HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Einführung in die Entwicklungsumgebung

Tutorium
im Rahmen des Softwaretechnik- & Software-Praktikums 2012

Inhalt

1. Was ist Eclipse?
2. Java-Programmierung in Eclipse
3. Eclipse erweitern durch Plug-ins
4. Debugging und Testen
5. Installation von Eclipse
6. Literatur und Referenzen

1. Was ist Eclipse?

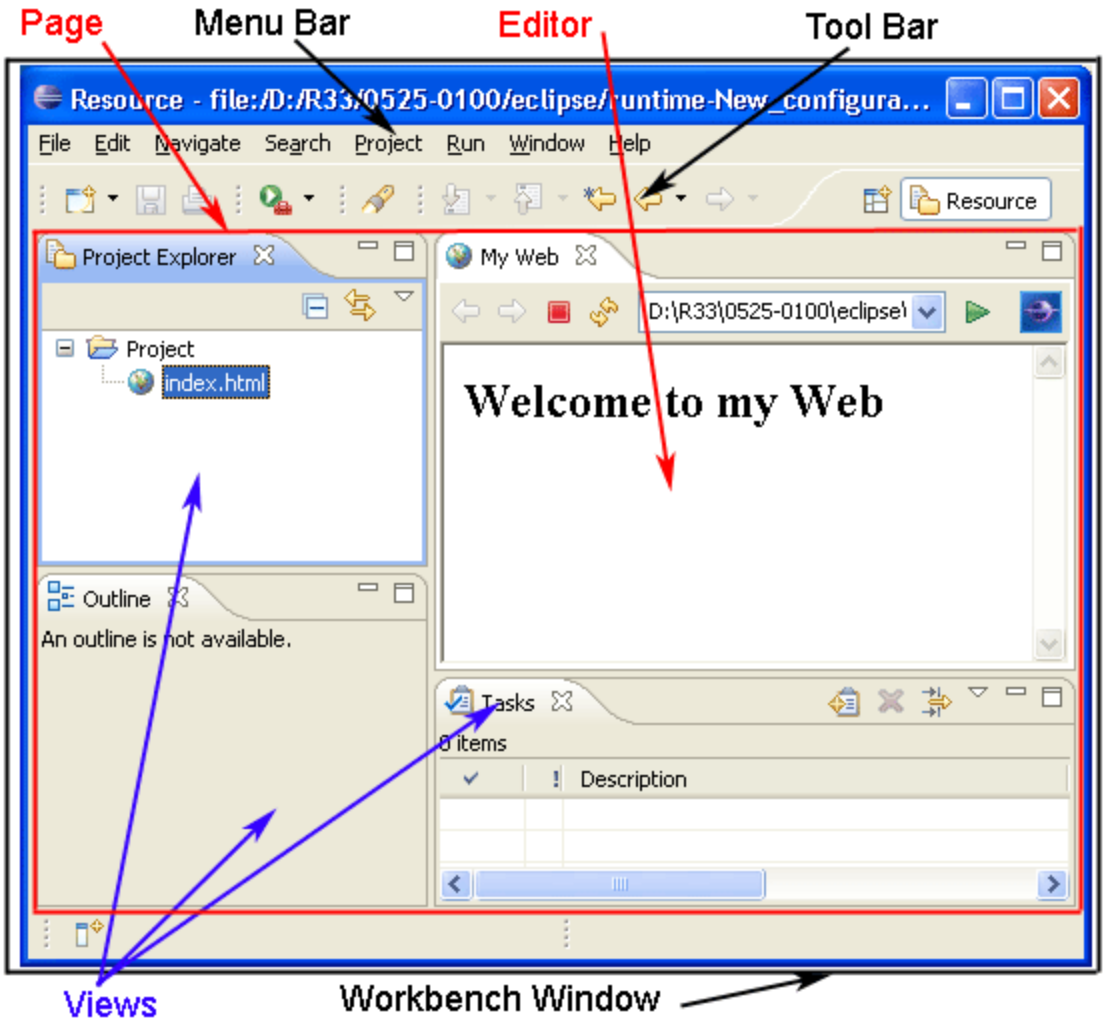
- **Eclipse** ist eine Open Source Community, die Werkzeuge zur Softwareentwicklung erstellt

- Das **Eclipse-Projekt**
  - stellt eine erweiterbare Plattform zur Entwicklung von Werkzeugen bereit
  - ist Framework für integrierte Entwicklungsumgebungen (Integrated Development Environments, IDEs)
  - wurde 2001 von IBM gegründet
  - stellt eine der zur Zeit am meisten verbreiteten IDEs für Java bereit (aber auch für C/C++, PHP, Perl,…)

HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Eclipse bietet

- Grundfunktionalität für Anwendungen wie Editoren, z.B.:
  - Ressourcen-Management
  - GUI-Elemente
  - speicherbare Einstellungen
  - diverse Editoren
  - Online-Hilfe
  - und vieles mehr



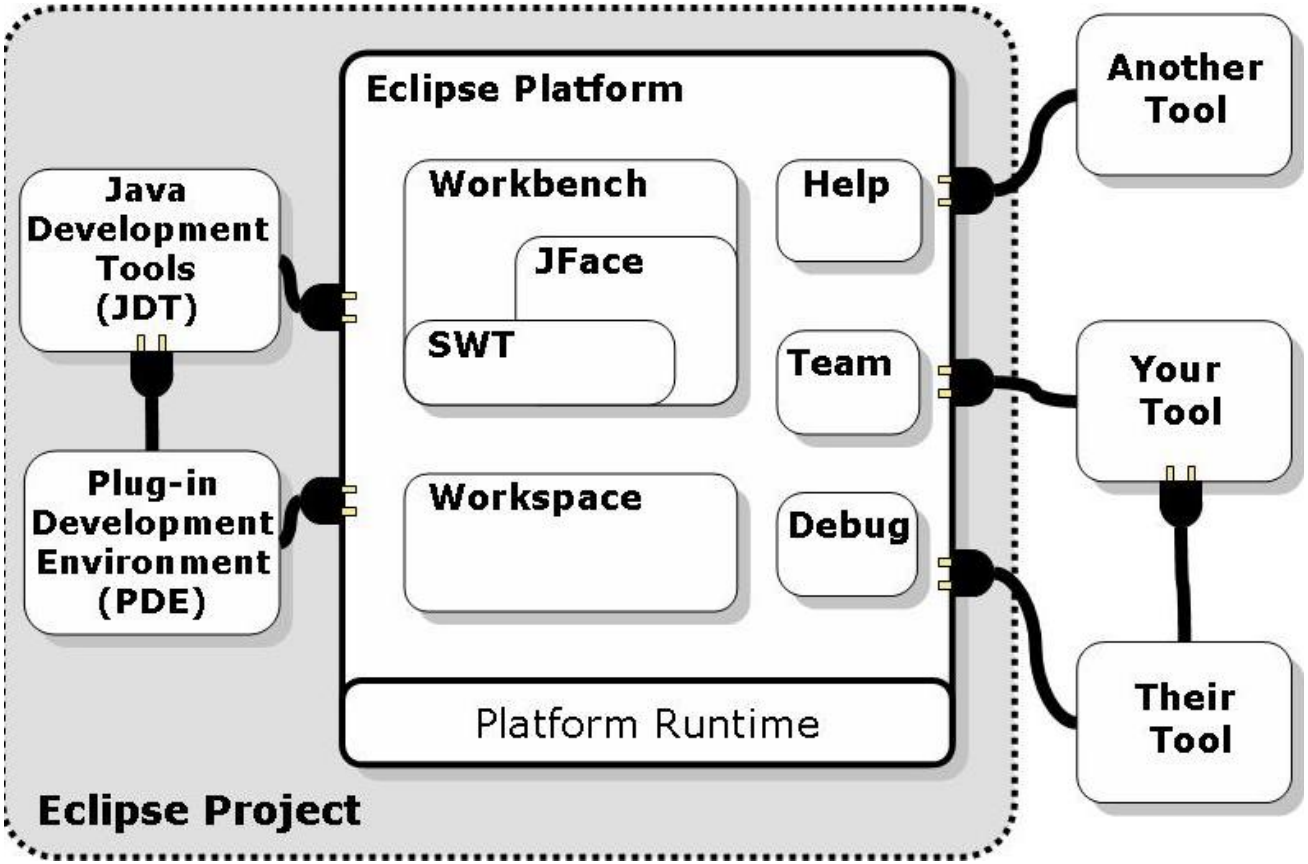
HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Eclipse

- Ist Erweiterbar durch Plug-ins
- Ist (fast) Plattform-unabhängig
  (Support für Windows, Linux, Mac OS,…)
- Werkzeuge erweitern Eclipse um
  - Java-IDE (Java Development Tools, JDT)
  - IDE für Eclipse-Plug-ins (Plug-in Development Environment, PDE)
  - Versions- und Konfigurationsmanagement (z.B. CVS- und SVN)
  - Modellierungswerkzeuge (Eclipse Modeling Framework, EMF; Graphical Editing Framework, GEF; Graphical Modeling Framework, GMF; etc.)
  - Und vieles, vieles mehr durch Hunderte von weiteren Plug-ins

HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Architektur von Eclipse (vereinfacht)



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

- **Java-Perspektive** (Empfohlene Auswahl und Anordnung von Views und Editoren)
  - Öffnen durch *Window* → *Open Perspective* → *Java*

- Verwaltung aller (Sourcecode-)Dateien in einem **Java-Projekt**
  - Anlegen durch *File* → *New* → *Java Project* 

- **Automatisches Kompilieren** von Quellcode
  - Ein-/Ausschalten durch *Project* → *Build Automatically*

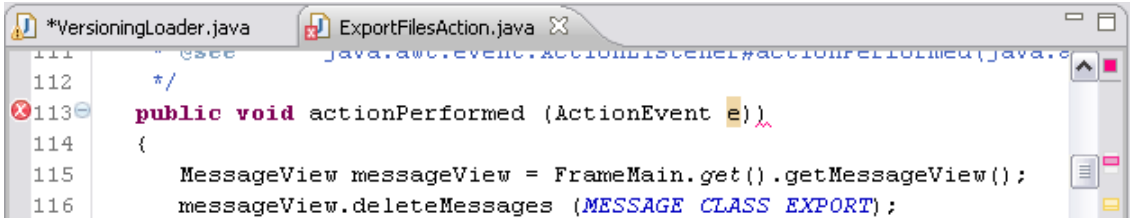
- **Ausführen** von Java-Programmen
  - Ausführen eines Programms durch *Run* → *Run As* → *Java Application* oder den Button  in der Tool Bar

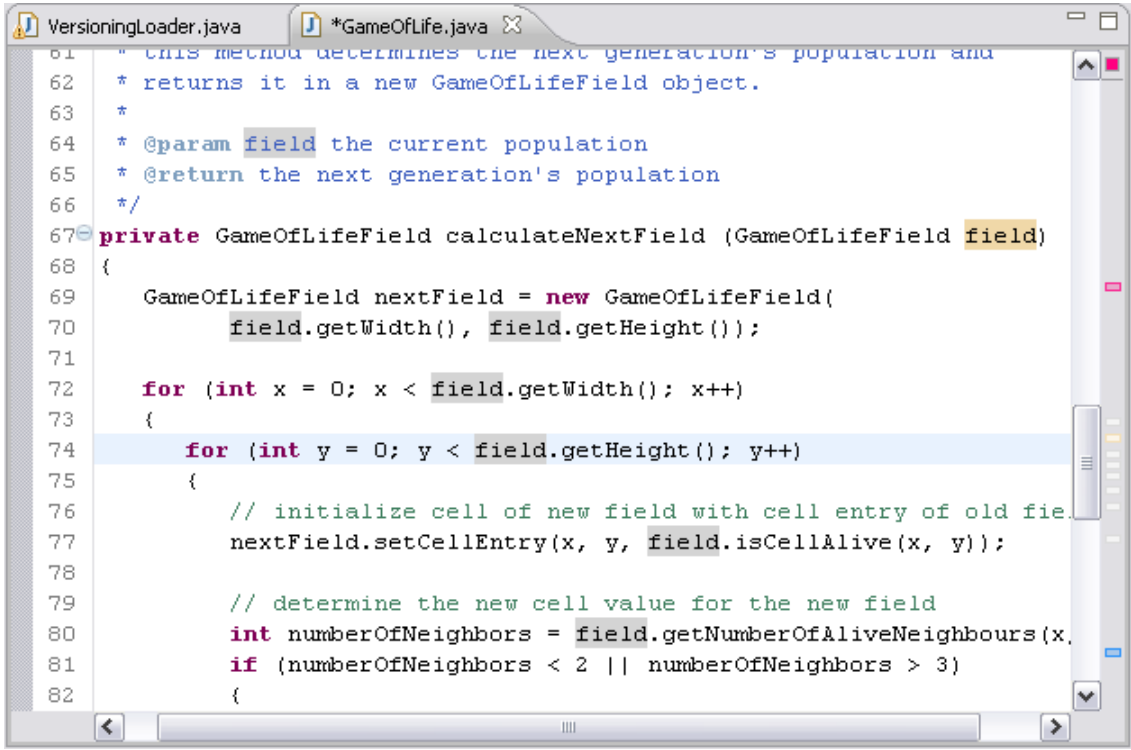
HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Java-Editor

- Code Completion (*Strg & Space*)
- Refactorings (*Kontextmenü → Refactor*)
- Code Formatter (*Strg & Shift & F*)
- Div. Editieroper. (*Kontextmenü → Source*)
- Auto-Build
- Markierungen
- Korrektur-Vorschläge
- Etc.

Die meisten Funktionen im Kontextmenü & der Tool Bar



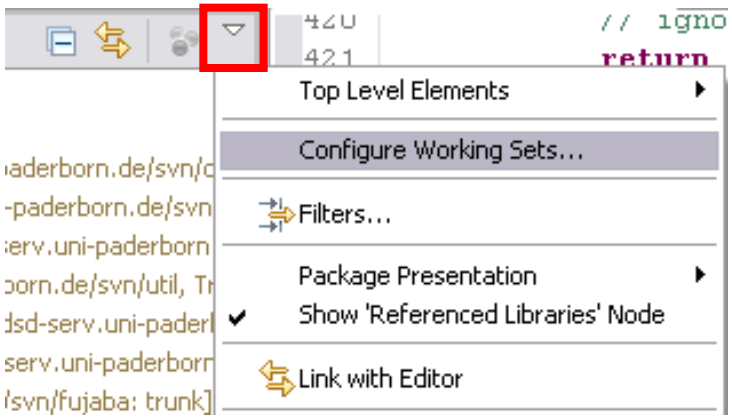


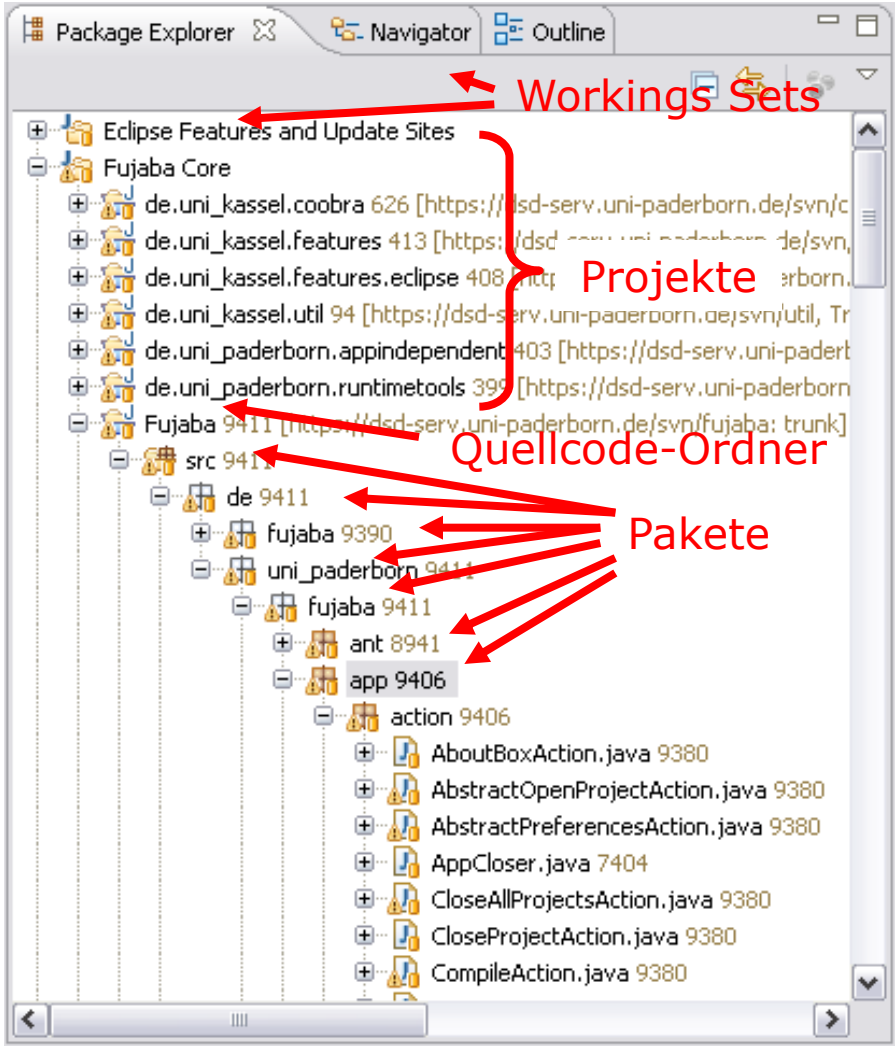
HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Views

Beliebige View öffnen über
Window → Show View → …

- *Package Explorer*
  - Baumdarstellung aller Projekte, Quellcode-Dateien, Paketstruktur, Bibliotheken, etc.
  - Gruppierung verschiedener Projekte durch Working Sets



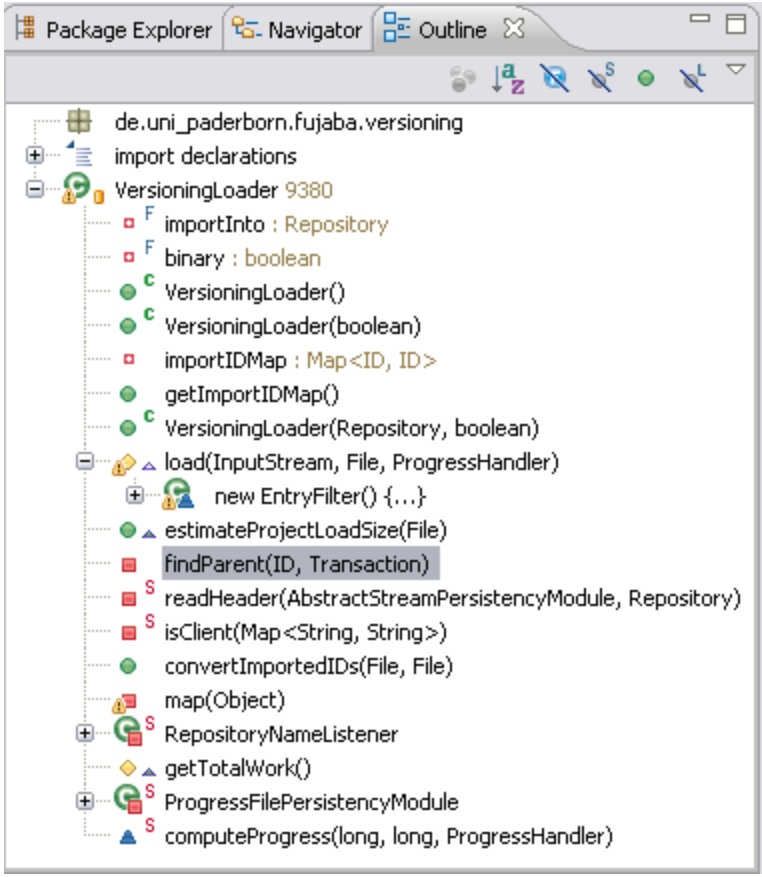


HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Views

- *Outline*
  - (meist) Baumdarstellung der Struktur eines geöffneten Dokuments, z.B. Attribute, Methoden und Klassendeklarationen in einer Klasse
  - Diverse Filter und Sortierungen möglich
- *Console*

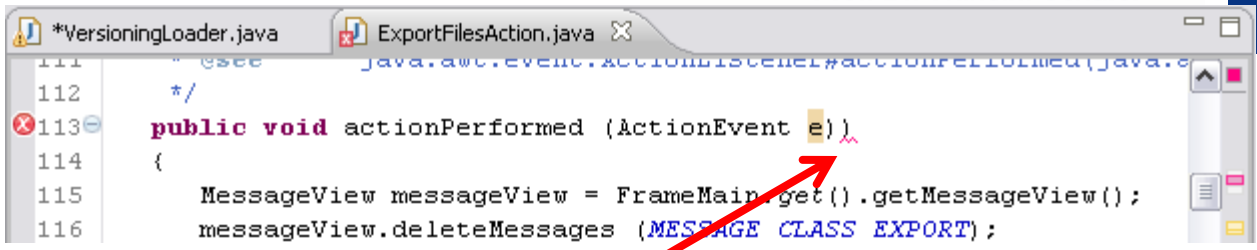
  - Konsolenausgaben, z.B. über `System.out.println()` oder `System.err.println()`

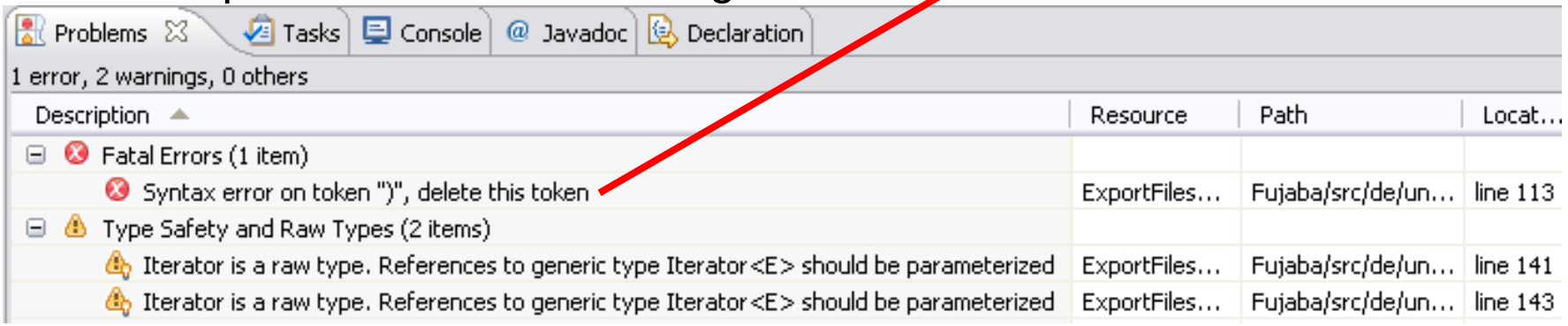


HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

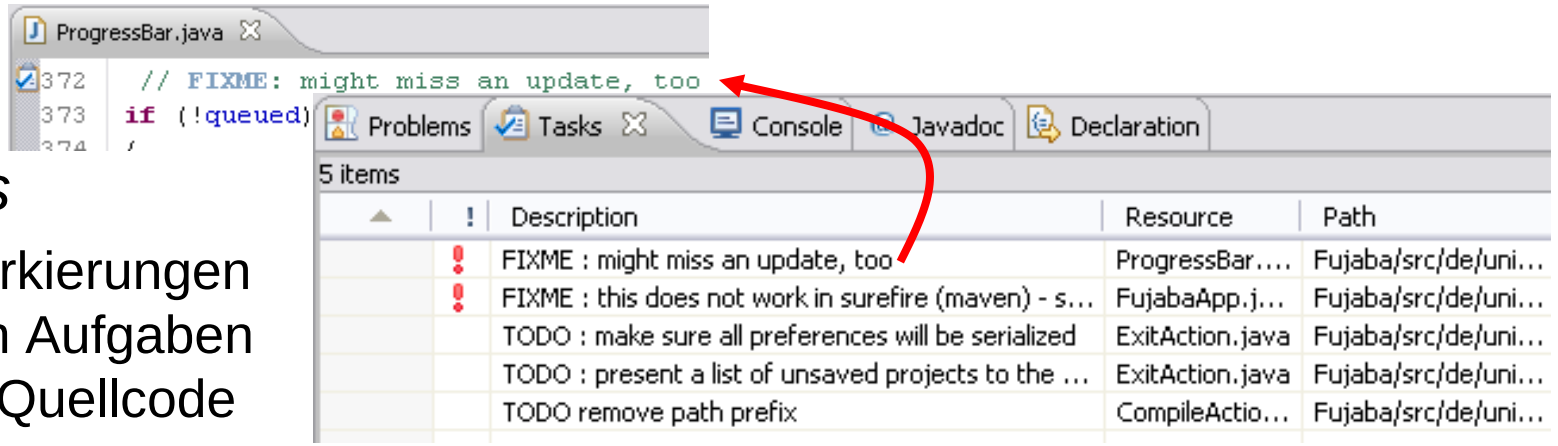
Views

- *Problems*
  - Kompilierfehler und Warnungen





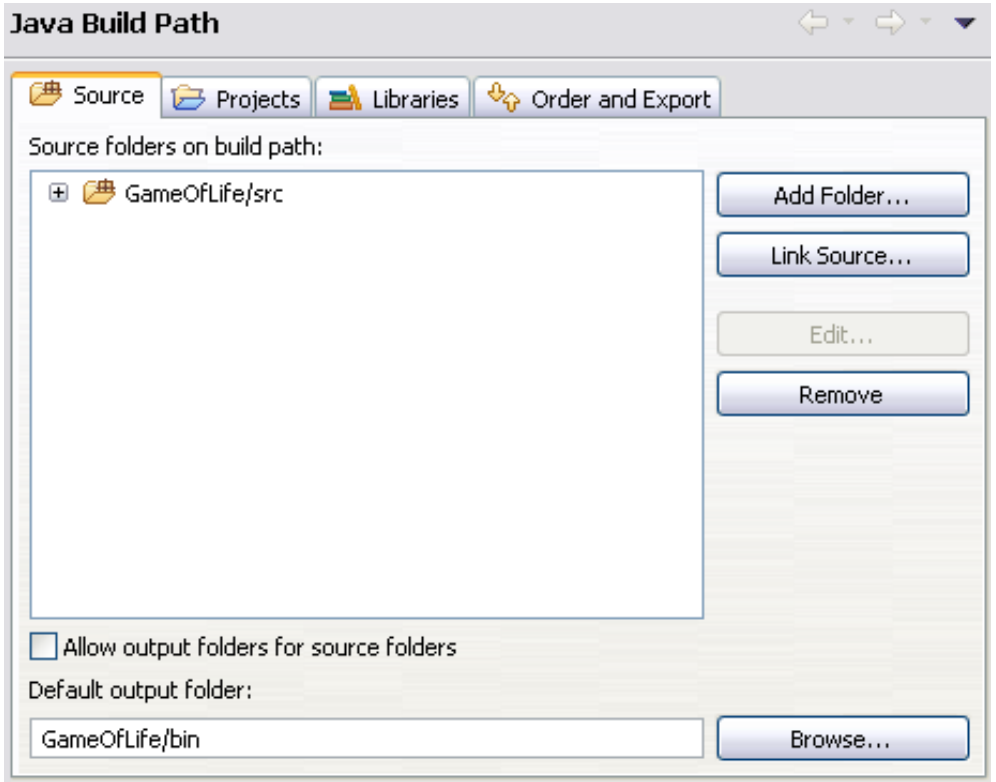
- *Tasks*
  - Markierungen von Aufgaben im Quellcode



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Java Build Path

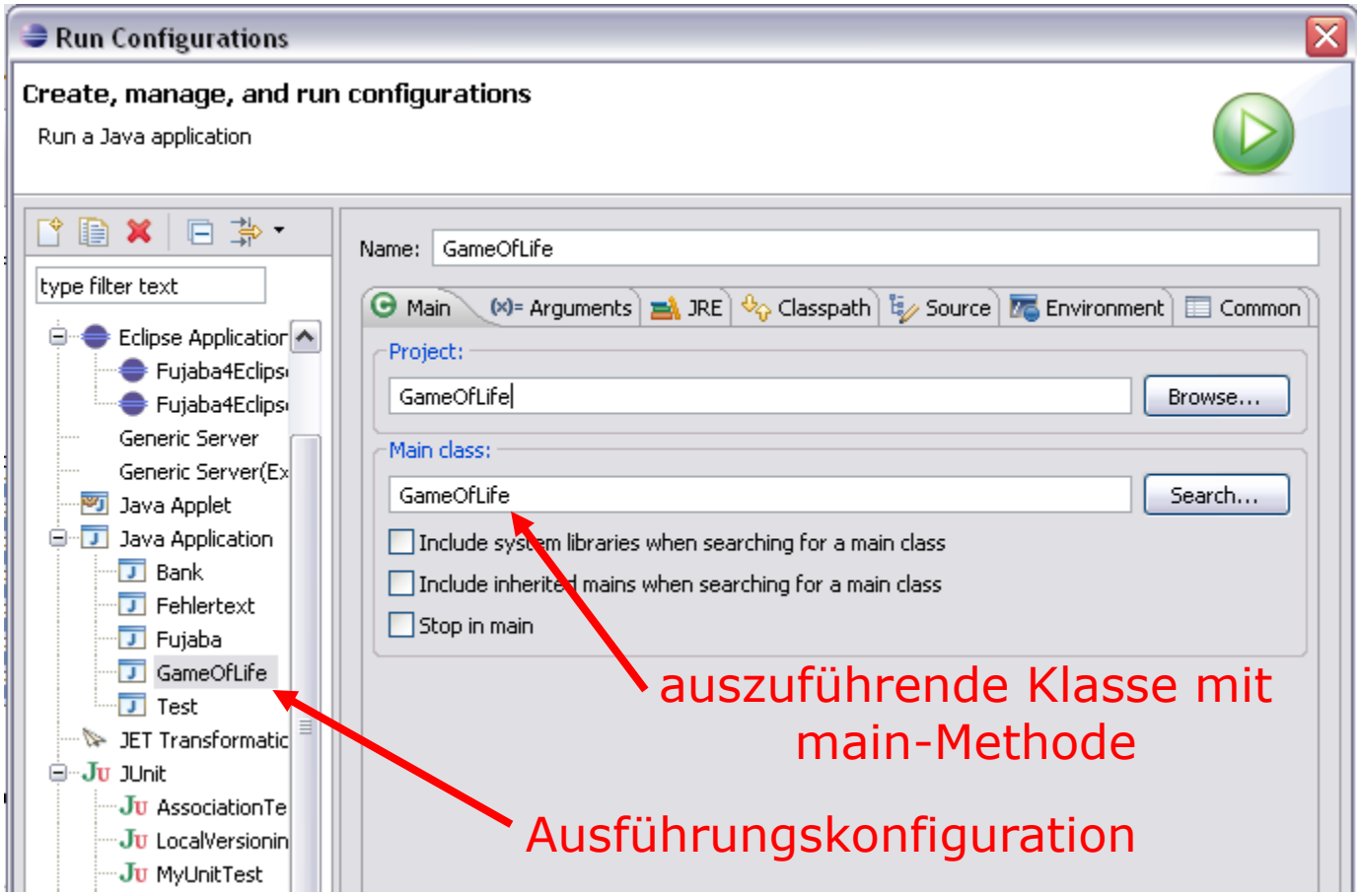
- Bestimmt Quellcode-Verzeichnis(se) und Abhängigkeiten zu anderen Projekten, Bibliotheken, JRE-Version, etc.
- Auswahl beim Erstellen eines neuen Java-Projekts oder später durch *Project → Properties → Java Build Path*



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

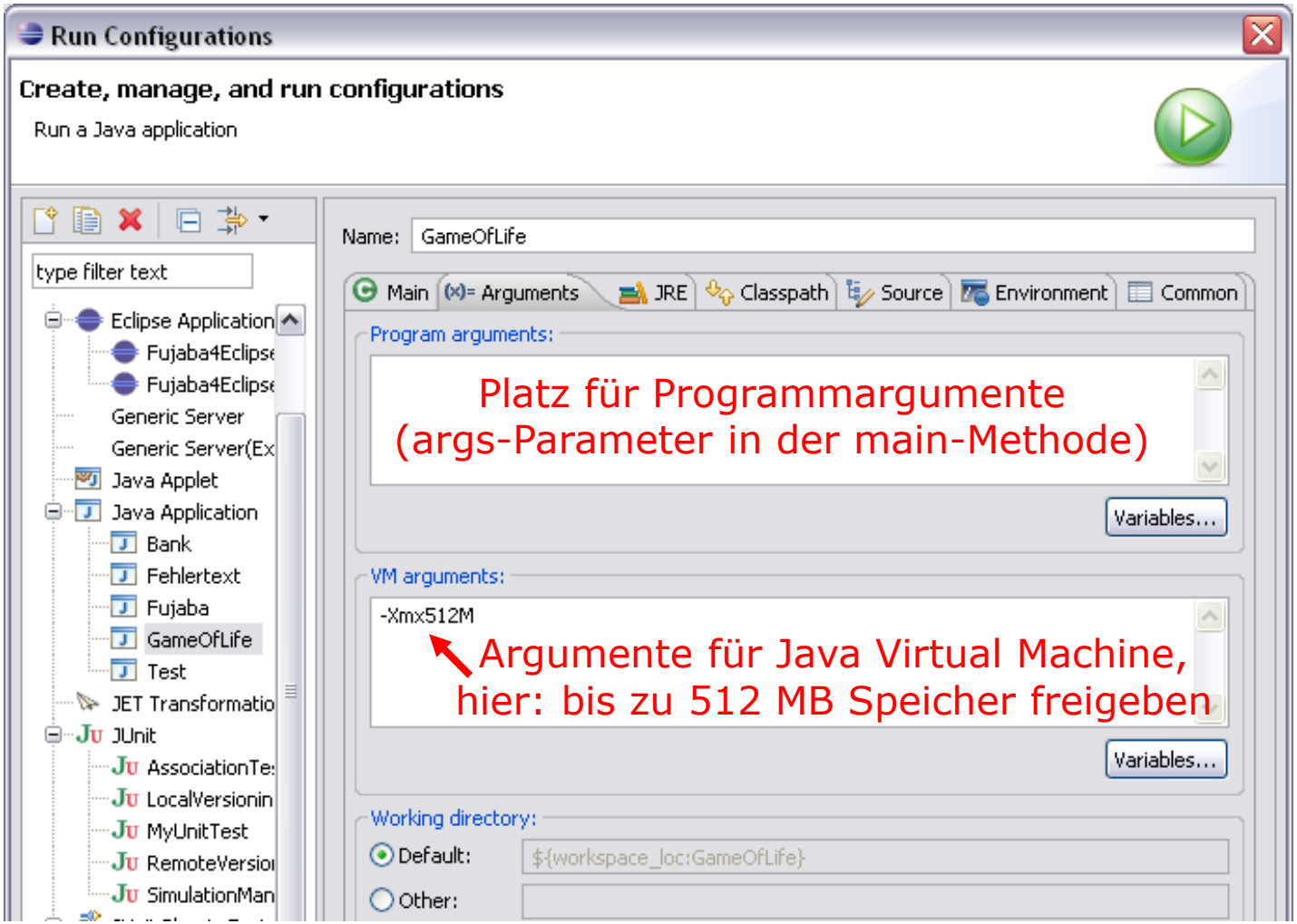
Ausführung von Java-Programmen

- Einstellungen unter *Run → Run Configurations…*



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Ausführung von Java-Programmen



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Entwicklung größtenteils analog zur „gewöhnlichen“ Java-Programmierung

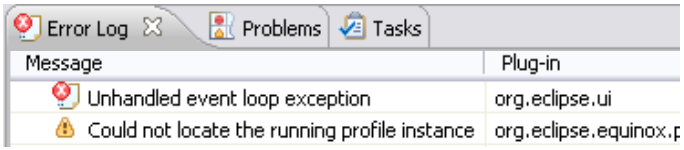
- **Plug-ins-Perspektive** (optional)

- **Plug-in-Projekt**

  - Anlegen durch *File → New → Other… → Plug-in Development → Plug-in Project*

Kein Java-Projekt anlegen!

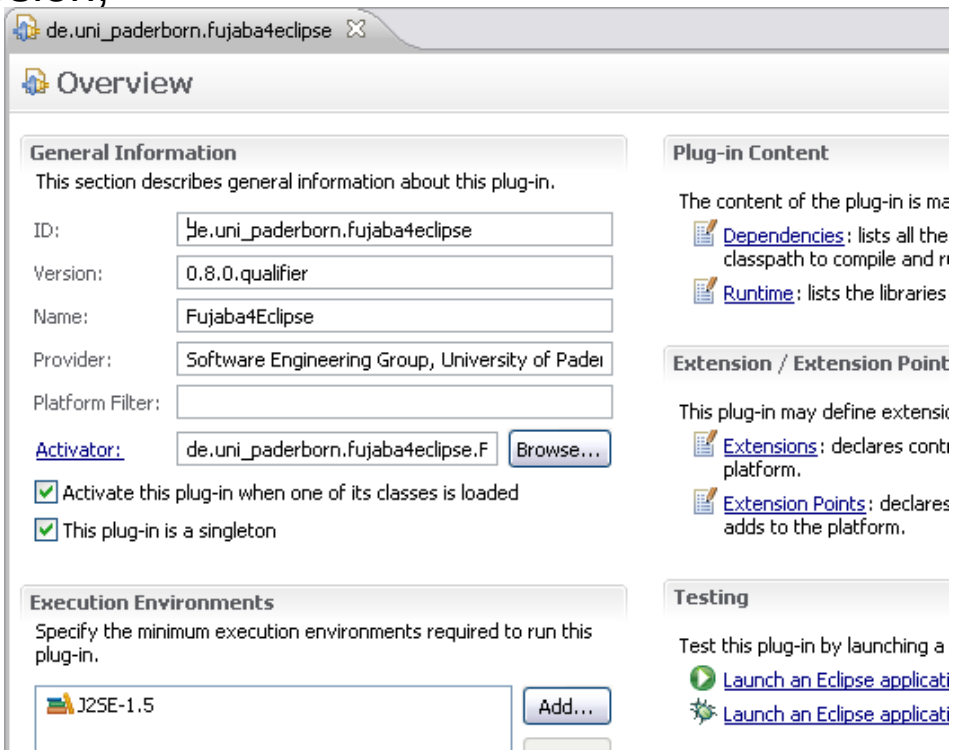
- **Ausführen** von Plug-ins in einer neuen Eclipse-Workbench (weitere Instanz von Eclipse)
  - Ausführen durch *Run → Run As → Eclipse Application* oder den Button  in der Tool Bar
  - mehr Einstellungen (z.B. zu ladende Plug-ins) unter *Run → Run Configurations…*
  - Error Log View in Test-Workbench



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Zu beachten bei Plug-in-Entwicklung

- Plug-in-Projekte werden anders kompiliert
  - *plugin.xml*, *MANIFEST.MF* beschreiben Eigenschaften eines Plug-ins, z.B. Plug-in-Abhängigkeiten, Version, ID,…
  - Anstatt der Build-Path-Einstellungen bestimmen *MANIFEST.MF* und *build.properties* den Klassenpfad (class path)
  - Spezieller Editor für *plugin.xml*, *MANIFEST.MF* und *build.properties*



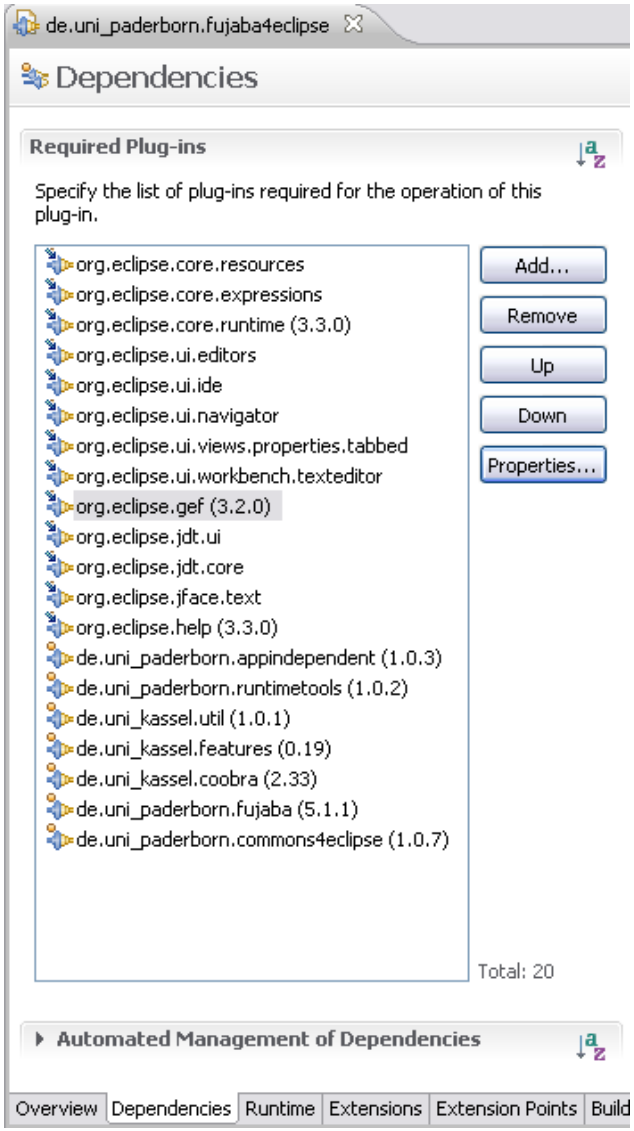
Bei Plug-ins nie Abhängigkeiten unter *Java Build Path* einstellen!

HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Plug-in-Abhängigkeiten

- Definition in *MANIFEST.MF*
- Angabe der Version der benötigten Plug-ins optional (über *Properties…*-Button)

- Plug-ins haben eingeschränkte Sicht auf andere Plug-ins:
  für andere Plug-ins sind nur die Klassen in exportierten Paketen sichtbar (siehe Reiter *Runtime* → *Exported Packages*)



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

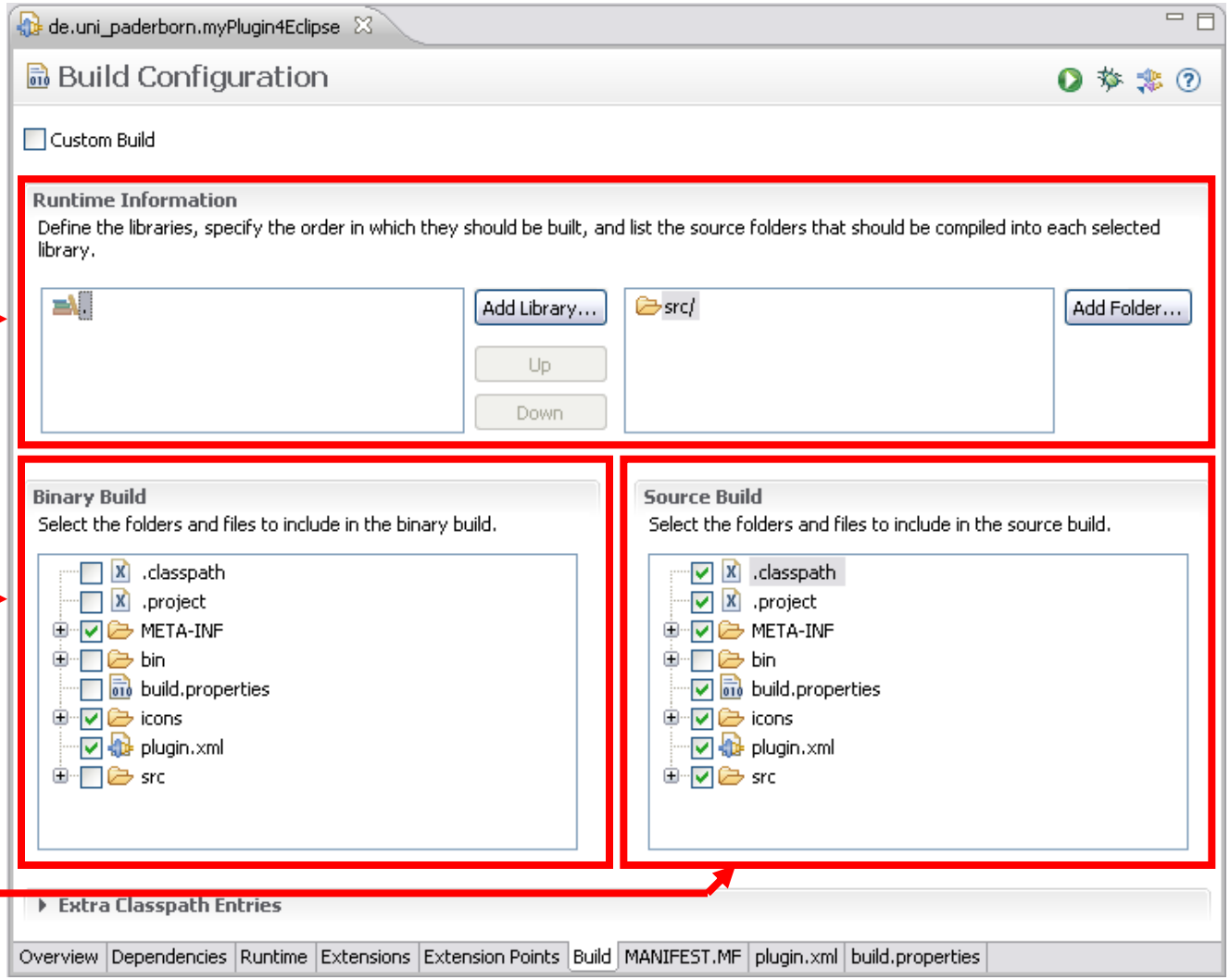
Build-Einstellungen für Plug-ins

- Einstellungen in build.properties

Bibliotheken des Plug-ins, hier: nur das Plug-in selbst

Vom Plug-in zur Laufzeit benötigte Dateien

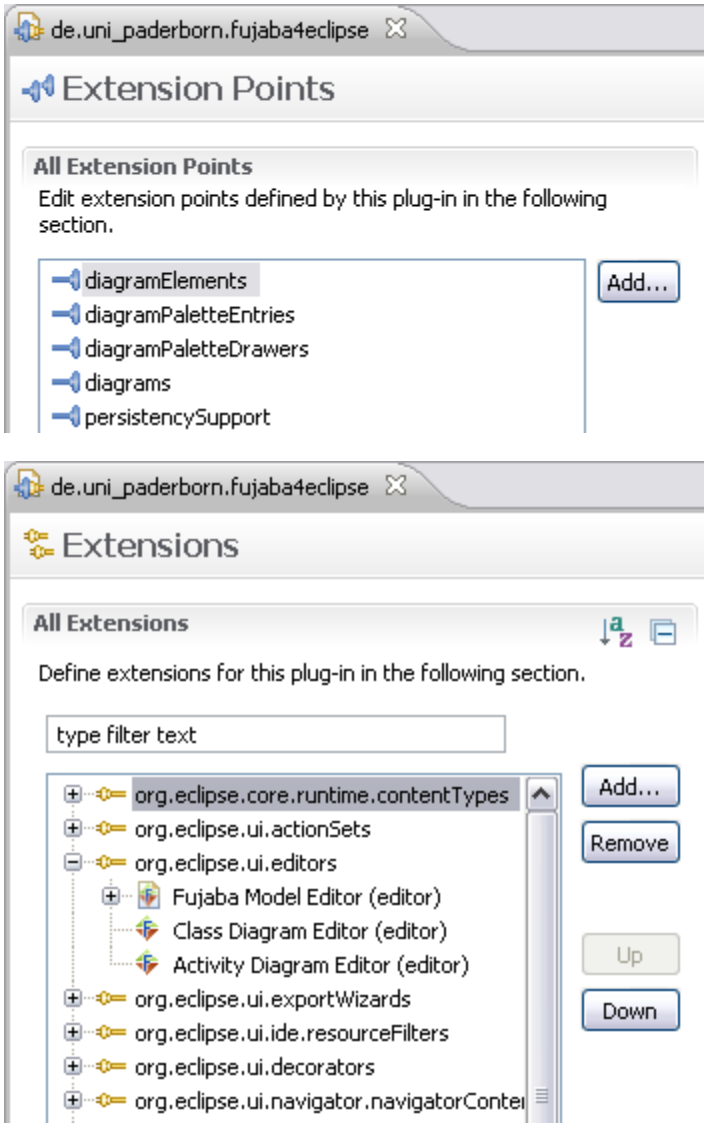
Vom Plug-in zur Laufzeit benötigte Dateien *mit Quellcode*



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Extension Points und Extensions

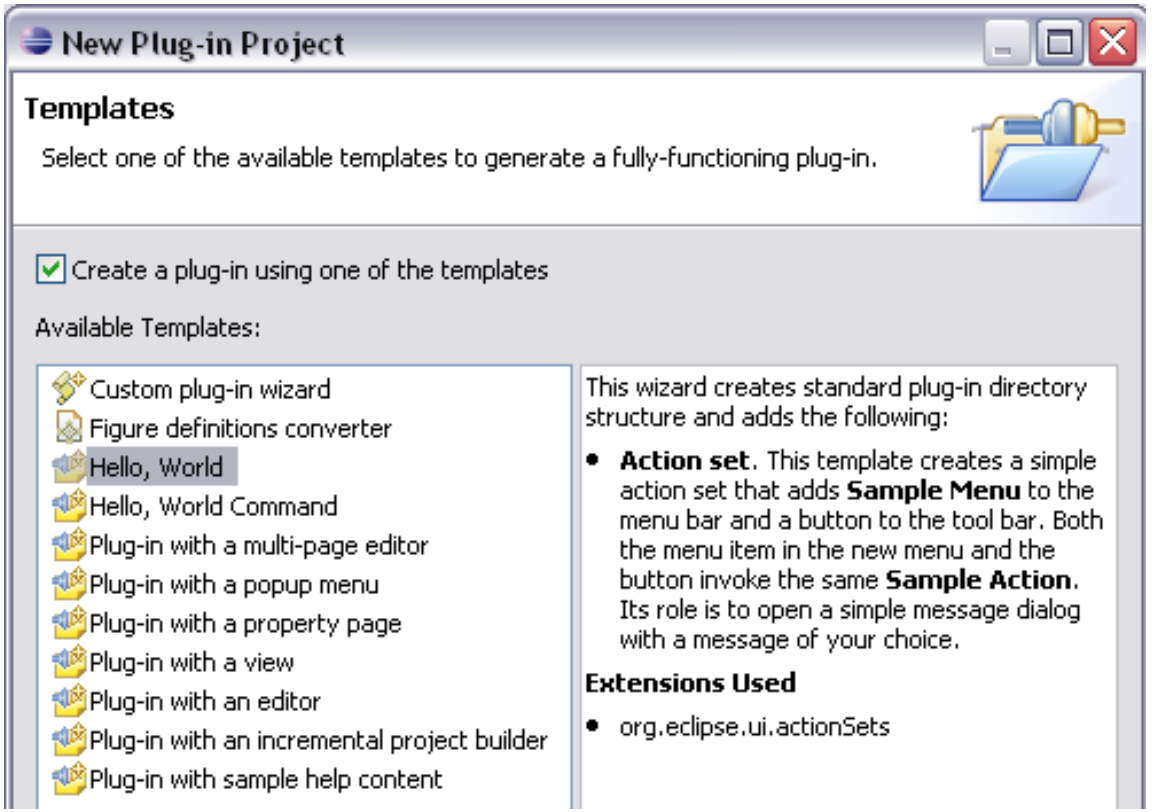
- Erweiterungsmöglichkeiten werden durch *Extension Points* definiert
  - Definieren Schnittstelle für Klassen, die von anderen Plug-ins bereitgestellt werden können
  - Haben Namen und eindeutige IDs
  - Eclipse bietet zahlreiche Extension Points, z.B. für Views, Editoren, Actions, etc.
- Konkrete Erweiterungen durch *Extensions*
- Beides in *plugin.xml* definiert



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Vorlagen und Beispiele für Plug-ins

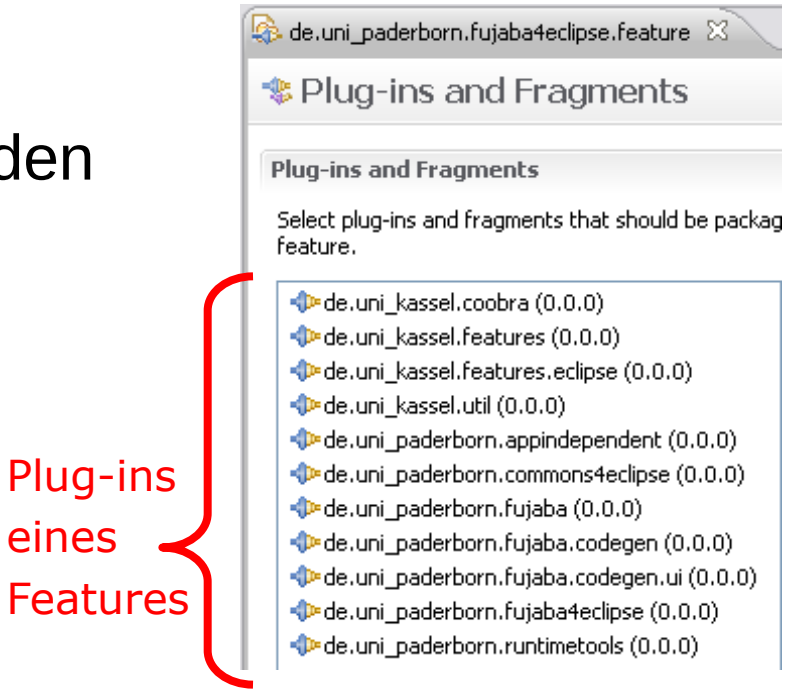
- Erzeugen von kleinen Plug-ins vereinfacht durch Vorlagen im New-Plug-in-Wizard
- Vorlagen für Editoren, Views, Menüeinträge, etc.
- Bsp.: Action im Menü, die einen Dialog mit dem Text „Hello, World“ öffnet.

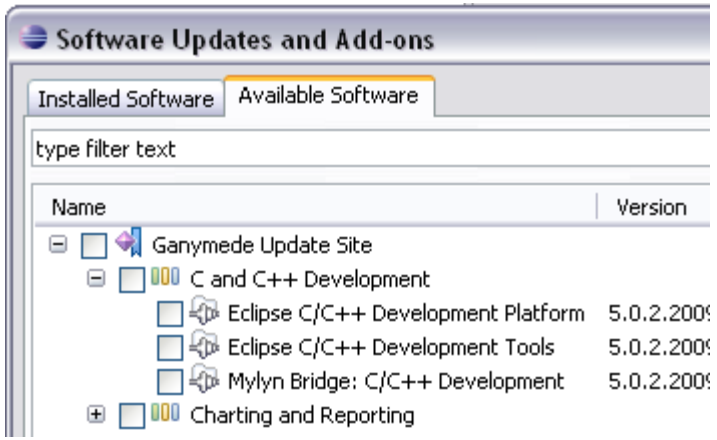


HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Features und Update Sites

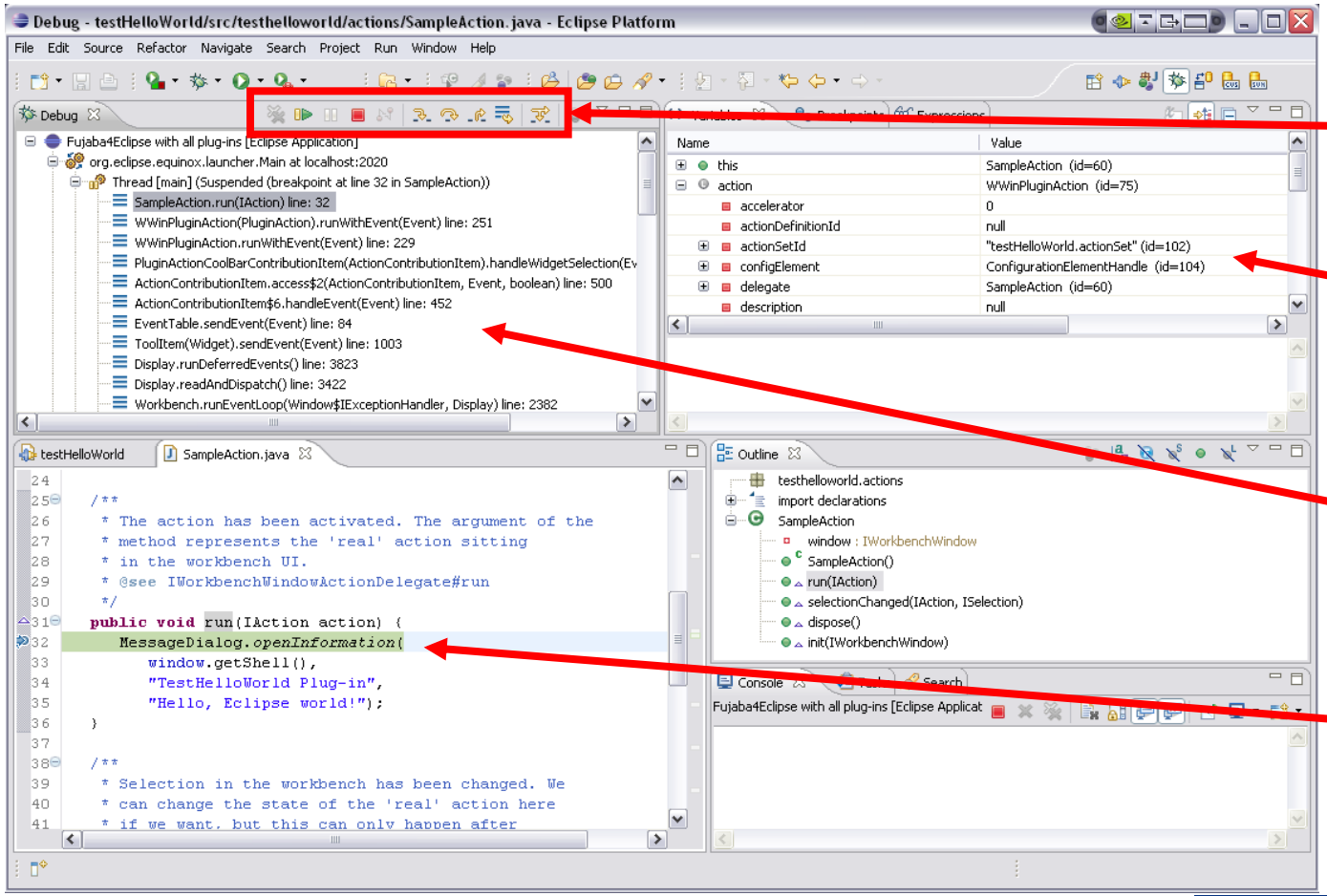
- Gruppe von zusammenhängenden Plug-ins kann in einem Feature zusammengefasst werden
  - Feature-Projekt mit *feature.xml*
  - File → New → Other… → Plug-in Development → Feature Project
- Installation und Update von Plug-ins über Update Sites
  - Update Site bietet verschiedene Versionen diverser Features zum Download an
  - Update-Site-Projekt mit *site.xml*
  - File → New → Other… → Plug-in Development → Update Site Project





HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

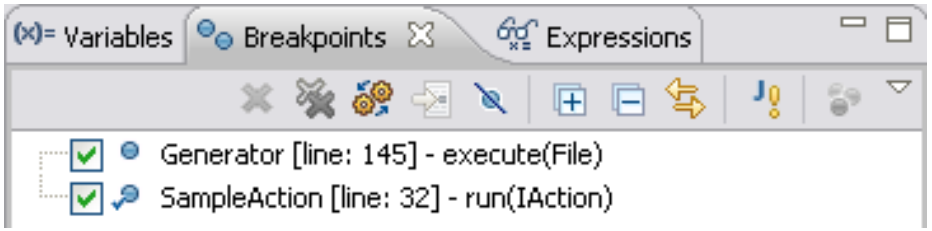
- Perspektive für‘s **Debugging**
- Debug-Ausführung von Programmen über Run → Debug As oder Run → Debug Configurations… oder den Button 



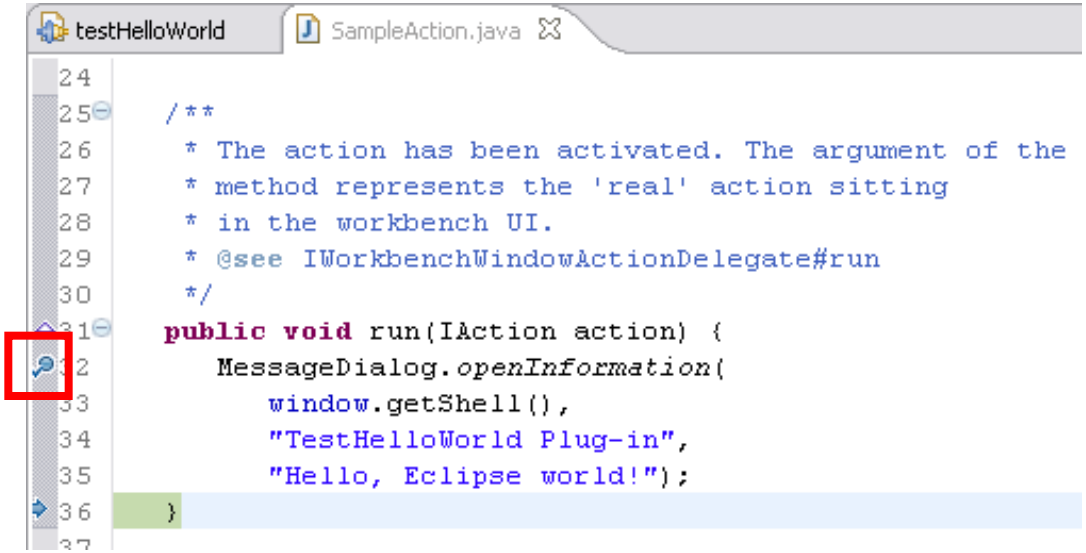
HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Haltepunkte im Programm (Breakpoints)

- Bestimmen, wo der Debugger das Programm anhalten soll
- Spezielle View in Debugging-Perspektive

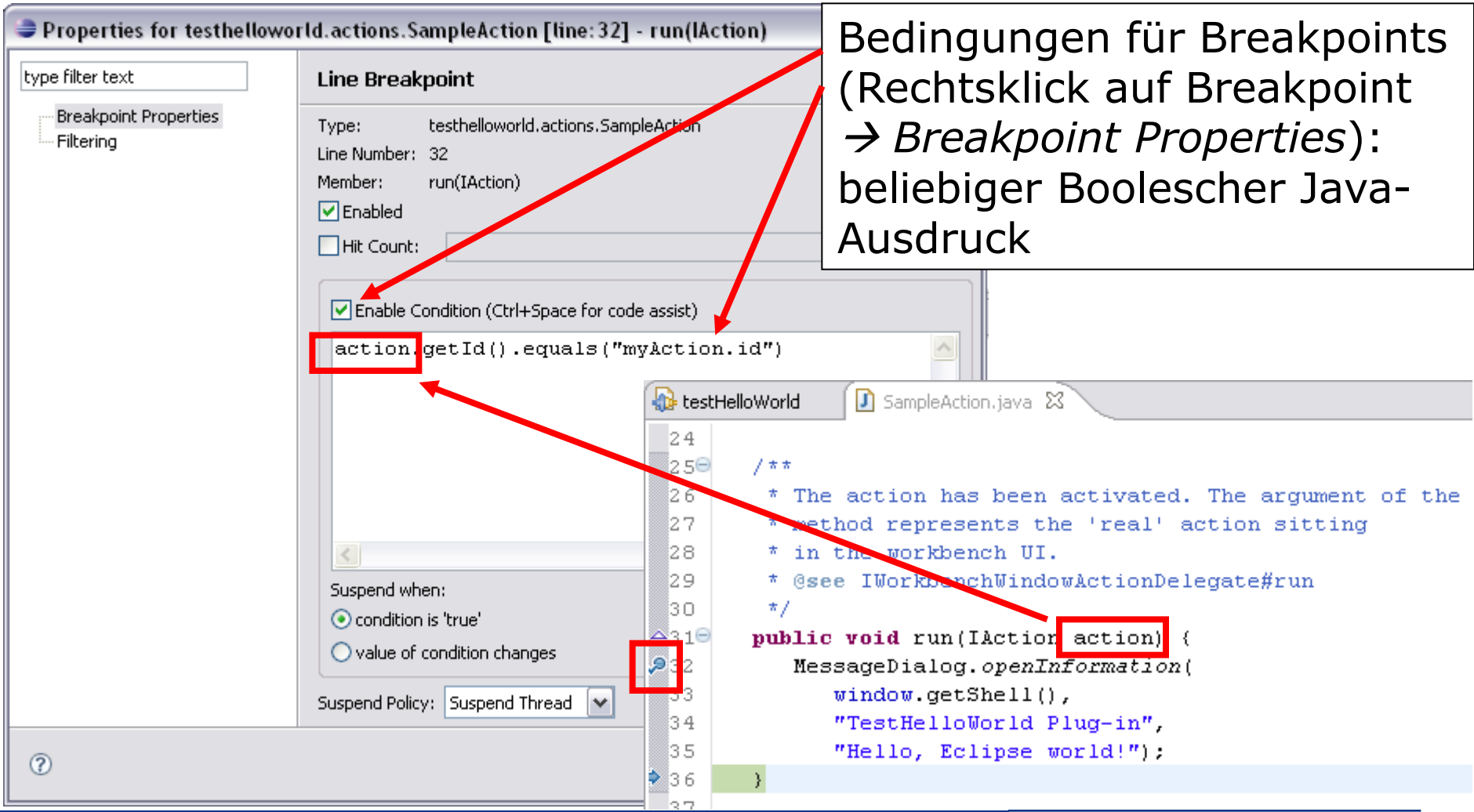


- Breakpoint anlegen durch Rechtsklick auf die graue Leiste links neben dem Quellcode und Auswahl von *Toggle Breakpoint*



HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Haltepunkte im Programm (Breakpoints)



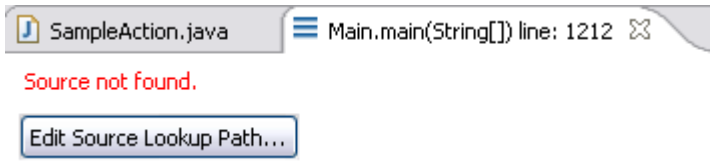
Bedingungen für Breakpoints (Rechtsklick auf Breakpoint → *Breakpoint Properties*): beliebiger Boolescher Java-Ausdruck

HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

- Resume (F8)
- Suspend
- Terminate
- Step Into (F5)
- Step Over (F6)
- Step Return (F7)

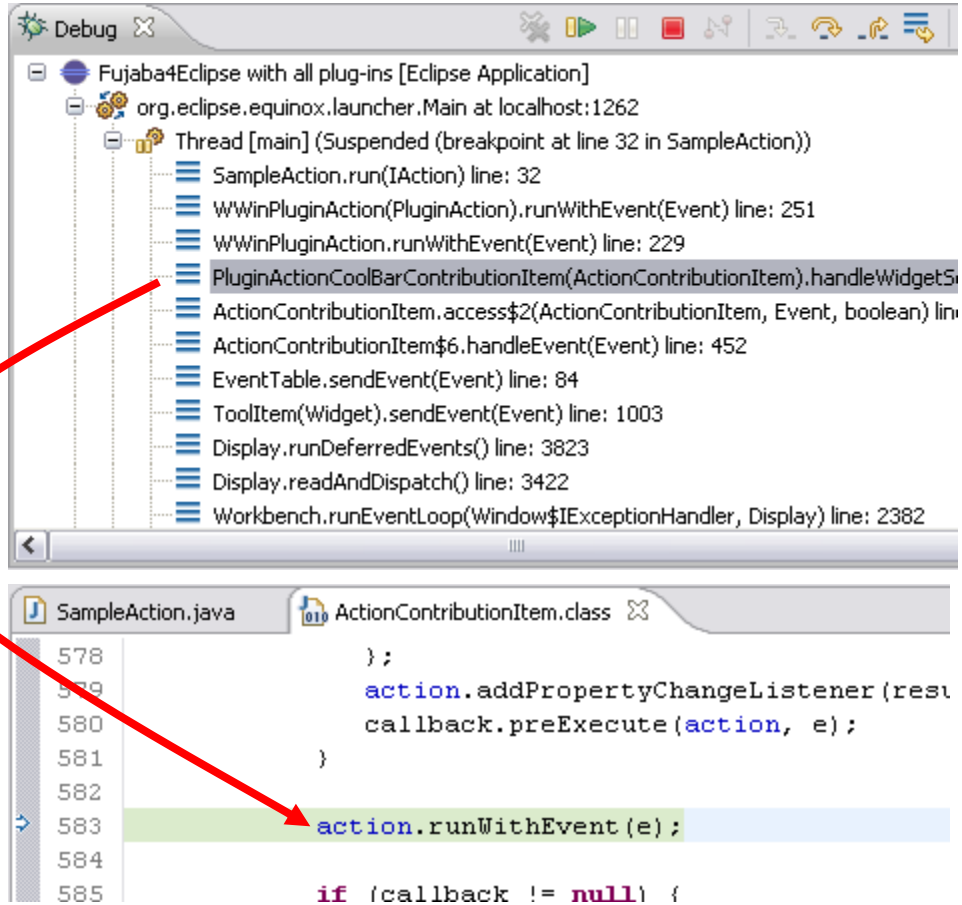
Bei fremdem Quellcode Quelle angeben (z.B. src.zip bei Java):

Rechtsklick in Debug-View → *Edit Source Lookup…*
oder direkt im Editor:



Debug View

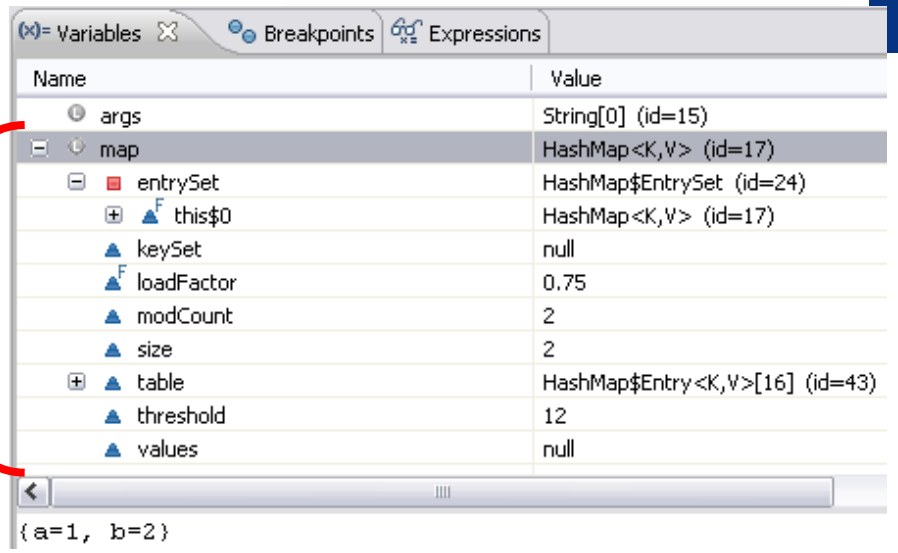
Sprung zu beliebiger Stelle im Stack möglich

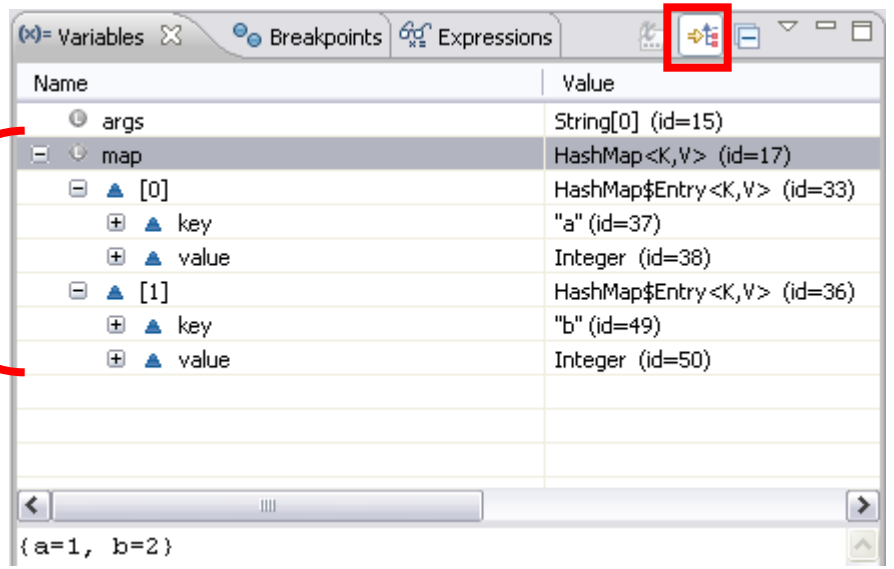


HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

Logische Strukturen

- Vereinfachte Darstellung komplexer Datenstrukturen
- Bsp.: HashMap
  - Viele Variablen, wesentliche Struktur nicht auf Anhieb erkennbar
  - Darstellung der logischen Struktur vereinfacht Ansicht: Liste von Key-Value-Paaren 
- Einschalten über Button
- Eigene Strukturen definieren unter *Window → Preferences → Java → Debug →Logical Structures*

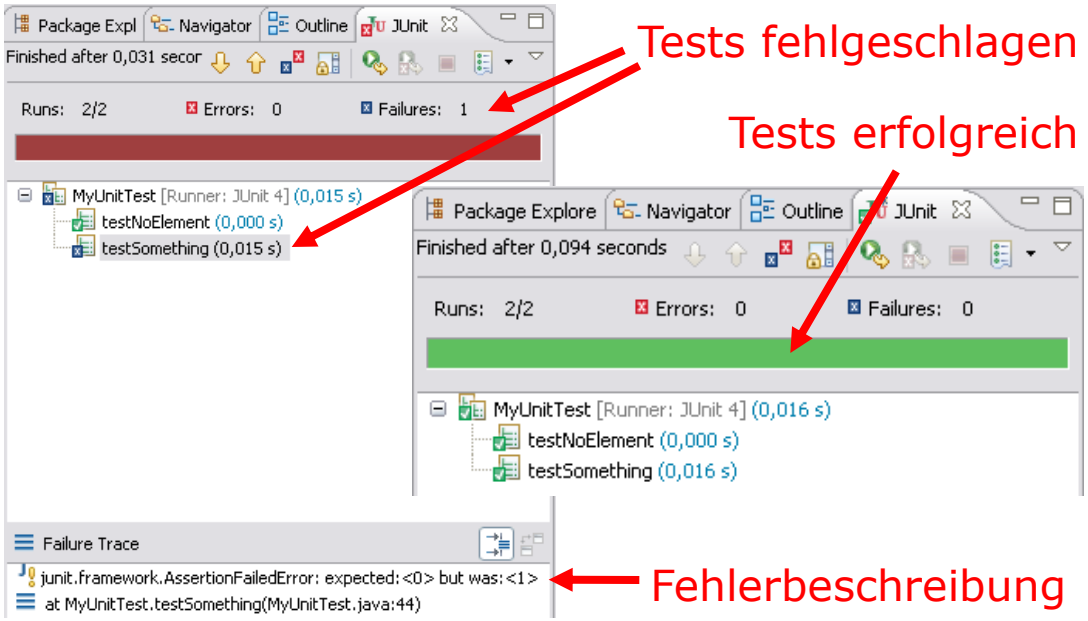




HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

JUnit-Tests in Eclipse

- Automatisierte Tests von Code-Abschnitten (siehe http://junit.org)
- Zusammenfassung mehrerer Tests in Test Cases und Test Suites
- Ausführung über *Run → Run As → JUnit Test*



MyUnitTest.java | AssociationTest.java

```
12 public class MyUnitTest
13 {
14    private Set<String> testSet;
15
16    @Before
17    public void setUp() throws Exception
18    {
19       testSet = new HashSet<String>();
20    }
21
22    @After
23    public void tearDown() throws Exception
24    {
25       testSet.clear();
26       testSet = null;
27    }
28
29    @Test(expected=NoSuchElementException.class)
30    public void testNoElement()
31    {
32       testSet.iterator().next();
33    }
34
35    @Test
36    public void testSomething()
37    {
38       TestCase.assertFalse(testSet.isEmpty());
39
40       testSet.add("Word");
41
42       Iterator<String> iter = testSet.iterator();
43       TestCase.assertNotNull(iter);
44       TestCase.assertEquals("Word", iter.next());
45    }
46 }
```

5. Installation von Eclipse

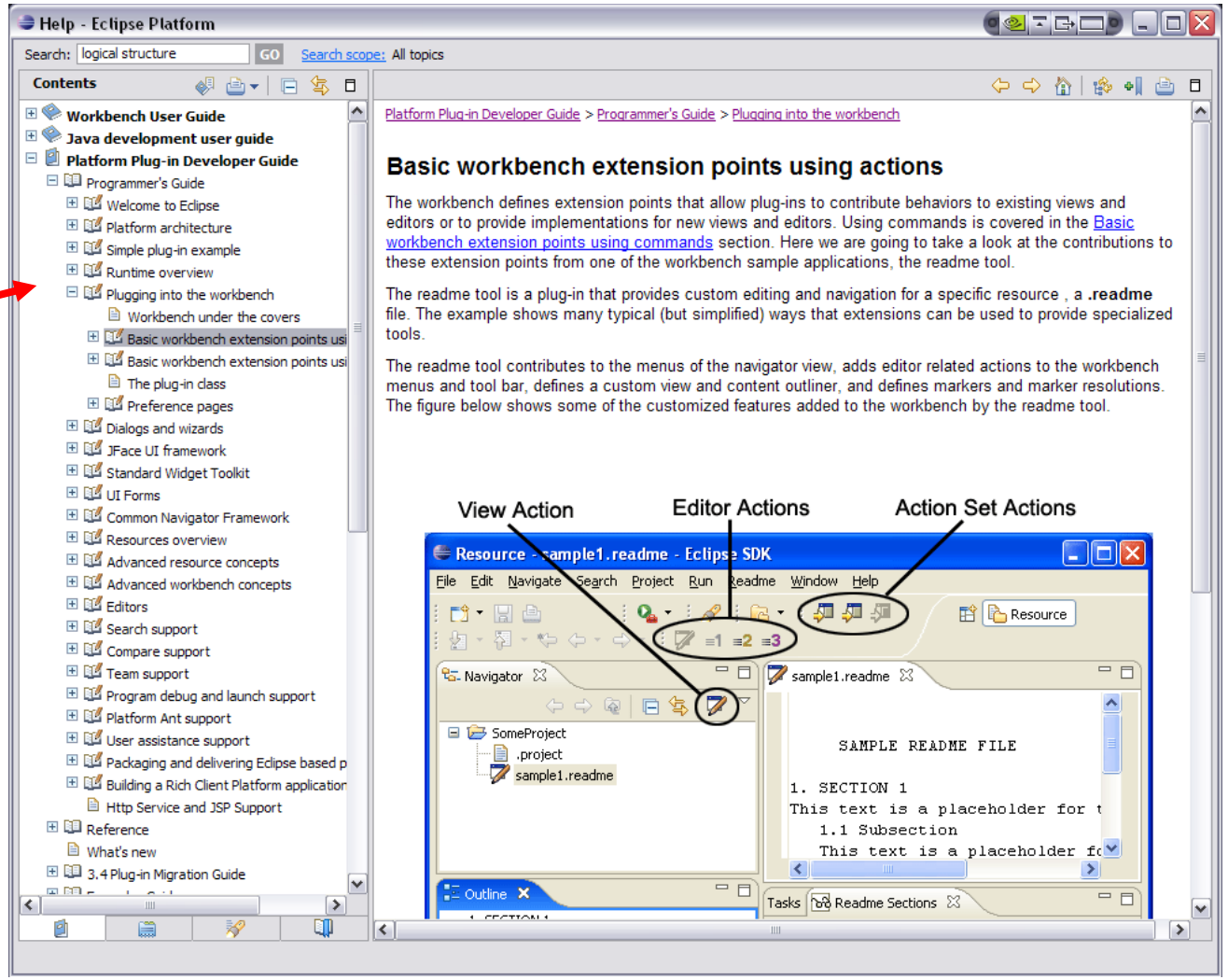
HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

- Java SDK 6
- Eclipse Modeling Tools 3.7.2 (Indigo)
  - Enthält bereits einige der benötigten Plug-Ins (z.B. EMF, GEF, Plug-In Development Tools)
- Eclipse Plug-ins:
  - Subversive SVN Client & Konnektoren
  - EclEMMA
  - …

HEINZ NIXDORF INSTITUT
Universität Paderborn
Softwaretechnik
Prof. Dr. Wilhelm Schäfer

- **Online Hilfe** in Eclipse *(Help → Help Contents)*

Diverse Hilfe-Themen, z.B. zur Eclipse-Plattform, JDT, PDE, EMF, GEF, SVN, etc.



6. Literatur und Referenzen

- **Bücher** zu Eclipse, EMF, GEF & Co.
  - *„Eclipse – Building Commercial-Quality Plug-ins“*, Eric Clayberg, Dan Rubel, Addison-Wesley, 2006
  - *„Contributing to Eclipse – Principles, Patterns, and Plug-ins“*, Erich Gamma, Kent Beck, Addison-Wesley, 2004
- Eclipse-Web-Seite: http://www.eclipse.org
  - **Artikel** zu Eclipse, z.B. GEF, SWT, JFace: http://www.eclipse.org/articles
  - **Newsgroups** (Fragen & Antworten zu diversen Fragen): http://www.eclipse.org/newsgroups
- Eclipse-Wiki: http://eclipsewiki.editme.com
- Eclipse User Interface Guidelines (Ver. 2.1)
  http://www.eclipse.org/articles/Article-UI-Guidelines/Index.html