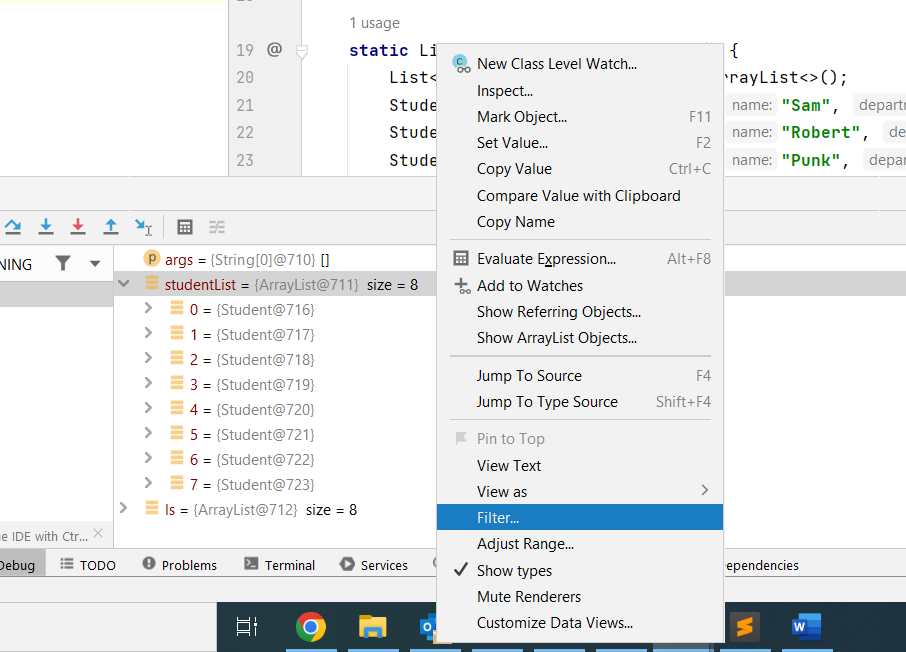
**Udemy Course: Java Debugging with IntelliJ Idea**

**Debugging Filters: -**

When we have large data coming, then we can filter that data based on condition.

**Ex:** We have studentList, lets break it on age > 15.



A screenshot of a computer

Description automatically generated

**Exception Breakpoint: -**

Suppose there is a loop and at certain point exception is coming instead of moving debugger for the entire loop, we can create Exception Breakpoint under **[View Breakpoint]** because of this our debugger will always stops at null pointer exception.

A screenshot of a computer

Description automatically generated

**Action Breakpoint: -**

Whenever we want to do some evaluation in code without making any pause during debugging, in that case we can use Action Breakpoint.

This breakpoint can be placed by **Shift Click**; it will appear in yellow mark.

We can remove the Breakpoint hit message & write expression in Evaluate & Log section. In that way debugger won't stop anywhere but evaluate the expression & print it.

A screenshot of a computer

Description automatically generated

**Force Return & Throw Exception: -**

While debugging, we can go inside a particular method & return a specific value or thrown an exception even though the normal implementation returns some other value.

A screenshot of a computer

Description automatically generated

**Field Watchpoint: -**

These watchpoints are placed on the instance variable & when any modification or access will be done on them, debugger simple pause there.

**Drop Frame: -**

Drop Frame is the feature that is used to rewind the stack in debugger.

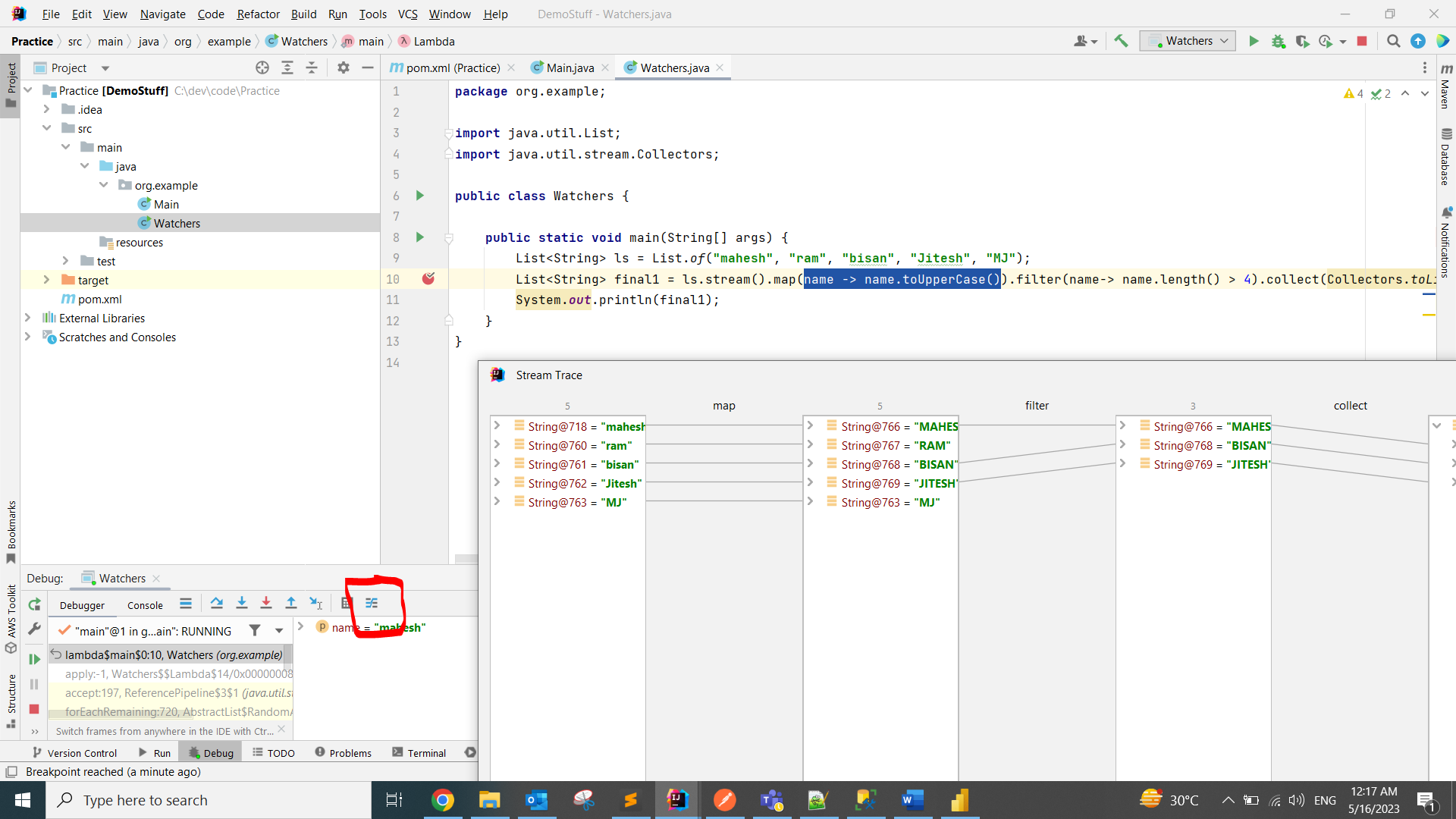
Suppose while debugging the code, we missed a critical part to debug then we can drop some frame & reach back to previous stage.

**Note:** Only the values of local variable will be reverted, but global variable will always be same.

**Debugging the Streams:**

We can debug the streams in java by putting the breakpoint on either of its intermediatory functions.

Also, once we do debug the streams, we have Stream Trace enable which give better clarity about the stream.



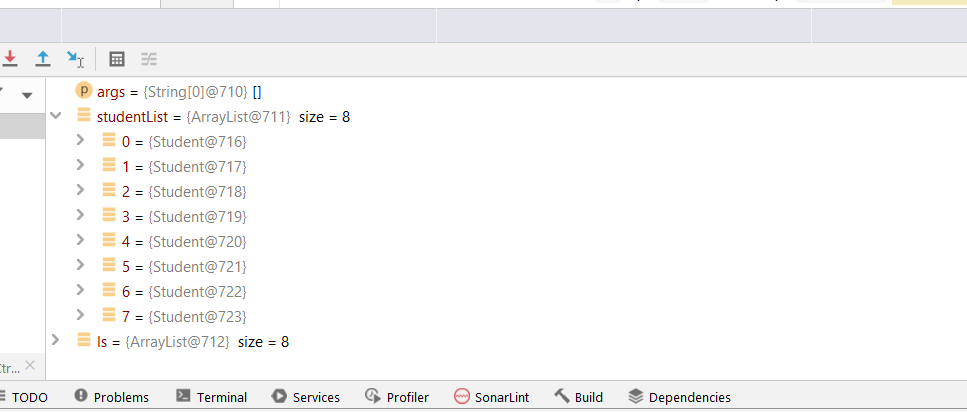
**On Demand Custom Data Type Renderer**

Whenever our object is containing too many values & we want to render it into format that we want in the console, then we can use renderer.

Here, we can explicitly select any class & defined the format in which we want to display its object in the console.

**Ex:** In studentList, we want to only see studentName and age inside the debugger.

**Note:** This could decrease our performance though, used it when necessary.



A screenshot of a computer

Description automatically generated

