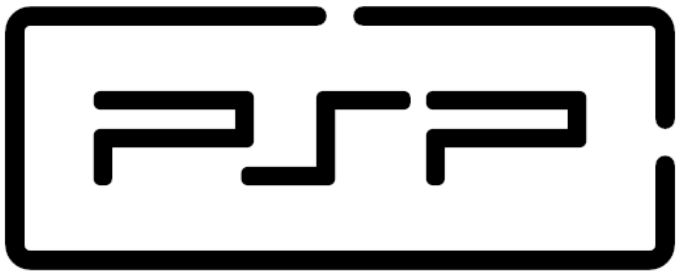


Process and Service Programming

3.5 Annex I - Debugging multithread apps in Netbeans

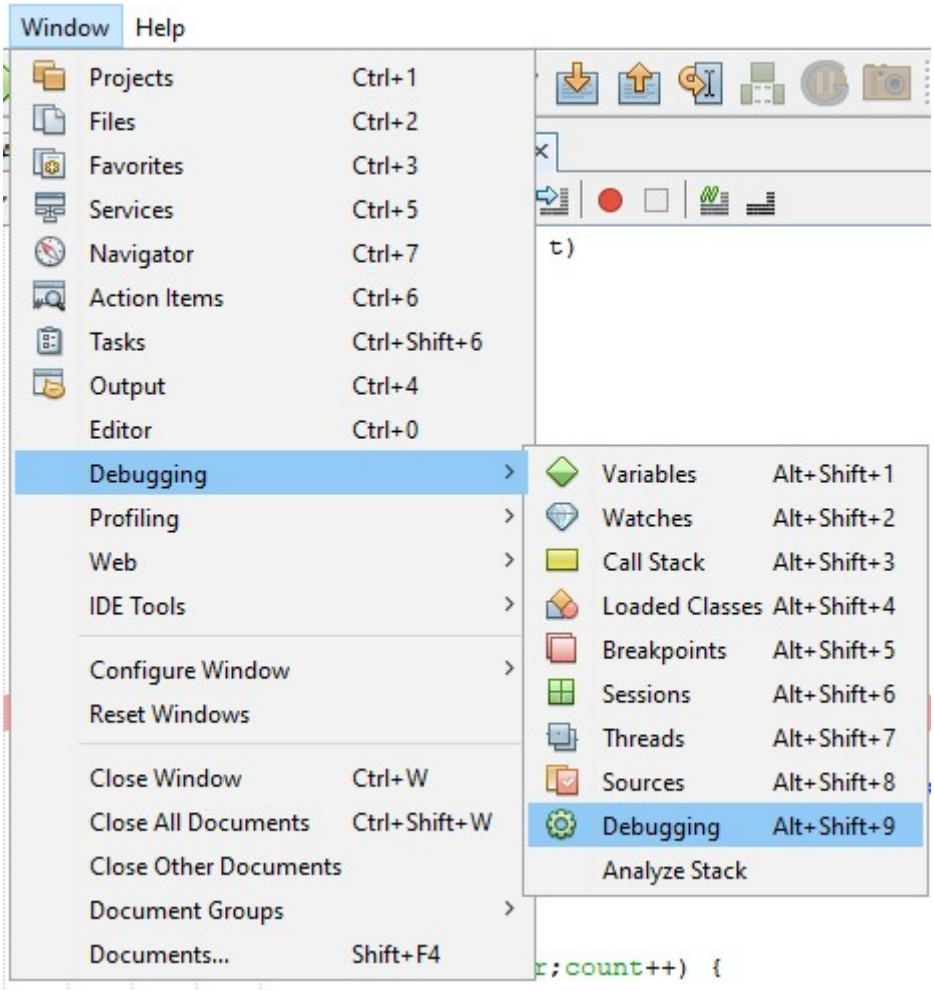
I.E.S.
Doctor Balmis

3.5 Annex I - Debugging multithread apps in Netbeans

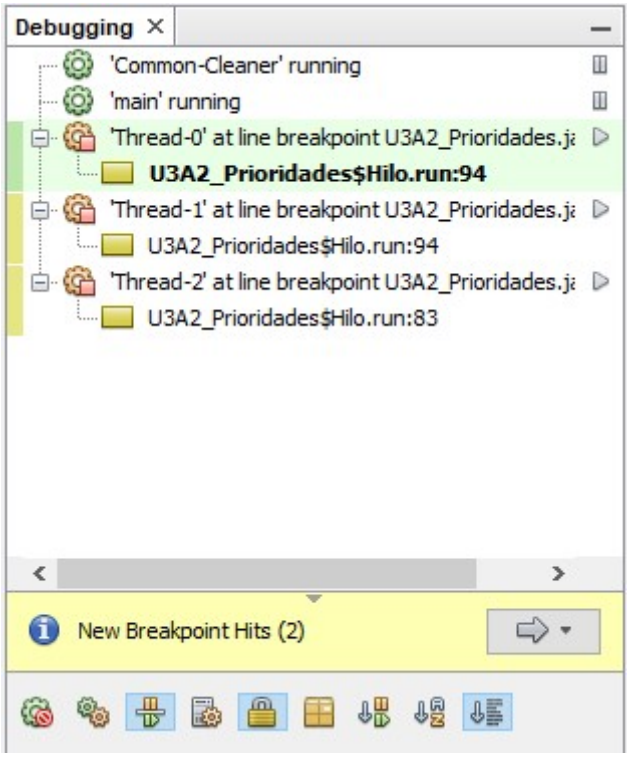
Modern IDE provide the user with facilities to debug multithread apps.

Just like in monothread apps, we can use breakpoints to stop a thread execution to inspect object state, properties values, etc.

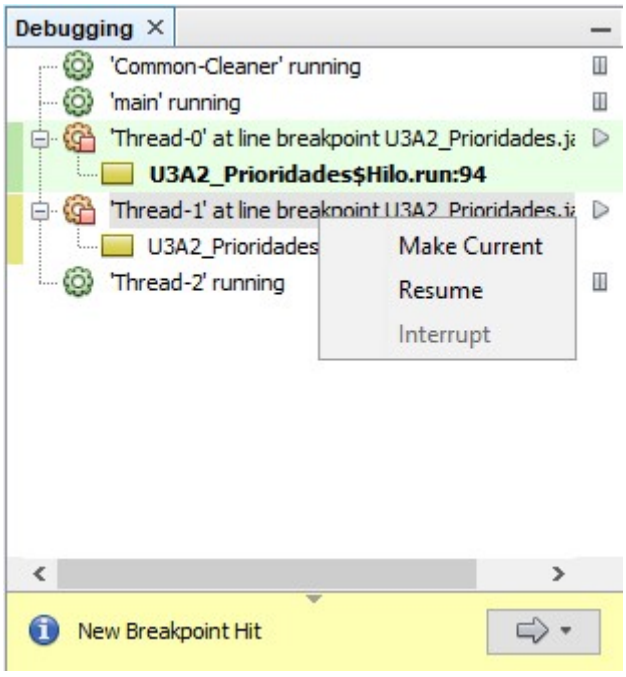
If we want to use the Debugging window in NetBeans IDE to debug multi-threaded applications we have to activate it by setting it from menu *Window* > *Debugging* > *Debugging (Alt+Shift +9)*



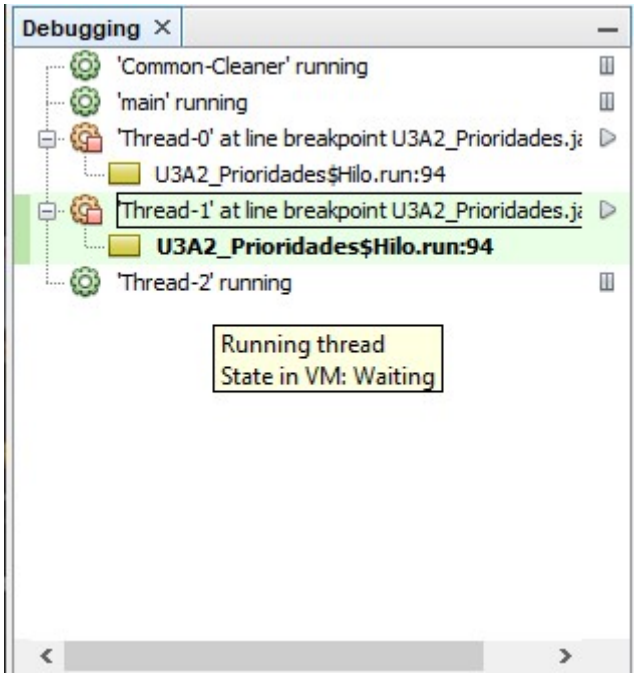
The Debugging window simplifies the debugging process by integrating into one window the information about debugging sessions, application threads and thread call stacks. The Debugging window enables you to easily see the status of application threads and suspend and resume any of the threads in the session.



- The *current thread* is indicated by a green bar in the margin (it is also highlighted in green). The current thread is the one we can work on by using StepInto, StepOver, Pause, Continue actions from the debugger. We can also access the variables inspection on that thread.
- Threads that invoked the notification, by hitting a breakpoint, are indicated by a yellow bar and the thread icon (orange color) indicates that the thread is suspended by a breakpoint.
- In the right side we have a quick access to Resume/Pause each active thread.

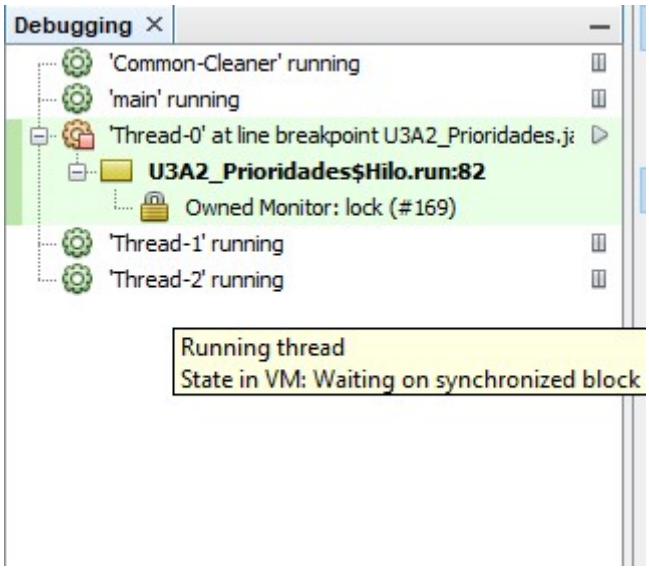


Clicking with the right mouse button over any thread we can make it the *current thread*. This way we get control over it and we can inspect this thread while other are paused or keep on running.



Furthermore, the cogwheel next to each thread identifier is giving us many information about a thread state.

- When the cogwheel is orange that means the thread is suspended and it requires our attention.
- When the cogwheel is green that means the thread is running. If we move the mouse over the thread we can get a tip showing the thread state information. As you can observe in the image thread-2 is running but it is in a waiting state. Actually this thread is locked in a join (waiting for another thread to finish)



Finally, thread debugging helps us with synchronization using monitors. We can know when a thread owns a monitor (lock) and we can also know which monitors (locks) a thread is waiting for.

As an additional tool Netbeans provides an utility (Debug > Check for deadlocks) that checks if any deadlock has happened, telling the monitors owned by each thread and the monitors each thread is waiting for.