Debugging the Eclipse IDE for Java Developers

Debugging is the routine process of locating and removing bugs, errors or abnormalities from programs. It's a must have skill for any Java developer because it helps to find subtle bug that are not visible during code reviews or that only happens when a specific condition occurs. The Eclipse Java IDE provides many debugging tools and views grouped in the **Debug Perspective** to help the you as a developer debug effectively and efficiently.

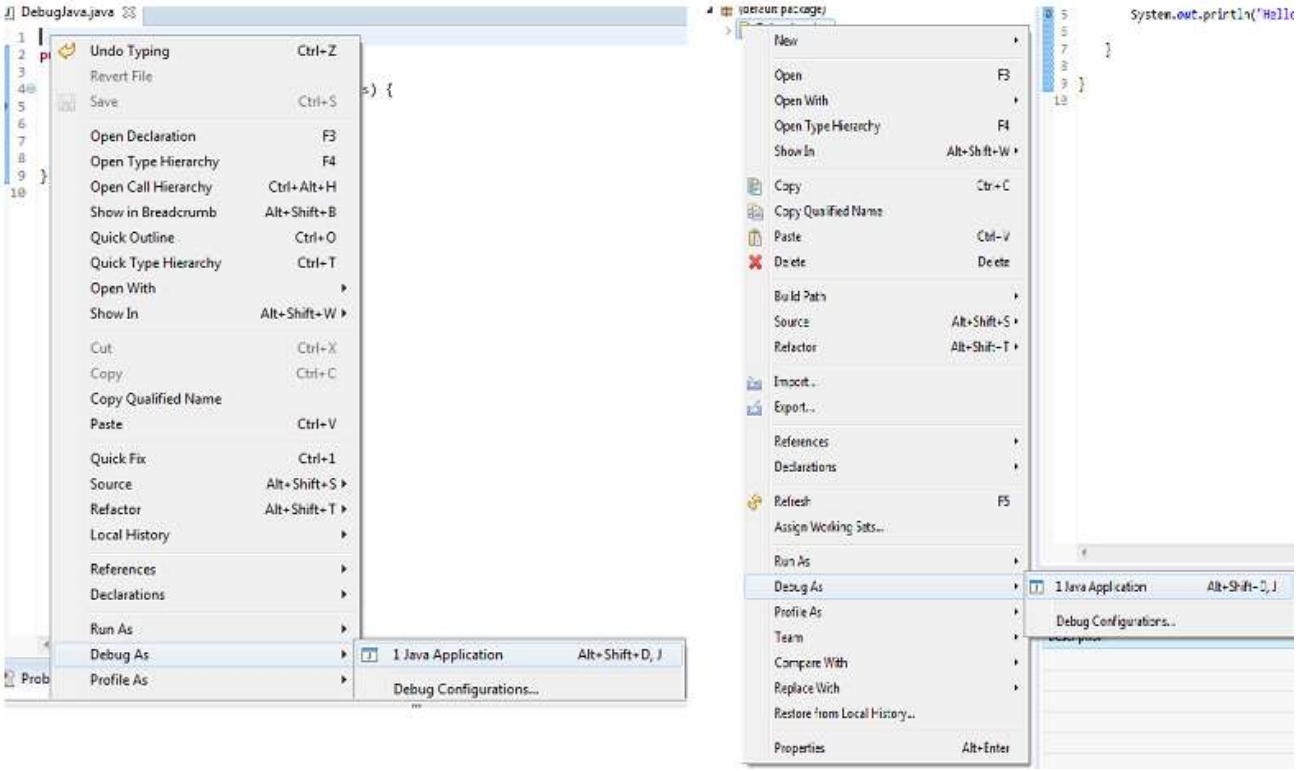
There are many improvements included in the latest Eclipse Java Development Tools (JDT) (https://projects.eclipse.org/projects/eclipse.jdt) release included in the Eclipse Oxygen (https://www.eclipse.org/oxygen/) Simultaneous Release. This article will start with a beginner's guide to start you with debugging. In the second part of the article, you will find a more advanced guide to debugging and you'll discover what's new for debugging in Eclipse Oxygen.

Beginner's Guide to Quick Start Debugging

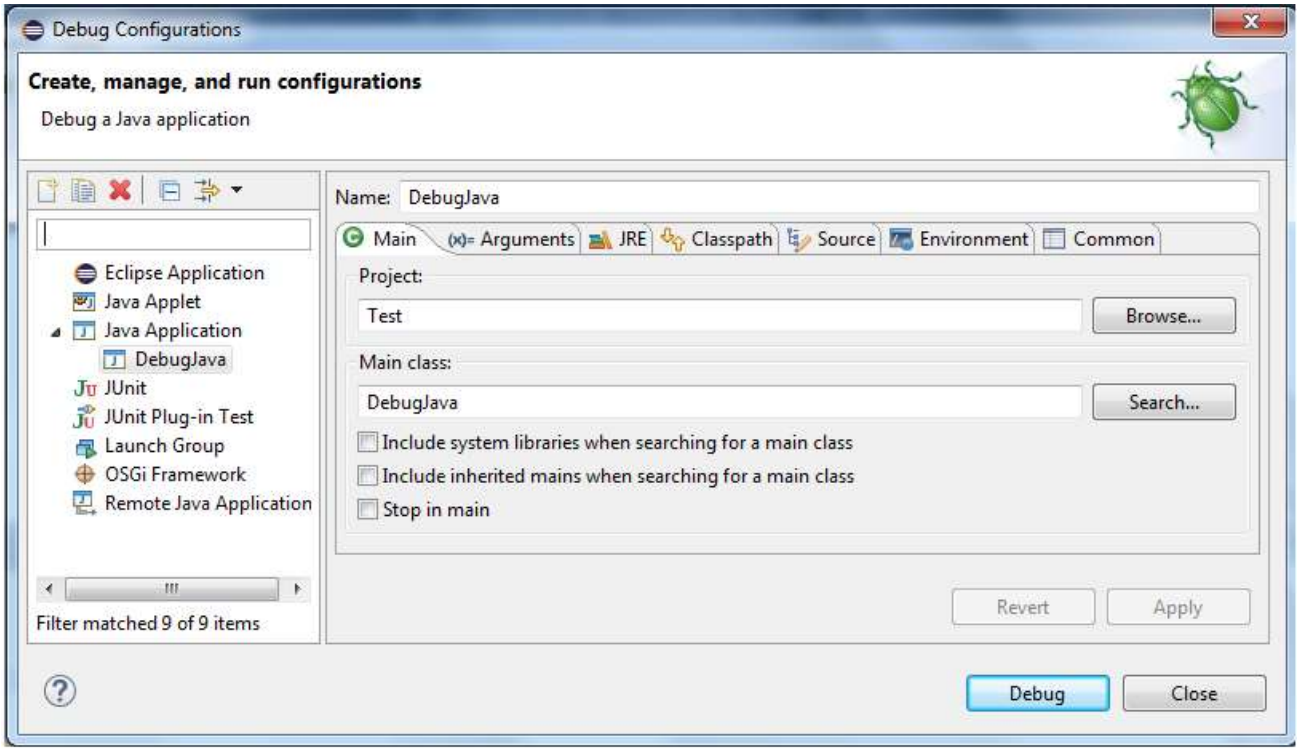
Here are some quick tips and tools that will help you get started quickly with debugging your Java project.

1. Launching and Debugging a Java program

A Java program can be debugged simply by right clicking on the Java editor class file from Package explorer. Select **Debug As → Java Application** or use the shortcut **Alt + Shift + D, J** instead.



Either actions mentioned above creates a new **Debug Launch Configuration** and uses it to start the Java application.

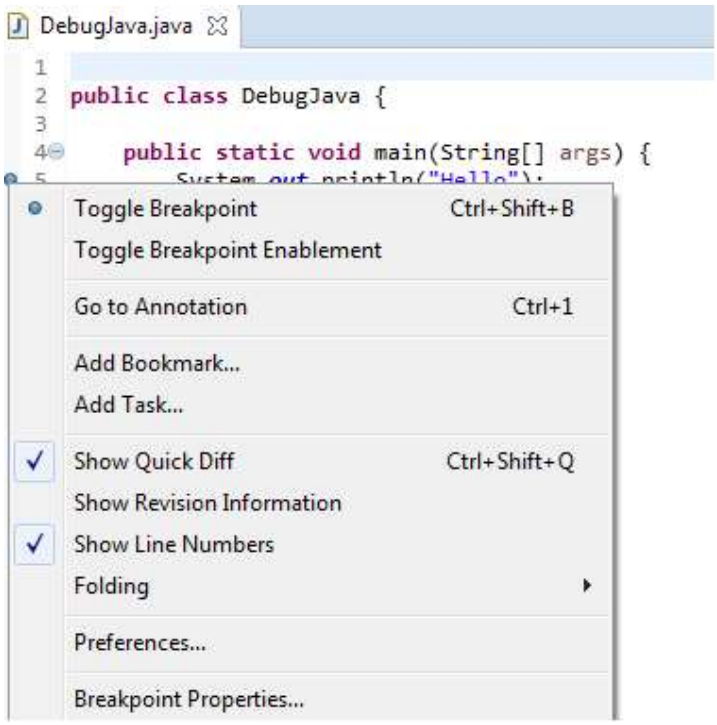


In most cases, users can edit and save the code while debugging without restarting the program.This works with the support of **HCR** (Hot Code Replacement), which has been specifically added as a standard Java technique to facilitate experimental development and to foster iterative trial-and-error coding.

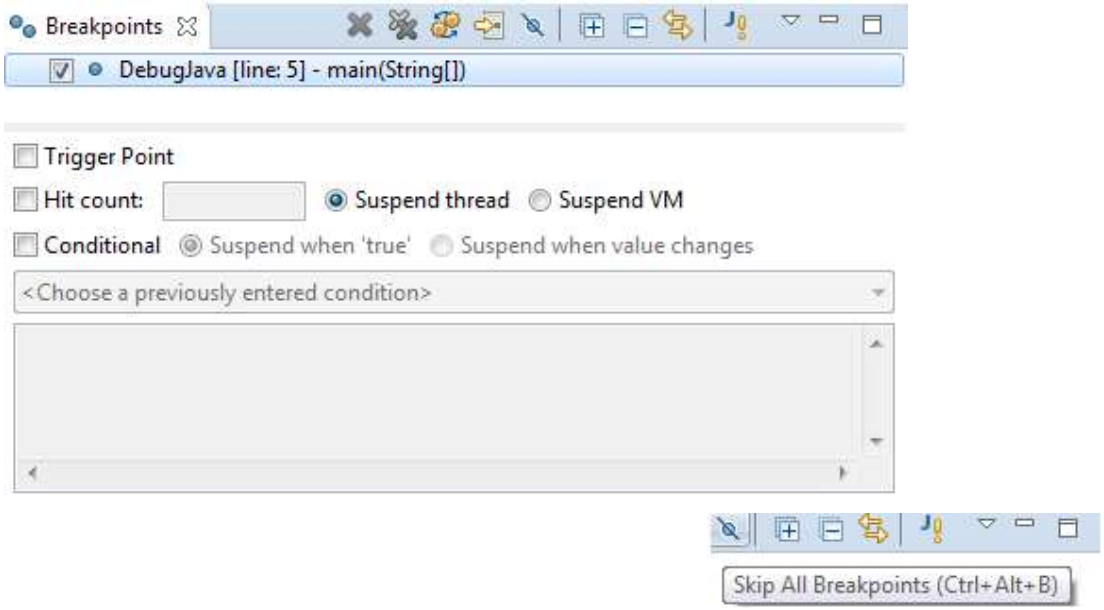
2. Breakpoints

A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point in the code.

To define a breakpoint in your source code, right-click in the left margin in the Java editor and select *Toggle Breakpoint*. Alternatively, you can double-click on this position.



The *Breakpoints* view allows you to delete and deactivate Breakpoints and modify their properties.



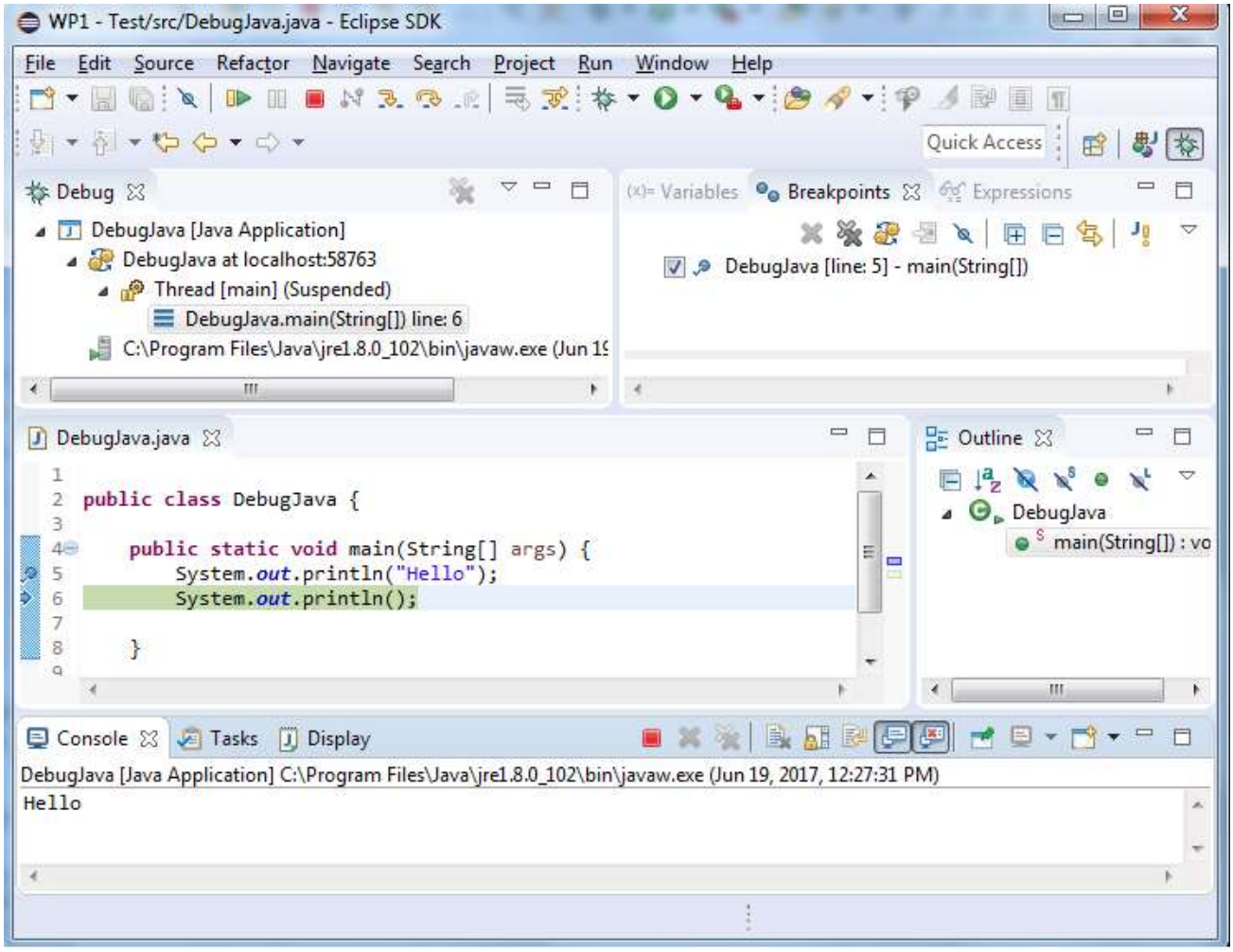
All breakpoints can be enabled/disabled using **Skip All Breakpoints**. Breakpoints can also be imported/exported to and from a workspace.

3. Debug Perspective

The debug perspective offers additional views that can be used to troubleshoot an application like Breakpoints, Variables, Debug, Console etc. When a Java program is started in the debug mode, users are prompted to switch to the debug perspective.

- **Debug view –** Visualizes call stack and provides operations on that.
- **Breakpoints view –** Shows all the breakpoints.
- **Variables/Expression view –** Shows the declared variables and their values. Press **Ctrl+Shift+d** or **Ctrl+Shift+i** on a selected variable or expression to show its value. You can also add a permanent watch on an expression/variable that will then be shown in the *Expressions view* when debugging is on.
- **Display view –** Allows to Inspect the value of a variable, expression or selected text during debugging.

- **Console view –** Program output is shown here.



4.Stepping commands

The Eclipse Platform helps developers debug by providing buttons in the toolbar and key binding shortcuts to control program execution.

Shortcut	Toolbar	Description
F5 (Step Into)		Steps into the call
F6 (Step Over)		Steps over the call
F7 (Step Return)		Steps out to the caller
F8 (Resume)		Resumes the execution
Ctrl + R (Run to Line)		Run to the line number of the current caret position
Drop to Frame		Rerun a part of your program
Shift + F5 (Use Step Filters)		Skipping the packages for Step into
Ctr + F5 / Ctrl + Alt + Click		Step Into Selection

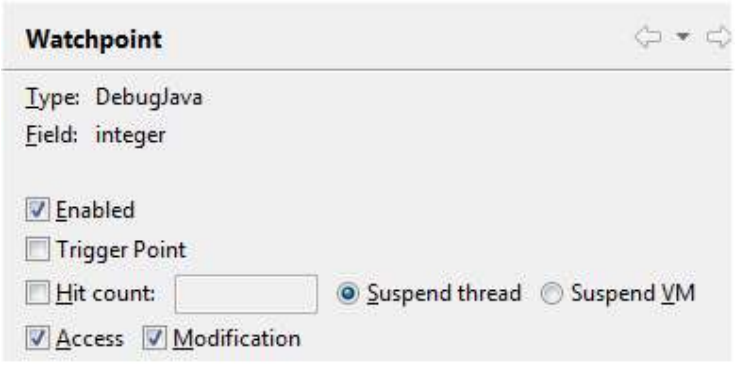
For more information about debugging visit: Eclipse Stepping Commands Help (http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftasks%2Ftask-stepping.htm)

Advanced Tools to Debug Complex Scenarios

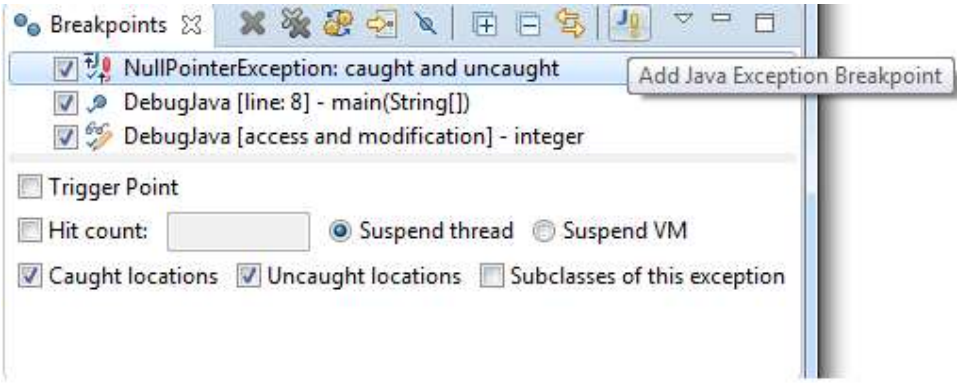
This section will give you more advanced tips and tricks to help you debug your Java project. The Eclipse Oxygen release includes many great improvements for Java debugging. Here's a quick overview.

1. Watchpoints, Exception Breakpoints, Conditional Breakpoints

a. **Watchpoints -** A watchpoint is a special breakpoint that stops the execution of an application whenever the value of a given expression/field changes, without specifying where it might occur. User can specify by **Breakpoint Properties…** if they want the execution to stop when the watch expression is **Accessed**, **Modified** or both.

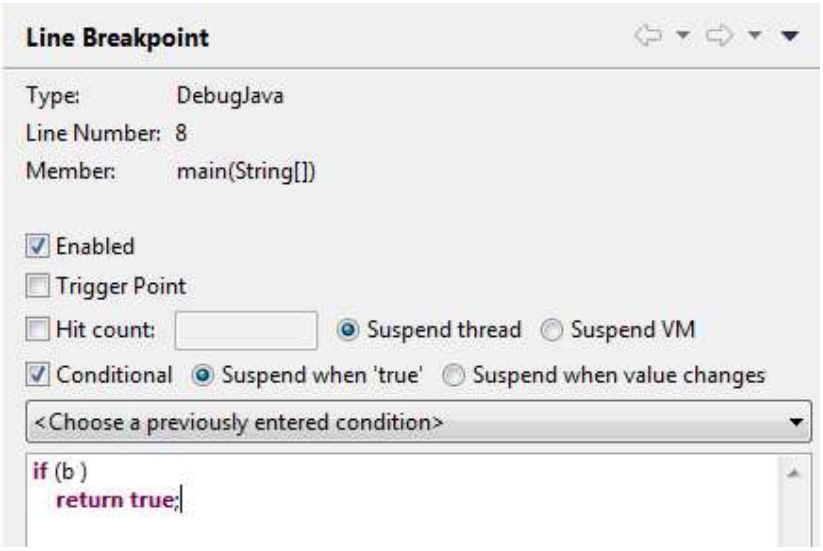


b. **Exception Breakpoints –** An exception breakpoint is specified for thrown exception using **Add Java Exception Breakpoint**.



Breakpoint for **NullPointerException** will halt whenever/wherever this exception is thrown.

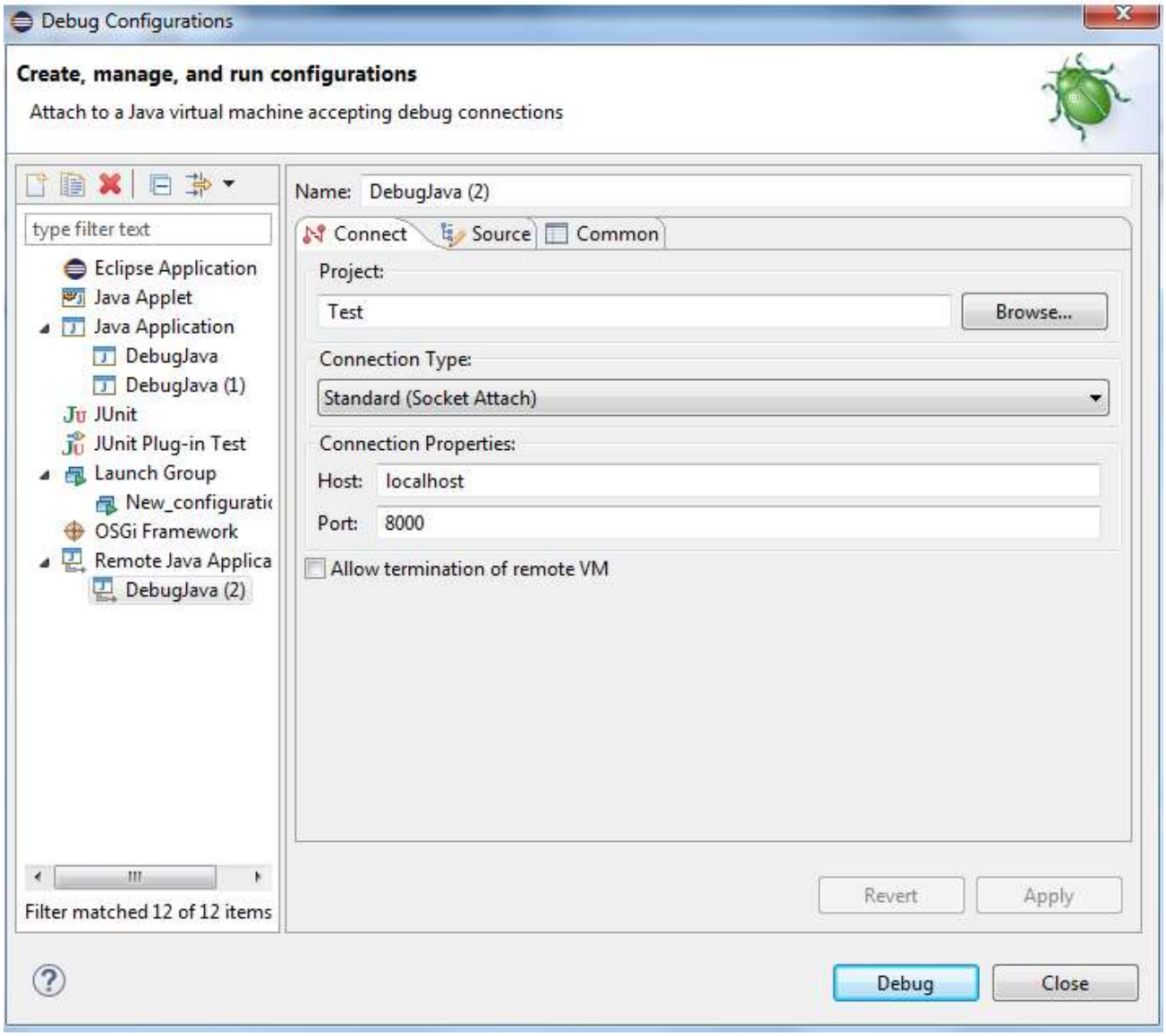
c. **Condition Breakpoints –** Eclipse users can create conditions to restrict the activation of breakpoints.



Breakpoint will be activated only if value of Boolean b is true. Hit count can be provided to halt the execution at nth hit of the breakpoint. The breakpoint is disabled until either it is re-enabled or its hit count is changed or the program ends.

2. Remote Debugging

– The Eclipse IDE allows you to debug applications that runs on another Java Virtual Machine (JVM) or even on another machine. You can create a new debug configuration of the *Remote Java Application* type. To enable remote debugging you need to start your Java application with certain flags. Connection Type can be specified as a Socket Attach or Socket Listen. Socket Listen supports multiple incoming connections.

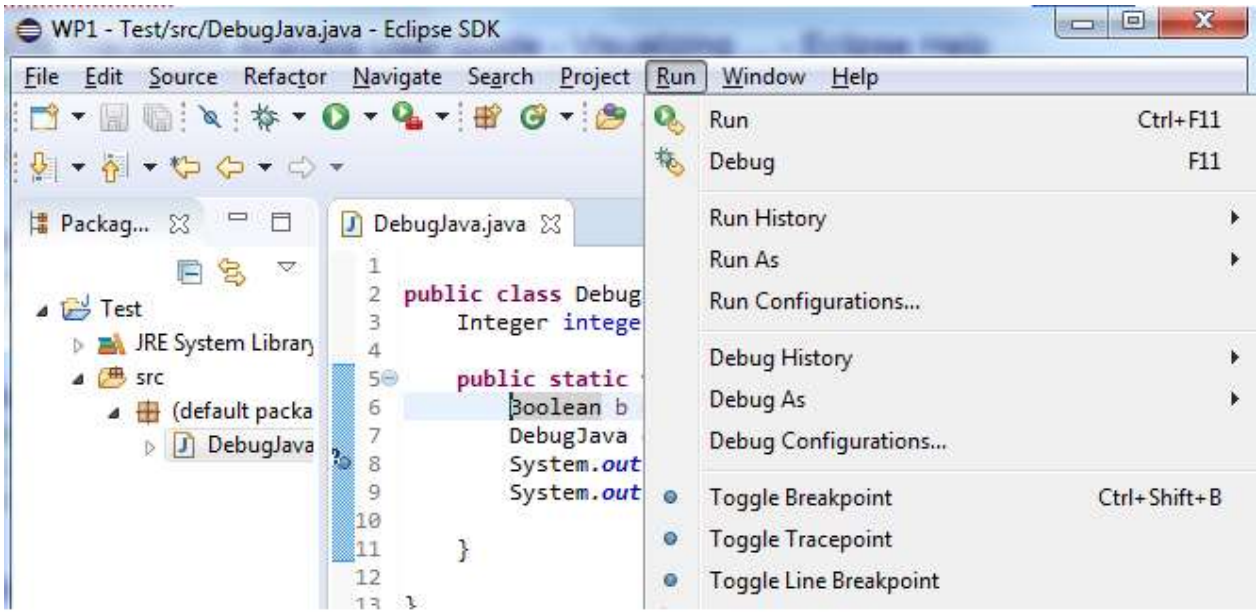


New Features in Eclipse Oxygen

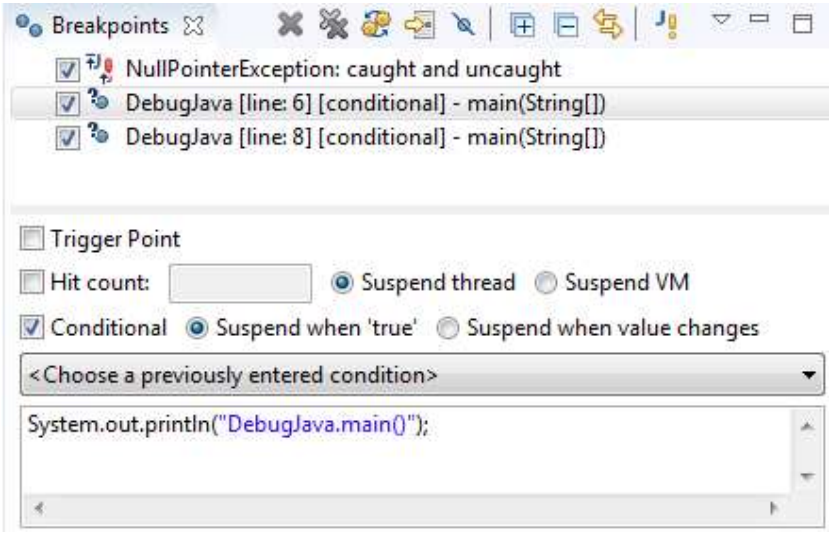
Here are the new features that have been added to the latest Eclipse Java IDE release.

1. Tracepoints

A new feature in the Eclipse Platform that allows users to creates conditional breakpoints to print out messages without halting at the breakpoints and cluttering the code base.



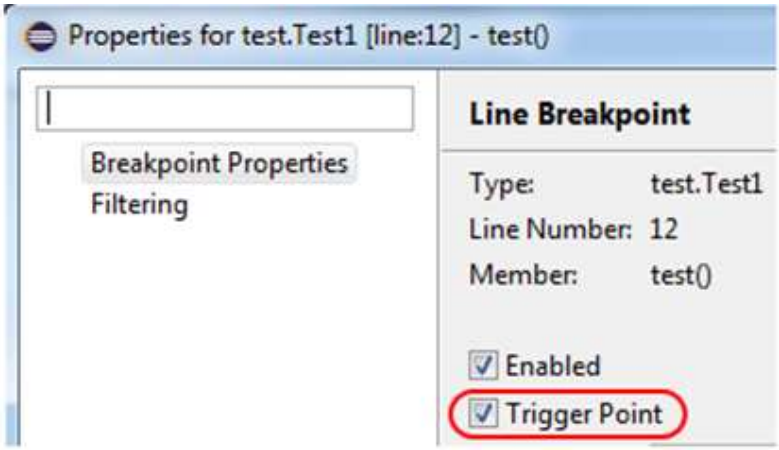
The Eclipse Platform created tracepoint with systrace template.



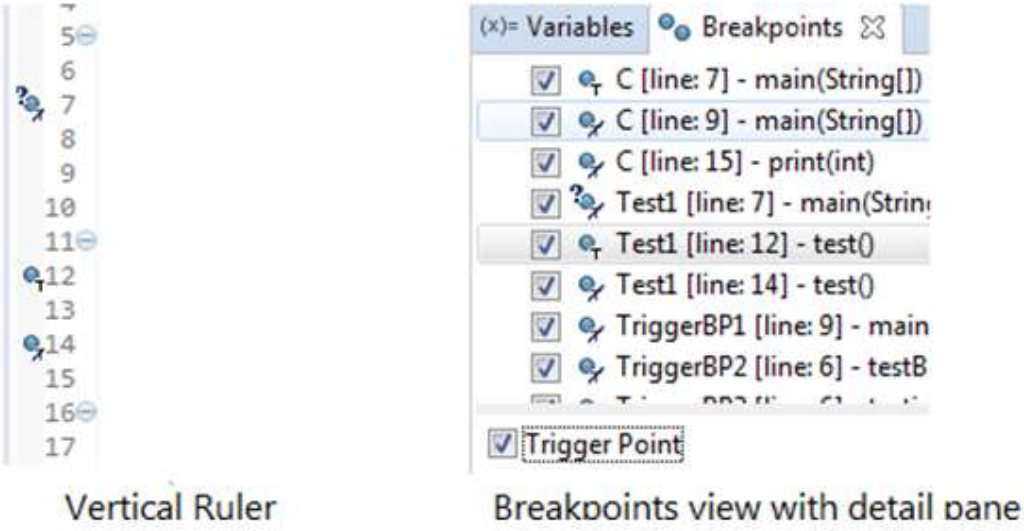
2. Trigger Point

Now users can activate *Trigger Point*. A set of trigger points can be defined for the breakpoints in a workspace.

All the other breakpoints that are initially suppressed by triggers will be hit only after any of the all Trigger Points have been hit. All the triggers are disabled after a Trigger Point is hit and will be re-enabled after the run.



Any breakpoint can be set as a trigger point by using Breakpoint Properties via the dialog or the detail pane of the **Breakpoints view**.

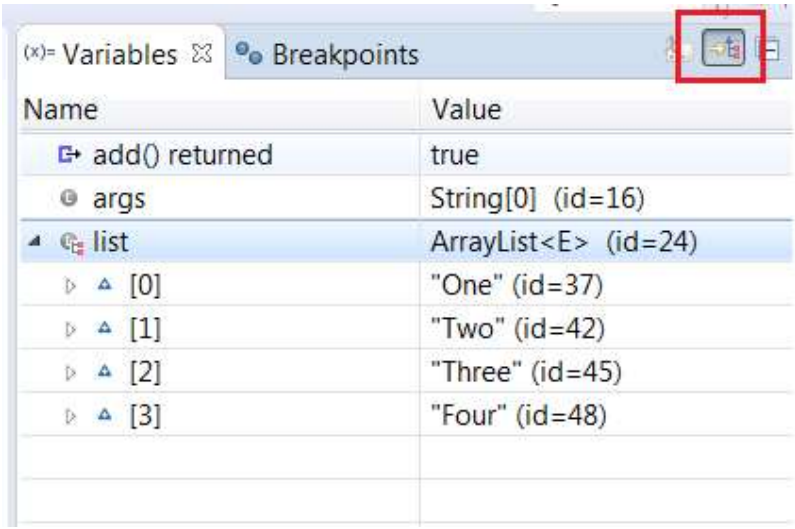


Vertical Ruler Breakpoints view with detail pane

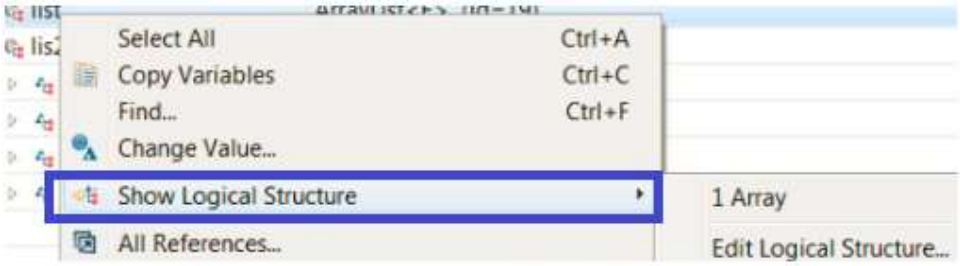
Triggers are rendered with an overlay of "**T**" and the breakpoints suppressed by the triggers are rendered with an overlay of "**T**" with a cut.

2. Logical Structures

In the **Variables view**, collection objects directly show their contained elements instead of their internal structure. Logical structures are now activated by default in the Oxygen release. **Show Logical Structure** can be turned off to show the internal structure.

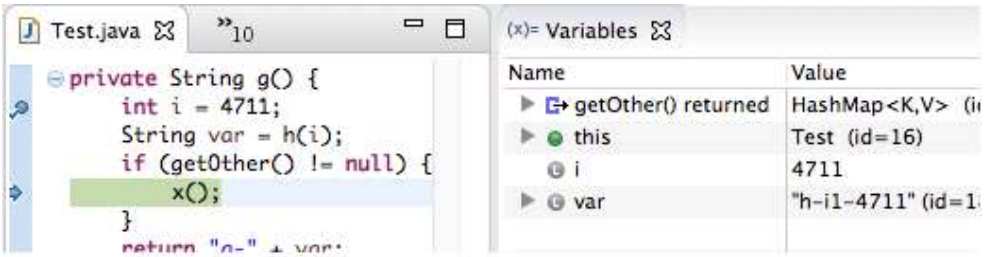


The Show Logical Structure context menu lets you create, choose or edit the representation.



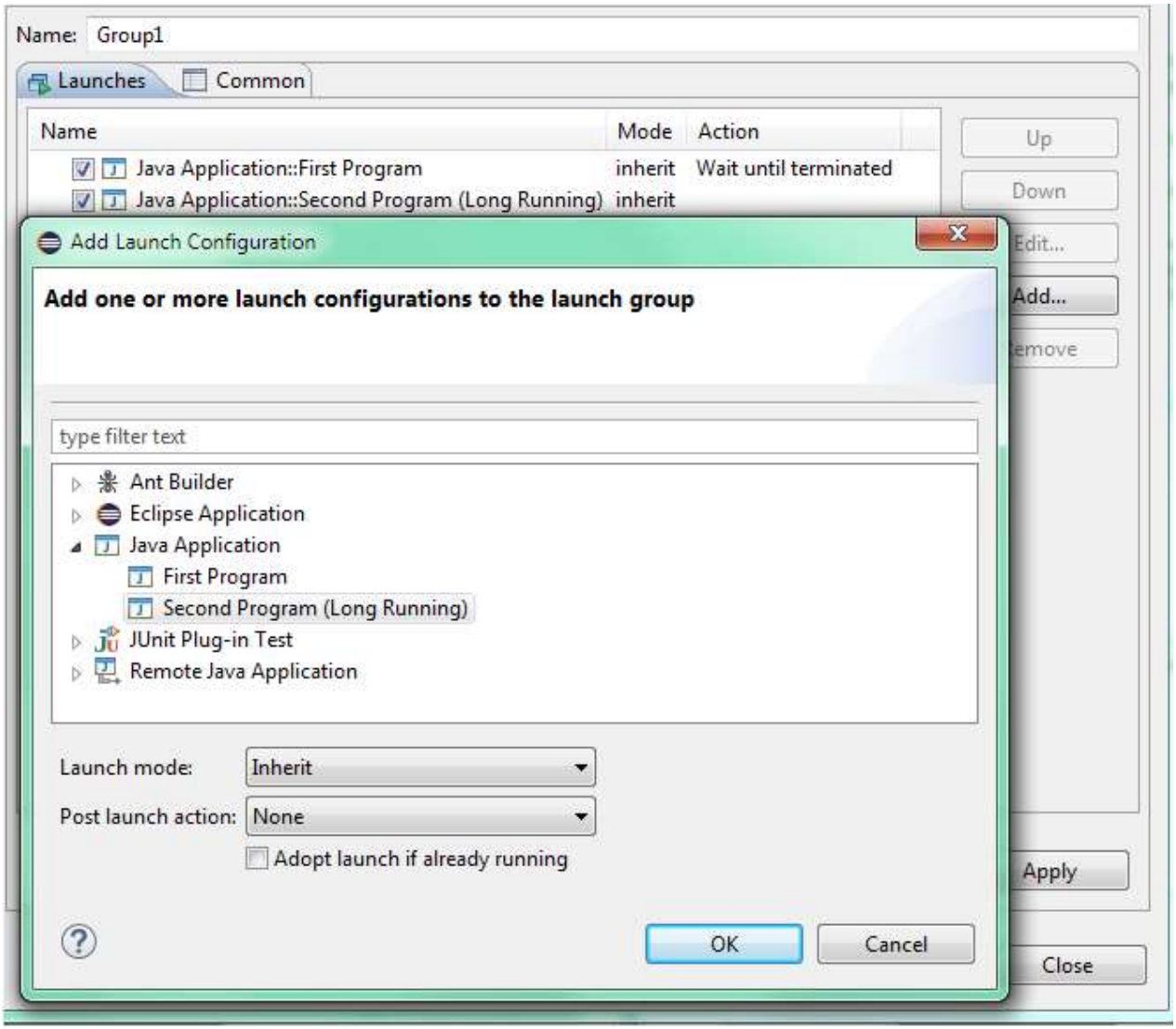
3. Method Result After Step Operation

Also new to the Oxygen release are Method Results. During debugging, the last method result (per return or throw) that was observed during **Step Into**, **Step Over** or **Step Return**, is shown as first line in the **Variables view**.



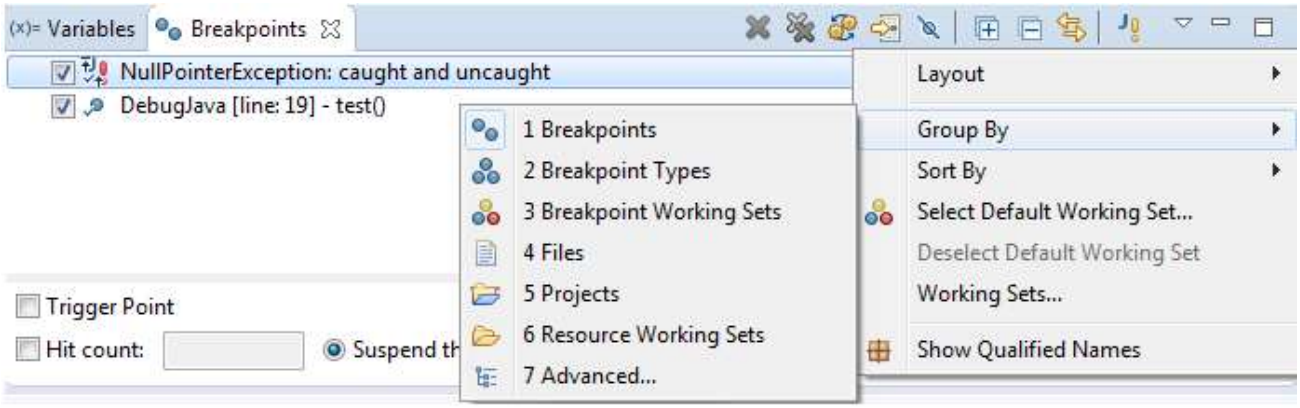
4. Launch Group

Also new in Oxygen, **Launch Group** launch configuration type allows you to launch multiple other launch configurations sequentially, with configurable actions after launching each group member. New launch groups can be created via the **Run** → **Run Configurations**... or **Run** → **Debug Configurations**... dialogs.

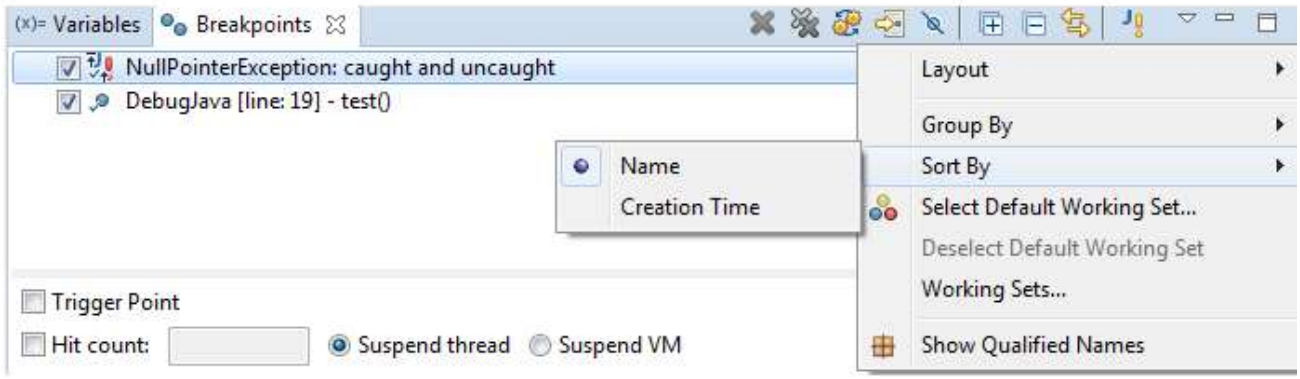


Breakpoints Grouping and Sorting

Eclipse users can now group the breakpoints by different categories in Eclipse Oxygen. Breakpoint Working Sets defines a group of breakpoints. User can perform actions like enable/disable on a working set.



Breakpoints are sorted by **Name** by default, sorting order can be changed to **Creation Time**.



For more information about debugging visit: Eclipse Remote Debugging Help (http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftasks%2Ftask-stepping.htm)

Conclusion

This article covered some of the important debugging tools and some new tools provided by the Eclipse Java IDE in the Oxygen Release. Eclipse Oxygen is now available for download here (https://www.eclipse.org/downloads/).

For in depth knowledge, user can refer to:

- Eclipse IDE Debugging Help (http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftasks%2Ftask-running_and_debugging.htm&cp=1_3_6)
- Eclipse IDE Debugging Tips and Tricks (https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftips%2Fjdt_tips.html&cp=1_5_6&anchor=debugging_section)

Authors: Sarika Sinha