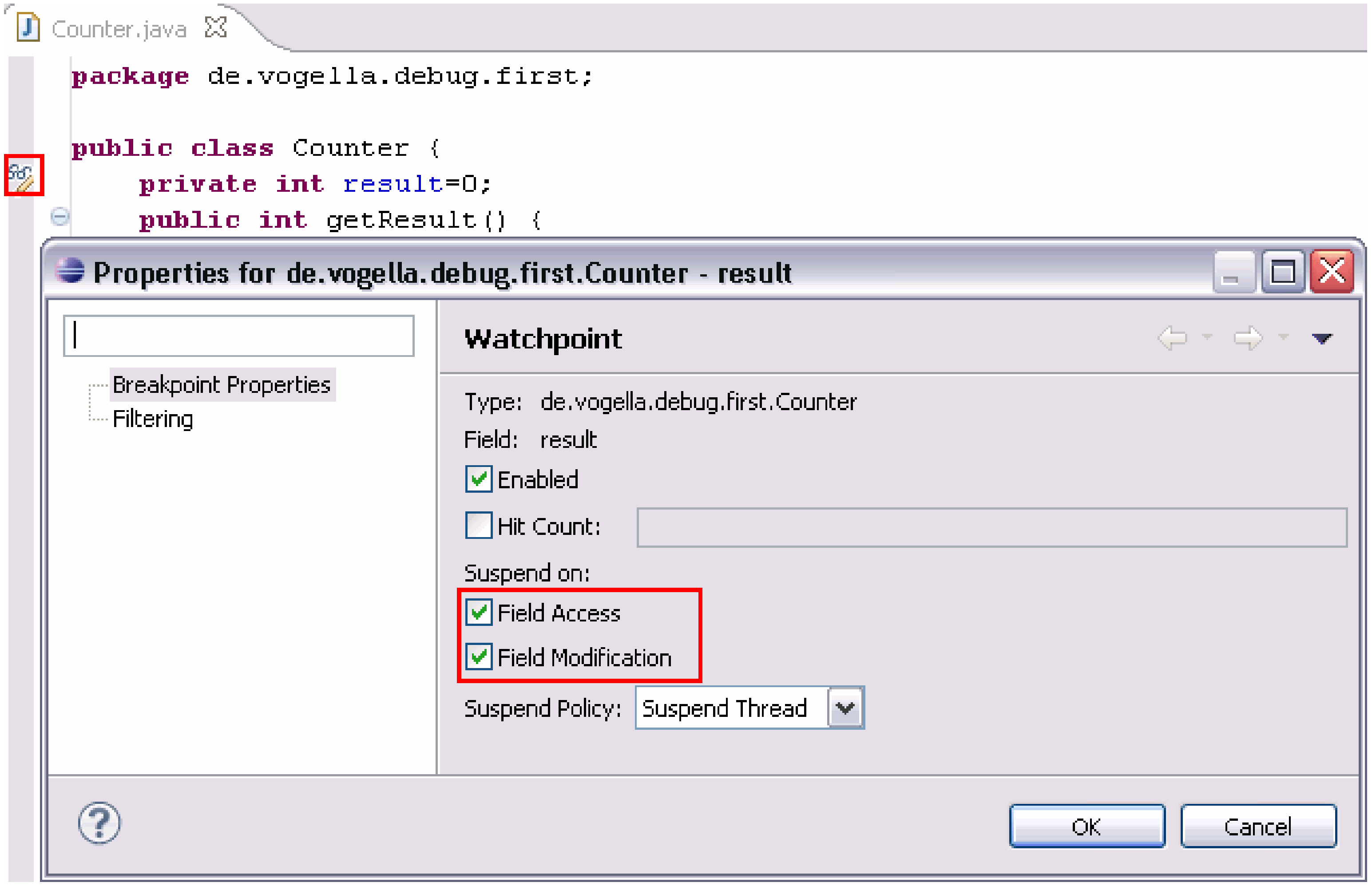
```
40
```

Watchpoint (1)

- A watchpoint is a breakpoint set on a field. The debugger will stop whenever that field is read or changed.

- You can set a watchpoint by double-clicking on the left margin, next to the field declaration. In the properties of a watchpoint you can configure if the execution should stop during read access (Field Access) or during write access (Field Modification) or both.

Watchpoint (2)



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

public class Counter {
    private int result=0;
    public int getResult() {
```

Properties for de.vogella.debug.first.Counter - result

Watchpoint

Type: de.vogella.debug.first.Counter

Field: result

[✓] Enabled

[ ] Hit Count:

Suspend on:

[✓] Field Access

[✓] Field Modification

Suspend Policy: Suspend Thread