Borland JBuilder 5 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 1.3, September 2001

1. Hinweise

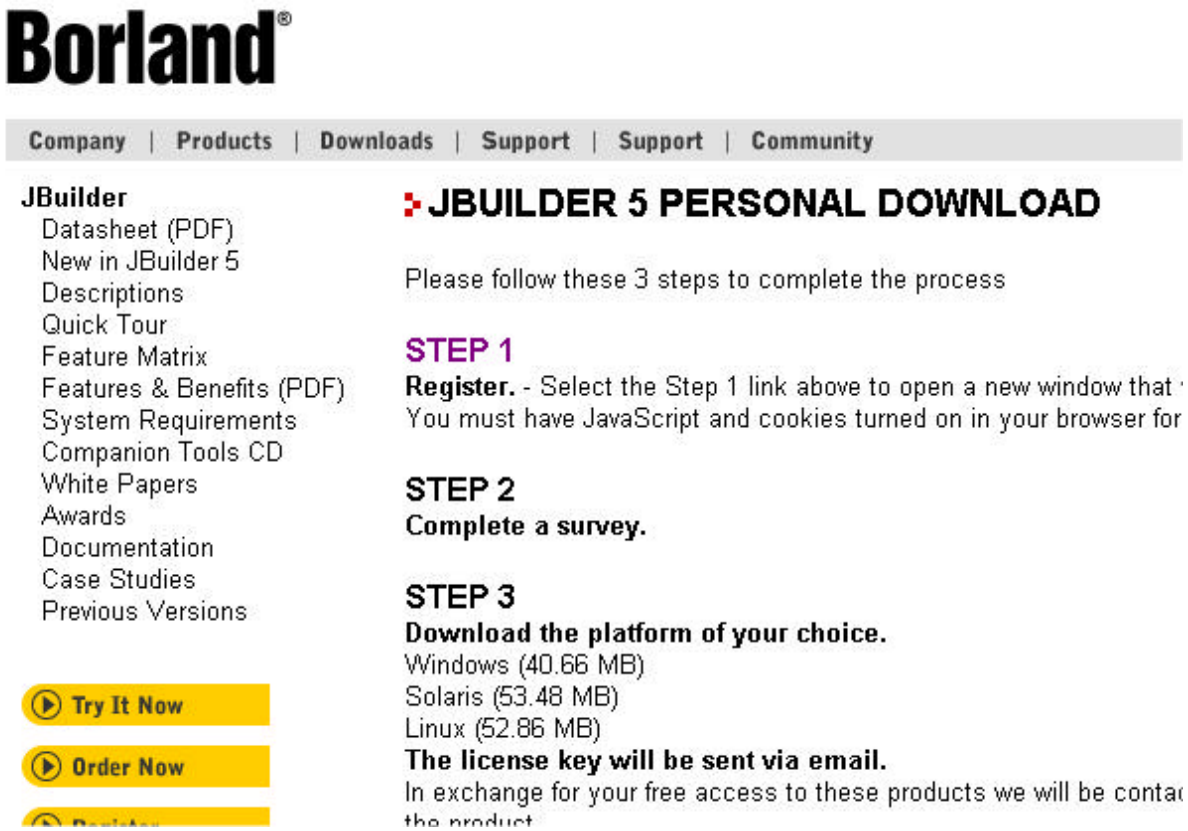
Borland und JBuilder sind eingetragene Warenzeichen der Borland Software Corporation und werden im Zusammenhang mit dieser Anleitung nicht jeweils explizit als solche gekennzeichnet.

2. Einleitung

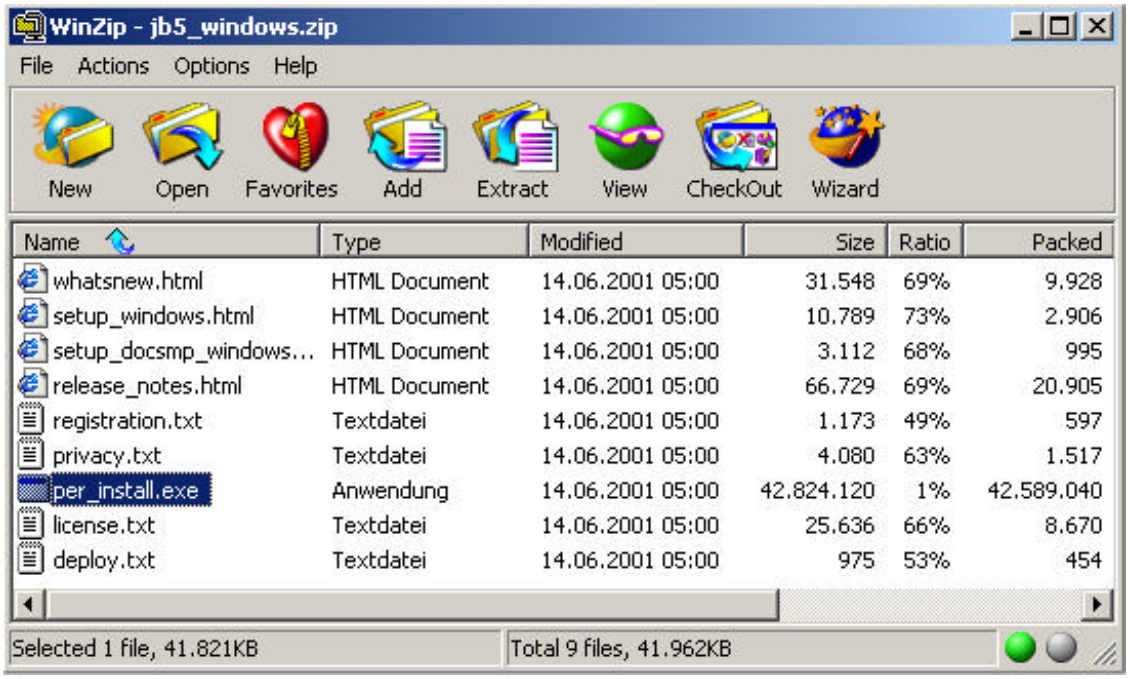
Dieses Skriptum wurde erstellt, um den Umgang mit JBuilder personal zu demonstrieren. Es ist für Einsteiger gemacht und soll anschaulich durch die einzelnen Schritte bis zu einem ersten Programm führen.
Es ist nicht nötig, gleichzeitig das Skriptum für Sun's Forte durchzusehen, weil diese beiden Skripten bis auf programmspezifische Dialogunterschiede gleich gehalten sind.
Die verwendete JBuilder Version ist 5.0.296.0. Versionsunterschiede können leider nicht berücksichtigt werden.
Feedback bitte an zeilinger@ildico.com oder anonym unter http://www.ildico.com/eprog.

3. Installation von JBuilder

Jbuilder kommt – gedownloaded von Borland (www.borland.com/jbuilder/personal) - verpackt als .zip Datei.

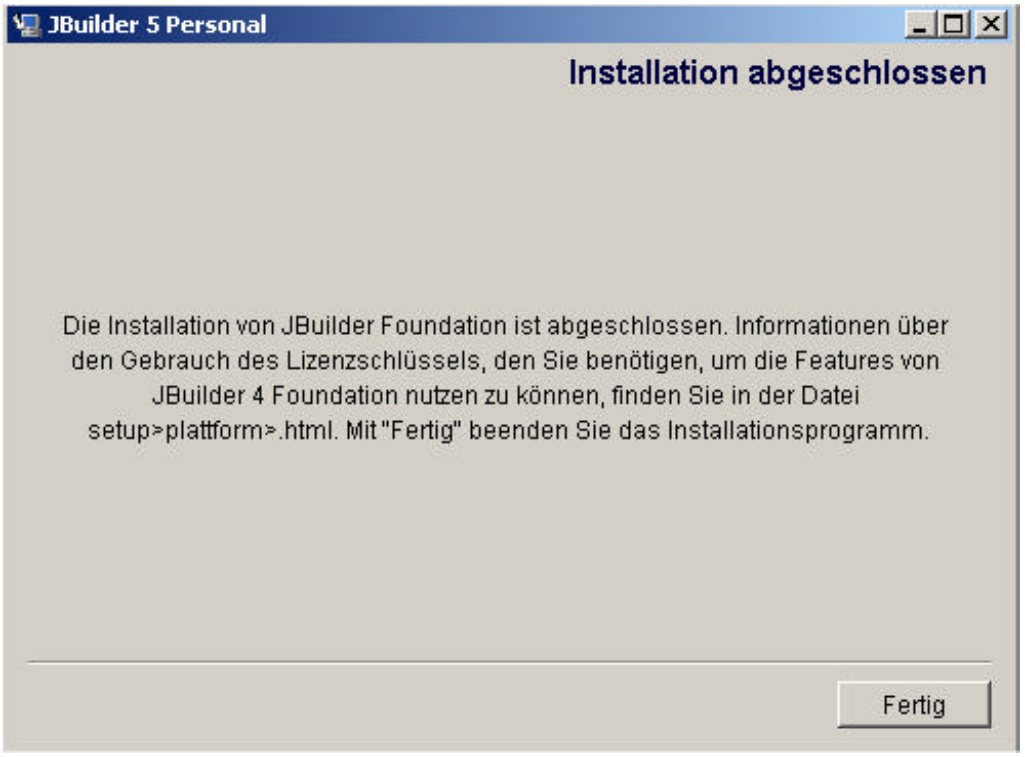


In dieser Datei befindet sich eine per_install.exe. Diese Datei entpacken und ausführen oder direkt aus dem Komprimierungsprogramm heraus aufrufen (Doppelklick).

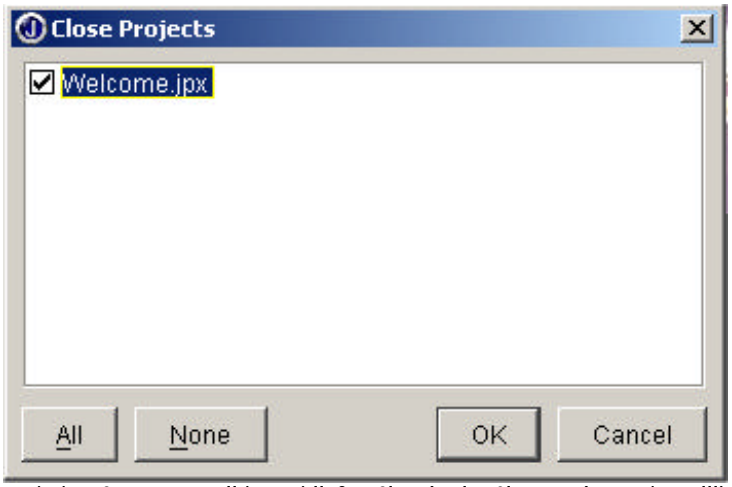


Nach der Sprachwahl wird nach der Einverständnis mit den Lizenzbedingungen gefragt. Im Wesentlichen besagen diese, dass man das Produkt nicht für kommerzielle Software verwenden darf. Dies trifft aber auf „educational use“ ohnehin nicht zu.

Es empfiehlt sich, JBuilder in einen Ordner zu installieren, der im Standardverzeichnis für Programme liegt (klassischerweise unter 95/98/ME/2000: C:\Programme).



War die Installation erfolgreich, dann lässt sich JBuilder starten.
Den Registrierungsschlüssel bekommen Sie per Mail zugeschickt, wenn Sie JBuilder personal korrekt bei Borland heruntergeladen haben.



Nach dem Start von JBuilder, schließen Sie mittels „Close Projects“ das Willkommens-Projekt.
Sie haben nun JBuilder erfolgreich installiert.

4. Erklärungen

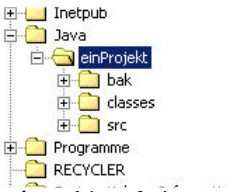
JBuilder legt normalerweise die Projekte unter neueren Versionen von Windows in einem Verzeichnis ähnlich c:\Dokumente und Einstellungen\username\JBProject\projektname an.
Dieses Verzeichnis empfehlen wir Ihnen zu meiden.
Legen Sie sich am besten in Ihrem Hauptverzeichnis einen Unterordner „Java“ oder „Entwicklung“ an.

Sollten Sie mit mehreren Projekten arbeiten, so legt JBuilder 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.

JBuilder benutzt in jedem Projekt folgende Verzeichnisse



bak speichert frühere Versionen der Codedateien
src speichert die aktuelle Version des Codes
classes speichert den Maschinencode
dok für allfällige Dokumentation

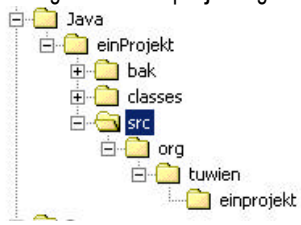
4.2. Packages

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 im JBuilder 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 JBuilder die Datei wohl anzeigen, aber der Java Development Kit, der in JBuilder integriert ist, wird einen Fehler ausgeben.

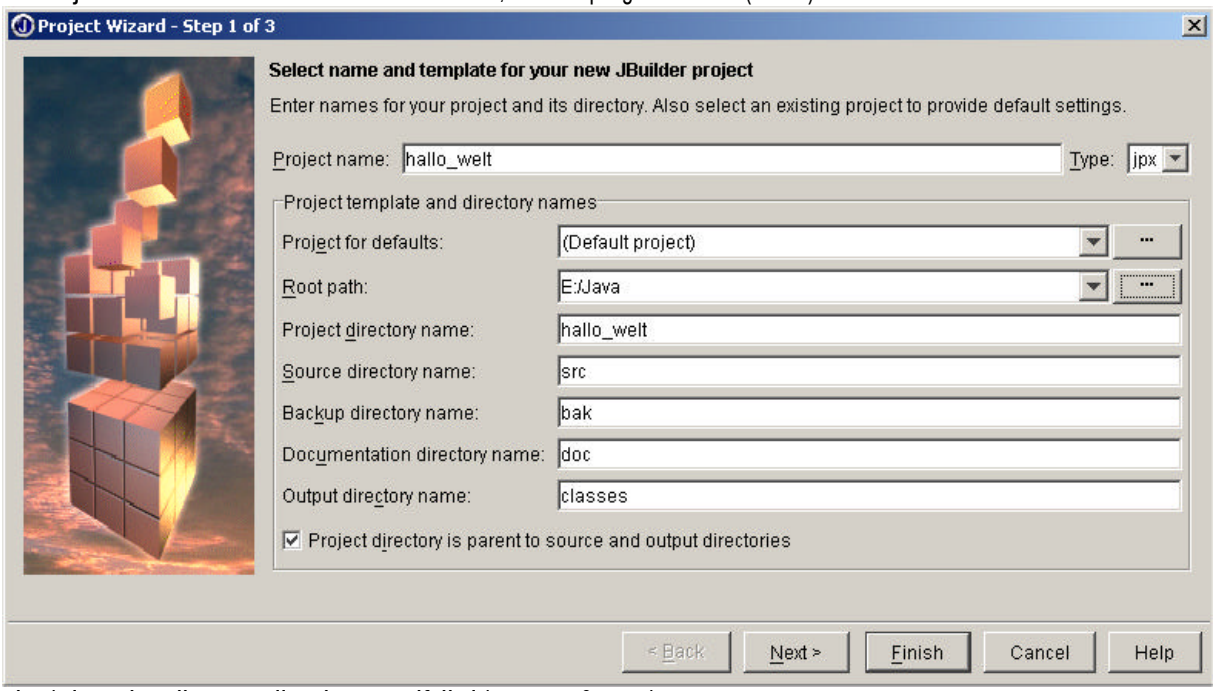
Alle Dateien, die von Ihnen programmiert werden, sollten weiters nur Klassen beinhalten, die einem einzigen Package zugeordnet werden.
Ansonsten wird bei kleinen Projekten die Struktur meistens unübersichtlicher und sie müssen den einzelnen Klassen beibringen, dass andere Packages zu inkludieren sind.
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

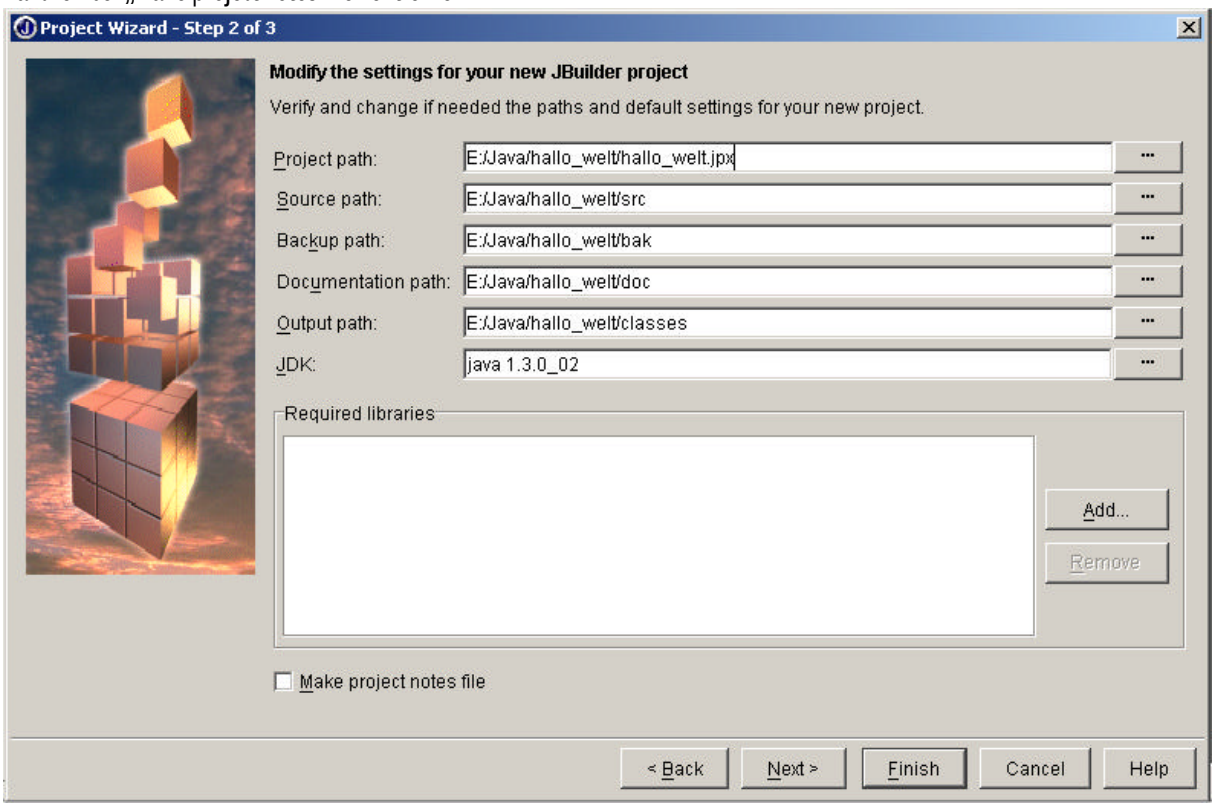
Öffnen Sie ihren JBuilder.
Mit Menüpfad File -> New Project... gelangen Sie zur ersten Maske des Project Wizard, wo Sie Projektnamen und Root Path vergeben. Wie bereits erläutert empfiehlt es sich, einen eigenen root Pfad zu definieren.
Der Projektname sollte etwas mit dem zu tun haben, was Sie programmieren (sollen).



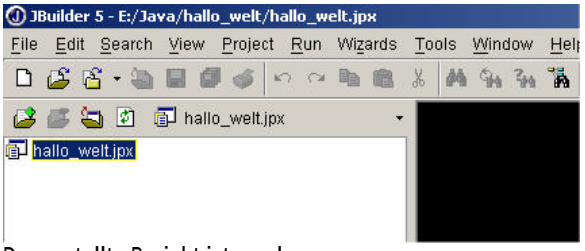
Die übrigen Einstellungen sollten im Normalfall nicht angepaßt werden müssen.

Mittels Next> gelangen Sie auf den nächsten Bildschirm.

Dort können Sie – falls Sie dies wünschen – eventuell eine andere Java Version verwenden. Momentan (September 2001) wird JavaDevelopmentKit 1.3.0_02 mit JBuilder ausgeliefert.
Wenn Sie nicht wünschen, dass JBuilder automatisch eine Dokumentation erstellt (eine leere HTML Seite), dann können Sie das Häkchen bei „Make project notes file“ entfernen.



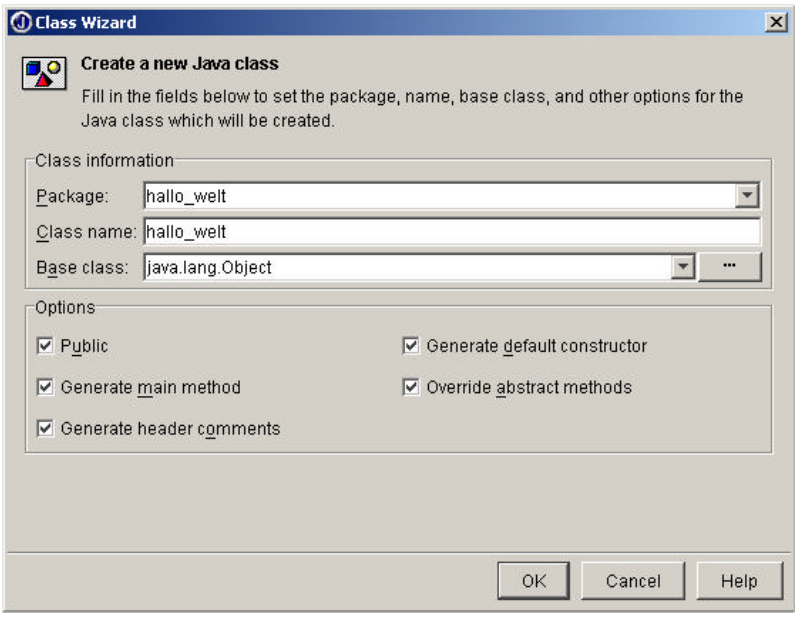
Wählen Sie Finish, um den Vorgang abzuschließen, oder Next>, wenn Sie die Dokumentation gleich beginnen möchten.



Das erstellte Projekt ist nun leer.
Die nächsten Schritte sind das Erstellen einer ersten Datei und die Programmierung einer Klasse.

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 Class...

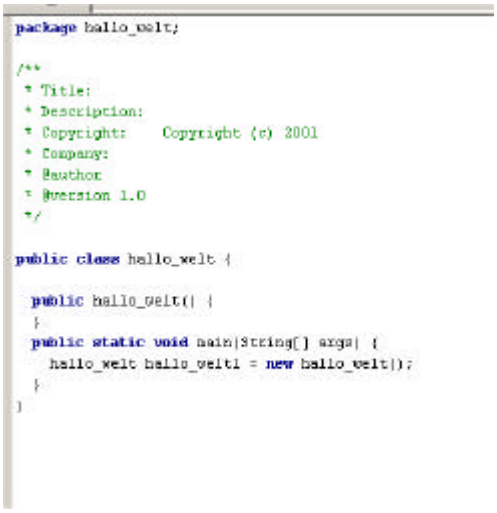


Standardmäßig wird der Projektname als Packagename vorgegeben. Dies brauchen Sie üblicherweise nicht zu verändern. Für Erläuterung siehe 4.2.

Der Class Name ist frei wählbar. JBuilder geht in diesem Fall 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.

Wenn Sie sich die Main-Methode bereits vorgenerieren lassen möchten (um die komplette Hülle eines ausführbaren Programmes schon zu haben), können Sie das entsprechende Kästchen anwählen.

Die Ausgabe sollte wie folgt aussehen.



```
package hallo_welt;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:
 * @author
 * @version 1.0
 */

public class hallo_welt {

  public hallo_welt() {
  }
  public static void main(String[] args) {
    hallo_welt hallo_welt1 = new hallo_welt();
  }
}
```

Erläuterungen:

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

Die nächsten Zeilen (in grün) sind Kommentarzeilen.
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.

Mit **`public class hallo_welt {`** wird eine öffentliche Klasse eingeleitet. Die Klasse trägt in diesem Beispiel den Namen hallo_welt und wird mit der letzten geschwungenen Klammer auf dem Bildausschnitt 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 ist der Konstruktor leer, weil er von JBuilder vorerst nicht befüllt wird.

Über die **`void Main`**, die ebenfalls generiert wurde, läßt sich eine Klasse starten (anders kann Sie nur vererbt oder importiert werden). Die Main geht wieder von der öffnenden geschwungenen Klammer bis zur schließenden zwei Zeilen darunter. 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.

Nun können Sie in der Main Methode beginnen zu programmieren. Die erste Zeile Code – von JBuilder erstellt – weist einer Variable namens **`hallo_welt1`** den Objekttyp **`hallo_welt`** zu und befüllt diese Variable initial mit einer neuen Instanz des Programmes **`hallo_welt`**. Diese Zeile brauchen wir für den Anfang nicht, da wir noch keine Objektvariablen einsetzen müssen. Am besten, Sie löschen diese Zeile weg und speichern das Projekt.

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 im JBuilder bei der Ausführung auf einem Teil des Bildschirmes ausgegeben) und springt in die nächste Zeile (**l**ine = print**ln** = print a **l**ine).

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

7.2. Implementierung

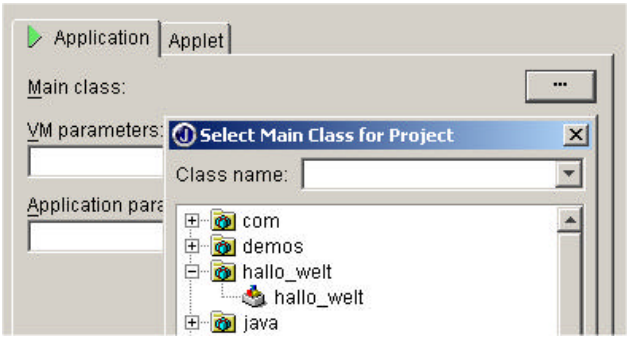
```
public static void main(String[] args) {

  //Ausgabe eines Strings
  System.out.println("Willkommen.");

  //Zuweisen einiger Zeichen zu einem String _s
  String _s = "-----------";

  //Ausgabe dieses Strings
  System.out.println(_s);

 }
}
```

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.

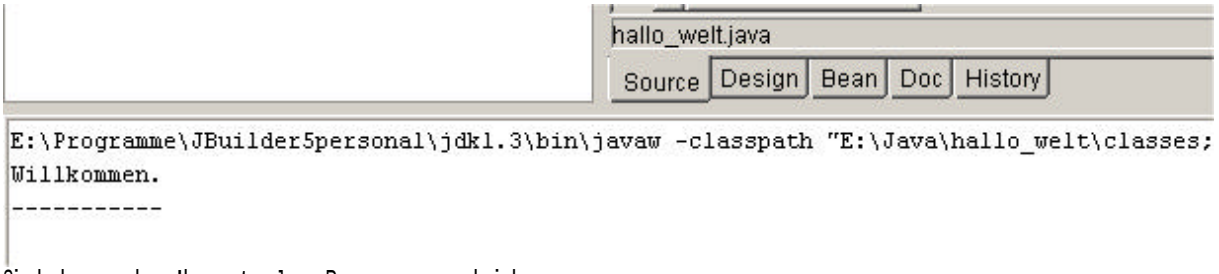
Mit **`String _s = „-----------“;`** wird die Zeichenkette „----------“ auf die Variable _s vom Typ String gelegt.

Mit dem nächsten Kommando wird diese Zeichenkette ausgegeben.

Mit F9 wird das Programm compiliert und ausgeführt.
Eventuell müssen Sie die Startklasse angeben. Diese ist der Packagename gefolgt vom Klassennamen der ausführbaren Klasse (jener mit Main-Methode). Im Falle unseres Beispieles ist dies **`hallo_welt.hallo_welt`**.



Die Ausgabe im unteren Bildschirmbereich sollte folgendermaßen aussehen:

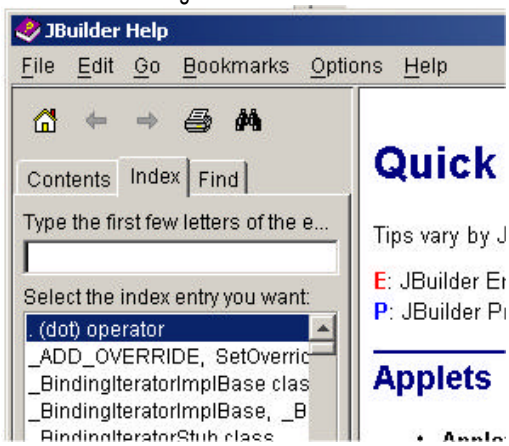


Sie haben soeben Ihr erstes Java Programm geschrieben.

8. Hilfe

8.1. Online Hilfe

Niemand kennt alle Java Befehle auswendig.
In der Hilfe (Menü Help) können Sie zu allen Befehlen Informationen erhalten.
Es sind in einem eigenen Punkt unter Contents auch die kompletten Spezifikationen von Sun enthalten.



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)

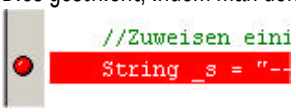
Go To Java 2, Guido Krüger – Deutsch

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

9. Debugger

Ein Debugger ermöglicht das Testen eines Programmes während dieses abläuft.

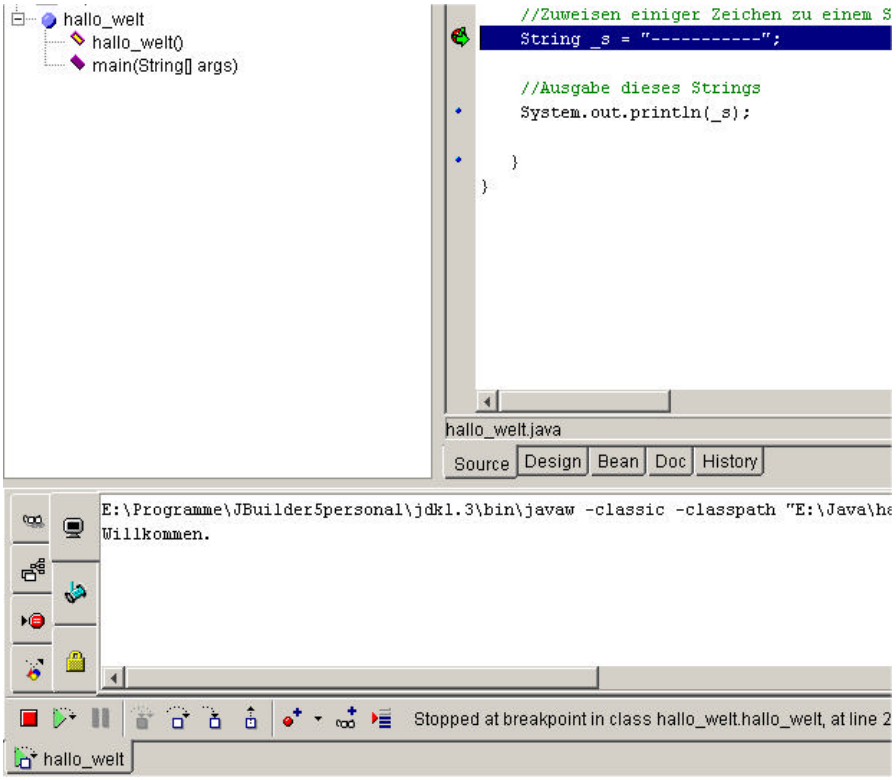
Man kann damit Haltepunkte setzen, bei denen das Programm stehen bleibt und genau sehen, wo ein möglicher Fehler auftritt.
Wir setzen in unserem Programm einen Haltepunkt auf die Zeile **`String _s =`**...
Dies geschieht, indem man den linken Rand neben einer Zeile anklickt.



Die Zeile ist nun als Haltepunkt für den Debugger markiert.

Um den Debugger zu starten, drücken Sie anstelle von F9 zum Ausführen die Tastenkombination Shift + F9.

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



Nun kommen Sie mit F8 zum nächsten Befehl oder Sie können mit F9 fortsetzen.
Diese Funktion ist sehr hilfreich, wenn Ihnen ein Programm aus einem nicht erklärbaren Grund mit einer Exception abstürzt.