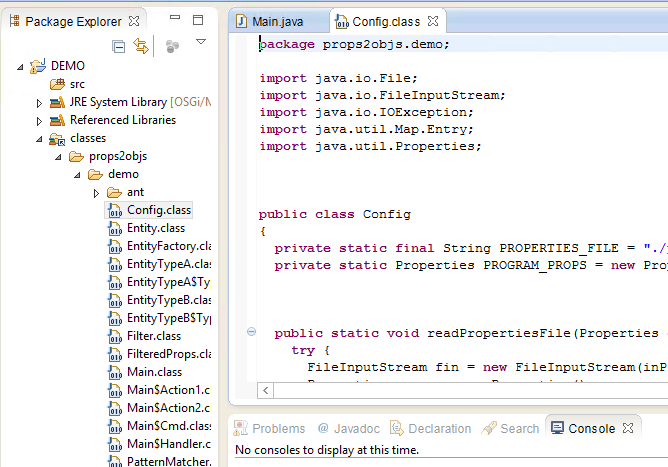
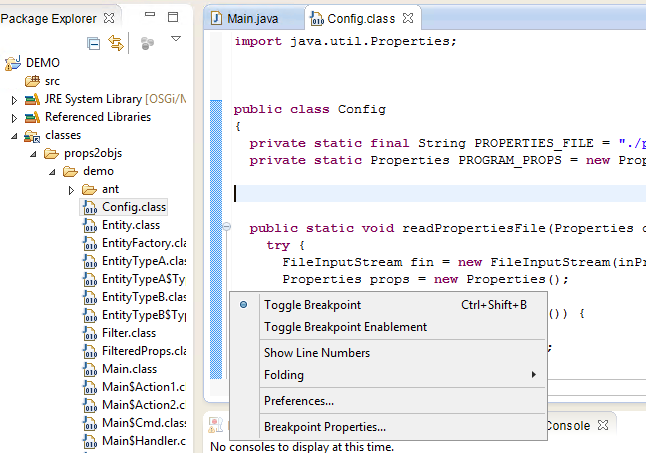
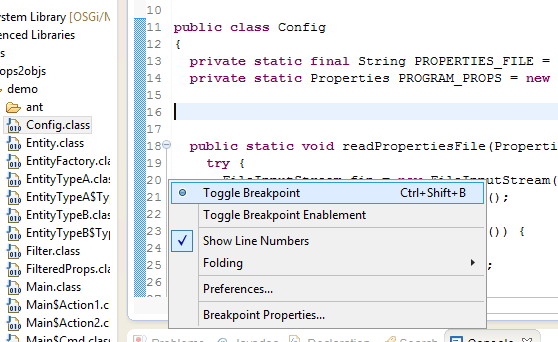
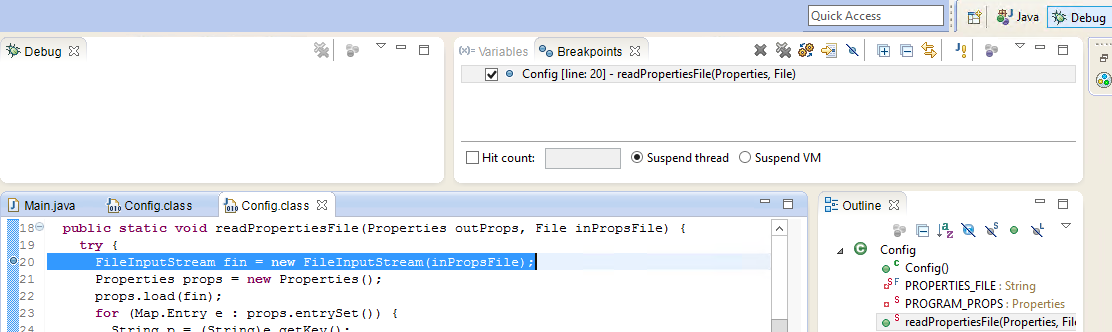
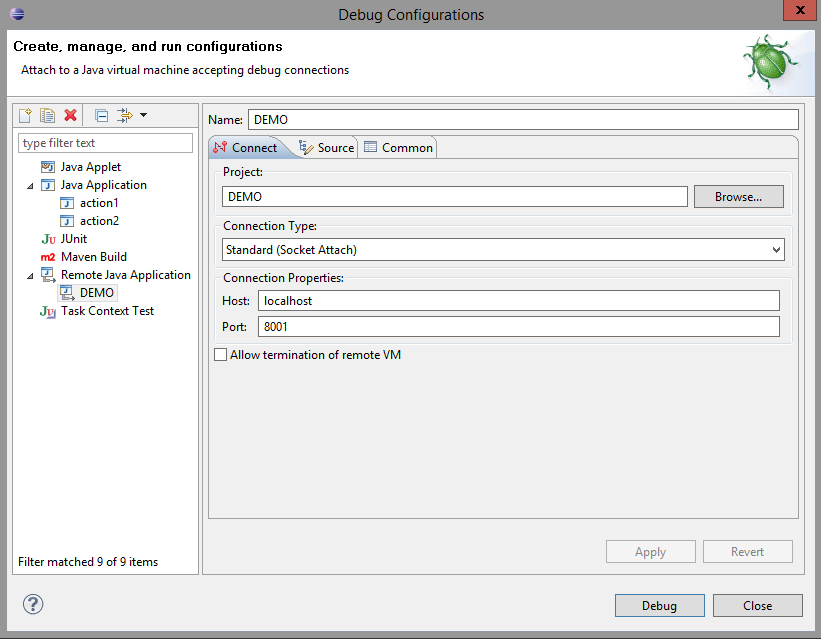
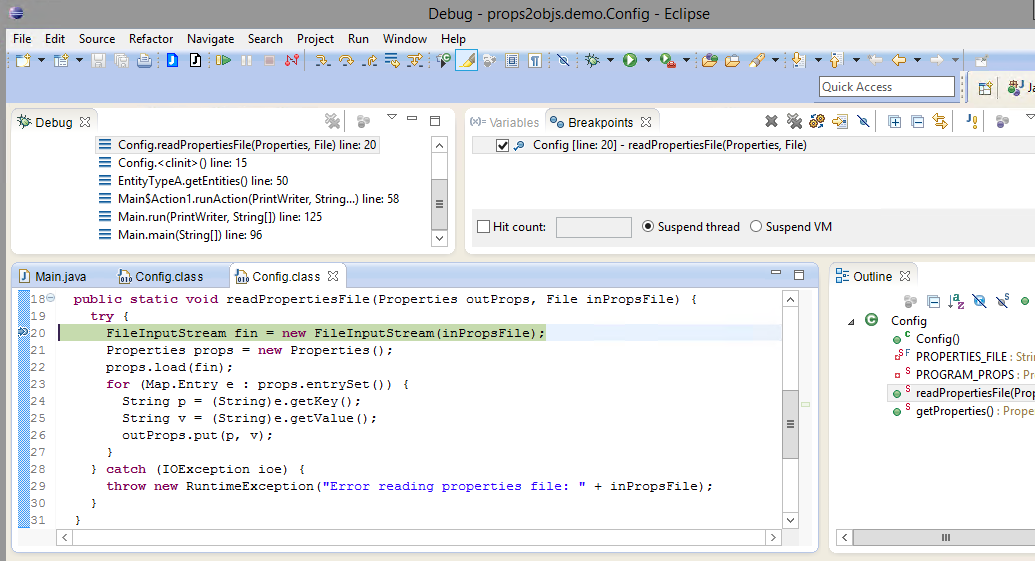
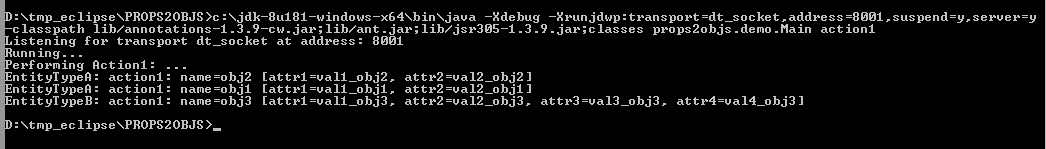
**Use (my special) Eclipse to debug a Java program with only the class files**  
**Introduction**We have only the class files and jar files necessary to run our program as follows:

java -cp lib/annotations-1.3.9-cw.jar;lib/ant.jar;lib/jsr305-1.3.9.jar;classes props2objs.demo.Main action1  
  
*We now want to debug this program in Eclipse, but we don’t have the Java source files. With my special Eclipse that includes a Java decompiler plugin, we can do this as follows…*  
  
o **Create a Windows batch file** that launches the program with some special arguments, so that the program waits for a debugger (Eclipse) to attach to it:

|  |
| --- |
| c:\jdk-8u181-windows-x64\bin\java -Xdebug -Xrunjdwp:transport=dt\_socket,address=8001,suspend=y,server=y  -classpath lib/annotations-1.3.9-cw.jar;lib/ant.jar;lib/jsr305-1.3.9.jar;classes props2objs.demo.Main action1  Listening for transport dt\_socket at address: 8001 |

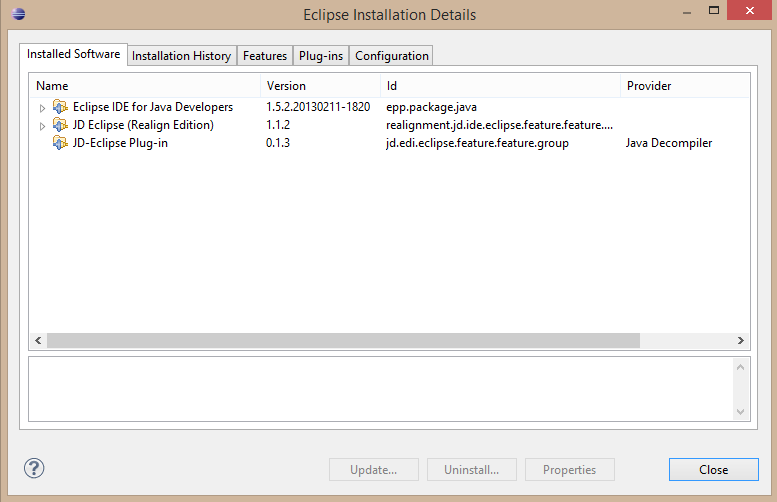
o **Create a project in Eclipse**:  
File > New > Java Project  
Give it a name, e.g. “DEMO”, and click “Finish”.  
  
o Select the “DEMO”project in Eclipse and open “Project > Properties > Java Build Path”.  
- Select the “Libraries” tab.  
- Select “Add Class Folder…”  
- Click “Create New Folder…”  
- Click “Advanced >>”  
- Tick “Link to folder in the file system” and use the “Browse…” button to select our “classes” directory. Click “OK”. Click “OK” again. Click “OK” one final time.  
  
**NOTE**: There should now be a “classes” folder under the “DEMO” project, and double-clicking on any of these files should invoke the decompiler plugin and display the source code decompiled:  
  


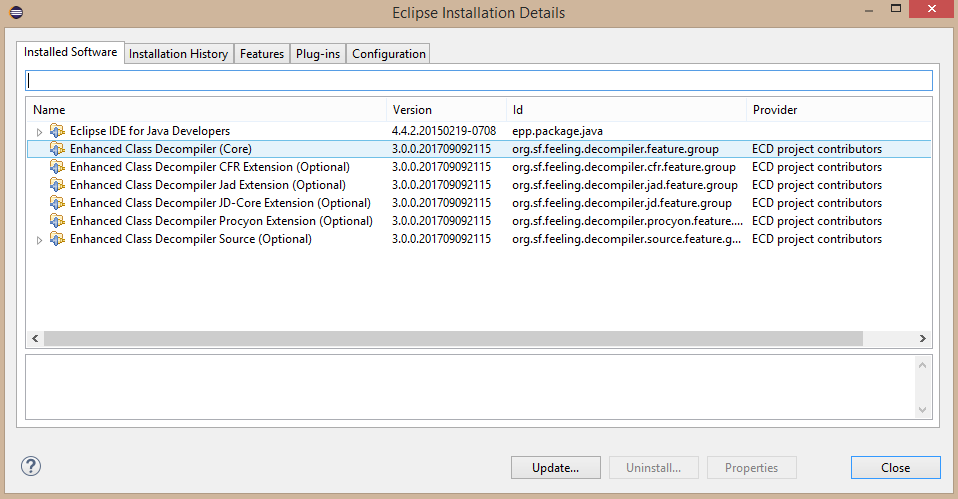
o **Set a breakpoint in one of the class files.**- In Config.class, right-click in the left margin to display line numbers:  
  
- Right-click on line 20 to add a breakpoint:  
  
- Select the “Debug” perspective and verify that the breakpoint is listed in the “Breakpoints” tab (Note: double-clicking the breakpoint in this tab will highlight the breakpoint in the code):  
  
  
  
o **Select “Run > Debug Configurations…”.**  
- Right-click on “Remote Java Application” and select “New”.   
- Give it some name, e.g. “DEMO” and click “Apply”:  
- Change “Port” to 8001, because that’s what we used earlier to launch the Java program.  
   
- Click “Debug” and it should attach to the Java program, run it, and hit our breakpoint:  
  
  
**NOTE:** The output from the program will appear on the command-line, i.e. where we started the Java program:  
  
  
  
 **--\*\*--**  
If you need to run your Java program as a specific user, you might need to do something like this:

runas /profile /user:edp\edpadmin1 cmd.exe

Or you might have a customer scritpt for launching the Java progam, e.g.:

catalina.bat jpda start

**--\*\*--  
  
My special Eclipse versions with a decompiler plugin pre-installed**(Help > About Eclipse > Installation Details).  
  
  
**NOTE:** These free decompiler plugins date back several years now, meaning that they cannot handle more recent Java (8) features such as generics. So don’t expect the decompilers to be able to decompile every class file you have! You may see some blank sections in the decompiled source file - but you can still step through these blank lines and view the variables in the debugger. Switching between my two special Eclipse versions (juno and luna) might give you better results decompiling certain problem files.  
  
Eclipse **Juno** with decompiler plugin (eclipse\_juno\_cw\_debug.zip):  


Eclipse **Luna** with decompiler plugin (eclipse\_luna\_cw\_debug.zip):  
  
  
**NOTE:** When using the luna Eclipse, I've noticed that the breakpoints I set in the decompiled source view don't seem to match the lines highlighted when I double-click on these breakpoints from the Breakpoints tab in Eclipse. I usually end-up resetting the same breakpoints but from the Breakpoints tab view. These are the ones that are actually hit.  
  
*JeremyC 22-11-2018***END**