



8375

Java Application Development using Eclipse Part I

Jezz Kelway – kelwayj@uk.ibm.com
Java Technology Centre, z/OS Service
IBM Hursley Park Labs, United Kingdom

Abstract



- Learn how to use the powerful features of Eclipse to aid software development.
 - 'Ease-of-use' features such as code generation, automatic syntactic checking and code completion reduce the time needed to write software
 - Debugging options including, step through execution, breakpoints, display of variable values, memory, registers, ... make finding those algorithmic bugs easier
 - Remote debugging facilities will leave you open mouthed.

Objectives



- Presentation 1
 - An overview of Eclipse and its Architecture
- Presentation 2
 - The Debugger
 - Plugin's

Agenda



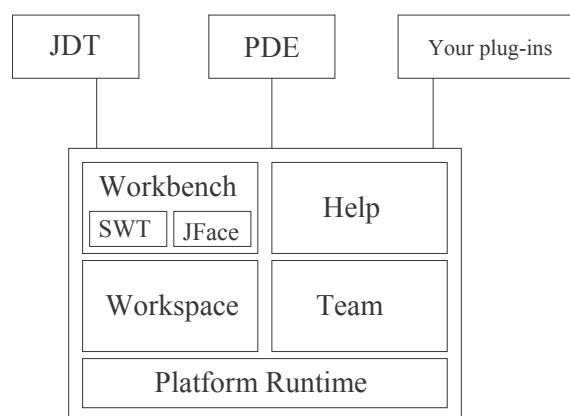
- Introduction
 - Eclipse Overview and History
 - Eclipse Architecture
 - Perspectives
- The Resource Perspective
 - Editing and syntax checking
 - Running Programs

Eclipse Overview and History



- Eclipse is an extensible software development framework based on a plug-in architecture.
- Eclipse is open source, released under a common public license (CPL).
- Originally developed by IBM, version 1.0 being released in November 2001, Eclipse is now under the stewardship of an independent consortium including: IBM, SAP, Sybase, Fujitsu, HP, BEA...

Eclipse Architecture



Eclipse Architecture explained



Platform runtime

Primary job is to discover the available plug-ins. Each plug-in has an XML manifest file which lists the connections the plug-in requires.

The workspace

Manages the user's resources which are organised into projects. The workspace maintains a history of changes to resources.

The workbench

Eclipse's Graphical User Interface, menus, toolbars and perspectives. Perspectives provide a GUI for specific areas of functionality such as debugging, plug-in development,...

The Standard Widget Toolkit (SWT) are graphics toolkits which map directly to the OS native graphics. JFace is a toolkit built on SWT.

Team Support

Version control. Eclipse provides a client for Concurrent Version System (CVS)

Help

An extensible help component

Java Development Toolkit (JDT)

Toolkit for the writing and debugging of Java programs. C Development Toolkit is the equivalent plug-in for C development.

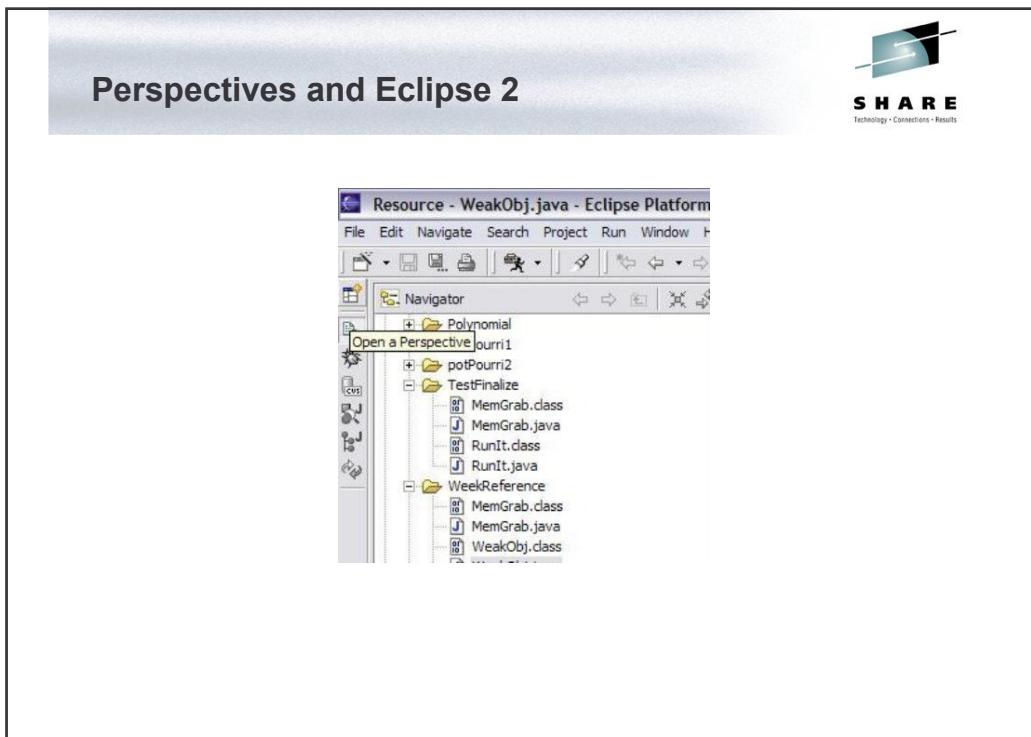
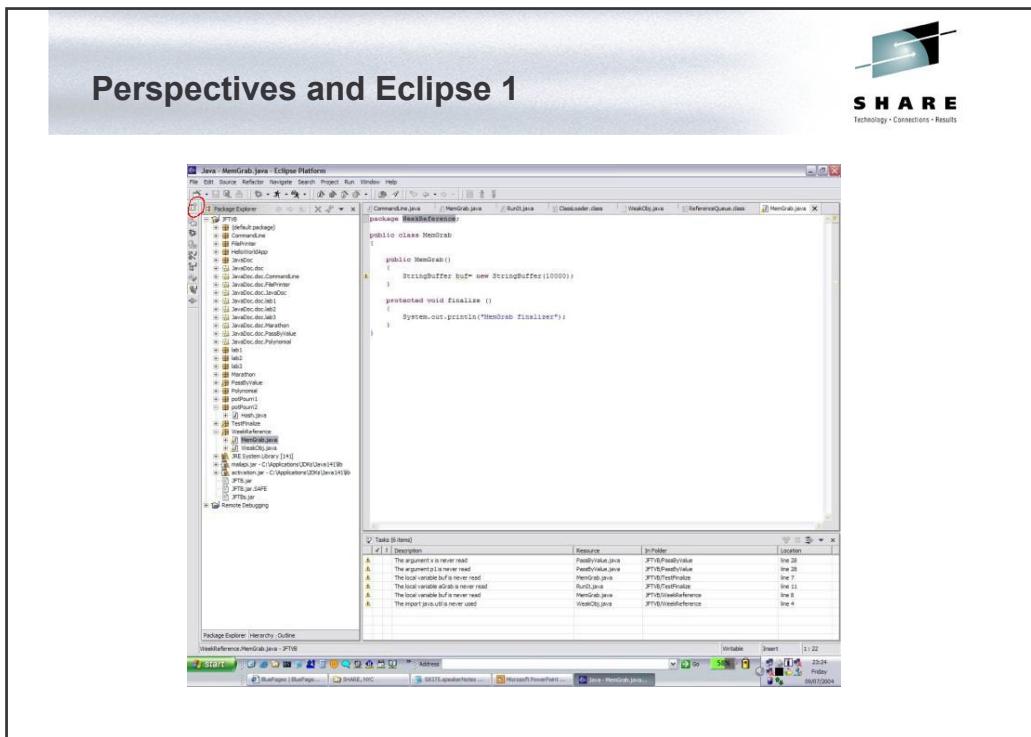
Plug-in Development Environment (PDE)

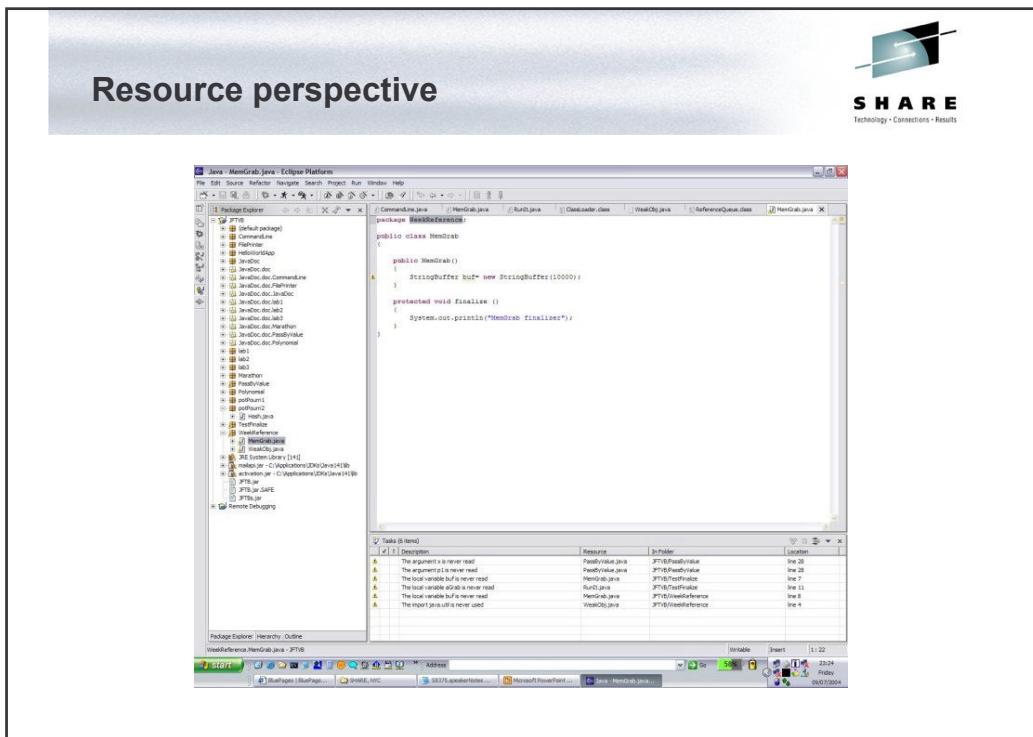
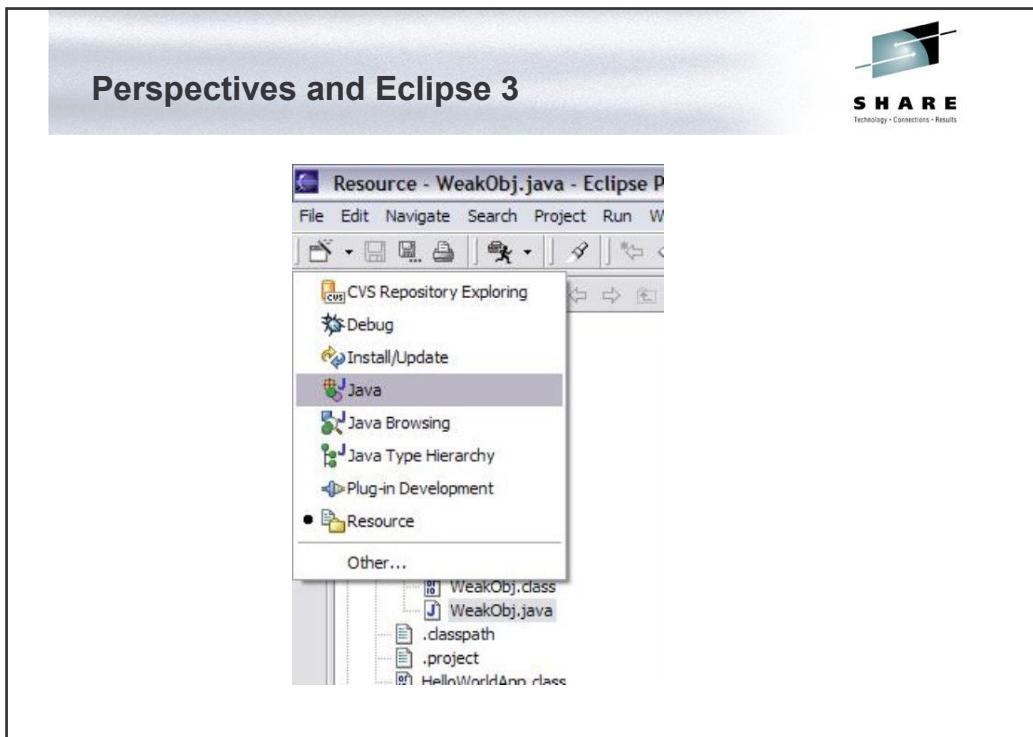
Developing plug-ins for extending ECLIPSE.

Perspectives



- An Eclipse perspective is a pre-selected set of views arranged in a predetermined way.
- These presentations are structured around describing functionality of the following perspectives:
 - Resource
 - Debug

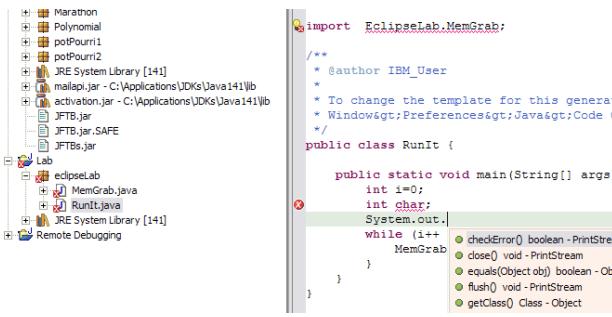




Lab 1 (1)

SHARE
Technology • Connections • Results

- Create a project, package and java classes: RunIt.java and MemGrab.java
- Syntax errors and typos are in the code to demonstrate:
 - Icons used to identify errors and warnings
 - Incremental compilation (on file save)



The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer view, displaying a Java project named 'Lab' with two source files: 'MemGrab.java' and 'RunIt.java'. The code editor on the right contains the following Java code:

```

import EclipseLab.MemGrab;

/**
 * @author IBM_User
 *
 * To change the template for this generator
 * Windows->Preferences->Java->Code Generation
 */
public class RunIt {

    public static void main(String[] args)
        int i=0;
        int char;
        System.out.[]
        while (i++
            MemGrab
        )
    }
}

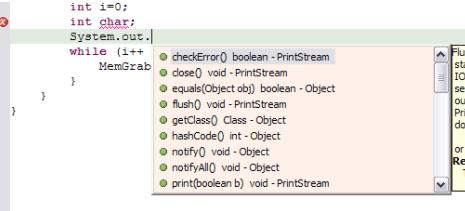
```

The code editor highlights several errors and warnings with red and orange markers. A code completion dropdown is open over the 'System.out.' call, showing suggestions like 'checkError()', 'close()', 'equals()', 'flush()', 'getClass()', 'hashCode()', 'notify()', 'notifyAll()', and 'print()'.

Lab 1 (2)

SHARE
Technology • Connections • Results

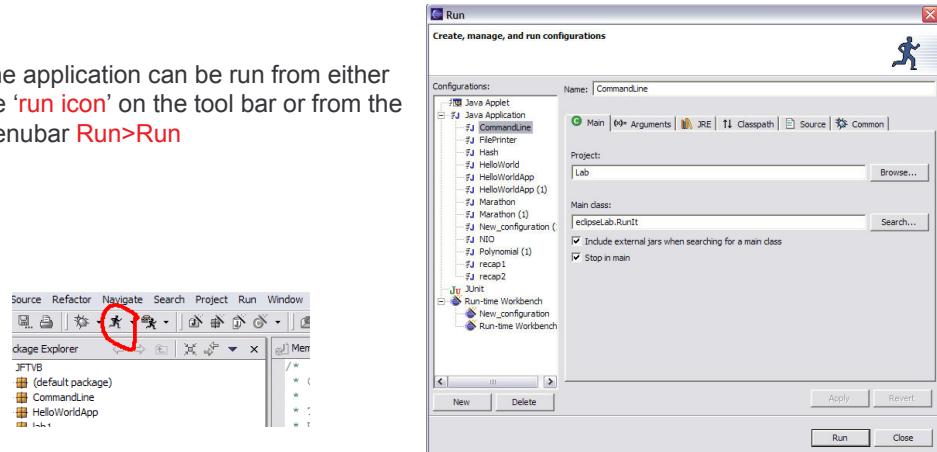
- code completion



The screenshot shows the Eclipse IDE interface with the code editor displaying the same Java code as in the previous screenshot. A code completion dropdown is open over the 'System.out.' call, showing the same list of methods: 'checkError()', 'close()', 'equals()', 'flush()', 'getClass()', 'hashCode()', 'notify()', 'notifyAll()', and 'print()'.

Lab 1 (3)

- The application can be run from either the 'run icon' on the tool bar or from the menubar Run>Run



Lab 1 (4)

- Passing arguments to the JVM, for example -verbosegc

