CS206

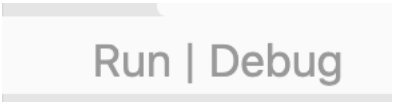
Breakpoints and debugging in VSC

This document assume you are using Visual Studio Code (VSC) with the “Java Extension Pack” installed. If you set up VSC on you own device at the beginning of the semester per instructions, this should be true.

All of the discussion below uses this fairly pointless Java program:

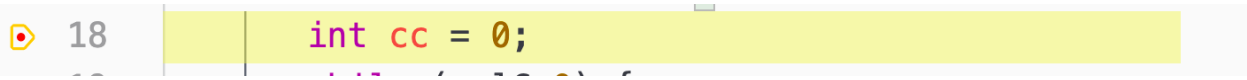
```
public class ABC {
   public int a(int value) {
       int sm = 0;
       for (int i=0; i<value; i++) {
           sm += i;
       }
       return b(sm);
   }
   public int b(int valB) {
       int j= valB/10;
       while (j>0) {
           valB -= j;
           j--;
       }
       return c(valB);
   }
   public int c(int valC) {
       int cc = 0;
       while (valC>0) {
           valC = valC/2;
           cc++;
       }
       return cc;
   }
   public static void main(String[] args) {
       int jj = (new ABC()).a(4);
       System.out.println(jj);
   }
}
```

First, your VSC window should have the following features:

 along the left edge	 probably in the upper right	 above the Main method

Go to line 18 “int cc=0”. Position you mouse to the left of the line number, you should see a fuzzy red circle. Click here. The circle should now be solid and not fuzzy. That red circle indicates that you have set a ‘breakpoint’ at line 18. (To remove a breakpoint, just tap on the solid red dot.) You can set as many breakpoints as you want in a program. One is often sufficient, but I have been known to have 5 or 10 in very large programs

Now tap on the word Debug above Main, or either of the right pointing arrows with a bug icons. When you do so you should see something like this



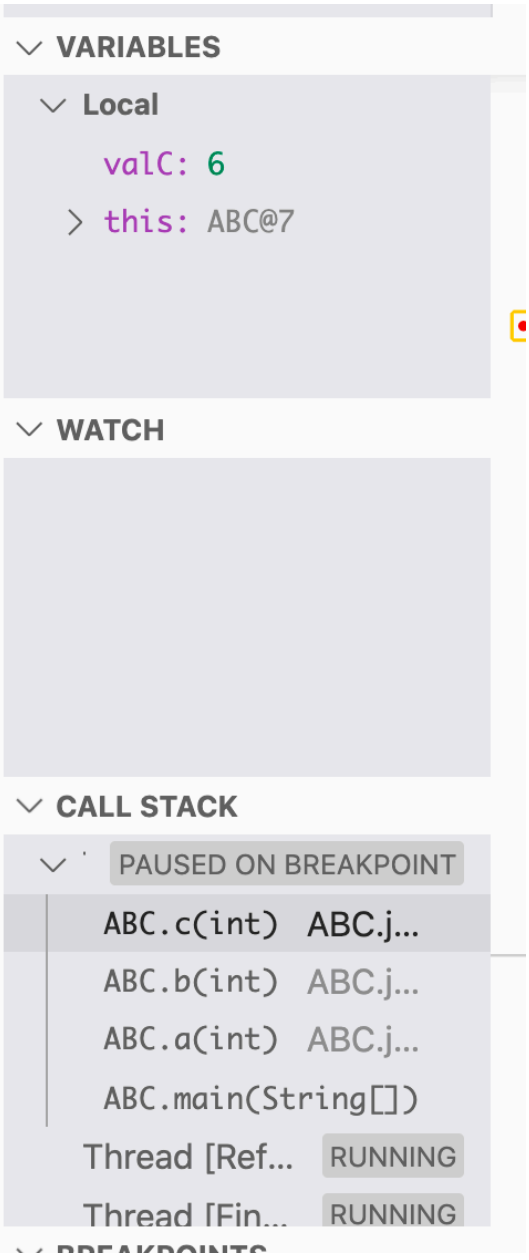
The program has run normally until it got to line 18 for the first time, at which time it paused (as is indicated by the line highlighted in yellow). This pause is the effect of setting a breakpoint. When the program’s execution is paused at a breakpoint, somewhere on your screen you should see

Hover over each of these buttons: you should see (from left to right):
continue
step over,
step into
step out
restart
stop
hot code replace

These have the following behaviors:
continue: restarts the program execution. The program will go along as normal until hitting another breakpoint (if there is one).
step over: advance the program one step within the current method
step into. Advance the program one step, going into method is a method is called in the step
step out: Advance the program one step, going out of the current method.
Restart: restart the program from the beginning
Stop: kill the program.
hot code replace (I never use this and will not discuss)

Try each of these buttons, especially step over, step out and step into. Be sure you understand what each do. For instance, from the image of above of line 18 in yellow, if I hit step over once, then line 19 would become yellow. Again, line 20, line 21, then back to line 19, etc.
So, using the step over you can follow the exact steps your program takes within a method. Step in and step out allow you to follow the program into methods or out.

But wait, there is more. Look at the left side of the screen. You should see something like the image at left. (This is after I restarted the program so it is stopped again at line 18). The "Variables" section (at the top shows the values of the local variables for the current method. Here we see than valC is 6. After hitting "step over" 4 times I can now see that valC is 3 and cc is 1 (not shown). The variables section thus allows you to watch values of the variables of the method I step through it.



Note that the variables section also has "> this: ABC@7". This is the current instance of the class in which the method is running. If the instance had variables (it does not in this program) you could tap on the ">" to see the values of the instance variables in addition to the variables local to the method. By watching the values in the variables section, you can go well beyond print statements.

The watch section is beyond the scope of this document, but can be very useful.

Finally, the "call stack". In class I have described this as the "method stack". This shows exactly how you got to the point in the program where you are. So in the image, it says, I am currently in method c, which I got to from method b which I got to from method a. So, if I hit the step out button, the call stack would change to show that I am currently in method b and the highlighted line would be line 15