

Building Application using IDE



Objectives

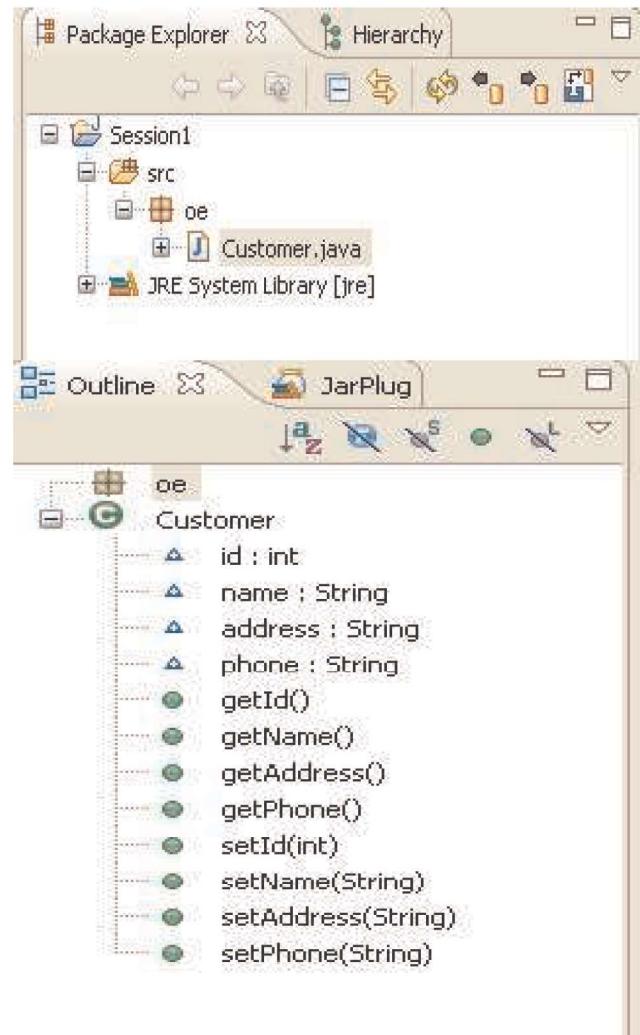
After completing this lesson, you should be able to do the following:

- Create applications and new projects
- Build Java applications in IBM Eclipse
- Enhance user interface frame design
- Debug applications with the Eclipse debugger
- Define classes with Eclipse
- Describe how Eclipse can be used to build enterprise applications



- IBM Eclipse 3.X provides an integrated development environment (IDE).
- It enables you to:
 - Build, compile, and run Java applications
 - Use wizards to help build source code
 - View objects from many perspectives: code, structure, layout, and so on

Eclipse Environment



Package Explorer

```
Customer.java
package oe;

public class Customer
{
    int id;
    String name;
    String address;
    String phone;

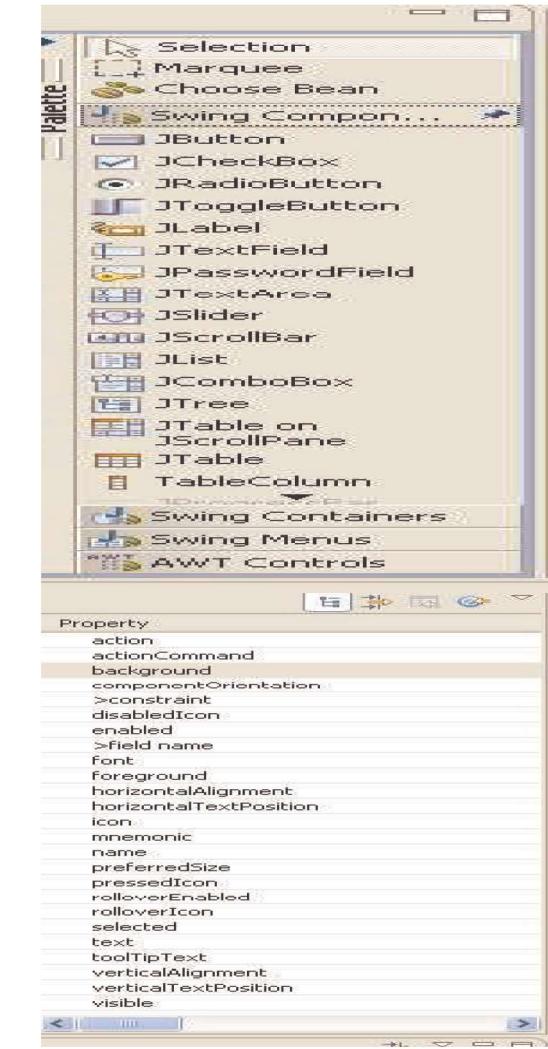
    public int getId()
    {
        return id;
    }

    public String getName()
    {
        return name;
    }

    public String getAddress()
    {
        return address;
    }

    public String getPhone()
    {
    }
}
```

Code Editor



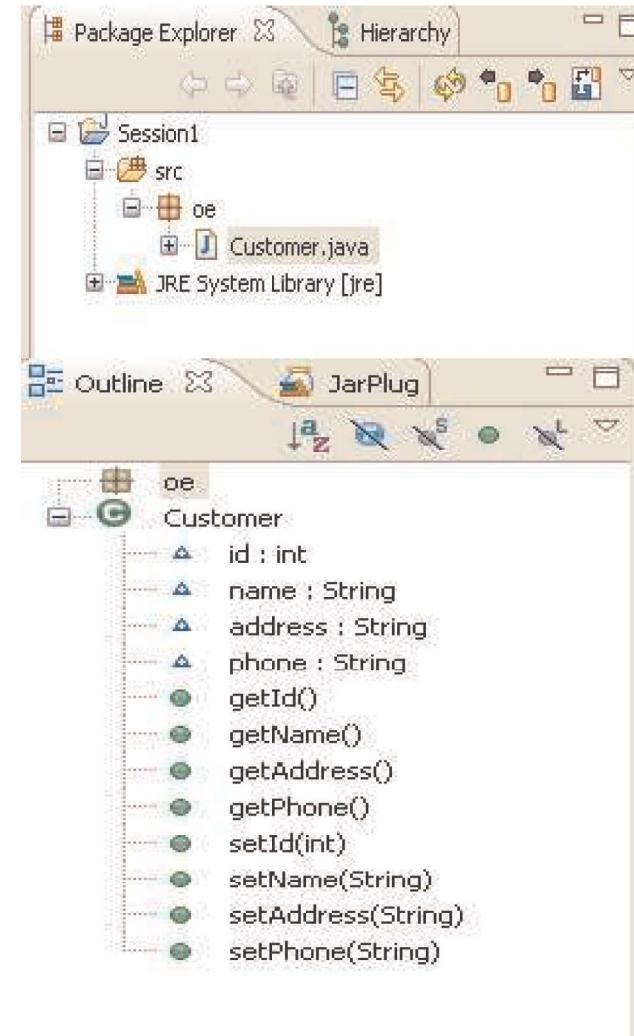
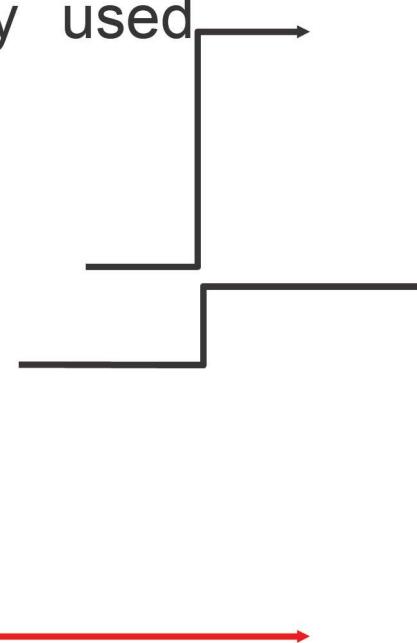
Property Inspector

Projects

- May contain multiple projects
- Enable you to view currently used objects

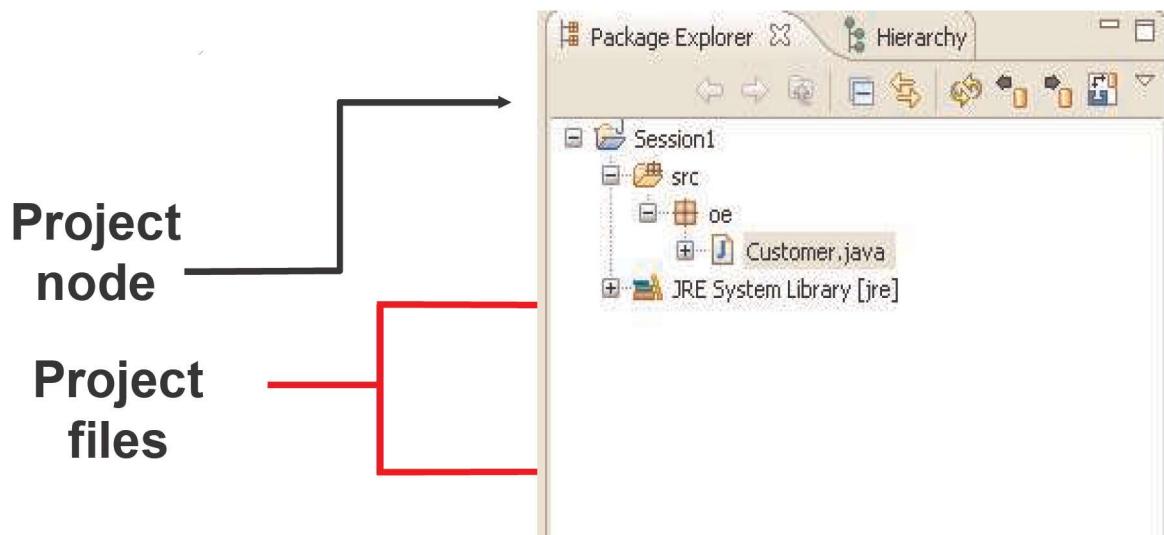
Project node
Projects
Navigator pane

Outline pane

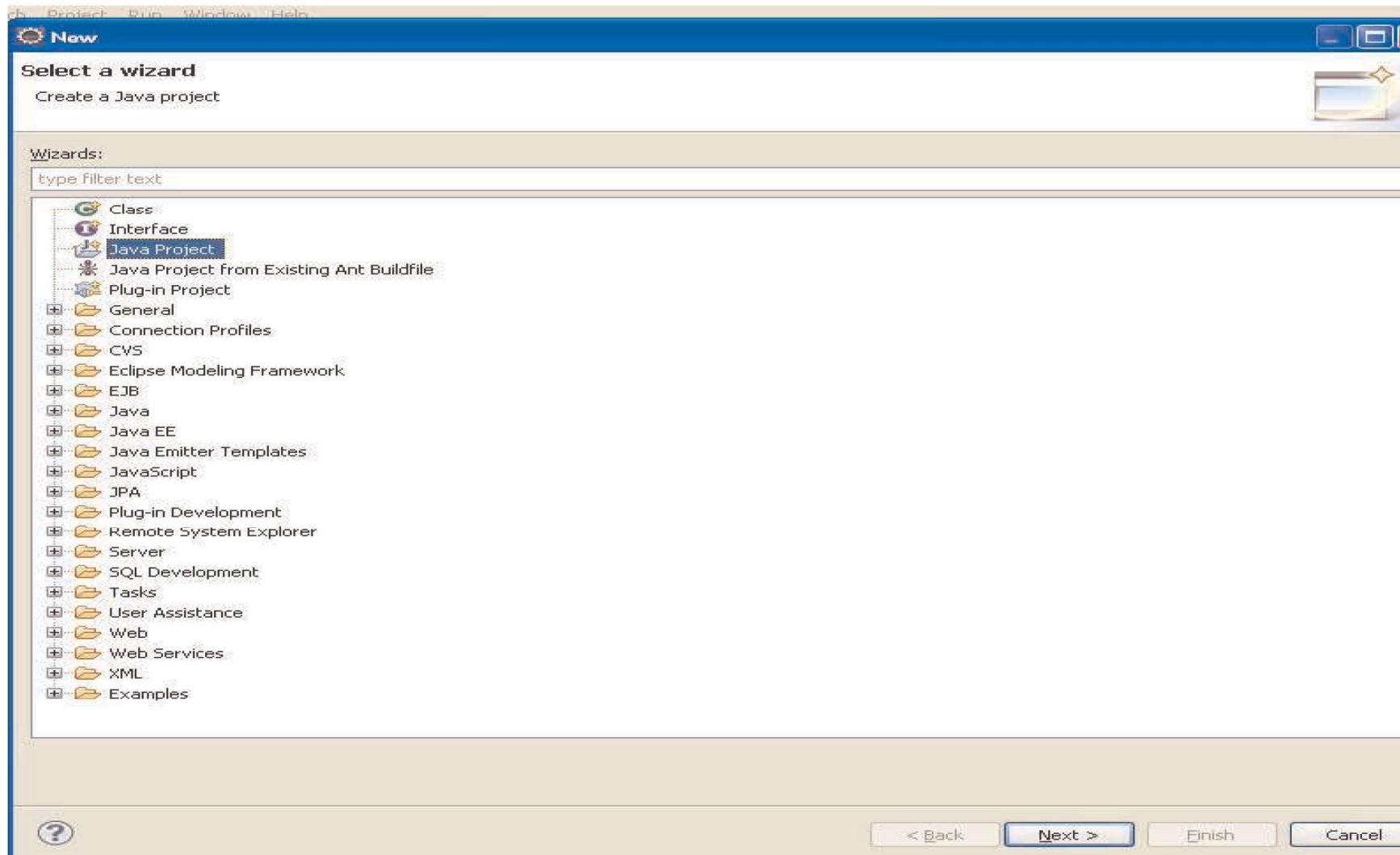


Projects

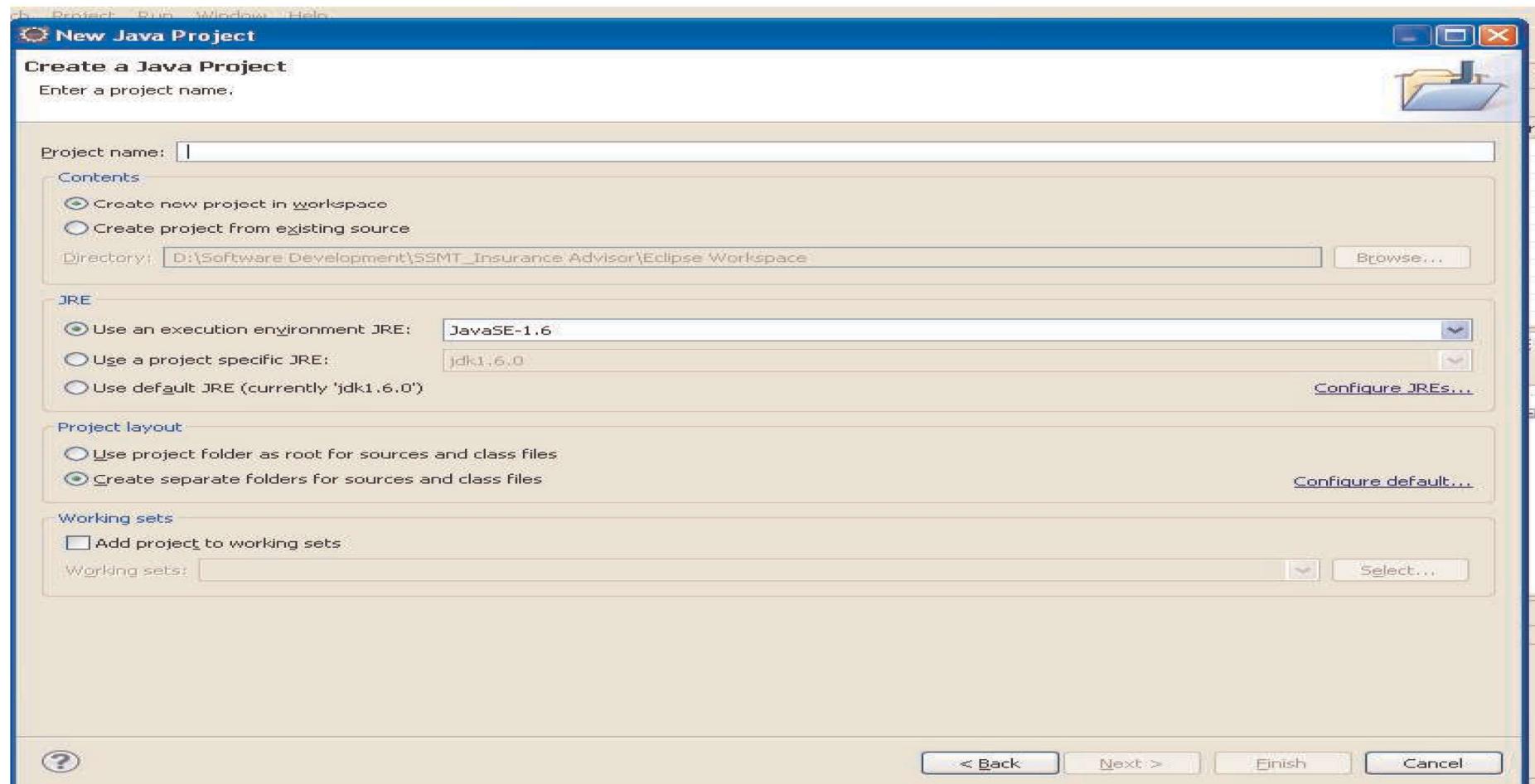
- Contain related files
- Manage project and environment settings
- Manage compiler and debug options



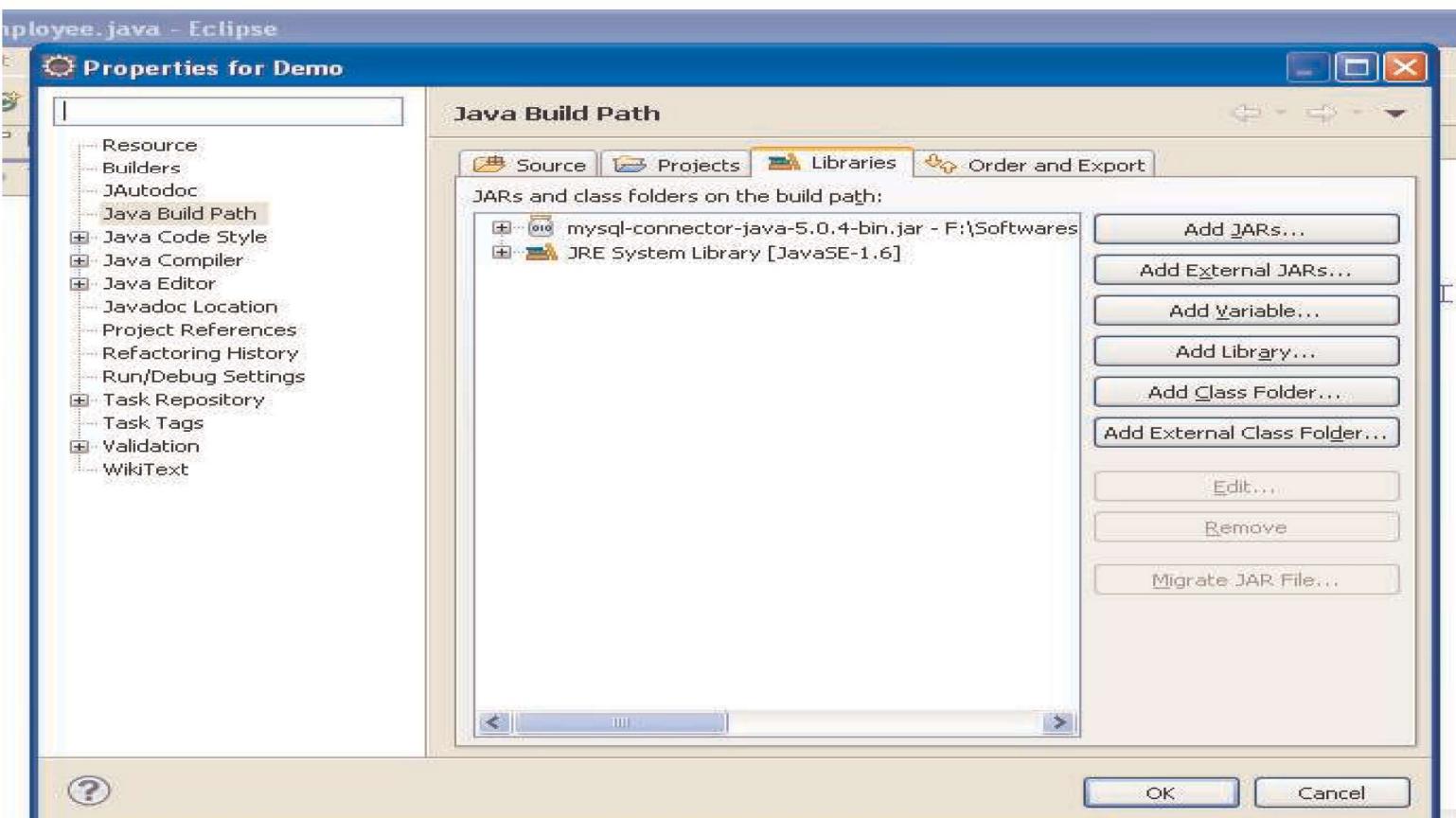
Creating Eclipse Items



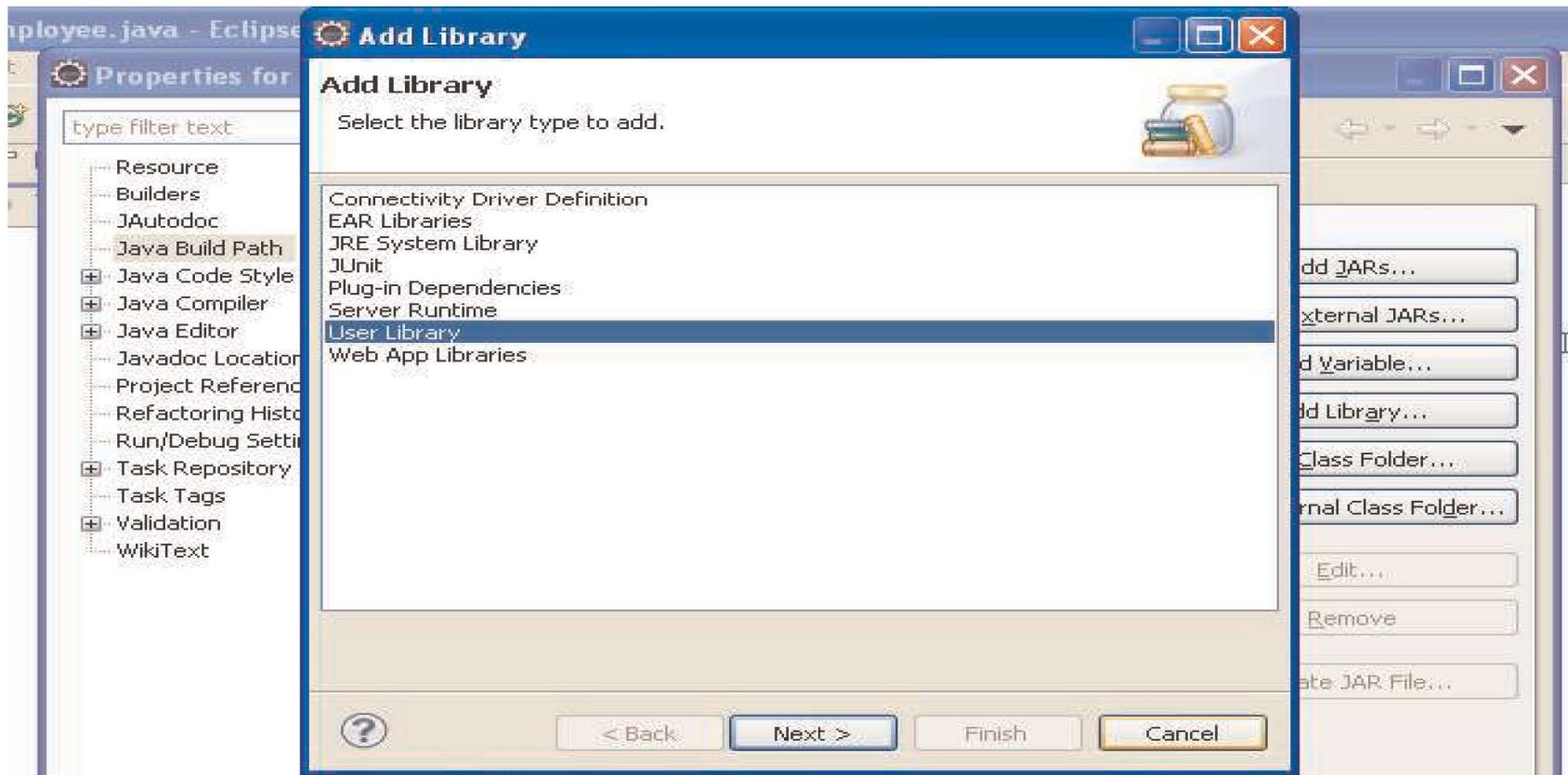
Creating a Project



Project Properties: Specifying Project Details

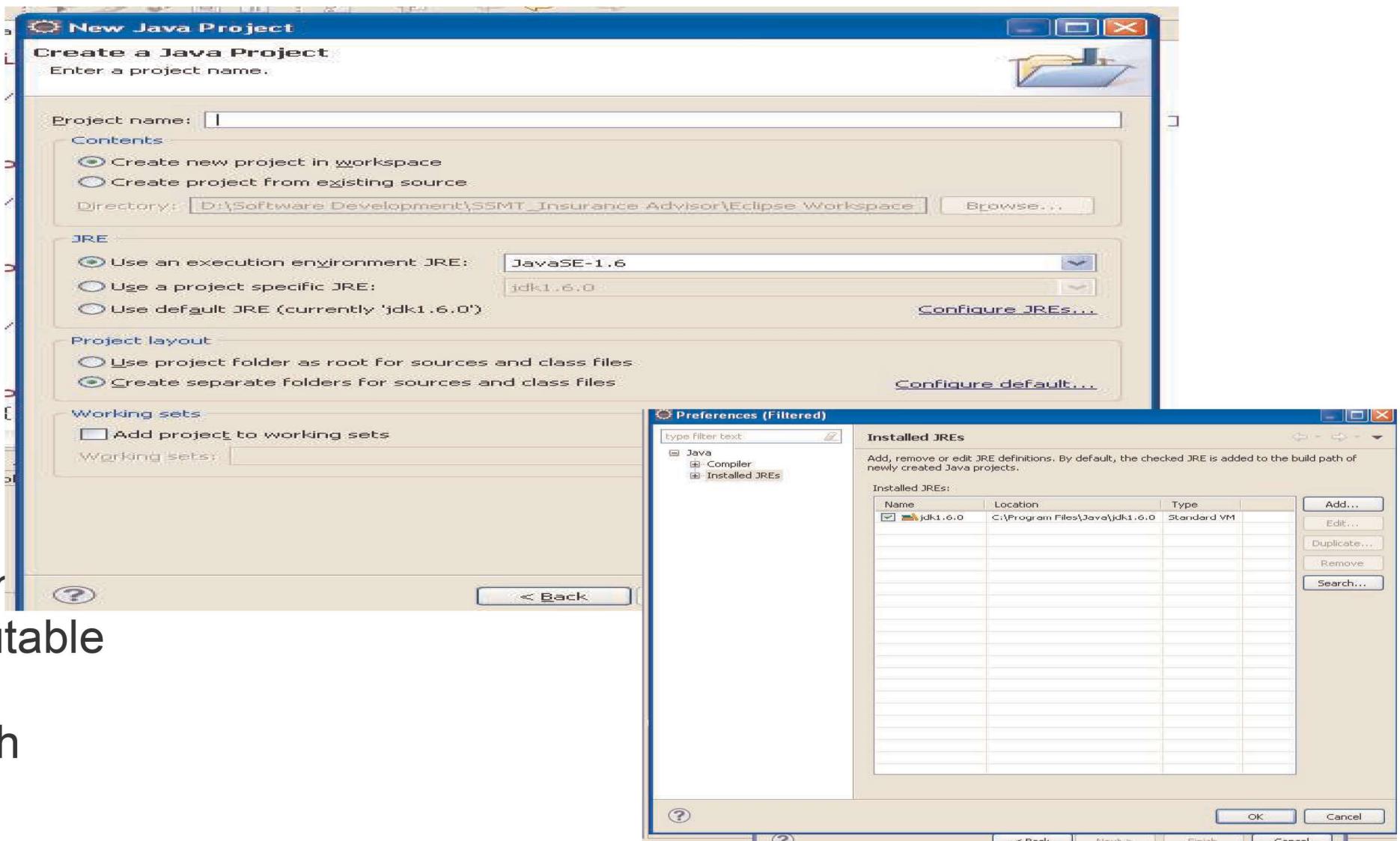


Project Properties: Selecting Additional Libraries



Adding a New Java SE

- Java SE definition
 - Java executable
 - Classpath
 - Source path
 - Doc path



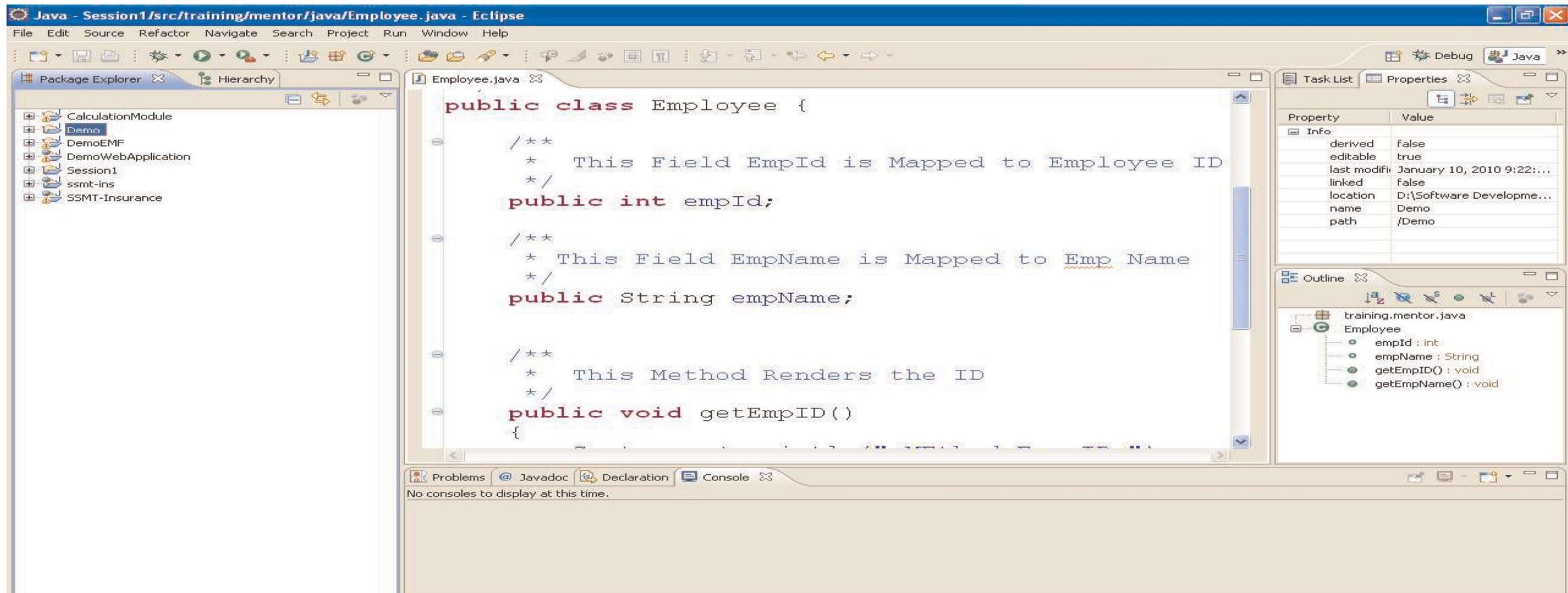
Directory Structure

Eclipse creates and stores .java and .class files by using the following conventions:

- <Workspace>\<Name>
- Followed by the application name
- Followed by the project name
 - \bin\<package name>\
 - \src\<package_name>\
- Followed by bin and src files

Exploring the Skeleton Java Application

Contains application classes:



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java - Session1/src/training/mentor/java/Employee.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows projects: CalculationModule, Demo, DemoEMF, DemoWebApplication, Session1, ssmt-ins, SSMT-Insurance. The Demo project is selected.
- Editor:** Displays the code for Employee.java:

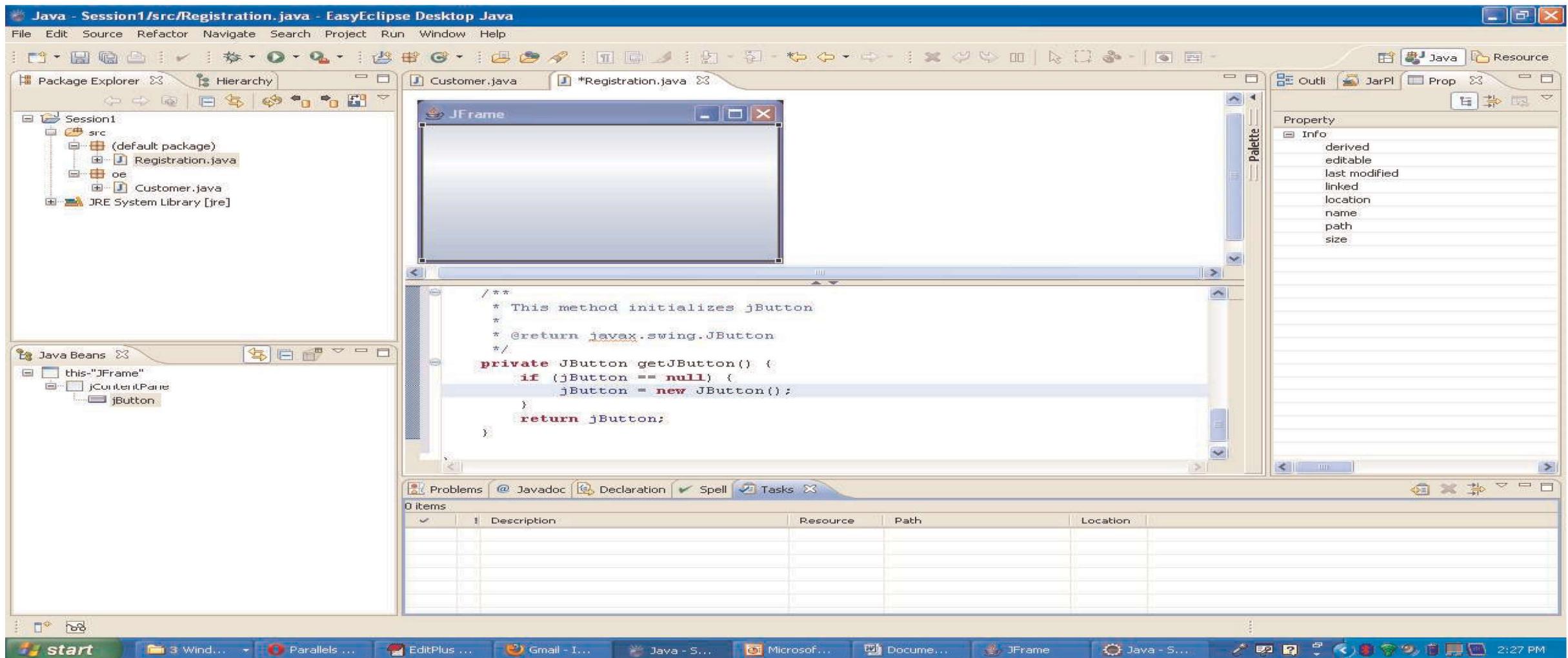
```
public class Employee {  
    /**  
     * This Field EmpId is Mapped to Employee ID  
     */  
    public int empId;  
  
    /**  
     * This Field EmpName is Mapped to Emp Name  
     */  
    public String empName;  
  
    /**  
     * This Method Renders the ID  
     */  
    public void getEmpID()  
    {  
    }  
}
```
- Properties View:** Shows properties for the selected item (Info):

| Property | Value |
|---------------|------------------------------------|
| derived | false |
| editable | true |
| last modified | January 10, 2010 9:22:... |
| linked | False |
| location | D:\Software Development\Java\Demo\ |
| name | Demo |
| path | /Demo |
- Outline View:** Shows the class structure of Employee:

```
Employee  
  - empId : int  
  - empName : String  
  - getEmpID() : void  
  - getEmpName() : void
```
- Bottom Bar:** Problems, Javadoc, Declaration, Console. The Console tab shows: No consoles to display at this time.

Finding Methods and Fields

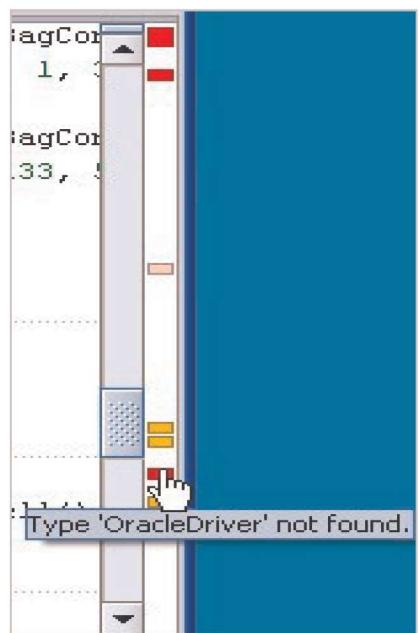
Find methods and fields using the Structure pane:



Supporting Code Development

- Improve code quality with Code Coach.
- Evaluate execution stack with the Execution Sample profiler.
- Examine heap memory usage with the Memory profiler.
- Analyze event occurrence and duration with the Event profiler for:
 - JVM events
 - Business components for Java events
 - Custom events

New Code Editor Features



Overview margin

Code Assist

```
private void jButton1ActionPerformed(ActionEvent evt) {
    Connection conn;
    try {
        conn = getConnection();
    } catch (SQLException sqle) {
        // TODO add better exception handling
        sqle.printStackTrace();
    }
}
```

Tasks list

| Done | Description | Prio... | File | Line |
|--------------------------|---------------------------------------|---------|----------------|------|
| <input type="checkbox"/> | add better exception handling | Low | MainFrame.java | 142 |
| <input type="checkbox"/> | Remember to do this before going home | Medium | MainFrame.java | 142 |

Tasks list

```
public MainFrame() {
    try {
        jbInit();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
public MainFrame() { ... }
```

Scope and code folding

```
public void setVisible(boolean b) {
    super.setVisible(b);
}

public void paintIcon(Component c, Graphics g, int x, int y) {
    implements method in javax.swing.Icon
}

public int getIconWidth() {
    return 0;
}
```

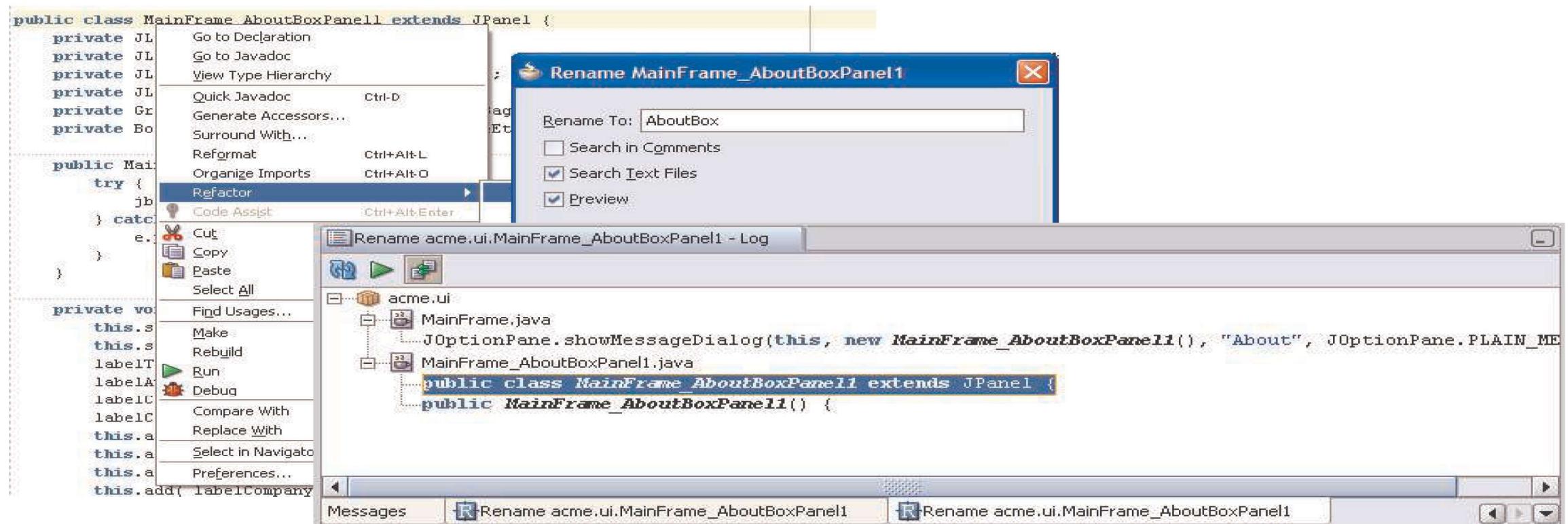
Implements and overrides
navigation

Customize the IDE:

- Look and feel
- General environment
- Dockable windows
- Component Palette
- Preset keymaps

Refactoring

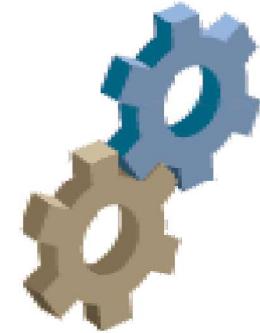
Modify the structure of code without changing its behavior (or breaking it).



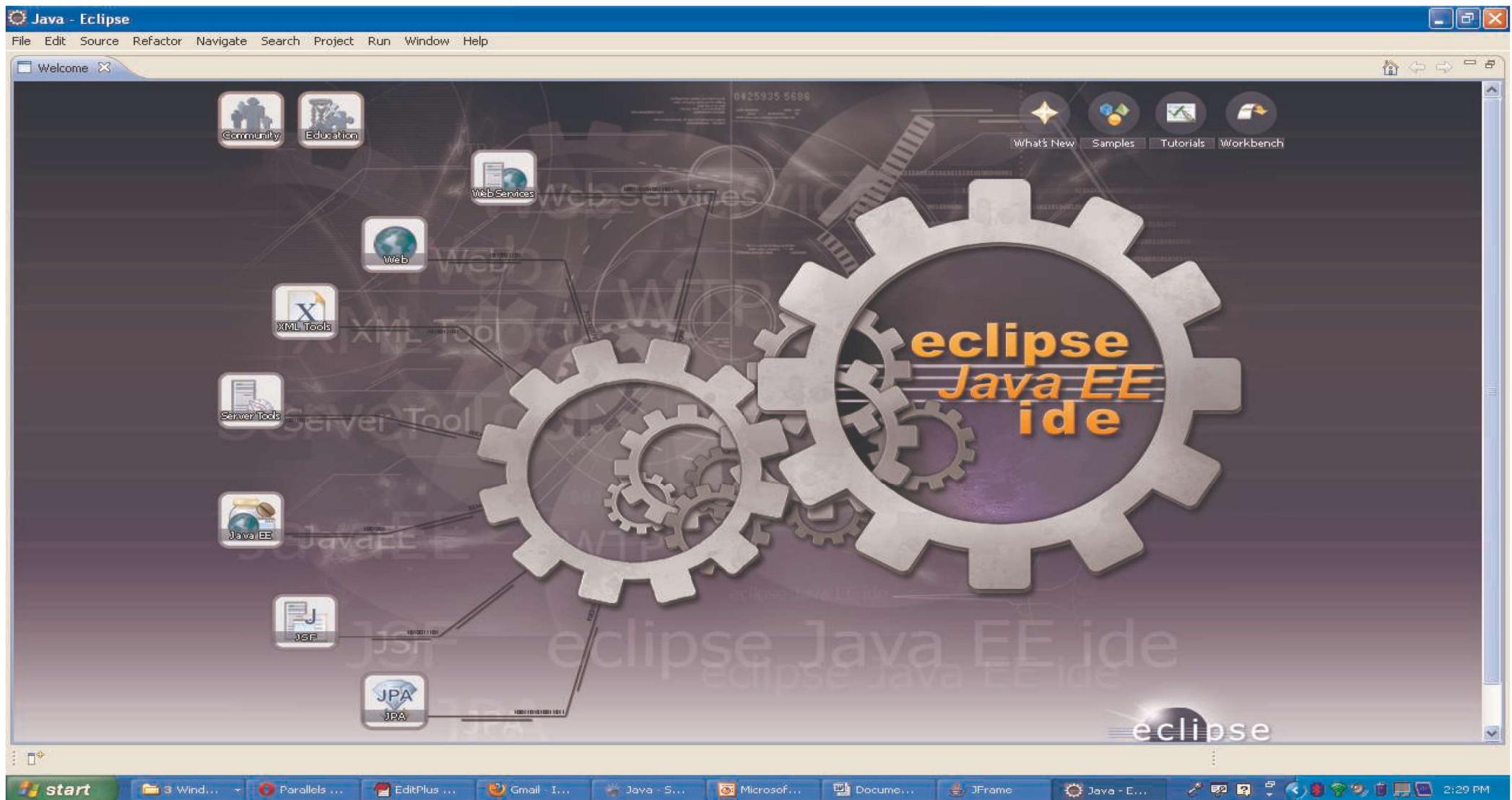
Refactoring

- Drag-and-drop refactoring
- Refactor across entire application
- Refactor across source control
- More than 35 new refactoring operations, including:

- **Rename Class**
- **Rename Field**
- **Rename Method**
- **Rename Package**
- **Rename Parameter**
- **Change Method Signature**
- **Introduce Variable**
- **Introduce Field**
- **Extract Interface**
- **Use Supertype Where Possible**
- **Move Class**
- **Duplicate Class**
- **Pull Members Up**
- **Safe Delete**

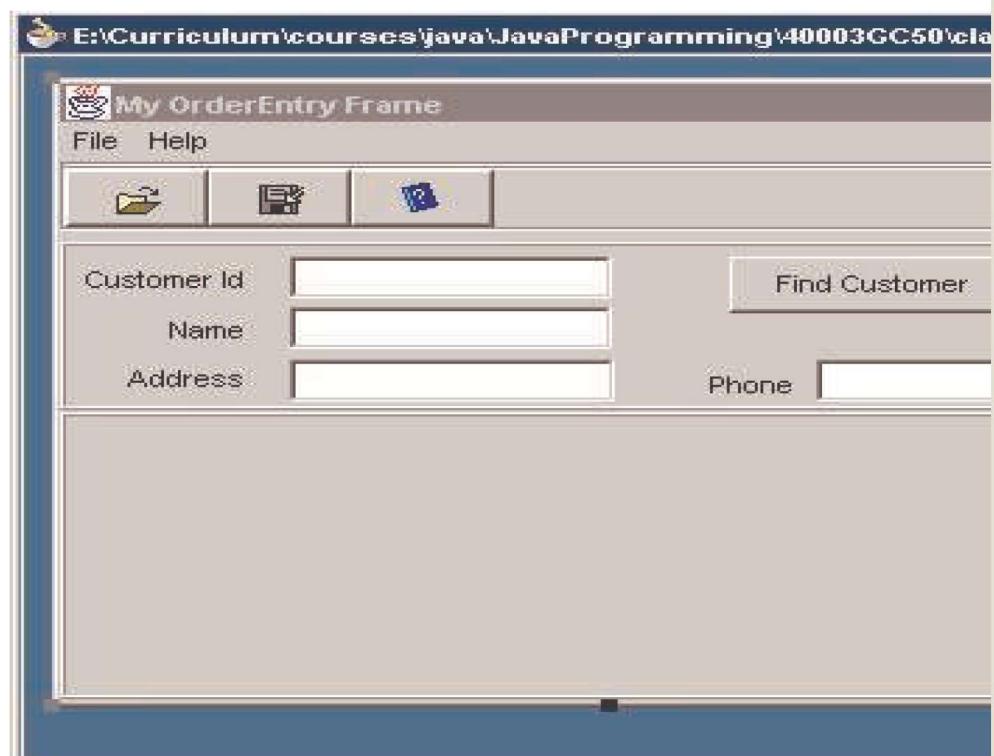


Eclipse Help System



Obtaining Help on a Topic

Use [F1] to invoke context-specific help.



Java UI Editor

Displays the visual components of a user interface in Editing mode.

When a Java UI Editor is open, its corresponding elements are displayed hierarchically in the Structure window. If the Property Inspector is open, selecting elements in either the Structure window or the Java UI Editor changes the selection in the Inspector as well.

The Java UI Editor displays a GUI hierarchy. If these are menu items, the hierarchy is displayed in one fashion; if these are nonmenu items, it is displayed in another. The mode of presentation differs, as the sort of editing that you are engaged in differs.

To open a hierarchy initially in the Java UI Editor, you have only to select a node in the Navigator and then right-click and choose **Edit**, or use the **View** menu. What displays in the editor is the entire GUI hierarchy for the `this` node. The method of display depends upon whether this hierarchy consists of menu or nonmenu items.

Help Content

A screenshot of the Java UI Editor window. The title bar says "Frame1.java". The main content area is titled "Java UI Editor". It contains a paragraph of text about the editor's function. Below that is another paragraph about displaying a GUI hierarchy. At the bottom, there is a "Help Content" button.

Eclipse Debugger

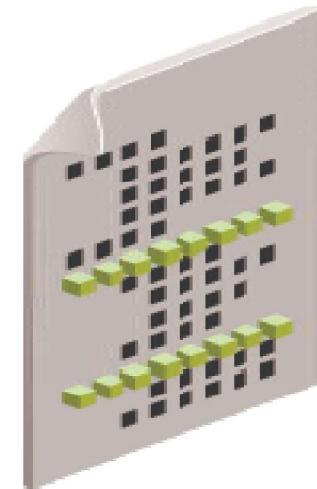
- Helps find and fix program errors:
 - Run-time errors
 - Logic errors
- Allows control of execution
- Allows examination of variables



Breakpoints

Setting breakpoints:

- Manage multiple breakpoints
- Manage conditional breakpoints
- Define columns displayed in window
 - Description
 - Type
 - Status
- Control scope of action
 - Global > Application > Project



Debugger Windows

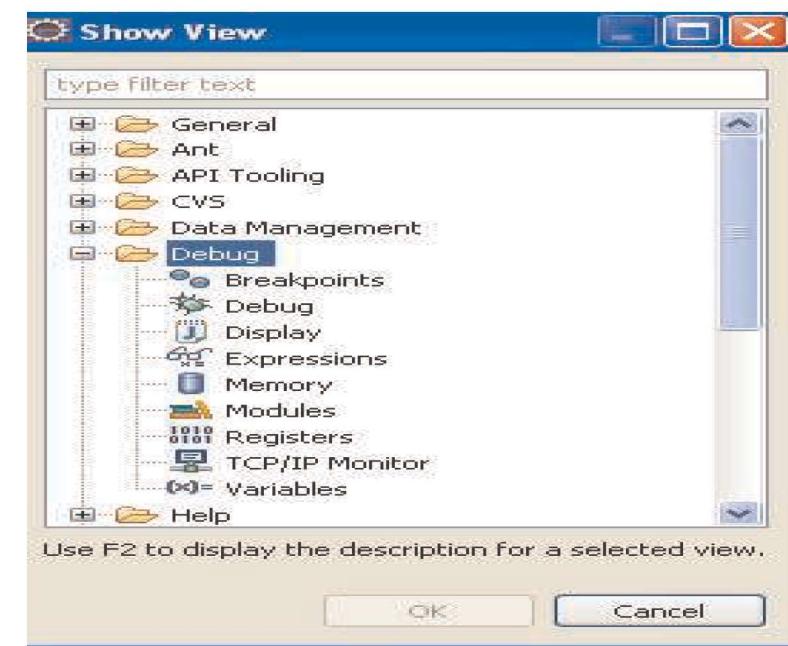
View debugging information:

- Classes: Displays list of loaded classes and status
- Watch: Evaluates and displays expressions
- Monitors: Displays information about active monitors
- Threads: Displays the names and statuses of all threads
- Smart Data: Analyzes source code near execution point
- ... and more

Stepping Through a Program

Use the buttons on the debugger toolbar:

- Start the debugger.
- Resume the program.
- Step over a method call.
- Step into a method call.
- Step out of a method call.
- Step to the end of the method.
- Pause execution.
- Stop the debugger.



Summary

In this lesson, you should have learned that:

- Eclipse builds, debugs, and runs all types of Java applications
- Eclipse can be used to develop:
 - Java applications
 - Java servlets
 - JSPs
 - EJBs
- Eclipse can be used to build enterprise applications



Practice : Overview

This practice covers the following topics:

- Exploring the Eclipse 3.x IDE
- Creating an application and a project
- Populating the project with existing files

