

## Session 3.11

### Debug Java code Scripts in Eclipse or IntelliJ

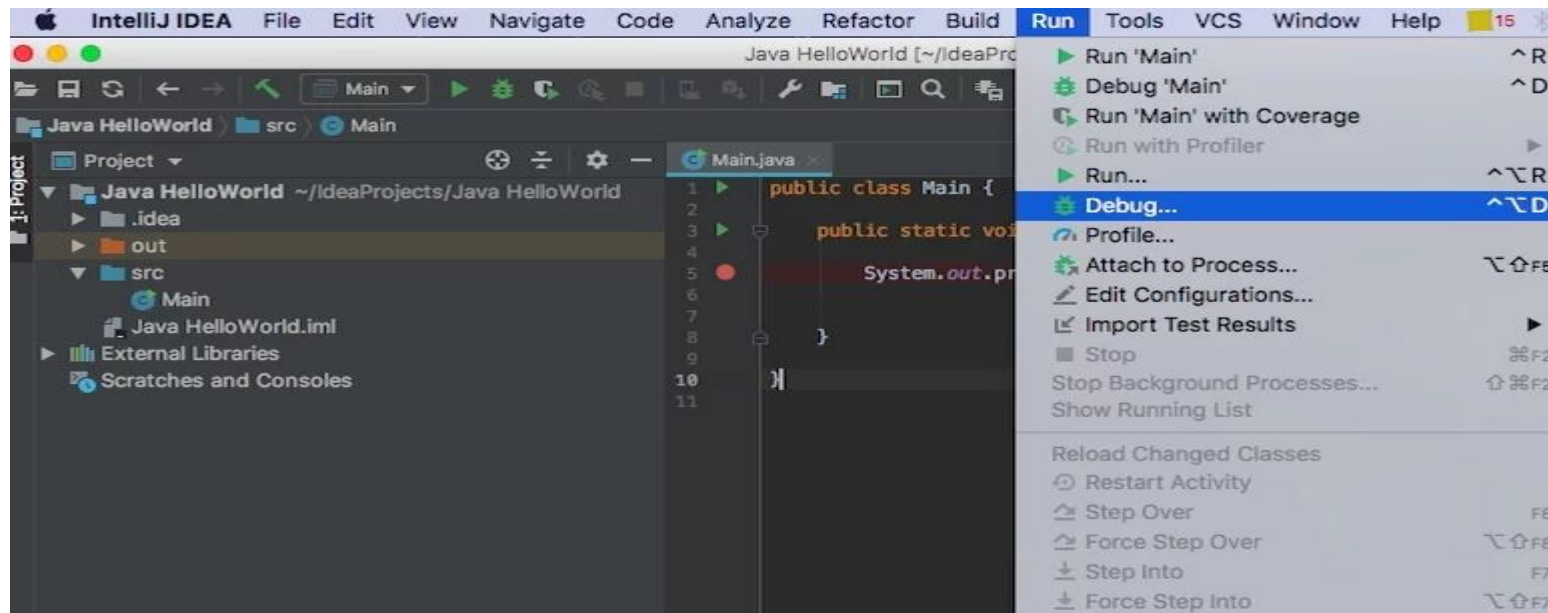
AN INITIATIVE BY

**UNICAL ACADEMY**



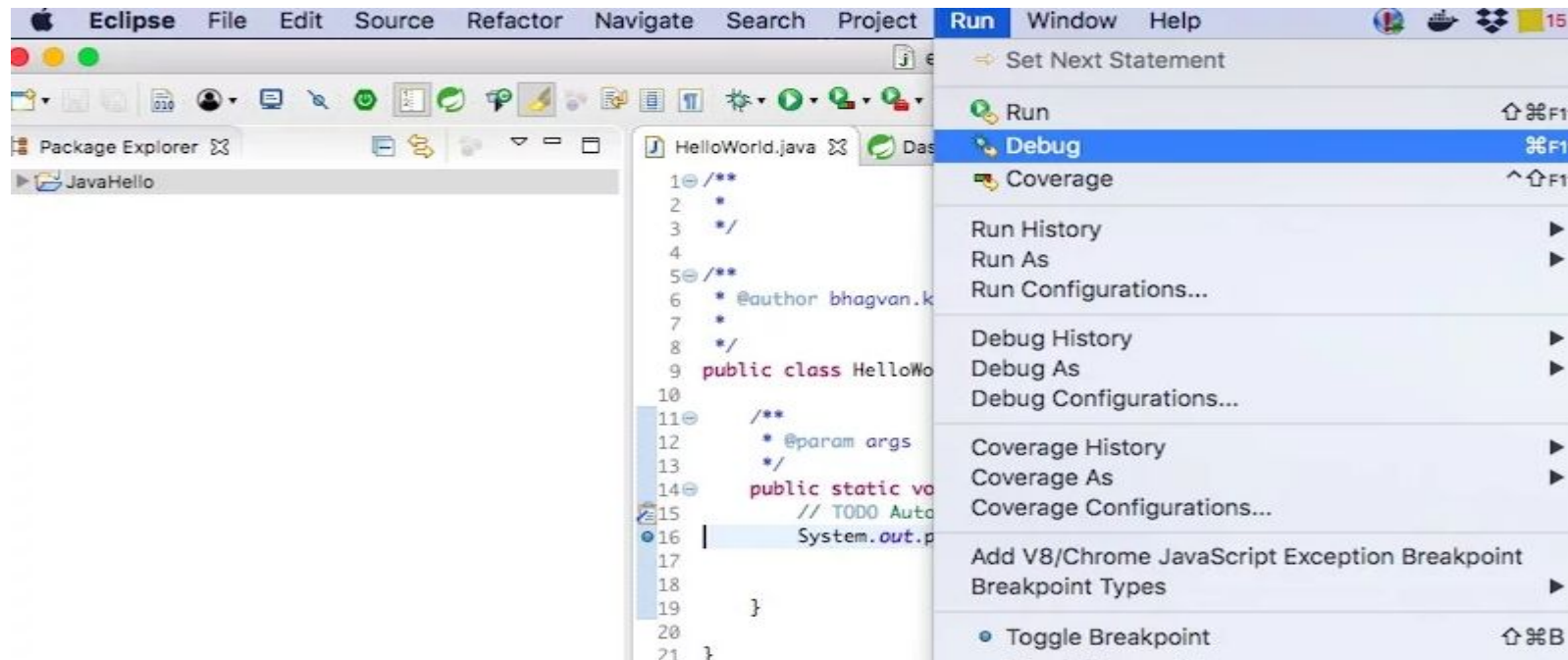
## Debugging in IntelliJ:

- In IntelliJ, you can set the debugger options choosing Settings and Preferences. Debugger will be in the build, execution and deployment section.
- You can start the debugger by choosing the run and debug configuration. Debugger can be started by setting the break point.
- Debugger can be started by selecting startDebugger menu item. You can also use the key  $\wedge D$  (Mac OS) and  $\wedge F9$  (Windows OS) for starting the debugger



## Debugging in Eclipse:

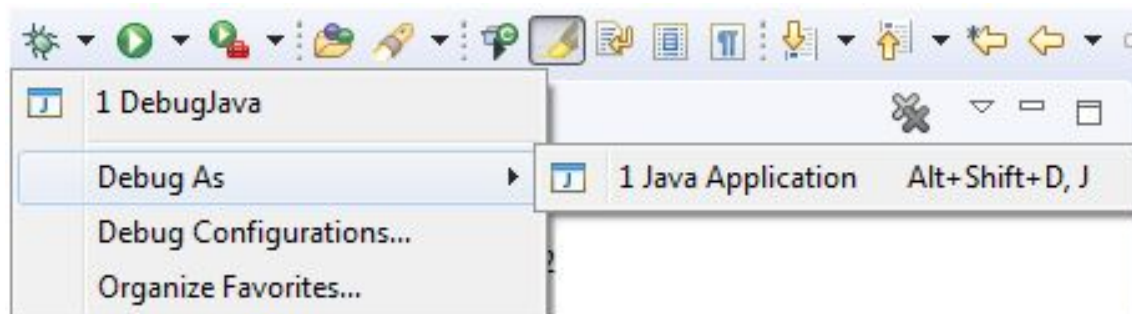
- In eclipse, you can set the break points in the source code. You can choose Toggle breakpoint in the menu to set the break point. The screen shot is attached below to show the breakpoint created.
- You can choose Run and Debug to start the debugger. Debugger starts at the breakpoint. The attached screenshot shows the debugger start menu.



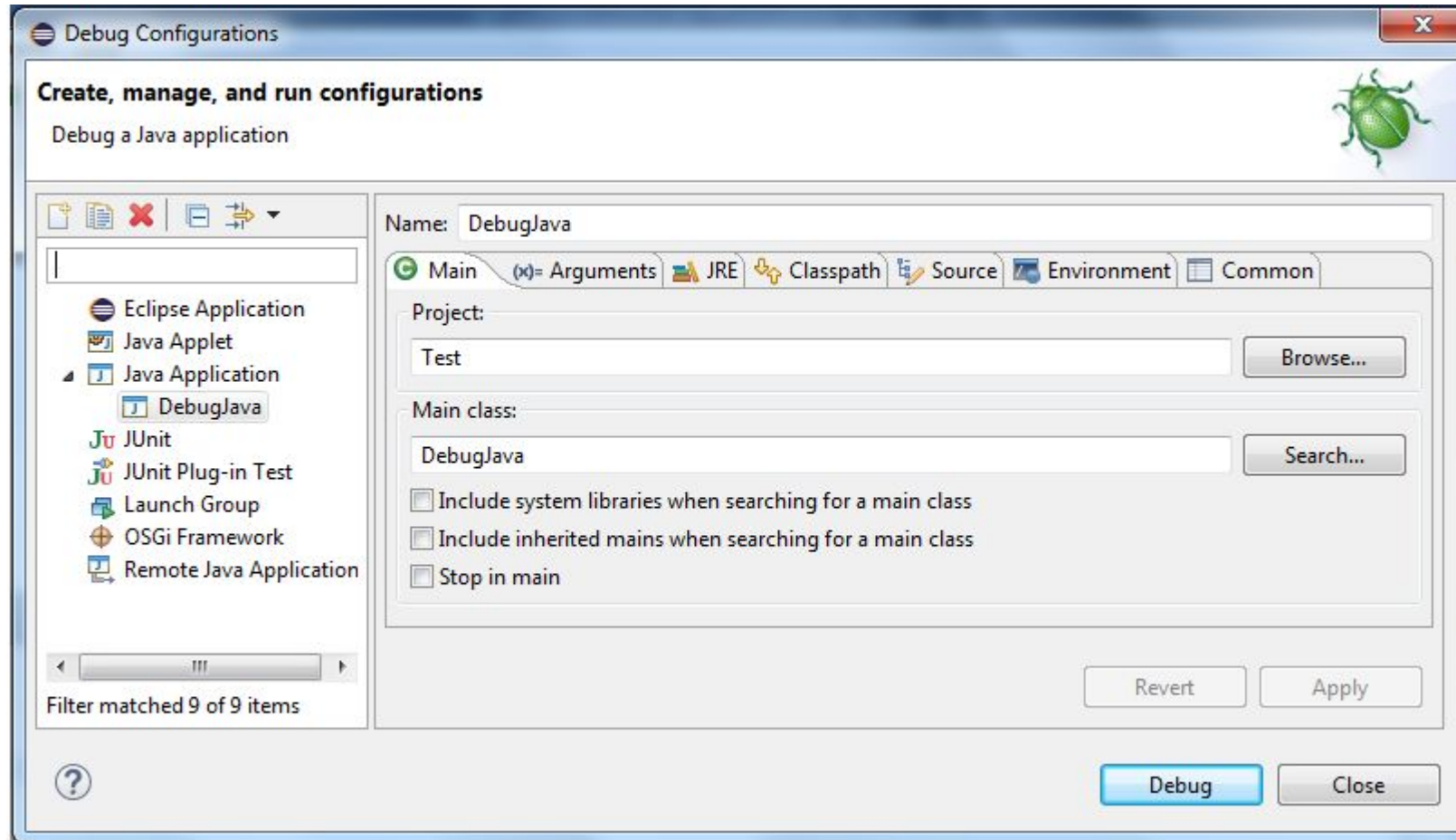
Debugging means to run the code step by step in a debugging tool like Visual Studio, eclipse, to find the exact point where we made a programming mistake. Then we can understand what corrections we need to make in the code, and debugging tools often allows us to make temporary changes so we can continue running the program.

## Debugging a Java Program:

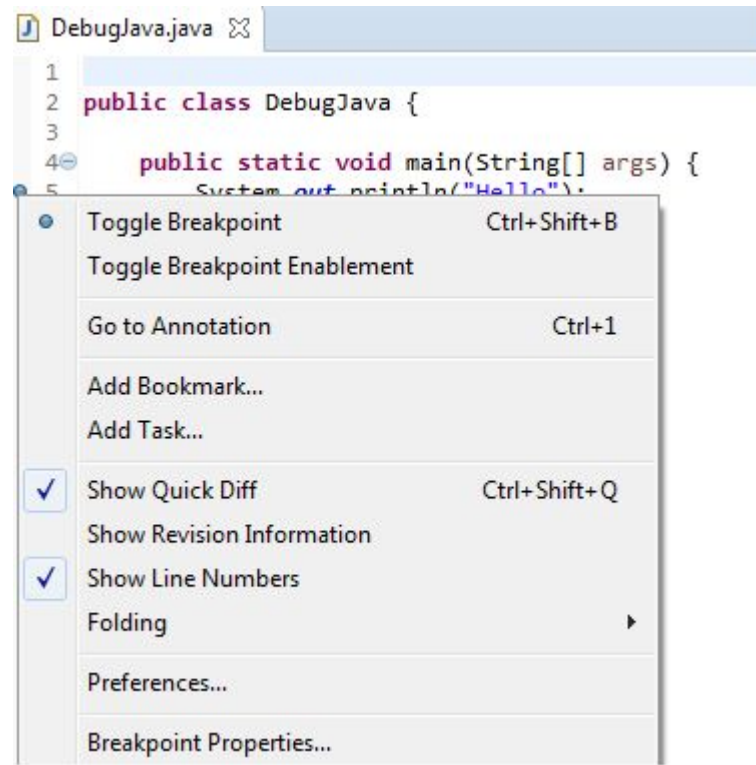
A Java program can be debugged simply by right clicking on the Java editor class file from Package explorer. Select Debug As → Java Application or use the shortcut Alt + Shift + D, J instead.



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

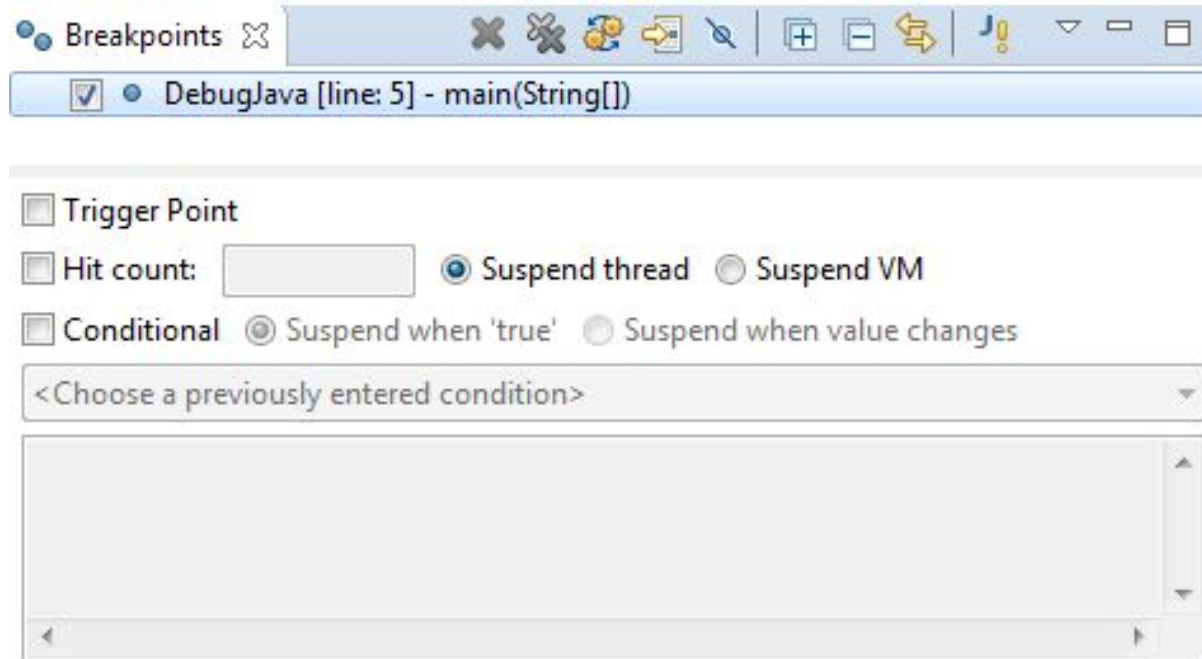


- A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point in the code.
- To define a breakpoint in your source code, right-click in the left margin in the Java editor and select Toggle Breakpoint. Alternatively, you can double-click on this position.

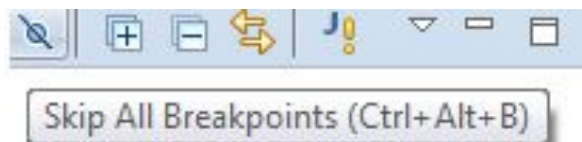




- The Breakpoints view allows you to delete and deactivate Breakpoints and modify their properties.

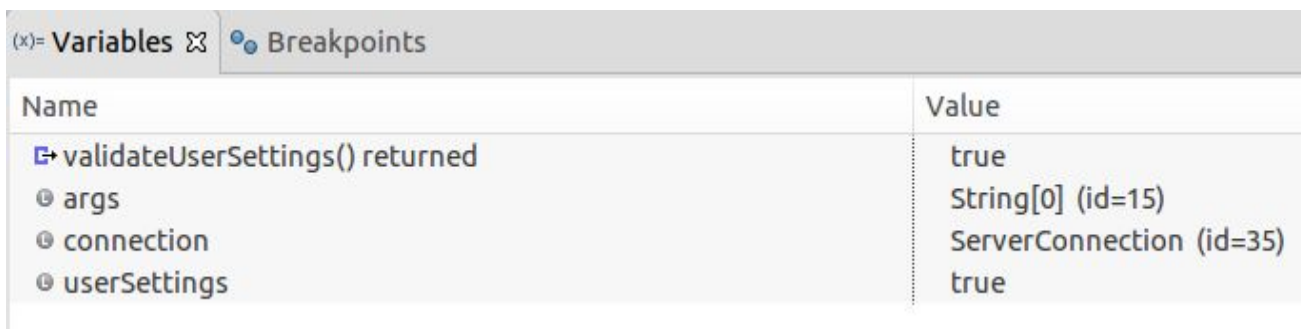


- All breakpoints can be enabled/disabled using Skip All Breakpoints. Breakpoints can also be imported/exported to and from a workspace.



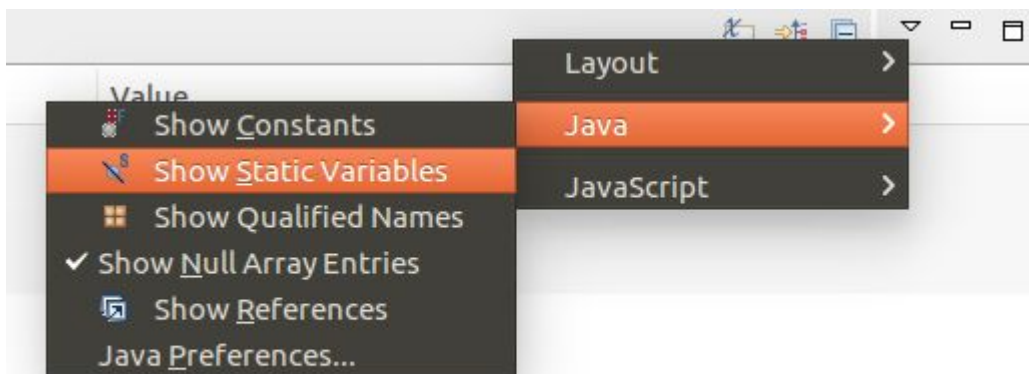


- The Variables view displays fields and local variables from the current executing stack. Please note you need to run the debugger to see the variables in this view.



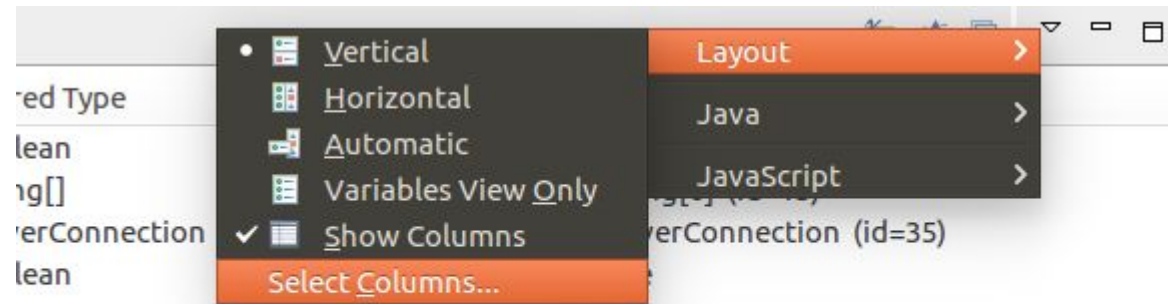
Name	Value
validateUserSettings() returned	true
args	String[0] (id=15)
connection	ServerConnection (id=35)
userSettings	true

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



- 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.



- **Step over** – An action to take in the debugger that will step over a given line. If the line contains a function the function will be executed and the result returned without debugging each line.
- **Step into** – An action to take in the debugger. If the line does not contain a function it behaves the same as “step over” but if it does the debugger will enter the called function and continue line-by-line debugging there.



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
Ctrl + F5 / Ctrl + Alt + Click		Step Into Selection

## Session Recap

