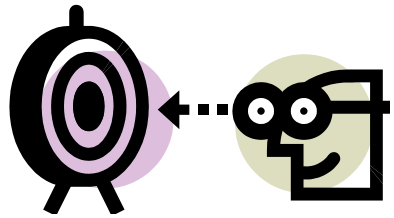
Introduction to Eclipse

Overview

- **Objectives:**
  - The primary intent of this presentation is to familiarize you with the Eclipse Integrated Development Environment (IDE).
  - Through this conversation it is hoped you also gain insight into the larger importance of IDEs and software tools in the software development process
- **Inquiry:**
  - When you write code (or consider when you wrote code for your Data Structures class), what tools did you use?
  - What is “Computer-Aided Software Engineering” and CASE tools? How do they differ from IDEs? Check out the Wikipedia pages for each and consider similarities and differences.

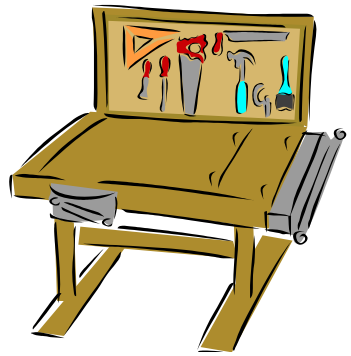
Eclipse Project Aims

- **Provide open platform for application development tools**
  - Designed from its core to be extensible
  - “Open extensible platform for anything and nothing in particular”
- **Resource agnostic**
  - Must teach Eclipse about types
    - Accomplished with plug-ins – editors, views, perspectives, builders
  - Permits unrestricted content types - Java, C/C++, XML, HTML, PHP, devices, etc.
- **Facilitate seamless tool integration**
  - At UI and deeper through plug-in extension points
  - Easily integrate new tools into existing ones
- **Attract community of tool developers**
  - Capitalize on open source community and popularity of
  - Including independent software vendors (ISVs)



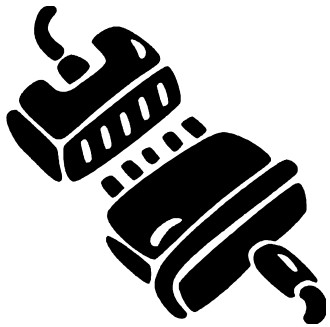
Eclipse Terminology

- **Workbench**
  - An installation of Eclipse with behavior defined by a set of plug-ins
- **Workspace**
  - Manages projects and related projects files
    - Projects may be stored in workspace directory or referenced externally
  - Stores Eclipse setting and preference information
- **Plug-in**
  - Set of behavior to manipulate resources (files, database information, devices, etc.)
  - Behavior exposed through views, editors, perspectives, builders

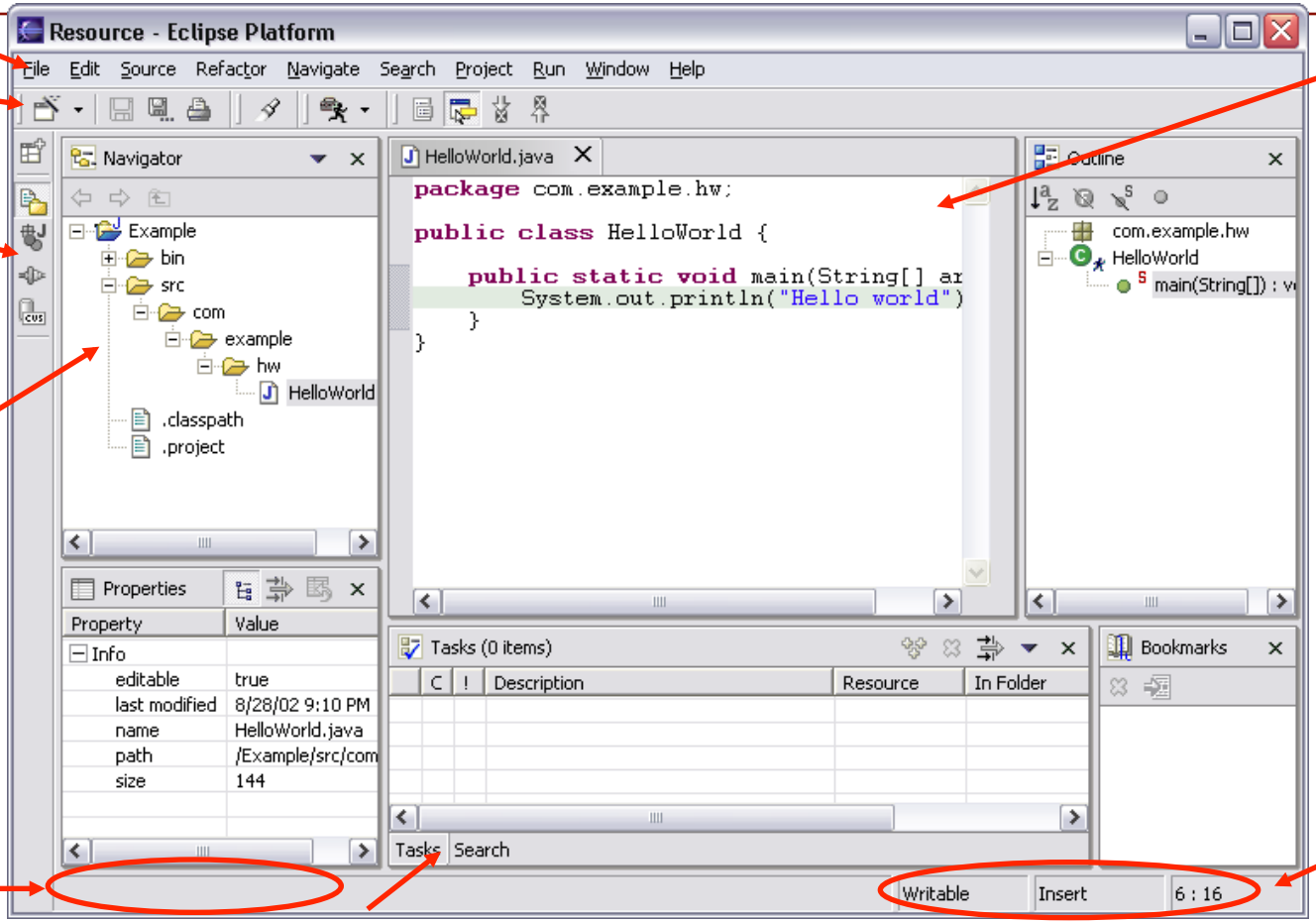


Plug-ins Provide Editors, Views, Perspectives

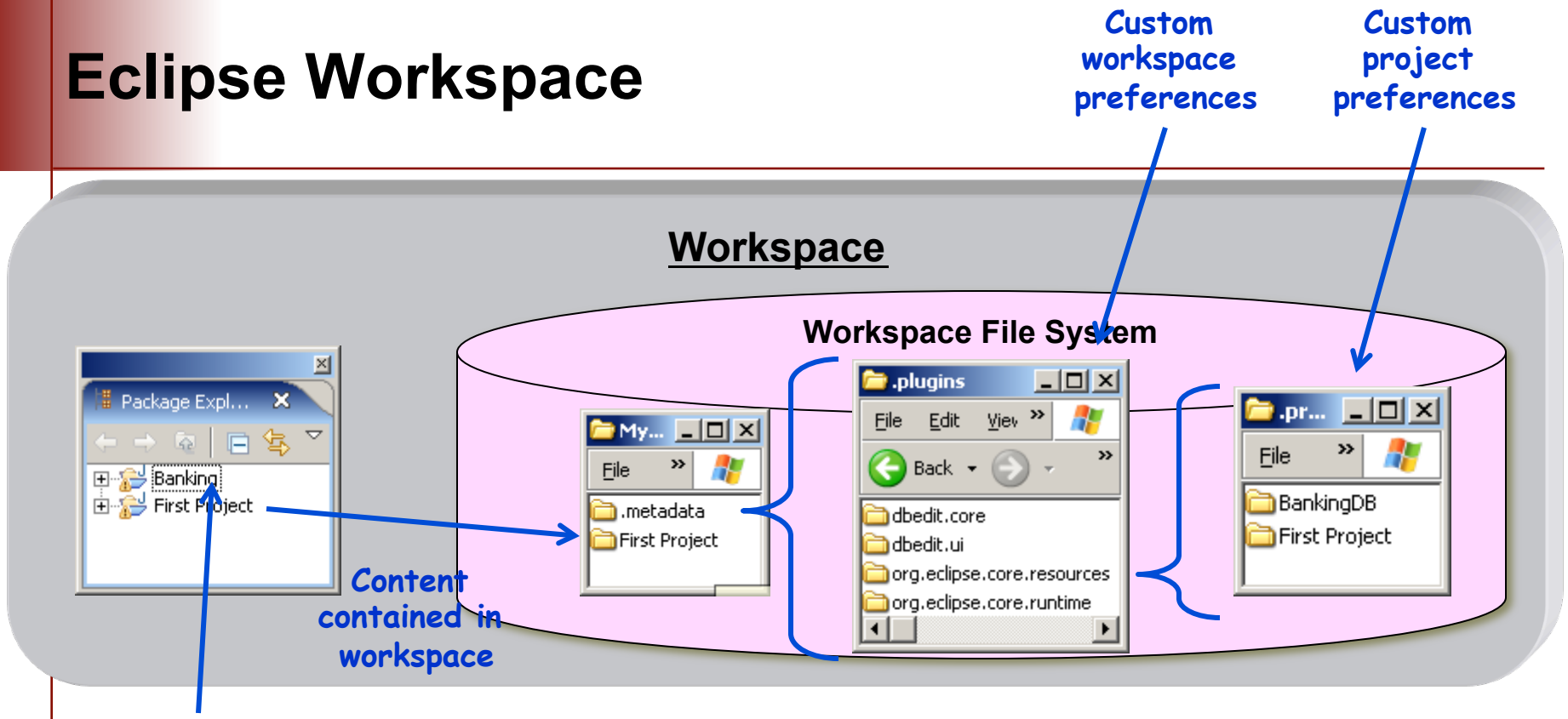
- **Editor**
  - Provides open/edit/save/close model for changing a resource
- **View**
  - Provides information on one or more resources
    - State changes are usually actions based, which differentiates them from editors
  - Views augment editors (ex/ Outline view summarizes content)
  - Views augment other views (ex/ Properties view describes a selection)
- **Perspective**
  - Arrangement of views and editors on the workbench
  - Allows users to quickly switch their views for different tasks
  - User can create manually, but plug-ins usually provide



Eclipse Workbench

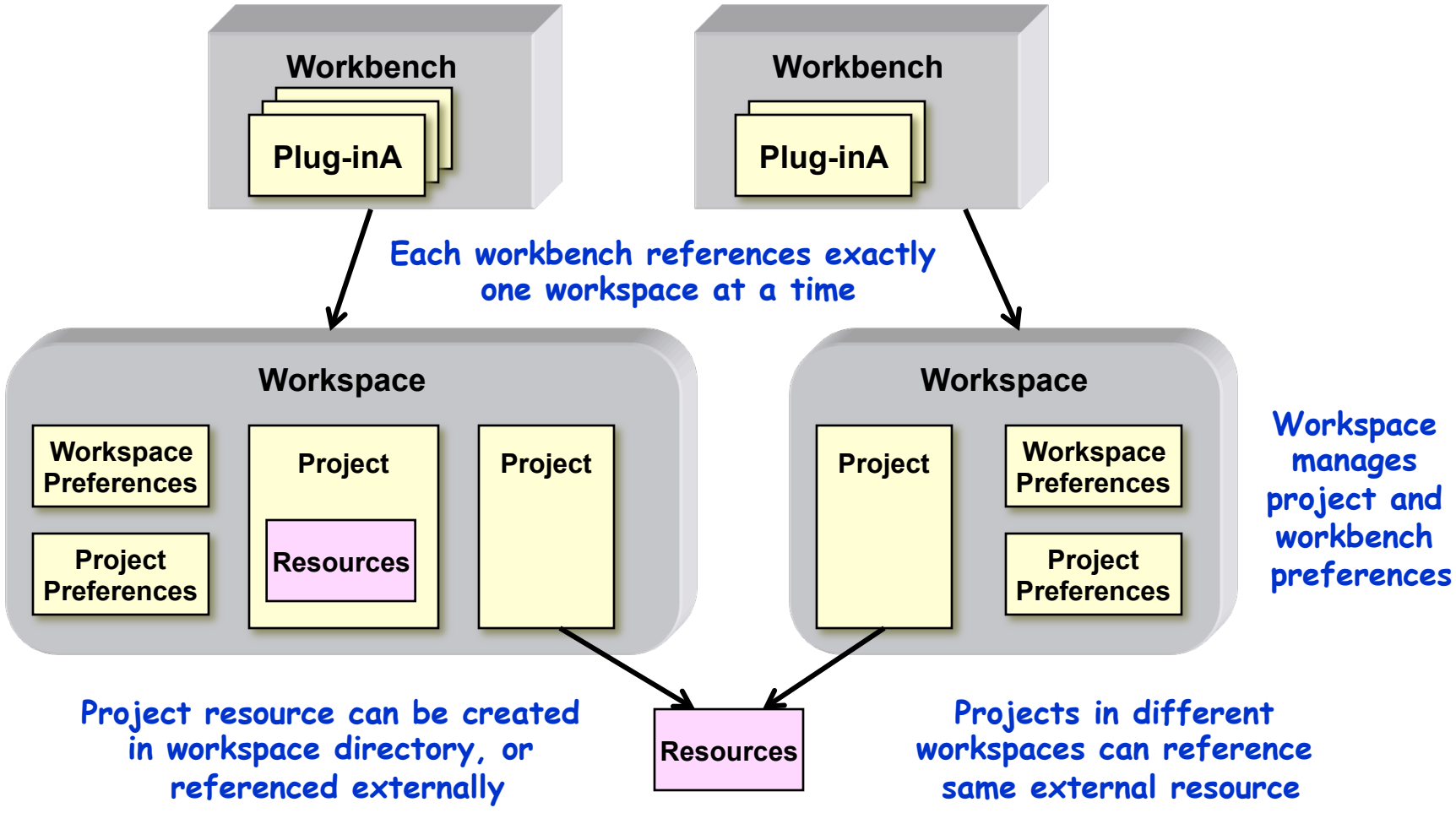


Eclipse Workspace



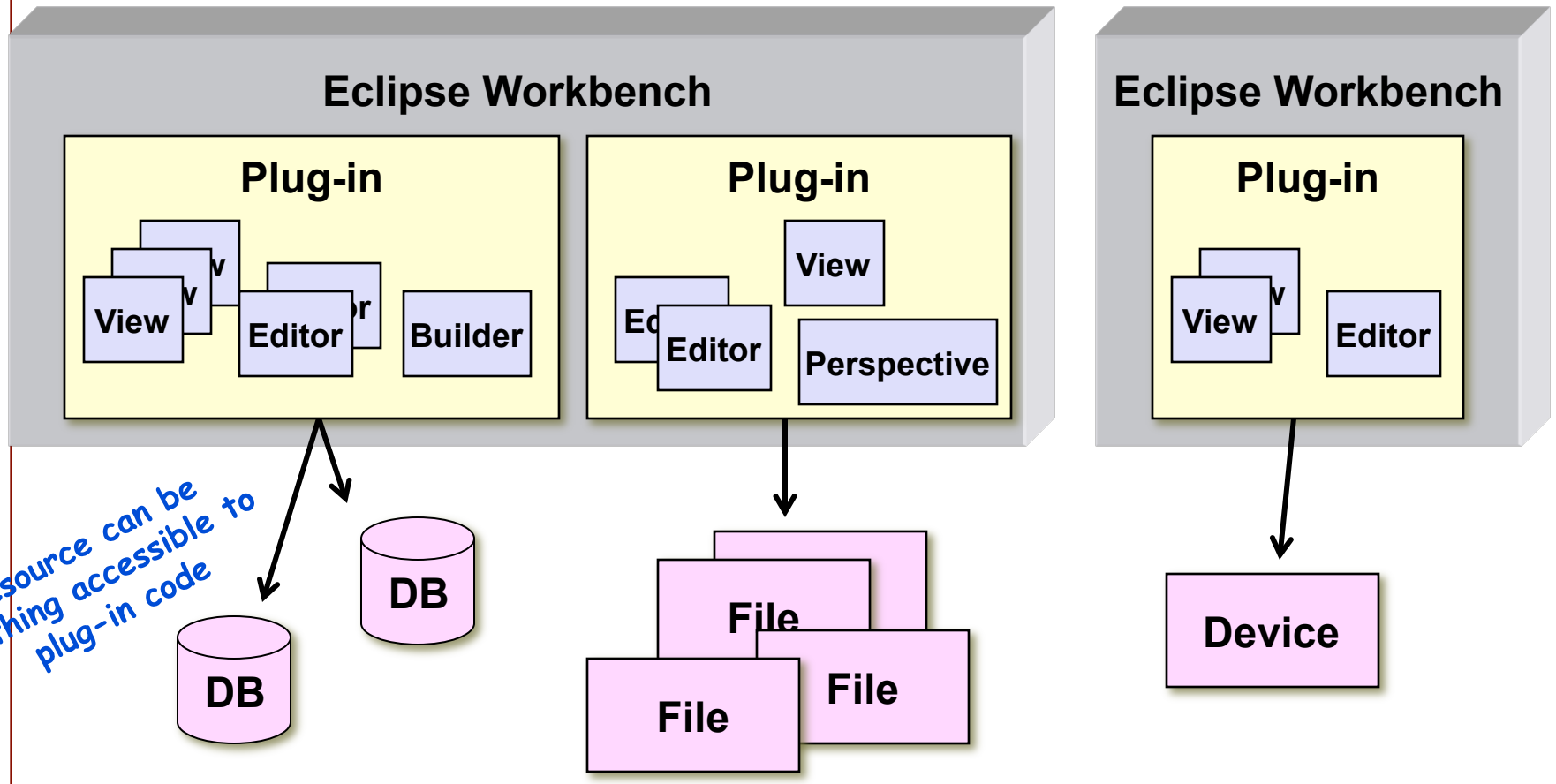
- **Project files can be contained inside (First Project) or outside (Banking) workspace**
- **Workspace manages custom plug-in preferences for both workspace and individual projects**
  - When created, project inherits workspace preferences, which where inherited from the plug-in defaults in the workbench

Workbench and Workspace



Workbench and Plug-ins

Workbench behavior defined by their plug-ins (configured by preferences in workspace)



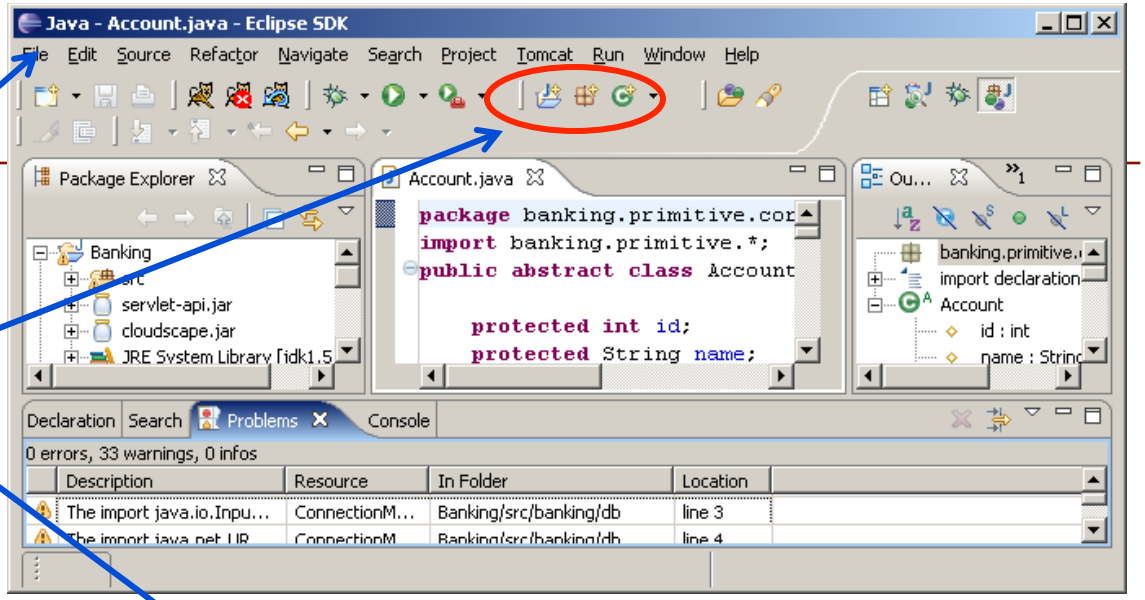
Perspectives

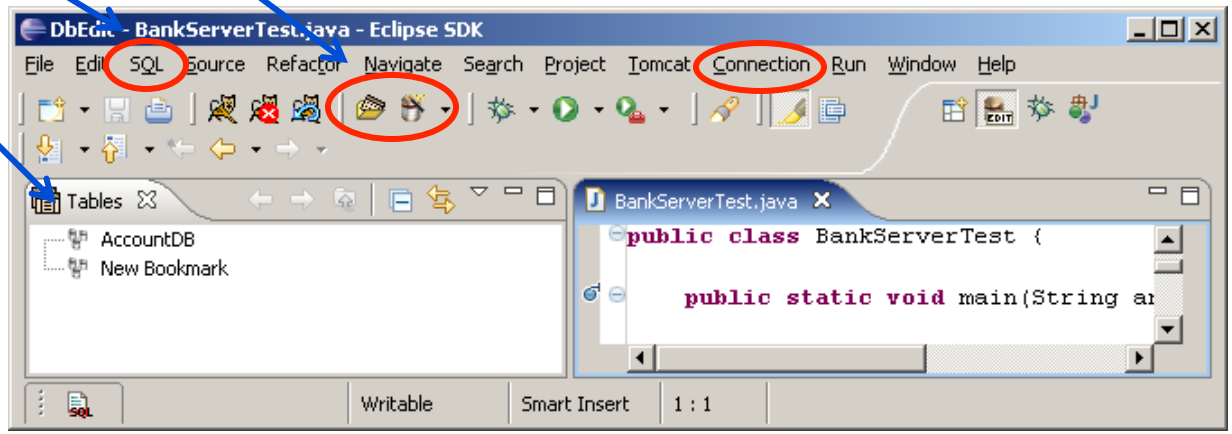
Perspective name

Removed Java toolbar and added database toolbar

Added menu items

New set of views





- **Perspectives control all visible items on the workbench**

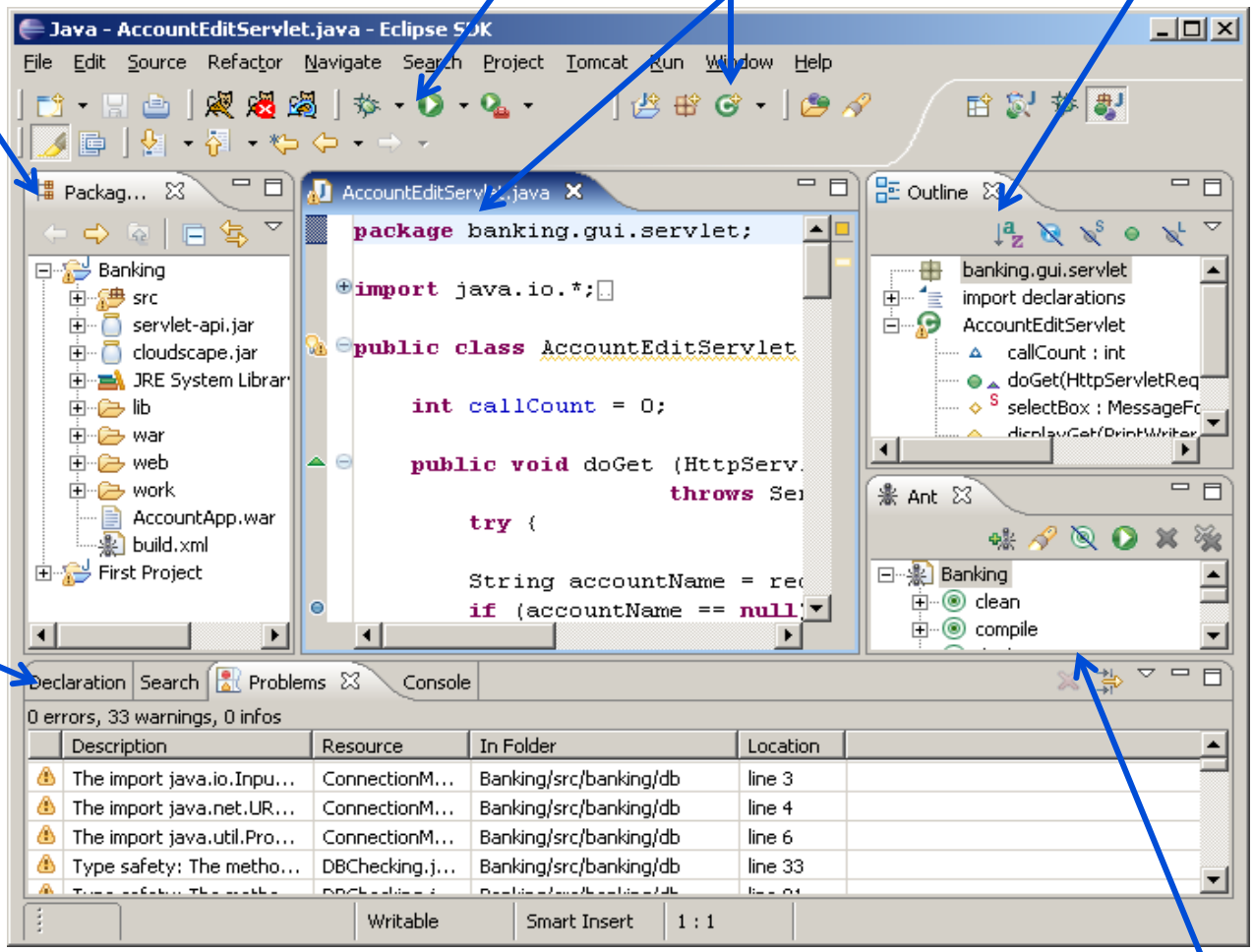
Java Perspective

Java compiler integration and builders

Java-aware outline

Package Explorer

Java Editor

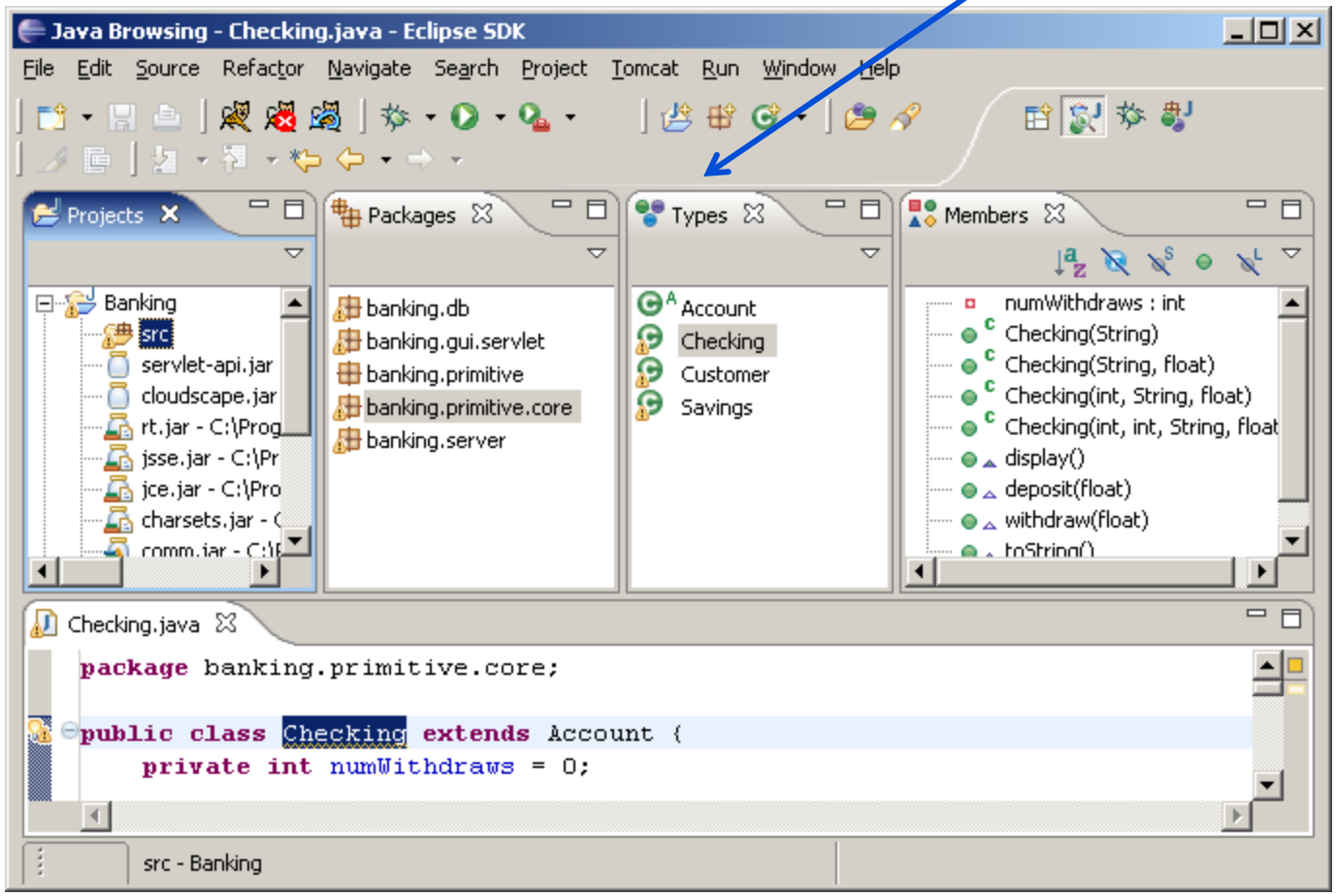


Tab additional views

Ant integration

Java Browsing Perspective

Views update based on selection in other related views



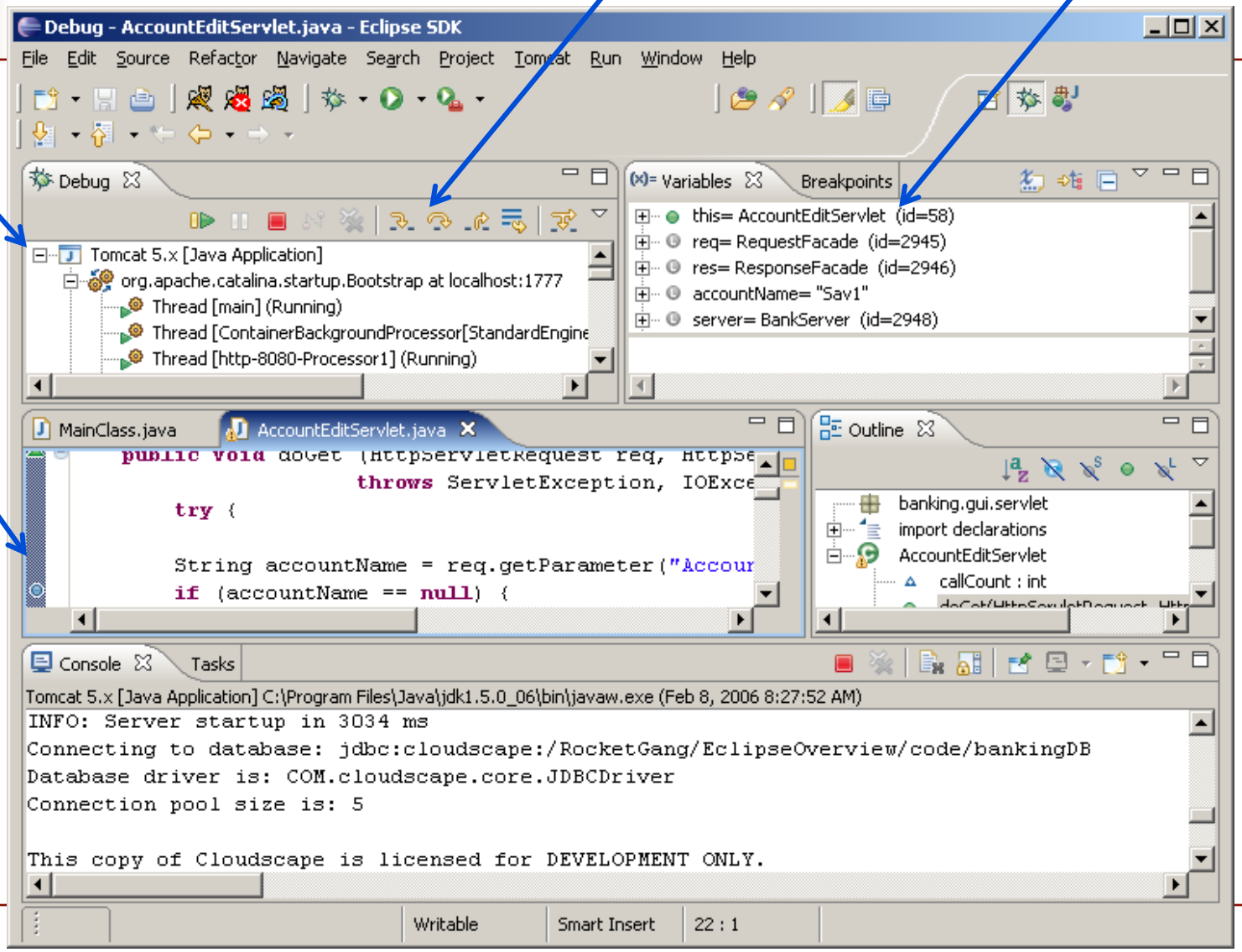
Debug Perspective

Step over/into, run, stop, ...

Currently visible state variables

Processes and Threads

Active editor updates with program execution



JDT Views

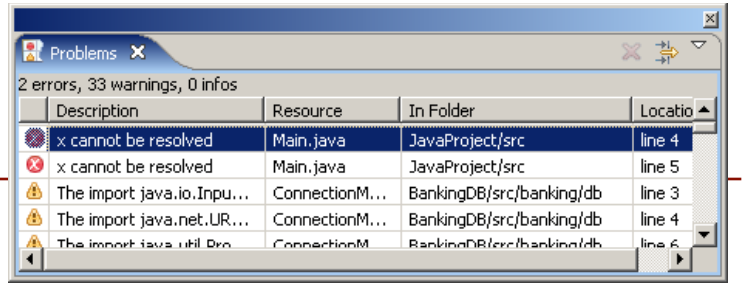
JDT = Java Developer Toolkit

Used for Java projects

Provides default views and builders

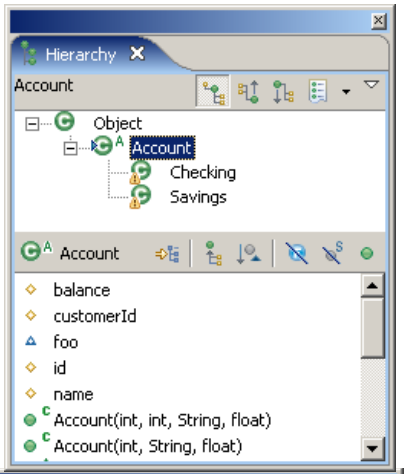
Problems

Aggregates errors/warnings in project or workspace



Type Hierarchy

Shows context for item including marked elements

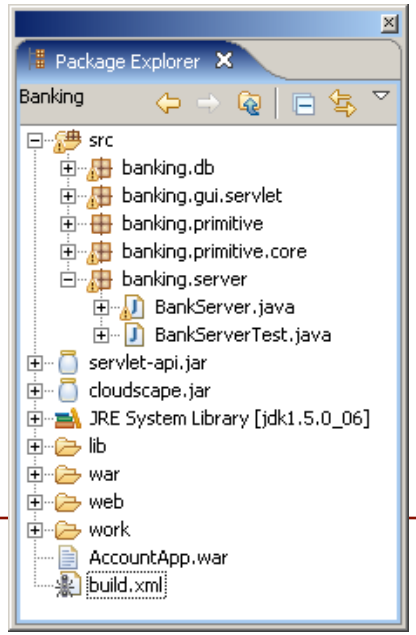


Package Explorer (below)

Use ‘Go-Inside’ to drill into projects

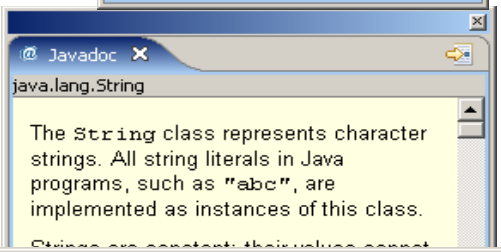
Use ‘Working Set’ to narrow projects

Source files organized by package



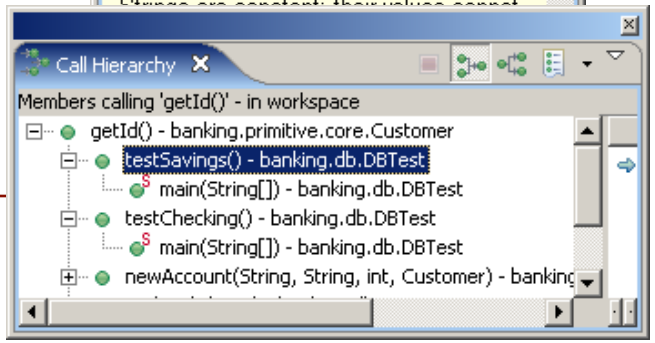
Javadoc

Shows javadoc for elements highlighted in Java editor



Call Hierarchy

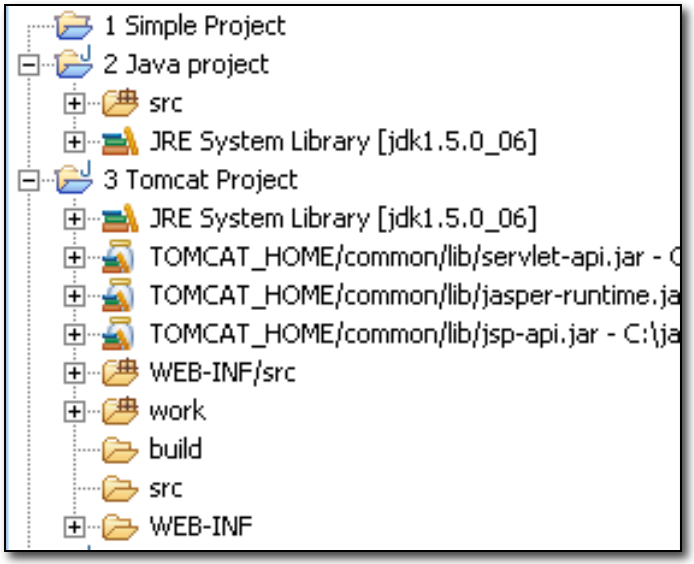
Shows all call sequences for a selected method



Using Eclipse

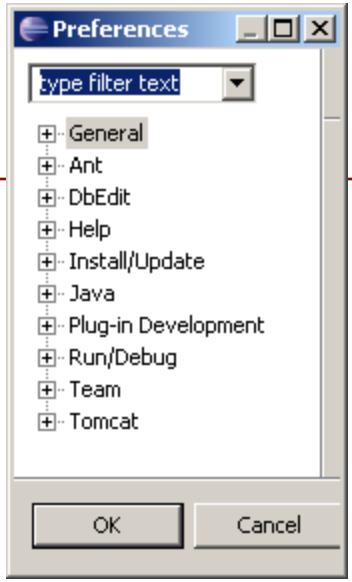
- **Launching Eclipse**
  - ➢ Loads projects from workspace directory
    - ■ You may have many such directories
  - ➢ Loads preferences from workspace preferences
- **Creating a project**
  - ➢ Projects creation uses wizards for different project types
    - ■ Defines initial structure and files
    - ■ Sets 'nature' which sets builders and other actions
  - ➢ You can also import and export a project
    - ■ Useful if you get a working project copy from another developer

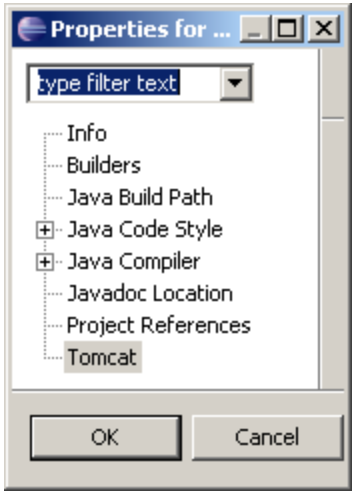
Project wizards pre-populate project with common files in a typical organization for the project type



Workbench Preferences and Project Properties

- **Customize plug-ins for use in the workbench and project**
- **Preferences used by all projects in workbench**
  - ➢ Fonts, colors, external libraries, JVMs, etc.
- **Properties are project specific**
  - ➢ Builders, other plug-in specific preferences
  - ➢ Many properties override workbench preferences
    - ■ Build path, coding style, format, errors/warnings

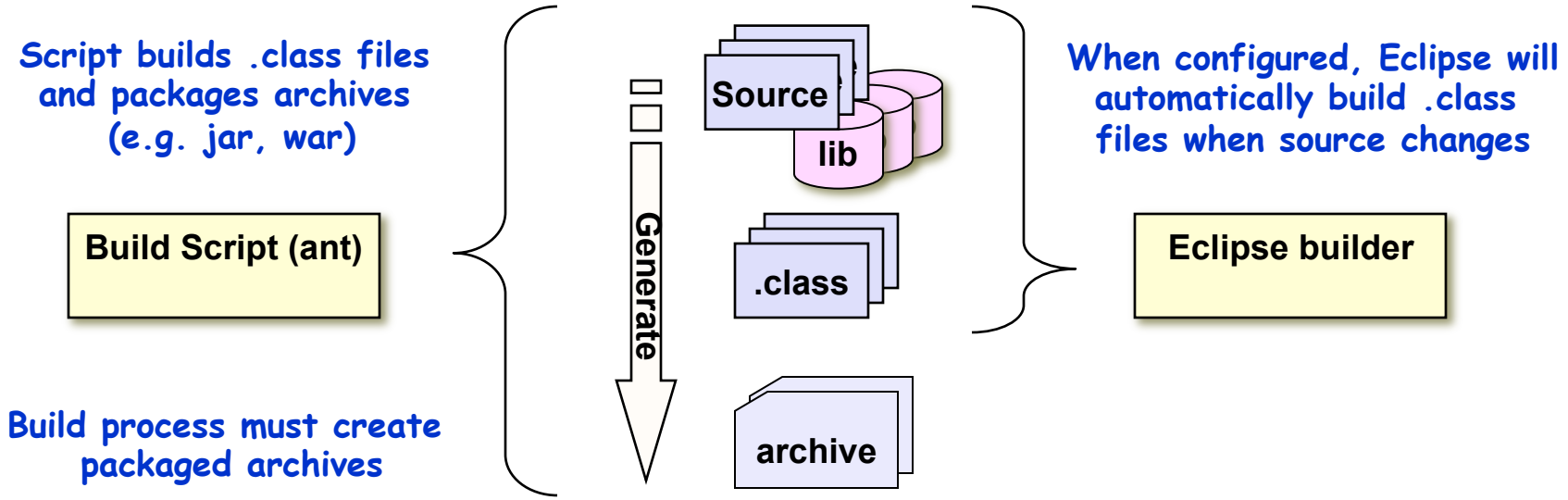




You will most likely not have to do much customization of your environment if you use Eclipse. However, you should be aware of these in case you need to troubleshoot configuration problems

Configuring the Java Builder

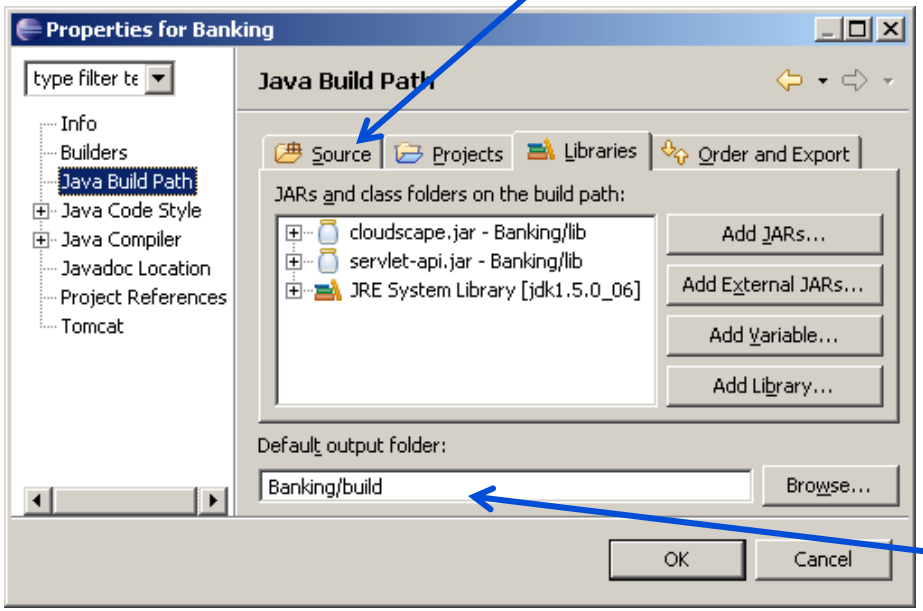
- **Both Java builder and build scripts (ant) can perform compilation**
  - Eclipse builder provides incremental compilation, but cannot perform post compilation activities (e.g. package system into archives)
  - Should synchronize Eclipse builder with build process



Configuring Builder

- **Builder configured in project preferences**
  - ➢ Provides per-project customized preferences, defaults inherited from workspace (Window > Preferences)

Instruct builder which directories contain sources



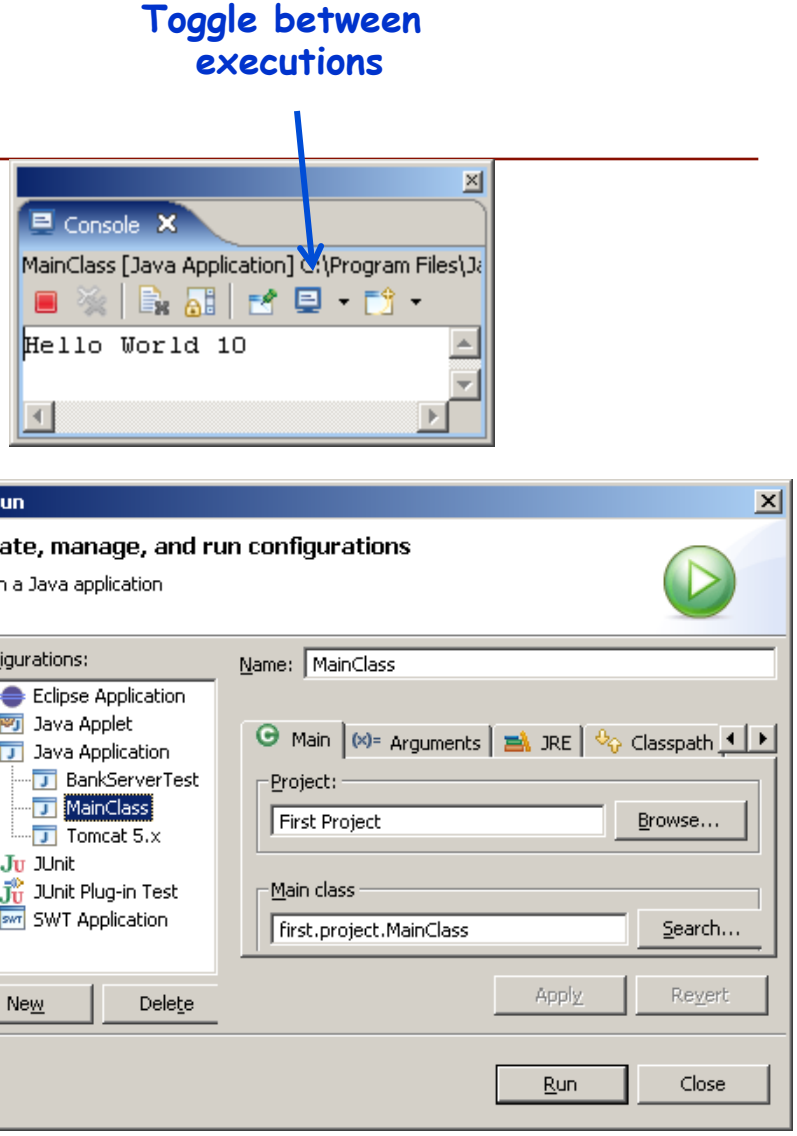
Project-specific configurations for coding violations, JVM selection, etc.

Libraries may come from project, external locations, or a workspace-defined variable

One and only location where builder writes .class files

Project Execution

- **Outputs for all executions shown in console**
- **Quick execution for any class with a main() method**
  - ➢ Right-click Run As > Java Application
- **Launch configuration provides more control**
  - ➢ JVM and command line args, JRE, classpath, environment vars
  - ➢ Can pipe console output to a file
  - ➢ Can export to a file and share with team

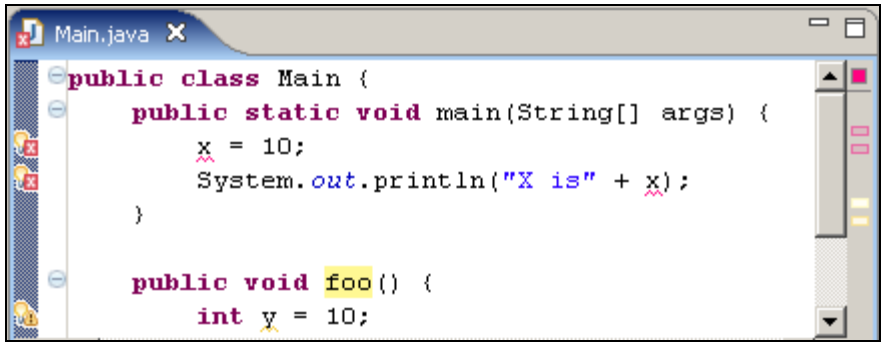


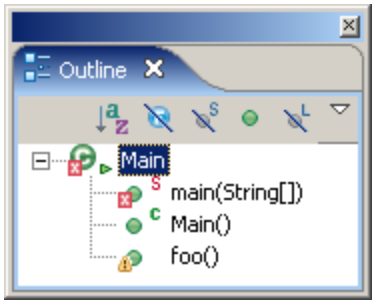
Markers

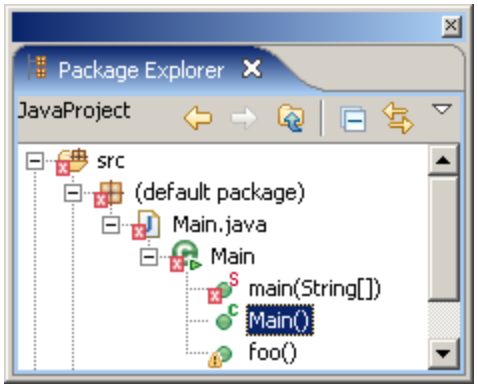
Markers are important to us as several of the tool plugins we will use alert you to possible quality issues visually through markers!

- **Annotate items within a resource**
  - ➢ Do not modify resource, only highlight locations
- **Apply to both views and editors**

Java builder highlights errors and warnings in Java editor and Java-related views



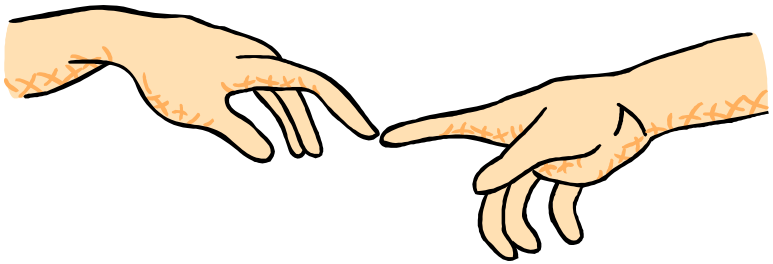




Markers can indicate erroneous items or provide information (e.g. static, constructor)

Eclipse Editors: “At your fingertips”

- **Text manipulation**
  - Tab indentation
  - Comment/uncomment select code
  - Add a comment, comment/import folding
  - Syntax highlighting
  - Code Completion



- **Code templates and completion (<cntrl>-space)**
  - Compound statements – for, while, try, etc. (control-space)
  - Create inherited method stubs (Source-Override Methods)
  - Generate attribute getter and setter methods
  - Callable methods (‘.’) - -show what you can call in context, with its javadoc
- **Real-time code review**
  - Indications of issues in multiple locations – Explorer, Editor, Problems pane
  - Quick Fix automatically updates source for common solutions

IDEs (and just some of the) Pros and Cons

- **Eclipse is just one of many many popular IDEs**
  - Java: NetBeans, IntelliJ…
  - Javascript/HTML5: Sublime, Atom, Brackets...
  - OS-specific: Visual Studio (Windows), Xcode (Mac)
- **Benefits of IDEs:**
  - Productivity – saves keystrokes, provides basic automation
  - Extensible – Community contributes extensions and integrations
  - Enhances quality – code-level “just-in-time” discovery and fix
- **Drawbacks to IDEs:**
  - Builds may not resemble downstream builds
  - Learning curve and migrating with revisions
  - May add crud to your source code control repository if not careful