

# CSC 7003 : Basics of Software Engineering

**J Paul Gibson, D311**

paul.gibson@telecom-sudparis.eu

<http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC7003/>

## Debugging

[.../~gibson/Teaching/CSC7003/L13-Debugging.pdf](http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC7003/L13-Debugging.pdf)



"It's not an error. It's a debugging opportunity."

9/9

0800 Anton started  
 1000 stopped - action ✓ { 1.2700 9.020 897 025  
 13:00 100 MP - M1 ✓ 9.017 896 715 result  
 010 PRO ✓ 2.1304:696  
 020 2.1306:76W  
 Relays are in use failed speed test  
 in Italy  
 Relay: changed  
 1100 Started Cosmo Tape (Sine check)  
 1575 Started Multi-Header Test  
 1545 Relay to Panel F  
 (motion relay)  
 First actual case of bug being found.  
 1600 changed check.  
 1700 closed down.



www.phdcomics.com

# When Do You Debug?

When you want to find and fix an error

# What Type Of Error?

When a test fails or a run-time exception occurs

**Easiest -> Hardest**

Exception, Unit test, Validation test, Integration test

# Debugging - When no tools are available

*System.out.println()*

*System.err.println()*

*LOG.debug(...)*

**Question:** which of these *techniques* do you currently use?

# Using the Eclipse IDE debugger (for Java)

The screenshot displays the Eclipse IDE interface during a Java debug session. The top toolbar shows standard IDE actions like Run, Stop, and Step Over. The Package Explorer on the left shows the project structure, with the current thread 'Thread [main] (Suspended (breakpoint at line 120 in Validation\_Segment1D))' selected. The Variables view on the right shows the state of local variables: 'args' is a String array of length 15; 'segment1' and 'segment2' are Segment1D objects with IDs 16 and 18 respectively; 'NUMBER\_OF\_RANDOM\_CASES' is 10; 'rng' is a Random object with seed 10; and 'random\_count' is 1. The code editor shows the source code for 'Validation\_Segment1D.java', with line 120 highlighted in green, indicating the current execution point. The console at the bottom shows the output of the program, including a message about the random number generator seed and the start of the random test cases.

```
workspace-Neon-Java - Debug - SegmentOverlap/src/tests/Validation_Segment1D.java - Eclipse
Debug [X] Package Explorer
Validation_Segment1D [Java Application]
  tests.Validation_Segment1D at localhost:63387
    Thread [main] (Suspended (breakpoint at line 120 in Validation_Segment1D))
      Validation_Segment1D.main(String[]) line: 120
Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/bin/java (Feb 7, 2017, 12:56:02 PM)

(4) Variables [X] Breakpoints
Name Value
  args String[] (id=15)
  segment1 Segment1D (id=16)
  segment2 Segment1D (id=18)
  NUMBER_OF_RANDOM_CASES 10
  rng Random (id=10)
  random_count 1

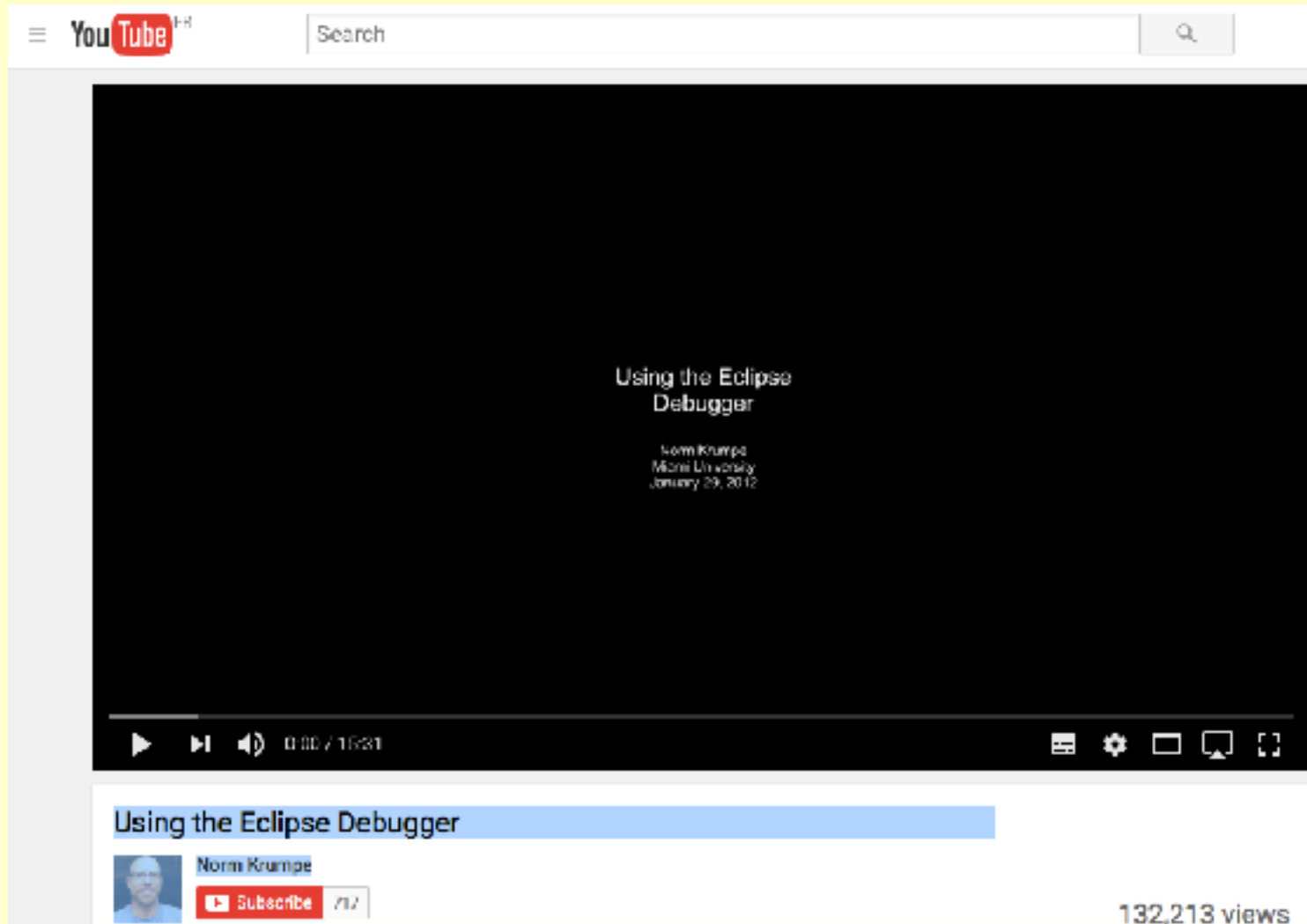
Validation_Segment1D.java [X]
111 Random rng = new Random(System.currentTimeMillis());
112
113
114 System.out.println("\n *** The following random test cases need to be validated by the tester *** \n");
115
116
117 for (int random_count = 1; random_count <= NUMBER_OF_RANDOM_CASES; random_count++){
118     segment1 = new Segment1D(rng);
119     segment2 = new Segment1D(rng);
120     System.out.println(" **** random test " + random_count + " ****");
121     System.out.println("segment: " + segment1);
122     System.out.println("invariant is " + segment1.invariant());
123     System.out.println("segment2: " + segment2);
124     System.out.println("invariant is " + segment2.invariant());
125     System.out.println("They overlap is " + segment1.overlaps(segment2));
126 }
127 }
128 }
129

Console [X] Tasks
Validation_Segment1D [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/bin/java (Feb 7, 2017, 12:56:02 PM)

The seed used for the random number generator in the test is 0.
You can override this value by passing an integer value as a main argument parameter, if you so wish.

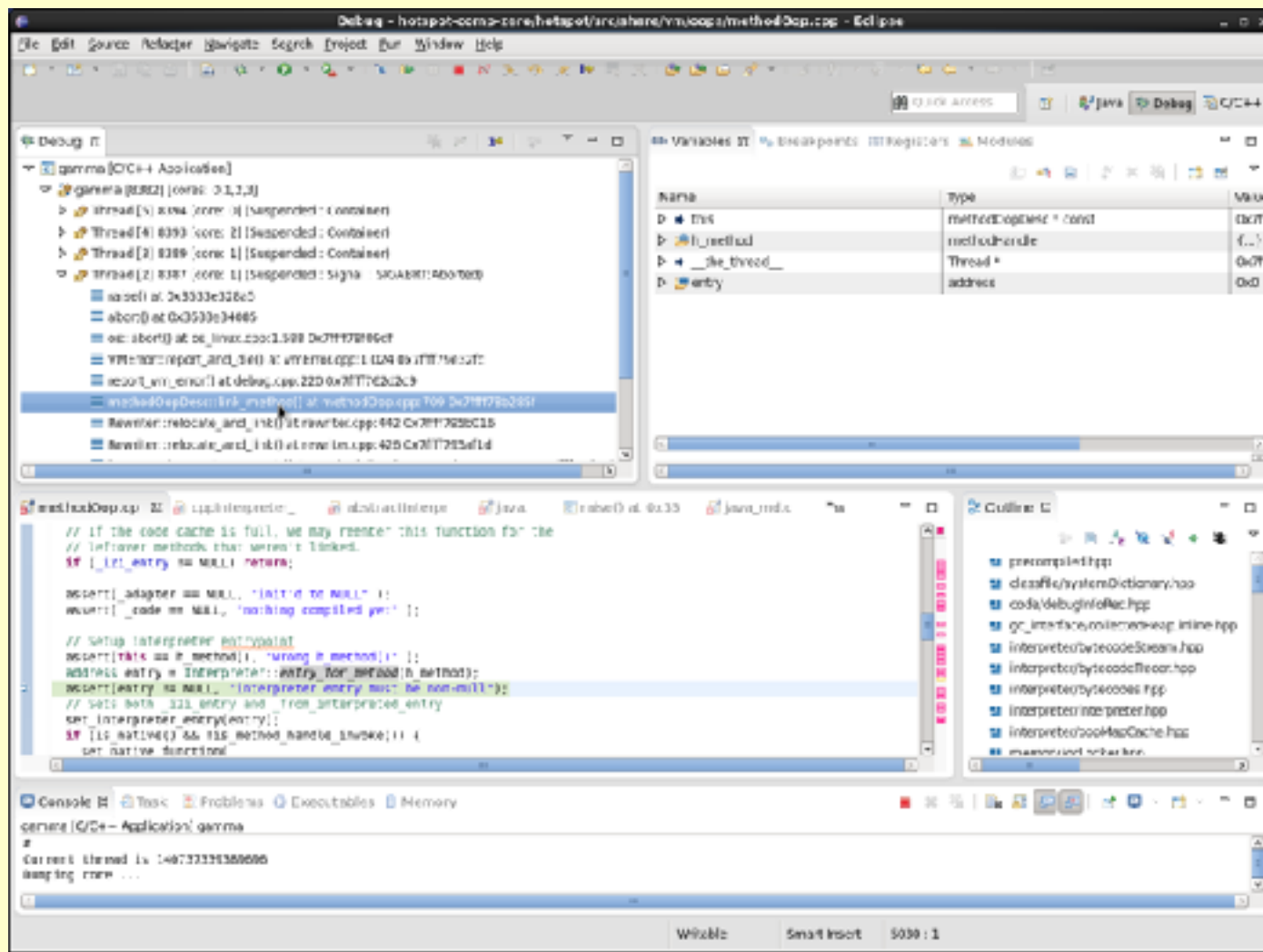
**** The following random test cases need to be validated by the tester ****
```

# On-line video tutorial



<https://www.youtube.com/watch?v=9gAjIQc4bPU>

# Eclipse also supports debugging the languages (like C++)



All reasonable IDEs provide debugging functionality, eg NetBeans:

**NetBeans IDE Features** [Download](#)

» Overview

Now in NetBeans 8.1

» Base IDE

- Project Management
- Databases
- Versioning
- Team Collaboration

Java

- Editing and Refactoring
- Build Tools
- Debugger and Profiler**
- Testing and Code Analysis

» Java on the Server

- Java EE
- Web Services
- Deployment and Monitoring

Java on the Client

- JavaFX
- Swing
- Java ME and Embedded

HTML5 Web Development

» PHP

- Editing and Refactoring
- Frameworks and Tools
- Testing and Code Analysis

### Debugger and Profiler

Click image for full screen preview

To identify and solve problems in your applications, such as deadlocks and memory leaks, NetBeans IDE provides a feature rich debugger and profiler.

#### Debugger

The NetBeans Debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots and monitor execution as it occurs. You can also attach the debugger to an already running process.

The IDE includes a visual debugger to let you take GUI snapshots and visually explore the GUI of JavaFX and Swing applications. It lets you view component properties, the hierarchy of components in the container, and locate the source code of components. You can use the visual debugger to easily add listeners to GUI actions without requiring you to search through the source code.

#### Profiler

The NetBeans Profiler provides expert assistance for performance issue identification, analysis and memory usage.

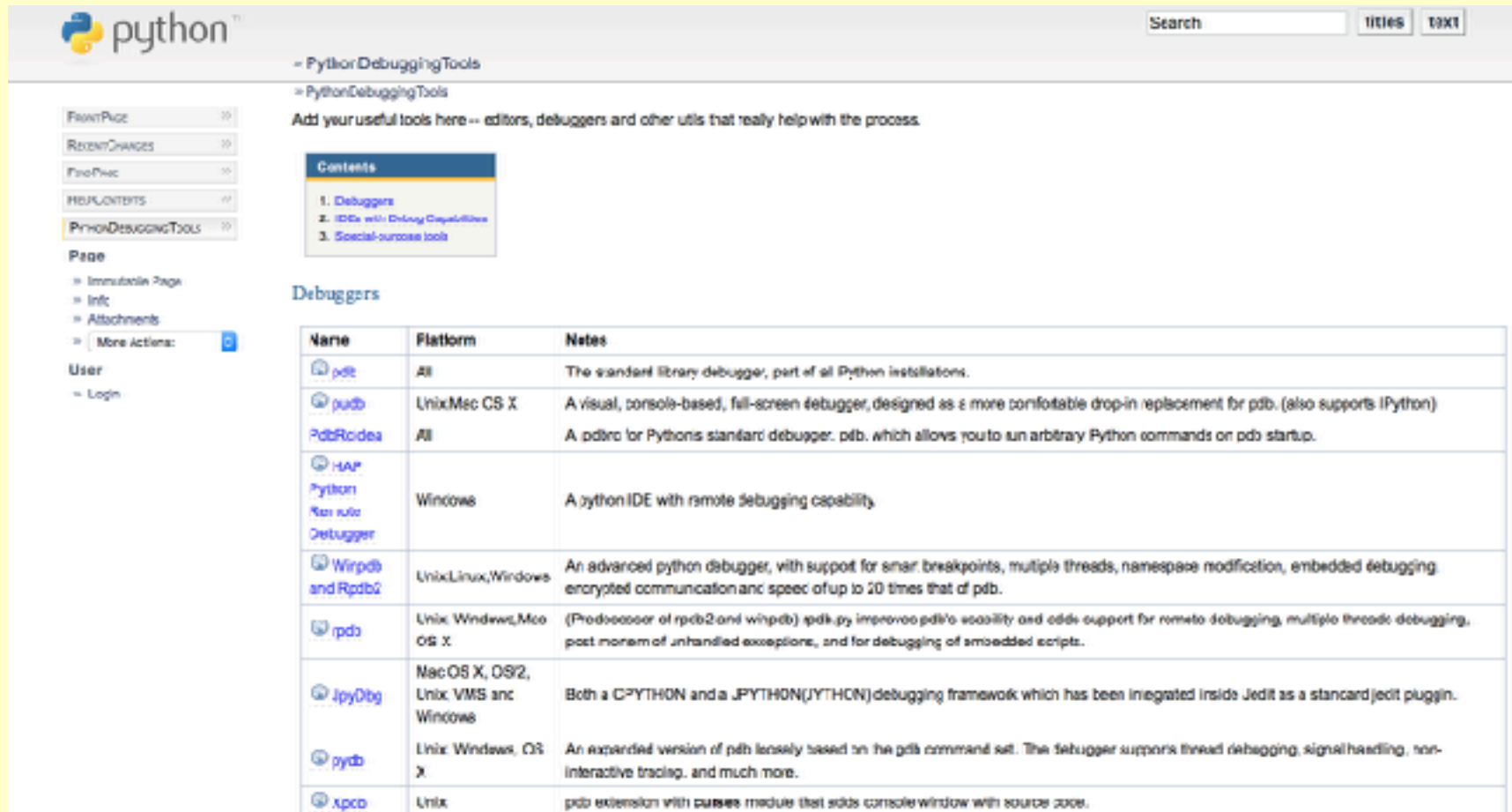
[Debugging Multi-threaded Applications in NetBeans IDE](#)

[Using the Visual Debugger in NetBeans IDE](#)

<https://netbeans.org/features/java/debugger.html>



# All reasonable programming languages provide a selection of debugging tools, e.g. Python



The screenshot shows the PythonDebuggingTools wiki page. The page title is '- PythonDebuggingTools'. It features a search bar, a navigation menu on the left, and a main content area with a 'Contents' table of contents and a 'Debuggers' table.

**Contents**

1. Debuggers
2. IDEs with Debug Capabilities
3. Special-purpose tools

**Debuggers**

Name	Platform	Notes
<a href="#">pdb</a>	All	The standard library debugger, part of all Python installations.
<a href="#">pudb</a>	Unix/Mac OS X	A visual, console-based, full-screen debugger, designed as a more comfortable drop-in replacement for pdb. (also supports IPython)
<a href="#">PdbRcidea</a>	All	A pdbrc for Python's standard debugger, pdb, which allows you to run arbitrary Python commands on pdb startup.
<a href="#">HAP Python Remote Debugger</a>	Windows	A python IDE with remote debugging capability.
<a href="#">Winpdb and Rpdb2</a>	Unix/Linux/Windows	An advanced python debugger, with support for smart breakpoints, multiple threads, namespace modification, embedded debugging, encrypted communication and speed of up to 20 times that of pdb.
<a href="#">rpdb</a>	Unix, Windows, Mac OS X	(Predecessor of rpdb2 and winpdb) rpdb.py improves pdb's scalability and adds support for remote debugging, multiple threads debugging, post-mortem of unhandled exceptions, and for debugging of embedded scripts.
<a href="#">JpyDbg</a>	Mac OS X, OS/2, Unix, VMS and Windows	Both a CPYTHON and a JPYTHON(JYTHON) debugging framework which has been integrated inside Jedit as a standard jedit plugin.
<a href="#">pydb</a>	Linux, Windows, OS X	An expanded version of pdb loosely based on the pdb command set. The debugger supports thread debugging, signal handling, non-interactive tracing, and much more.
<a href="#">xpcdb</a>	Unix	pdb extension with curses module that adds console window with source code.

<https://wiki.python.org/moin/PythonDebuggingTools>

You can even debug at the command line, eg using gdb for C:



The screenshot shows a blog post from 'THE GEEK STUFF'. The logo features a large red 'G' with a magnifying glass over it. Below the logo, it says 'Linux | DB | Open Source | Web'. The navigation bar includes 'Home', 'Free eBook', 'Start Here', 'Contact', and 'About'. The main heading is 'How to Debug C Program using gdb in 6 Simple Steps' by SATHIYAMOORTHY on MARCH 15, 2010. There are social media buttons for GitHub, Facebook (115 likes), and Twitter. The text of the post begins with 'Earlier we discussed the basics of how to write and compile a C program with [C Hello World Program](#). In this article, let us discuss how to debug a c program using gdb debugger in 6 simple steps. Write a sample C program with errors for debugging purpose To learn C program debugging, let us create the following C program that calculates and prints the factorial of a number. However this C program contains some errors in it for our debugging purpose.' To the right of the text is a cartoon illustration of a fish with a thought bubble containing the letters 'C' and 'c'.

<http://www.thegeekstuff.com/2010/03/debug-c-program-using-gdb>