[_________________________]

# Eclipse project briefing materials.

The most up-to-date briefing materials on the Eclipse project are found on the eclipse.org website at

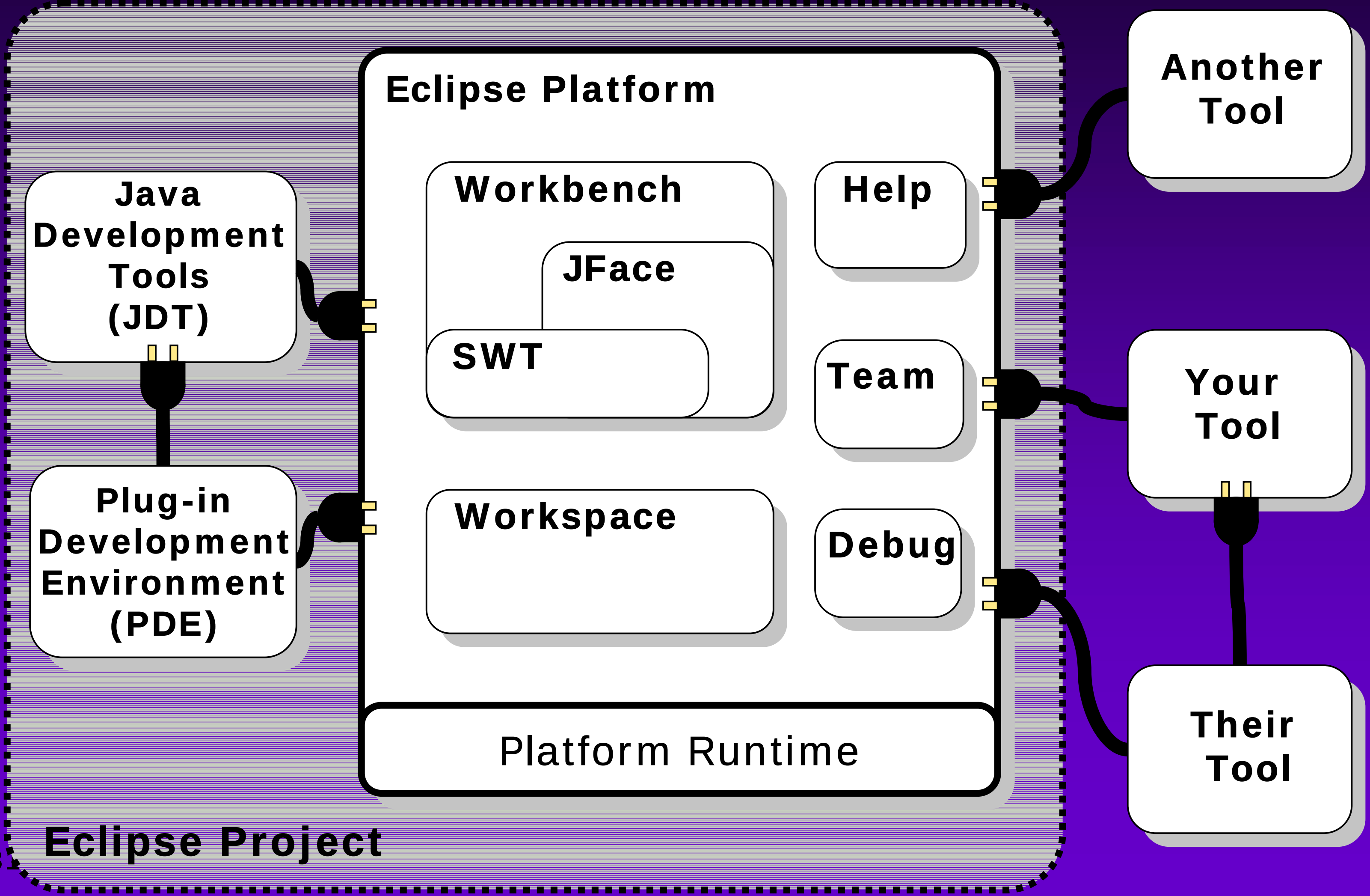
http://eclipse.org/eclipse/

## Eclipse Project

# Eclipse Project Aims

- Provide open platform for application development tools
  - Run on a wide range of operating systems
  - GUI and non-GUI
- Language-neutral
  - Permit unrestricted content types
  - HTML, Java, C, JSP, EJB, XML, GIF, …
- Facilitate seamless tool integration
  - At UI and deeper
  - Add new tools to existing installed products
- Attract community of tool developers
  - Including independent software vendors (ISVs)
  - Capitalize on popularity of Java for writing tools

# Eclipse Overview



## Eclipse Origins

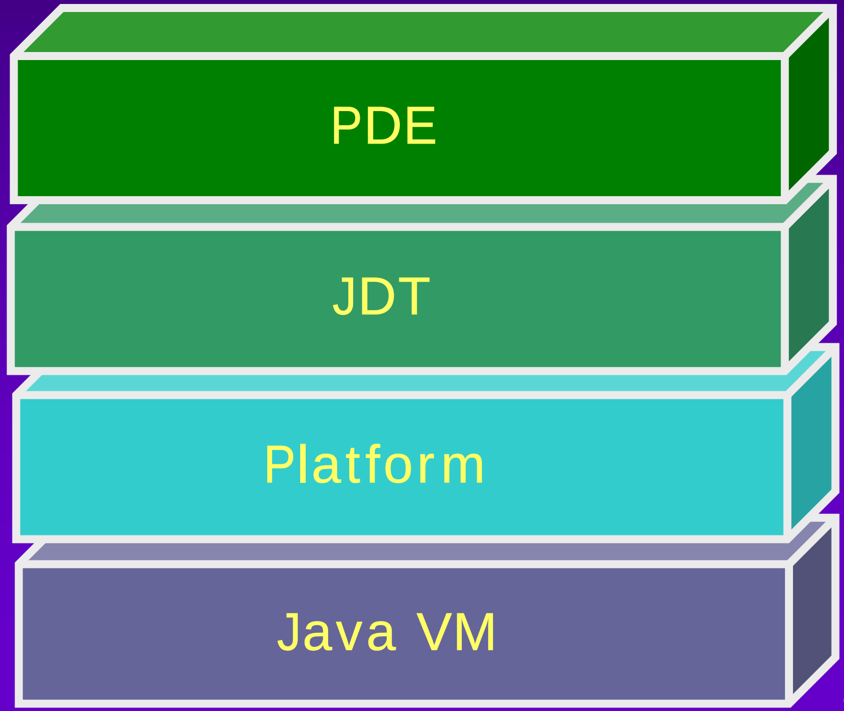
- Eclipse created by OTI and IBM teams responsible for IDE products
  - IBM VisualAge/Smalltalk (Smalltalk IDE)
  - IBM VisualAge/Java (Java IDE)
  - IBM VisualAge/Micro Edition (Java IDE)
- Initially staffed with 40 full-time developers
- Geographically dispersed development teams
  - OTI Ottawa, OTI Minneapolis, OTI Zurich, IBM Toronto, OTI Raleigh, IBM RTP, IBM St. Nazaire (France)
- Effort transitioned into open source project
  - IBM donated initial Eclipse code base
    - Platform, JDT, PDE

# Brief History of Eclipse

| | |
|---|---|
| **1999** | |
| April | - Work begins on Eclipse inside OTI/IBM |
| **2000** | |
| June | - Eclipse Tech Preview ships |
| **2001** | |
| March | - http://www.eclipsecorner.org/ opens |
| June | - Eclipse 0.9 ships |
| October | - Eclipse 1.0 ships |
| November | - IBM donates Eclipse source base<br>- eclipse.org board announced<br>- http://www.eclipse.org/ opens |
| **2002** | |
| June | - Eclipse 2.0 ships |
| September | - Eclipse 2.0.1 ships |
| November | - Eclipse 2.0.2 ships |
| **2003** | |
| March | - Eclipse 2.1 ships |

## What is Eclipse?

- Eclipse is a universal platform for integrating development tools
- Open, extensible architecture based on plug-ins


Plug-in development environment
PDE
Java development tools
JDT
Eclipse Platform
Platform
Standard Java2 Virtual Machine
Java VM

## Eclipse Plug-in Architecture

- **Plug-in -** smallest unit of Eclipse function
  - Big example: HTML editor
  - Small example: Action to create zip files

- **Extension point -** named entity for collecting "contributions"
  - Example: extension point for workbench preference UI

- **Extension -** a contribution
  - Example: specific HTML editor preferences

## Eclipse Plug-in Architecture

- Each plug-in
  - Contributes to 1 or more extension points
  - Optionally declares new extension points
  - Depends on a set of other plug-ins
  - Contains Java code libraries and other files
  - May export Java-based APIs for downstream plug-ins
  - Lives in its own plug-in subdirectory

- Details spelled out in the **plug-in manifest**
  - Manifest declares contributions
  - Code implements contributions and provides API
  - plugin.xml file in root of plug-in subdirectory

## Plug-in Manifest

plugin.xml

```
<plugin
    id = “com.example.tool"
    name = “Example Plug-in Tool"
    class = "com.example.tool.ToolPlugin">
  <requires>
    <import plugin = "org.eclipse.core.resources"/>
    <import plugin = "org.eclipse.ui"/>
  </requires>
  <runtime>
    <library name = “tool.jar"/>
  </runtime>
  <extension
    point = "org.eclipse.ui.preferencepages">
   <page id = "com.example.tool.preferences"
     icon = "icons/knob.gif"
     title = “Tool Knobs"
     class = "com.example.tool.ToolPreferenceWizard“/>
  </extension>
  <extension-point
    name = “Frob Providers“
    id = "com.example.tool.frobProvider"/>
</plugin>
```

Plug-in identification

Other plug-ins needed

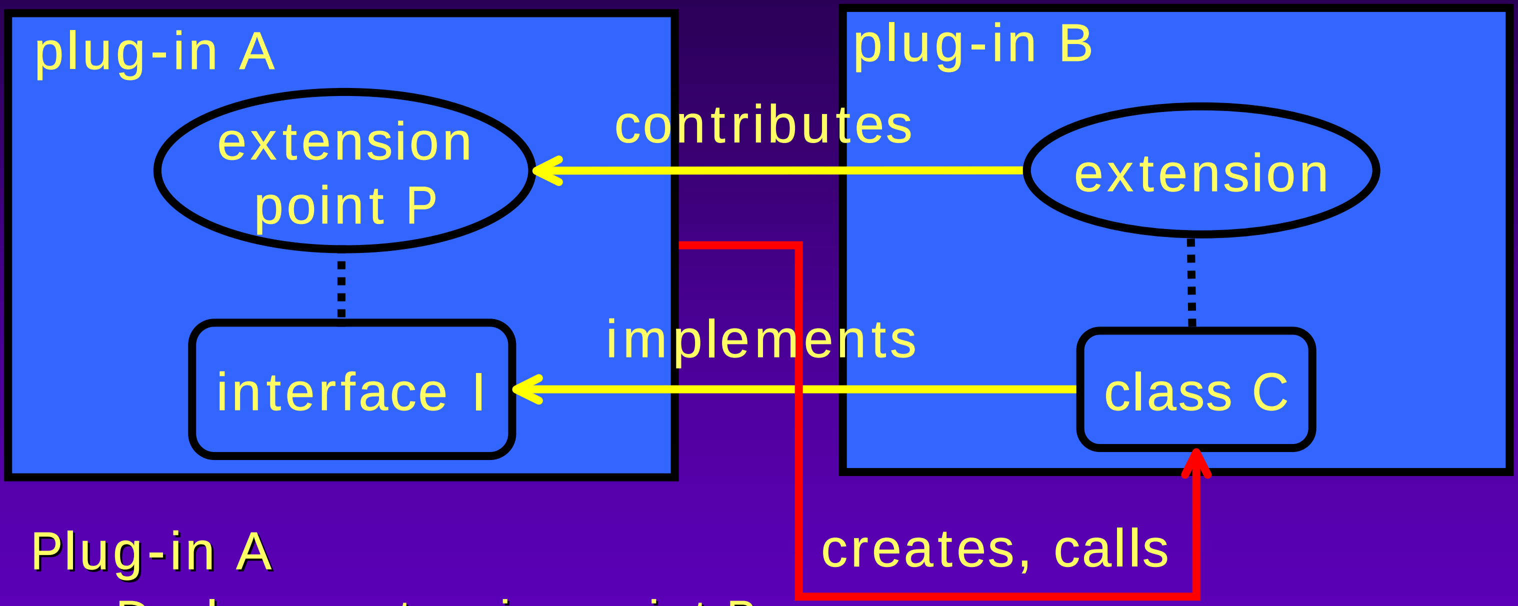
Location of plug-in’s code

Declare contribution this plug-in makes

Declare new extension point open to contributions from other plug-ins

## Eclipse Plug-in Architecture

- Typical arrangement



- Plug-in A
  - Declares extension point P
  - Declares interface I to go with P
- Plug-in B
  - Implements interface I with its own class C
  - Contributes class C to extension point P
- Plug-in A instantiates C and calls its I methods

## Eclipse Platform Architecture

- Eclipse Platform Runtime is micro-kernel
  - All functionality supplied by plug-ins
- Eclipse Platform Runtime handles start up
  - Discovers plug-ins installed on disk
  - Matches up extensions with extension points
  - Builds global plug-in registry
  - Caches registry on disk for next time

# Plug-in Activation

- Each plug-in gets its own Java class loader
  - Delegates to required plug-ins
  - Restricts class visibility to exported APIs
- Contributions processed without plug-in activation
  - Example: Menu constructed from manifest info for contributed items
- Plug-ins are activated only as needed
  - Example: Plug-in activated only when user selects its menu item
  - Scalable for large base of installed plug-ins
  - Helps avoid long start up times

## Plug-in Fragments

- **Plug-in fragments** holds some of plug-in's files
  - Separately installable
- Each fragment has separate subdirectory
  - Separate manifest file
- Logical plug-in = Base plug-in + fragments
- Plug-in fragments used for
  - Isolation of OS dependencies
  - Internalization – fragments hold translations

## Plug-in Install

- **Features** group plug-ins into installable chunks
  - Feature manifest file
- Plug-ins and features bear version identifiers
  - major . minor . service
  - Multiple versions may co-exist on disk
- Features downloadable from web site
  - Using Eclipse Platform update manager
  - Obtain and install new plug-ins
  - Obtain and install updates to existing plug-ins

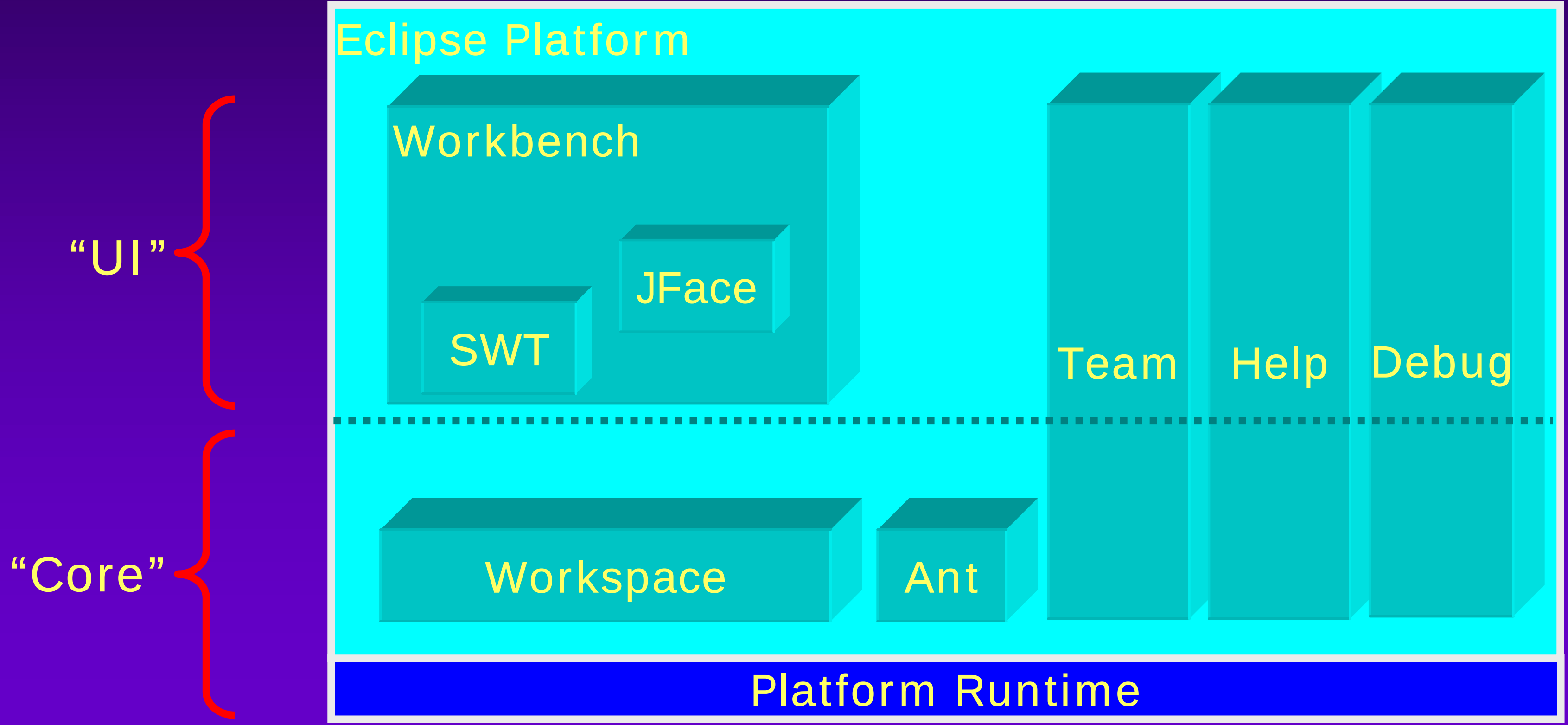
## Plug-in Architecture - Summary

- All functionality provided by plug-ins
  - Includes all aspects of Eclipse Platform itself
- Communication via extension points
  - Contributing does not require plug-in activation
- Packaged into separately installable features
  - Downloadable

**Eclipse has open, extensible architecture based on plug-ins**

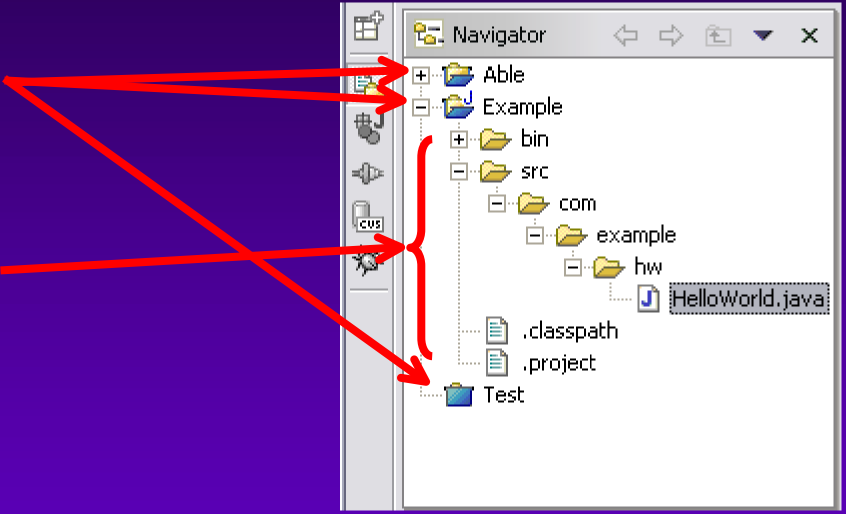
# Eclipse Platform

- Eclipse Platform is the common base
- Consists of several key components



## Workspace Component

- Tools operate on files in user's **workspace**
- Workspace holds 1 or more top-level **projects**
- Projects map to directories in file system
- Tree of **folders** and **files**
- {Files, Folders, Projects} termed **resources**



- Tools read, create, modify, and delete resources in workspace
- Plug-ins access via workspace and resource APIs

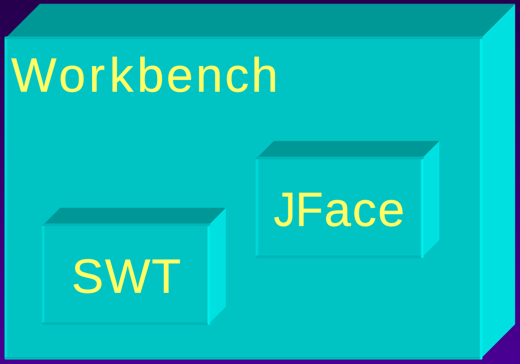
## Workspace and Resource API

- Allows fast navigation of workspace resource tree
- Resource change listener for monitoring activity
  - Resource deltas describe batches of changes
- Maintains limited history of changed/deleted files
- Several kinds of extensible resource metadata
  - Persistent resource properties
  - Session resource properties
  - Markers
  - Project natures
- Workspace session lifecycle
  - Workspace save, exit, restore
- Incremental project builders

## Incremental Project Builders

- Problem: coordinated analysis and transformation of thousands of files
  - Compiling all source code files in project
  - Checking for broken links in HTML files
- Scalable solution requires incremental reanalysis
- Incremental project builder API/framework
  - Builders are passed resource delta
  - Delta describes all changes since previous build
  - Basis for incremental tools
- Extensible – plug-ins define new types of builders
  - JDT defines Java builder
- Configurable – any number of builders per project

## Workbench Component



- SWT – generic low-level graphics and widget set
- JFace – UI frameworks for common UI tasks
- Workbench – UI personality of Eclipse Platform

## SWT

- SWT = Standard Widget Toolkit
- Generic graphics and GUI widget set
  - buttons, lists, text, menus, trees, styled text…

- Simple
- Small
- Fast
- OS-independent API
- Uses native widgets where available
- Emulates widgets where unavailable

## Why SWT?

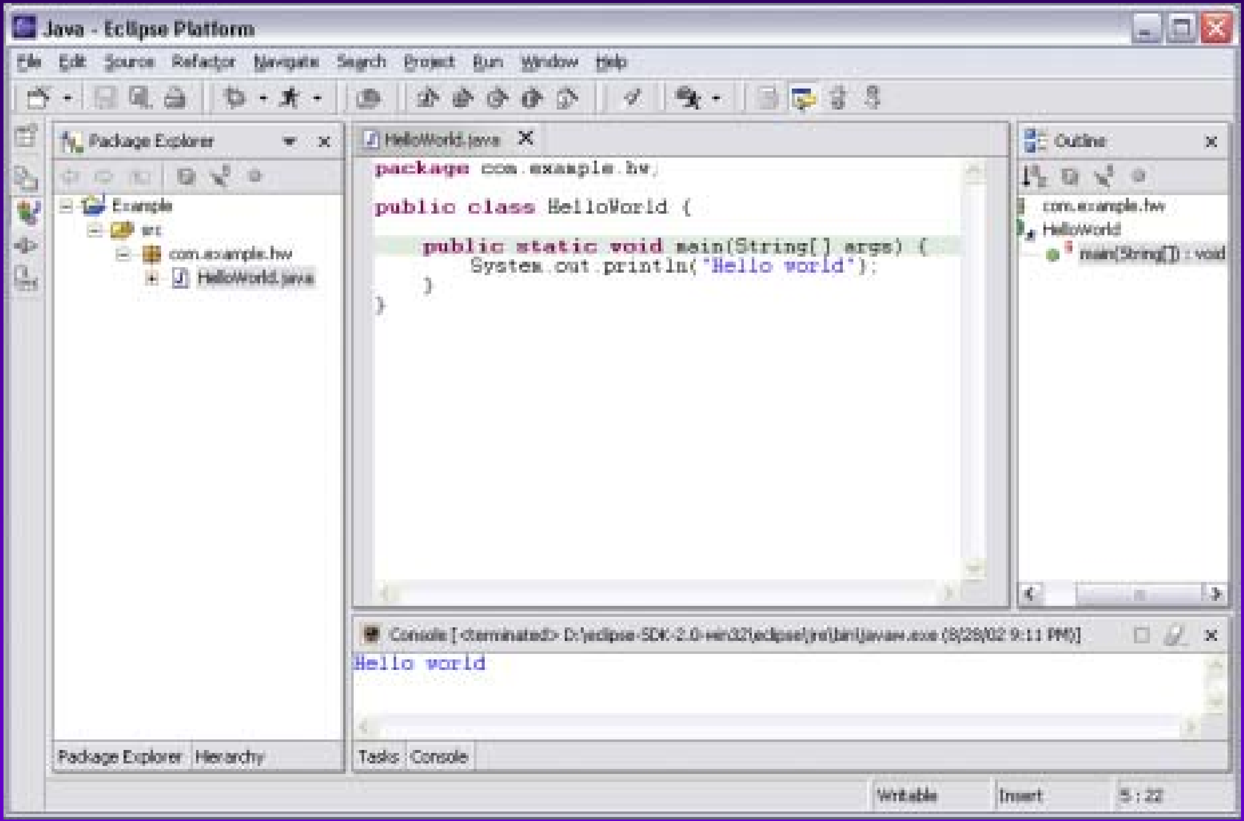
- Consensus: hard to produce professional looking shrink-wrapped products using Swing and AWT

- SWT provides
  - Tight integration with native window system
  - Authentic native look and feel
  - Good performance
  - Good portability
  - Good base for robust GUIs

- The proof of the pudding is in the eating…

## Why SWT?

- Eclipse Platform on Windows XP



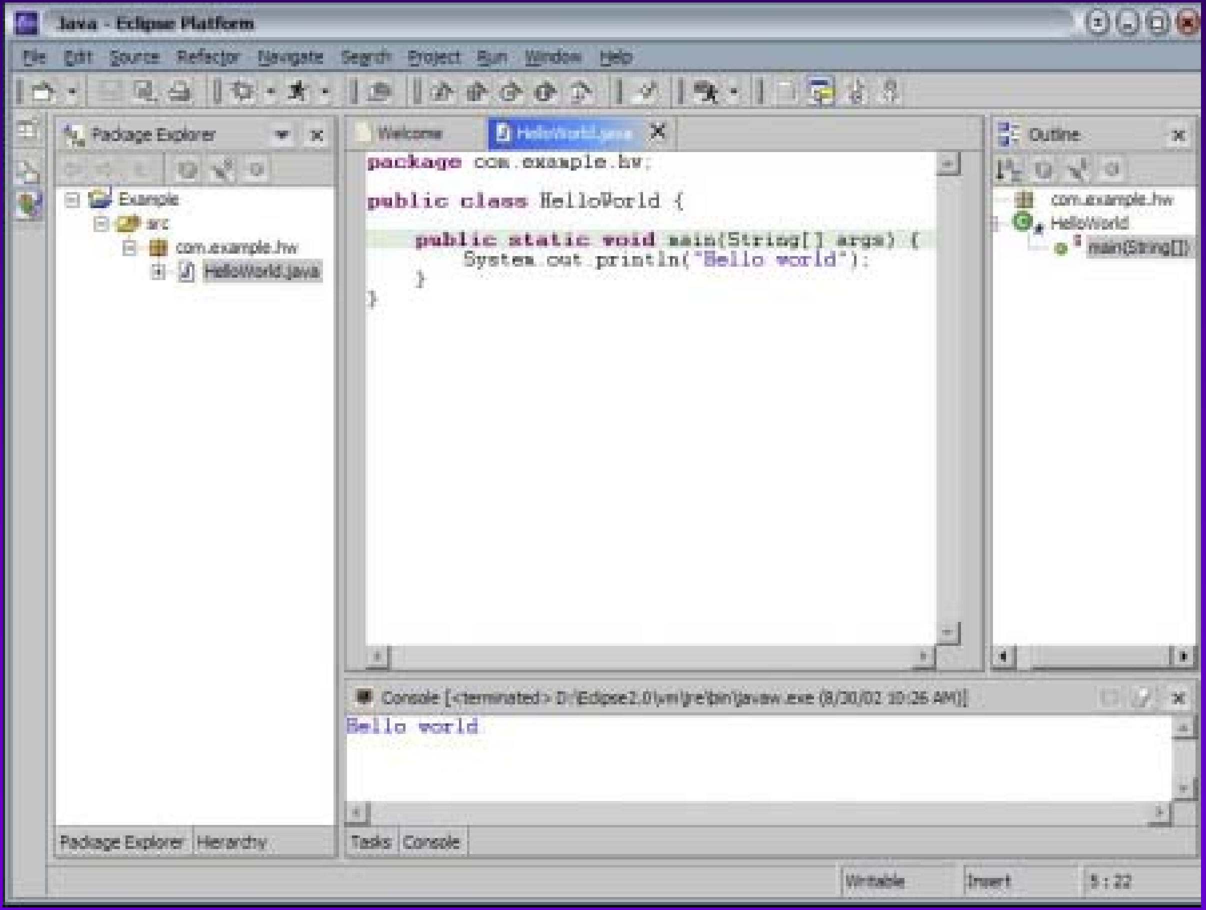
```
package com.example.hw;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Console [<terminated> D:\eclipse-SDK-2.0-win32\eclipse\jre\bin\javaw.exe (8/28/02 9:11 PM)]

```
Hello world
```

## Why SWT?

- Eclipse Platform on Windows XP (skinned)



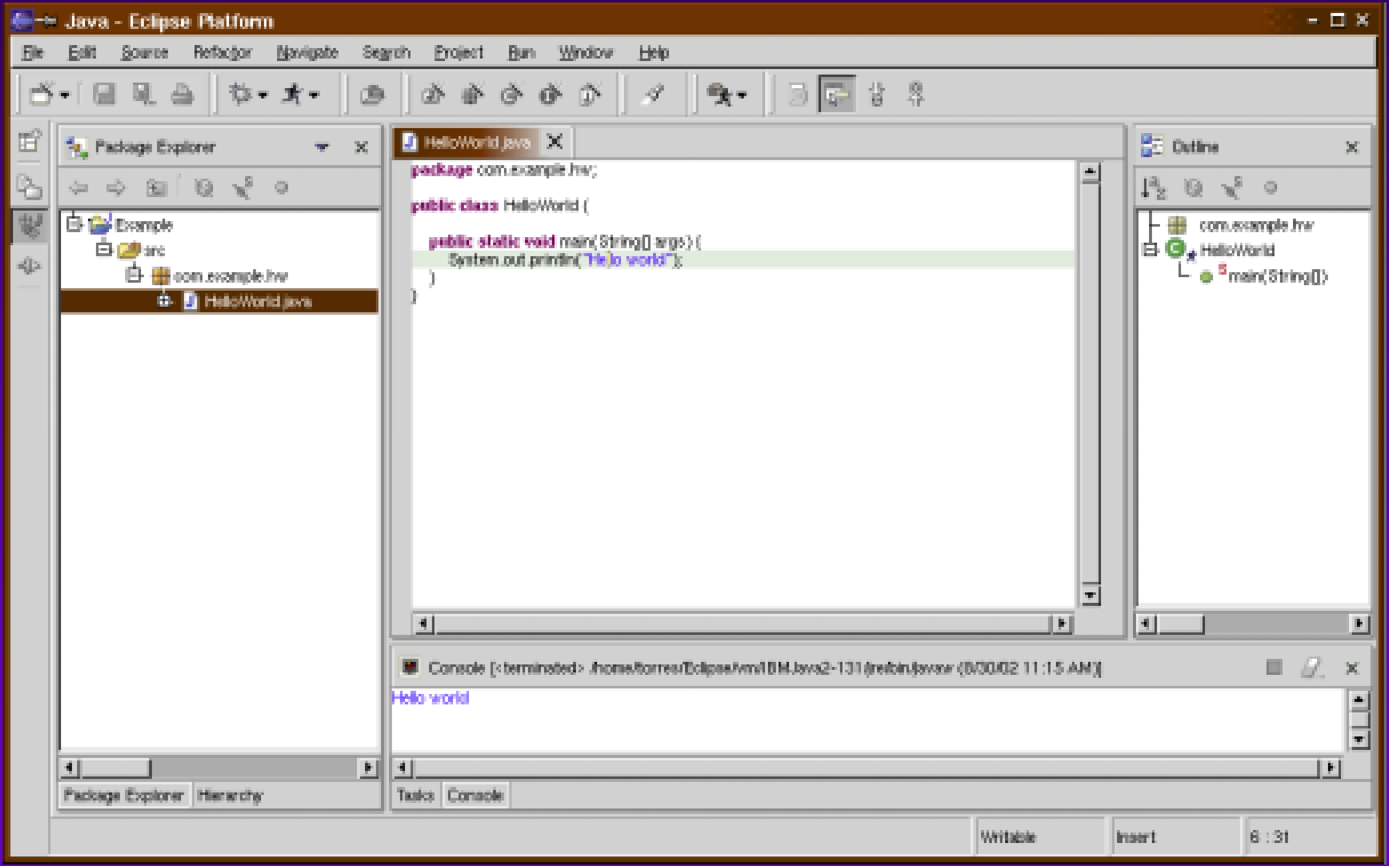
```
package com.example.hw;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Console [<terminated> D:\Eclipse2.0\jvm\jre\bin\javaw.exe (8/30/02 10:26 AM)]

```
Hello world
```

## Why SWT?

- Eclipse Platform on Linux - GTK 2.0



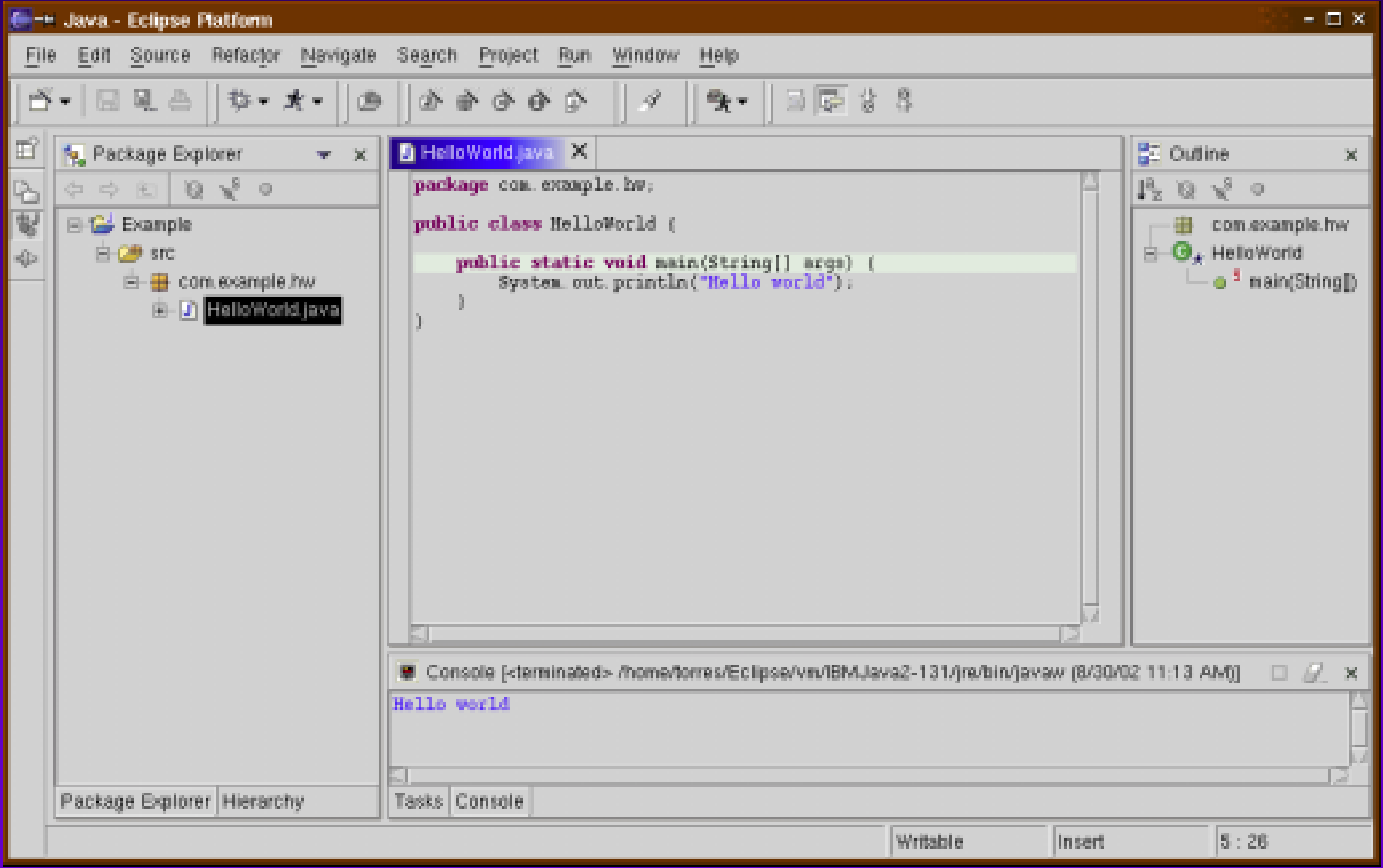
```
package com.example.hw;

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello world!");
  }
}
```

Console [<terminated> /home/torres/Eclipse/vm/IBMJava2-131/jre/bin/java (8/30/02 11:15 AM)]

```
Hello world
```

## Why SWT?

- Eclipse Platform on Linux - Motif



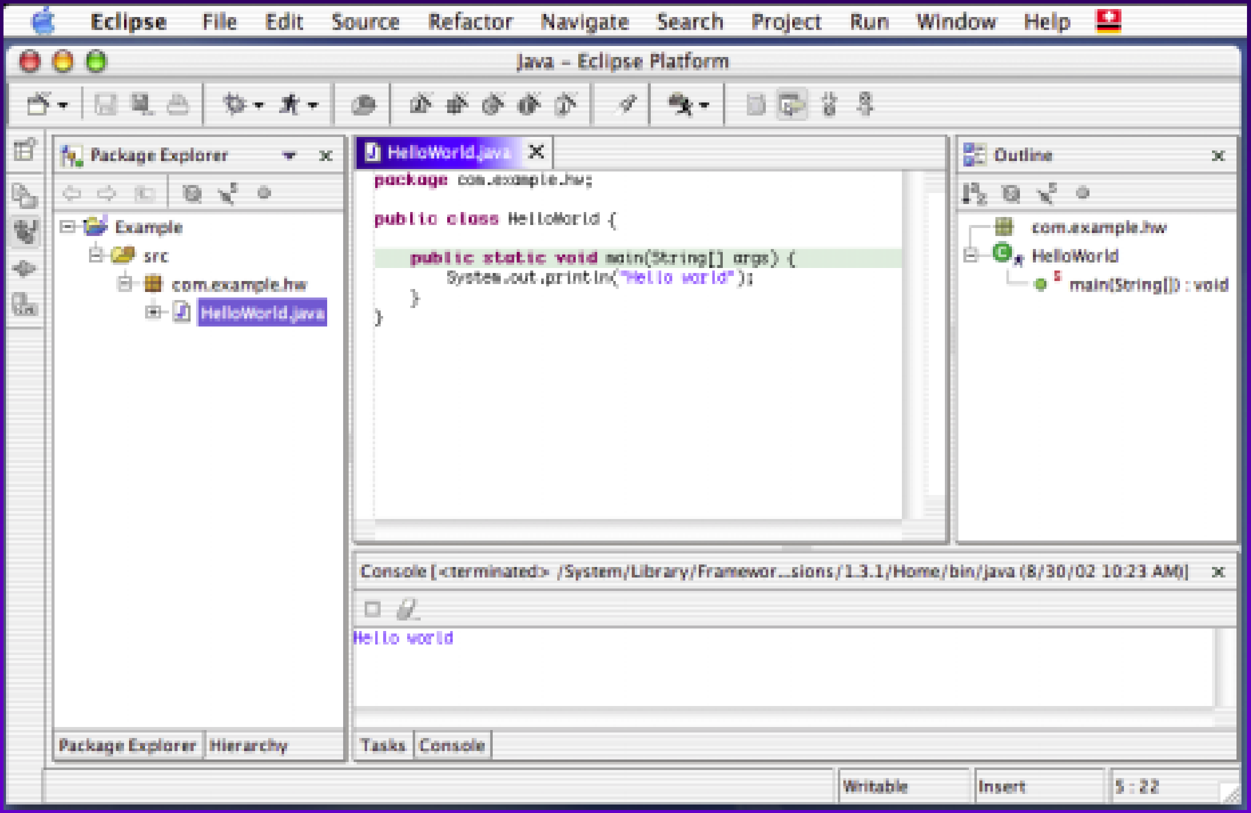
```
package com.example.hw;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Console [<terminated> /home/torres/Eclipse/vm/IBMJava2-131/jre/bin/javaw (8/30/02 11:13 AM)]

```
Hello world
```

# Why SWT?

- Eclipse Platform on Mac OS X - Carbon



```
package com.example.hw;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

```
Hello world
```

## JFace

- JFace is set of UI frameworks for common UI tasks
- Designed to be used in conjunction with SWT
- Classes for handling common UI tasks
- API and implementation are window-system independent

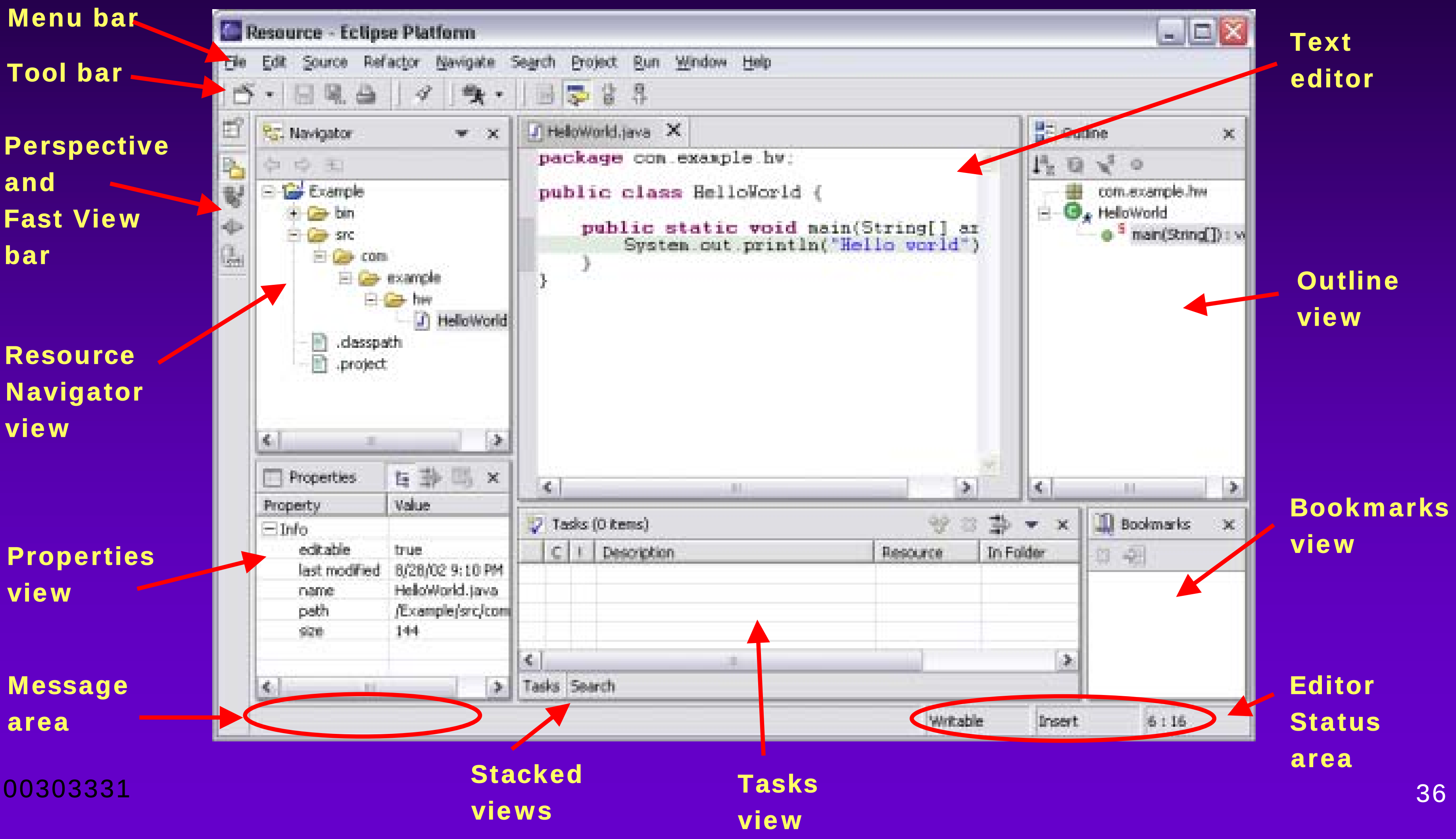
## JFace APIs

- Image and font registries
- Dialog, preference, and wizard frameworks
- Structured viewers
  - Model-aware adapters for SWT tree, table, list widgets
- Text infrastructure
  - Document model for SWT styled text widget
  - Coloring, formatting, partitioning, completion
- Actions
  - Location-independent user commands
  - Contribute action to menu, tool bar, or button

## Workbench Component

- Workbench is UI personality of Eclipse Platform
- UI paradigm centered around
  - Editors
  - Views
  - Perspectives

# Workbench Terminology



```
package com.example.hw;

public class HelloWorld {

    public static void main(String[] ar
        System.out.println("Hello world")
    }
}
```

| Property | Value |
|---|---|
| Info | |
| editable | true |
| last modified | 8/28/02 9:10 PM |
| name | HelloWorld.java |
| path | /Example/src/com |
| size | 144 |

## Editors

- Editors appear in workbench editor area
- Contribute actions to workbench menu and tool bars
- Open, edit, save, close lifecycle
- Open editors are stacked

- Extension point for contributing new types of editors
- Example: JDT provides Java source file editor
- Eclipse Platform includes simple text file editor
- Windows only: embed any OLE document as editor
- Extensive text editor API and framework

## Views

- Views provide information on some object
- Views augment editors
  - Example: Outline view summarizes content
- Views augment other views
  - Example: Properties view describes selection

- Extension point for new types of views
- Eclipse Platform includes many standard views
  - Resource Navigator, Outline, Properties, Tasks, Bookmarks, Search, ...
- View API and framework
  - Views can be implemented with JFace viewers

## Perspectives

- Perspectives are arrangements of views and editors
- Different perspectives suited for different user tasks
- Users can quickly switch between perspectives
- Task orientation limits visible views, actions
  - Scales to large numbers of installed tools
- Perspectives control
  - View visibility
  - View and editor layout
  - Action visibility
- Extension point for new perspectives
- Eclipse Platform includes standard perspectives
  - Resource, Debug, ...
- Perspective API

## Other Workbench Features

- Tools may also
  - Add global actions
  - Add actions to existing views and editors
  - Add views, action sets to existing perspectives

- Eclipse Platform is accessible (Section 508)
- Accessibility mechanisms available to all plug-ins

## Workbench Responsibilities

- Eclipse Platform manages windows and perspectives
- Eclipse Platform creates menu and tool bars
  - Labels and icons listed in plug-in manifest
  - Contributing plug-ins not activated
- Eclipse Platform creates views and editors
  - Instantiated only as needed
- Scalable to large numbers of installed tools

## Team Component

- Version and configuration management (VCM)
- Share resources with team via a **repository**
- Repository associated at project level
- Extension point for new types of repositories
- Repository provider API and framework
- Eclipse Platform includes CVS repository provider
- Available repository providers*
  - ChangeMan (Serena) - AllFusion Harvest (CA)
  - ClearCase (Rational) - Perforce
  - CM Synergy (Telelogic) - Source Integrity (MKS)
  - PVCS (Merant) - TeamCode (Interwoven)
  - Microsoft Visual Source Safe

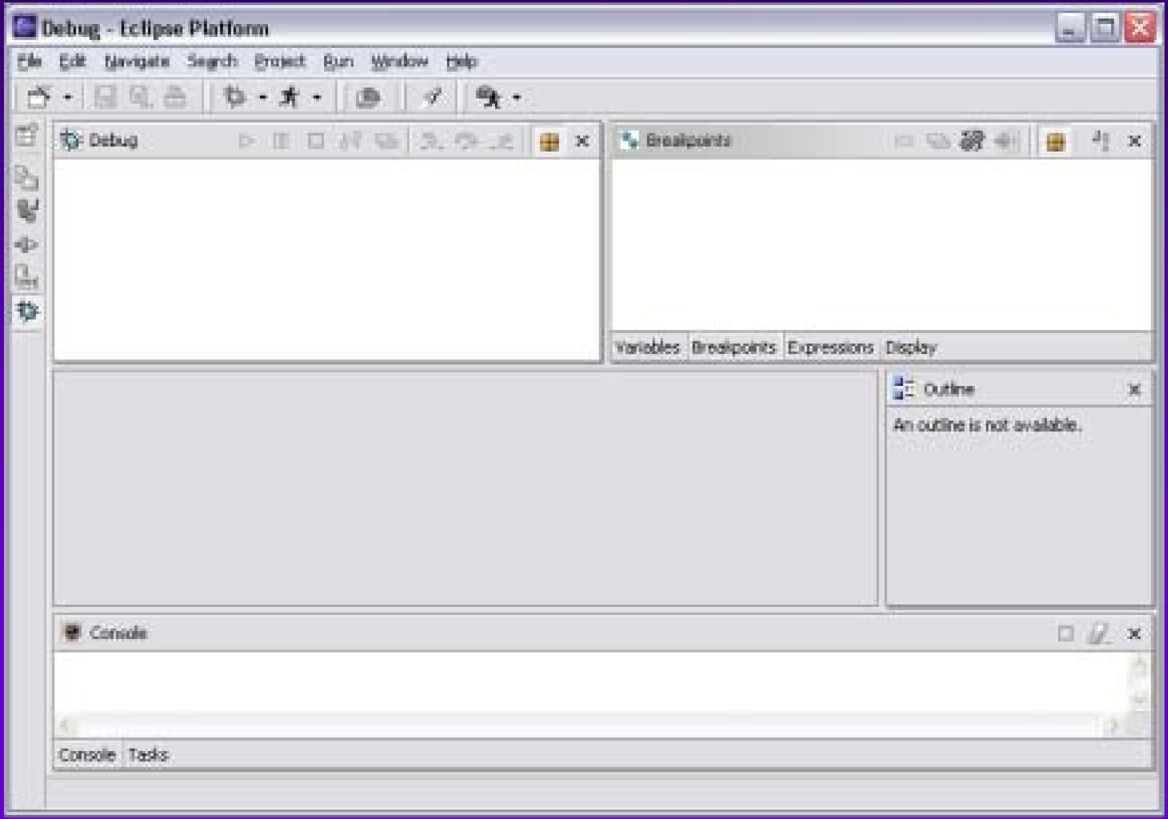
*** March 2003**

## Team Component

- Repository providers have wide latitude
  - Provide actions suited to repository
  - No built-in process model
- Integrate into workbench UI via
  - Share project configuration wizard
  - Actions on Team menu
  - Resource decorators
  - Repository-specific preferences
  - Specialized views for repository browsing, …

## Debug Component

- Common debug UI and underlying debug model


Debug - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Debug
Breakpoints
Variables Breakpoints Expressions Display
Outline
Console
Console Tasks

An outline is not available.

## Debug Component

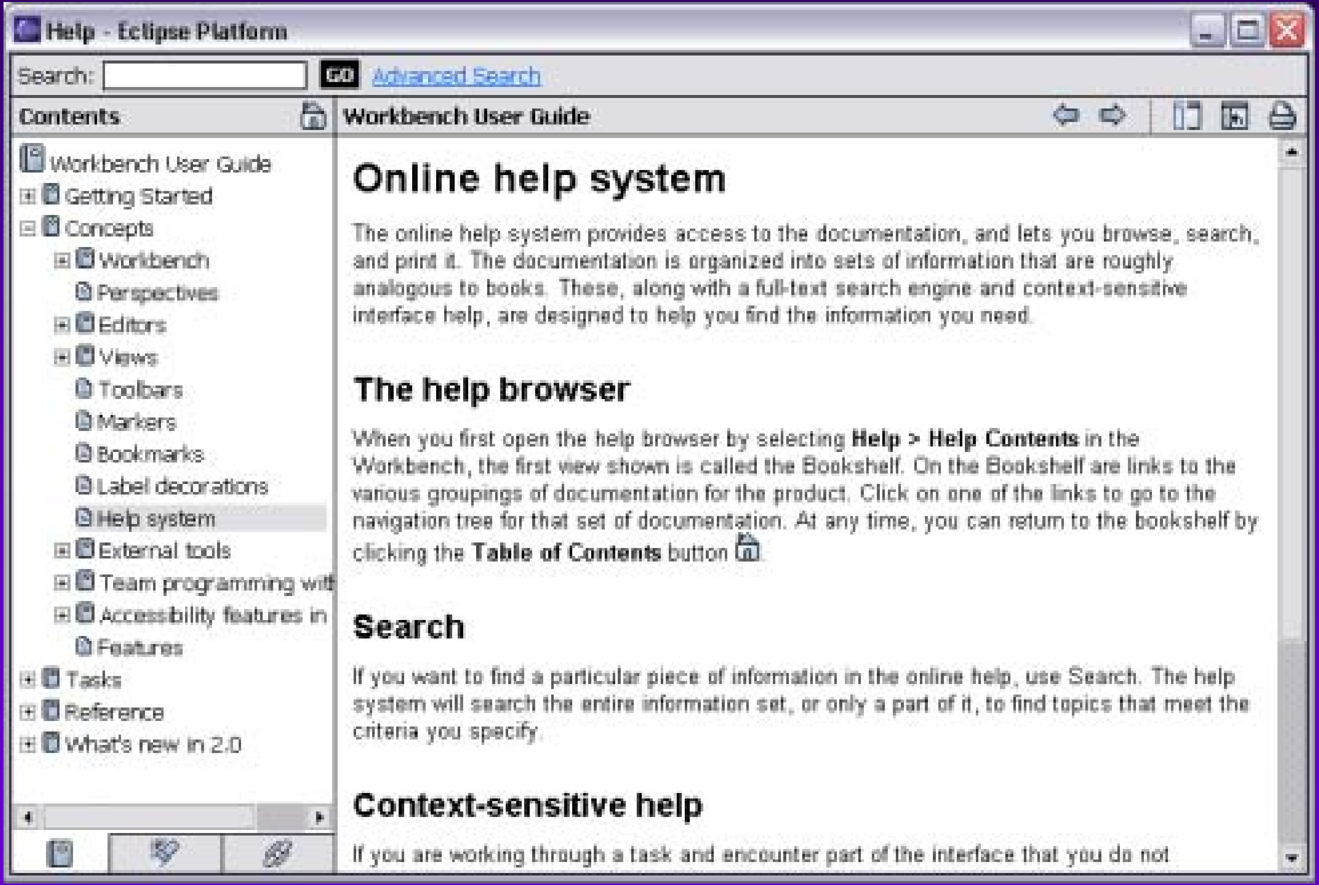
- Launch configurations
  - How to run a program (debug mode option)
- Generic debug model
  - Standard debug events: suspended, exit, ...
  - Standard debug actions: resume, terminate, step, ...
  - Breakpoints
  - Expressions
  - Source code locator
- Generic debug UI
  - Debug perspective
  - Debug views: stack frames, breakpoints, ...
- Example: JDT supplies Java launcher and debugger
  - Java debugger based on JPDA
- Debug mechanisms available to other plug-ins

## Ant Component

- Eclipse incorporates Apache Ant
- Ant is Java-based build tool
  - "Kind of like Make...without Make's wrinkles"
- XML-based build files instead of makefiles
- Available from workbench External Tools menu
- Run Ant targets in build files inside or outside workspace
- PDE uses Ant for building deployed form of plug-in

## Help Component

- Help is presented in a standard web browser



### Online help system

The online help system provides access to the documentation, and lets you browse, search, and print it. The documentation is organized into sets of information that are roughly analogous to books. These, along with a full-text search engine and context-sensitive interface help, are designed to help you find the information you need.

### The help browser

When you first open the help browser by selecting **Help > Help Contents** in the Workbench, the first view shown is called the Bookshelf. On the Bookshelf are links to the various groupings of documentation for the product. Click on one of the links to go to the navigation tree for that set of documentation. At any time, you can return to the bookshelf by clicking the **Table of Contents** button.

### Search

If you want to find a particular piece of information in the online help, use Search. The help system will search the entire information set, or only a part of it, to find topics that meet the criteria you specify.

### Context-sensitive help

If you are working through a task and encounter part of the interface that you do not

## Help Component

- Help books are HTML webs
- Extension points for contributing
  - entire books
  - sections to existing books
  - F1-help pop ups
- Eclipse Platform contributes
  - “Workbench User Guide”
  - “Platform Plug-in Developer Guide” (APIs)
  - F1-help for views, editors, dialogs, …
- JDT and PDE contribute their own help
- Help mechanisms available to all plug-ins

- Help search engine based on Apache Lucene
- Headless help server based on Apache Tomcat

## Internationalization

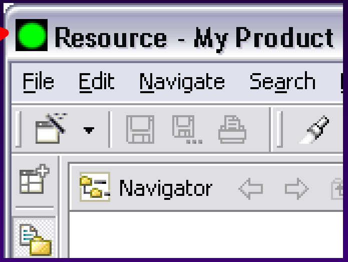
- Eclipse Platform is internationalized
- 2.0 translations available for following languages

| | |
|---|---|
| English | German |
| Spanish | Italian |
| French | Portugese (Brazil) |
| Japanese | Korean |
| Chinese (Traditional) | Chinese (Simplified) |

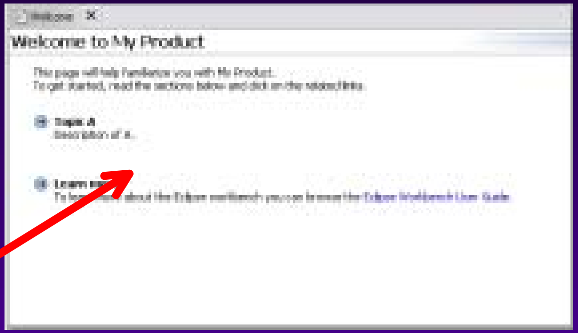
- Translations live in plug-in fragments
  - Separately shippable
- Internalization mechanisms available to all plug-ins

# Product Information

Window image



Welcome pages



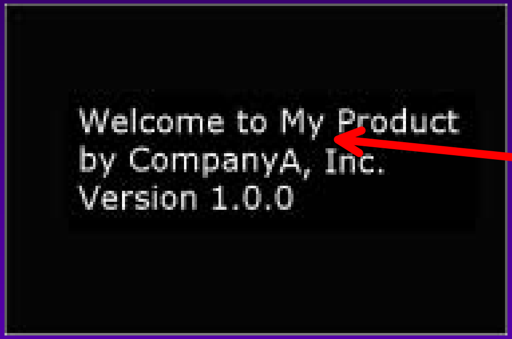
Welcome to My Product

This page will help familiarize you with My Product.
To get started, read the sections below and click on the related links.

**Topic A**
Description of A...

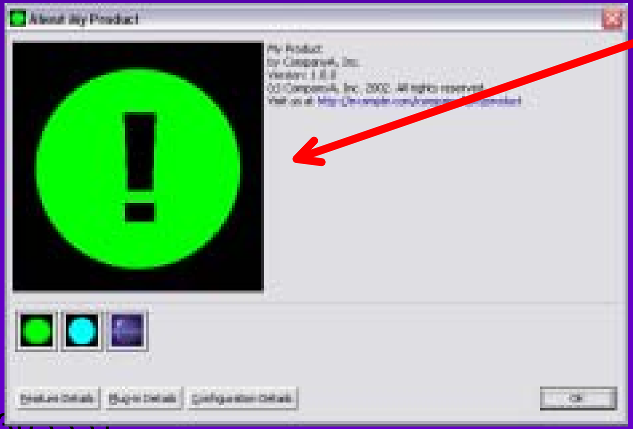
**Learn more**
To learn more about the Eclipse workbench you can browse the Eclipse Workbench User Guide

Splash screen



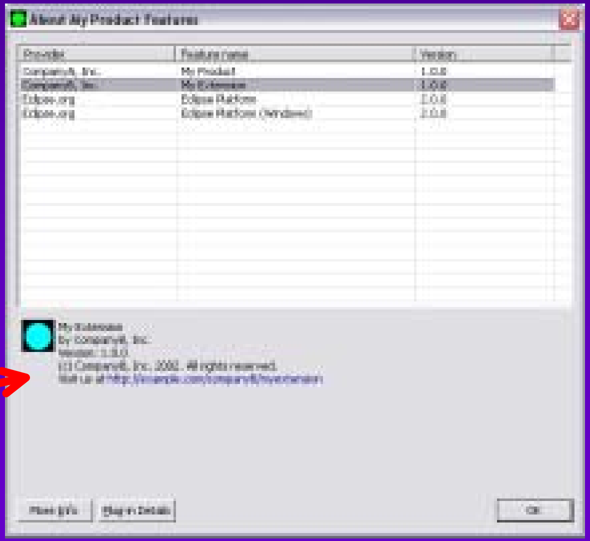
Welcome to My Product
by CompanyA, Inc.
Version 1.0.0

About product info



My Product
by CompanyA, Inc.
Version: 1.0.0
(c) CompanyA, Inc. 2002. All rights reserved.
Visit us at http://example.com/companyA/myproduct

About feature info



| Provider | Feature name | Version |
|---|---|---|
| CompanyA, Inc. | My Product | 1.0.0 |
| CompanyB, Inc. | My Extension | 1.0.0 |
| Eclipse.org | Eclipse Platform | 2.0.0 |
| Eclipse.org | Eclipse Platform (Windows) | 2.0.0 |

My Extension
by CompanyB, Inc.
Version: 1.0.0
(c) CompanyB, Inc. 2002. All rights reserved.
Visit us at http://example.com/companyB/myextension

## Product Information

- Primary feature controls product information
  - Splash screen
  - Window image
  - About product info
  - Initial welcome page
  - Default perspective
  - Preference default overrides
- All features can provide
  - Welcome page
  - About feature info

## Eclipse Platform - Summary

- Eclipse Platform is the nucleus of IDE products
- Plug-ins, extension points, extensions
  - Open, extensible architecture
- Workspace, projects, files, folders
  - Common place to organize & store development artifacts
- Workbench, editors, views, perspectives
  - Common user presentation and UI paradigm
- Key building blocks and facilities
  - Help, team support, internationalization, …

**Eclipse is a universal platform for integrating development tools**

## Java Development Tools

- JDT = Java development tools
- State of the art Java development environment

- Built atop Eclipse Platform
  - Implemented as Eclipse plug-ins
  - Using Eclipse Platform APIs and extension points

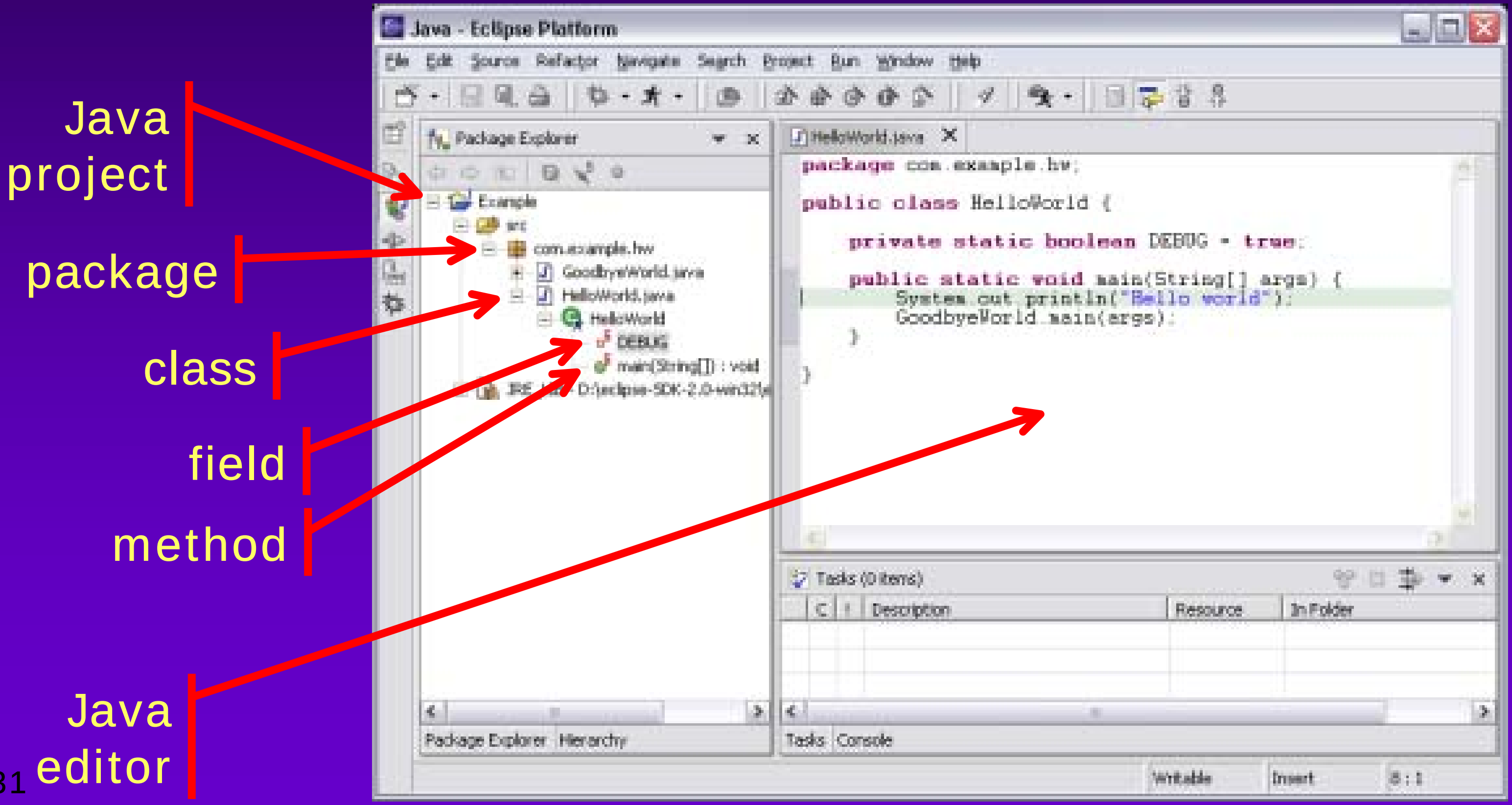
- Included in Eclipse Project releases
  - Available as separately installable feature
  - Part of Eclipse SDK drops

## JDT Goals

- Goal: To be #1 Java IDE
- Goal: To make Java programmers smile

# Java Perspective

- Java-centric view of files in Java projects
  - Java elements meaningful for Java programmers



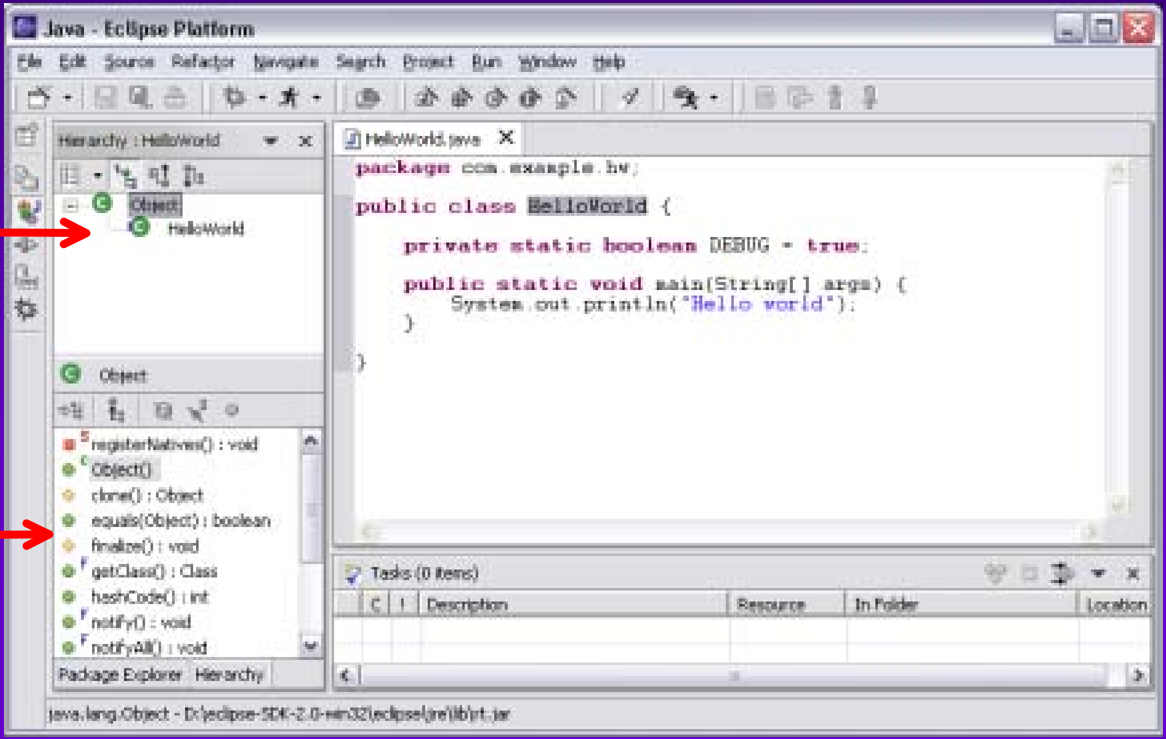
```
package com.example.hw;

public class HelloWorld {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
        GoodbyeWorld.main(args);
    }

}
```

# Java Perspective

- Browse type hierarchies
  - "Up" hierarchy to supertypes
  - "Down" hierarchy to subtypes

Type hierarchy

Selected type's members

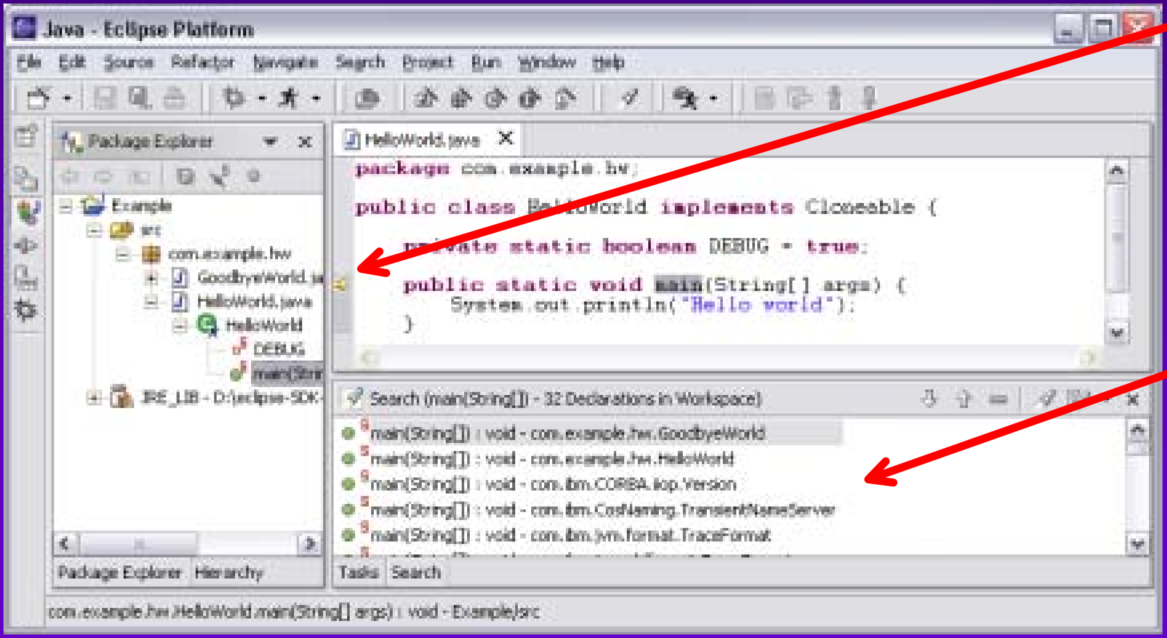


```
package com.example.hw;

public class HelloWorld {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
    }

}
```

java.lang.Object - D:\eclipse-SDK-2.0-win32\eclipse\jre\lib\rt.jar

# Java Perspective

- Search for Java elements
  - Declarations or references
  - Including libraries and other projects



```
package com.example.hw;

public class HelloWorld implements Cloneable {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
```

Search (main(String[]) - 32 Declarations in Workspace)

```
main(String[]) : void - com.example.hw.GoodbyeWorld
main(String[]) : void - com.example.hw.HelloWorld
main(String[]) : void - com.ibm.CORBA.iiop.Version
main(String[]) : void - com.ibm.CosNaming.TransientNameServer
main(String[]) : void - com.ibm.jvm.format.TraceFormat
```

com.example.hw.HelloWorld.main(String[] args) : void - Example/src

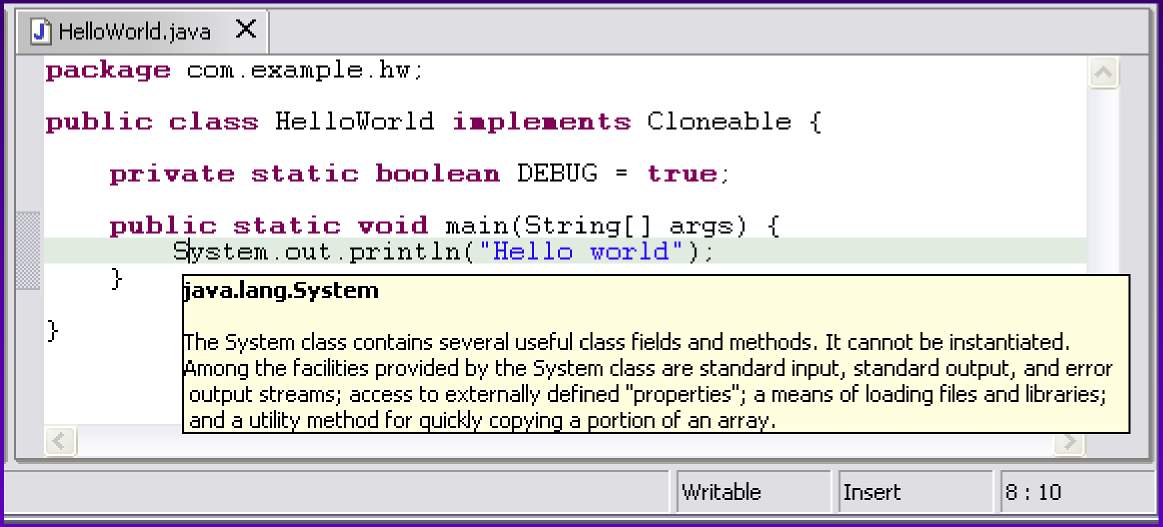
Hits flagged in margin of editor

All search results

eclipse

# Java Editor

- Hovering over identifier shows Javadoc spec



HelloWorld.java

```
package com.example.hw;

public class HelloWorld implements Cloneable {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
    }

}
```

**java.lang.System**

The System class contains several useful class fields and methods. It cannot be instantiated. Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined "properties"; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

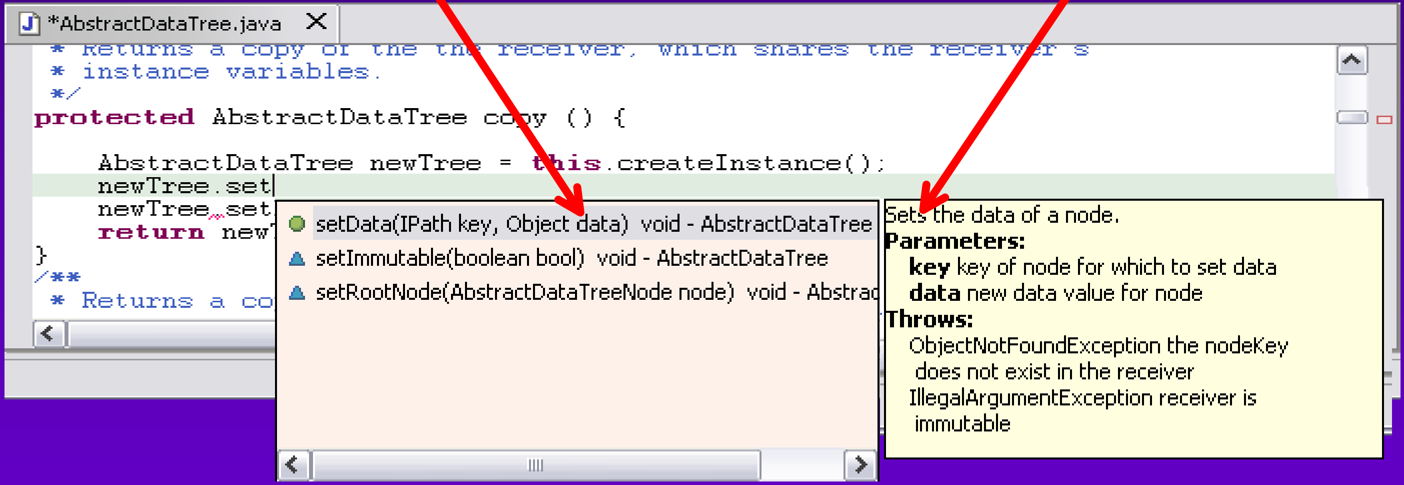
Writable | Insert | 8 : 10

# Java Editor

- Method completion in Java editor

List of plausible methods

Doc for method



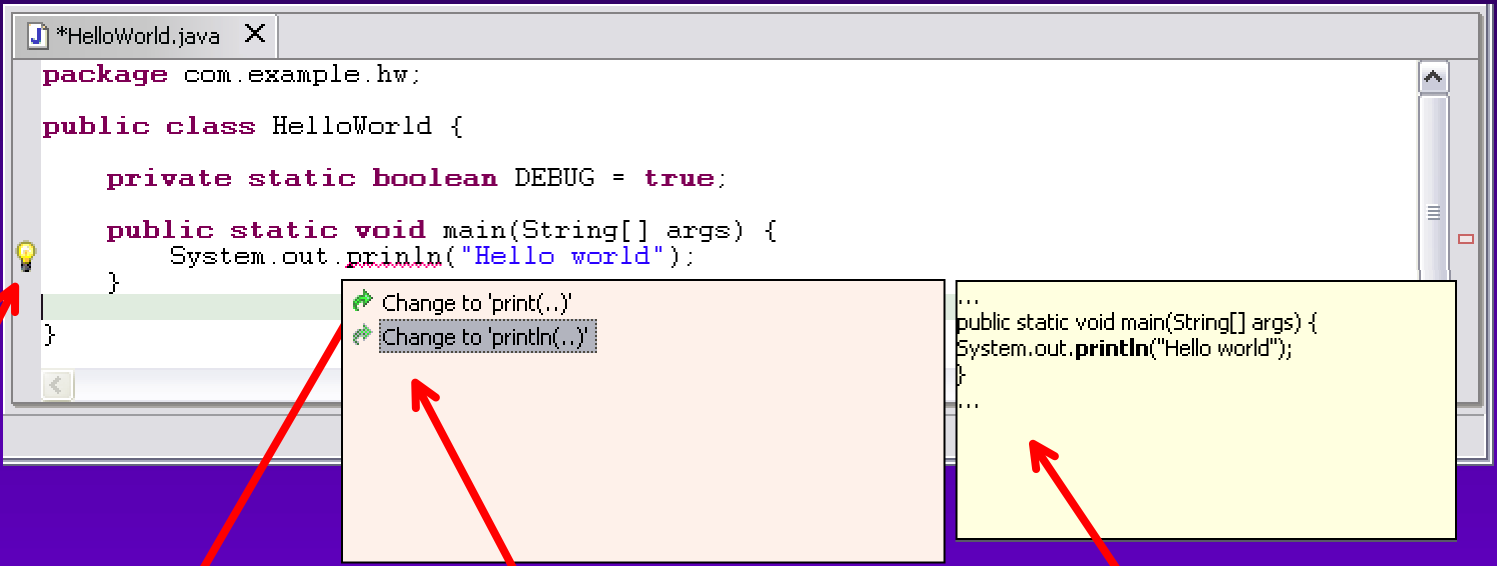
```
 * Returns a copy of the receiver, which shares the receiver's
 * instance variables.
 */
protected AbstractDataTree copy () {

    AbstractDataTree newTree = this.createInstance();
    newTree.set
    newTree.set
    return new
}
/**
 * Returns a co
```

```
setData(IPath key, Object data)  void - AbstractDataTree
setImmutable(boolean bool)  void - AbstractDataTree
setRootNode(AbstractDataTreeNode node)  void - Abstrac
```

Sets the data of a node.
**Parameters:**
**key** key of node for which to set data
**data** new data value for node
**Throws:**
ObjectNotFoundException the nodeKey does not exist in the receiver
IllegalArgumentException receiver is immutable

# Java Editor

- On-the-fly spell check catches errors early

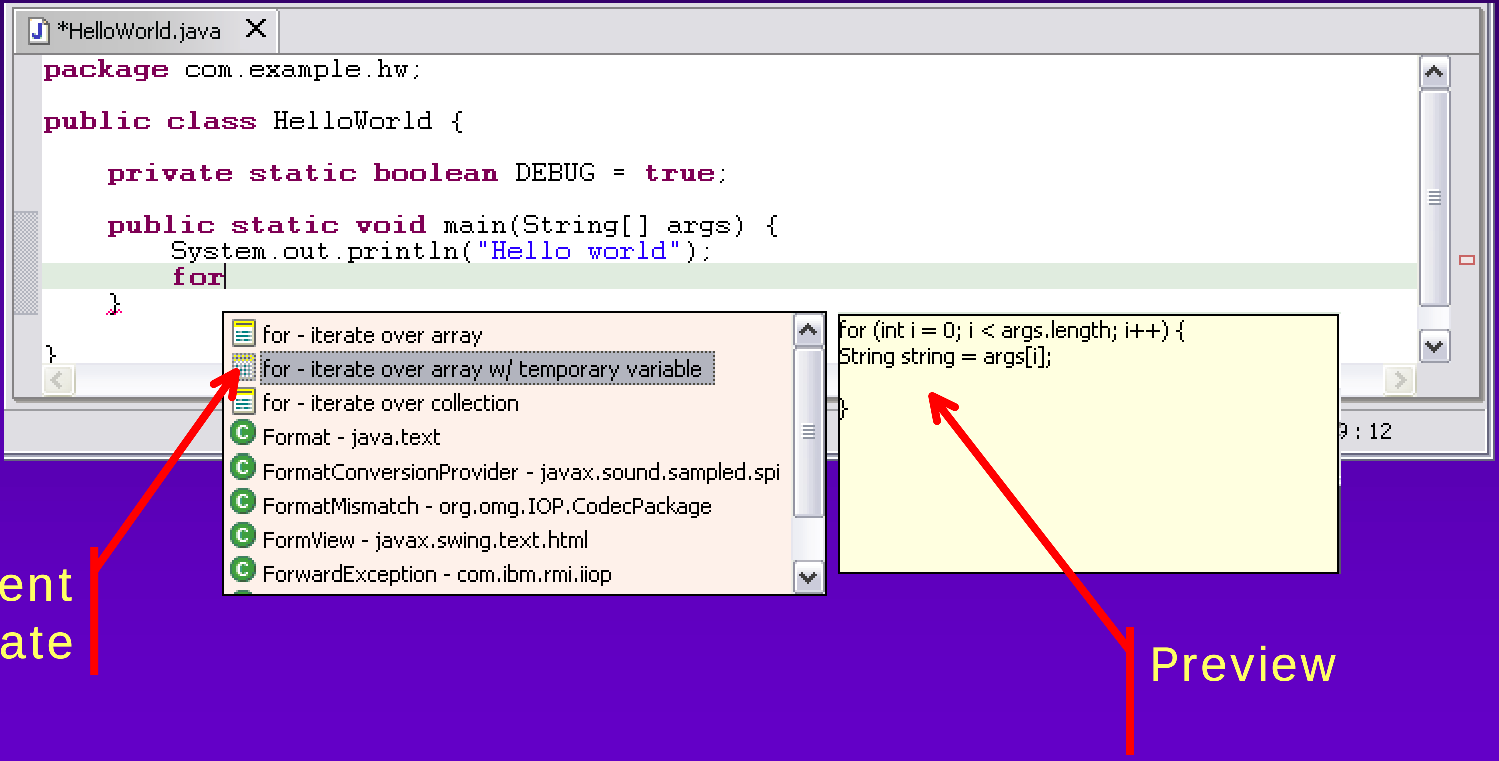


```
*HelloWorld.java
package com.example.hw;

public class HelloWorld {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.prinln("Hello world");
    }

}
```

```
...
public static void main(String[] args) {
System.out.println("Hello world");
}
...
```

## Java Editor

- Code templates help with drudgery



```
package com.example.hw;

public class HelloWorld {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
        for
    }

}
```

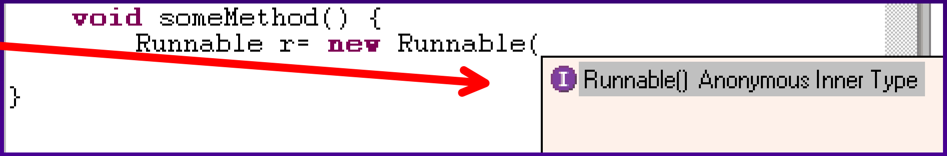
```
for - iterate over array
for - iterate over array w/ temporary variable
for - iterate over collection
Format - java.text
FormatConversionProvider - javax.sound.sampled.spi
FormatMismatch - org.omg.IOP.CodecPackage
FormView - javax.swing.text.html
ForwardException - com.ibm.rmi.iiop
```

```
for (int i = 0; i < args.length; i++) {
String string = args[i];

}
```

# Java Editor

- Java editor creates stub methods

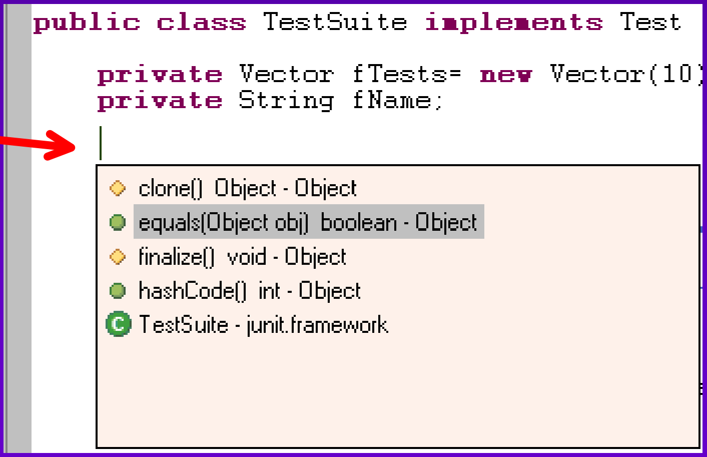
Method stub insertion for anonymous inner types



```
    void someMethod() {
        Runnable r= new Runnable(
}
```

Runnable() Anonymous Inner Type

Method stub insertion for inherited methods



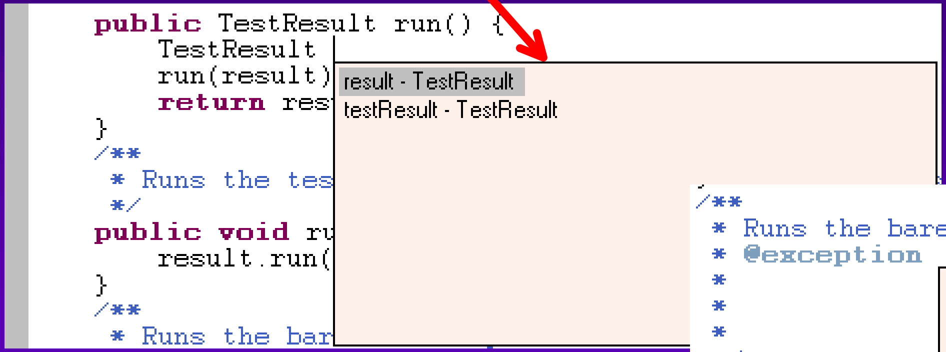
```
public class TestSuite implements Test

    private Vector fTests= new Vector(10)
    private String fName;
```

clone() Object - Object
equals(Object obj) boolean - Object
finalize() void - Object
hashCode() int - Object
TestSuite - junit.framework

# Java Editor

- Java editor helps programmers write good Java code

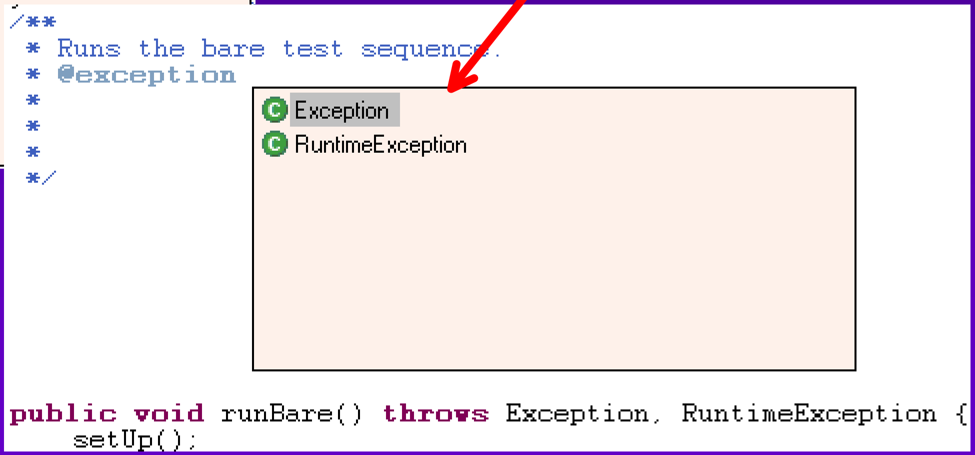
Variable name suggestion



```
public TestResult run() {
    TestResult
    run(result)
    return res
}
/**
 * Runs the tes
 */
public void ru
    result.run(
}
/**
 * Runs the bar
```

result - TestResult
testResult - TestResult

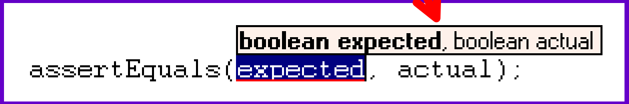
JavaDoc code assist



```
/**
 * Runs the bare test sequence.
 * @exception
 *
 *
 *
 */
public void runBare() throws Exception, RuntimeException {
    setUp();
```

Exception
RuntimeException

Argument hints and proposed argument names



**boolean expected**, boolean actual

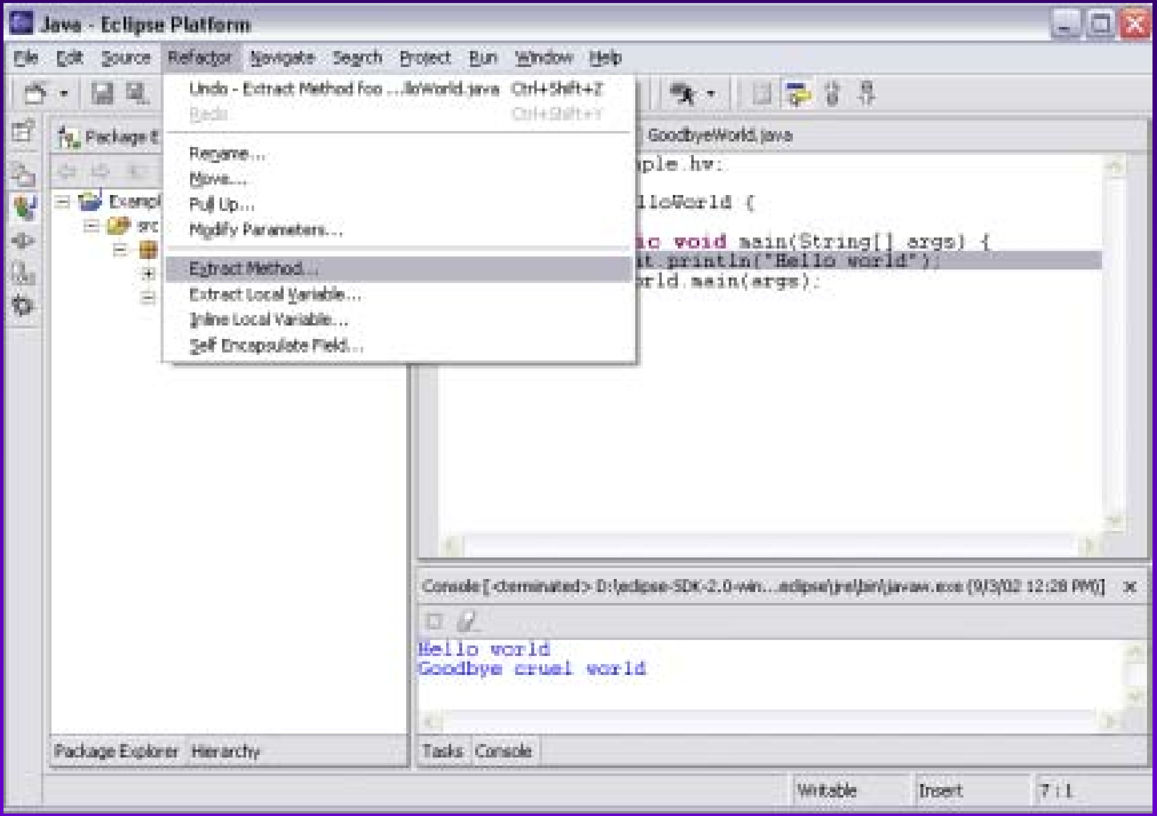
```
assertEquals(expected, actual);
```

## Java Editor

- Other features of Java editor include
  - Local method history
  - Code formatter
  - Source code for binary libraries
  - Built-in refactoring

## Refactoring

- JDT has actions for refactoring Java code


Java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Undo - Extract Method foo ...lloWorld.java Ctrl+Shift+Z
Redo Ctrl+Shift+Y
Rename...
Move...
Pull Up...
Modify Parameters...
Extract Method...
Extract Local Variable...
Inline Local Variable...
Self Encapsulate Field...
Package E
Exampl
src
GoodbyeWorld.java
Package Explorer Hierarchy
Tasks Console
Writable Insert 7 : 1

```
ple.hw;

lloWorld {

ic void main(String[] args) {
t.println("Hello world");
rld.main(args);
```

Console [<terminated> D:\eclipse-SDK-2.0-win...eclipse\jre\bin\javaw.exe (9/3/02 12:28 PM)]

```
Hello world
Goodbye cruel world
```

## Refactoring

- Refactoring actions rewrite source code
  - Within a single Java source file
  - Across multiple interrelated Java source files
- Refactoring actions preserve program semantics
  - Does not alter what program does
  - Just affects the way it does it

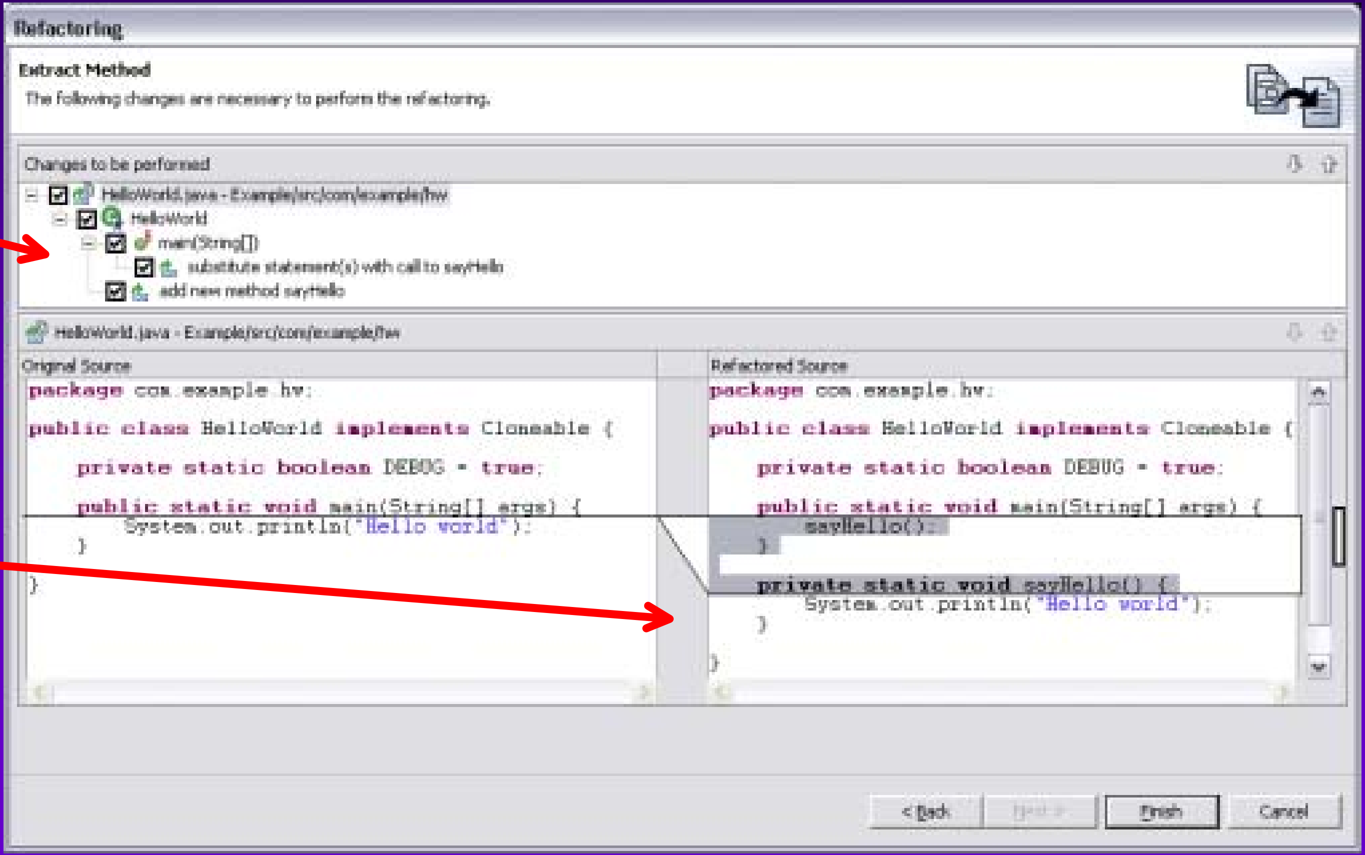
- Encourages exploratory programming
- Encourages higher code quality
  - Makes it easier to rewrite poor code

# Refactoring

- Full preview of all ensuing code changes
  – Programmer can veto individual changes

List of changes

"before" vs. "after"



Refactoring

**Extract Method**

The following changes are necessary to perform the refactoring.

Changes to be performed

```
☑ HelloWorld.java - Example/src/com/example/hw
  ☑ HelloWorld
    ☑ main(String[])
      ☑ substitute statement(s) with call to sayHello
    ☑ add new method sayHello
```

HelloWorld.java - Example/src/com/example/hw

Original Source

```
package com.example.hw;

public class HelloWorld implements Cloneable {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Refactored Source

```
package com.example.hw;

public class HelloWorld implements Cloneable {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        sayHello();
    }

    private static void sayHello() {
        System.out.println("Hello world");
    }
}
```

## Refactoring

- Growing catalog of refactoring actions
  - Organize imports
  - Rename {field, method, class, package}
  - Move {field, method, class}
  - Extract {method, local variable, interface}
  - Inline {method, local variable}
  - Reorder method parameters
  - Push members down

  ...

## Eclipse Java Compiler

- Eclipse Java compiler
  - JCK-compliant Java compiler (selectable 1.3 and 1.4)
  - Helpful error messages
  - Generates runnable code even in presence of errors
  - Fully-automatic incremental recompilation
  - High performance
  - Scales to large projects
- Multiple other uses besides the obvious
  - Syntax and spell checking
  - Analyze structure inside Java source file
  - Name resolution
  - Content assist
  - Refactoring
  - Searches

# Eclipse Java Debugger

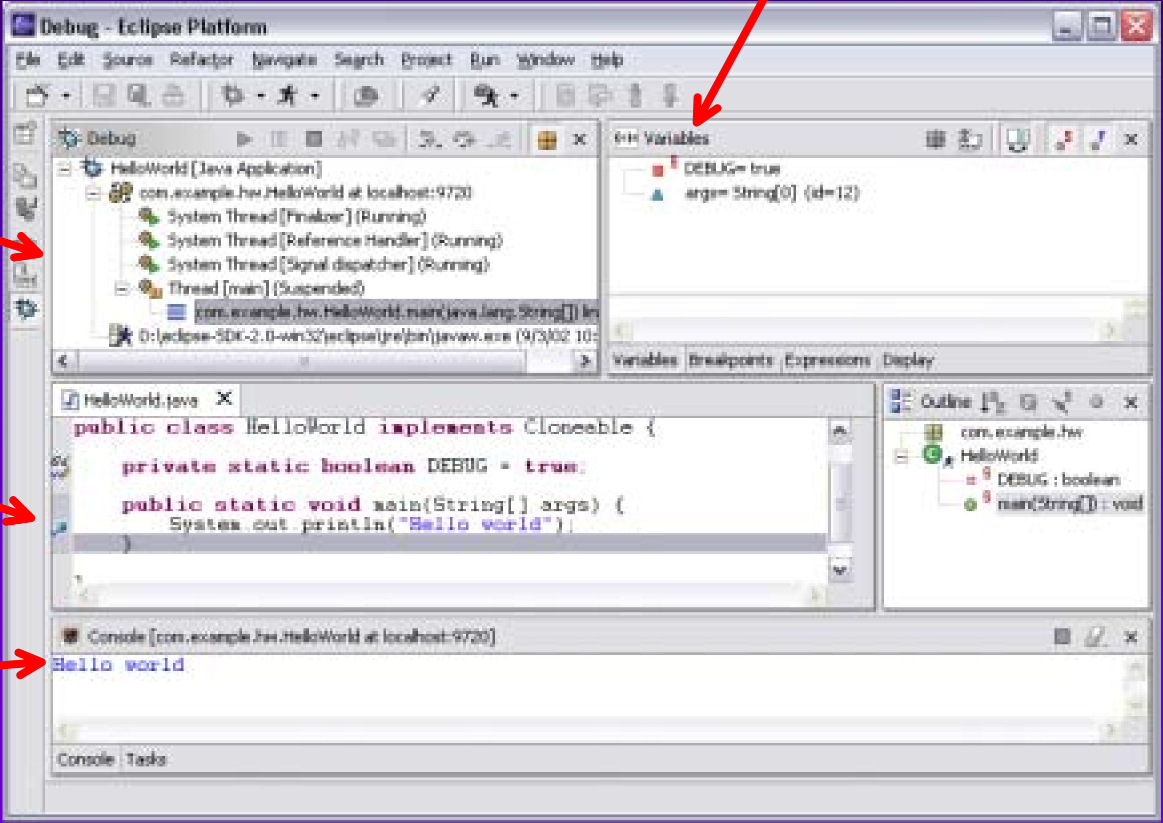
- Run or debug Java programs

Local variables

Threads and stack frames

Editor with breakpoint marks

Console I/O


Debug - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Variables
Variables Breakpoints Expressions Display
HelloWorld.java
Outline
Console Tasks

```
HelloWorld [Java Application]
  com.example.hw.HelloWorld at localhost:9720
    System Thread [Finalizer] (Running)
    System Thread [Reference Handler] (Running)
    System Thread [Signal dispatcher] (Running)
    Thread [main] (Suspended)
      com.example.hw.HelloWorld.main(java.lang.String[]) lin
  D:\eclipse-SDK-2.0-win32\eclipse\jre\bin\javaw.exe (9/3/02 10:
```

```
DEBUG= true
args= String[0] (id=12)
```

```java
public class HelloWorld implements Cloneable {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
```

```
com.example.hw
HelloWorld
  DEBUG : boolean
  main(String[]) : void
```

Console [com.example.hw.HelloWorld at localhost:9720]

```
Hello world
```

## Eclipse Java Debugger

- Run Java programs
  - In separate target JVM (user selectable)
  - Console provides stdout, stdin, stderr
  - Scrapbook pages for executing Java code snippets
- Debug Java programs
  - Full source code debugging
  - Any JPDA-compliant JVM
- Debugger features include
  - Method and exception breakpoints
  - Conditional breakpoints
  - Watchpoints
  - Step over, into, return; run to line
  - Inspect and modify fields and local variables
  - Evaluate snippets in context of method
  - Hot swap (if target JVM supports)

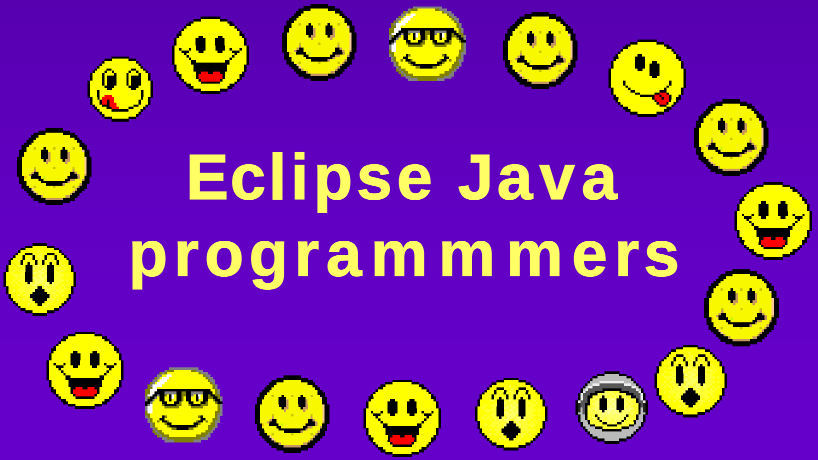
## JDT APIs

- JDT APIs export functionality to other plug-ins

- Java model
  - Java-centric analog of workspace
  - Tree of Java elements (down to individual methods)
  - Java element deltas
  - Type hierarchies
  - Model accurate independent of builds
- Building blocks
  - Java scanner
  - Java class file reader
  - Java abstract syntax trees (down to expressions)
- Many others…

# Eclipse JDT - Summary

- JDT is a state of the art Java IDE
- Java views, editor, refactoring
  - Helps programmer write and maintain Java code
- Java compiler
  - Takes care of translating Java sources to binaries
- Java debugger
  - Allows programmer to get inside the running program



## Plug-in Development Environment

- PDE = Plug-in development environment
- Specialized tools for developing Eclipse plug-ins

- Built atop Eclipse Platform and JDT
  - Implemented as Eclipse plug-ins
  - Using Eclipse Platform and JDT APIs and extension points

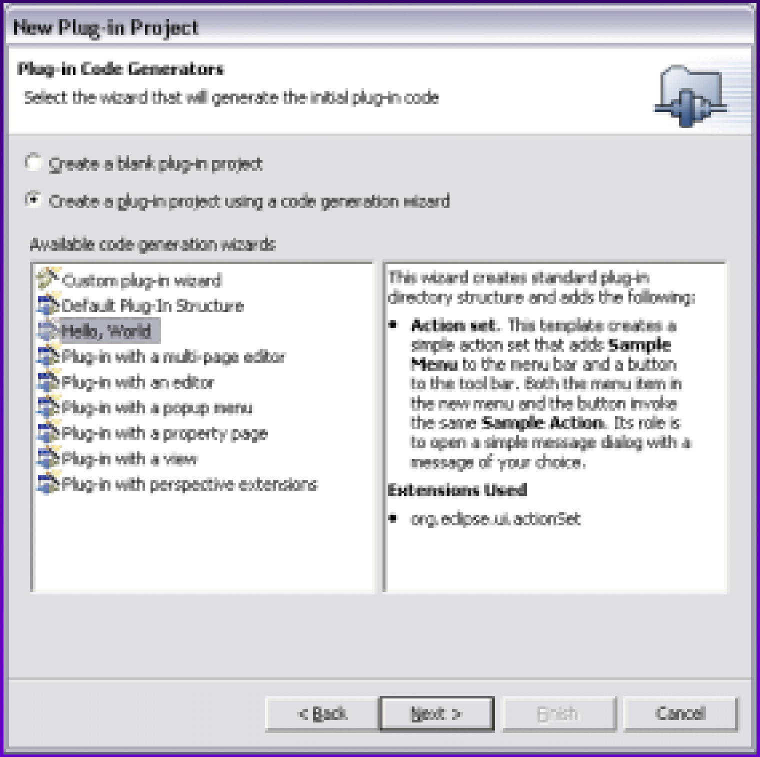
- Included in Eclipse Project releases
  - Separately installable feature
  - Part of Eclipse SDK drops

## PDE Goals

- Goal: To make it easier to develop Eclipse plug-ins
- Goal: Support self-hosted Eclipse development

# PDE

- PDE templates for creating simple plug-in projects



**Plug-in Code Generators**

Select the wizard that will generate the initial plug-in code

( ) Create a blank plug-in project

(•) Create a plug-in project using a code generation wizard

Available code generation wizards:

- Custom plug-in wizard
- Default Plug-In Structure
- Hello, World
- Plug-in with a multi-page editor
- Plug-in with an editor
- Plug-in with a popup menu
- Plug-in with a property page
- Plug-in with a view
- Plug-in with perspective extensions

This wizard creates standard plug-in directory structure and adds the following:

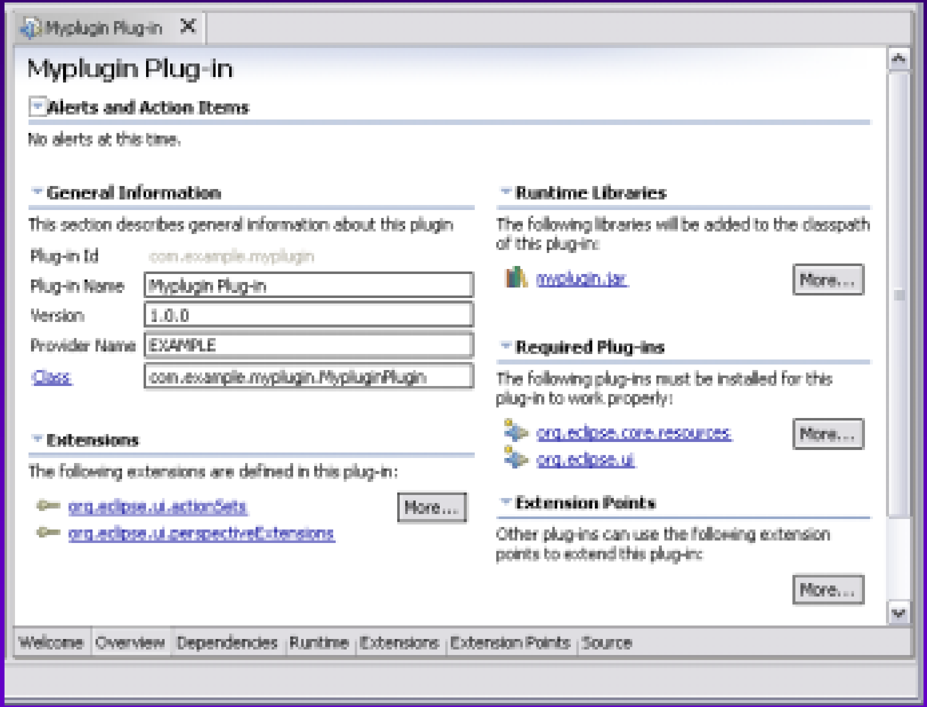
- **Action set.** This template creates a simple action set that adds **Sample Menu** to the menu bar and a button to the tool bar. Both the menu item in the new menu and the button invoke the same **Sample Action**. Its role is to open a simple message dialog with a message of your choice.

**Extensions Used**

- org.eclipse.ui.actionSet

# PDE

- Specialized PDE editor for plug-in manifest files



Myplugin Plug-in

Alerts and Action Items

No alerts at this time.

General Information

This section describes general information about this plugin

| | |
|---|---|
| Plug-in Id | com.example.myplugin |
| Plug-in Name | Myplugin Plug-in |
| Version | 1.0.0 |
| Provider Name | EXAMPLE |
| Class | com.example.myplugin.MypluginPlugin |

Extensions

The following extensions are defined in this plug-in:

org.eclipse.ui.actionSets
org.eclipse.ui.perspectiveExtensions

Runtime Libraries

The following libraries will be added to the classpath of this plug-in:

myplugin.jar

Required Plug-ins

The following plug-ins must be installed for this plug-in to work properly:

org.eclipse.core.resources
org.eclipse.ui

Extension Points

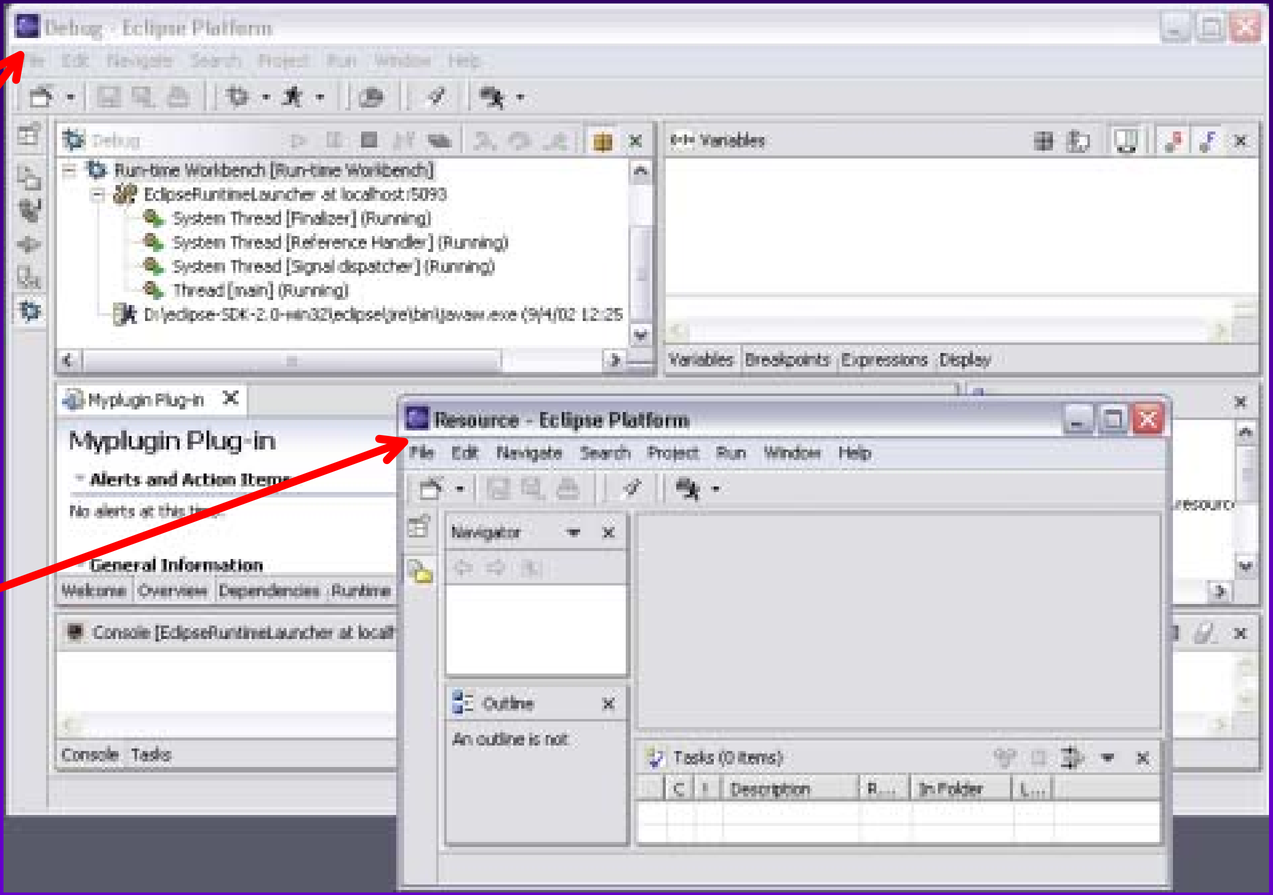
Other plug-ins can use the following extension points to extend this plug-in:

# PDE

- PDE runs and debugs another Eclipse workbench

1. Workbench running PDE (host)

2. Run-time workbench (target)


Debug - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Debug
Variables
Variables Breakpoints Expressions Display
Myplugin Plug-in
Myplugin Plug-in
Alerts and Action Items
No alerts at this time.
General Information
Welcome Overview Dependencies Runtime
Console [EclipseRuntimeLauncher at local
Console Tasks
Resource - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Navigator
Outline
An outline is not
Tasks (0 items)
C ! Description R... In Folder L...

```
Run-time Workbench [Run-time Workbench]
  EclipseRuntimeLauncher at localhost:5093
    System Thread [Finalizer] (Running)
    System Thread [Reference Handler] (Running)
    System Thread [Signal dispatcher] (Running)
    Thread [main] (Running)
  D:\eclipse-SDK-2.0-win32\eclipse\jre\bin\javaw.exe (9/4/02 12:25
```

## PDE - Summary

- PDE makes it easier to develop Eclipse plug-ins
- PDE also generates Ant build scripts
  - Compile and create deployed form of plug-in

**PDE is basis for self-hosted Eclipse development**

## Eclipse Operating Environments

- Eclipse Platform currently* runs on
  - Microsoft® Windows® XP, 2000, NT, ME, 98SE
  - Linux® on Intel x86 - Motif, GTK
    - RedHat Linux 8.0 x86
    - SuSE Linux 8.1 x86
  - Sun Solaris 8 SPARC – Motif
  - HP-UX 11i hp9000 – Motif
  - IBM® AIX 5.1 on PowerPC – Motif
  - Apple Mac OS® X 10.2 on PowerPC – Carbon
  - QNX® Neutrino® RTOS 6.2.1 - Photon®

*** Eclipse 2.1 - March 2003**

## Other Operating Environments

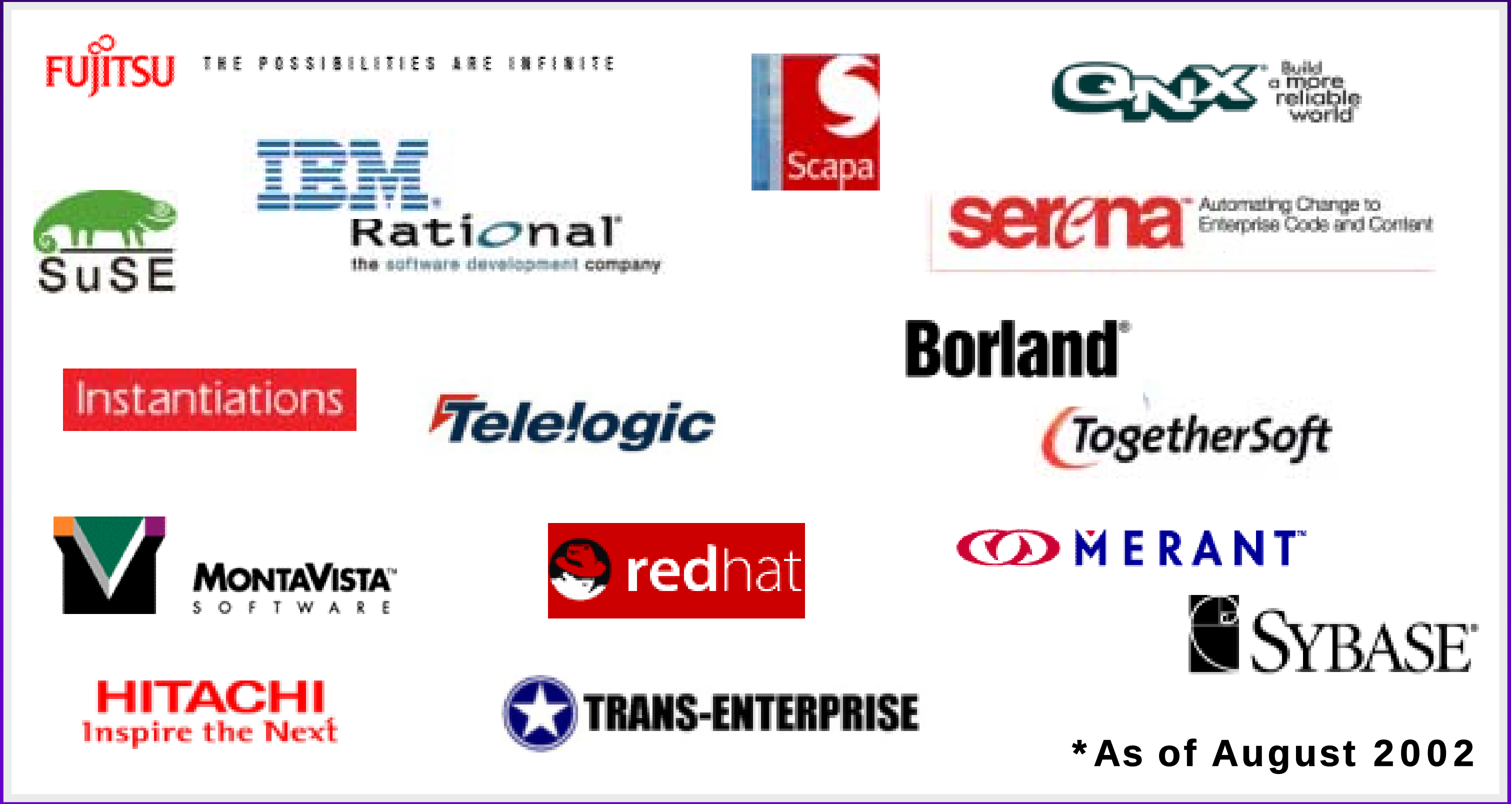
- Most Eclipse plug-ins are 100% pure Java
  - Freely port to new operating environment
  - Java2 and Eclipse APIs insulate plug-in from OS and window system
- Gating factor: porting SWT to native window system
- Just added in 2.1*
  - Mac OS X PowerPC – Carbon window system
  - QNX Neutrino RTOS Intel x86 - Photon window system
- Eclipse Platform also runs “headless”
  - Example: help engine running on server

* March 2003

eclipse

## Who's on Board?

- Wide range of software vendors on Eclipse board
- Represent various development tool markets


FUJITSU THE POSSIBILITIES ARE INFINITE
Scapa
QNX Build a more reliable world
IBM Rational the software development company
SuSE
serena Automating Change to Enterprise Code and Content
Borland
Instantiations
Telelogic
TogetherSoft
MONTAVISTA SOFTWARE
redhat
MERANT
SYBASE
HITACHI Inspire the Next
TRANS-ENTERPRISE
*As of August 2002

eclipse

## Who’s on Board?

- New members joined Sept.-Dec. 2002

ETRI
hp invent
PARASOFT
SAP
teamstudio
TimeSys
mks
slickedit.
OMG OBJECT MANAGEMENT GROUP
FLASHLINE
AltoWeb
catalyst SYSTEMS CORPORATION

## Who's Shipping on Eclipse?

- Commercial products*
  - 10 Technology – Visual PAD
  - Assisi – V4ALL Assisi GUI-Builder
  - Bocaloco – XMLBuddy
  - Borland – Together Edition for WebSphere Studio
  - Catalyst Systems – Openmake
  - Computer Associates – AllFusion Harvest Change Manager VCM
  - Ensemble Systems – Glider for Eclipse
  - Fujitsu – Interstage
  - Genuitec – EASIE Plug-ins
  - HP – OpenCall Media Platform OClet Development Environment
  - James Holmes – Struts Console
  - Instantiations – CodePro Studio

*** As of March 2003**

## Who’s Shipping on Eclipse?

- IBM uses Eclipse for
  - WebSphere® Studio Family
    - WebSphere Studio Homepage Builder
    - WebSphere Studio Site Developer (WSSD)
    - WebSphere Studio Application Developer (WSAD)
    - WebSphere Studio Application Developer Integration Edition (WSADIE)
    - WebSphere Studio Enterprise Developer (WSED)
    - WebSphere Studio Device Developer (WSDD)
    - WebSphere Development Studio for iSeries
  - Rational® XDE Professional: Java Platform Edition
  - Tivoli Monitoring Workbench

*** As of March 2003**

## Who's Shipping on Eclipse?

- Commercial products*
  - Interwoven – TeamSite repository
  - Intland – CodeBeamer
  - LegacyJ – PERCobol
  - Merant – PVCS Version Manager
  - MKS – Source Integrity Enterprise plug-in
  - Mobile Media – Grand-Rapid Browser
  - mvmsoft – Slime UML
  - No Magic Inc. – MagicDraw UML
  - Object Edge – Weblogic Plug-in
  - ObjectLearn – Lomboz
  - Omondo – EclipseUML
  - Ontogenics – hyperModel

*** As of March 2003**

## Who's Shipping on Eclipse?

- Commercial products*
  - Parasoft – Jtest
  - ProSyst – Eclipse OSGi Plug-in
  - QNX – QNX Momentics
  - Quest Software – JProbe integration
  - Serena Software – ChangeMan DS
  - SlickEdit – Visual SlickEdit Plug-in
  - Systinet – WASP Developer
  - THOUGHT – CocoBase Enterprise O/R
  - TimeSys – TimeStorm 2.0
  - xored – WebStudio IDE for PHP

* As of March 2003

## Who’s Building on Eclipse?

- Plus more than 40* other open source projects based on Eclipse
- See http://eclipse.org/community/plugins.html

* As of March 2003