# Forte 3.0 Community Edition Basics

*Installation und erste Schritte*


Siegfried Zeilinger, Institut für Softwaretechnik / TU Wien
Letzte Version @ www.ildico.com/eprog
Institut @ www.ifs.tuwien.ac.at/ifs/lehre/eprog
Version 1.1, September 2001

# 1. Hinweise

Sun und Forte sind eingetragene Warenzeichen der Sun Microsystems, Inc. und werden im Zusammenhang mit dieser Anleitung nicht jeweils explizit als solche gekennzeichnet.

# 2. Einleitung

Dieses Skriptum wurde erstellt, um den Umgang mit Forte community edition zu demonstrieren. Es ist für Einsteiger gemacht und soll anschaulich durch die einzelnen Schritte bis zu einem ersten Programm führen.
Es ist nicht nötig, gleichzeitig das Skriptum für Borland's JBuilder durchzusehen, weil diese beiden Skripten bis auf programmspezifische Dialogunterschiede gleich gehalten sind.
Die verwendete Forte Version ist 3.0 (build 010817). Versionsunterschiede können leider nicht berücksichtigt werden.
Feedback bitte an zeilinger@ildico.com oder anonym unter http://www.ildico.com/eprog.

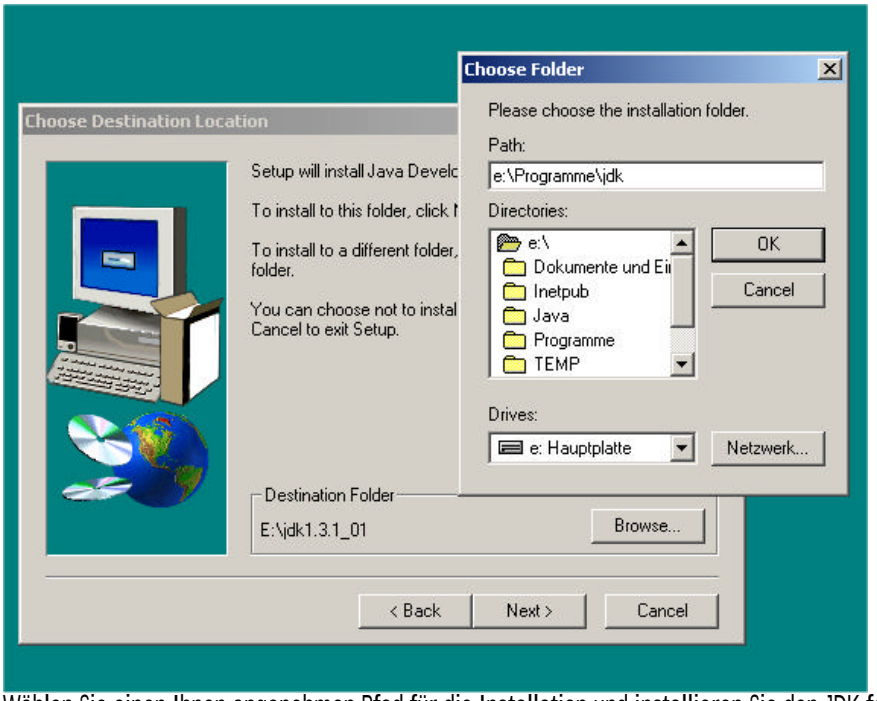
# 3. Installation von Forte

## *3.1. JDK installieren*

Bevor Sie Forte installieren können, brauchen Sie auf Ihrem System einen laufenden JavaDevelopmentKit (JDK).
In der aktuellen Version von Forte (3.0) wird JDK 1.3.1. oder größer empfohlen.
Dieses JDK bekommen Sie unter http://java.sun.com/j2se/1.3/

Die Seite ist derzeit (Sept 2001) etwas komisch aufgebaut. Zuerst wählen Sie Windows und auf der nächsten Seite findet sich der continue Schalter GANZ UNTEN auf der Seite. Die anderen Grafiken sollten nicht relevant sein.

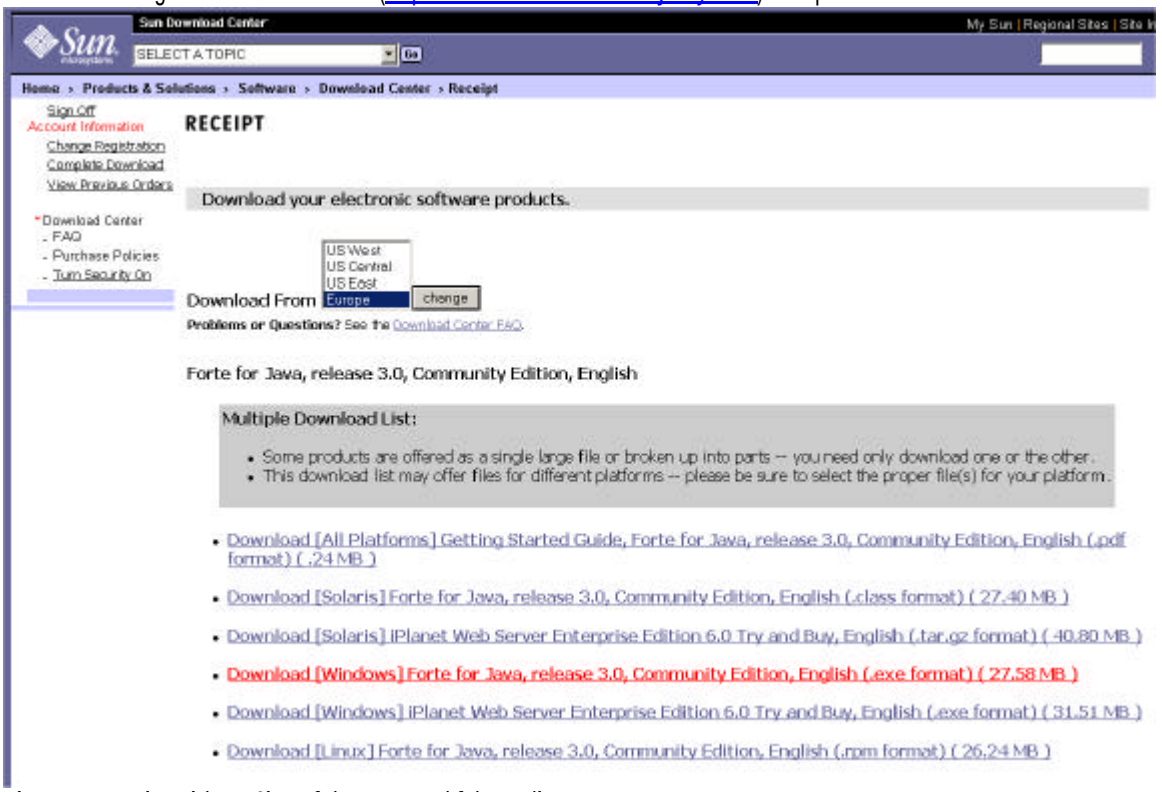
Akzeptieren Sie die Lizenzbedingungen und wählen Sie am besten einen amerikanischen (East Coast) Downloadserver. Diese sind im Allgemeinen schneller (34 MB Download !!!).
Speichern Sie die .exe Datei in ein bekanntes Verzeichnis und führen Sie diese aus.



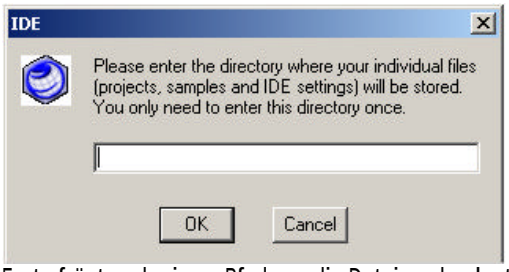

Wählen Sie einen Ihnen angenehmen Pfad für die Installation und installieren Sie den JDK fertig.

## *3.2. Forte installieren*

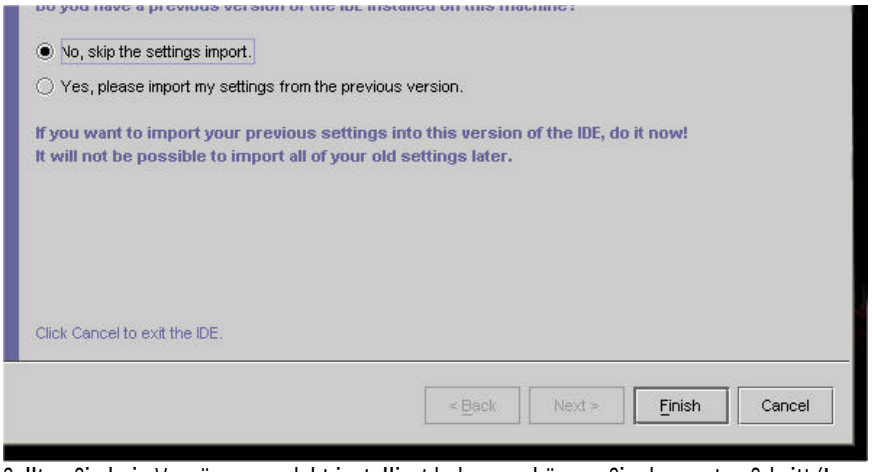
Forte kommt – gedownloaded von Sun (http://www.sun.com/forte/ffj/buy.html) - verpackt als .exe Datei.




Diese EXE Datei speichern Sie auf Ihrem Pc und führen diese aus.

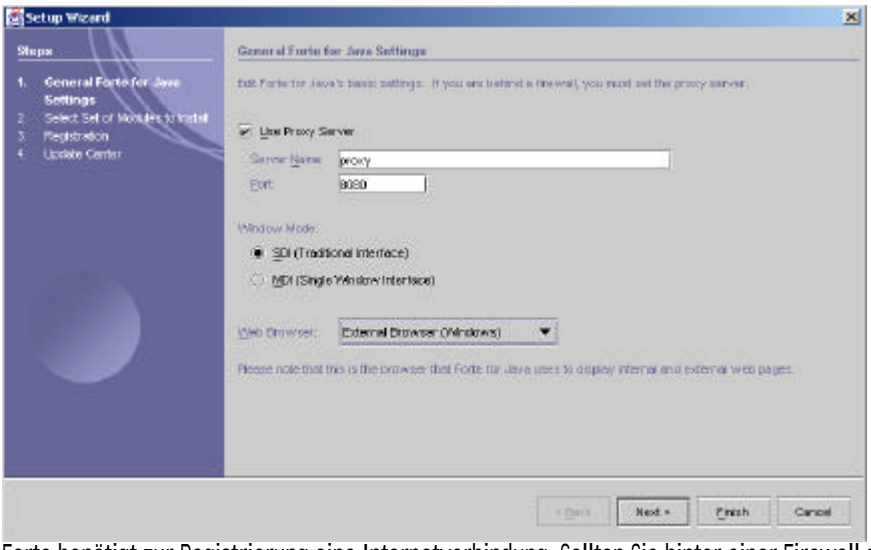



Forte frägt nach einem Pfad, wo die Dateien abgelegt werden sollen, die Sie selbst erstellen.
Es wird empfohlen, ein Verzeichnis wie z.B. c:\java zu wählen. Erläuterungen siehe Kap. 4.

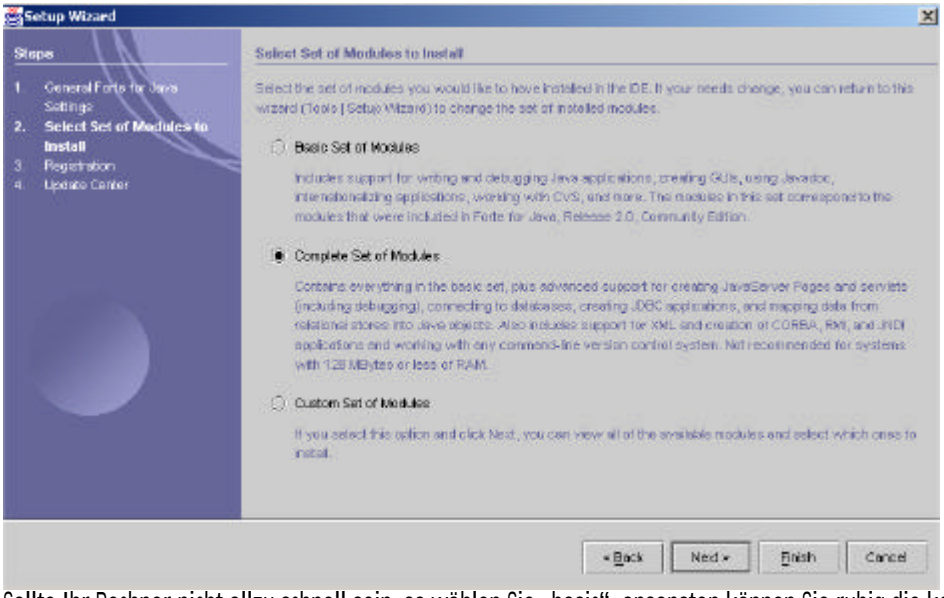



Sollten Sie kein Vorgängerprodukt installiert haben, so können Sie den ersten Schritt (Import settings) mittels Finish- Button beenden.


Da der iPlanet Webserver nicht von uns verwendet wird, können Sie diesen Schritt mit Cancel überspringen.




Forte benötigt zur Registrierung eine Internetverbindung. Sollten Sie hinter einer Firewall sitzen, nehmen Sie bitte die entsprechenden Einstellungen vor, sonst gehen Sie mittels Next> weiter.




Sollte Ihr Rechner nicht allzu schnell sein, so wählen Sie „basic“, ansonsten können Sie ruhig die kompletten Module von Forte installieren. Im nächsten Fenster beenden Sie mittels Finish die Installation.

Um einen schönen Desktop zu haben, empfiehlt es sich, das „Show welcome screen“-Häkchen abzuschalten.

# 4. Erklärungen

Legen Sie sich am besten in Ihrem Hauptverzeichnis einen Unterordner „Java“ oder „Entwicklung“ an, um die Forte Dateien zu speichern,.

Sollten Sie mit mehreren Projekten arbeiten, so legt Forte an sich automatisch für jedes Projekt ein Verzeichnis an.

## *4.1. Speicherstruktur von Java Anwendungen*

Wenn Sie Java Code compilieren (in für Maschinen ausführabren Code umwandeln), so erzeugen Sie aus den einzelnen Dateien mit der Endung .java solche mit der Endung .class. Eine solche Datei wird (kompilierte oder unkompilierte) meist (eigentlich fälschlicherweise) als Klasse bezeichnet. Wohl kann es aber sein, dass in einer Datei mehrere Klassen und auch innere Klassen vorkommen.

Forte speichert sowohl kompilierte als auch unkompilierte sowie auch Sicherungsdateien in einem Verzeichnis, wobei gilt: .java ist eine Datei mit Quelltext, .class ist eine kompilierte Datei und .~java ist die Dateiendung für Sicherungsdateien.

## *4.2. Packages*

Packages sind Gruppierungen von Klassen.
Das gesamte Java Framework ist in Klassen organisiert.
java.lang ist zum Beispiel so eine Klasse.
Die Packagenamen erlauben eine einfache Zuordnung der Klasse zu Bereichen. So ist zum Beispiel org für nicht kommerzielle, com für kommerzielle und java für Basisklassen gedacht.
Wollte man nun diese Package-Struktur auch in der TU Wien nach Spezifikation benützen, so müßten alle Packages in org.tuwien.apps.*programmname* angelegt werden.
Dies steht natürlich allen Programmierern frei, da in der Übung nicht kommerziell und nach Normen spezifiziert programmiert wird.

## Achtung, bitte beachten:

Was Sie in Forte aber beherzigen sollten:
Wenn eine Klasse einmal in einem Package angelegt wurde, ist es nicht sehr empfehlenswert, dieses Package wieder zu verändern.
Dies ist dadurch bedingt, dass Java beim Compilieren und Ausführen immer in der Verzeichnisstruktur nach Dateien in solchen Verzeichnissen sucht, die den Packages entsprechen.

Zum Beispiel würde eine Source-Klasse im Package org.tuwien.einprojekt auch in einer Datei in einem Verzeichnis src/org/tuwien/einprojekt/ gesucht werden.


Wenn Sie diese Klasse nun in ein anderes Package „umdefinieren“ ohne seine Position im Dateisystem zu verändern, dann wird Forte die Datei wohl anzeigen, aber der Java Development Kit wird beim kompilieren einen Fehler ausgeben.

Alle Dateien, die von Ihnen programmiert werden, sollten weiters nur Klassen beinhalten, die einem einzigen Package zugeordnet werden.
Ansonsten wird bei kleinen Projekten die Struktur meistens unübersichtlicher und sie müssen den einzelnen Klassen beibringen, dass andere Packages zu inkludieren sind.
Zum Beispiel muss eine Klasse im Package org.tuwien.einprojekt.hilfspackage mittels import zu einer Klasse im package org.tuwien.einprojekt hinzugefügt werden.

```
import org.tuwien.einprojekt.hilfspackage.*;
```

Ein solches Statement kann mit einem Stern (zum Import aller Klassen) oder mit einem Klassennamen abgeschlossen werden.
Bessere Programmierung ist es natürlich, wenn man die Klassennamen einzeln angibt (spart normalerweise Performance und schafft – bei entsprechender Kommentierung – Ordnung).

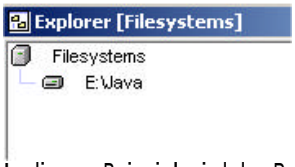
## 5. Erstellen eines neuen Projektes

Ein Projekt wird in Forte immer einem Filesystem zugeordnet.
Diese Filesystems werden gemounted bzw. unmounted.

Pro Filesystem (zugeordnetem Verzeichnis) empfiehlt es sich, ein Programm abzulegen.

Öffnen Sie Forte.

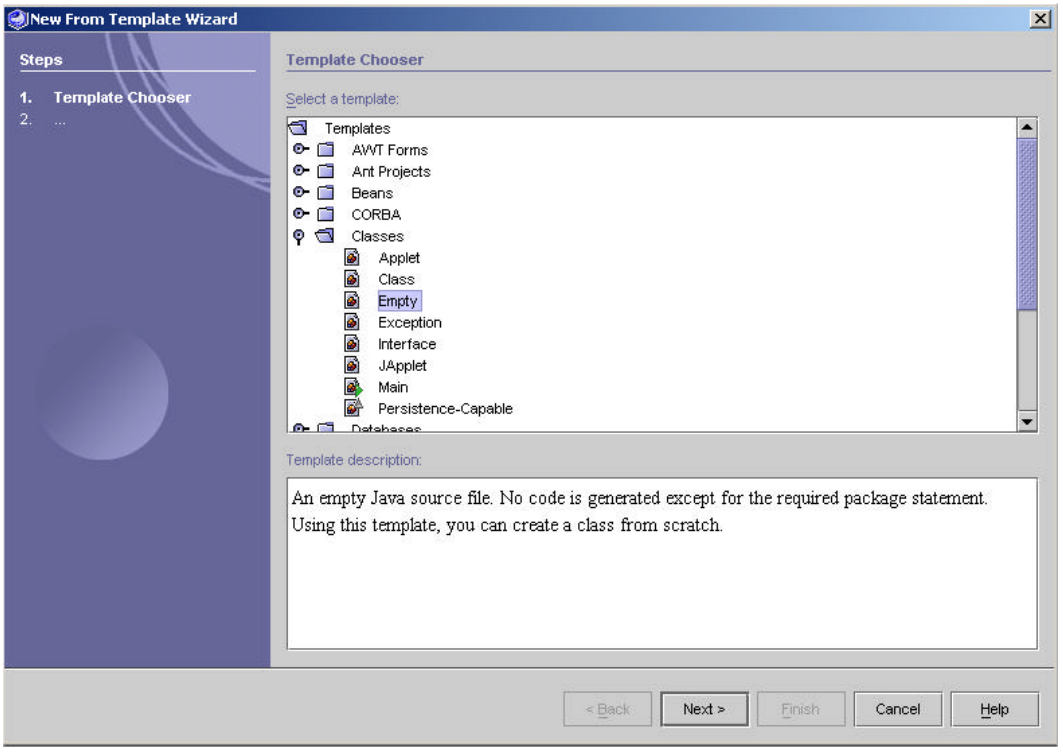
Überprüfen Sie, welches Filesystem derzeit gemounted ist.



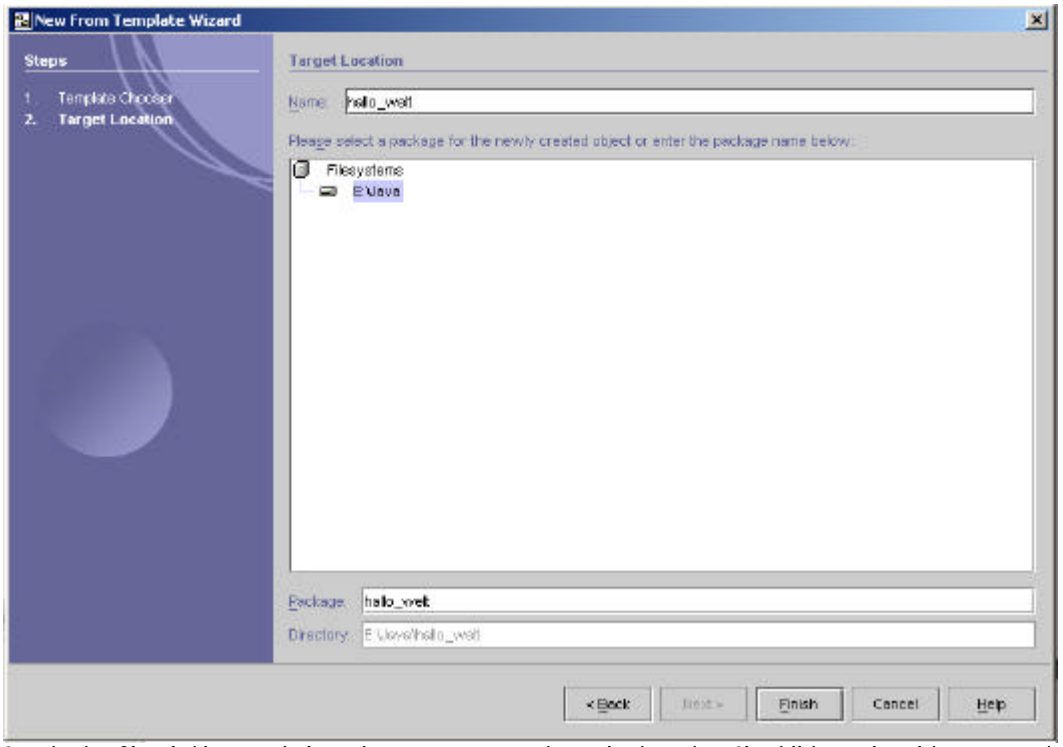

In diesem Beispiel wird das Projekt auf e:\java abgelegt.
Sollten Sie einen anderen Pfad wollen, so klicken Sie mit der rechten Maustaste auf Filesystems und wählen Sie „Mount Directory“

## 6. Eine erste Datei

Um eine erste Klasse zu erstellen (und damit zu programmieren zu beginnen) benötigen Sie eine Datei, in der die Inhalte abgelegt werden können.
Diese erstellen Sie am besten über den Menüpfad File -> New




Als Vorlage dient uns die leere – Empty –Klasse.




Standardmäßig wird in Forte kein Packagename vorgegeben. Dies brauchen Sie üblicherweise nicht zu verändern, da Package-Zuordnungen unsere Uiniprojekte nicht wirklich betreffen. Für Erläuterung siehe 4.2.

Der Class Name ist frei wählbar. Forte geht – wie JBuilder – davon aus, dass eine Klasse pro Datei gespeichert wird, überprüft dies aber später nicht.
Normalerweise nimmt man für die Hauptklasse (mit der das Programm gestartet wird) den Projektnamen. Benutzen Sie keine Sonderzeichen für den Namen, insbesondere keine Umlaute, da dies zu Problemen führen könnte.

Mittels Finish beenden Sie den Assistenten für die Dateierstellung.

Die Ausgabe sollte wie folgt aussehen.


## Erläuterungen:

Die Package-Zeile regelt die Zuordnung zum Package hallo_welt (siehe 4.2.).
Diese Zeile sollte nicht mehr verändert werden.

In Forte muss man die Hülle einer Klasse nun selbst erstellen.

Im Folgenden wird versucht, einige Zeilen Kommentare, eine Klassendefinition, eine Hauptmethode und einen Konstruktor zu erstellen.

Kommentarzeilen erscheinen am Bildschirm in grau.
Kommentare in Java werden mit **`/*`** eingeleitet und mit **`*/`** beendet. Alternativ kann man auch **`//`** am Anfang einer Zeile verwenden, wenn es sich z.B. nur um eine einzelne Zeile Kommentar handeln sollte.
Wir schreiben **`//Dies ist das Hallo Welt-Programm`** als Kommentar.

Mit **`public class hallo_welt {`** wird eine öffentliche Klasse eingeleitet. Die Klasse trägt in diesem Beispiel den Namen hallo_welt und wird mit mit einer geschwungenen Klammer am Ende der Klasse (klassischerweise die letzte Zeile im Programm) wieder geschlossen.
In einer solchen Klasse können Sie nun Funktionen und Methoden definieren. Funktionen haben einen Rückgabewert wohingegen Methoden „nur" ausgeführt werden, ohne einen Rückgabewert zu haben. Daher werden Methoden mit void (engl. Nichts, also kein Rückgabewert) bezeichnet.

Jede Klasse in Java hat einen (zumindest leeren) Konstruktor.
Dieser wird aufgerufen, sobald die Java Programmumgebung die Klasse aufruft (d.h. sobald die Klasse angesprochen wird, sei es durch direktes Aufrufen oder durch einen Import, Vererbung o.ä.). Ein Konstruktor wird meistens verwendet, um konstante Werte für die ganze Klasse festzulegen. In diesem Beispiel schreiben wir einen leeren Konstruktor. Dieser wäre z.B.
**`public hallo_welt() {}`**

Über die **`static void Main (String[] args) {`**, läßt sich eine Klasse starten (anders kann Sie nur vererbt oder importiert werden). Die Main geht wieder von einer öffnenden geschwungenen Klammer bis einer schließenden am Ende der Methode. Main-Methoden haben die Besonderheit, dass Sie Werte vom Starten der Anwendung (Argumente bzw. Befehlszeilenargumente) mitbekommen können. Daher haben Sie immer einen Array des Types String (siehe 7.1.) als Übergabewert.

Unser Beispiel sieht also folgendermaßen aus:

# 7. Das erste Programm

Im ersten Programm soll Text auf der Befehlszeile ausgegeben werden.
Weitere Beispiele entnehmen Sie Ihrer Java Literatur. Sie können auch Ihren Tutor um Probebeispiele ersuchen.

## *7.1. notwendige Befehle*

**`void System.out.println (String param)`** Diese, von Java zur Verfügung gestellte Methode, gibt den übergebenen Parameter vom Typ String auf der Befehlszeile aus (Anm. Die Befehlszeile wird in Forte bei der Ausführung auf einem eigenen Bildschirm ausgegeben(man wechselt über die Tabs unter der Menüleiste zwischen den Bildschirmsichten – ähnlich Linux KDE)) und springt in die nächste Zeile (line = println = print a line).

**`public class String`**: String wird von Java zur Verfügung gestellt und bezeichnet einen Datentyp, mit dem man Zeichenketten speichern kann.

## *7.2. Implementierung*

Source Editor [hallo_welt]

```
1  package hallo_welt;
2
3  //Dies ist das hallo_welt Programm
4
5  public class hallo_welt {
6
7    public hallo_welt() {
8    }
9    public static void main(String[] args) {
10
11     //Ausgabe eines Strings
12     System.out.println("Willkommen.");
13
14     //Zuweisen einiger Zeichen zu einem String _s
15     String _s = "-----------";
16
17     //Ausgabe dieses Strings
18     System.out.println(_s);
19   }
20 }
21
```

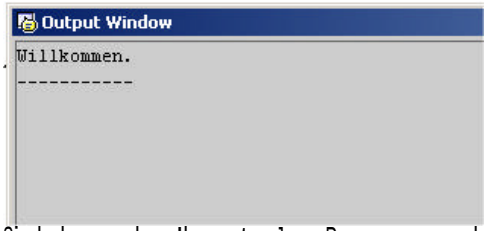
Zunächst wird mit **`System.out.println(„Willkommen.“);`** die Zeichenkette „Willkommen.“ auf der Befehlszeile ausgegeben. Die doppelten Anführungszeichen kennzeichnen eine zusammenhängende Zeichenkette und der Strichpunkt beschließt jeden Befehl. Auf diese Weise lassen sich z.B. mehrere Befehle in eine Zeile schreiben.

Mit **`String _s = „-----------“;`** wird die Zeichenkette „----------“ auf die Variable _s vom Typ String gelegt.

Mit dem nächsten Kommando wird diese Zeichenkette ausgegeben.

Mit F6 wird das Programm compiliert und ausgeführt.

Die Ausgabe am Ausführungsbildschirm sollte folgendermaßen aussehen:

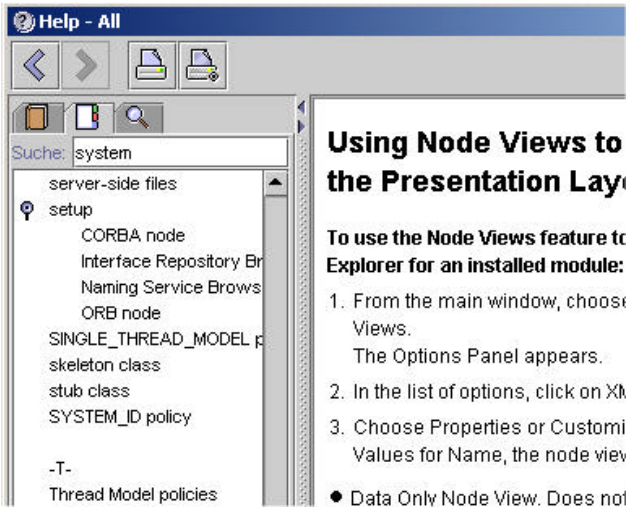



Sie haben soeben Ihr erstes Java Programm geschrieben.

# 8. Hilfe

## *8.1. Online Hilfe*

Niemand kennt alle Java Befehle auswendig. In der Hilfe (Menü Help) können Sie zu Forte Informationen erhalten.
Für die Dokumentation der Java Funktionen sei auf die Dokumentation von SUN unter http://java.sun.com verwiesen.




## *8.2. Web*

Sollte sich ein Befehl nicht in diesem Index finden, so können Sie auch die sehr ausführlichen Seiten von Sun Microsystems unter http://java.sun.com aufsuchen.

Gut weiterhelfen kann auch Yahoo:
http://de.dir.yahoo.com/Computer_und_Internet/Programmiersprachen/Java/

## *8.3. Bücher*

Schließlich vielleicht noch eine Orientierung bei den Büchern zu Java (auch zu finden auf der offiziellen Eprog-Seite des Institutes):

Java How to Program (für Fortgeschrittene), Harvey Deitel, Paul Deitel - Englisch

Lehrbuch der Programming mit Java (für Anfänger), Klaus Echtle, Michael Goedicke - Deutsch
(lagernd im Lehrmittelzentrum der TU Wien, Buchhandlung im Bibliotheksgebäude der TU Wien, Wiedner Hauptstraße 6, 1040 Wien, Telefon: +43 1 587 10 06/16)

Go To Java 2, Guido Krüger – Deutsch

Core Java 2, Bd.1 von Horstmann, Cornell - Deutsch

## 9. Debugger

Ein Debugger ermöglicht das Testen eines Programmes während dieses abläuft.

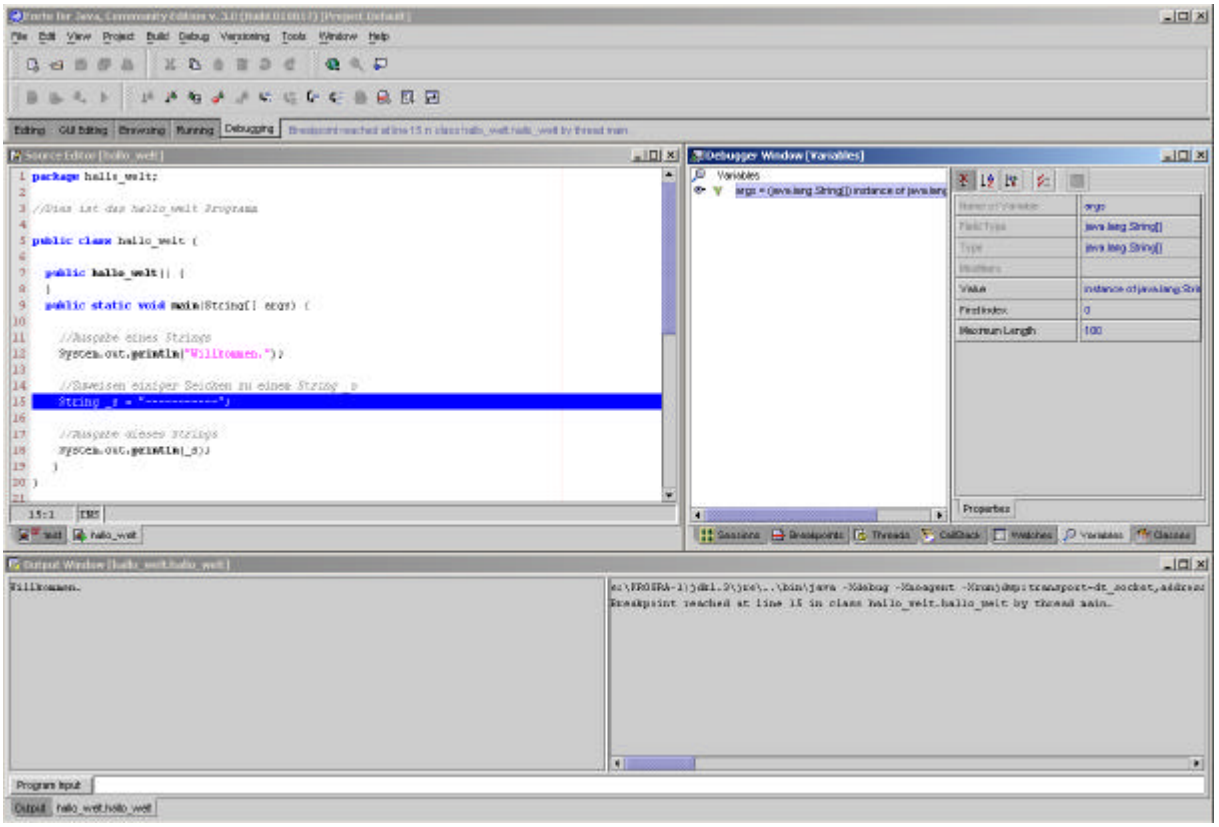
Man kann damit Haltepunkte setzen, bei denen das Programm stehen bleibt und genau sehen, wo ein möglicher Fehler auftritt.
Wir setzen in unserem Programm einen Haltepunkt auf die Zeile **`String _s =`**...
Dies geschieht, indem man in der gewünschten Zeile Strg+F8 drückt.

```
14      //Zuweisen einiger Zeichen zu e.
15      String _s = "-----------";
16
```

Die Zeile ist nun als Haltepunkt für den Debugger markiert.

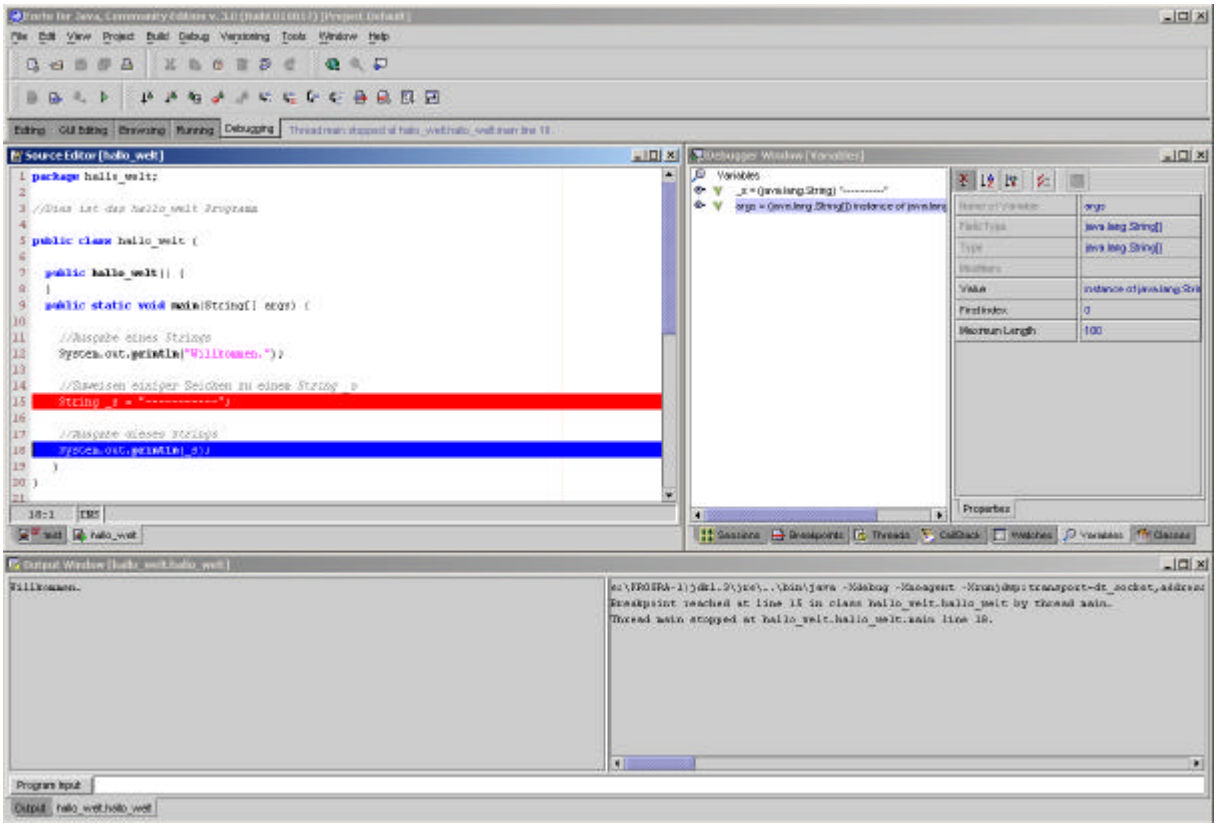
Um den Debugger zu starten, drücken Sie anstelle von F6 zum Ausführen die Taste F5.

Anstelle der gesamten Ausgabe bekommen Sie in der Ausgabeleiste nur das Wort Willkommen ausgegeben und eine etwas veränderte Ansicht.
Das Programm steht nun in der gewünschten Zeile auf Pause.



Nun kommen Sie mit F7 zum nächsten Befehl oder Sie können mit Strg+F5 fortsetzen.

Wenn Sie einmal F7 drücken, können Sie im Variable Stack sehen, dass die Variable s dazugekommen ist.



Diese Funktion ist sehr hilfreich, wenn Ihnen ein Programm aus einem nicht erklärbaren Grund mit einer Exception abstürzt.