



# *Einführung in die Entwicklungsumgebung*



*Tutorium  
im Rahmen des Softwaretechnik-  
& Software-Praktikums 2012*



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



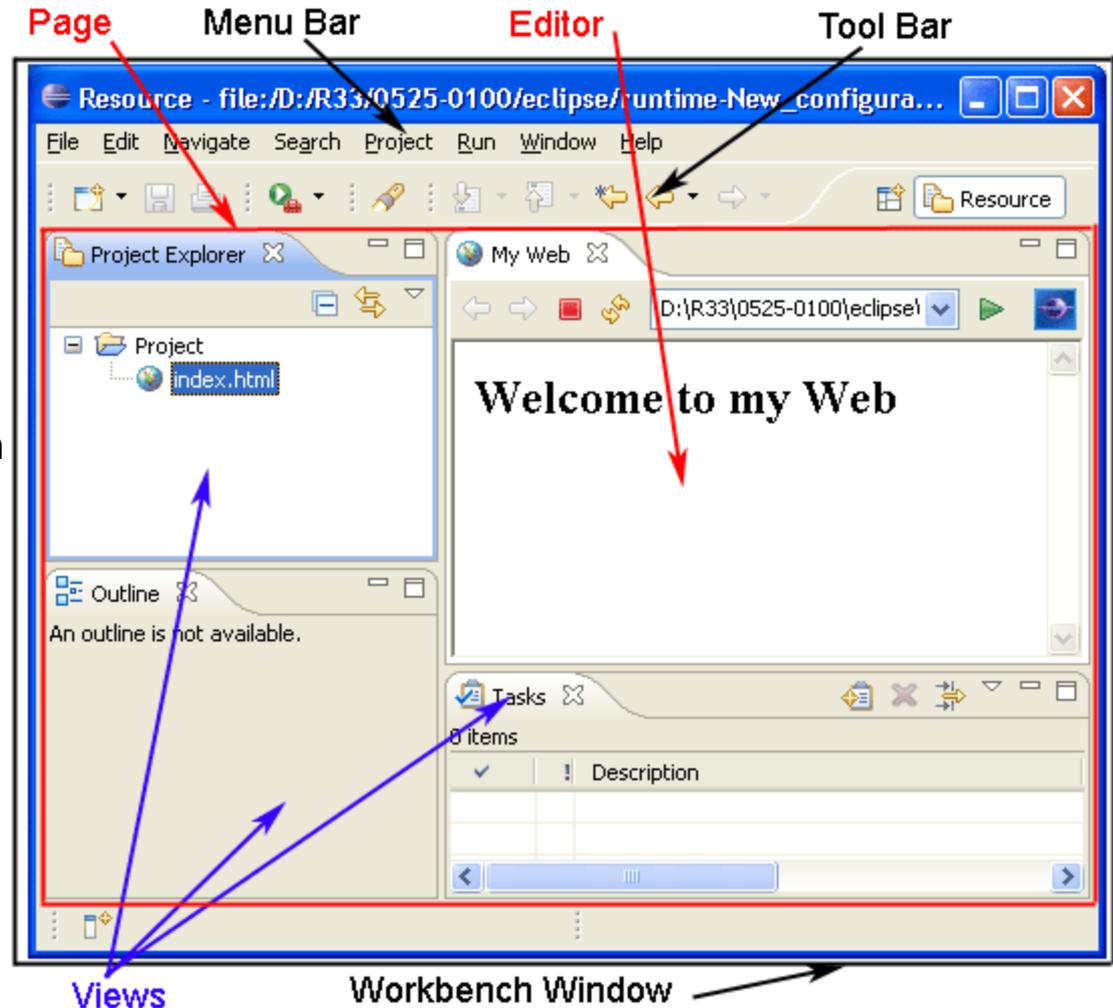
- **Das Eclipse-Projekt**
  - stellt eine erweiterbare Plattform zur Entwicklung von Werkzeugen bereit
  - ist Framework für integrierte Entwicklungsumgebungen (Integrated Development Environments, IDEs)
  - wurde 2001 von IBM gegründet
  - stellt eine der zur Zeit am meisten verbreiteten IDEs für Java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

# 1. Was ist Eclipse?



## Eclipse bietet

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



# 1. Was ist Eclipse?



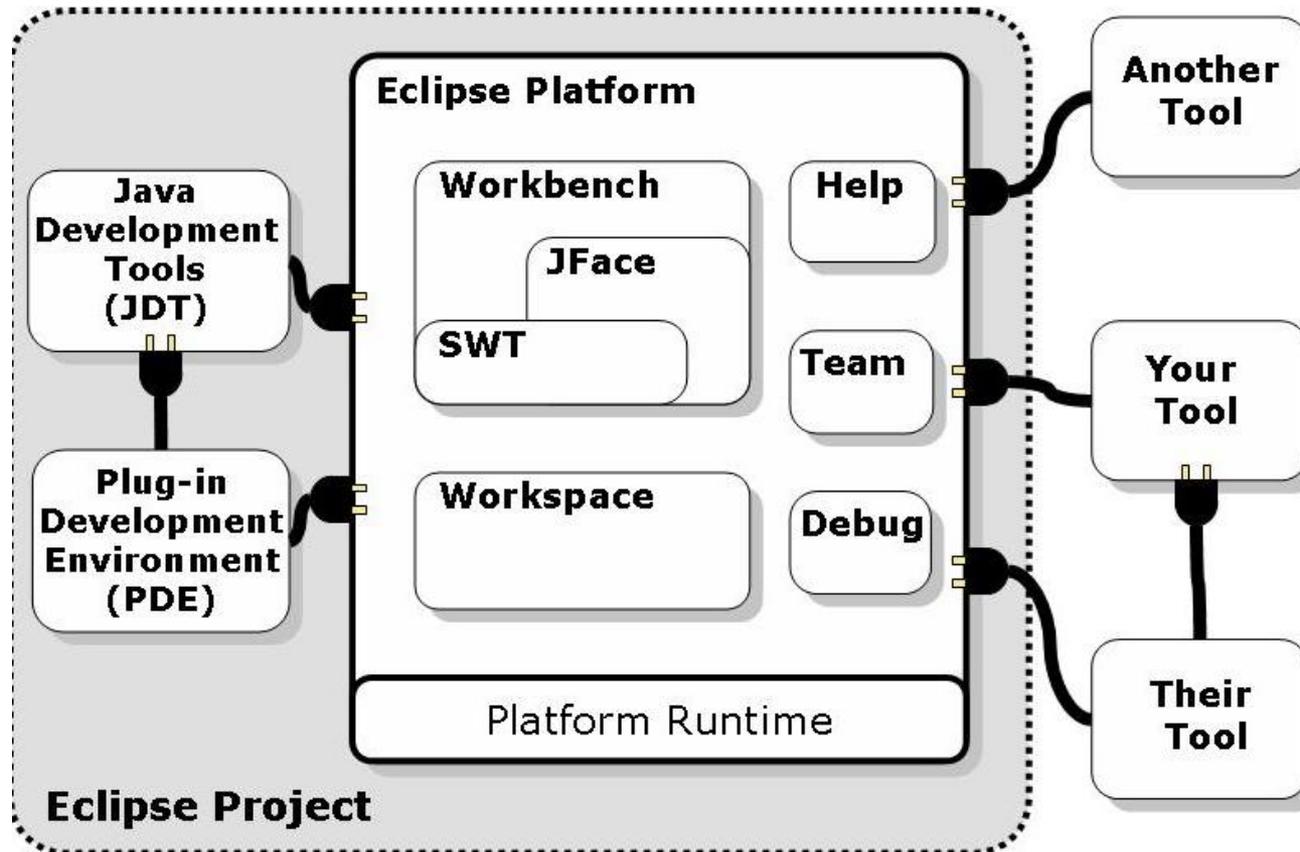
## Eclipse

- 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

# 1. Was ist Eclipse?



## Architektur von Eclipse (vereinfacht)



## 2. Java-Programmierung in Eclipse



- **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*  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

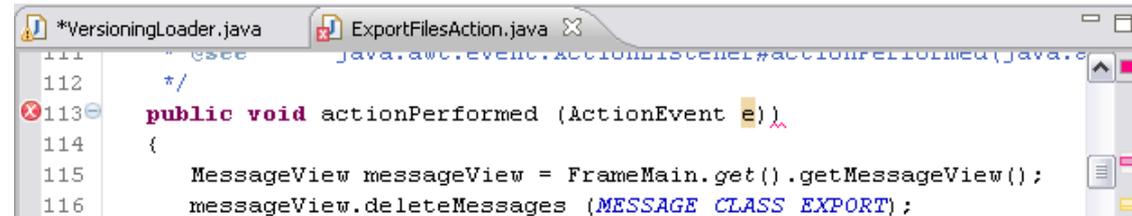
## 2. Java-Programmierung in Eclipse



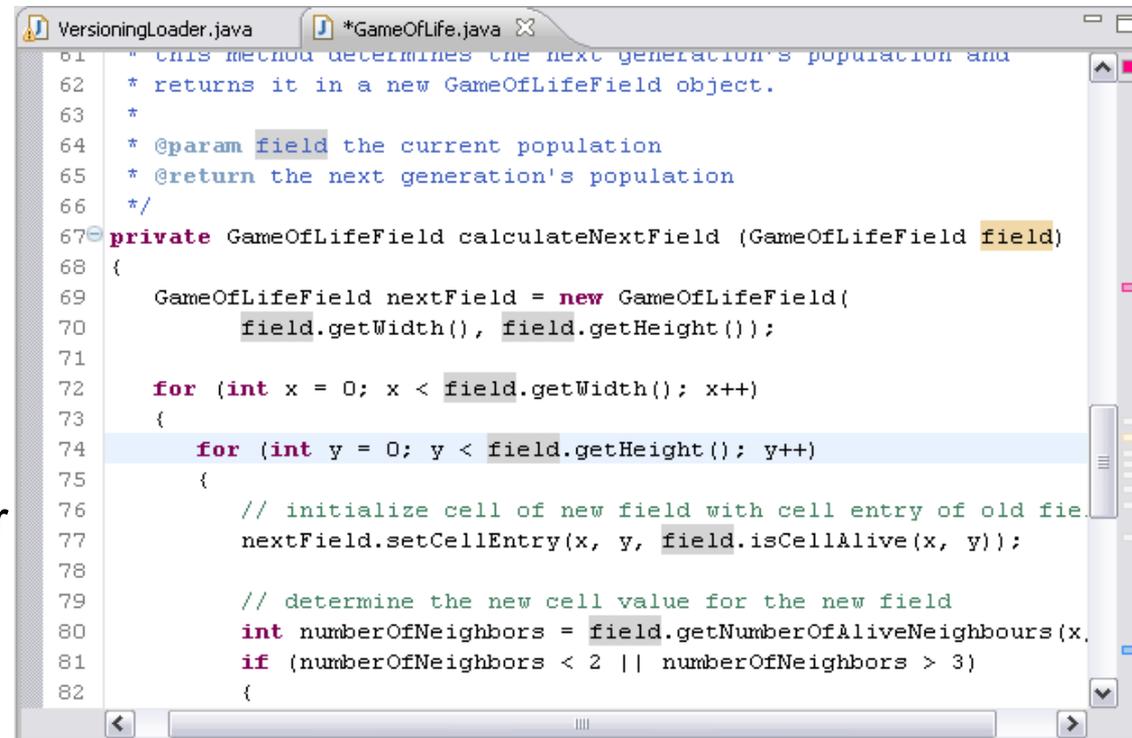
### Java-Editor

- 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



```
*VersioningLoader.java ExportFilesAction.java
111
112
113 public void actionPerformed (ActionEvent e)
114 {
115     MessageView messageView = FrameMain.get().getMessageView();
116     messageView.deleteMessages (MESSAGE_CLASS_EXPORT);
```



```
VersioningLoader.java *GameOfLife.java
61 * this method determines the next generation's population and
62 * returns it in a new GameOfLifeField object.
63 *
64 * @param field the current population
65 * @return the next generation's population
66 */
67 private GameOfLifeField calculateNextField (GameOfLifeField field)
68 {
69     GameOfLifeField nextField = new GameOfLifeField(
70         field.getWidth(), field.getHeight());
71
72     for (int x = 0; x < field.getWidth(); x++)
73     {
74         for (int y = 0; y < field.getHeight(); y++)
75         {
76             // initialize cell of new field with cell entry of old fie
77             nextField.setCellEntry(x, y, field.isCellAlive(x, y));
78
79             // determine the new cell value for the new field
80             int numberOfNeighbors = field.getNumberOfAliveNeighbours(x,
81             if (numberOfNeighbors < 2 || numberOfNeighbors > 3)
82             {
```

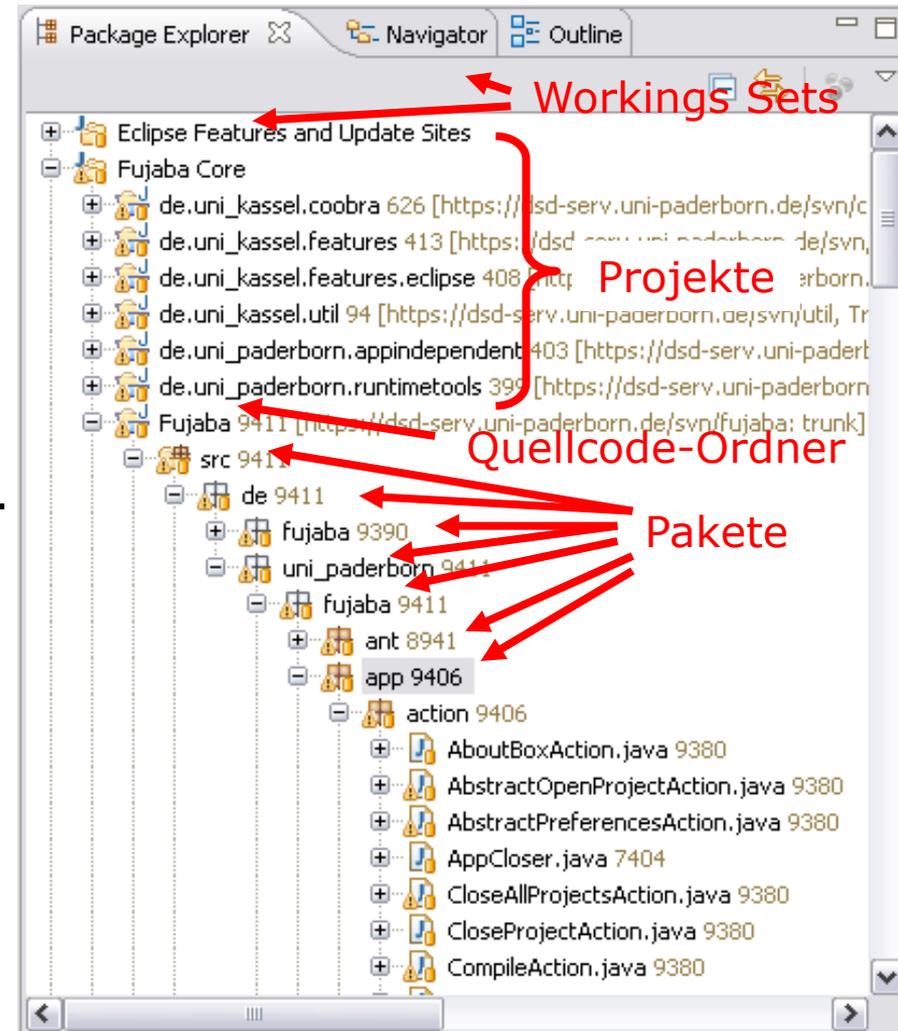
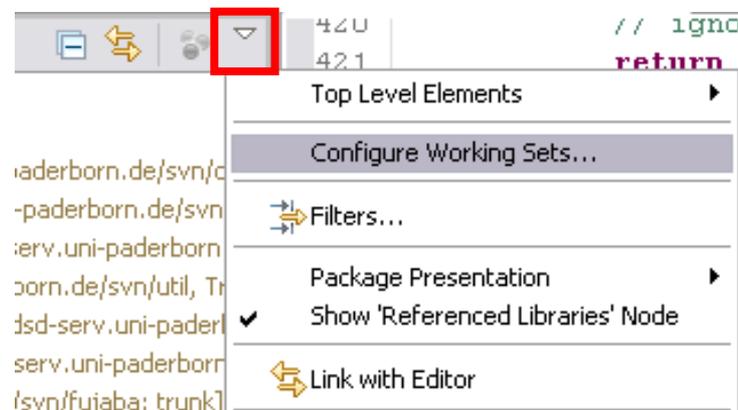
## 2. Java-Programmierung in Eclipse



### Views

Beliebige View öffnen über  
*Window* → *Show View* → ...

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



## 2. Java-Programmierung in Eclipse

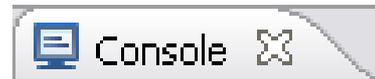


### Views

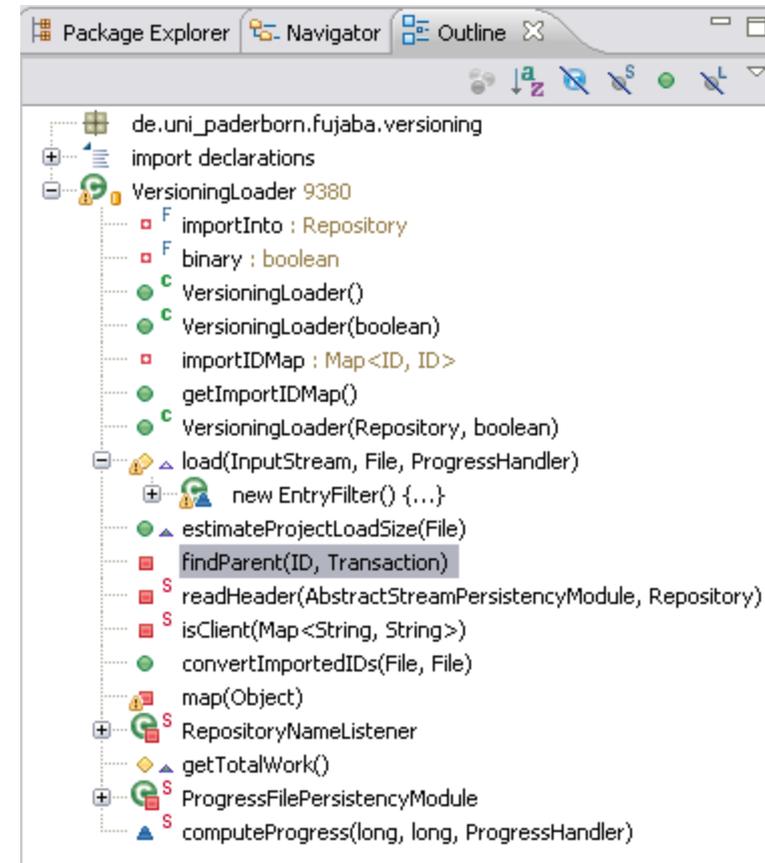
#### ■ *Outline*

- (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()`



## 2. Java-Programmierung in Eclipse



### Views

- *Problems*
  - Kompilierfehler und Warnungen

```
*VersioningLoader.java | ExportFilesAction.java |
111 | // see java.awt.event.ActionListener#actionPerformed(java.e
112 | */
113 | public void actionPerformed (ActionEvent e) {
114 | {
115 |     MessageView messageView = FrameMain.get().getMessageView();
116 |     messageView.deleteMessages (MESSAGE_CLASS_EXPORT);
```

Description	Resource	Path	Locat...
1 error, 2 warnings, 0 others			
Fatal Errors (1 item)			
Syntax error on token ")", delete this token	ExportFiles...	Fujaba/src/de/un...	line 113
Type Safety and Raw Types (2 items)			
Iterator is a raw type. References to generic type Iterator<E> should be parameterized	ExportFiles...	Fujaba/src/de/un...	line 141
Iterator is a raw type. References to generic type Iterator<E> should be parameterized	ExportFiles...	Fujaba/src/de/un...	line 143

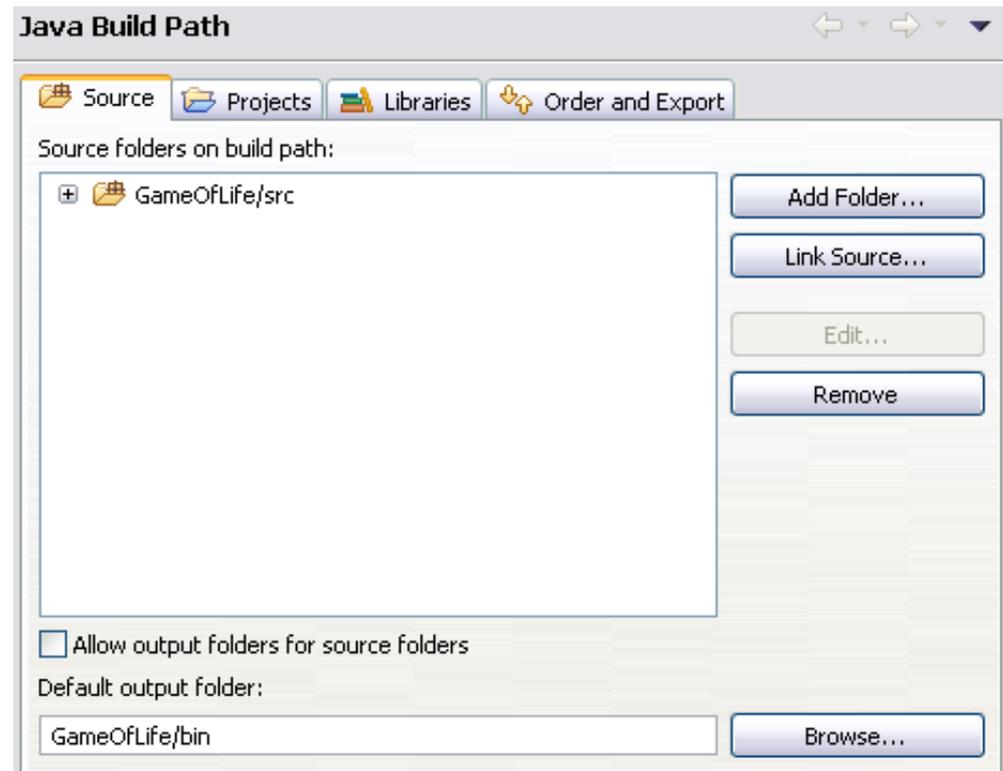
```
ProgressBar.java |
372 | // FIXME: might miss an update, too
373 | if (!queued)
374 | {
```

- *Tasks*
  - Markierungen von Aufgaben im Quellcode

Description	Resource	Path
5 items		
FIXME : might miss an update, too	ProgressBar....	Fujaba/src/de/uni...
FIXME : this does not work in surefire (maven) - s...	FujabaApp.j...	Fujaba/src/de/uni...
TODO : make sure all preferences will be serialized	ExitAction.java	Fujaba/src/de/uni...
TODO : present a list of unsaved projects to the ...	ExitAction.java	Fujaba/src/de/uni...
TODO remove path prefix	CompileActio...	Fujaba/src/de/uni...

### 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*

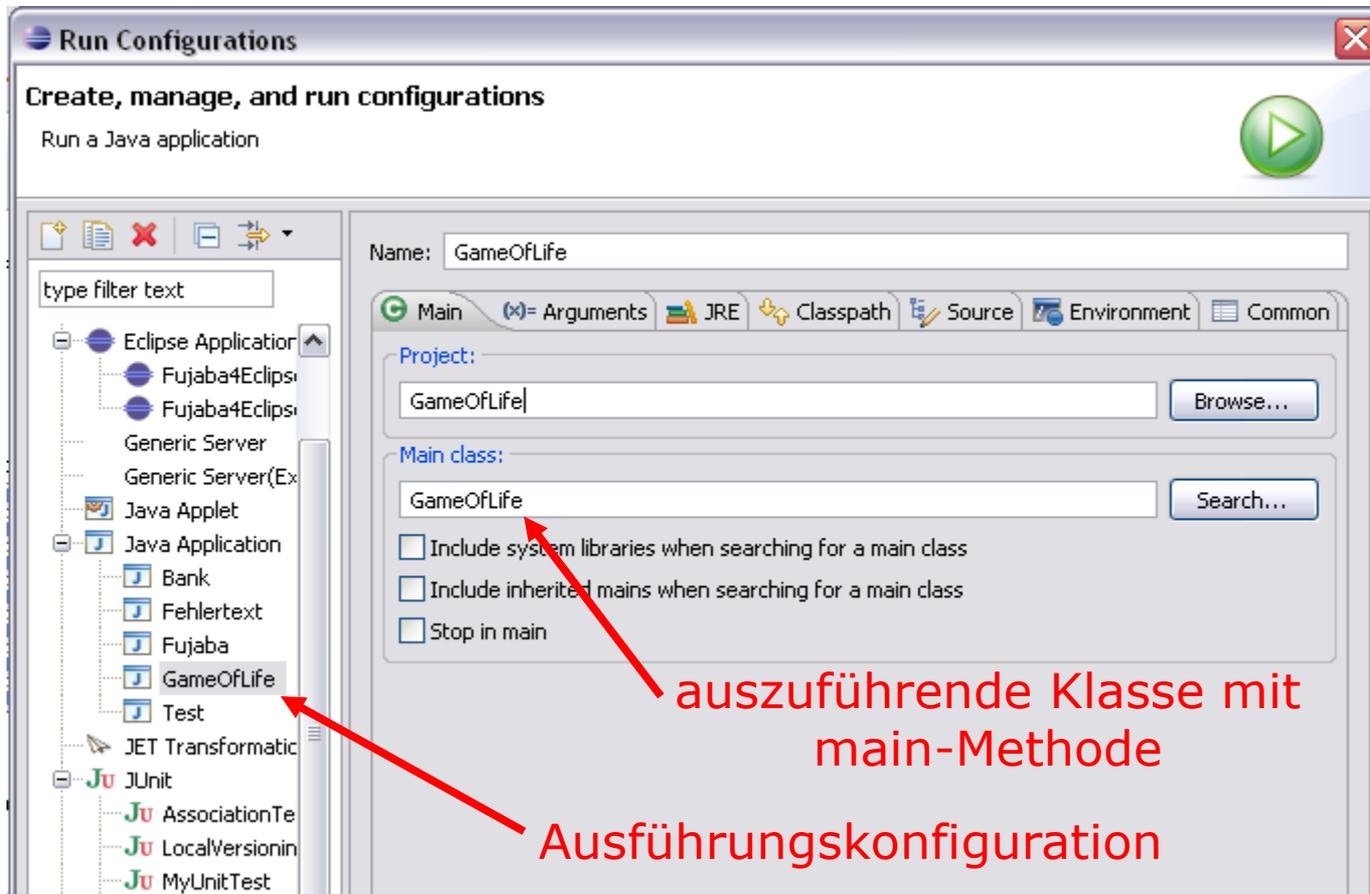


## 2. Java-Programmierung in Eclipse

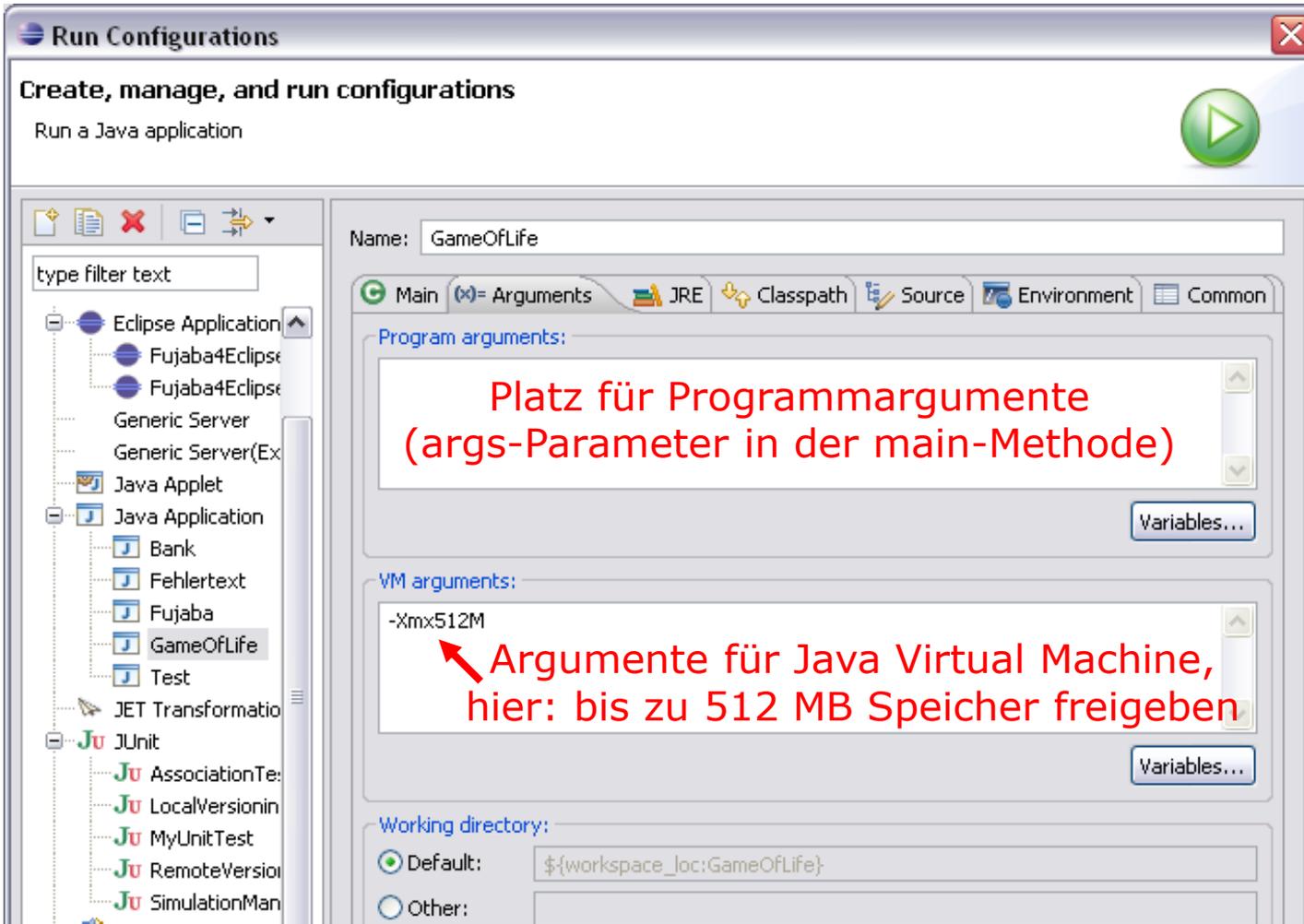


### Ausführung von Java-Programmen

- Einstellungen unter *Run* → *Run Configurations...*



# Ausführung von Java-Programmen



The screenshot shows the Eclipse Run Configurations dialog box. The title bar reads "Run Configurations" and the subtitle is "Create, manage, and run configurations". Below the subtitle, it says "Run a Java application". On the left, there is a tree view of configurations, with "GameOfLife" selected under the "Java Application" category. The main area shows the configuration for "GameOfLife". The "Name" field contains "GameOfLife". The "Main" tab is selected, showing "Program arguments:" and "VM arguments:". The "Program arguments:" field is empty, with a red text overlay: "Platz für Programmargumente (args-Parameter in der main-Methode)". The "VM arguments:" field contains "-Xmx512M", with a red arrow pointing to it and a red text overlay: "Argumente für Java Virtual Machine, hier: bis zu 512 MB Speicher freigeben". The "Working directory:" section has "Default:" selected with the value "\${workspace\_loc:GameOfLife}" and "Other:" is empty. There are "Variables..." buttons for both argument fields.

### 3. Eclipse erweitern durch Plug-ins



Entwicklung größtenteils analog zur „gewöhnlichen“ Java-Programmierung

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



- **Plug-in-Projekt**

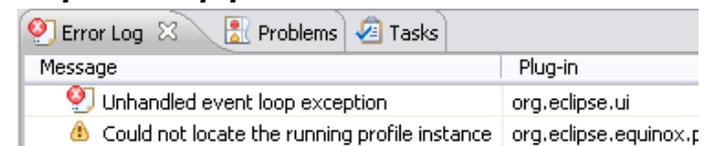


- Anlegen durch *File* → *New* → *Other...* → *Plug-in Development* → *Plug-in Project*

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



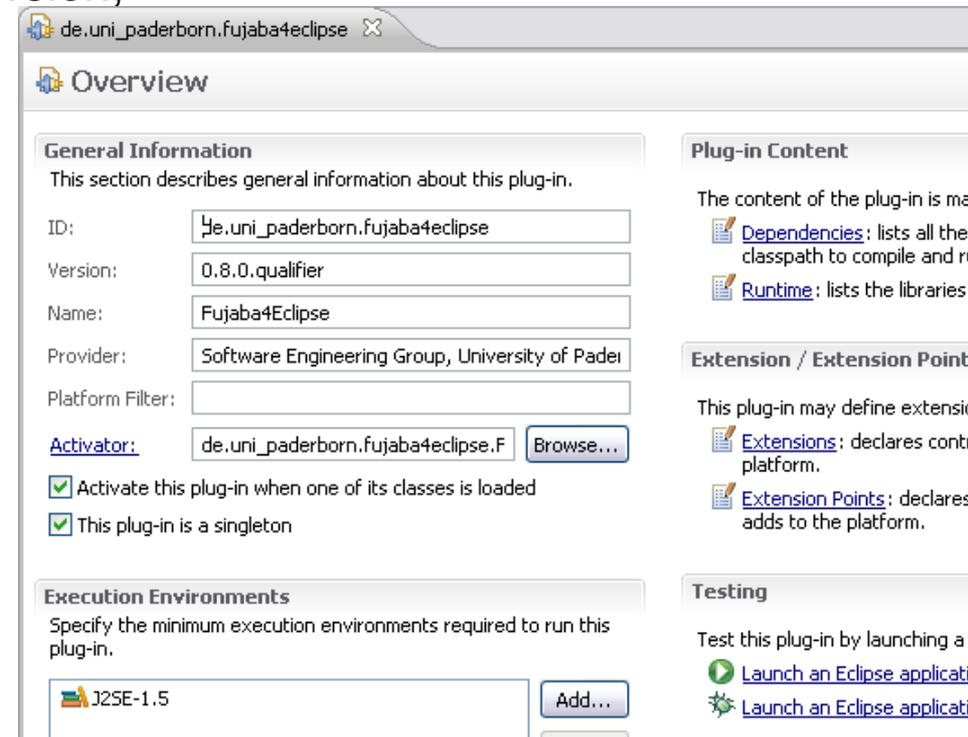
### 3. Eclipse erweitern durch Plug-ins



## 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!

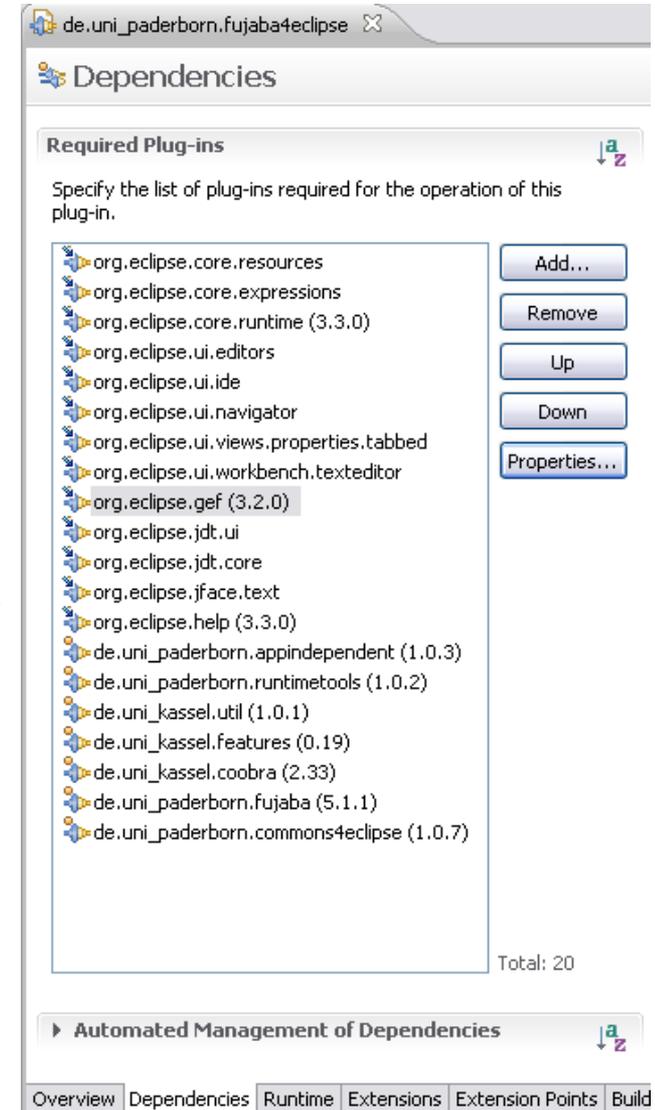


### 3. Eclipse erweitern durch Plug-ins



## 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*)



### 3. Eclipse erweitern durch Plug-ins



## 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

de.uni\_paderborn.myPlugin4Eclipse

### Build Configuration

Custom Build

#### Runtime Information

Define the libraries, specify the order in which they should be built, and list the source folders that should be compiled into each selected library.

**Binary Build**  
Select the folders and files to include in the binary build.

- .classpath
- .project
- META-INF
- bin
- build.properties
- icons
- plugin.xml
- src

**Source Build**  
Select the folders and files to include in the source build.

- .classpath
- .project
- META-INF
- bin
- build.properties
- icons
- plugin.xml
- src

▶ Extra Classpath Entries

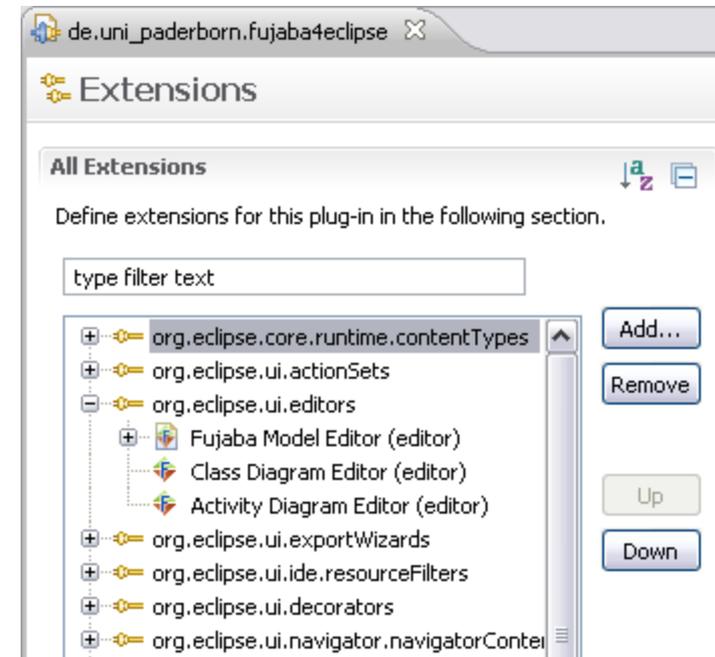
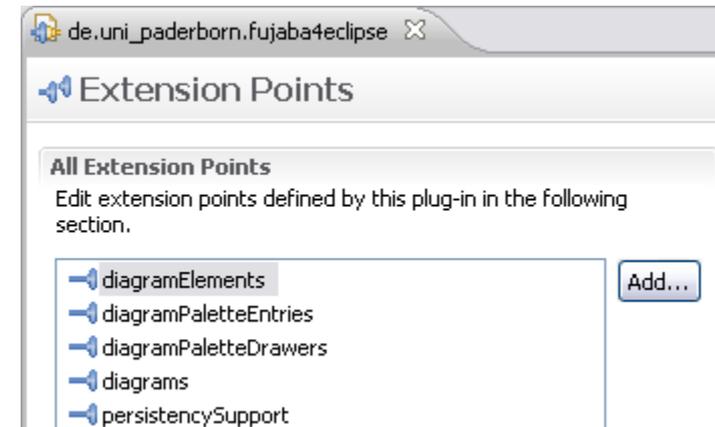
Overview Dependencies Runtime Extensions Extension Points Build MANIFEST.MF plugin.xml build.properties

### 3. Eclipse erweitern durch Plug-ins



## 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

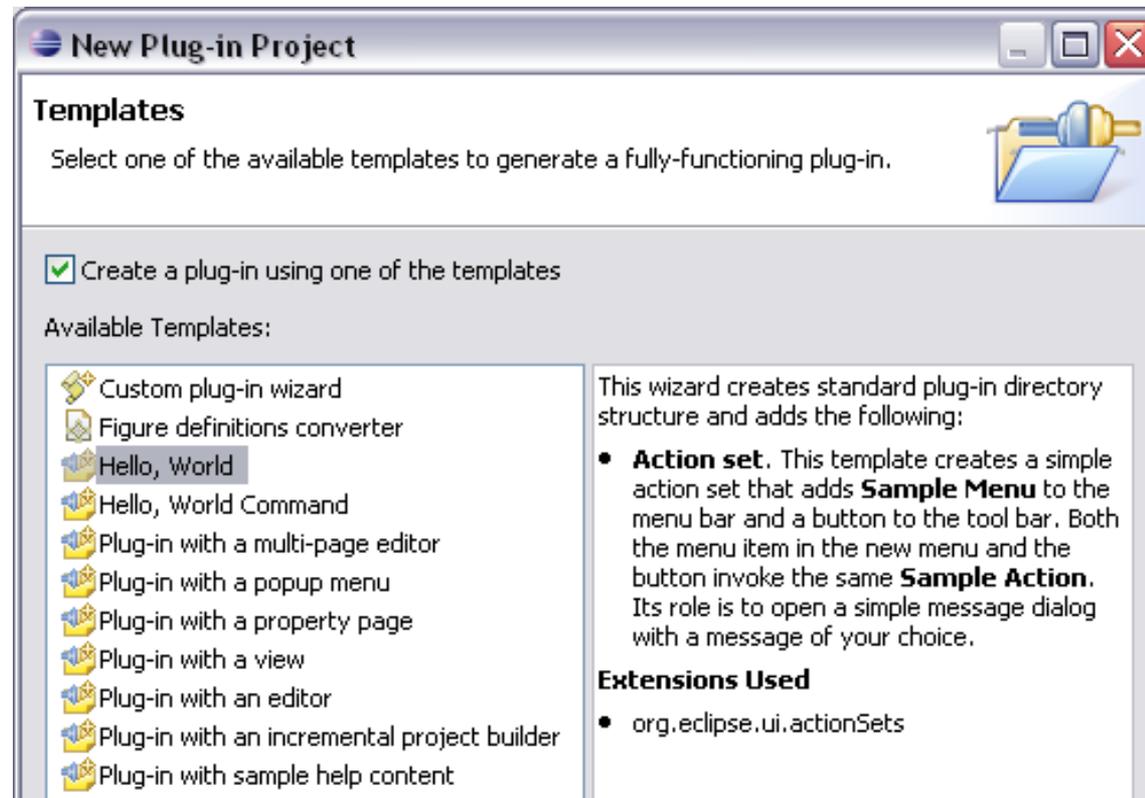


### 3. Eclipse erweitern durch Plug-ins



## 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.



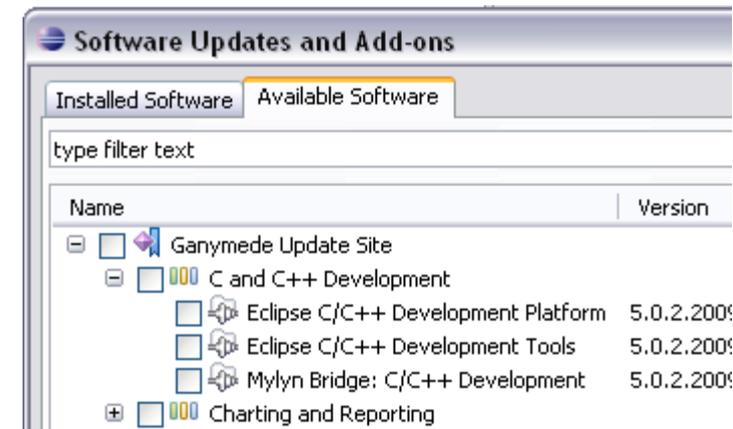
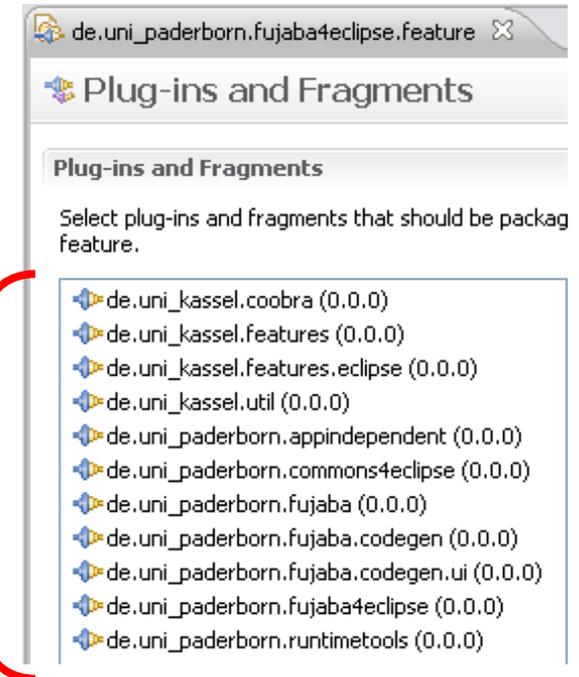
### 3. Eclipse erweitern durch Plug-ins



## 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

Plug-ins  
eines  
Features



# 4. Debugging und Testen



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



The screenshot shows the Eclipse IDE in Debug mode. The top toolbar has the Debug icon highlighted. The Debug Console shows the execution stack with the current line highlighted. The Variables view shows the current state of variables. The Source editor shows the code being debugged.

Kontrollleiste

Aktuelle Werte der Variablen

Laufzeit-Stack (Methodenaufrufe und Threads)

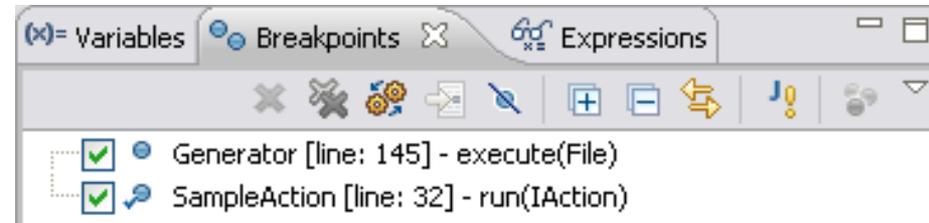
Aktuelle Position im ausgeführten Programm

## 4. Debugging und Testen



### 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*



## Haltepunkte im Programm (Breakpoints)

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

Properties for testhelloworld.actions.SampleAction [line: 32] - run(IAction)

Line Breakpoint

Type: testhelloworld.actions.SampleAction  
Line Number: 32  
Member: run(IAction)  
 Enabled  
 Hit Count:

Enable Condition (Ctrl+Space for code assist)

`action.getId().equals("myAction.id")`

Suspend when:  
 condition is 'true'  
 value of condition changes

Suspend Policy: Suspend Thread

```
testHelloWorld SampleAction.java
24
25 /**
26  * The action has been activated. The argument of the
27  * method represents the 'real' action sitting
28  * in the workbench UI.
29  * @see IWorkbenchWindowActionDelegate#run
30  */
31 public void run(IAction action) {
32     MessageDialog.openInformation(
33         window.getShell(),
34         "TestHelloWorld Plug-in",
35         "Hello, Eclipse world!");
36 }
37
```

# 4. Debugging und Testen



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

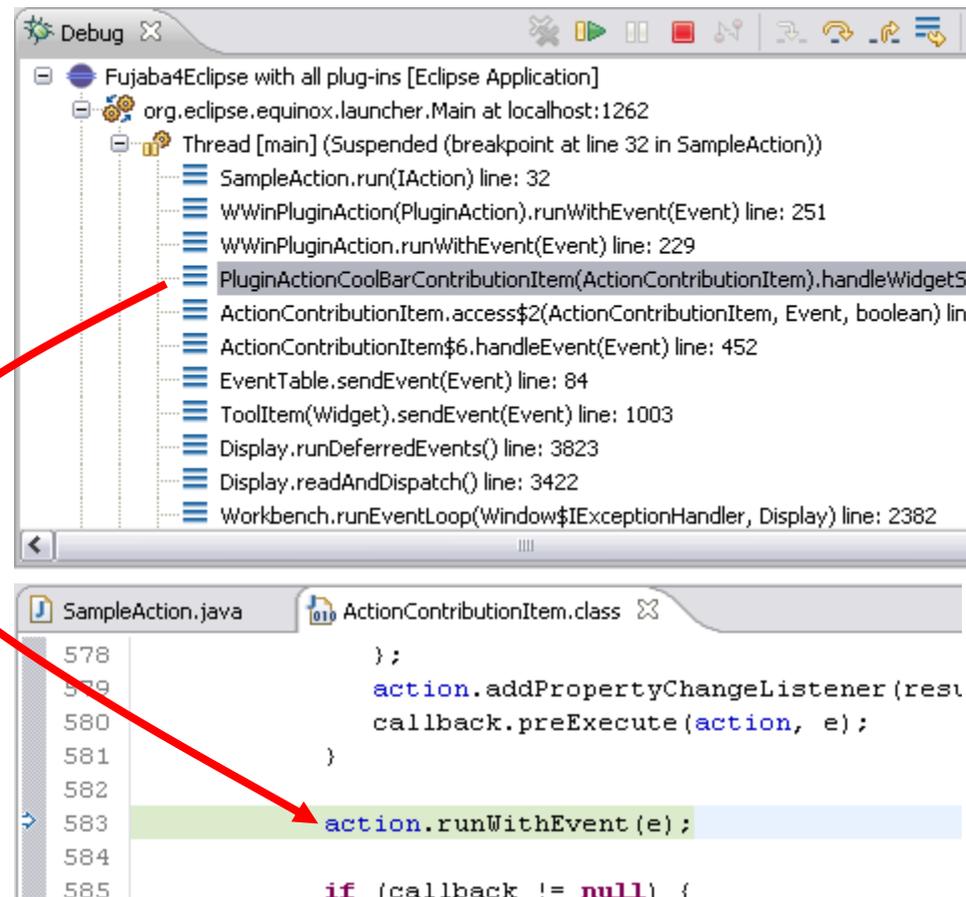
Bei fremdem Quellcode  
Quelle angeben  
(z.B. src.zip bei Java):

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



## Debug View

Sprung zu beliebiger Stelle im Stack möglich

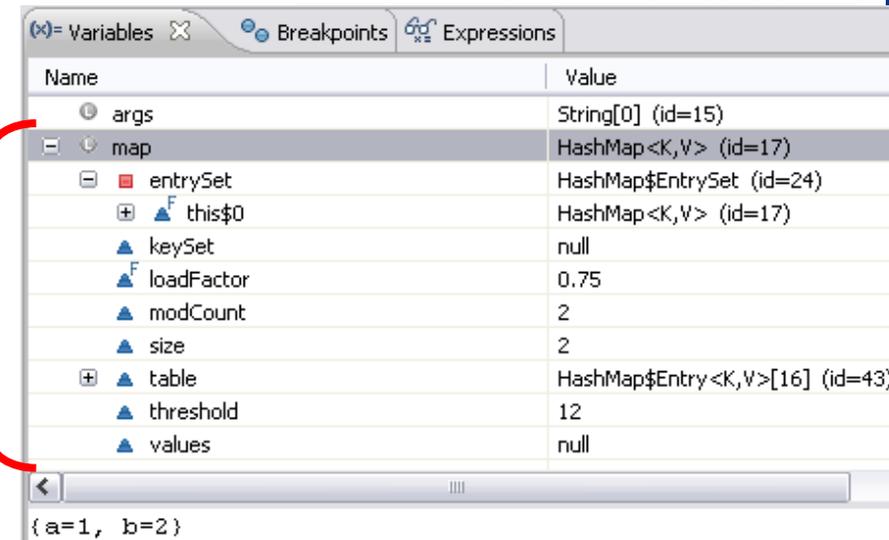


## 4. Debugging und Testen



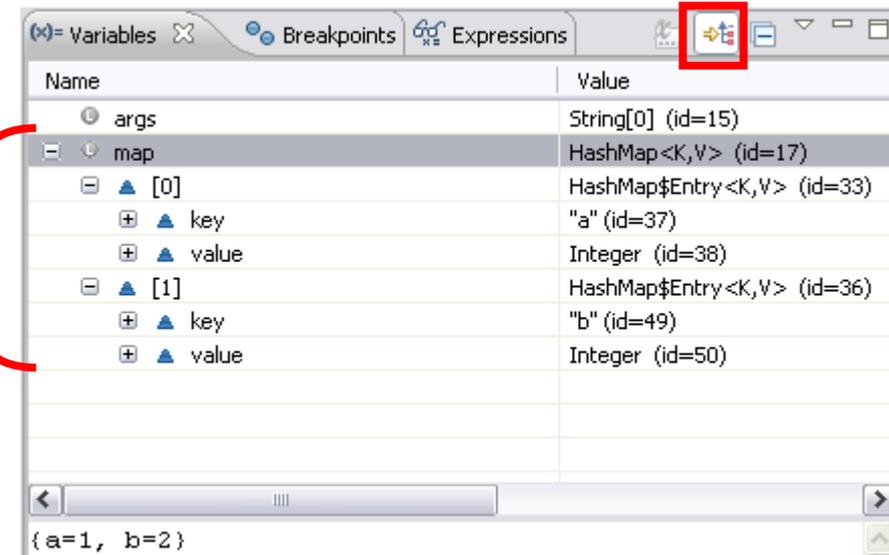
### Logische Strukturen

- 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*



Name	Value
args	String[0] (id=15)
map	HashMap<K,V> (id=17)
entrySet	HashMap\$EntrySet (id=24)
this\$0	HashMap<K,V> (id=17)
keySet	null
loadFactor	0.75
modCount	2
size	2
table	HashMap\$Entry<K,V>[16] (id=43)
threshold	12
values	null

{ a=1, b=2 }



Name	Value
args	String[0] (id=15)
map	HashMap<K,V> (id=17)
[0]	HashMap\$Entry<K,V> (id=33)
key	"a" (id=37)
value	Integer (id=38)
[1]	HashMap\$Entry<K,V> (id=36)
key	"b" (id=49)
value	Integer (id=50)

{ a=1, b=2 }

## 4. Debugging und Testen



### 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*

Tests fehlgeschlagen

Tests erfolgreich

Fehlerbeschreibung

```
MyUnitTest.java AssociationTest.java
11
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



- 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
  - EciEMMA
  - ...

The screenshot shows the Eclipse Downloads website. The navigation bar includes links for Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. The main heading is 'Eclipse Downloads'. Below this, there are tabs for 'Packages', 'Developer Builds', and 'Projects'. A dropdown menu is set to 'Eclipse Indigo (3.7.2) Packages for Windows'. The main content area lists several Eclipse packages with their respective download counts and details. Two packages are circled in red: 'Eclipse IDE for Java EE Developers, 212 MB' and 'Eclipse Modeling Tools, 272 MB'.

Package Name	Size	Download Count	Details	Download Link
Eclipse IDE for Java EE Developers	212 MB	1,092,986 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse Classic 3.7.2	174 MB	819,485 Times	Details Other Downloads	Windows 32 Bit Windows 64 Bit
Eclipse IDE for Java Developers	128 MB	365,714 Times	Details	Windows 32 Bit Windows 64 Bit
SpringSource Tool Suite	-	-	Promoted Download Complete IDE for enterprise Java, Spring, Groovy, Grails and the Cloud.	Download
Eclipse IDE for C/C++ Developers (includes Incubating components)	108 MB	146,236 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse IDE for Java and Report Developers	243 MB	45,987 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse IDE for JavaScript Web Developers	110 MB	43,490 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse for RCP and RAP Developers	181 MB	32,744 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse Modeling Tools	272 MB	31,257 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse for Testers	90 MB	17,286 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse IDE for Parallel Application Developers (includes Incubating components)	181 MB	17,132 Times	Details	Windows 32 Bit Windows 64 Bit
Eclipse for Scout Developers	175 MB	3,101 Times	Details	Windows 32 Bit Windows 64 Bit

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

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

The screenshot shows the Eclipse Help Contents window. The left pane displays a tree view of help topics, with 'Basic workbench extension points using actions' selected. The right pane shows the content of this topic, including text and a diagram. The diagram illustrates how actions are used in the Eclipse workbench, with labels for 'View Action', 'Editor Actions', and 'Action Set Actions' pointing to specific icons in the Eclipse IDE's toolbar.

Platform Plug-in Developer Guide > Programmer's Guide > Plugging into the workbench

### Basic workbench extension points using actions

The workbench defines extension points that allow plug-ins to contribute behaviors to existing views and editors or to provide implementations for new views and editors. Using commands is covered in the [Basic workbench extension points using commands](#) section. Here we are going to take a look at the contributions to these extension points from one of the workbench sample applications, the readme tool.

The readme tool is a plug-in that provides custom editing and navigation for a specific resource, a **readme** file. The example shows many typical (but simplified) ways that extensions can be used to provide specialized tools.

The readme tool contributes to the menus of the navigator view, adds editor related actions to the workbench menus and tool bar, defines a custom view and content outline, and defines markers and marker resolutions. The figure below shows some of the customized features added to the workbench by the readme tool.

**View Action**      **Editor Actions**      **Action Set Actions**

Resource - sample1.readme - Eclipse SDK

File Edit Navigate Search Project Run Readme Window Help

sample1.readme

SAMPLE README FILE

1. SECTION 1  
This text is a placeholder for t  
1.1 Subsection  
This text is a placeholder fo

## 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>