

# Remote z/OS Debugging

## SHARE Orlando, February 2008

Oliver Fenton

Java Technology Center, IBM Hursley Labs,  
Winchester, UK

Session: 8355



# Trademarks



IBM, AIX, CICS, DB2, IMS, z/OS, OS/390, S/390, System/390, VisualAge, WebSphere Application Server, WebSphere Studio, z/VM - are trademarks or registered trademarks of the IBM Corporation

Sun, Sun Microsystems, JavaSoft, Java, JavaBeans, JDK, Java 2 Micro Edition, J2ME, Java 2 Standard Edition, J2SE, Java 2 Enterprise Edition, J2EE - are trademarks or registered trademarks of Sun Microsystems Inc.

# Agenda



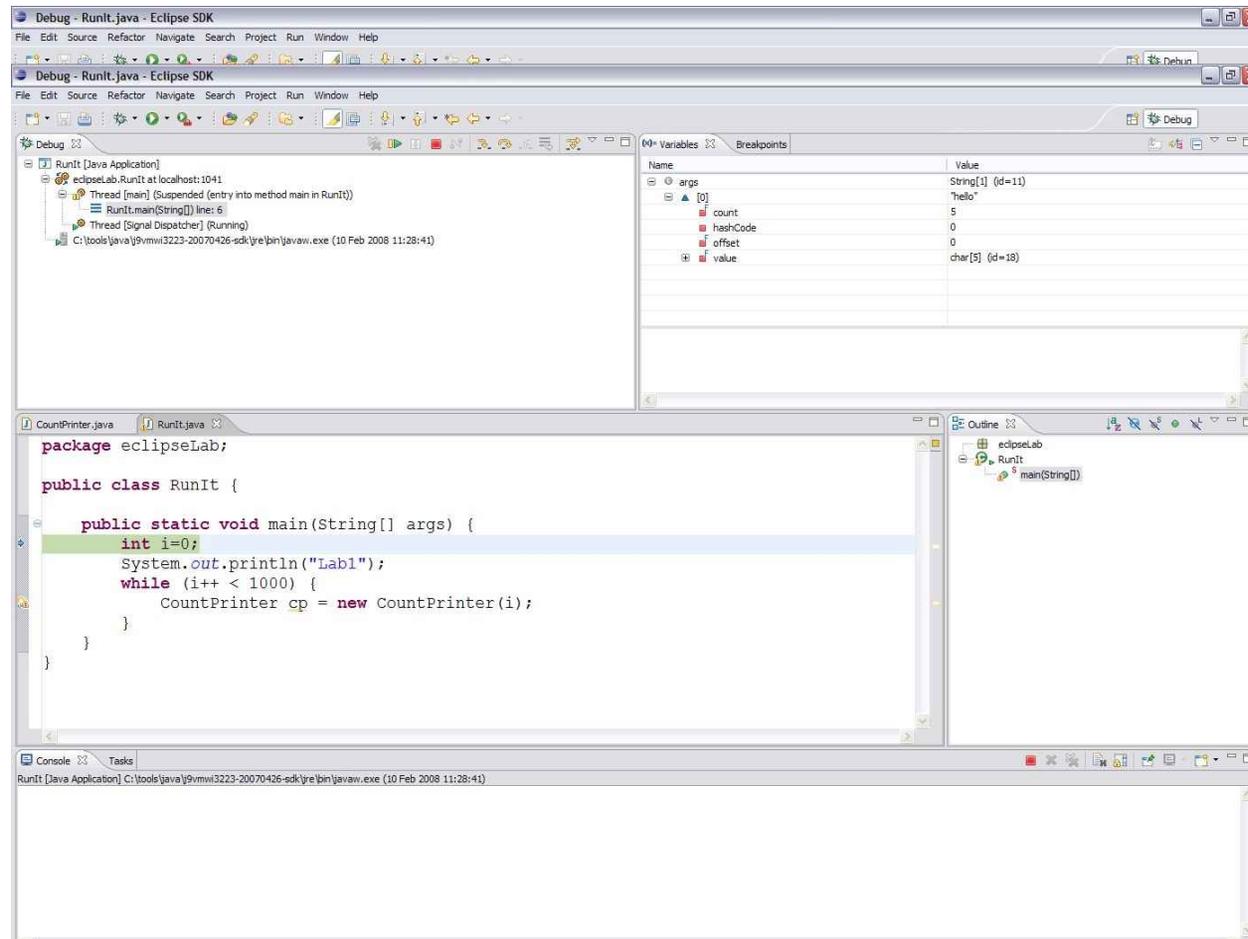
## ❖ The Debug Perspective

- The Debug Perspective
- Exercise 1
- Remote Debugging
- Exercise 2

# Debug Perspective

- ❖ The debug perspective gives you access to functionality which enables the user to run code interactively by:
  - 1. Stepping through the execution line by line
  - 2. Setting break points at places in the code where the execution will suspend
  - 3. Examining program attributes, such as variables, locks, memory, registers, ... at any given (break) point
  - 4. Modifying variables during program execution
  - 5. Modifying code during program execution
  
- ❖ This enables the user to:
  - 1. Find code errors
  - 2. Unit test

# Debug Perspective



The screenshot displays the Eclipse IDE in the Debug perspective. The main editor shows the source code of `RunIt.java` with the line `int i=0;` highlighted. The Variables view on the right shows the state of the program:

Name	Value
args	String[1] (d=11) "Hello"
count	5
hashCode	0
offset	0
value	char[5] (d=18)

The Outline view on the right shows the package structure: `eclipseLab` containing `RunIt` and `main(String[])`. The Console view at the bottom is empty.

## Exercise 1

---

- ❖ Run RunIt.java from Monday's Eclipse Lab in debug mode
- ❖ Use the debug entities and functions:
  - breakpoints
  - step through, step into, step out, continue to next break point
- ❖ Display runtime entities
  - variables
  - thread stacks
- ❖ Advanced debugging techniques
  - Modify variables in flight
  - Change code in flight

# Agenda



## ❖ The Debug Perspective

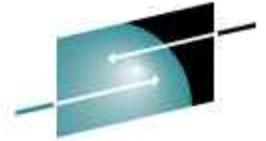
- Debug functions and hot code replacement
- The debug perspective layout
- Exercise 1
- **Remote Debugging**
- Exercise 2

# Remote Debugging



- ❖ In this exercise an application running on a z/OS box will be debugged from Eclipse running on a windows box.
- ❖ The application is the RunIt application
- ❖ The exercise consists of 2 parts:
  - starting the application on a remote machine (z/OS)
  - debugging the application from Eclipse

# The Application



**SHARE**  
Technology • Connections • Results

```
package eclipseLab;

public class RunIt {

    public static void
    main(String[] args) {
        int i=0;
        System.out.println("Lab1");
        while (i++ < 1000) {
            CountPrinter cp = new
            CountPrinter(i);
        }
    }
}
```

```
package eclipseLab;

public class CountPrinter {

    public CountPrinter(int num){
        System.out.println("Count = "
        + num);
    }
}
```

## On The Remote Machine

- ❖ The files and directory structure on z/OS

```

-rw-r----- 1 SHARA01 SHARUSER 128 Feb 11 04:33 CountPrinter.java
-rw-r----- 1 SHARA01 SHARUSER 201 Feb 11 04:33 RunIt.java
drwxr-xr-x  3 SHARA01 SHARUSER 8192 Feb 11 04:39 ..
-rw-r--r--  1 SHARA01 SHARUSER 680 Feb 11 04:39 RunIt.class
-rw-r--r--  1 SHARA01 SHARUSER 674 Feb 11 04:39 CountPrinter.class
drwxr-xr-x  2 SHARA01 SHARUSER 8192 Feb 11 04:39 .
MVS1:SHARA01:/sharelab/shara01/eclipseLab/eclipseLab: > cd ..
MVS1:SHARA01:/sharelab/shara01/eclipseLab: > ls -lrta
total 56
drwxr-xr-x  7 SHARA01 SHARUSER 8192 Feb 11 04:32 ..
-rwxr-xr-x  1 SHARA01 SHARUSER 204 Feb 11 04:39 runMeRemote
drwxr-xr-x  3 SHARA01 SHARUSER 8192 Feb 11 04:39 .
drwxr-xr-x  2 SHARA01 SHARUSER 8192 Feb 11 04:39 eclipseLab
MVS1:SHARA01:/sharelab/shara01/eclipseLab: >

```

- ❖ run runMeRemote which starts RemoteRunIt suspended waiting on debug instructions on port 8001

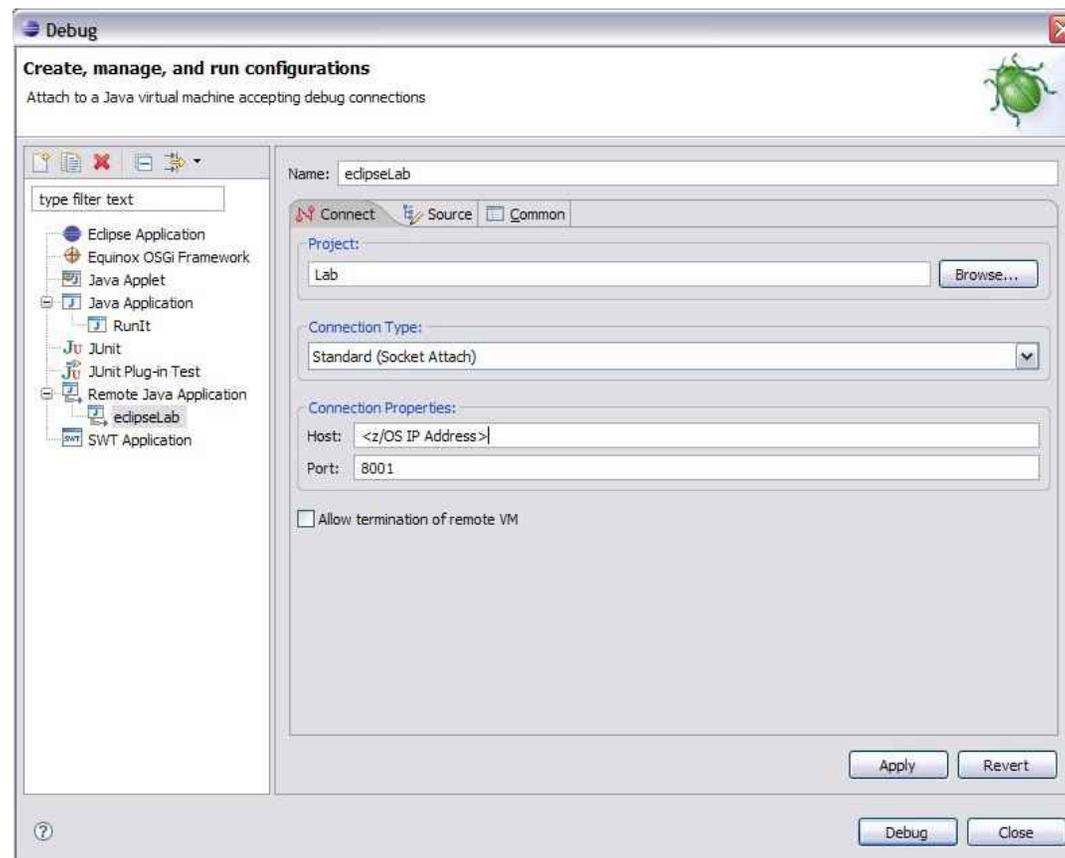
```

MVS1:SHARA01:/sharelab/shara01/eclipseLab: > cat runMeRemote
/usr/lpp/java15/J5.0/bin/javac -g eclipseLab/RunIt.java
/usr/lpp/java15/J5.0/bin/java -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdp:t
ransport=dt_socket,address=8001,suspend=y,server=y eclipseLab/RunIt
MVS1:SHARA01:/sharelab/shara01/eclipseLab: > runMeRemote
Listening for transport dt_socket at address: 8001

```

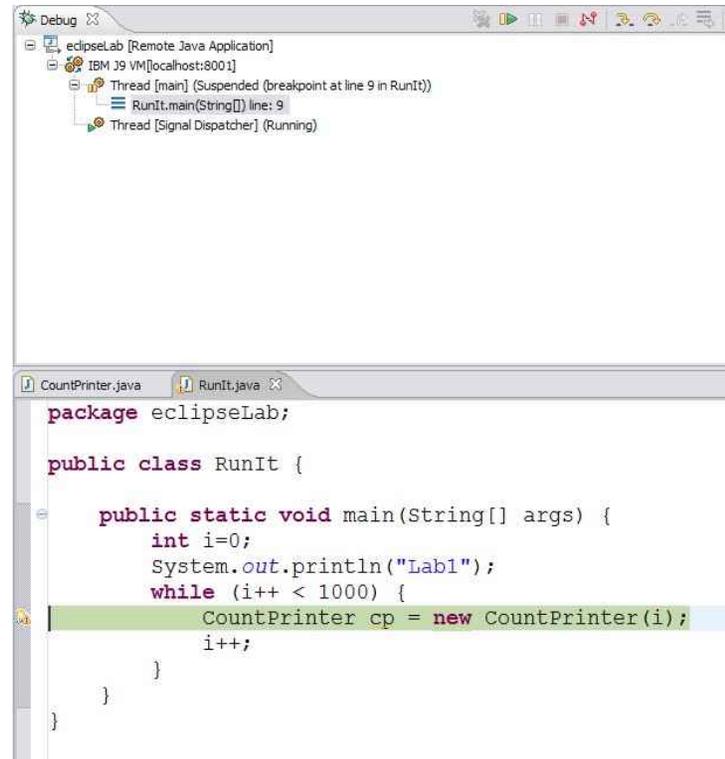
# In Eclipse

- ❖ Select **Run > Debug**, enter the following settings and select **Debug**



# In Eclipse

- ❖ A breakpoint was set at the start of the main method and execution is suspended here.

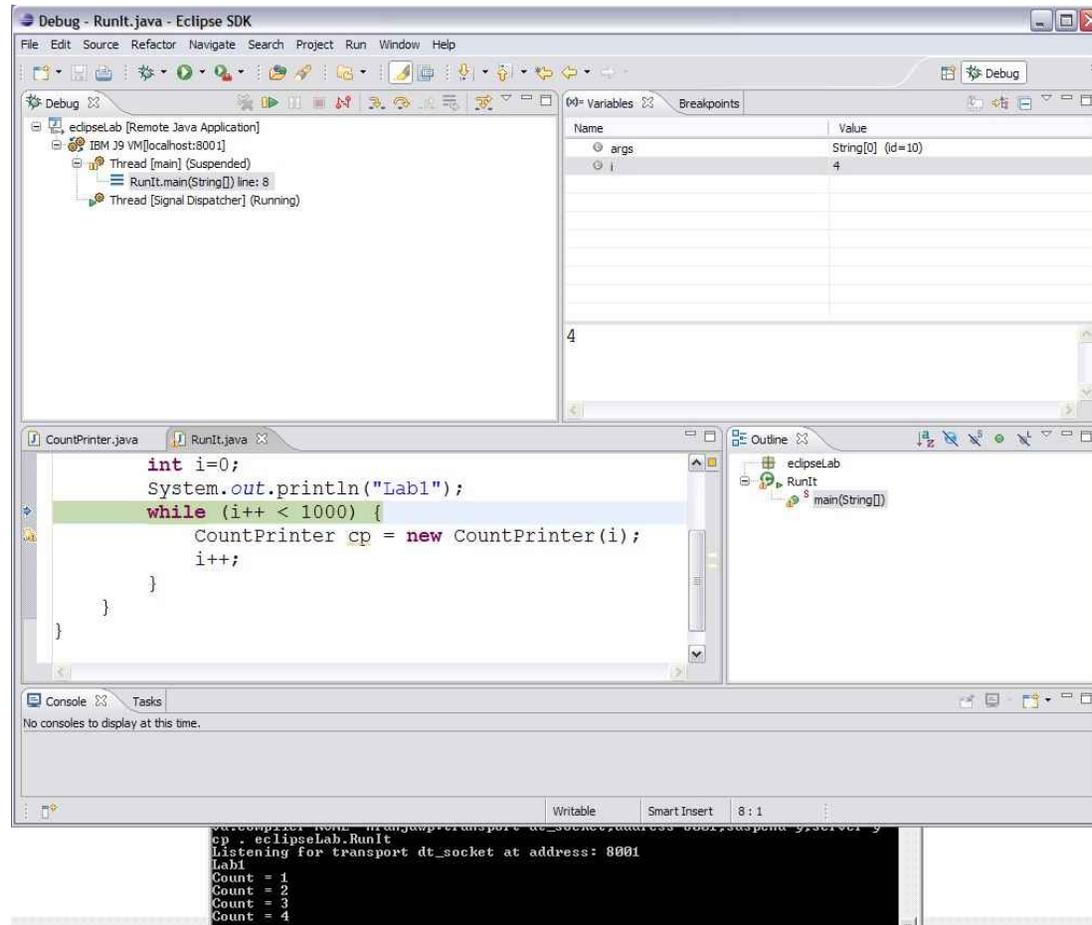


```
package eclipseLab;

public class RunIt {

    public static void main(String[] args) {
        int i=0;
        System.out.println("Lab1");
        while (i++ < 1000) {
            CountPrinter cp = new CountPrinter(i);
            i++;
        }
    }
}
```

# An In Flight Snapshot



The screenshot displays the Eclipse IDE interface during a debug session. The main editor shows the source code for `CountPrinter.java` with a `while` loop that prints "Lab1" and increments a counter from 0 to 1000. The `RunIt.java` file is also open, showing a `main` method that creates a `CountPrinter` object. The `Debug` window shows the current state of the application, including the `RunIt.main(String[])` method and the `Thread [Signal Dispatcher]` running. The `Variables` window shows the `args` array and the `i` variable. The `Console` window shows the output of the program, including the `Lab1` output and the `Count` values.

```
int i=0;
System.out.println("Lab1");
while (i++ < 1000) {
    CountPrinter cp = new CountPrinter(i);
    i++;
}
}
```

```
cp . eclipseLab.RunIt
Listening for transport dt_socket at address: 8001
Lab1
Count = 1
Count = 2
Count = 3
Count = 4
```

# Accessing the Remote z/OS Machine



Host name: mvs1.centers.ihost.com

Username: SHARA02 – SHARA30

p/w: firstpw

Putty config: mvs1\_2994

Suggest you create a session on mvs1 using putty which is available from your desktop