

# Factorial Calculator and Debug Mode

- Slides:  
<http://www.cs.unc.edu/~dewan/comp114/current/Lectures/Eclipse.pdf>
- Import the JavaTeaching project, loading the ObjectEditor and removing invalid projects
  - <http://cs.unc.edu/~dewan/comp401/current/Lectures/egit-import.pdf>.

# Factorial Calculator and Debug Mode

- You will need “Factorials”, “FactorialSpreadSheetDriver”, “ALoopingFactorialSpreadsheets”, “ARecursiveFactorialSpreadsheets” when using debug mode. These classes are in package “lectures.java\_basics\_overview”.
- Work in pairs
- Follow the quiz questions, modify the code accordingly, and test what happened.
- Test the performances of the two different implementations (recursion and loop).
- Go over slides 74-94 to get familiar with debug mode.

# Commands in Debug Mode

- Step into (F5): A method is about to be invoked, and you want to debug into the code of that method, so the next step is to go into that method and continue debugging step-by-step.
- Step over (F6): A method is about to be invoked, but you're not interested in debugging this particular invocation, so you want the debugger to execute that method completely as one entire step.
- Step return(F7): You're done debugging this method step-by-step, and you just want the debugger to run the entire method until it returns as one entire step.
- *Cited from: <http://stackoverflow.com/questions/3580715/eclipse-debug-step-into-step-over>*

# Notes

- Prof. Dewan changed the code a little bit. So when you run the program in debug mode, the process will not be identically the same as the process showed in the slides. The following slides demonstrate the real process when you run the code.
- Modification: In slides, the calculation of factorial is carried out when the method “getFactorial()” is invoked. In the code, the calculation is carried when “setNumber()” is invoked. Both of the two methods are used in classes “ARecursiveFactorialSpreadsheet” and “ALoopingFactorialSpreadsheet”

# Step into setNumber()

The screenshot shows an IDE with the following components:

- Code Editor:** Displays the source code for `ARecursiveFactorialSpreadsheet`. The `setNumber` method is highlighted in cyan. The code is as follows:

```
package lectures.java_basics_overview;

import lectures.recursion.Factorial;

public class ARecursiveFactorialSpreadsheet implements FactorialSprea
    int number;
    long factorial;
    public int getNumber() {
        return number;
    }
    public void setNumber(int newVal) {
        number = newVal;
        factorial = Factorials.recursingFactorial(number);
    }
    public long getFactorial() {
        return factorial;
    }
}
```
- Variables Window:** Shows the current state of variables. The variable `newVal` is set to `12`.

Name	Value
this	ARecursiveFactorialSpreads...
newVal	12
- Console Window:** Shows the output of the application. The text is:

```
FactorialSpreadsheetDriver [Java Application] C:\Program File:
Is FactorialSpreadsheet:true
Is ALoopingFactorialSpreadsheet:false
Is ARecursiveFactorialSpreadsheet:true
1000000 computations of factorial(15) took
Please enter a non-negative number if you w
12
```

# Step into recursingFactorial()

The screenshot shows an IDE with the following components:

- Code Editor:** Displays the source code for `ARecursiveFactorialSpreadsheet`. The line `factorial = Factorials.recursingFactorial(number);` is highlighted in cyan, indicating the current step-in point.
- Variables Window:** Shows the current state of variables:

Name	Value
this	ARecursiveFactorialSpreadsheet...
newVal	12
- Console Window:** Shows the output of the application:

```
FactorialSpreadsheetDriver [Java Application] C:\Program Files:
Is FactorialSpreadsheet:true
Is ALoopingFactorialSpreadsheet:false
Is ARecursiveFactorialSpreadsheet:true
1000000 computations of factorial(15) took
Please enter a non-negative number if you w
12
|
```

# Step into recursive call again

The screenshot shows the Eclipse IDE with the following components:

- Editor:** Displays the code for `Factorials.java`. The line `return n*recursingFactorial(n-1);` is highlighted in cyan, indicating the current execution point.
- Variables Window:** Shows a table with the following data:

Name	Value
n	12
- Console Window:** Shows the output of the Java application:

```
FactorialSpreadsheetDriver [Java Application] C:\Program File:
Is FactorialSpreadsheet:true
Is ALooopingFactorialSpreadsheet:false
Is ARecursiveFactorialSpreadsheet:true
1000000 computations of factorial(15) took
Please enter a non-negative number if you w
12
```