

# Using the IntelliJ Debugger

COMSM0086

Dr Simon Lock & Dr Sion Hannuna

# Apologies

Sorry if you are already using the debugger

Aim of this briefing is to convince you to switch !

Even if you are already using a debugger,  
you might still learn something new !

# What is the aim of debugging ?

Code can frequently become a VERY complex beast  
We often don't understand exactly what it is doing !  
Subtle run-time behaviour can cause strange results

Problem is that code usually runs as a 'black box'  
(We can't actually SEE what is going on inside)  
We really need to gain some kind of transparency...

```
System.out.println("Value of counter is: " + counter);
```

# Problems with Println

Most programmers start out using printf or println  
Provides basic but workable approach to debugging

You have to decide *\*which\** variable to print out  
Then add some temporary code to print it out  
Maybe even a loop to iterate (if its an array)

If you chose to print out the wrong variable  
(or just want to switch to another variable)  
You have to go through the whole process again

Not forgetting to delete all the printlns afterwards !

# Problems with volume of Data

I've written a Java renderer to render 3D models:

vertex-data

But there is a fault reading in the 3D vertices !

After we have read in data from the file

Let's print out position of each vertex

BunnyViewer

For professional level programming

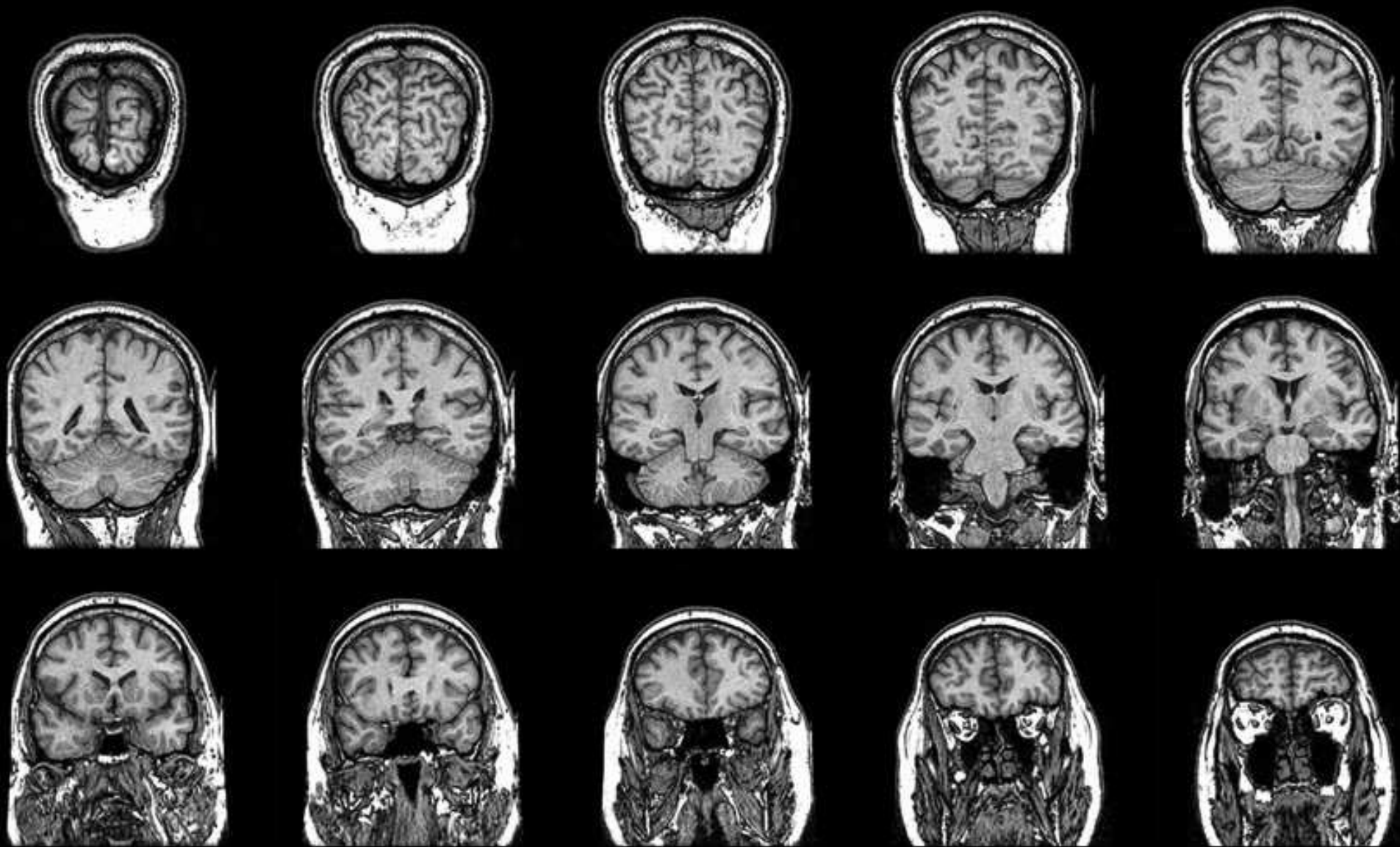
We need to use professional development tools

# Debuggers

Debuggers are like MRI scanners with freeze-frame !  
We can pause execution and see EVERYTHING inside

Look at content of ANY variable  
CHANGE that content manually  
Pause/resume execution  
Step through execution...  
...line by line





# Demonstration

We have some code to calculate the first 10 primes  
The problem is this code doesn't work properly :o(  
The code just loops forever and prints out nothing

Let's take a look at the "Primes" project in IntelliJ  
The use IntelliJ's debugger to find the problem !

Don't shout out if they see any faults in the code !

IntelliJ

# Switching over

Using printf/println probably feels safe and familiar  
Problem is that its tempting to continue using it...  
Long after it has become inefficient & ineffective

The learning overhead of switching to a debugger  
Often discourages people from making use of them

However the time invested will reap greater rewards  
Think about your longer-term career development !

# Medical Metaphor Revisited

If a debugger is a bit like an MRI scanner...

Then using println is a bit like:

hitting different body parts with a little hammer

MRI is more complex to operate

But a lot more powerful !

(once you get the hang of it)

Why wait until now to introduce the concepts ?

TBH some are still getting to grips with the hammer ;o)

## Pro User Features...

# Pro User Features: Reset Frame

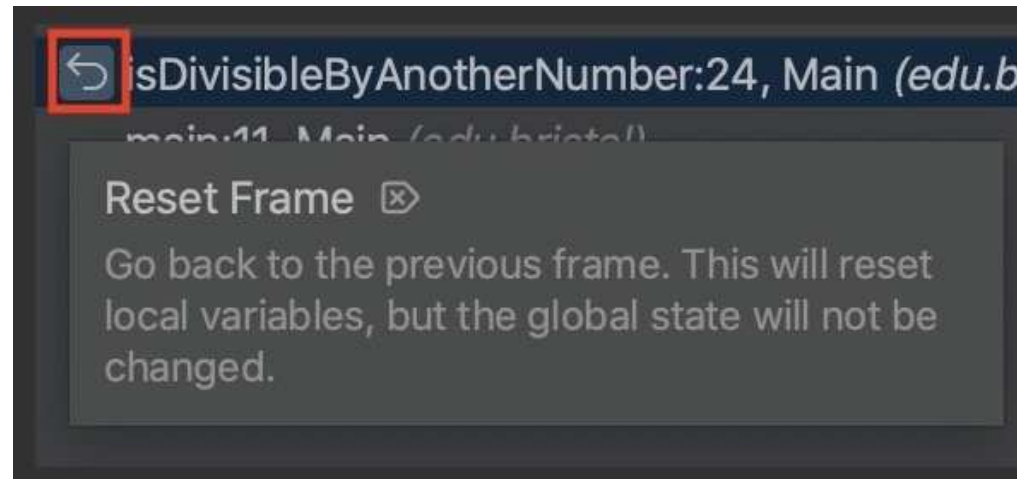
As close as you can get to "step back"

Returns back from the current method

(as though that method had never been called)

There is no true "step back" (rewind)

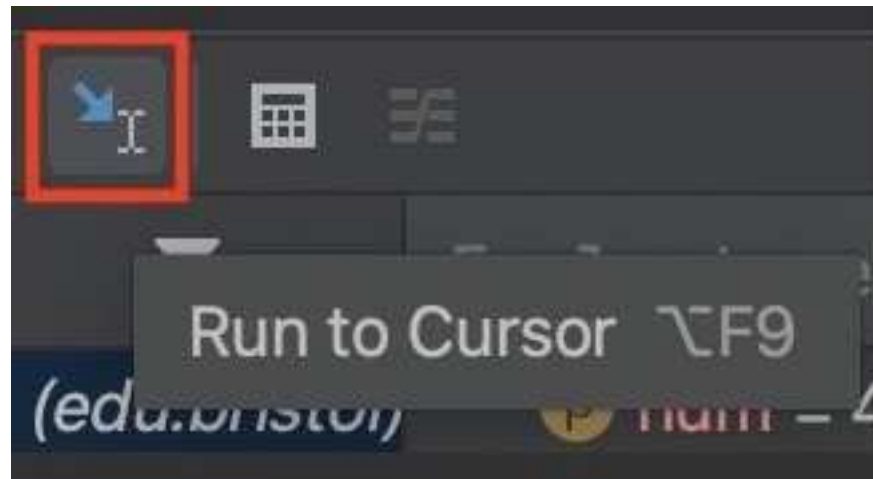
I guess it is just too hard to implement !



# Pro User Features: Run to Cursor

Unpauses execution, then continues running until...

The currently selected line of code is reached  
(basically like creating a temporary breakpoint)





It's up to you - you decide !

Let's take a look at this week's workbook  
**README**