
**AOT Lab**
**Dipartimento di Ingegneria dell'Informazione**
**Università degli Studi di Parma**

# Software Development Tools

## - The Eclipse Platform -

Alessandro Negri
negri@ce.unipr.it

# Eclipse

- “Eclipse, an extensible development platform and application frameworks for building software”


- Versione attuale: Eclipse 3.1.1 (30 Sett. 2005)
- Scaricabile dal sito ufficiale: www.eclipse.org


# La piattaforma attuale

Java Dev Tools

C/C++ Dev Tools

Test and Performance

Web Tools

Business Intelligence & Reporting

Frameworks

Modeling Frameworks

Graphical Frameworks

Tools Platform

Project Model

Rich Client Platform

Runtime (OSGi)

Generic Workbench

Update


# Il progetto Eclipse

- Eclipse è stato creato dai gruppi OTI e IBM già responsabili dei prodotti:
  - IBM VisualAge/Smalltalk (Smalltalk IDE)
  - IBM VisualAge/Java (Java IDE)
  - IBM VisualAge/Micro Edition (Java IDE)
- Inizialmente sviluppato da un gruppo di 40 programmatori impegnati a tempo pieno
- Migrato verso la licenza opensource di IBM e diviso in 3 progetti principali
  - Piattaforma
  - JDT (Java Development Tools)
  - PDE (Plug-in Development Environment)


# La licenza OpenSource IBM

- CPL (Common Public License)
- Licenza promossa da IBM
- La licenza CPL è stata approvata dall'Open Source Initiative (OSI) e permette agli sviluppatori di modificare il codice e riutilizzarlo all'interno di prodotti commerciali
- Recentemente anche Microsoft ha rilasciato alcune librerie con licenza CPL

# Obiettivo del Progetto

**“in many ways Eclipse is the Emacs for the 21st century“**

- Fornire una piattaforma aperta per lo sviluppo di applicativi che:
  - sia utilizzabile sul più alto numero di sistemi operativi
  - sia utilizzabile con un’interfaccia grafica o senza (GUI o non-GUI)
- Indipendente dal linguaggio di programmazione
  - Gestisca in modo completo vari tipi di linguaggi: HTML, Java, C/C++, XML, JSP, EJB …
- Facilitare l’integrazione di tool specifici
  - Possibilità di inserire nuovi tool al prodotto di base per venire incontro alle esigenze del singolo sviluppatore

# Architettura (1)

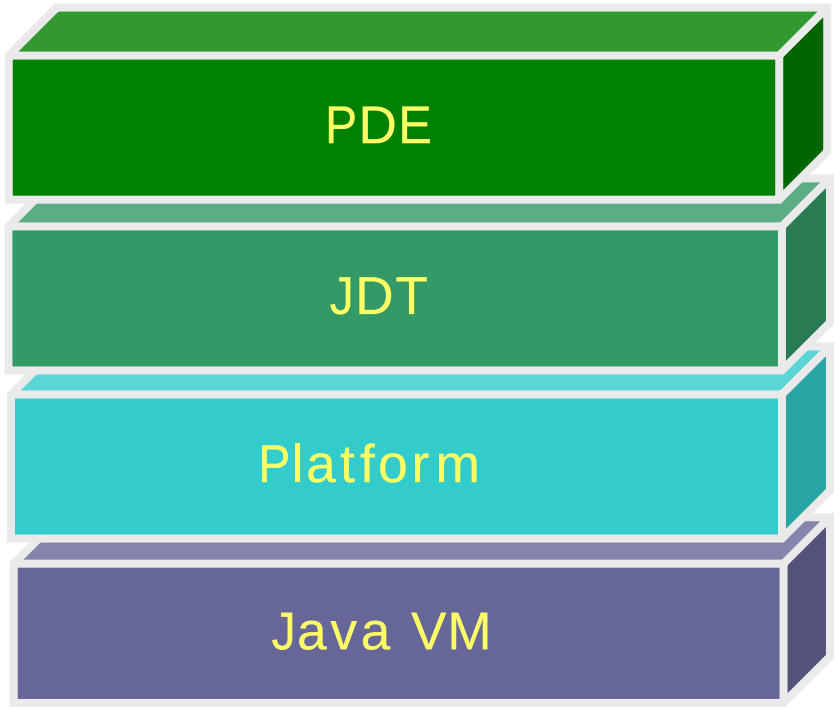
- Eclipse è una piattaforma universale per integrare tool di sviluppo
- Ha un'architettura aperta, estendibile, basata su plug-in

Plug-in Development Environment

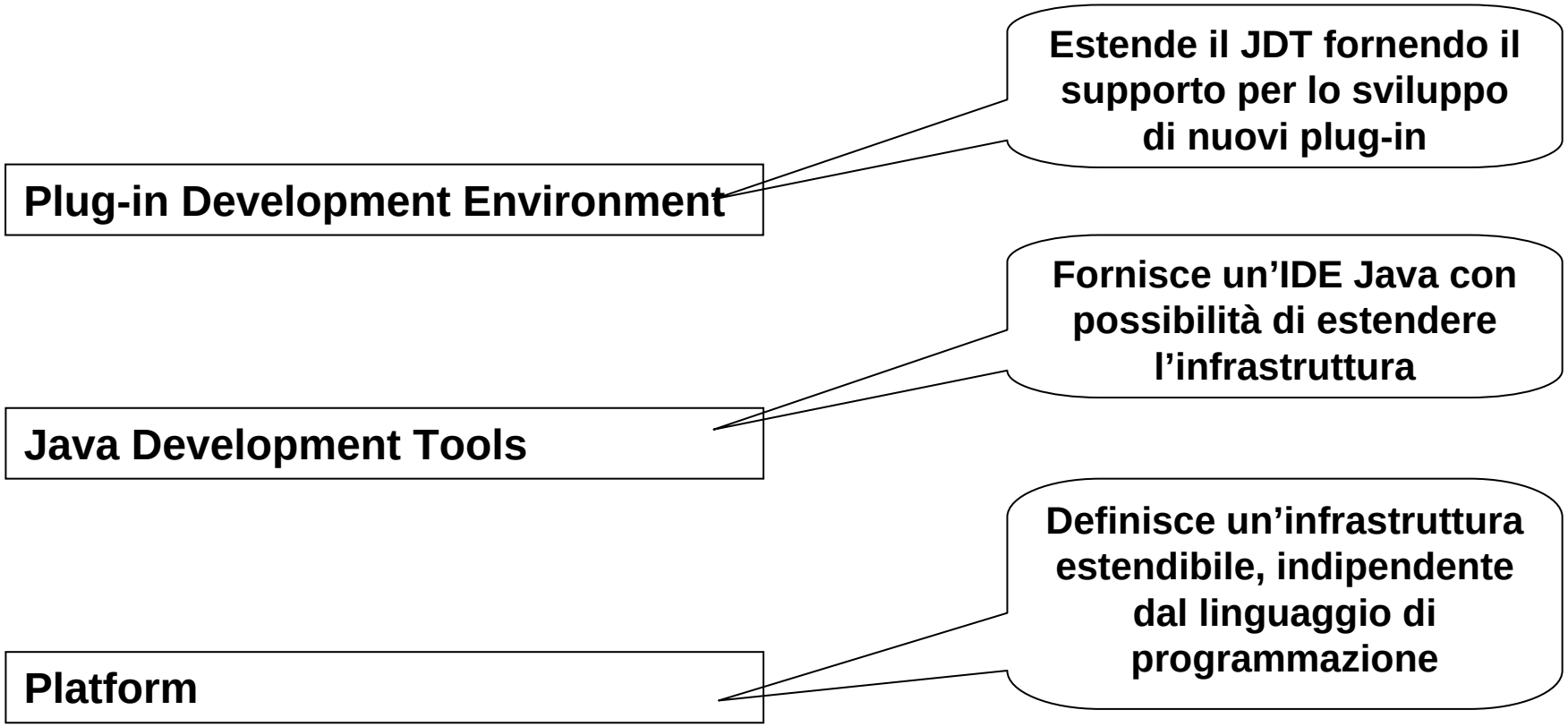
Java Development Tools

Eclipse Platform

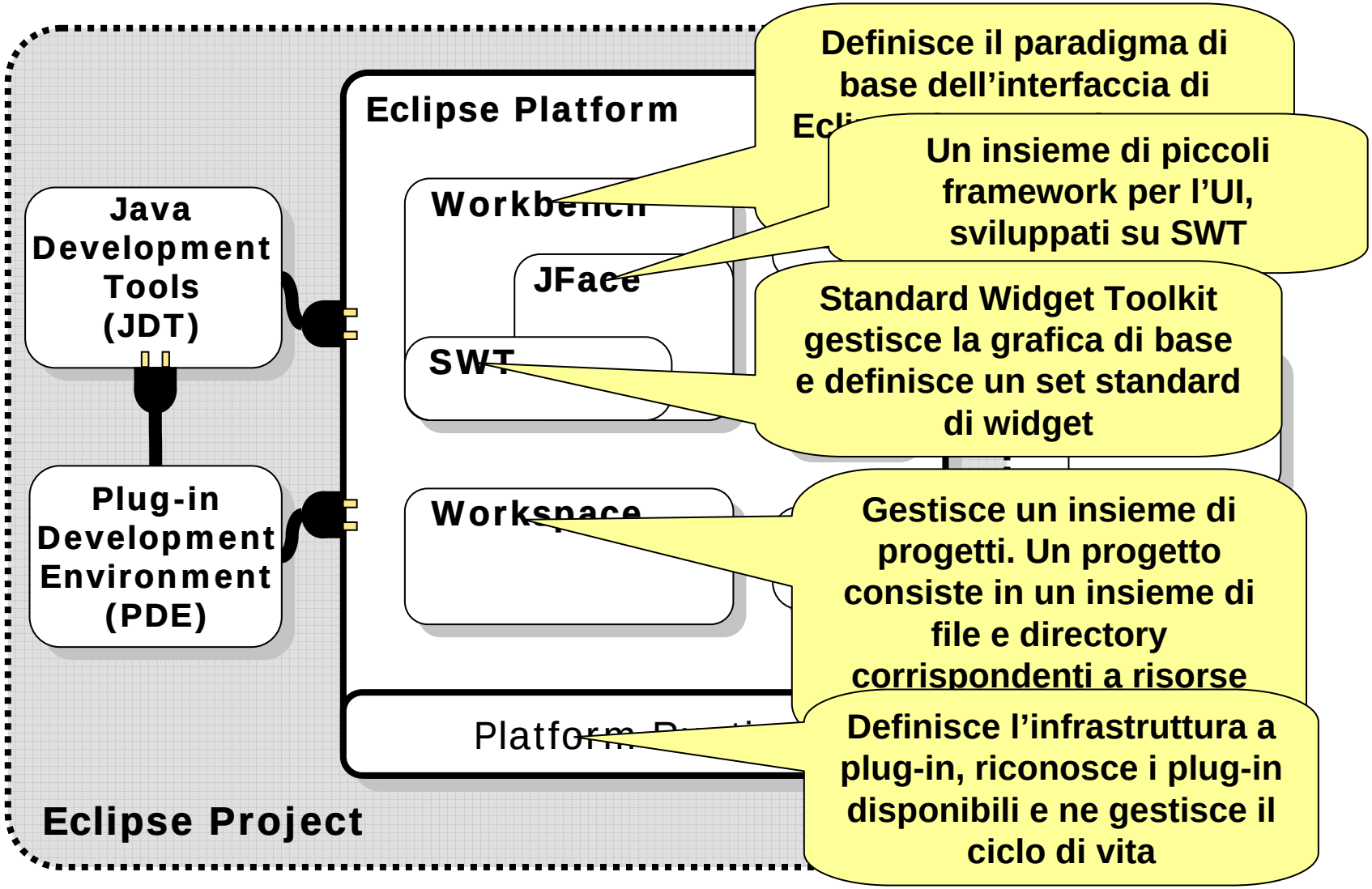
Standard Java2 Virtual Machine



# Architettura (2)

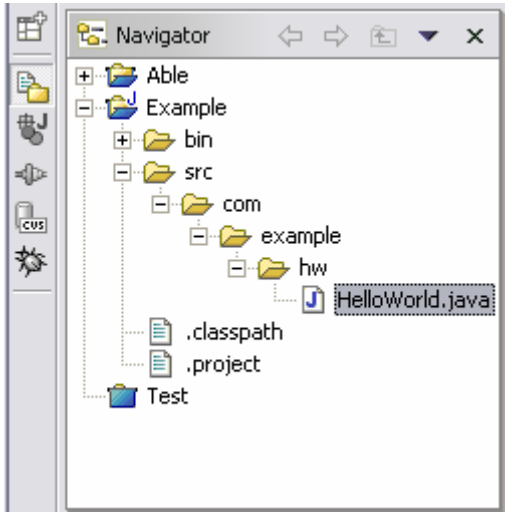


# Architettura (3)



# Workspace

- **Risorse**: permette di accedere a files, directory o progetti
- Il Workspace è in grado di gestire 1 o più progetti
- I progetti corrispondono a directory nel file system
- Vista Navigator
  - Albero di file e directory
- Vari tool permettono di leggere, creare, modificare, cancellare risorse nel workspace

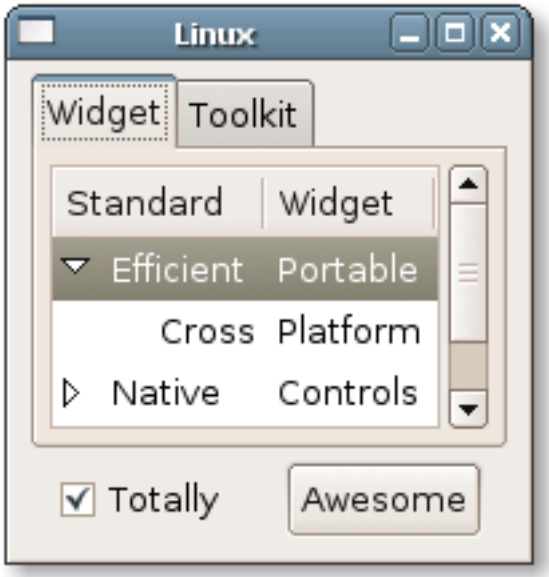



# Standard Widget Toolkit

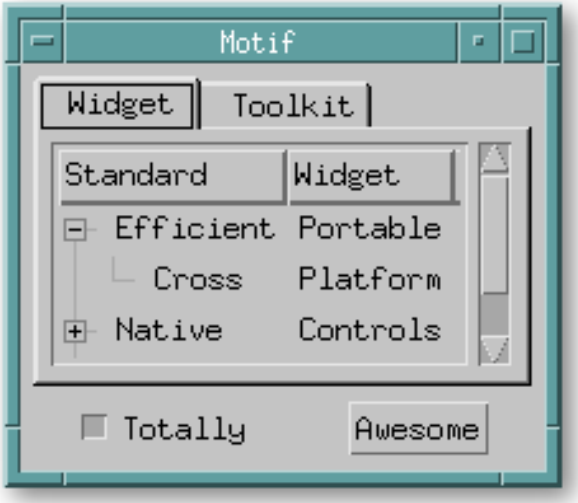
- Libreria contenente un insieme di strumenti grafici generici per la costruzione di GUI
  - bottoni, liste, testo, menu, alberi, testi formattati, …
- Ha API indipendenti dal sistema operativo, ma …
- … richiede componenti nativi per ogni sistema operativo
  - Ogni piattaforma operativa deve mettere a disposizione il proprio file swt.jar
    - In Windows è una .dll, in Linux un .so
- Ogni progetto grafico in Eclipse deve utilizzare SWT, ma …
- … SWT può essere utilizzato anche indipendentemente da Eclipse
    - Ha spesso dimostrato prestazioni migliori rispetto a Swing, perchè …???

# SWT - Esempi


Windows
Widget
Toolkit
Standard
Widget
Efficient
Portable
Cross
Platform
Native
Controls
Totally
Awesome


Linux
Widget
Toolkit
Standard
Widget
Efficient
Portable
Cross
Platform
Native
Controls
Totally
Awesome


Mac OS X
Widget
Toolkit
Standard
Widget
Efficient
Portable
Cross
Platform
Native
Controls
Totally
Awesome


Motif
Widget
Toolkit
Standard
Widget
Efficient
Portable
Cross
Platform
Native
Controls
Totally
Awesome


Photon
Widget
Toolkit
Standard
Widget
Efficient
Portable
Cross
Platform
Native
Controls
Totally
Awesome

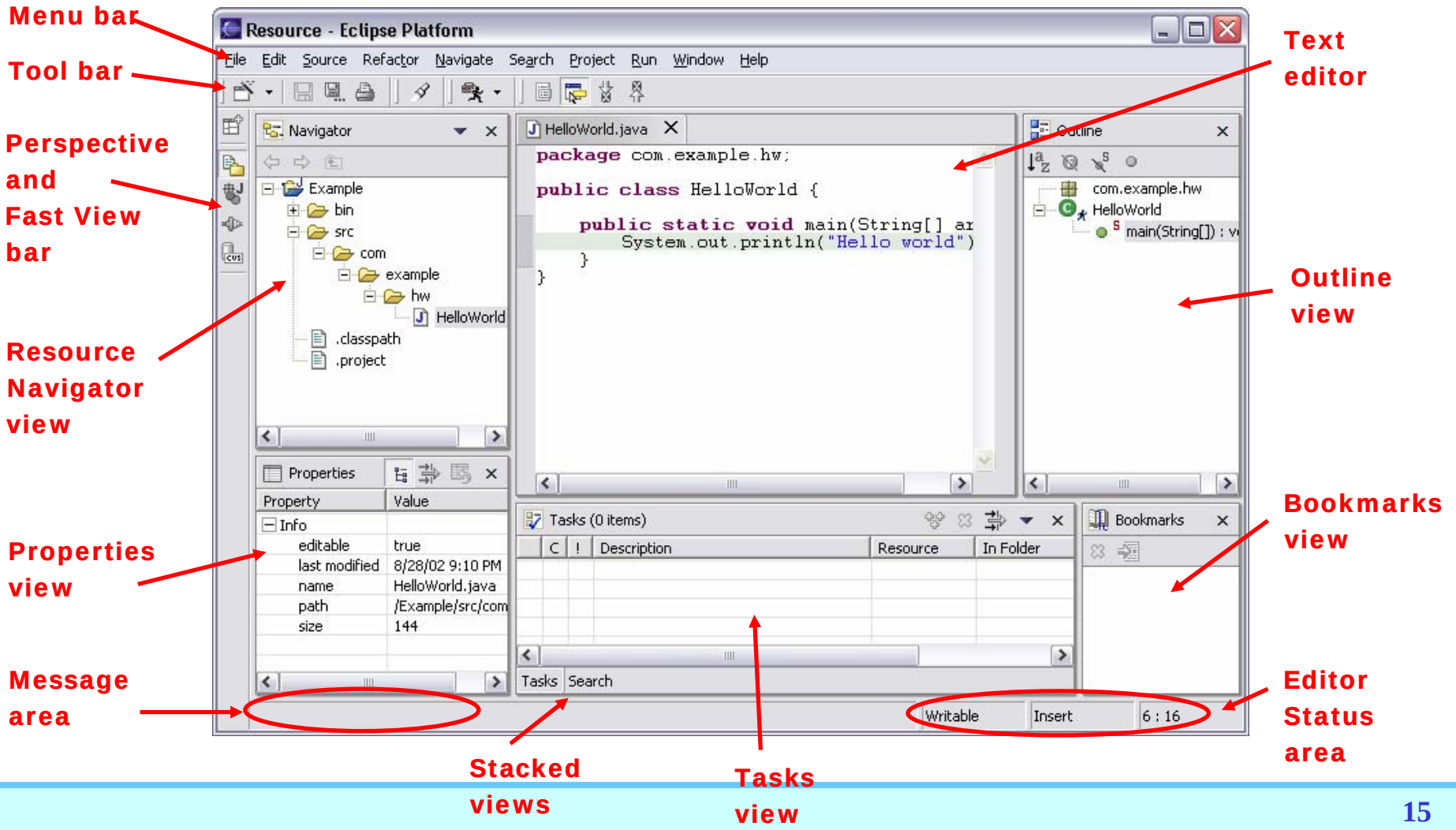

# JFace

- JFace è un framework grafico di più alto livello costruito su SWT
- Mette a disposizione una serie di classi per gestire le operazioni standard di un'interfaccia grafica
- Le API e l'implementazione sono indipendenti dal sistema operativo
- Permette di costruire in modo rapido:
  - Finestre di dialogo, preferenze, wizard, alberi, tabelle, liste, ecc…


# Workbench

- Il Workbench rappresenta l'interfaccia utente della piattaforma Eclipse
- Il Workbench usa JFace e vi aggiunge alcune funzionalità più complesse
- Componenti fondamentali
  - Editors
  - Views
  - Perspectives

# Workbench - Terminologia




# Editors

- Compaiono nell'area centrale del workbench
- Aggiungono azioni ai menu del workbench e alle toolbar
  - Open, edit, save, close lifecycle
- Il workbench fornisce
  - Extension Point per creare nuovi tipi di editors
    - Esempio: JDT crea un editor per gestire i file Java
  - API per editor di testo estendibili ed un framework


# Views

- Forniscono informazioni su di un oggetto
- Completano le funzionalità degli editors
  - Esempio: l'Outline View riassume il contenuto dell'editor
- Completano altre views
  - Esempio: la "Properties View" caratterizza una selezione
- Il workbench fornisce:
  - Extension Point per definire nuovi tipi di views
- La piattaforma Eclipse include un certo numero di views standard
  - Esempio: Resource Navigator, Outline, Properties, Tasks, Bookmarks, Search, …


# Perspectives

- Sono disposizioni di views ed editors
- Diverse perspectives possono essere utilizzate per gestire diverse operazioni
- Il workbench fornisce:
  - Extension Point per definire nuove perspectives
- La piattaforma Eclipse include un certo numero di perspectives standard
  - Esempio: Resource, Debug, …


# La Contribution

- Contribution Rule: “Everything is a contribution”
  - L’intera piattaforma Eclipse (escluso il Runtime Kernel) non ha nessuna funzionalità built-in, tutto è basato sulla Contribution
    - Eclipse + JDT ≈ 60 plugins
    - IBM Websphere IDE ≈ 600 plugins
  - Il Runtime Kernel è in grado di gestire migliaia di plugin
- Lazy Loading Rule: ” Contributions are only loaded when they are needed”
  - “Declaration vs Implementation”
  - La piattaforma mantiene le informazioni di tutti i plugin disponibili
    - Manifest file → plugin.xml
  - L’implementazione viene caricata alla prima richiesta di accesso
    - PLUGIN_XXX.jar

# Plug-in (1)

- Plug-in: la più piccola unità funzionale in Eclipse

  **“a piece of behavior that is outside the run-time kernel”**

  **(Contributing to Eclipse: Principles, Patterns and Plug-Ins by E.Gamma,K.Beck)**

  - E’ un componente strutturato che descrive se stesso al sistema utilizzando un manifest file (plugin.xml)
  - Può racchiudere numerose funzionalità (es. HTML editor) oppure una semplice funzione (es. l’azione per creare un file zip)

- Un plugin si concretizza fisicamente in una directory con all’interno:
  - plugin.xml: il manifest, una descrizione della Contribution
  - resources: risorse utili al plugin (es. icone) - OPZIONALE
  - Java code: strutturato in file .jar - OPZIONALE


# Plug-in (2)

- Ha un identificatore univoco – plugin ID
- Porta una Contribution alla piattaforma
  - Descritta tramite il manifest
  - Implementata (quando serve) attraverso le librerie del plug-in
  - Può estendere un aspetto specifico della piattaforma (**Extensions**), per esempio:
    - Introdurre un nuovo tipo di risorsa o estendere un tipo di risorsa già presente
  - Può contribuire con nuovi punti di estensione (**Extension Points**), per esempio:
    - Validatori di documenti XML
- Può dipendere da altri plug-in
  - Dipendenza esplicita nella descrizione e gestita in automatico dalla piattaforma

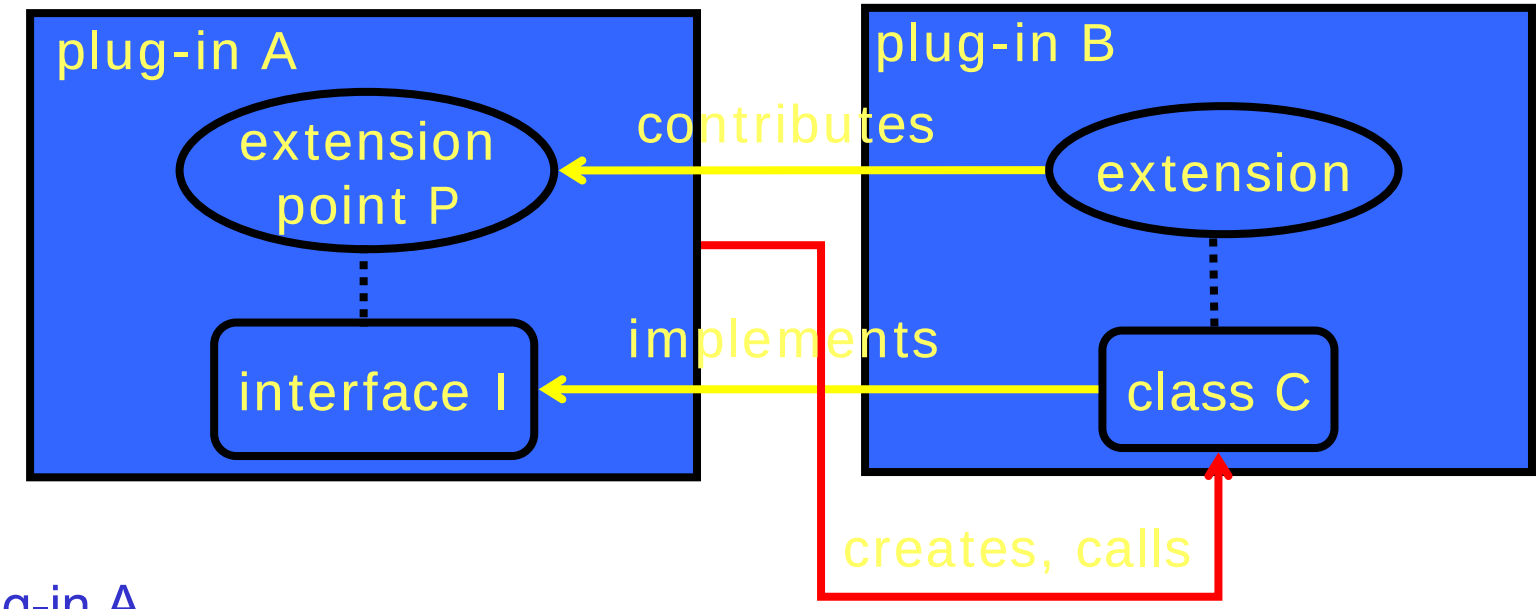

# Architettura del Plug-in (1)

- Ogni plug-in:
  - Contribuisce ad 1 o più Extension Point
  - Dichiara 1 o più Extension Point (OPZIONALE)
  - Dipende da un set di altri plug-in (OPZIONALE)
  - Contiene librerie di codice Java o altri file (OPZIONALE)
  - Risiede in una sottodirectory a lui dedicata
- Plug-in Manifest
  - Dichiara tutte le “Contributions”
  - Implementa interfacce o fornisce API
  - plugin.xml: descrive le proprietà del plug-in

# Extension e Extension Point

- Un Extension Point è un punto di aggancio per i contributi dei plugin
  - "Qualcuno" (un altro plugin) lo ha dichiarato
    - org.eclipse.ui
  - E' identificato univocamente nella piattaforma (Id)
    - org.eclipse.ui.ActionSets
  - Ha uno schema che regolamenta le informazioni dichiarative che i contributor devono fornire
  - Può contenere un set di Action, ecc…
  - Definisce un'interfaccia che la classe del contributor deve implementare
- Una Extension è la dichiarazione dalla Contribution
- Chi ha dichiarato l'Extension Point in genere interroga il registry per sapere chi offre delle Extension
  - Es. il workbench verifica chi fornisce una Extension per gli ActionSets per costruire l'interfaccia utente dell'IDE

# Architettura del Plug-in (2)

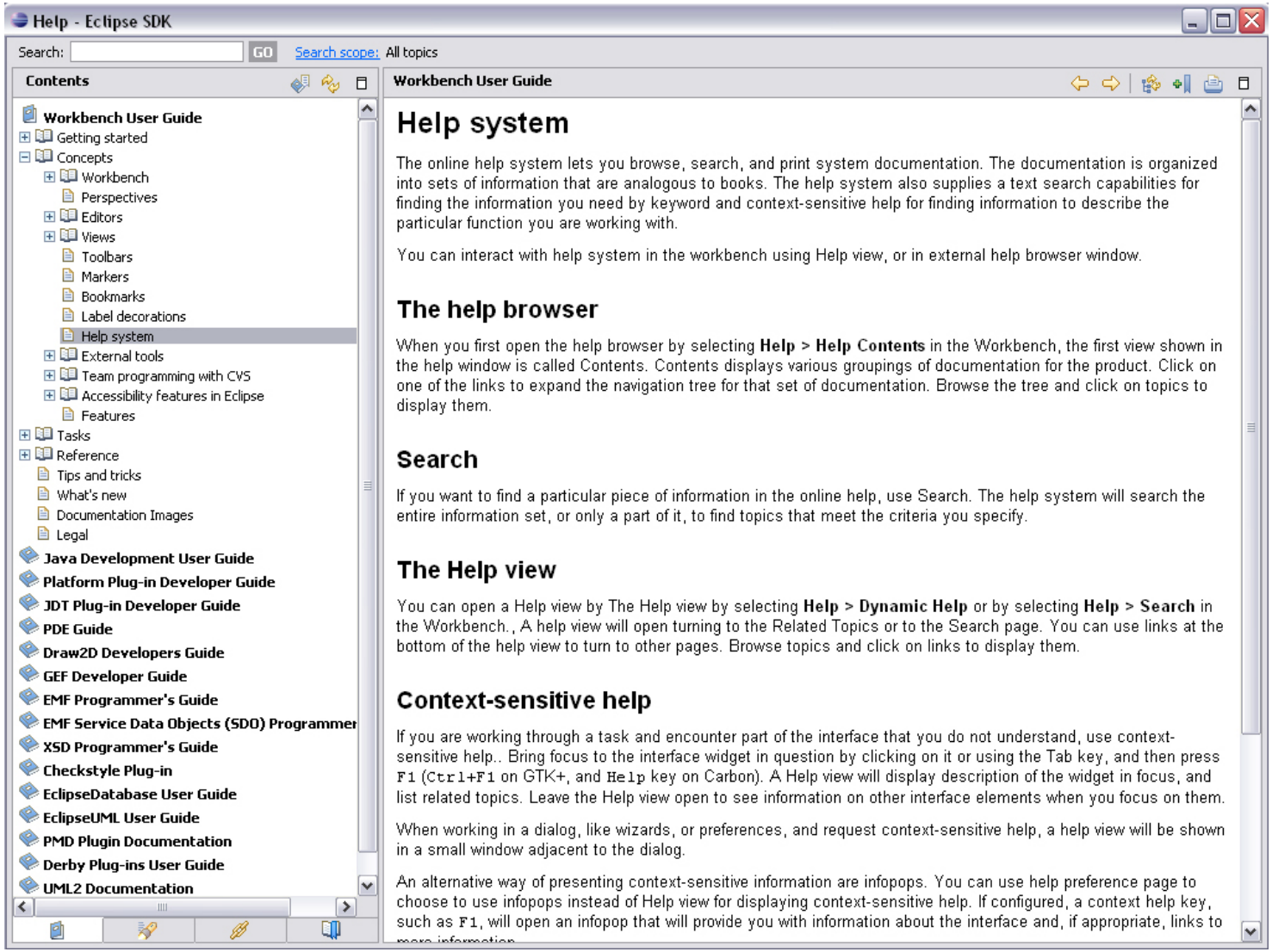



- Plug-in A
  - Dichiara un Extension Point P
  - Dichiara un'interfaccia I per il punto P
- Plug-in B
  - Implementa l'interfaccia I con la sua classe C
  - Inserisce la classe C come Contribution all'Extension Point P
- Il Plug-in A instanzia la classe C e richiama i metodi dell'interfaccia I


# Eclipse Platform Runtime

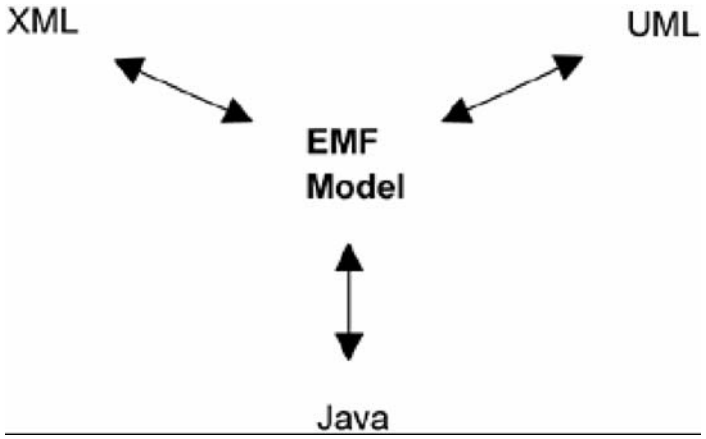
- Eclipse Platform Runtime è un micro-kernel
  - Tutte le funzionalità sono implementate in plug-ins
- Eclipse Platform Runtime gestisce l'inizializzazione e la creazione dell'ambiente di lavoro
  - Ricerca tutti i plug-ins installati sul disco
  - Unisce tutte le Extensions con i rispettivi Extension Points
  - Costruisce un registro globale dei plug-in
  - Crea una copia in cache del registro per l'esecuzione successiva

# Eclipse Help



# Eclipse Modeling Framework

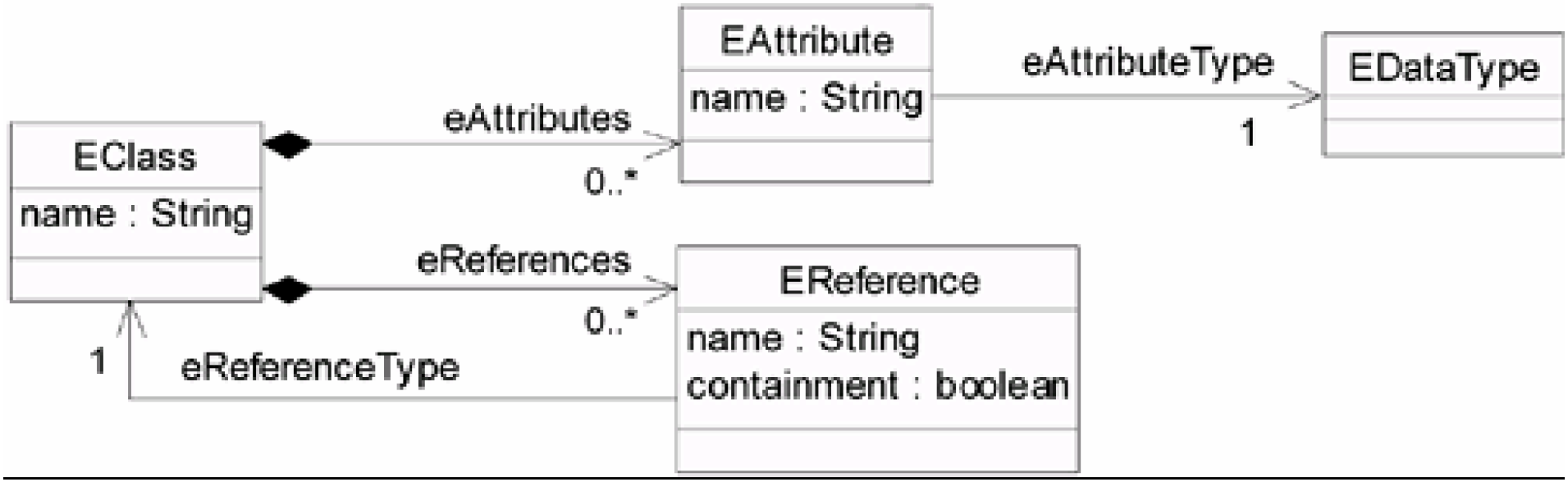
**“To model or to program, that is not the question”**




- E’ un framework per la generazione di codice Java
- Concetti fondamentali
  - Portable design
  - Dynamic integration
  - Flexible development and tools integration using metadata
  - Better use of design patterns and templates

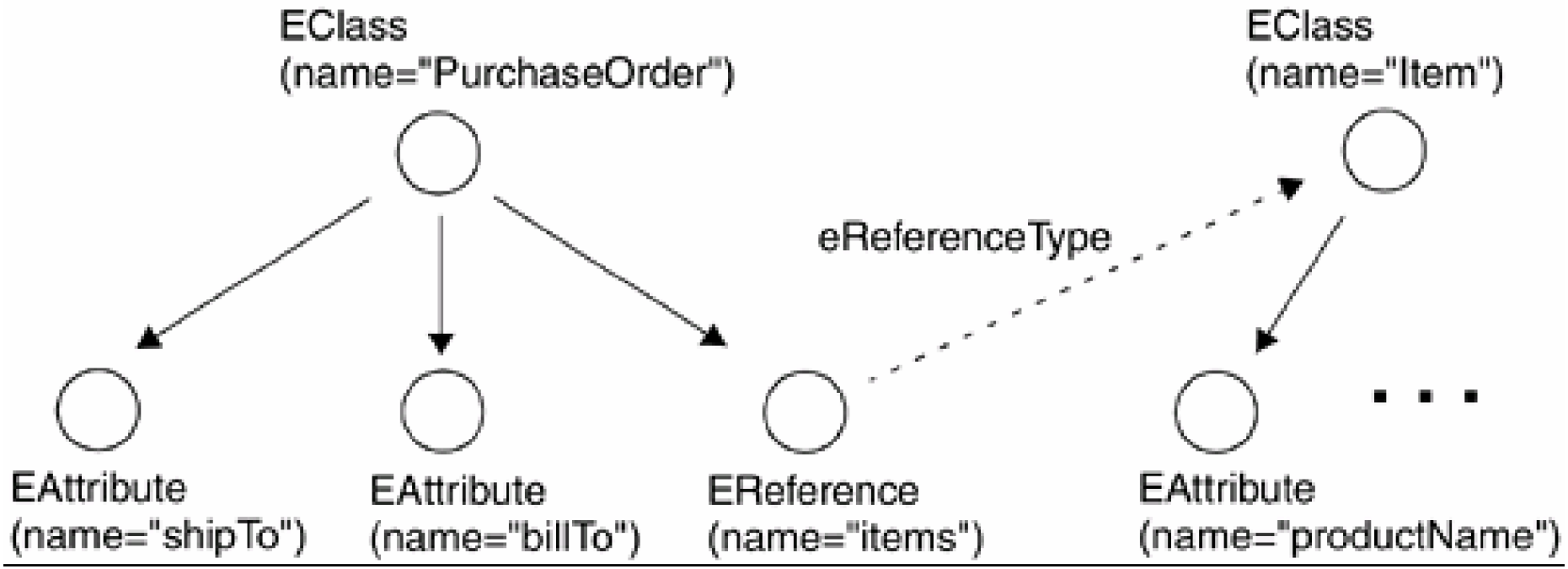
# EMF – Ecore (1)

- Il metamodello utilizzato da EMF per rappresentare modelli è chiamato Ecore
- Un sottoinsieme di un Ecore model



# EMF – Ecore (2)

- Esempio di un ordine di acquisto



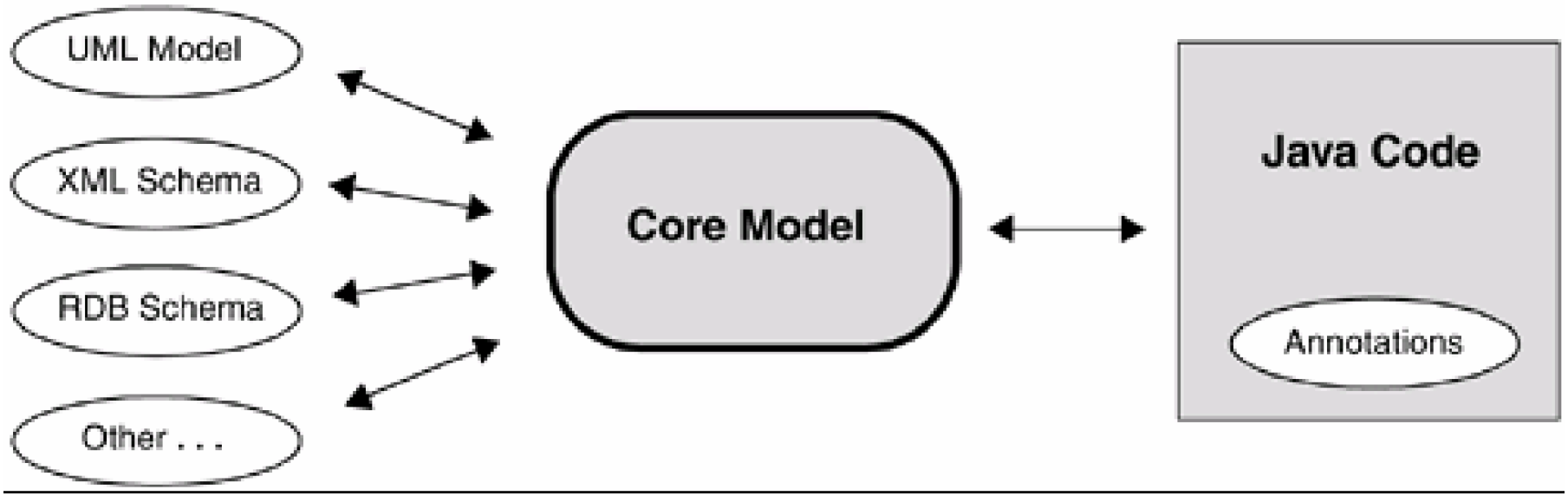

- Quando instanziamo le classi definite in un metamodello Ecore per definire un modello della nostra applicazione, stiamo creando un Core model → rappresentazione XMI

# EMF - Core

```
<?xml version="1.0" encoding="ASCII"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI“
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
    name="po" nsURI="http:///com/example/po.ecore"
    nsPrefix="com.example.po">
<eClassifiers xsi:type="ecore:EClass" name="PurchaseOrder">
    <eReferences name="items“ eType="#//Item" upperBound="-1" containment="true"/>
    <eAttributes name="shipTo“ eType="ecore:EDataType
    http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eAttributes name="billTo“ eType="ecore:EDataType
    http://www.eclipse.org/emf/2002/Ecore#//EString"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Item">
    <eAttributes name="productName“ eType="ecore:EDataType
    http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eAttributes name="quantity“ eType="ecore:EDataType
    http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eAttributes name="price“ eType="ecore:EDataType
    http://www.eclipse.org/emf/2002/Ecore#//EFloat"/>
</eClassifiers>
</ecore:EPackage>
```

- **Gli elementi XML corrispondono esattamente alle istanze del metamodello Ecore**

# EMF – Generazione del Codice Java



# Graphical Editing Framework


- Permette di sviluppare rappresentazioni grafiche di modelli precostruiti
- Utilizza:
  - Draw2D: standard framework per il disegno in 2D basato su SWT
  - Usa l'architettura MVC (Model-View-Controller)

# GEF e EMF

- GEF e EMF sono stati progettati per essere utilizzati insieme per la generazione di editor grafici
- Vantaggi
  - EMF permette di creare modelli consistenti, efficienti e facilmente modificabili
  - Tramite MVC c'è perfetta integrazione fra EMF e GEF
  - Viene mantenuto un legame reciproco fra modello ed implementazione
  - Si possono creare editor standard in modo automatico usando le EMF API
- Passi principali
  - Crea il tuo modello
  - Definisci la view
  - Scrivi le tue EditParts ed i Controllers
  - Metti insieme il tutto


# Java Development Tools

- Composto da una serie di plug-in per la creazione di un'IDE per la gestione di progetti Java
- Fornisce API per essere esteso a piacere
- Componenti principali:
  - **Core**: definisce l'infrastruttura non grafica
    - Java builder, search, refactoring, tree navigation
  - **UI**: workbench specifico per Java
    - Packages View, Type Hierarchy View, Java Outline View, Wizards per creare elementi Java, Java Editor
  - **Debug**: fornisce strumenti per il debugging
    - JVM introspection
  - **APT**: supporto per le annotazioni introdotte nella versione Java 5.0

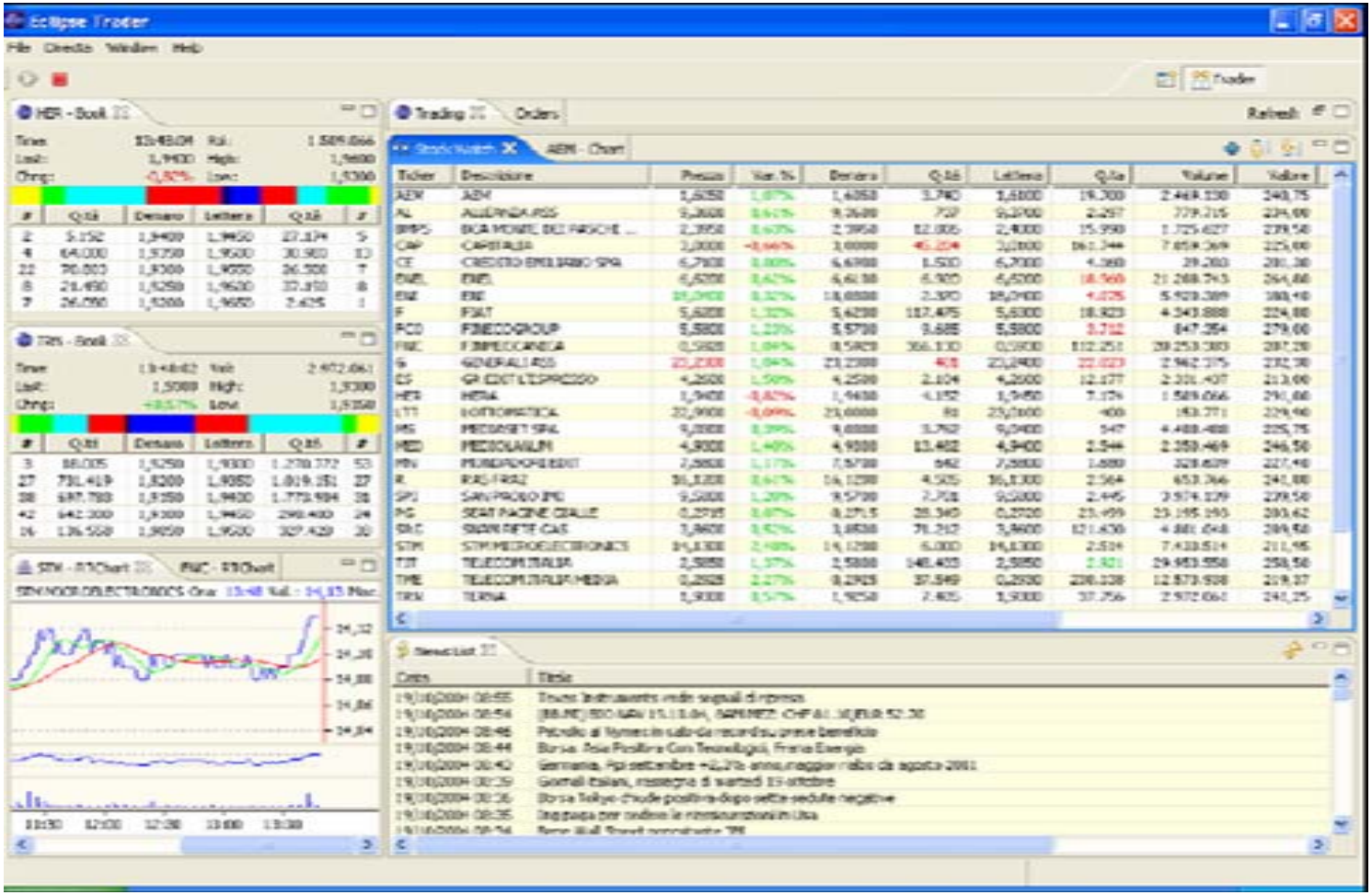

# Rich Client Platform

- “The Rich Client Platform (RCP) is an exciting new way to build Java applications that can compete with native applications on any platform.”
- Storia:
  - Eclipse 2.1: degli hacker nella comunità di Eclipse cominciarono a sviluppare delle applicazioni non-IDE
  - Eclipse 3.0: gli aspetti relativi all’IDE sono stati estratti dal workbench, dando origine a RCP

# RCP – Esempi (1)


**IBM Lotus Workplace**

# RCP – Esempi (2)



**Eclipse Trader**


# RCP - Prestazioni

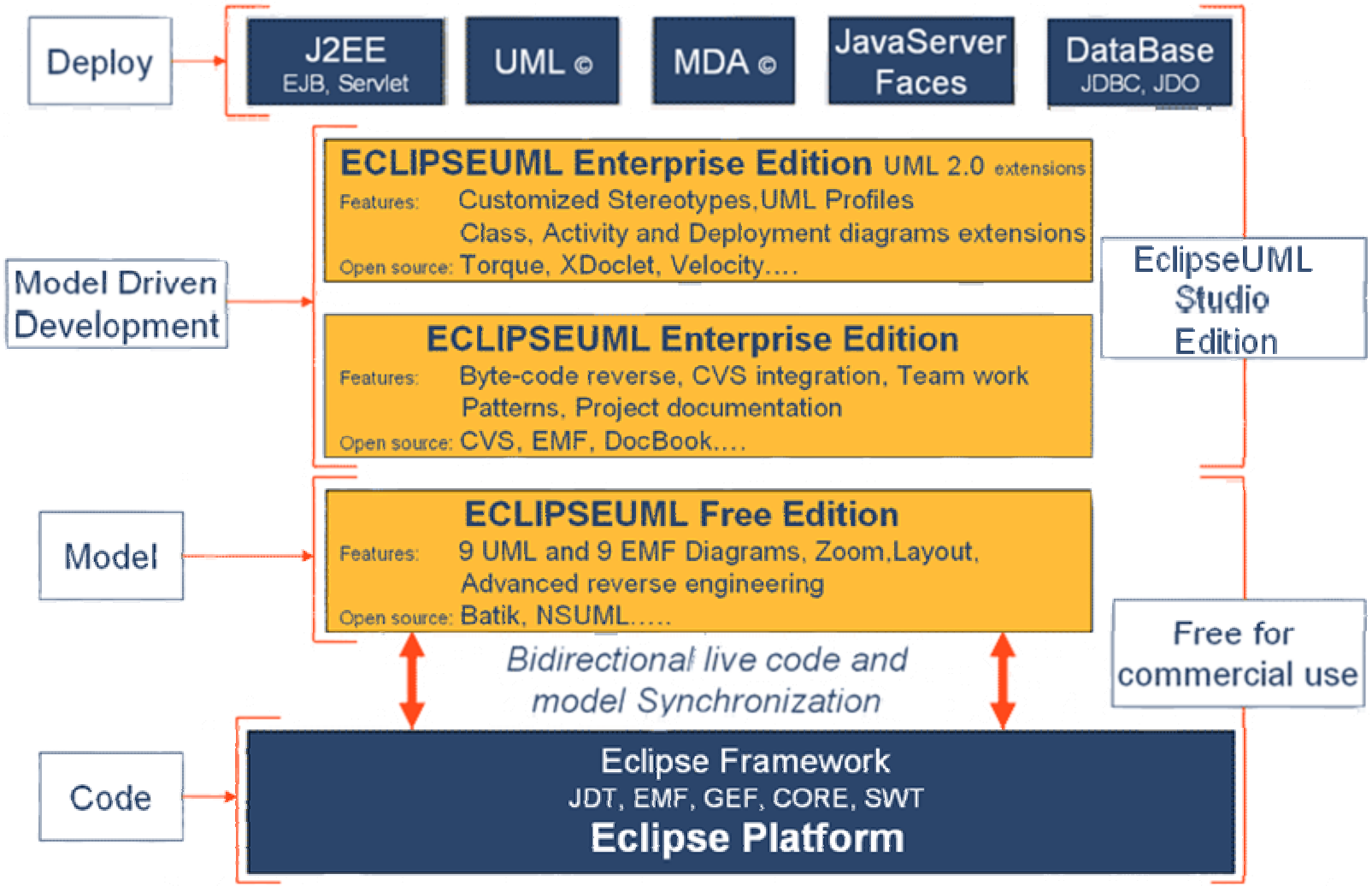
- Un'applicazione RCP è un'applicazione di Eclipse nella quale il workbench è eseguito nel main event loop
- Alcuni numeri:
  - Dimensione download ~ 5MB
  - Dimensione decompresso ~ 6MB
  - Dimensione VM di Hello World (non Eclipse) ~ 8MB
  - Dimensione VM di Hello World RCP ~ 9MB
  - Numero di plug-in inseriti nel bianrio di RCP: 10


# Omondo EclipseUML Plug-in

- Tool visuale per la modellizzazione UML, integrato in modo nativo con Eclipse e CVS


- Gratuito solo in versione personale
  - Versione attuale: 2.1.0
- Sviluppato in Java, disponibile al sito: ***http://www.omondo.org***
- Eclipse 100% native API
  - Permette di integrare il plug-in in altre applicazioni

# Omondo – Architettura Prodotti




# Omondo – Free & Studio Edition

Features

| MODEL | EclipseUML Free Edition | EclipseUML Studio Edition |
|---|---|---|
| UML 2.0 | ✓ | ✓ |
| UML 1.4 | ✓ | ✓ |
| JDK 1.5 | ✓ | ✓ |
| Eclipse 3.1 | ✓ | ✓ |
| Eclipse 3.0.2 | ✓ | ✓ |
| Eclipse 2.1.3 | ✓ | ✓ |
| GEF | ✓ | ✓ |
| EMF | ✓ | ✓ |
| Print function | ✓ | ✓ |
| Graphical Export functions | SVG | ✓ |
| Auto-Layout | ✓ | ✓ |
| Zoom | ✓ | ✓ |
| All UML Diagrams | ✓ | ✓ |
| UML 2.0 Import/Export | | ✓ |
| Database | | ✓ |
| J2EE | | ✓ |
| Team solution | | ✓ |
| CVS | | ✓ |
| Integrates with any SCC-compliant tool | | ✓ |
| UML Profiles | | ✓ |
| Open API | | ✓ |

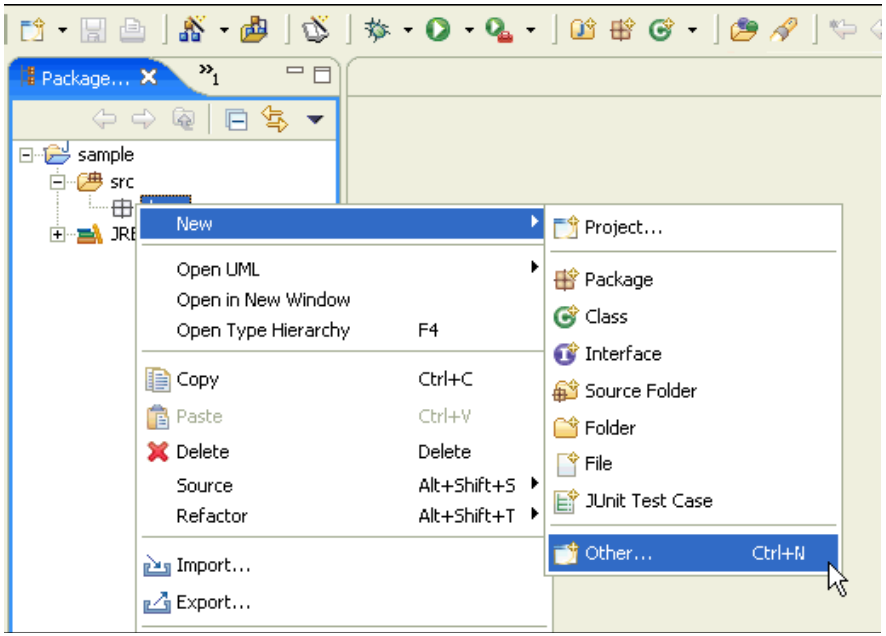

# Omondo - Funzionalità

- Supporto per gestire la creazione di diagrammi UML 2.0: disegno, modifica, stampa, zoom …
- Reverse Engeneering
- Sincronizzazione automatica fra i modelli ed il codice corrispondente
- Integrazione automatica con EMF e GEF
- Nella versione EclipseUML Studio
  - Graphical Database Diagram
  - EclipseJ2ee
  - CVS
  - Team solution

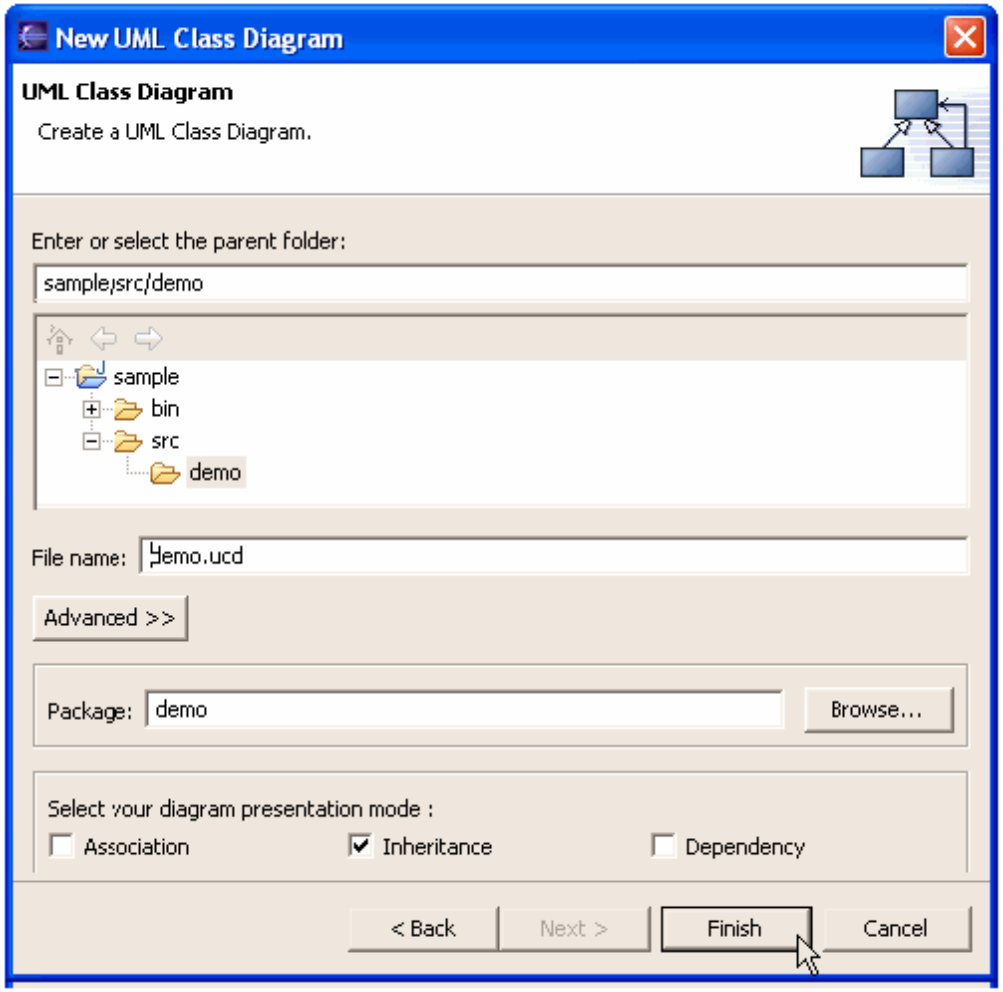

# Omondo - Diagrammi Supportati

- Class Diagram
- Sequence Diagram
- Use Case Diagram
- State Diagram
- Activity Diagram
- Object Diagram
- Component Diagram
- Deployment Diagram
- Robustness Diagram

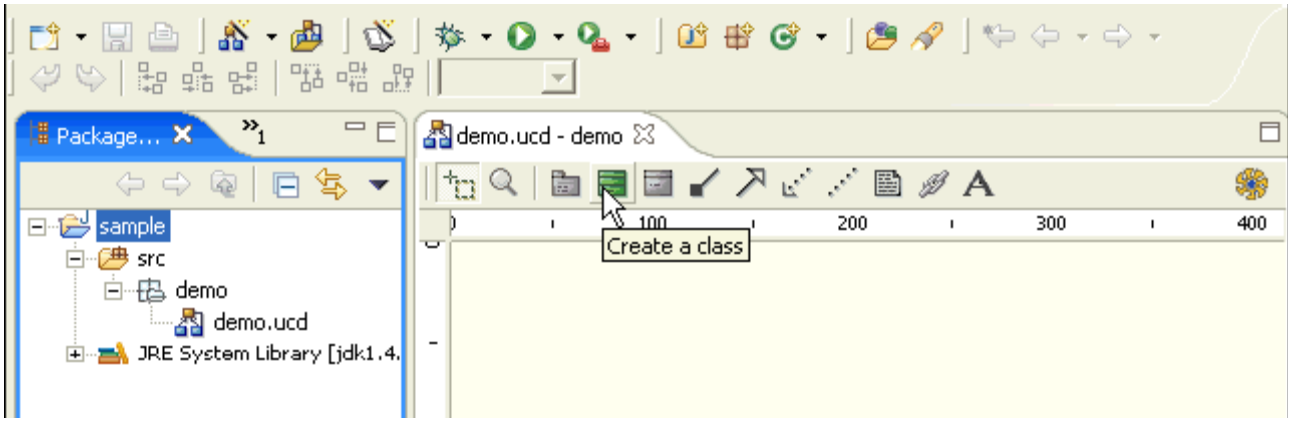
# Omondo – Class Diagram



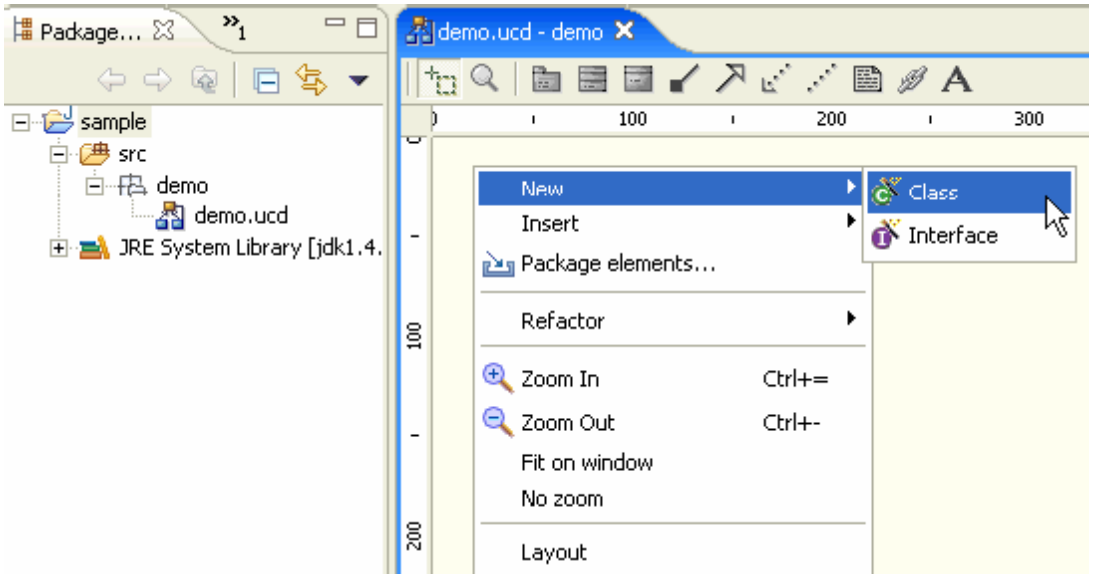

## Creare un class diagram



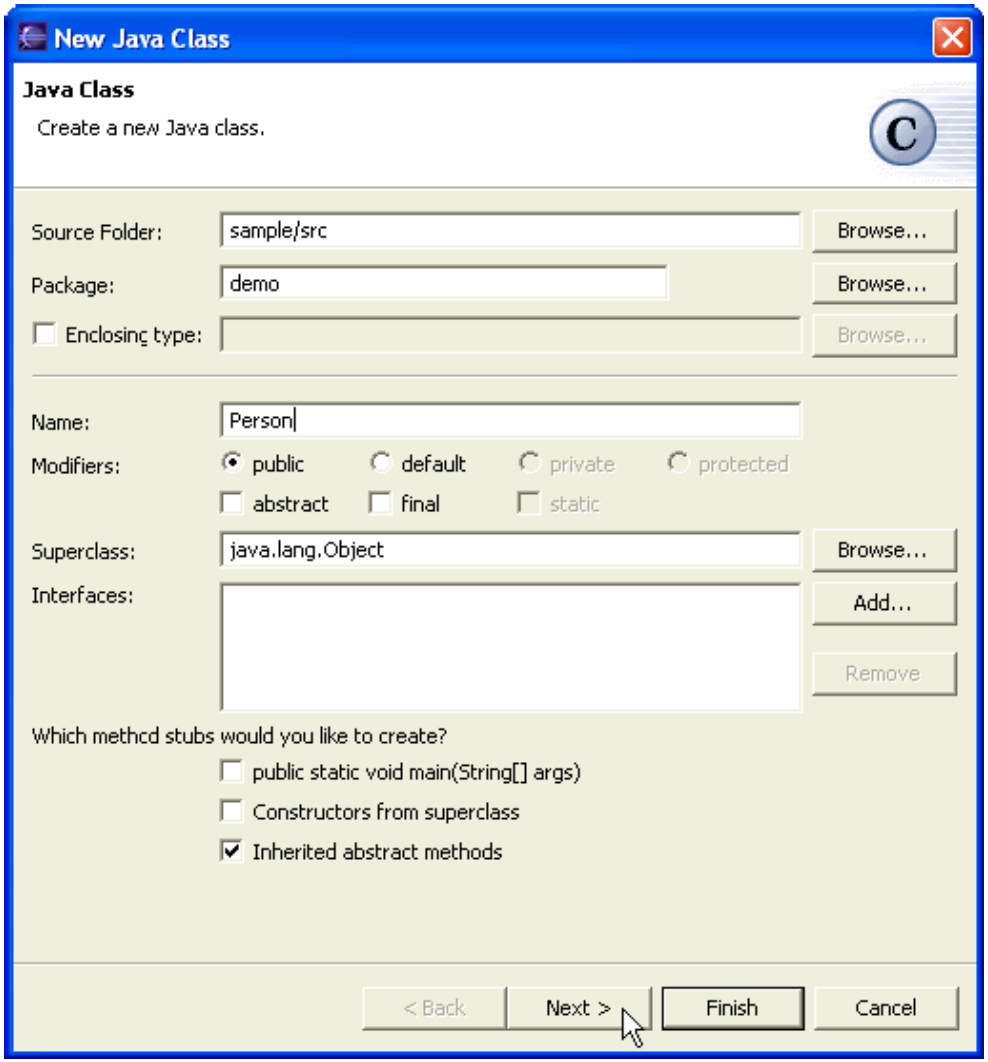
# Omondo – Class Diagram

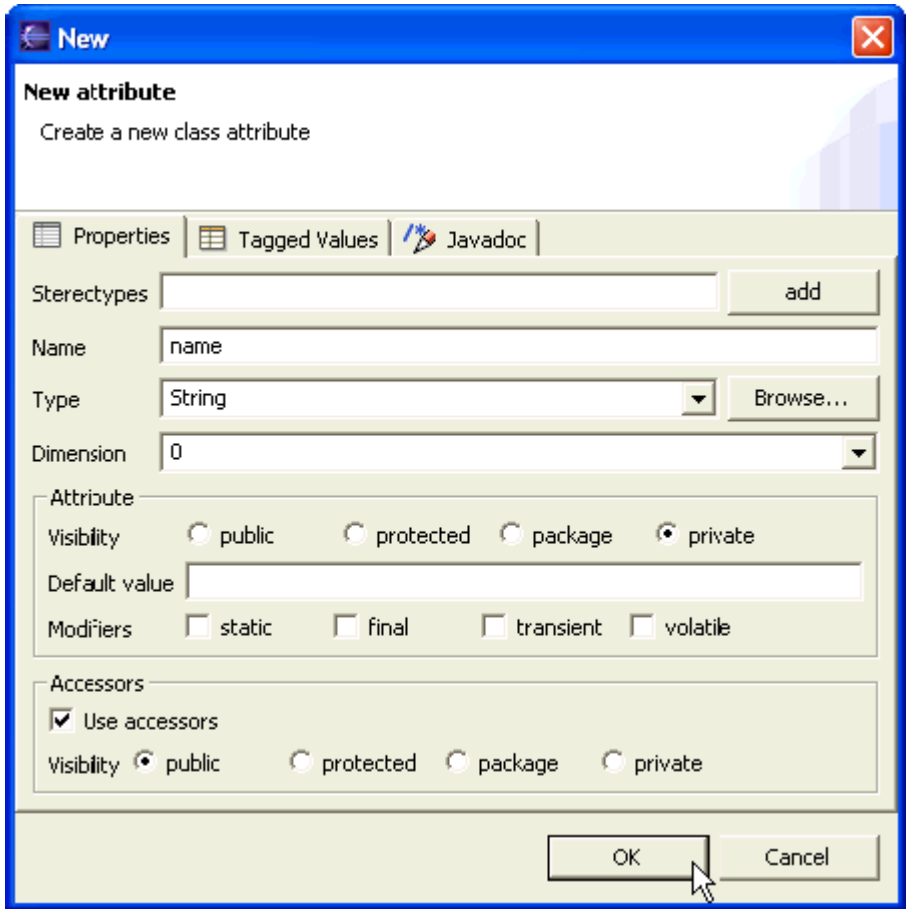



## Creare una classe

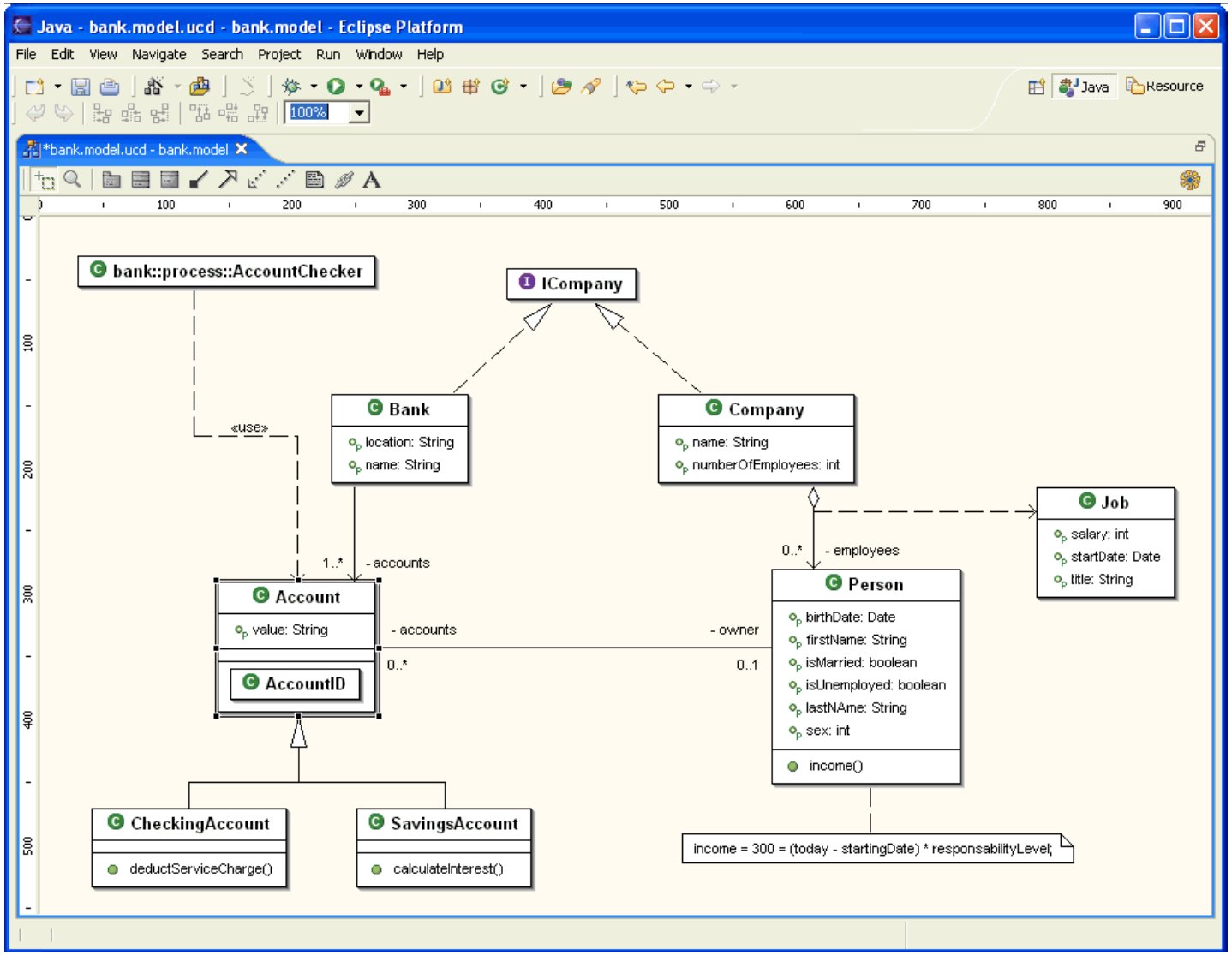


# Omondo – Class Diagram

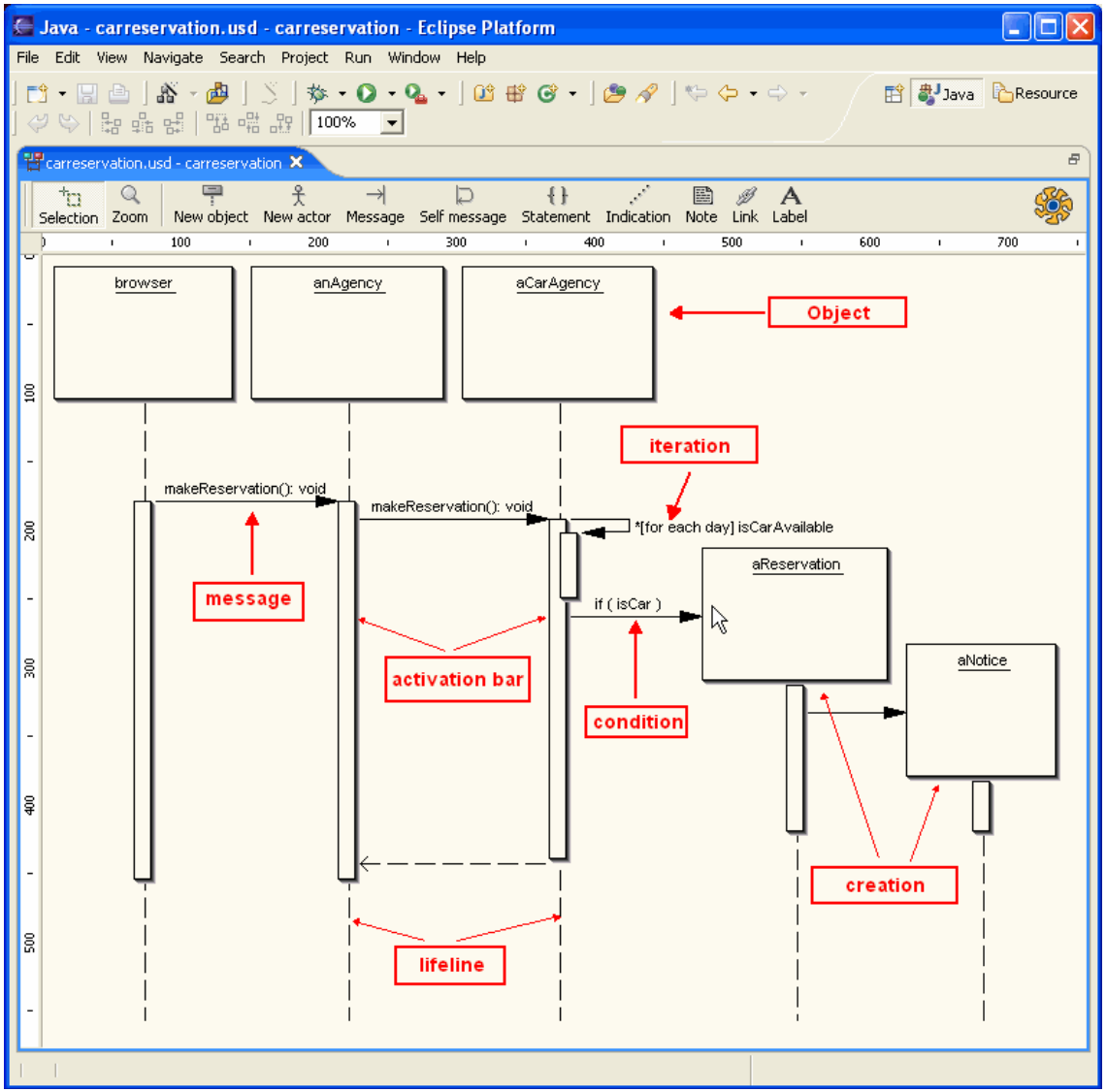


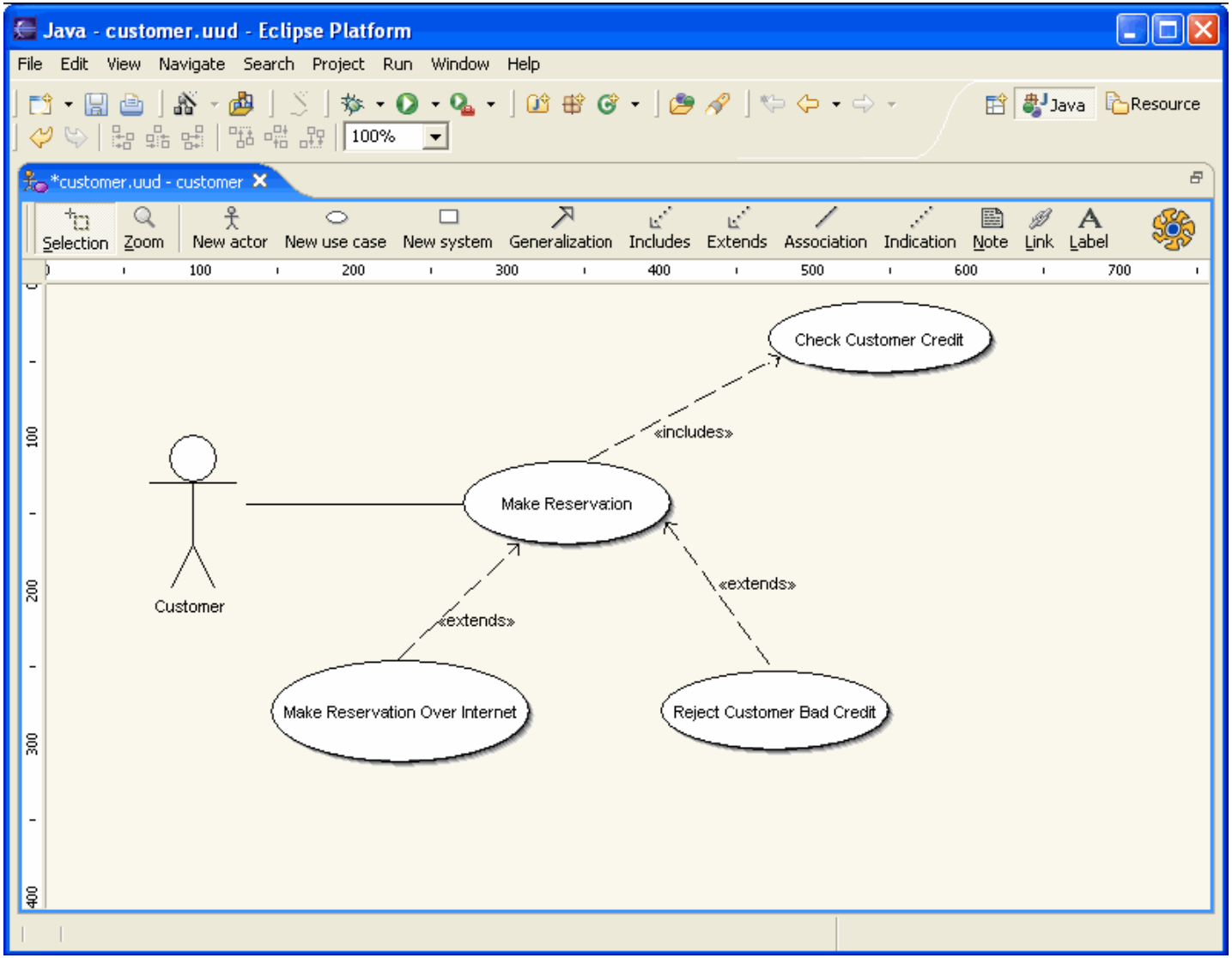
# Omondo – Class Diagram



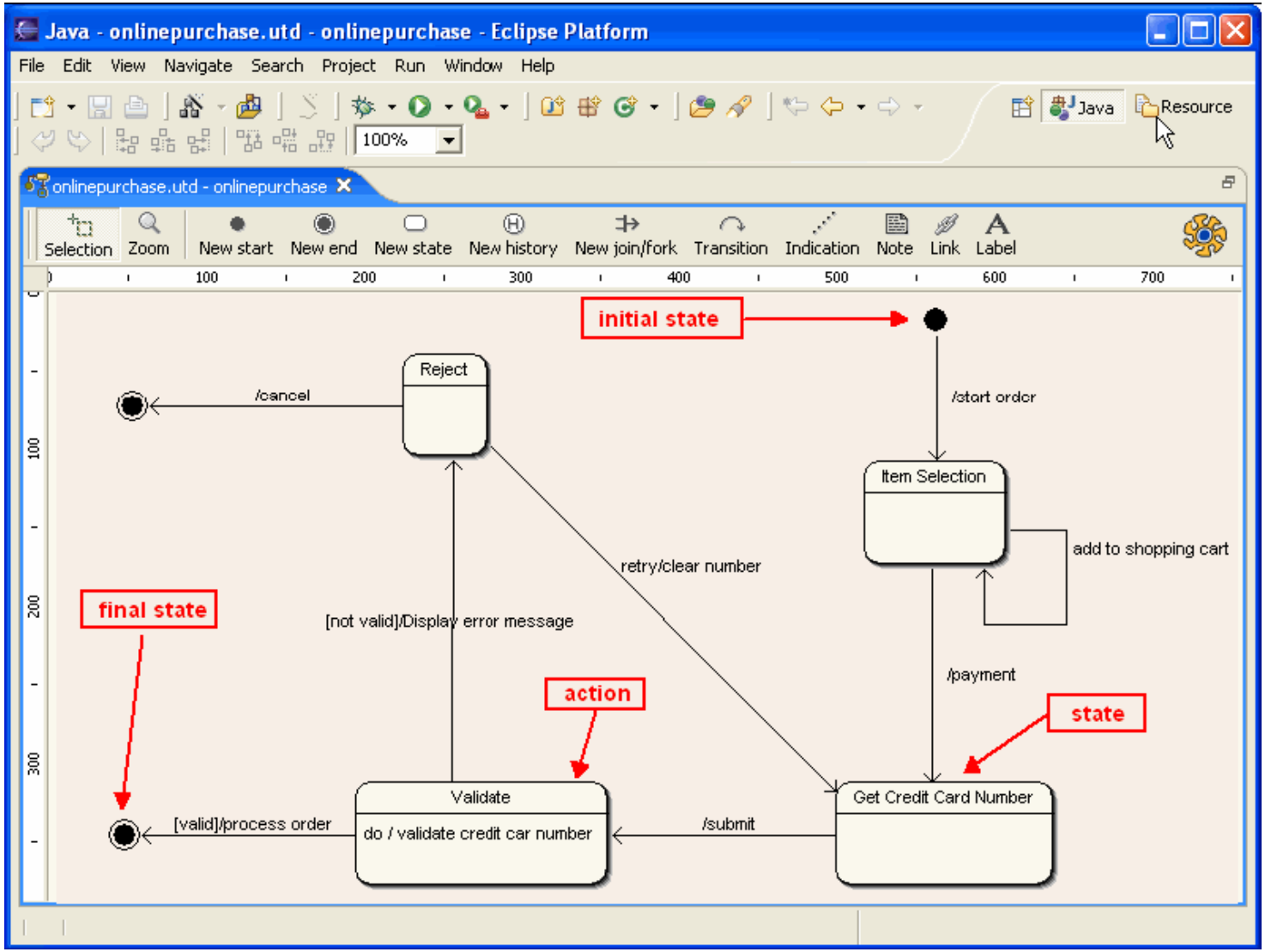
# Omondo – Sequence Diagram



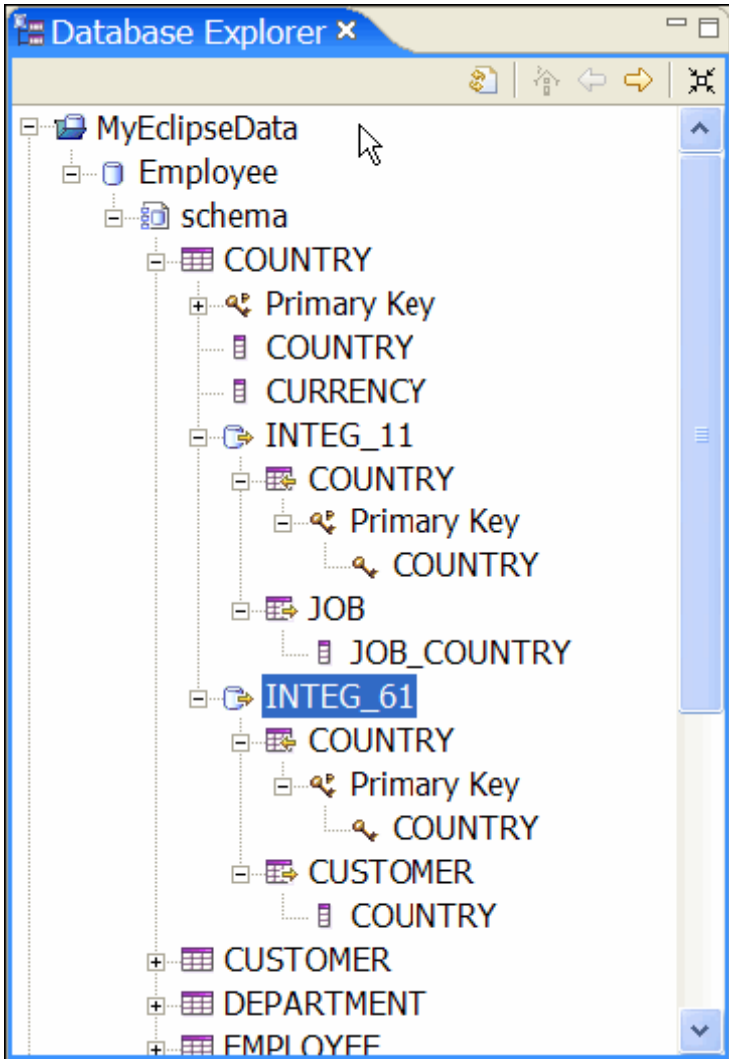
# Omondo – Use Case Diagram



# Omondo – State Diagram

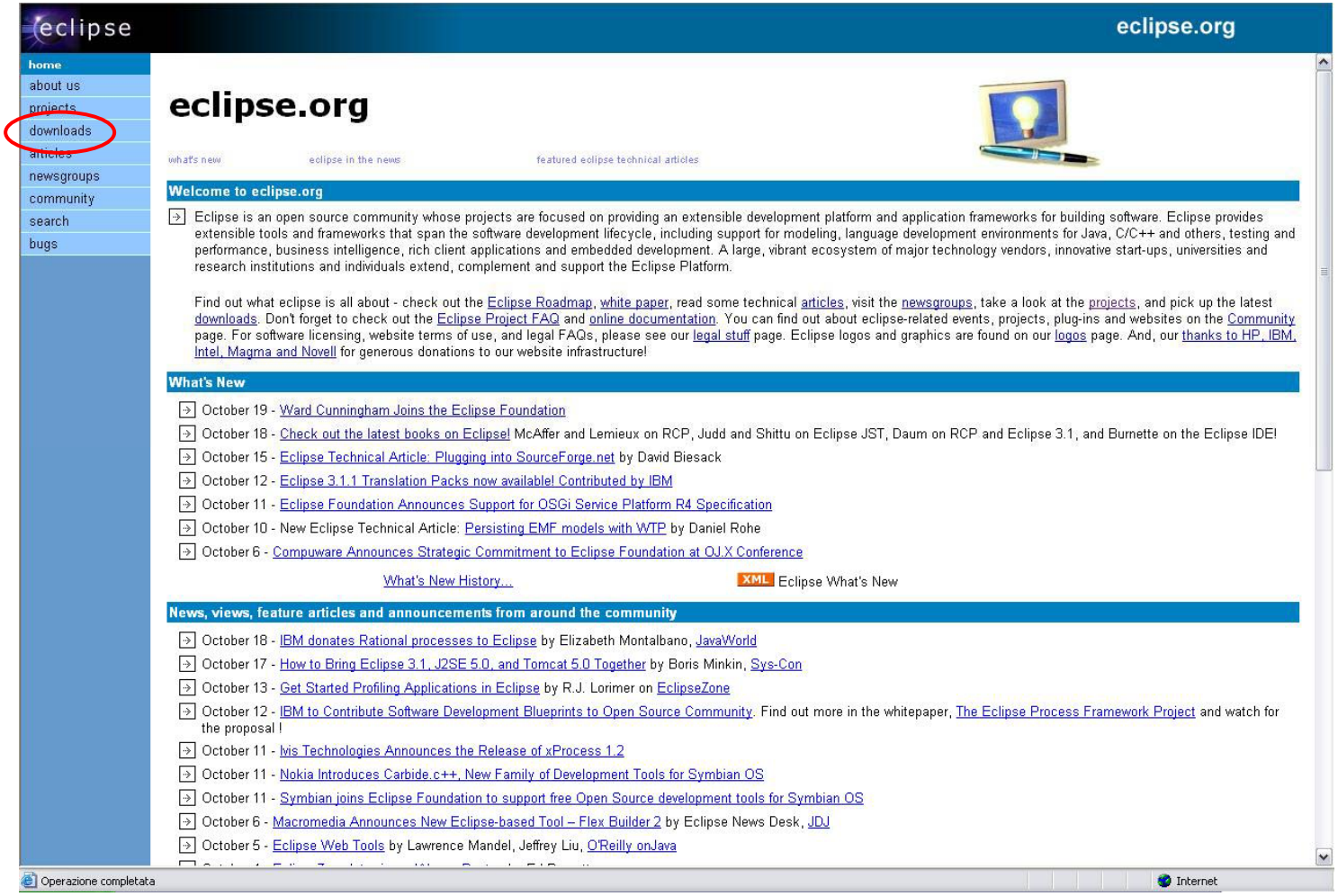


# Omondo – Database Explorer

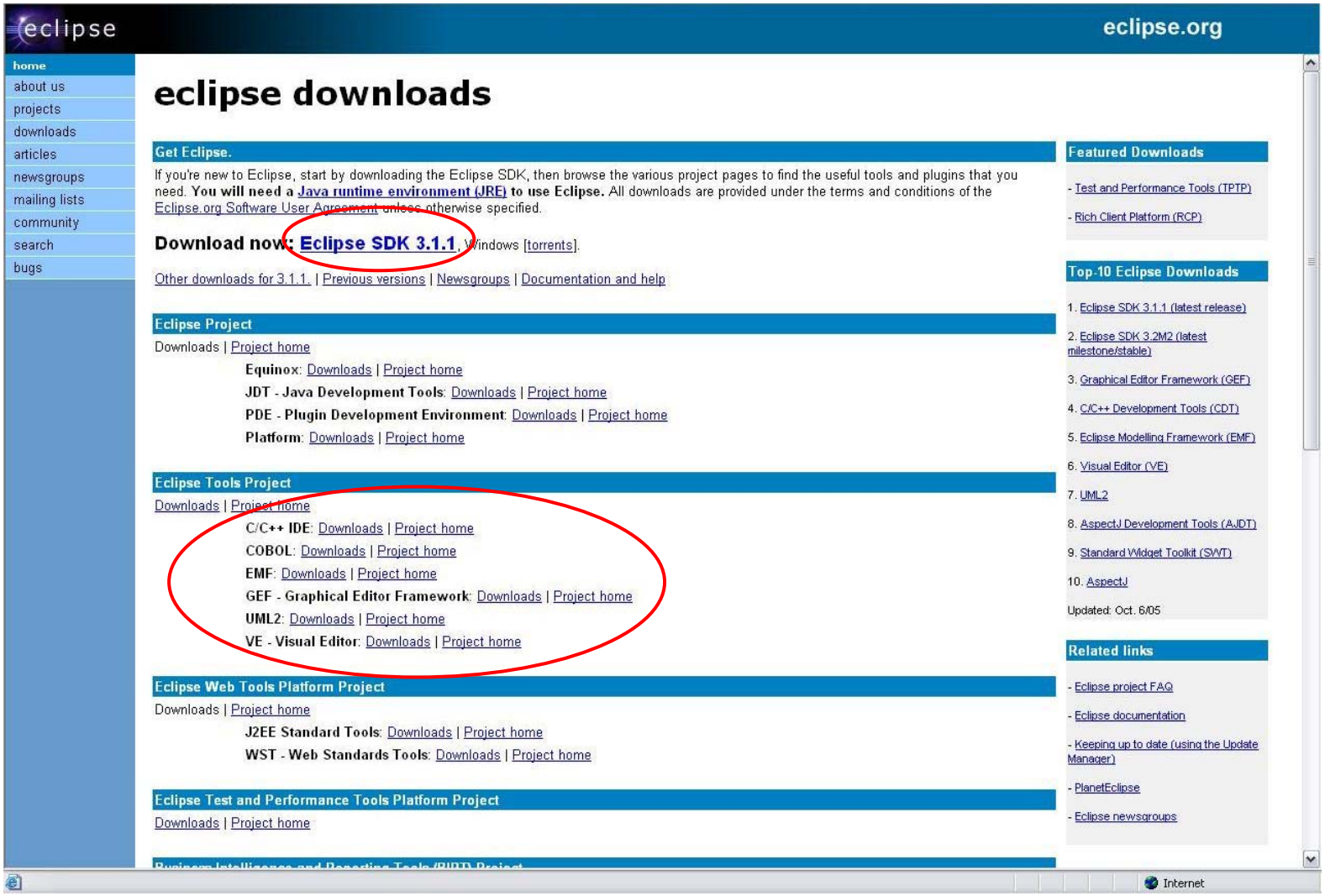


# Eclipse Homepage

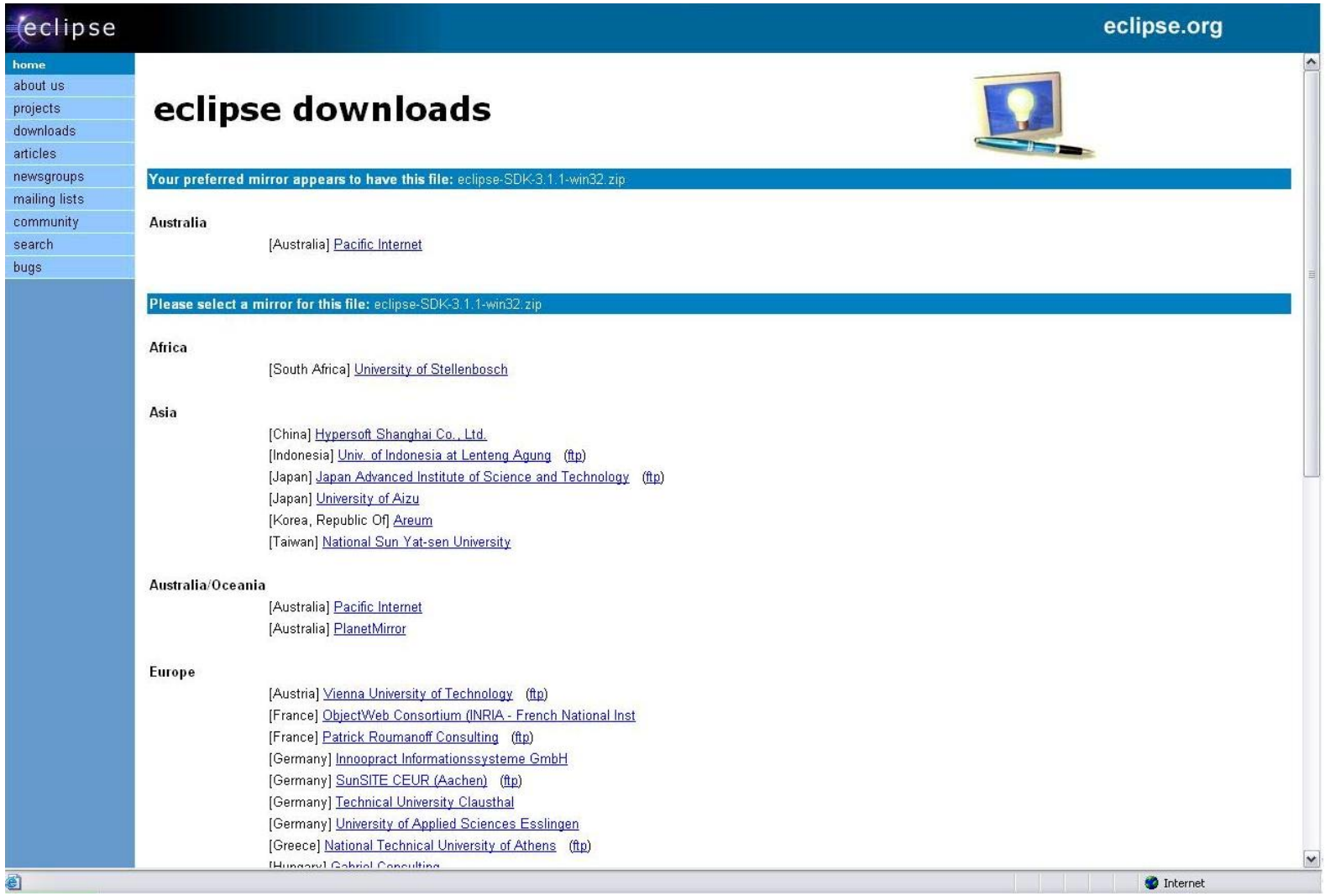
## www.eclipse.org



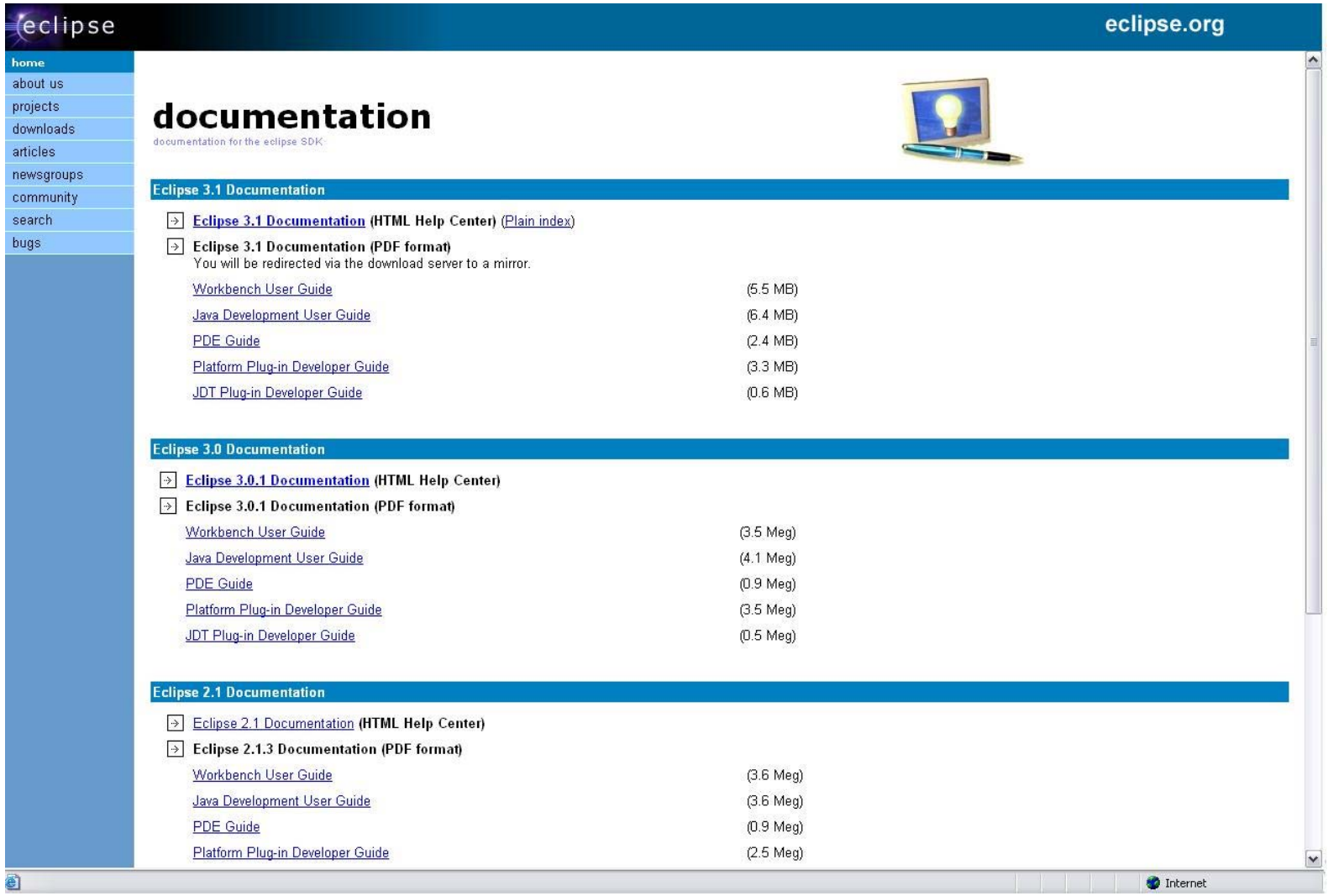
# Eclipse Download Page



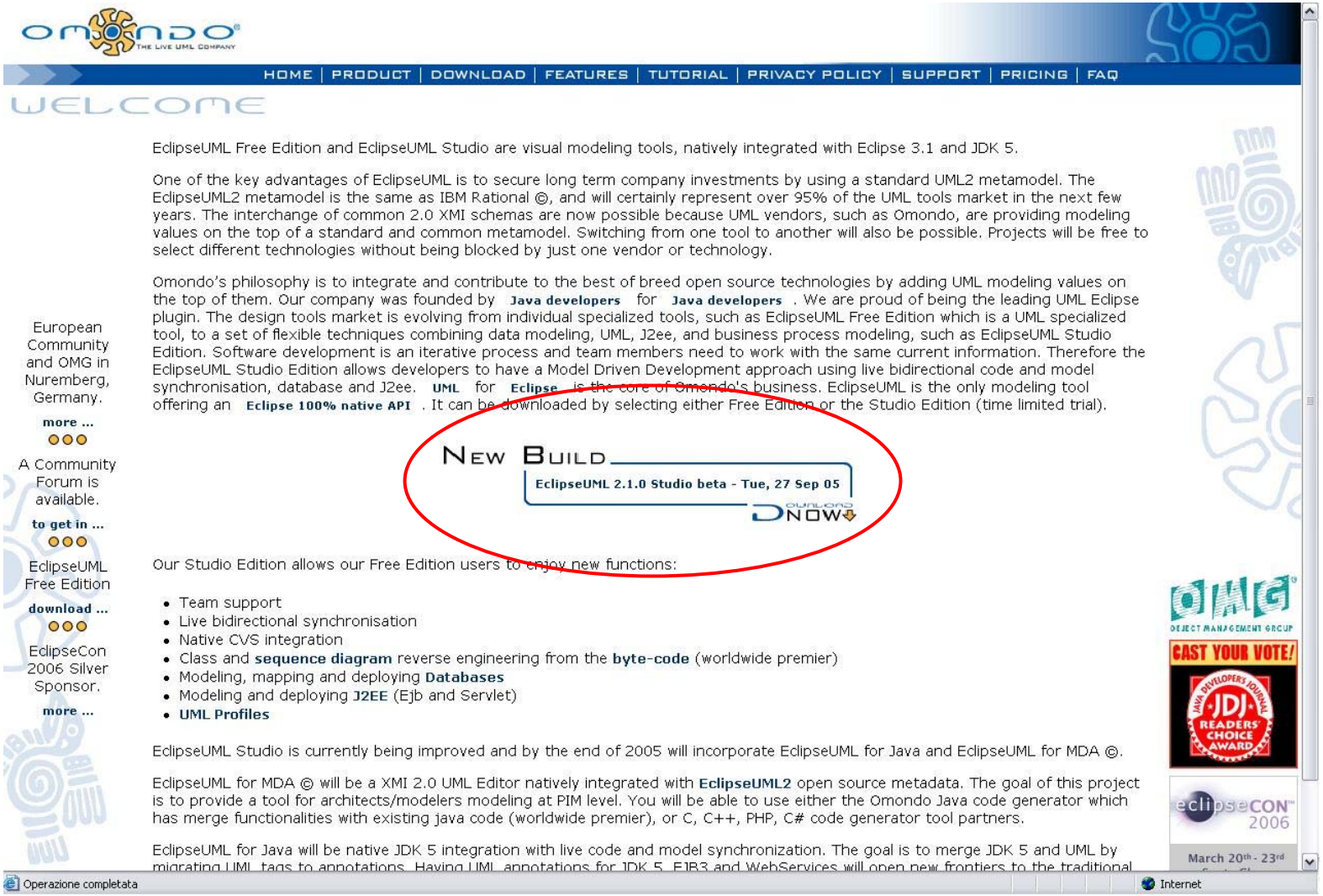
# Eclipse Download Page



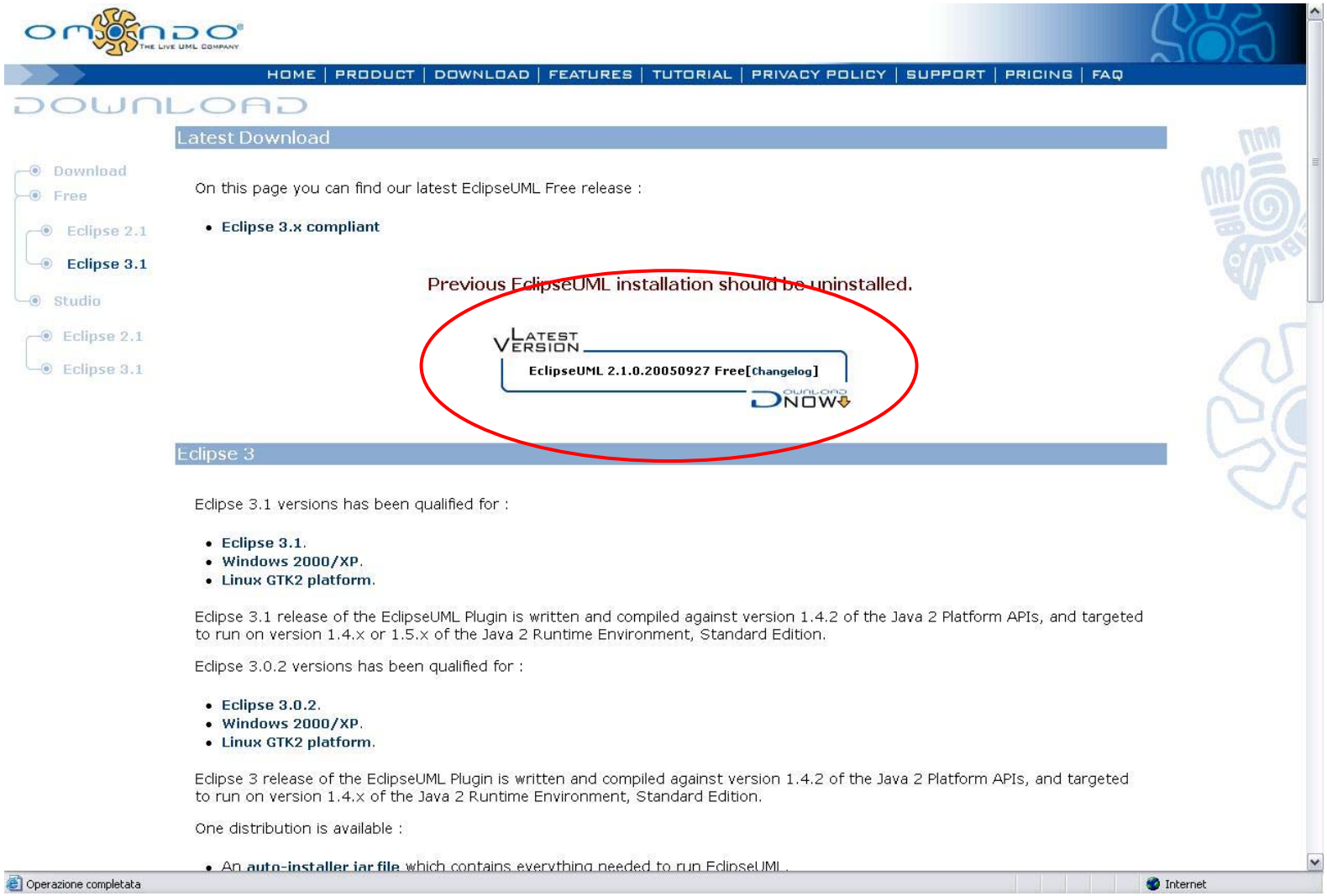
# Eclipse Documentation



# Omondo Homepage



# Omondo Download Page




# DEMO


**AOT Lab**
**Dipartimento di Ingegneria dell'Informazione**
**Università degli Studi di Parma**

# Software Development Tools

## - The Eclipse Platform -

Alessandro Negri
negri@ce.unipr.it