

Eine Einführung in



für mathematisch- technische Assistenten / Informatik (IHK)

**Autor: Michael Wiedau
Version vom 31. August 2005**

Inhaltsverzeichnis

1.	Den Workspace wählen.....	3
2.	Ein neues Projekt erstellen	4
2.1.	Die Projektart wählen.....	4
2.2.	Den Projektnamen vergeben	5
2.3.	Die Java-Perspektive	7
3.	Eine neue Java-Klasse erstellen	8
3.1.	Die erste Klasse	9
4.	Kompilieren & Ausführen.....	10
4.1.	Ein Java-Programm starten	10
4.2.	Das Ausgabefenster.....	11
5.	Der Debugger	12
5.1.	Einen Breakpoint setzen.....	12
5.2.	Den Debugger starten.....	12
5.3.	Die Debug-Perspektive	13
6.	Fehlermeldungen.....	14
7.	Importieren von Java-Dateien	15
8.	Einstellungen.....	16
8.1.	Zeilennummern	16
9.	Templates & Shortcuts	17
9.1.	Wichtige Templates.....	17
9.2.	Schnelle Shortcuts	18
10.	Abbildungs- und Tabellenverzeichnis.....	19
10.1.	Abbildungen	19
10.2.	Tabellen.....	19

1. Den Workspace wählen

Wenn Sie Eclipse starten, werden Sie zunächst aufgefordert einen „Workspace“ zu wählen. Der Eclipse Workspace ist ein Verzeichnis auf ihrem Computer oder dem Server, in welchem alle zu ihrer Arbeitsumgebung notwendigen Dateien gespeichert werden. Jedes in Eclipse erstellte Projekt wird in diesem Verzeichnis durch ein Unterverzeichnis repräsentiert.

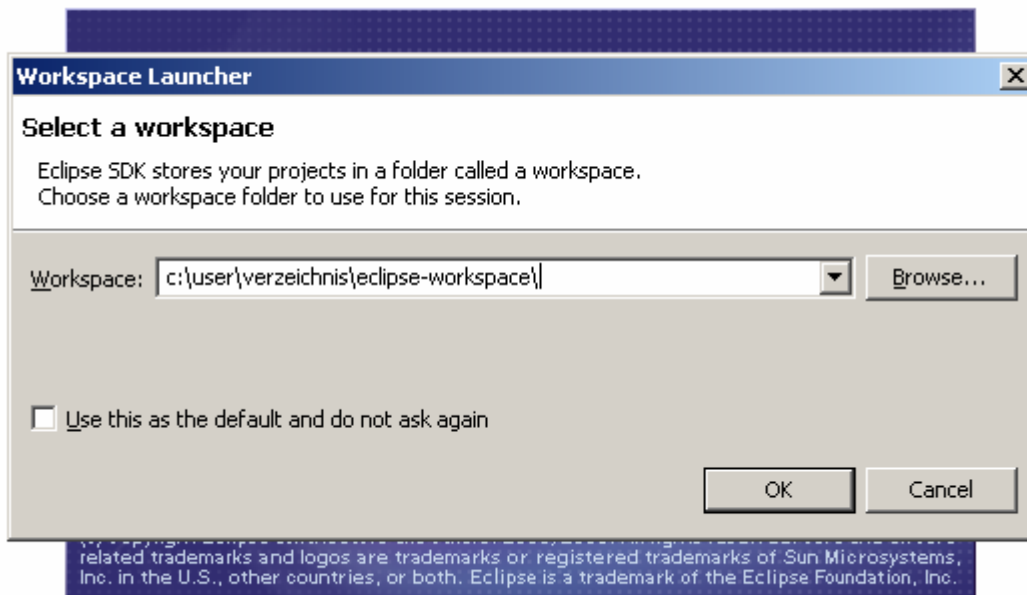


Abbildung 1 - Den Workspace auswählen

2. Ein neues Projekt erstellen

2.1. Die Projektart wählen

Um Java-Programme mit der Eclipse-Entwicklungsumgebung erstellen zu können, müssen Sie zunächst ein neues Java-Projekt erstellen. Wählen Sie dazu aus dem Menü *File* → *New* → *Project*. Im nun erscheinenden Fenster wählen Sie im Bereich „Wizards“ den Eintrag „Java Project“. Klicken Sie auf „Next“.

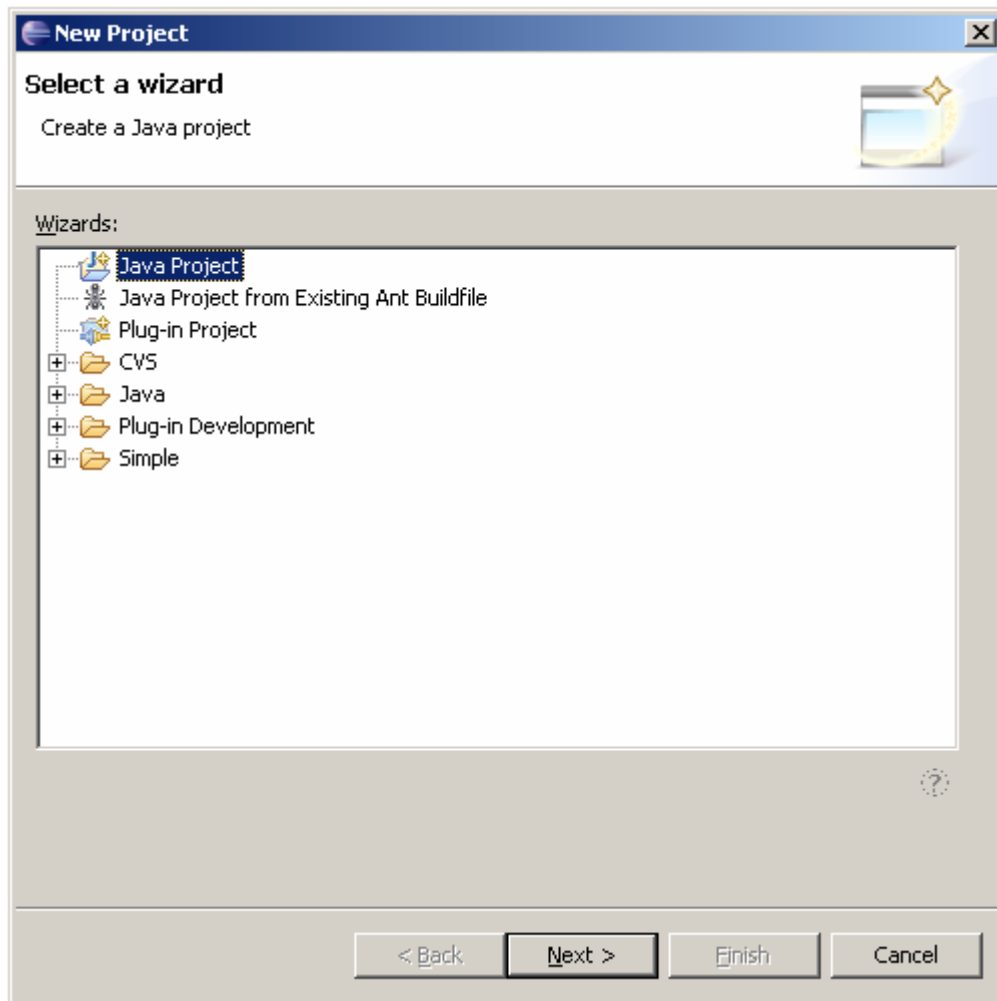


Abbildung 2 - Ein Java-Projekt erstellen

2.2. Den Projektnamen vergeben

Jedes in Eclipse erstellte Projekt muss einen Namen tragen. Im nächsten Fenster wählen Sie daher einen Namen für Ihr Projekt. Dieser Namen sollte mit einem Grossbuchstaben beginnen möglichst keine Leerzeichen oder Sonderzeichen (ä,ü,ö,ß, etc.) verwenden. Wählen Sie den Namen möglichst passend zum Einsatzzweck des Projektes. Wenn Sie z.B. in den Übungen verschiedene Aufgaben auf verschiedenen Übungsblättern durchführen, dann können Sie ihre Projekte z.B. mit „Vorkurs_blatt07_ag03“ bezeichnen. Da jedes Projekt als separates Unterverzeichnis in ihrem Workspace Verzeichnis erstellt wird, erhalten Sie so automatisch eine gute Gliederung in ihren Daten.

Wichtig!

Beim der ersten Erstellung wählen Sie bitte „Configure Default“ im Bereich „JDK Compliance“. In Abbildung 4 sehen Sie das nun erscheinende Fenster. Wählen Sie dort Java 5.0 aus. Bestätigen Sie die Änderung des Standard-JDK mit „OK“. Sie gelangen zurück zum Fenster aus Abbildung 3.

Geben Sie den Projektnamen ein und klicken Sie auf „Finish“.

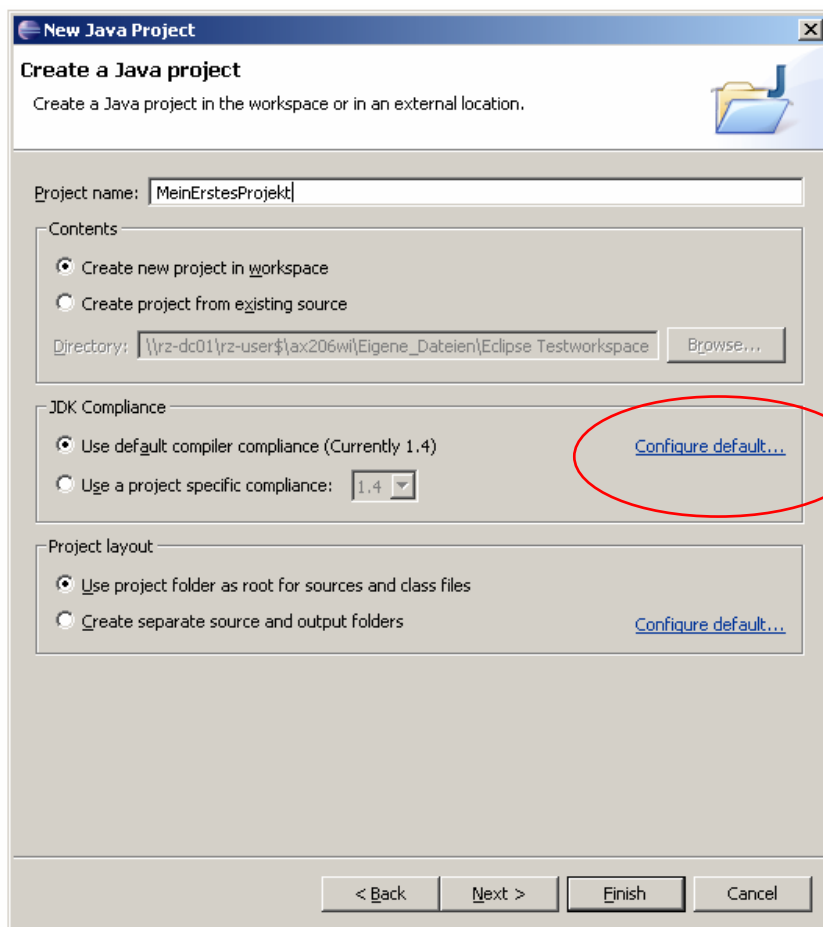


Abbildung 3 - Den Projektnamen wählen

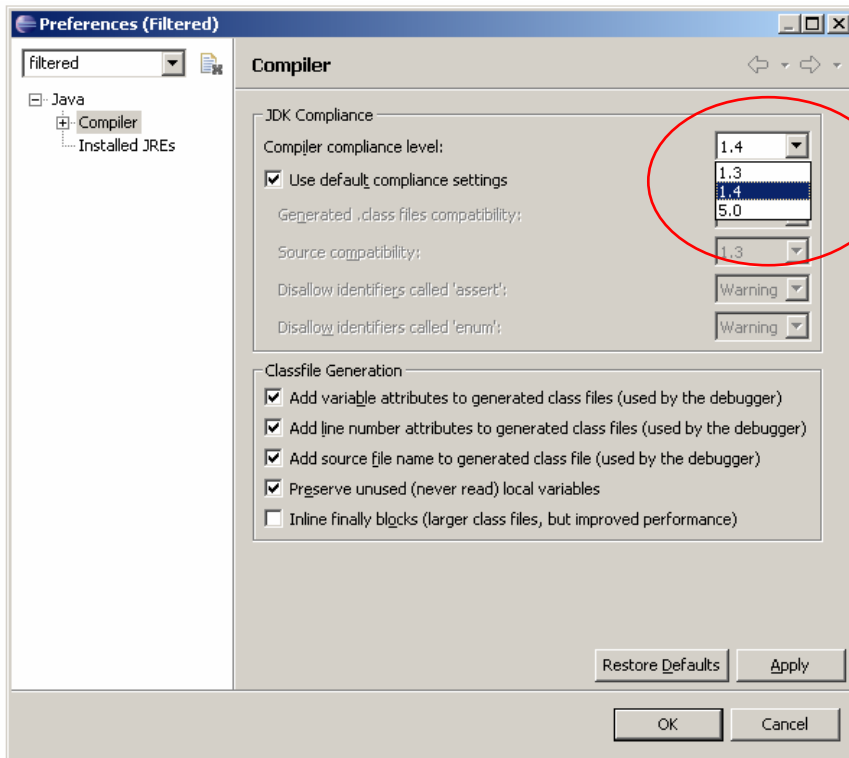


Abbildung 4 - Standard JDK auswählen

2.3. Die Java-Perspektive

Eclipse ist eine sehr umfangreiche Entwicklungsumgebung. Sie können mit ihr nicht nur Java-Programme schreiben, sondern auch viele andere Dinge machen. Für die Entwicklung von Java-Programmen schaltet Eclipse daher auf „Java-Perspektive“ um. Eclipse fragt unter Umständen nach, ob Sie auf die die Java-Perspektive wechseln möchten. Bestätigen Sie diese Frage positiv.

Ihr Eclipse Fenster enthält nun mehrer Unterfenster. (siehe Abbildung unten). Diese Unterfenster lassen sich untereinander verschieben. Dadurch können Sie die Perspektive an ihre individuellen Bedürfnisse anpassen.

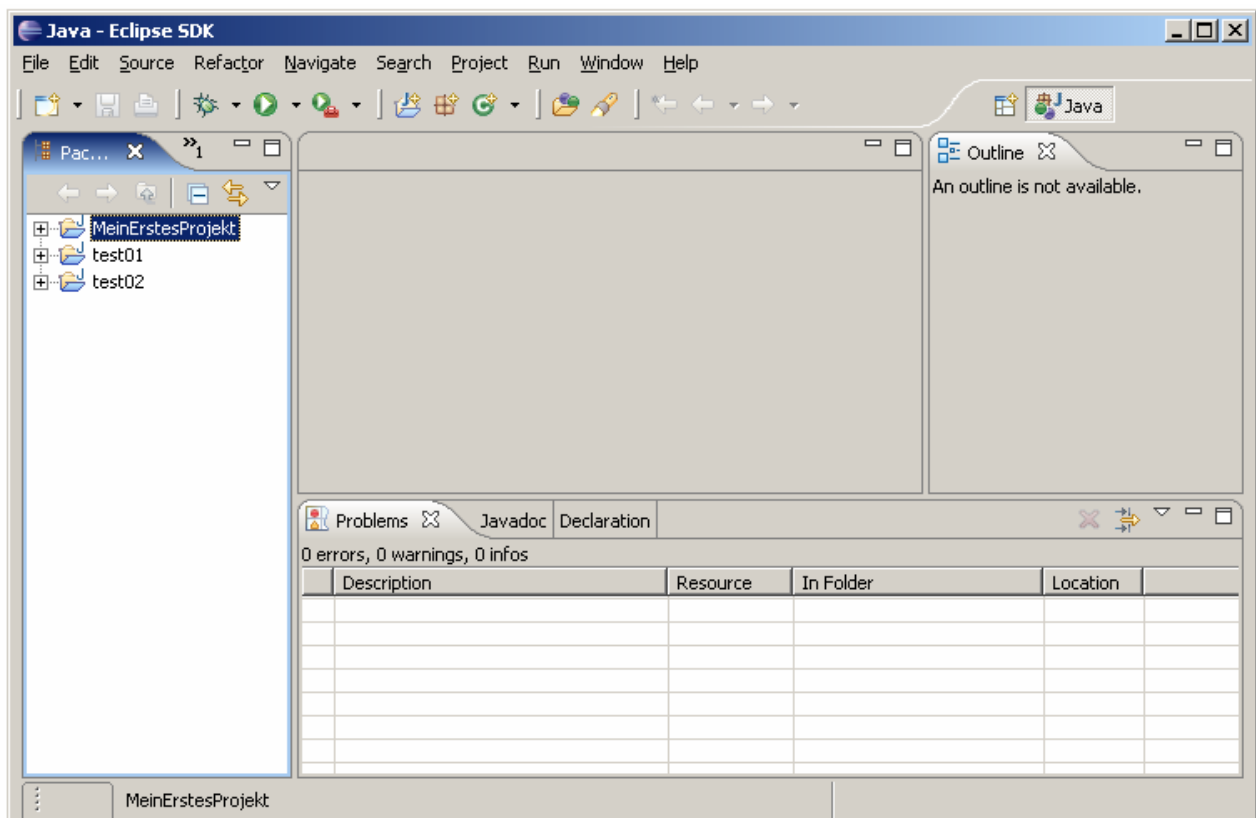


Abbildung 5 - Die Java-Perspektive mit den einzelnen Projekten

3. Eine neue Java-Klasse erstellen

Java-Programme benutzen Klassen um die Programmierung zu strukturieren. Um ein erstes Programm zu erstellen, müssen Sie eine neue Klasse in ihrem Projekt erstellen.

Wählen Sie dazu aus dem Menü *File* → *New* → *Class*.

Danach erscheint ein Fenster in dem Sie den Klassennamen eingeben müssen.

Wählen Sie die Methoden-Rümpfe (method stubs) wie in der Abbildung unten angegeben.

Klicken Sie auf „Finish“ um die Klasse erstellen zu lassen.

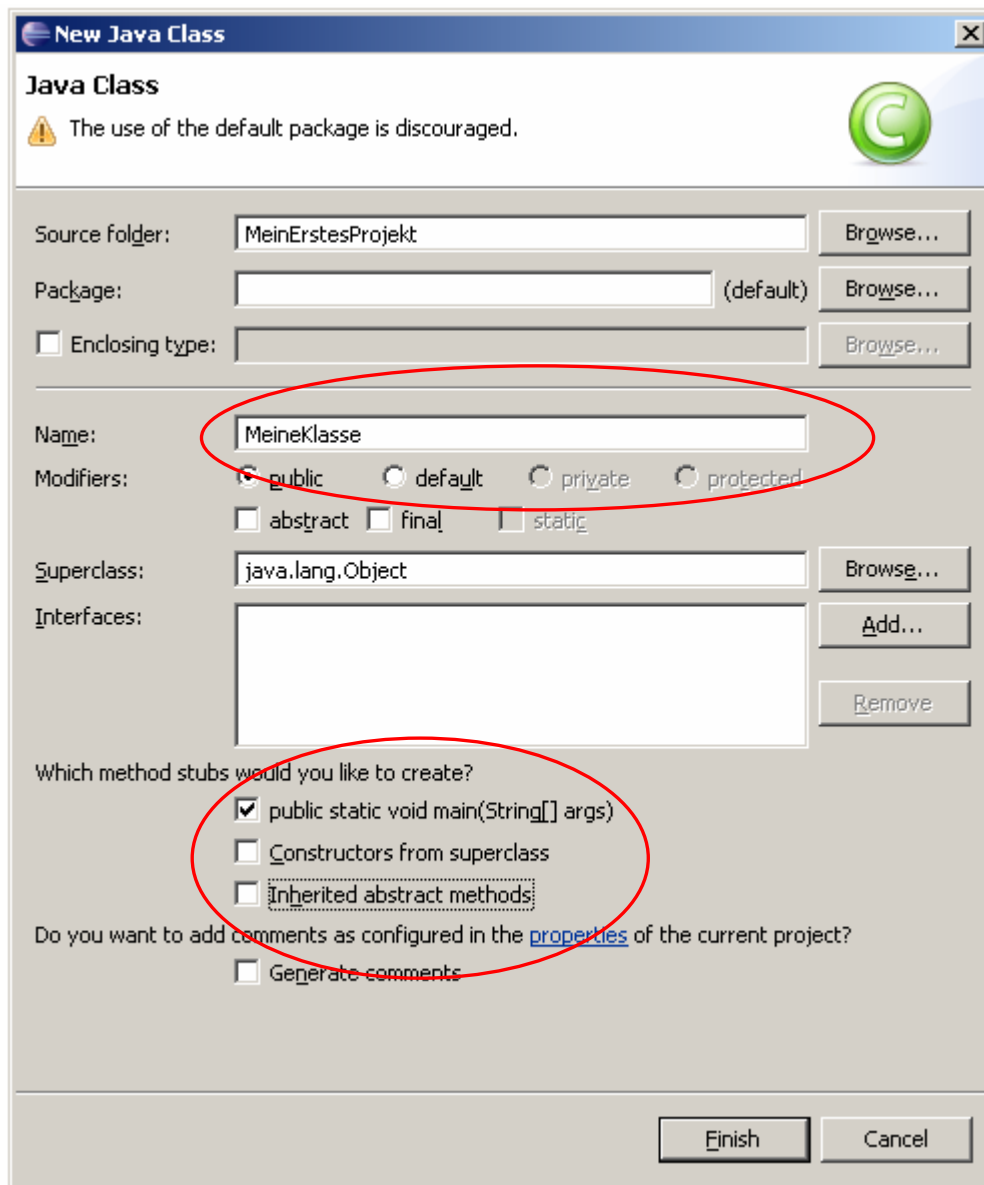


Abbildung 6 - Eine neue Klasse erstellen

4. Kompilieren & Ausführen

4.1. Ein Java-Programm starten

Zum Starten des Java-Programms klicken Sie mit der rechten Maustaste auf diejenige Klasse, welche ihre Main-Methode enthält. Wählen Sie nun, wie in Abbildung 8 gezeigt, aus dem Menü „Run As“ → „Java Application“.

Alternativ lässt sich auch die Tastenkombination Alt+Shift+X, J benutzen.

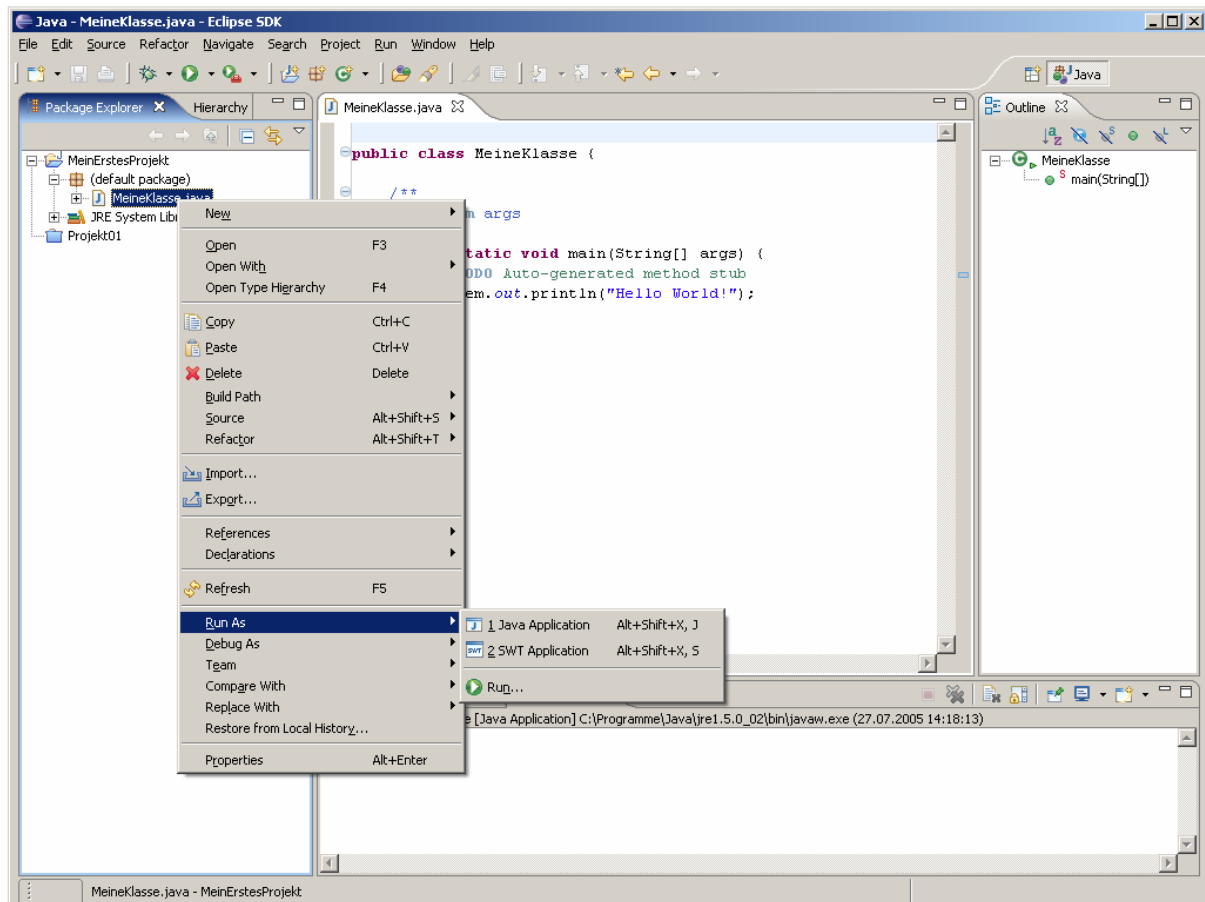



Abbildung 8 - Ein Java Programm ausführen

Nachdem die Klasse so das erste Mal zur Ausführung gebracht worden ist, reicht es ab dem zweiten Mal auf die grüne Pfeiltaste  in der oberen Symbolleiste zu klicken.

4.2. Das Ausgabefenster

Im Normalfall sollte im unteren Bereich neben den Reitern „Problems“, „Javadoc“ und „Declaration“ auch ein Reiter „Console“ zu finden sein. In diesem Reiter werden die von Java erzeugten Ausgaben hinein geschrieben. Sollte dieser Reiter bei Ihnen nicht vorhanden sein, so können sie ihn über das Menü „Window“ → „Show View“ → „Console“ oder die Tastenkombination Alt+Shift+Q, C zur Ansicht bringen.

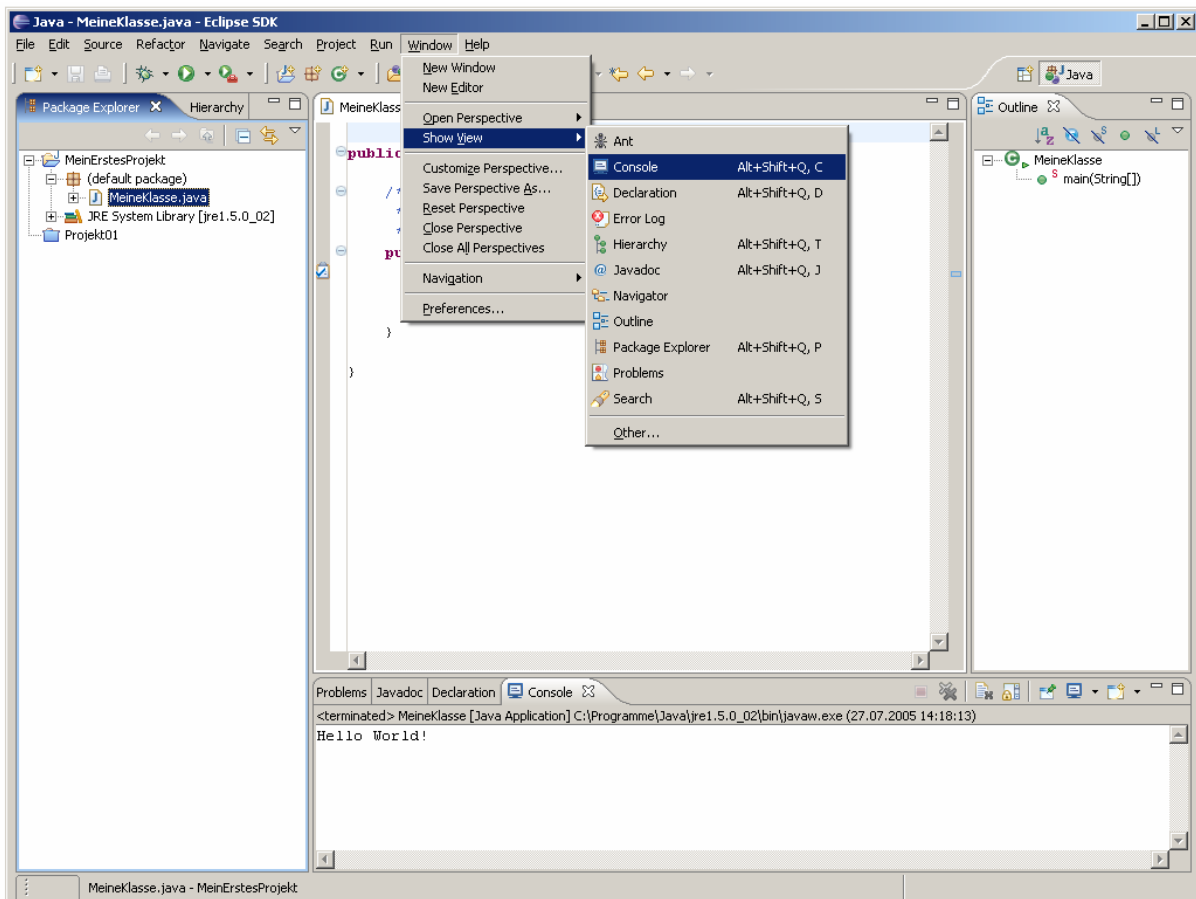


Abbildung 9 - Das Consolen-Fenster anzeigen

5. Der Debugger

Um bei der Programmerstellung den Inhalt von Variablen kontrollieren zu können, kann man sich die Variablen z.B. mit `System.out.println()` ausgeben lassen. Diese Vorgehensweise scheint zwar im ersten Moment die einfachste, aber es geht einfacher und komfortabler: Mit dem Debugger.

5.1. Einen Breakpoint setzen

Beim Debuggen kann das Programm bis zu einem bestimmten Punkt lauffengelassen und dann „pausiert“ werden. Dieser Punkt nennt sich „Breakpoint“. Ein Breakpoint lässt sich in einer beliebigen Programmzeile setzen (ausgenommen Kommentarzeilen, Leerzeilen, etc.). Um einen Breakpoint zu setzen, klicken Sie mit der Maus einfach auf die grau schraffierte Fläche vor der von Ihnen gewünschten Zeile. Nun erscheint dort ein Punkt. Sie können auch mit dem Cursor in die gewünschte Zeile gehen und die Tastenkombination `Strg+Shift+b` drücken.

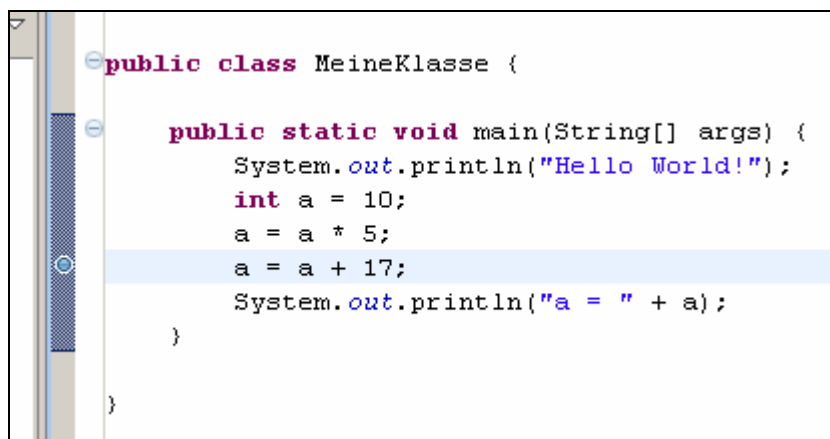



Abbildung 10 - Einen Breakpoint setzen

5.2. Den Debugger starten

Klicken Sie auf das kleine Käfer- (engl. Bug) Symbol  um den Debugger zu starten. Für den Debugger hat Eclipse eine eigene Perspektive. In der Perspektive sind die einzelnen Fenster neu angeordnet und es kommen weitere Unterfenster hinzu. Eclipse fragt sie, ob sie in die Debug-Perspektive wechseln wollen. Bestätigen Sie das mit „Yes“.

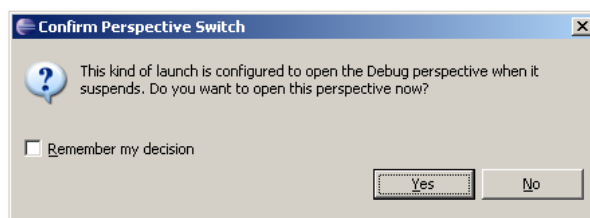


Abbildung 11 - In die Debug-Perspektive wechseln

5.3. Die Debug-Perspektive

In der Debug-Perspektive sehen Sie einige neue Unterfenster und einige alte Unterfenster in neuer Anordnung. In Abbildung 12 sehen Sie wie das Programm bis zum vorher gesetzten Breakpoint gelaufen ist (1). Es wird dort nun ein kleiner Pfeil über dem Breakpoint angezeigt. Dort ist das Programm nun stehe geblieben. Es hat bereits „Hello World!“ ausgegeben (2) und die Variable `a` besitzt nun den Wert 50 (3). Im Unterfenster „Variables“ werden alle Variablen des aktuellen Kontextes angezeigt. Sie können das Programm nun durch Drücken der gelb-grünen Pfeiltaste (4) bis zum nächsten Breakpoint laufen lassen. Es ist ebenfalls möglich das Programm zeilenweise weiter laufen zu lassen. Dabei gibt es zwei Tasten (5), die von Bedeutung sind. Mit der ersten Taste springen Sie in ein eventuelles Unterprogramm, mit der zweiten Taste übergehen Sie einen eventuellen Unterprogrammaufruf.

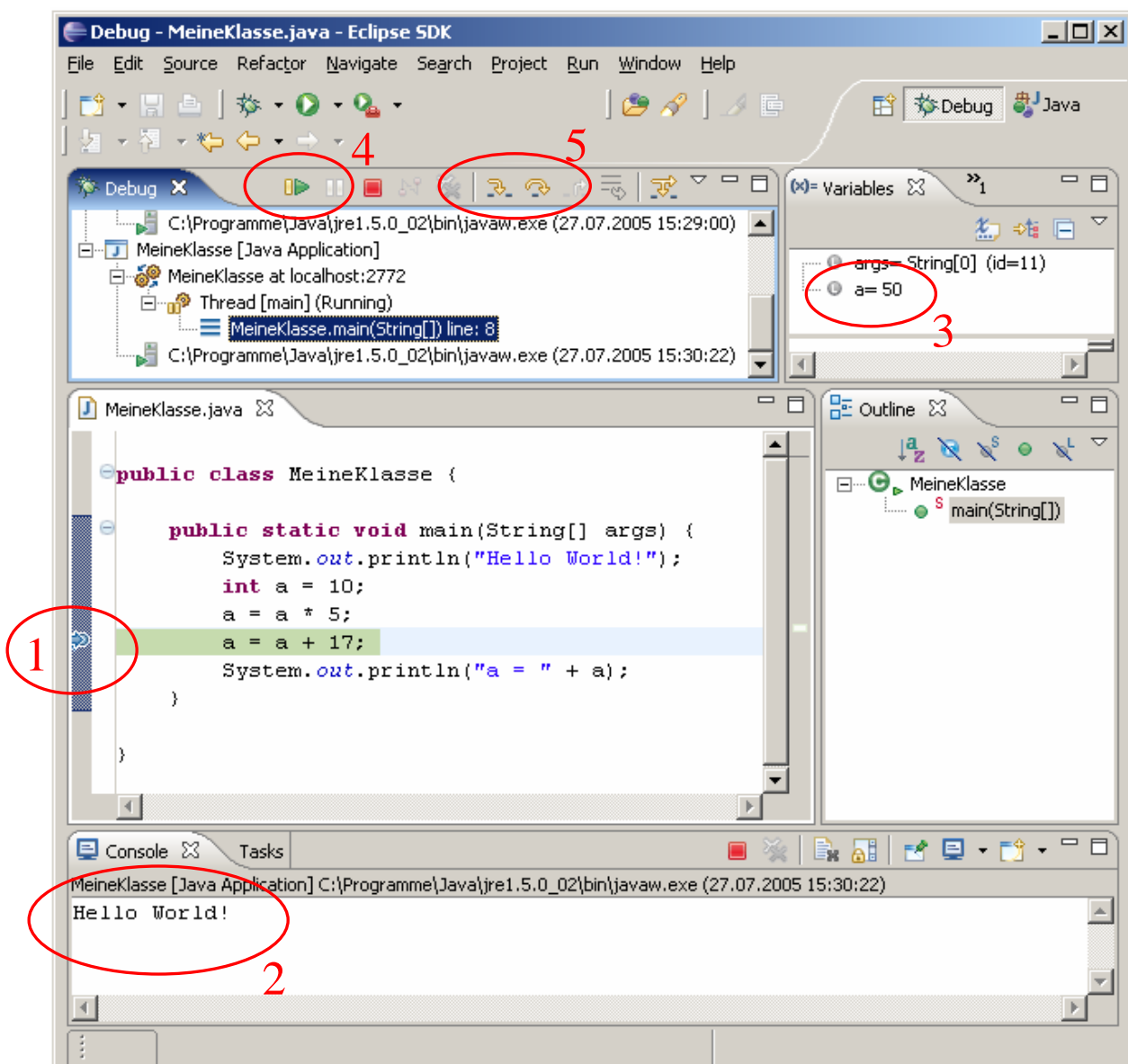


Abbildung 12 - Die Debug-Perspektive

6. Fehlermeldungen

Syntaxfehler werden in Eclipse bereits während der Eingabe markiert. Im Editorfenster werden die Fehler rot geschlängelt unterstrichen. Sie werden ebenfalls im Reiter „Problems“ angezeigt. Die Syntaxüberprüfung während der Eingabe kann abgeschaltet werden, in dem Sie das Menü „Window“ → „Preferences“ aufrufen und im dort erscheinenden Fenster unter „Java“ → „Editor“ den Haken vor „Report problems as you type“ entfernen.

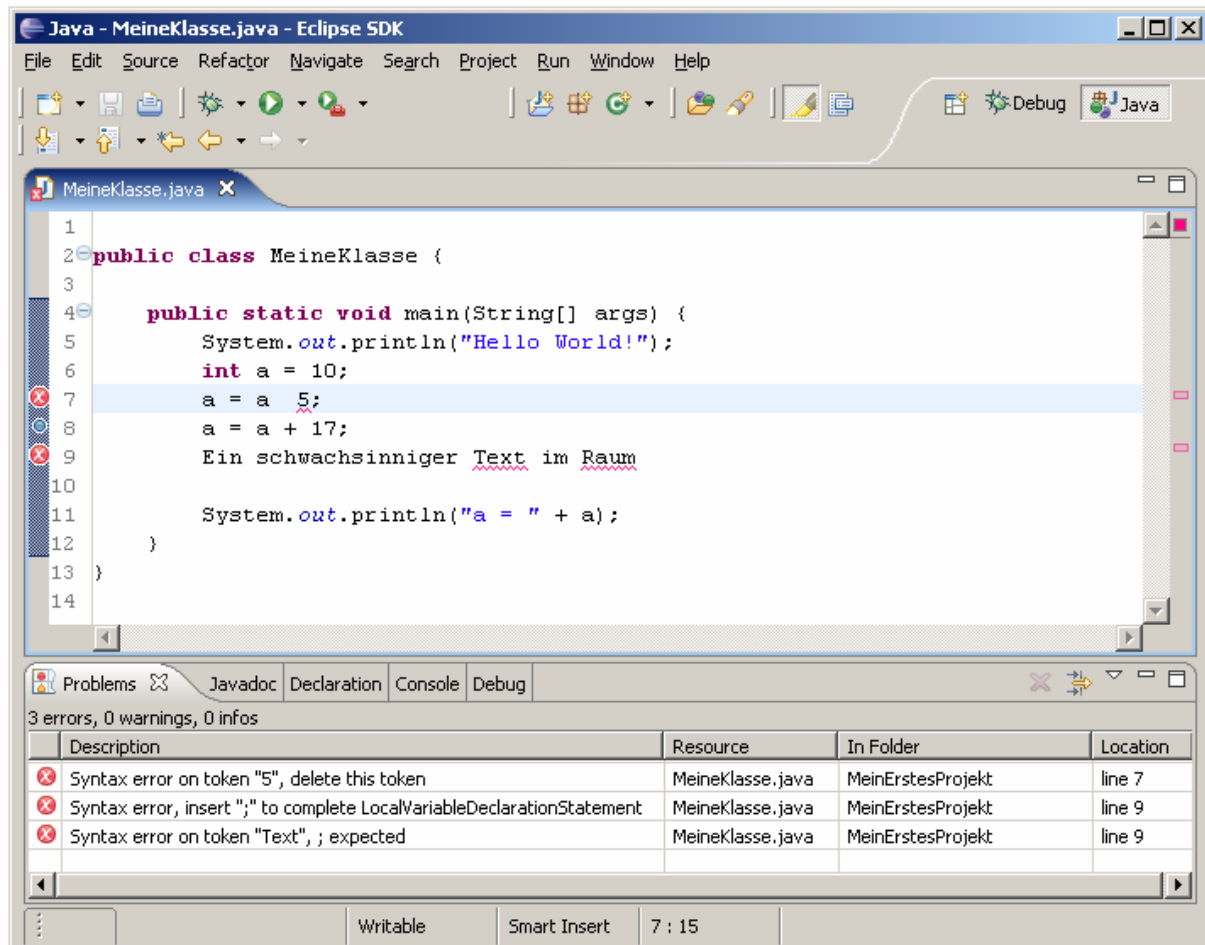


Abbildung 13 - Syntaxfehler während der Eingabe

7. Importieren von Java-Dateien

Zum Arbeiten mit Eclipse auf verschiedenen Rechnern empfiehlt es sich, den Eclipse-Workspace direkt auf einem USB-Stick abzuspeichern. Sie können einen lokal erstellten Workspace einfach über ihr Dateisystem (z.B. Microsoft Explorer oder Midnight Commander unter Linux) auf ihren USB-Stick und von dort auf einen zweiten Rechner kopieren. Sie können den Workspace auch direkt von Ihrem USB-Stick öffnen.

Sollten Sie Dateien aus einem anderen Eclipse Workspace bzw. Projekt importieren wollen, so können Sie im Paket-Explorer mit der rechten Maustaste auf Ihr aktuelles Projekt klicken und dann den Menüpunkt „Importieren...“ wählen.

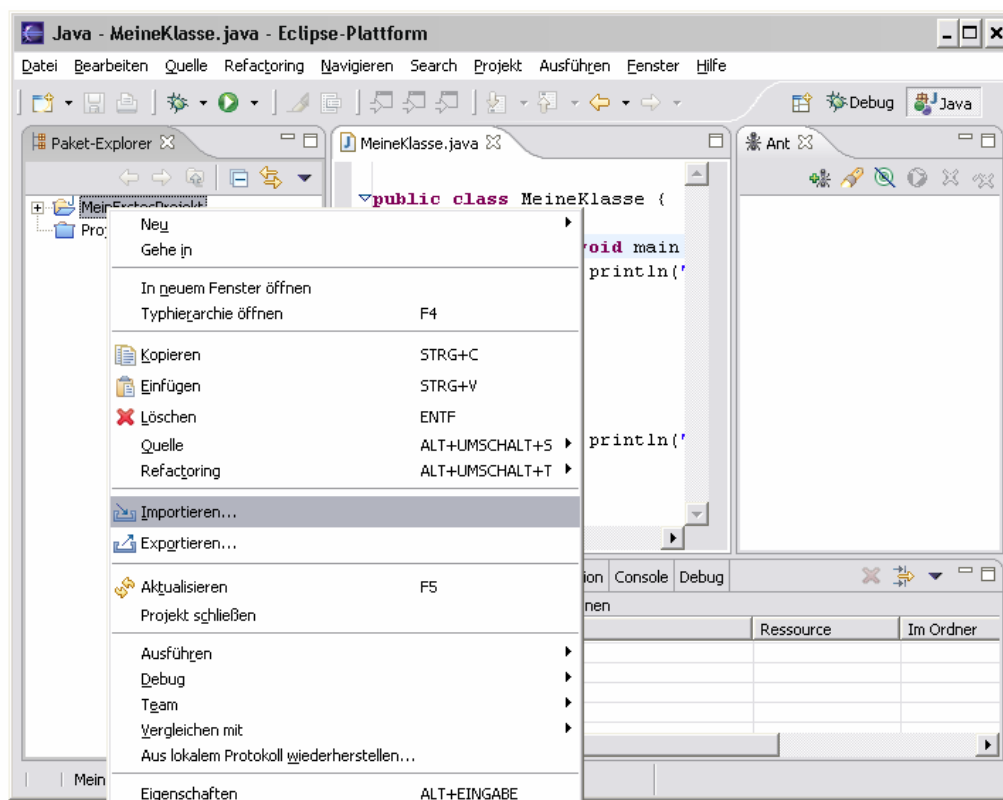


Abbildung 14 – Importieren (Screenshot von Eclipse 3.0, deutsch)

Im nun erscheinenden Fenster haben Sie verschiedene Möglichkeiten Dateien in Ihre Projekte zu importieren. Sie können z.B. „Filesystem“ für den Import aus einem anderen Java-Projekt benutzen.

8. Einstellungen

8.1. Zeilennummern

Sie können sich die Zeilennummerierung vor den Zeilen anzeigen lassen. Gehen Sie dazu über das Menü „Window“ → „Preferences“ in der Gruppe „General“ → „Editors“ → „TextEditors“ auf den Haken „Show Line Numbers“.

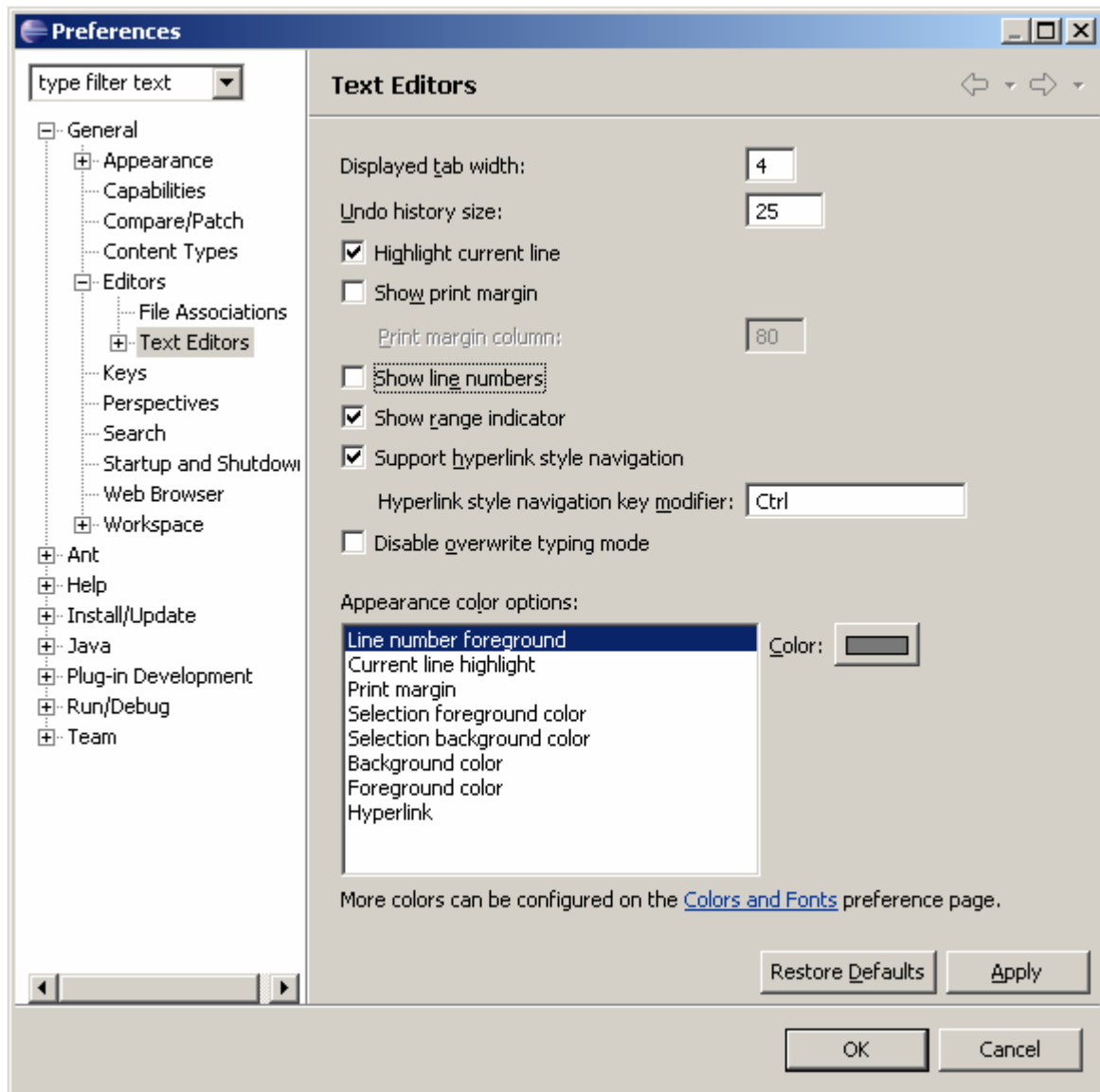


Abbildung 15 - Zeilennummerierung einschalten

9. Templates & Shortcuts

9.1. Wichtige Templates

In Eclipse können Sie sich ein bisschen Tipp-Arbeit sparen, indem Sie Templates für wichtige Befehle benutzen. Sie können diese Templates aufrufen, indem Sie den Kurznamen des Templates in den Editor schreiben (z.B. „sysout“) und dann die Tastenkombination „Strg+Leertaste“ drücken. Nun erscheint der vollständige Code (hier „System.out.println“).

Die wichtigsten Templates können Sie über das Menü „Window“ → „Preferences“ und dort über die Gruppe „Java“ → „Editor“ → „Templates“ sehen.

Hier eine kleine Auswahl der wichtigsten Templates:

Tabelle 1 - Wichtige Templates

Templatenname	Beschreibung	Erzeugter Code
sysout	print to standard output	<code>System.out.println();</code>
if	If statement	<code>if (...) {</code> <code>}</code>
ifelse	If else statement	<code>if (...) {</code> <code>} else {</code> <code>}</code>
main	Main method	<code>public static void main(String[] args) {</code> <code>}</code>
for	iterate over an array	<code>for (int i = 0; i < array.length; i++) {</code> <code>}</code>
while	while loop with condition	<code>while (...) {</code> <code>}</code>
try	try catch block	<code>try {</code> <code>} catch (Exception e) {</code> <code>}</code>

9.2. Schnelle Shortcuts

Viele Funktionen in Eclipse lassen sich mit Hilfe von Tastenkombinationen schneller aufrufen. Hier ist eine kleine Liste von wichtigen Tastenkombinationen:

Tabelle 2 - Shortcuts

Tastenkombination	Auswirkung
Alt + Shift + X, J	Java Anwendung ausführen
Alt + Shift + D, J	Java Anwendung debuggen
Alt + Shift + Q, C	Consolenfenster anzeigen
F8	Debugger weiterlaufen lassen (Resume)
F5	Debugger: Unterprogrammaufruf (Step Into)
F6	Debugger: Unterprogrammaufruf überspringen (Step over)
Strg + Shift + B	Breakpoint ein-/ausschalten
Strg + F	Suchen / Ersetzen
Strg + K	Nächstes Vorkommen finden
Strg + S	Aktuelle Datei speichern

10. Abbildungs- und Tabellenverzeichnis

10.1. Abbildungen

Abbildung 1 - Den Workspace auswählen.....	3
Abbildung 2 - Ein Java-Projekt erstellen	4
Abbildung 3 - Den Projektnamen wählen	5
Abbildung 4 - Standard JDK auswählen	6
Abbildung 5 - Die Java-Perspektive mit den einzelnen Projekten.....	7
Abbildung 6 - Eine neue Klasse erstellen	8
Abbildung 7 - Die erste Klasse im Java-Projekt	9
Abbildung 8 - Ein Java Programm ausführen	10
Abbildung 9 - Das Consolen-Fenster anzeigen.....	11
Abbildung 10 - Einen Breakpoint setzen	12
Abbildung 11 - In die Debug-Perspektive wechseln.....	12
Abbildung 12 - Die Debug-Perspektive	13
Abbildung 13 - Syntaxfehler während der Eingabe	14
Abbildung 14 – Importieren (Screenshot von Eclipse 3.0, deutsch).....	15
Abbildung 15 - Zeilennummerierung einschalten.....	16

10.2. Tabellen

Tabelle 1 - Wichtige Templates	17
Tabelle 2 - Shortcuts	18