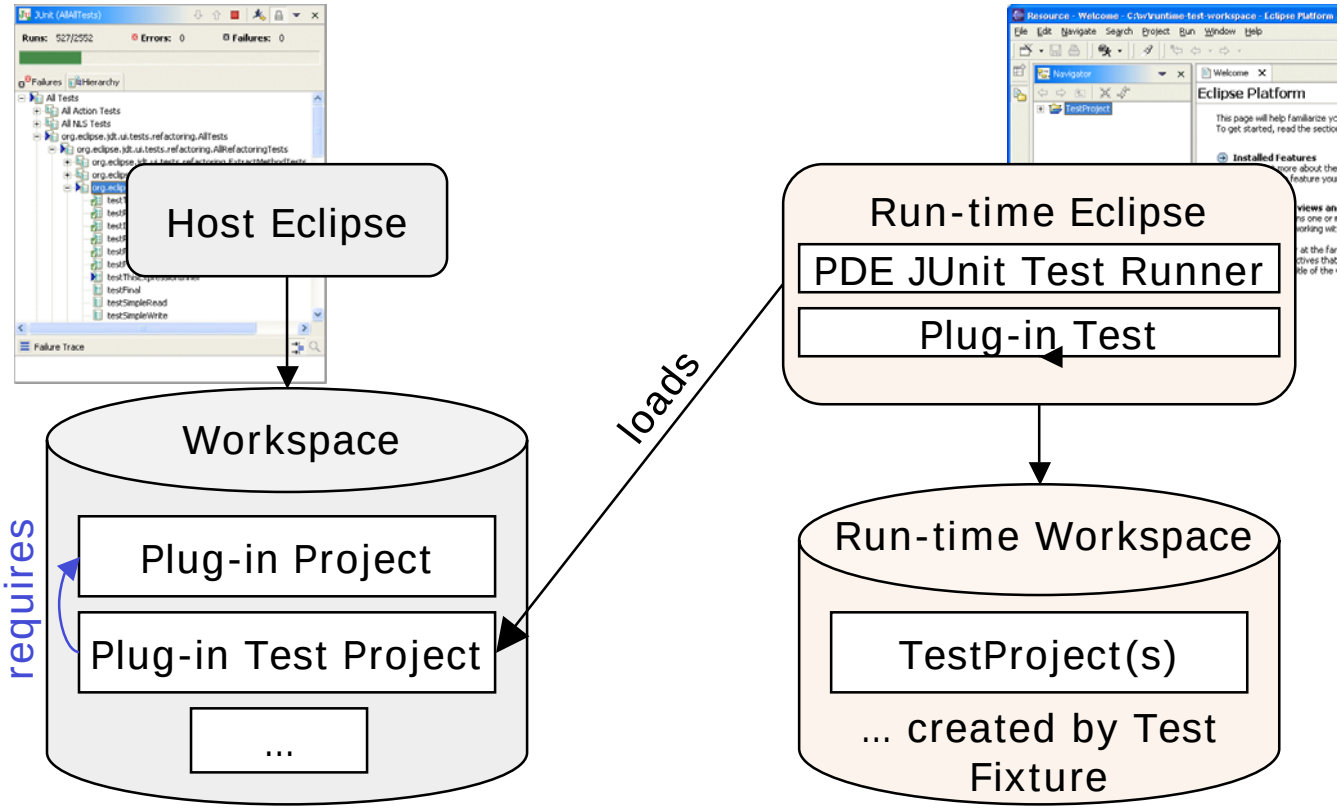
Eclipse Plug-in Testing

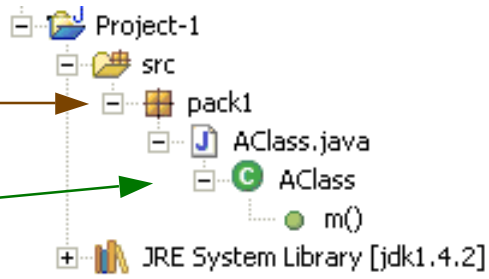
Markus Keller
IBM OTI Labs Zurich

- What is special with Eclipse?
  - TestCases located in **Plug-ins**
  - Test target is a **second Eclipse instance** with its own workspace
  - Special JUnit Plug-in test launcher: **Run > Run As > JUnit Plug-in test**



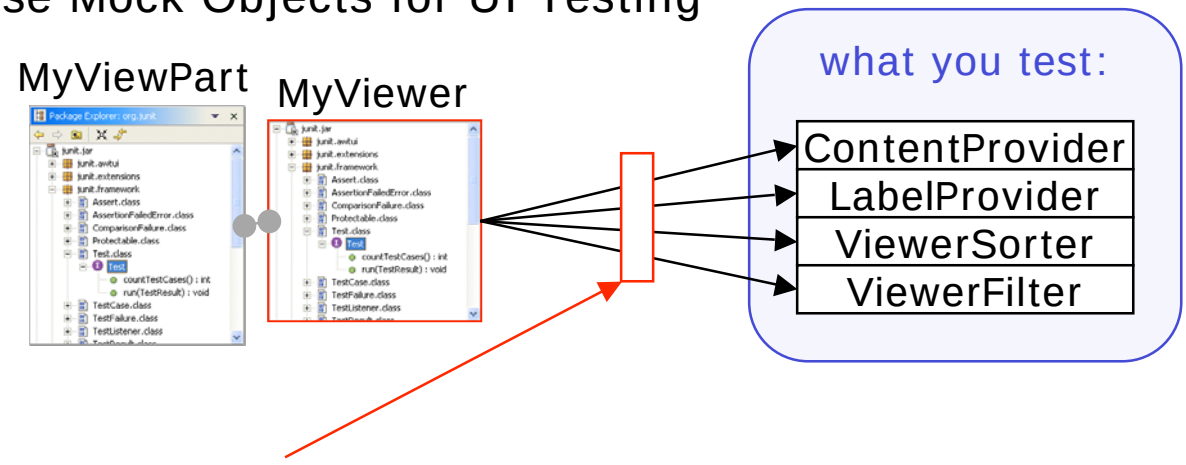
- Test Project Fixture
  - Run-time workspace initially empty
  - Must be populated with test-specific contents
    - For JDT Tests: create Java Projects, packages, and CUs
  - Use a helper class [1] :

```
TestProject prj = new TestProject();
IPackageFragment pack =
      prj.createPackage("pack1");
IType type = prj.createType(
      package, "AClass.java",
      "public class AClass {"
       + "public void m(){}}";
```



➢ JavaProjectHelper – in org.eclipse.jdt.ui.tests

- Tips & Tricks for Plug-in Testing
  - Use Mock Objects for UI Testing



- MockViewer extends MyViewer and observes interaction between provider and viewer

➢ MockPluginView – in org.eclipse.jdt.ui.tests

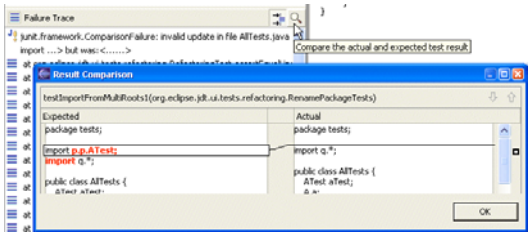
[1] E. Gamma, K. Beck. *Contributing to Eclipse – Principles, Patterns, and Plug-ins.* ➢ TestProject Fixture

- Tips & Tricks for Plug-in Testing (2)
  - Lazy loading: detect premature plug-in activation

    - ➢ JavaActivationTest – plug-in org.eclipse.jdt.ui.tests

  - Leak Tests
    - Create instance
    - Release reference to structure
    - Check that garbage collected
      - GC not enforceable
      - Wrote a DLL which collects all reachable objects
      - Compare before-after dumps to find “forgotten” links
    - ➢ JavaLeakTest – in org.eclipse.jdt.ui.tests

  - Code manipulation tests
    - Example: managing source files for refactoring tests
    - Ease complex test fixture creation
    - Need to compare outcome (expected vs. actual)
      - Failing tests: can use new JUnit Compare Viewer



    - Self-contained test methods (CUs from StringBuffers)
      - Easy to understand and duplicate
      - Can’t be executed or debugged directly
    - Store files & folders in “in” and “out” directories (per test)
      - Can easily copy-paste a test into a run-time eclipse to debug it

    - ➢ org.eclipse.jdt.ui.tests.refactoring

- Tips & Tricks for Plug-in Testing (3)
  - How to deal with text selections
    - Example: specifying range for extract method tests
    - Positions in testMethod:

```
testMethod() { … setSelection(rowA, colA, rowB, colB); … }
```

      - Each source file change requires manual update in test class
    - Comments surround selection:

```
public void /*[*/methodToRename/*]*/ ( ... ) { ... }
```

      - Interferes with formatting
      - Can't set caret *into* an identifier (e.g. for rename tests)

- Additional Testing Techniques [2]
  - Sequence Tests ("Log String")
    - Record chronology of notifications in a log
    - Check whether hooks are called in the right order
      - Example: Refactoring#checkActivation() is always called before Refactoring#createChange()
    - Compare collected log with expected
  - Self Shunt
    - Make your TestCase a listener and register it at the event source
      - In the event handler, change a variable in the TestCase
    - Run the event-generating code
    - Check that the variable is really changed

[2] K. Beck. *Test-Driven Development – By Example.*