Introduction to Eclipse, Creating Eclipse plug-ins and the Overture editor

David Holst Møller
Engineering College of Aarhus

Agenda

- Part I – Introduction to Eclipse and Eclipse Plug-ins
- Part II – The Overture IDE

A bit of history

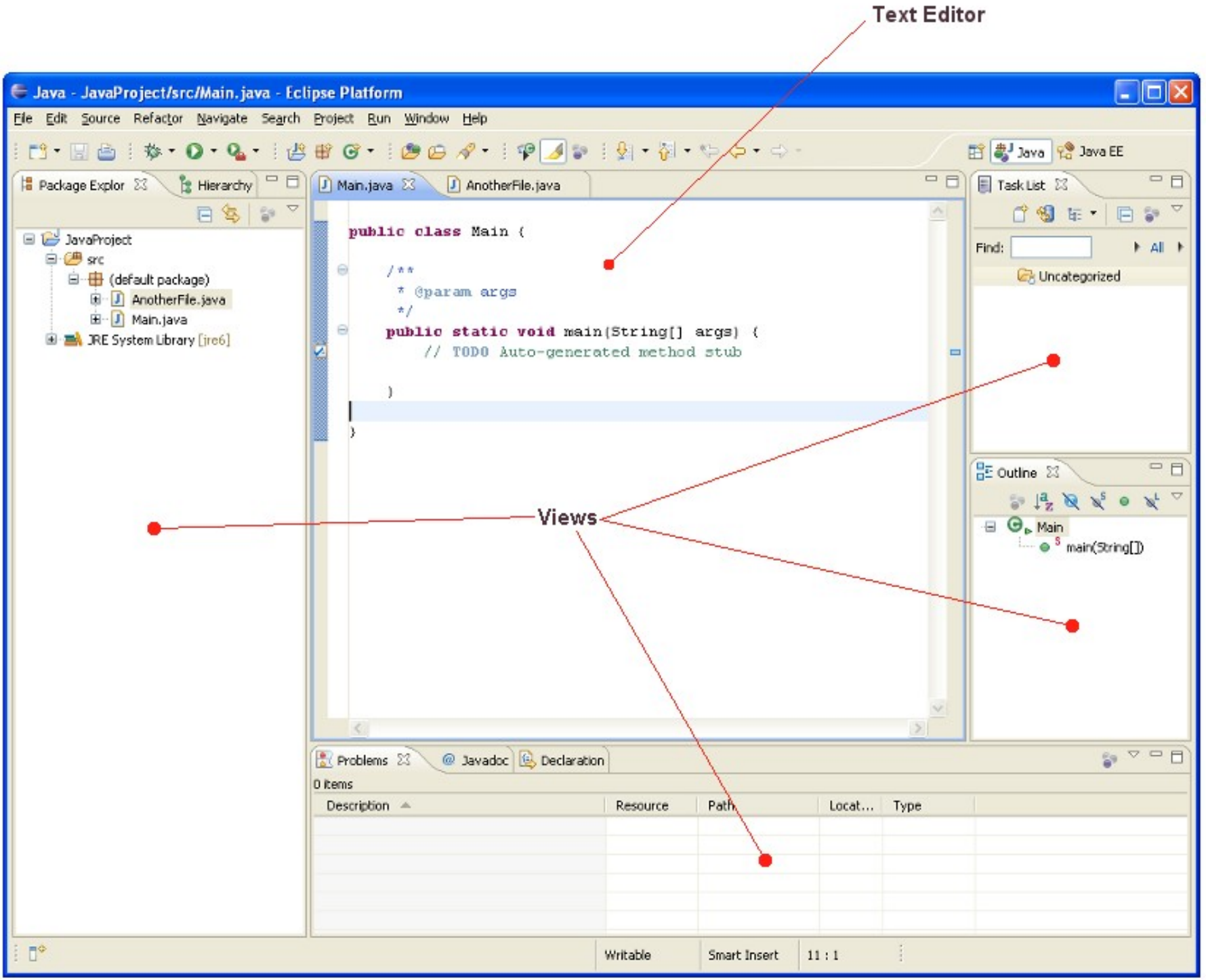
- Industry leaders formed the initial eclipse.org Board of Stewards in November 2001 (Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and Webgain)
- Originally an IBM project developed by OTI, the aim was to develop a platform which could be used for integrating all their tools in a common software base.
- In January 2004, the Eclipse Foundation was created.
- Annual release since 2006

Introduction to Eclipse

- An Integrated Development Environment(IDE)
- A Rich Client Platform
- Platform independent

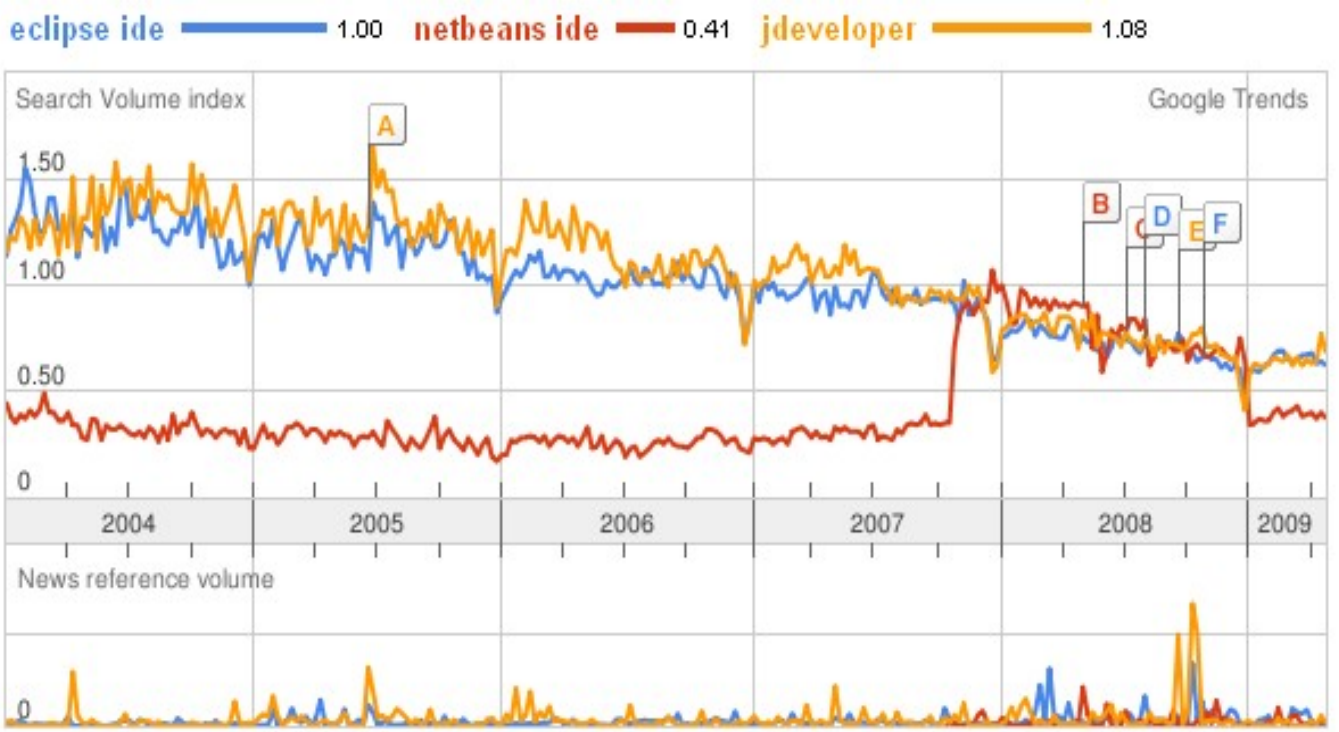
Terms

- Workbench
  - Resources
    - Projects
    - Folders
    - Files
  - Perspectives
    - Views
    - Editors


Text Editor
Java - JavaProject/src/Main.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
Package Explor
Hierarchy
JavaProject
src
(default package)
AnotherFile.java
Main.java
JRE System Library [jre6]
Main.java
AnotherFile.java
public class Main {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
}
}
Task List
Find:
All
Uncategorized
Outline
Main
main(String[])
Views
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Locat...
Type
Writable
Smart Insert
11 : 1

The Eclipse Java IDE

- Java development tooling (JDT)
- Competing with Netbeans (Sun) and Jdeveloper (Oracle)



Features in Java environment

- Outline
- Code Completion
- Team development (CVS - integrated)
- Refactoring
- Debugging
- Error
- Syntax
- Etc.

Eclipse Plug-in Architecture

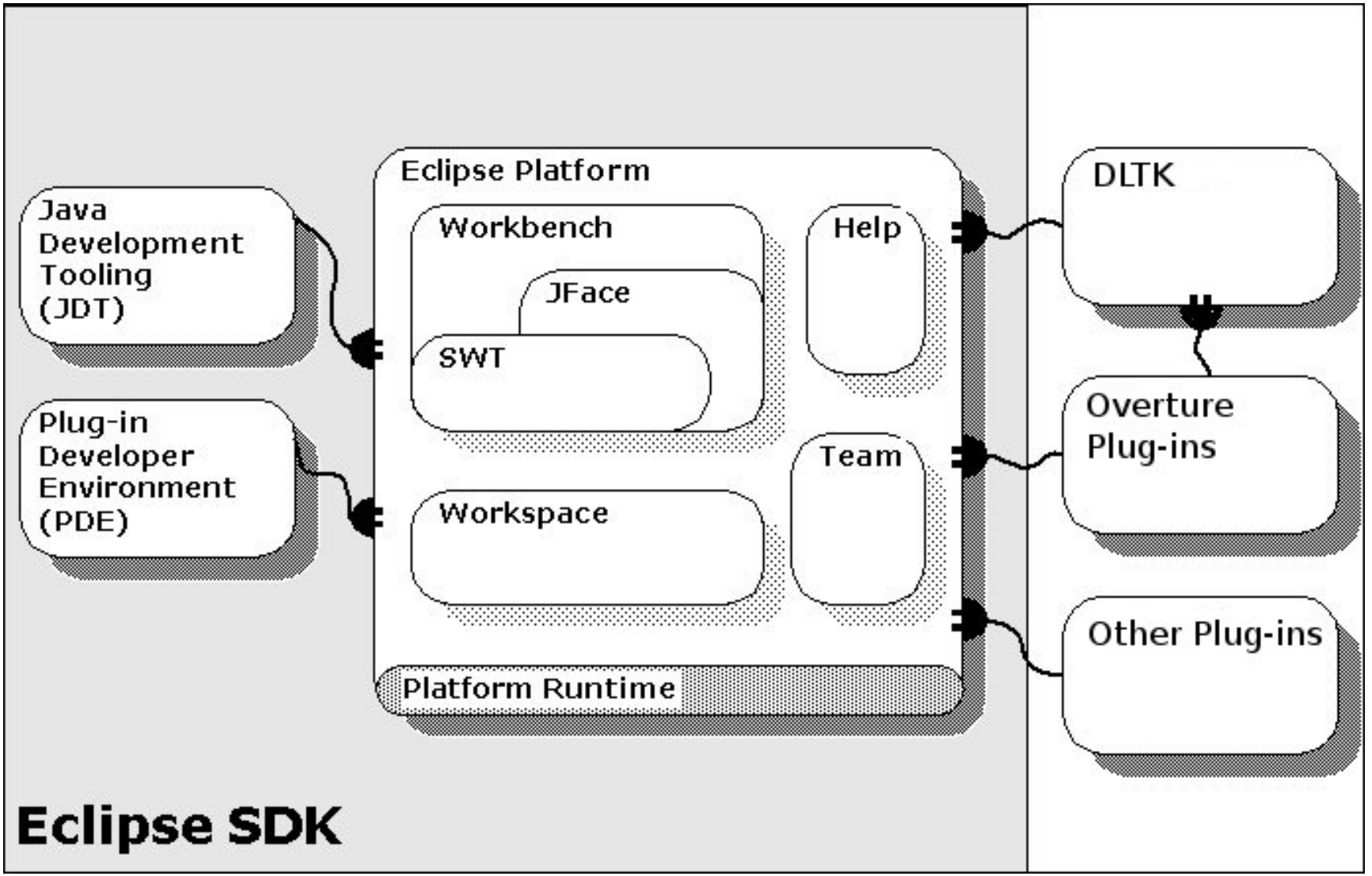
- Designed for plug-ins
- Far superior range of plug-ins.

Plug-in Terms 1

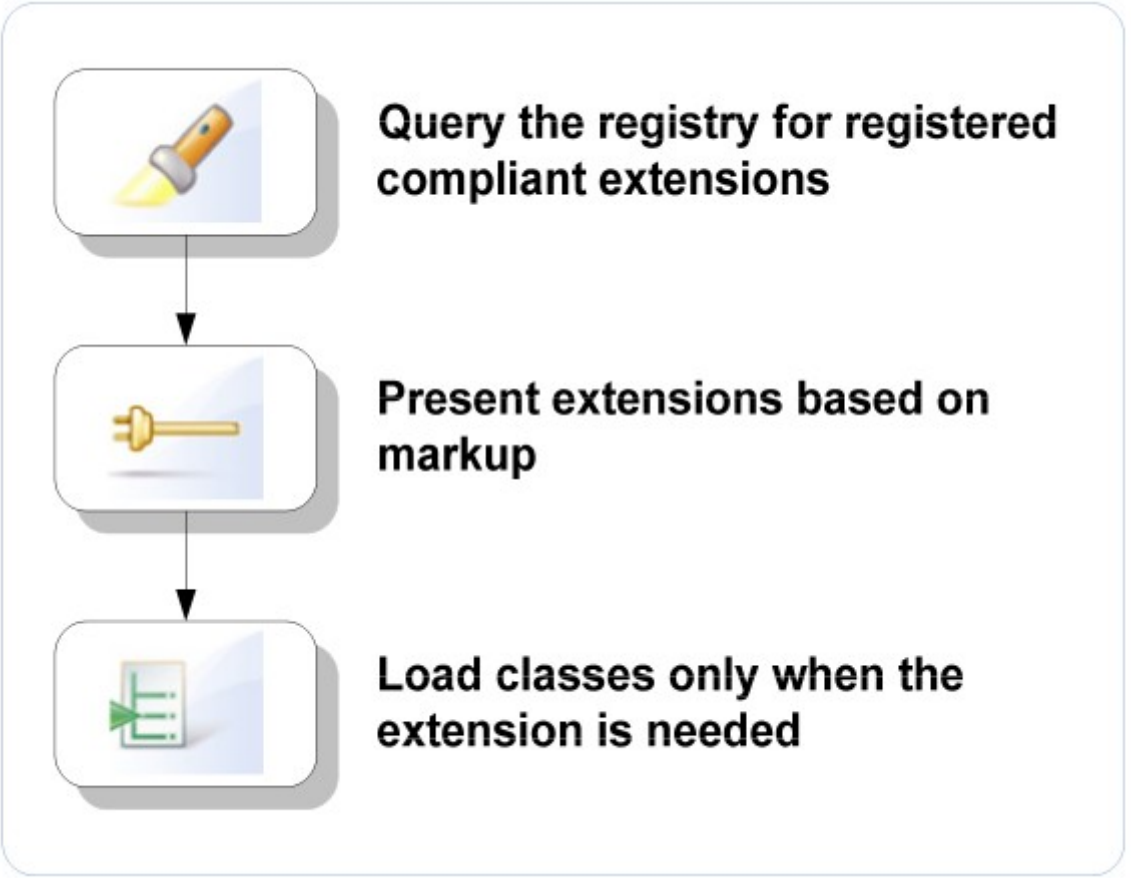
- A **plug-in** in Eclipse is a component that provides a certain type of service within the context of the Eclipse workbench.
- A **feature** is a way of grouping and describing different functionality that makes up a product. Grouping plug-ins into features allows the product to be installed and updated using the Eclipse update server and related support.

Plug-in Terms 2

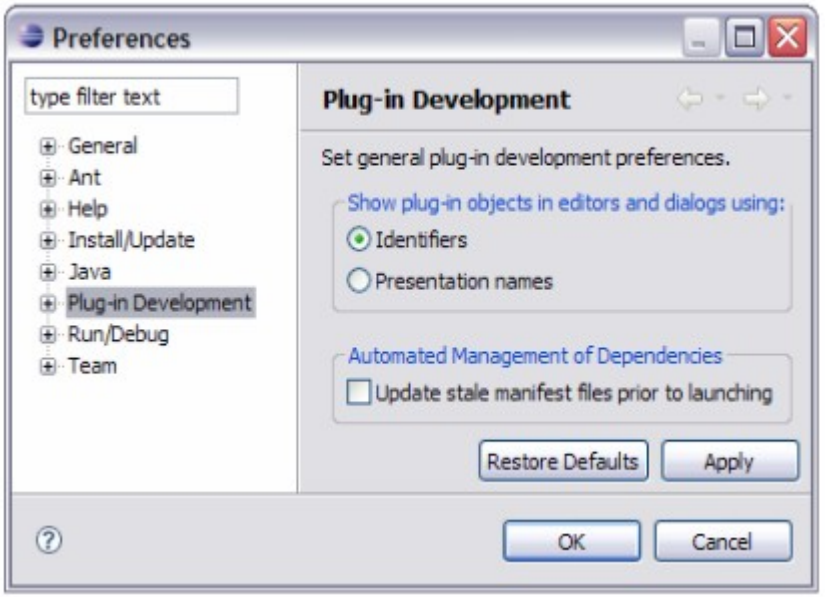
- **Extensions** are the central mechanism for contributing behaviour to the platform.
- **Extension points** define new function points for the platform that other plug-ins can plug into.
- Except for a small kernel known as the **Platform Runtime**, all of the Eclipse Platform's functionality is located in plug-ins.


Eclipse Platform
Java Development Tooling (JDT)
Plug-in Developer Environment (PDE)
Workbench
JFace
SWT
Help
Team
Workspace
Platform Runtime
DLTK
Overture Plug-ins
Other Plug-ins
Eclipse SDK

Extension loading

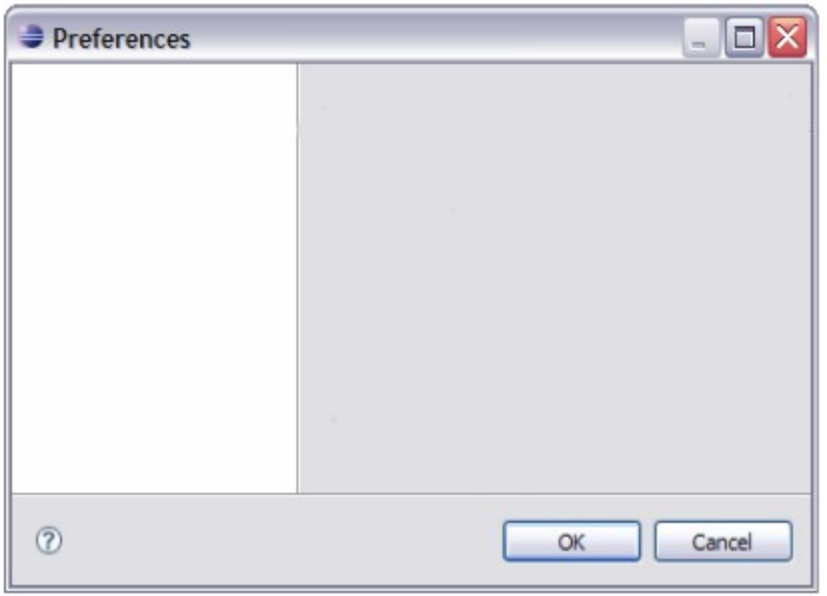


Example: Preference page



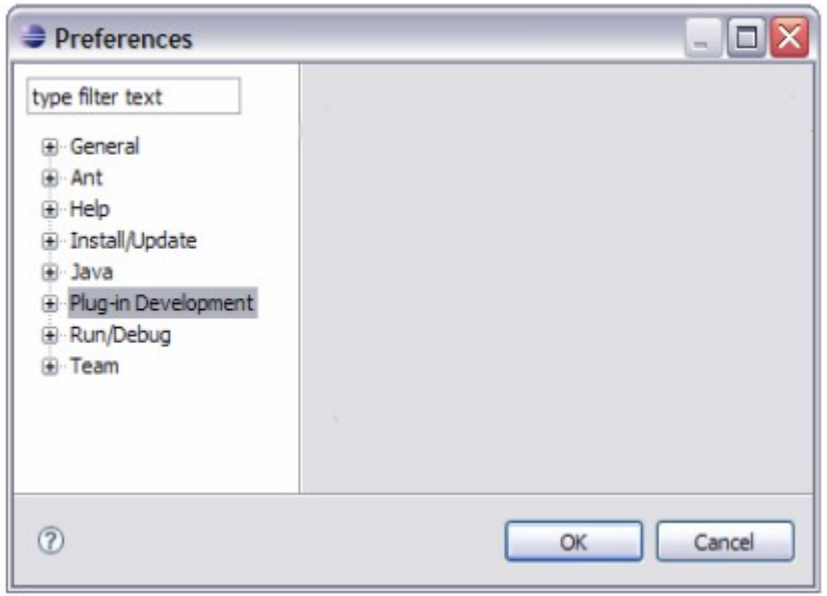
- Plug-ins may contribute preference pages
- All preference pages are assembled and categorized in the Preferences dialog
- How is the Preferences dialog created?
- How and when is a particular preference page created?

Create the Preferences Dialog - 1/3



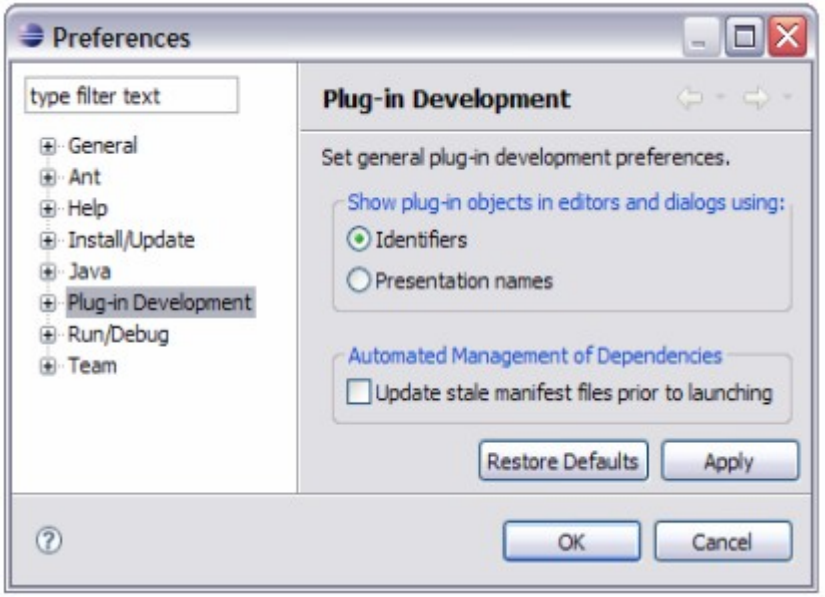
- The UI plug-in provides the org.eclipse.ui.preference Pages extension point
- The UI plug-in first creates an empty Preferences dialog
- Now the dialog needs to be populated...

Generate the Preference Page Index (2/3)



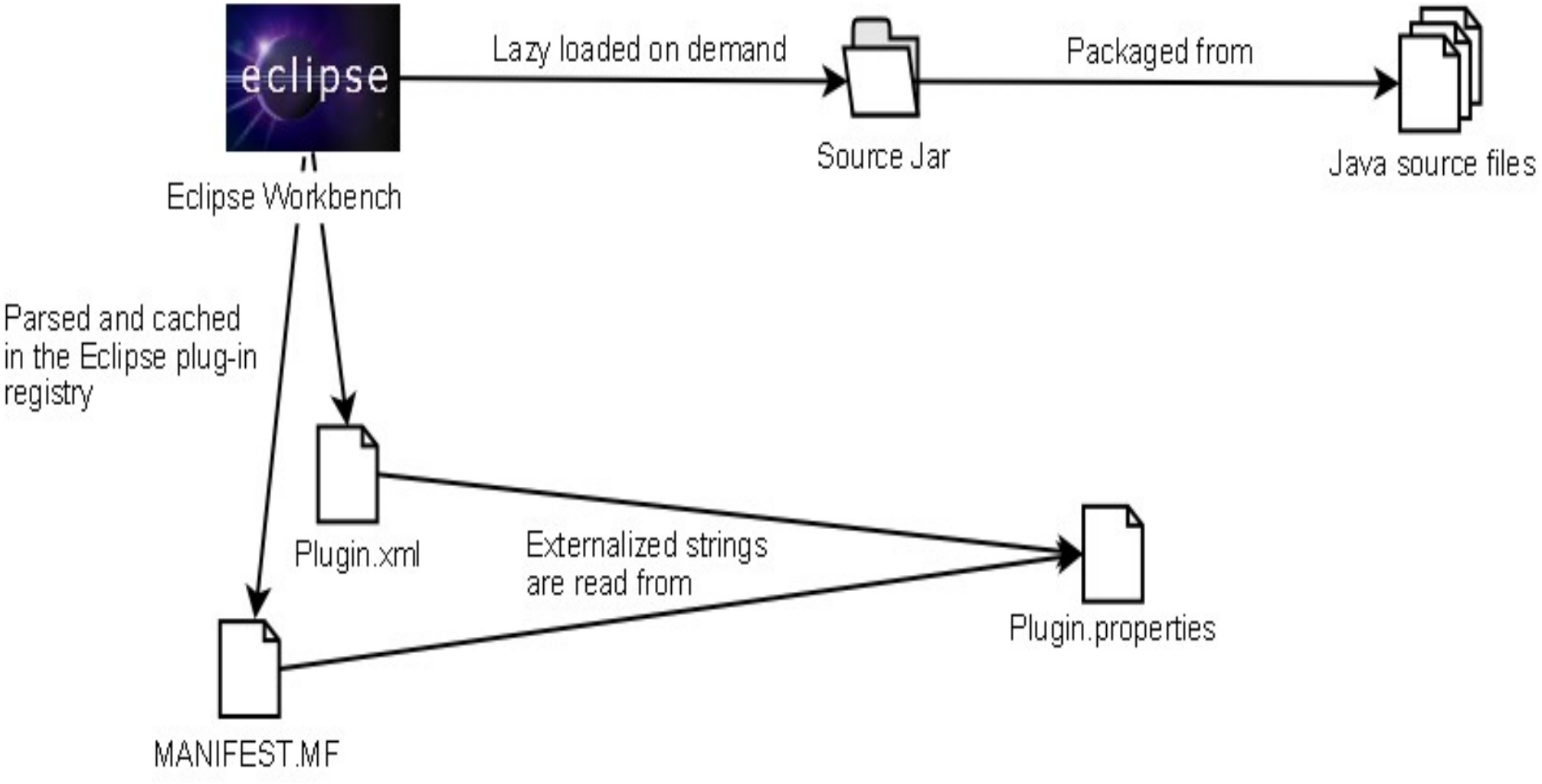
- The UI plug-in queries the extension registry for all *org.eclipse.ui.preferencePages extensions*
- The preference page index is then generated using the xml markup only:
  - Names for available preference pages are displayed in the tree using the *name* attribute
  - The *category* attribute is used to categorize the pages

Create the Plug-in Development Preference Page (3/3)



- When the Plug-in Development preference page gets selected, the UI plug-in asks the extension registry to load and instantiate the Java class specified by the class attribute of the corresponding extension
- The class gets loaded and the preference page gets created The plug-in providing that extension (i.e. the org.eclipse.pde.ui plug- in) may then get activated, if it's not already active

Defining Plug-ins



plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
...
        <extension point="org.eclipse.ui.perspectives">
                <perspective
                class="org.overturetool.ui.OverturePerspective"
                icon="icons/vdm16.png"
                id="org.overturetool.ui.OverturePerspective"
                name="%OverturePerspectiveName"/>
        </extension>
...
</plugin>
```

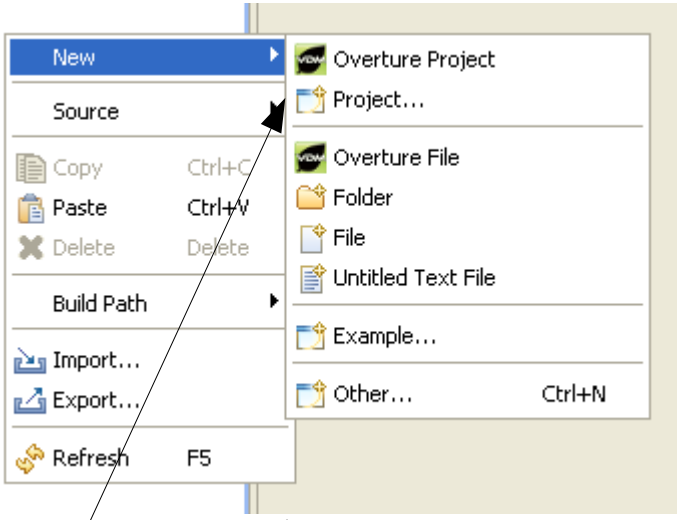
plugin.properties

```
OverturePerspectiveName=Overture
pluginName=Dynamic Languages Toolkit Overture UI
pluginProvider=Eclipse.org
...
```

MAINFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %pluginName
Bundle-SymbolicName: org.overturetool.ui; singleton:=true
Bundle-Version: 1.0.0.qualifier
Bundle-Activator: org.overturetool.internal.ui.UIPlugin
Bundle-Vendor: %pluginProvider
Bundle-Localization: plugin
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime,
 ...
Bundle-ActivationPolicy: lazy
Export-Package: org.overturetool.internal.ui,
 org.overturetool.internal.ui.editor,
 ...
Bundle-RequiredExecutionEnvironment: J2SE-1.5
```

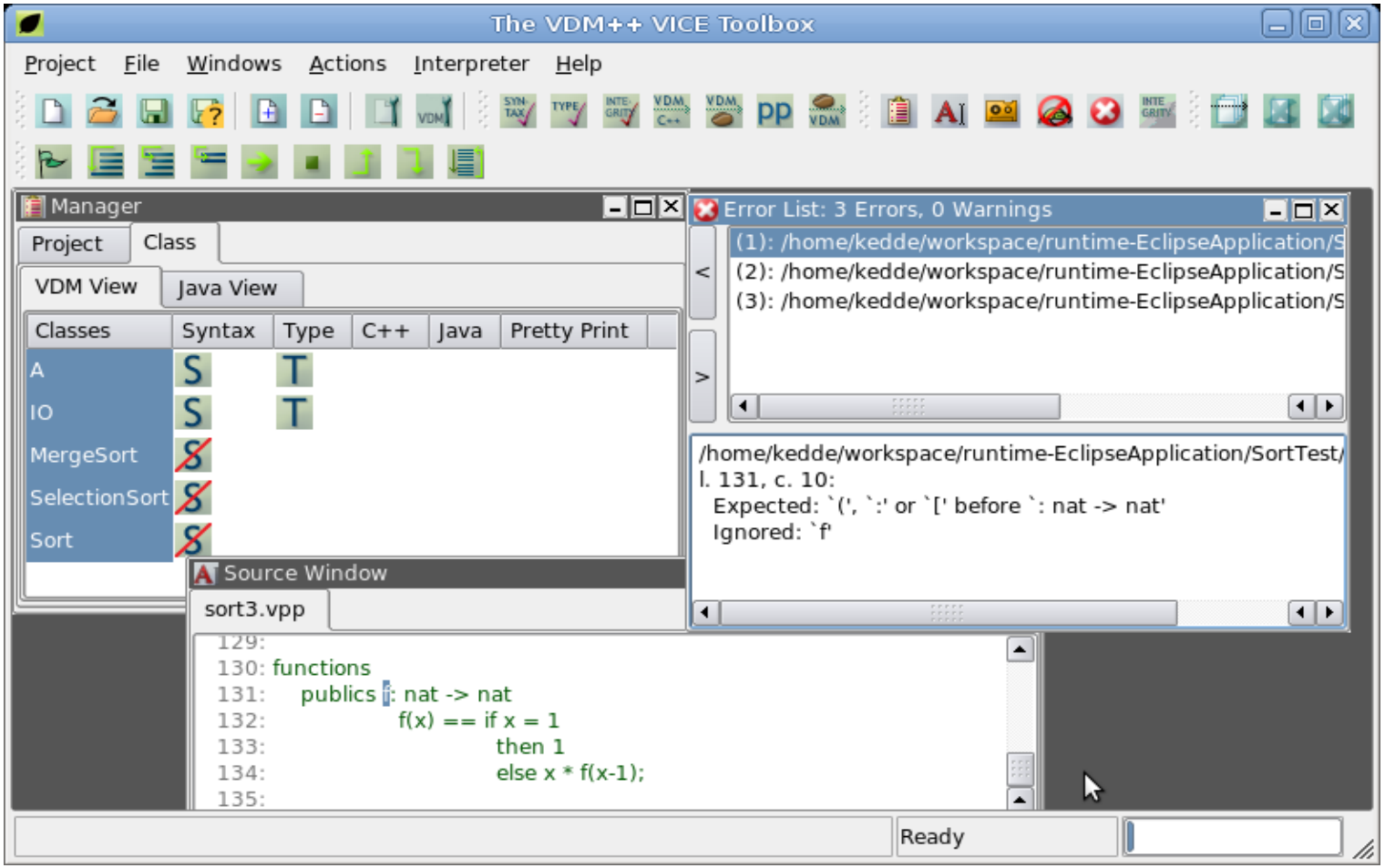
```
package org.overturetool.ui;

//IMPORTS...
public class OverturePerspective implements IPerspectiveFactory  {

    public void createInitialLayout(IPageLayout layout) {

        ....


        // views - standard workbench
        layout.addShowViewShortcut(IPageLayout.ID_OUTLINE)
        layout.addShowViewShortcut(IPageLayout.ID_PROBLEM_
        layout.addShowViewShortcut(IConsoleConstants.ID_CO
        layout.addShowViewShortcut(navigator);
        layout.addShowViewShortcut(IPageLayout.ID_TASK_LIS
        layout.addShowViewShortcut(IProgressConstants.PROGRESS_VIEW_ID);

        // new actions
        layout.addNewWizardShortcut(
            "org.overturetool.internal.ui.wizards.OvertureProjectWizard");
        layout.addNewWizardShortcut(
            "org.overturetool.internal.ui.wizards.OvertureFileCreationWizard");
        layout.addNewWizardShortcut(
            "org.eclipse.ui.wizards.new.folder");
        layout.addNewWizardShortcut(
            "org.eclipse.ui.wizards.new.file");
        layout.addNewWizardShortcut(
            "org.eclipse.ui.editors.wizards.UntitledTextFileWizard");
    }
}
```



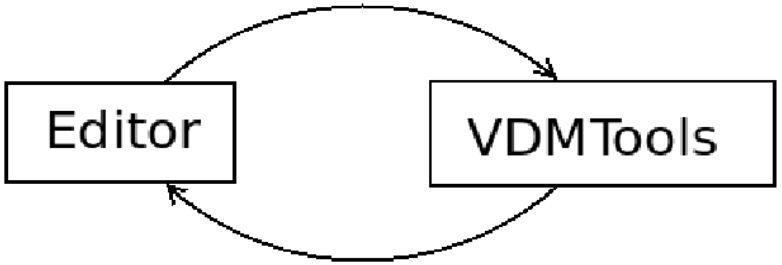
The Overture IDE

VDM development today

- No integration of interpreter and editor



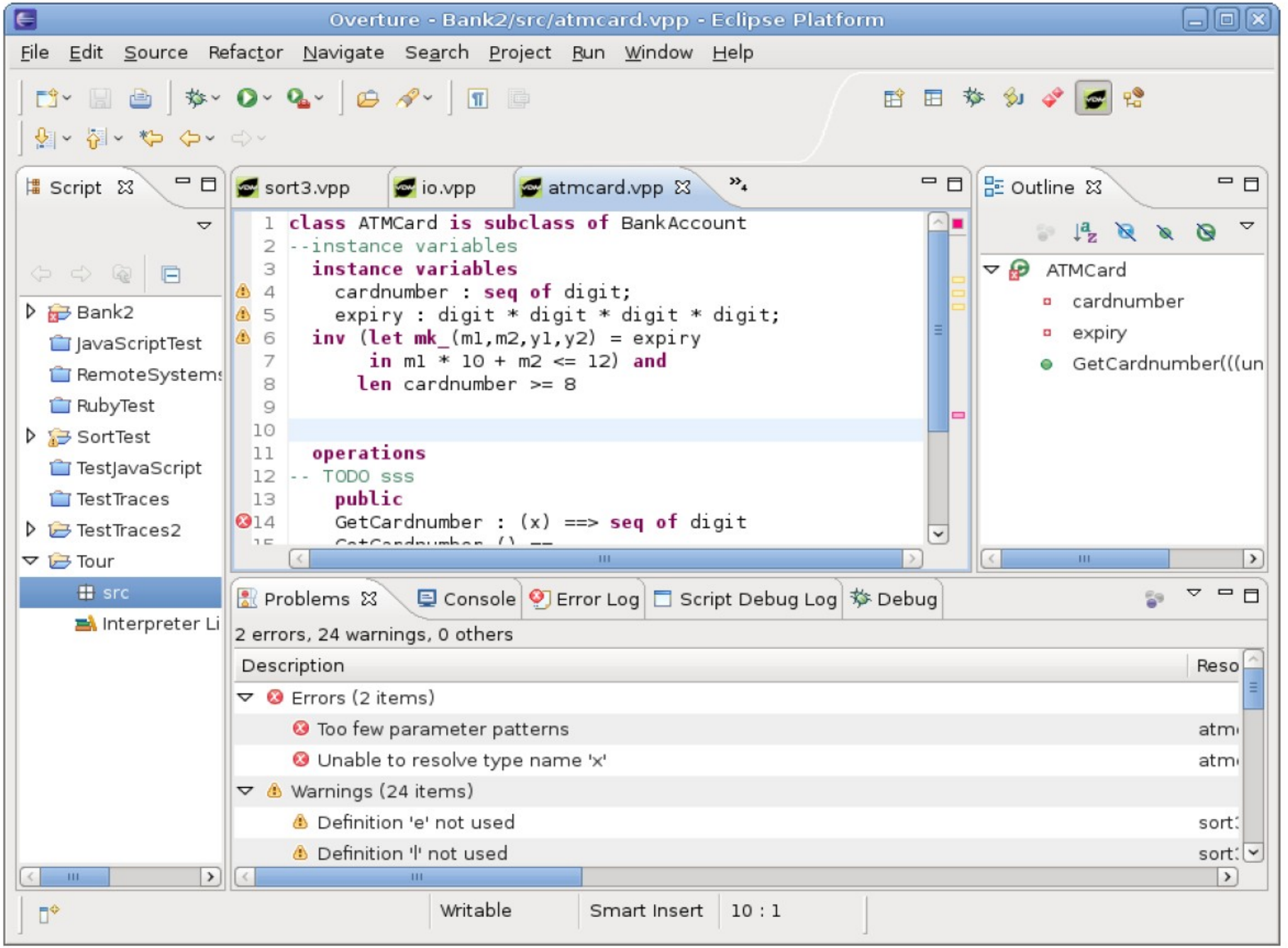
- Cumbersome development process
- Poor navigation
- No intellisense
- No support for refactoring
- Debugging support is good, but it could be better
- Test coverage measurement – but rather bothersome
- Support for Code Generation
- Support for UML→VDM→UML

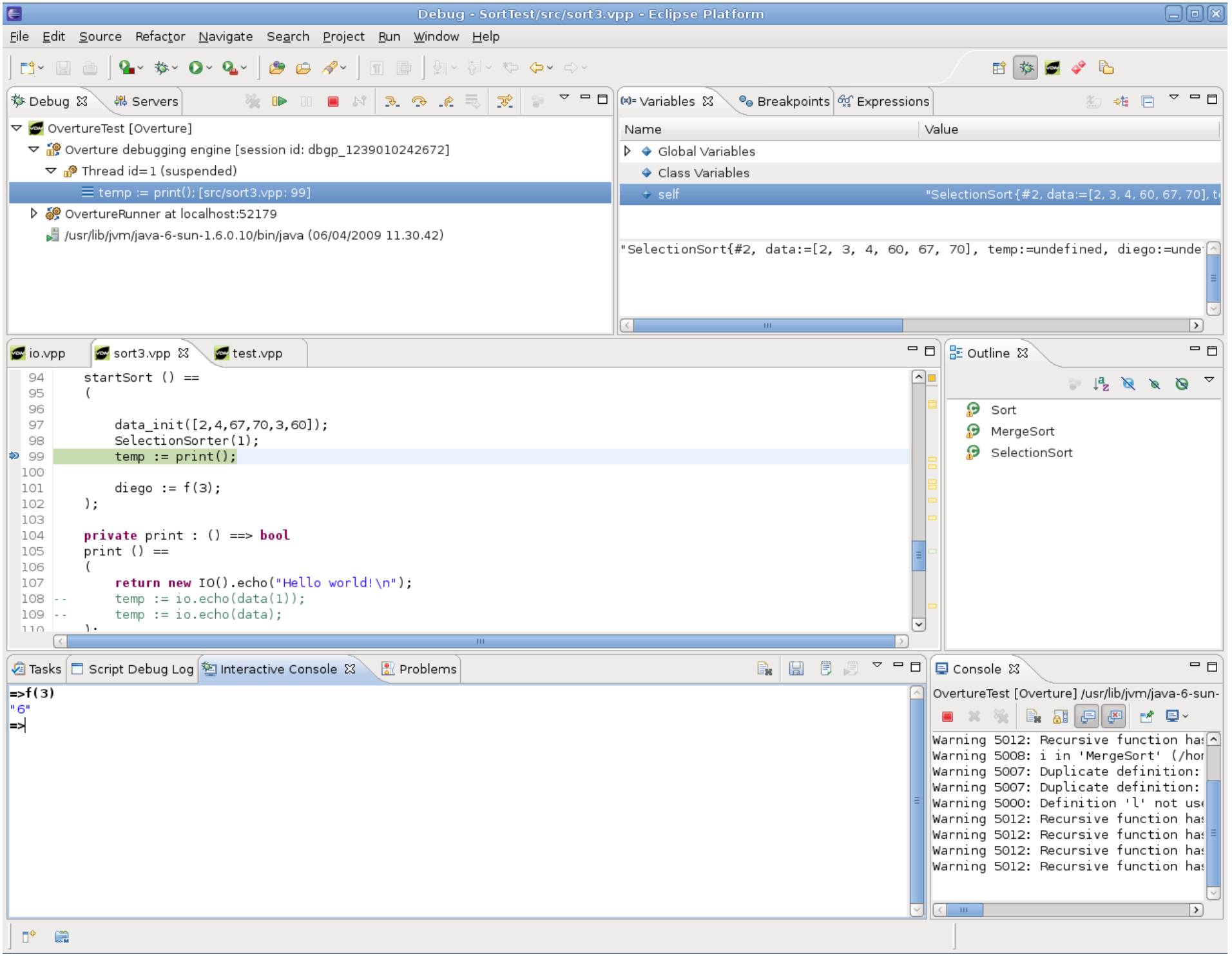


Vision for the Overture IDE

- Integrated Development Environment
- The IDE should offer features know from other IDEs. Such as:
  - Syntax highlighting
  - Easy navigation
    - Files, definitions, errors, warnings
  - Refactoring
  - Advanced debugging features
- Test coverage measurement
- Code generation and UML to and from VDM
- High Extensibility
- The only tool needed for all VDM development - including all dialects

Introduction to the Overture Editor

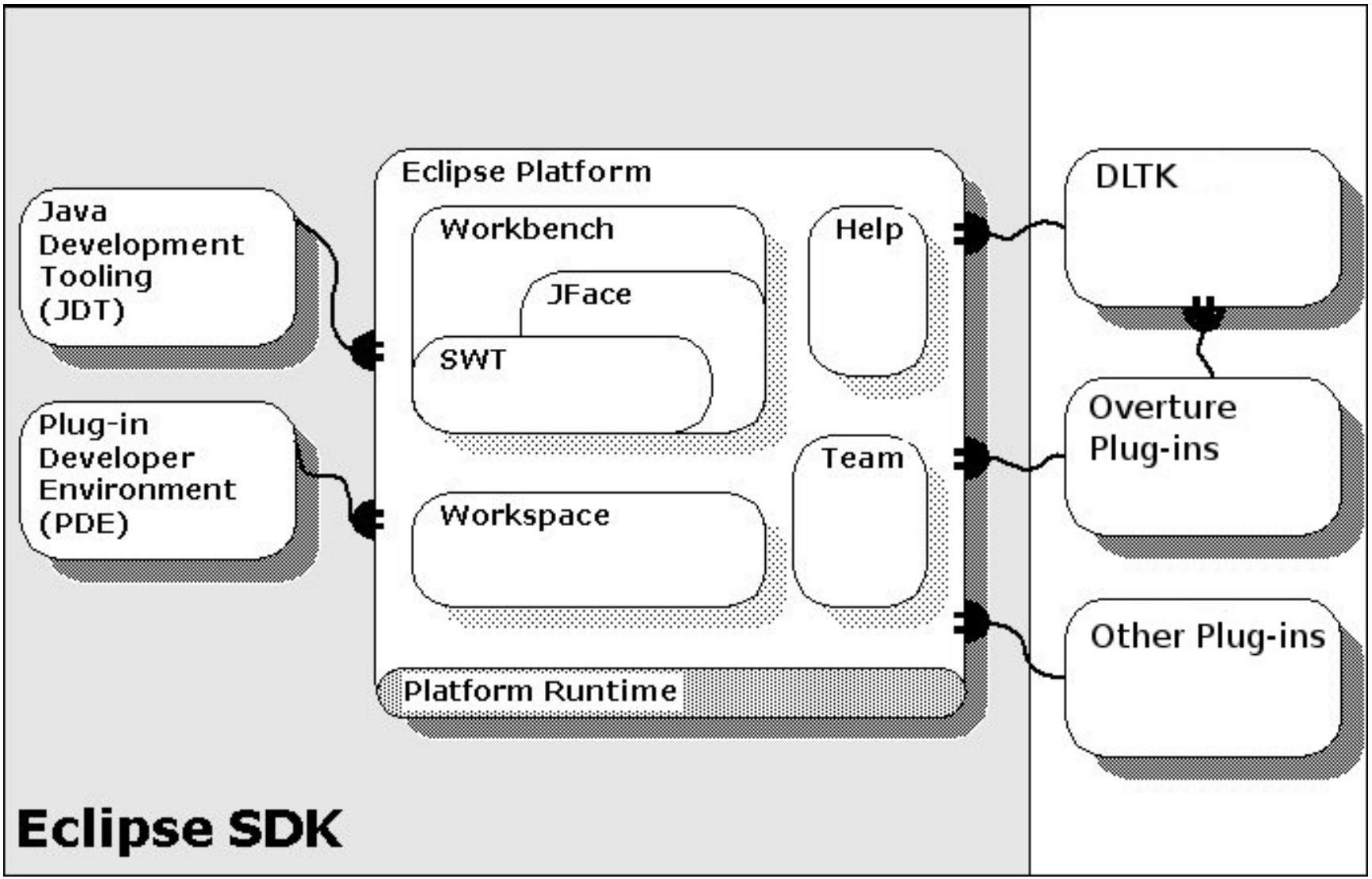

Overture - Bank2/src/atmcard.vpp - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Script
Bank2
JavaScriptTest
RemoteSystems
RubyTest
SortTest
TestJavaScript
TestTraces
TestTraces2
Tour
src
Interpreter Li
sort3.vpp
io.vpp
atmcard.vpp
1 class ATMCard is subclass of BankAccount
2 --instance variables
3 instance variables
4 cardnumber : seq of digit;
5 expiry : digit * digit * digit * digit;
6 inv (let mk_(m1,m2,y1,y2) = expiry
7 in m1 * 10 + m2 <= 12) and
8 len cardnumber >= 8
9
10
11 operations
12 -- TODO sss
13 public
14 GetCardnumber : (x) ==> seq of digit
Outline
ATMCard
cardnumber
expiry
GetCardnumber(((un
Problems
Console
Error Log
Script Debug Log
Debug
2 errors, 24 warnings, 0 others
Description
Reso
Errors (2 items)
Too few parameter patterns
atm
Unable to resolve type name 'x'
atm
Warnings (24 items)
Definition 'e' not used
sort
Definition 'l' not used
sort
Writable
Smart Insert
10 : 1


Debug - SortTest/src/sort3.vpp - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Servers
OvertureTest [Overture]
Overture debugging engine [session id: dbgp_1239010242672]
Thread id=1 (suspended)
temp := print(); [src/sort3.vpp: 99]
OvertureRunner at localhost:52179
/usr/lib/jvm/java-6-sun-1.6.0.10/bin/java (06/04/2009 11.30.42)
Variables
Breakpoints
Expressions
Name
Value
Global Variables
Class Variables
self
"SelectionSort{#2, data:=[2, 3, 4, 60, 67, 70], temp:=undefined, diego:=unde
io.vpp
sort3.vpp
test.vpp
94 startSort () ==
95 (
96
97 data_init([2,4,67,70,3,60]);
98 SelectionSorter(1);
99 temp := print();
100
101 diego := f(3);
102);
103
104 private print : () ==> bool
105 print () ==
106 (
107 return new IO().echo("Hello world!\n");
108 -- temp := io.echo(data(1));
109 -- temp := io.echo(data);
Outline
Sort
MergeSort
SelectionSort
Tasks
Script Debug Log
Interactive Console
Problems
=>f(3)
"6"
=>
Console
OvertureTest [Overture] /usr/lib/jvm/java-6-sun-
Warning 5012: Recursive function has
Warning 5008: i in 'MergeSort' (/hor
Warning 5007: Duplicate definition:
Warning 5007: Duplicate definition:
Warning 5000: Definition 'l' not use
Warning 5012: Recursive function has
Warning 5012: Recursive function has
Warning 5012: Recursive function has
Warning 5012: Recursive function has

Vision for the Overture IDE revisited

- Integrated Development Environment 
- The IDE should offer features know from other IDEs. Such as:
  - Syntax highlighting 
  - Easy navigation 
    - Files, definitions  , errors, warnings
  - Refactoring 
  - Advanced debugging features 
- Test coverage measurement 
- Code generation and UML to and from VDM 
- High Extensibility 
- The only tool needed for all VDM development - including all dialects 

Overture IDE Implementation



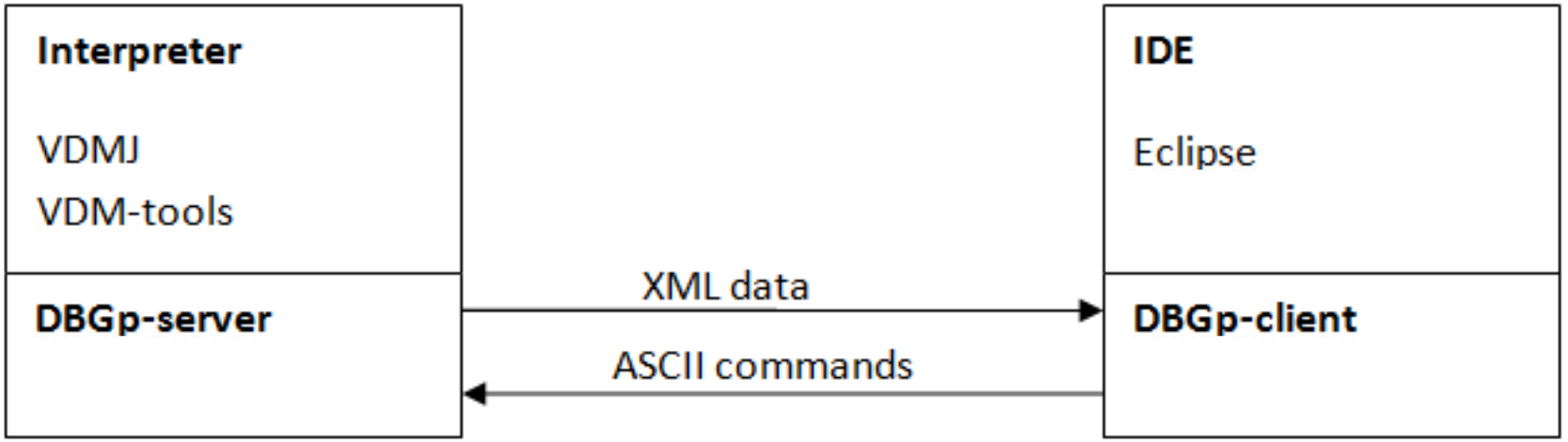
DLTK

Documentation

The project has no documentation.

Debug Protocol

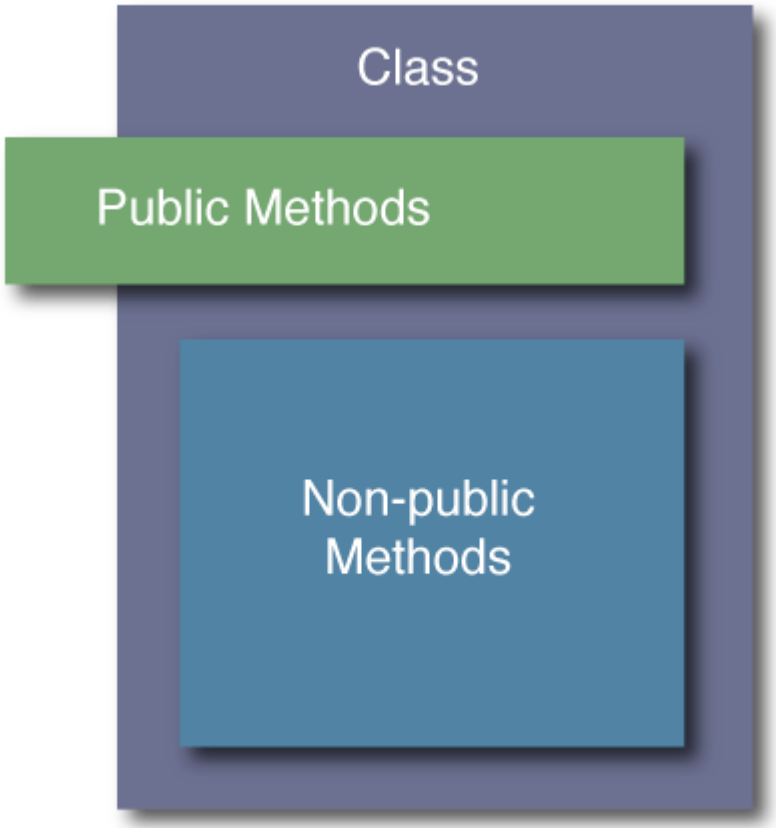
- A common debugger protocol for languages and debugger UI communication
- Part of the DLTK

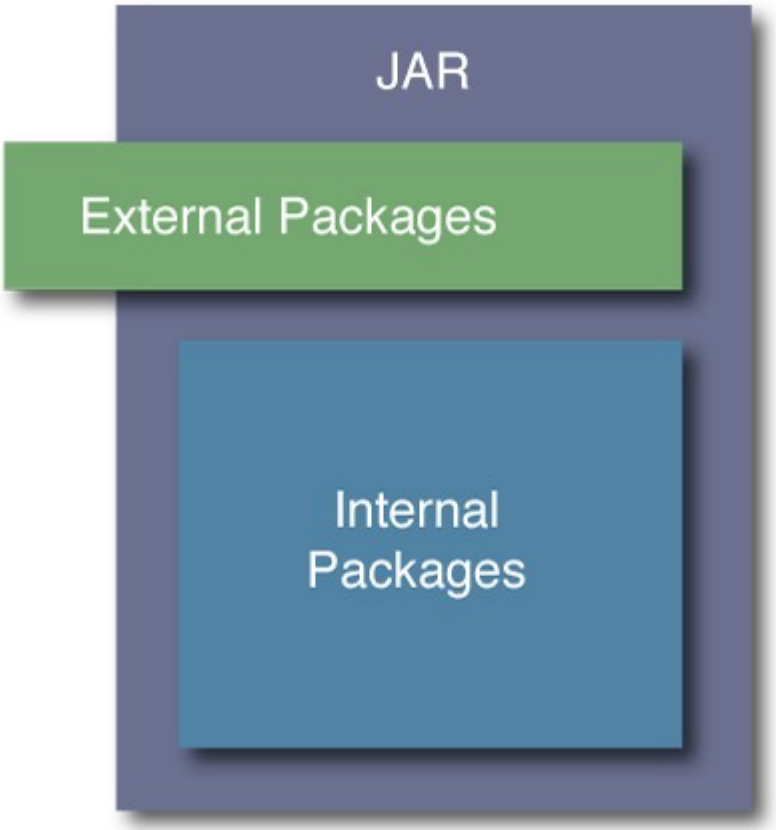


Extras...


org.overturetool.eclipse.plugins.debug
org.overturetool.eclipse.plugins.debug.ui
org.overturetool.eclipse.plugins.editor.core
org.overturetool.eclipse.plugins.editor.overturedebugger
org.overturetool.eclipse.plugins.editor.ui
org.overturetool.eclipse.plugins.launching
org.overturetool.eclipse.plugins.showtrace.core
org.overturetool.eclipse.plugins.stdlib
org.overturetool.eclipse.plugins.traces
org.overturetool.eclipse.plugins.traces.core
org.overturetool.eclipse.plugins.umltrans
org.overturetool.eclipse.plugins.umltrans.core

OSGi





object-oriented (instead of procedural)

Modular (as opposed to unmodular)