Leibniz Universität IT Services

Leibniz Universität Hannover

Java - Einführung in die Programmiersprache

Handbücher am IT Services

- Programmierung.
- Java: Grundlagen und Einführung
- Java: Fortgeschrittene Techniken und APIs
- Java und XML

Bücher zur Programmiersprache Java

- Hans-Peter Habelitz: Programmieren lernen mit Java
- Kai Günster: Einführung in Java: Ideal für Studium und Ausbildung
- Dirk Louis & Peter Müller: Java. Der umfassende Programmierkurs
- Michael Kofler: Java. Der Grundkurs
- Dirk Hardy: Java für IT-Berufe.
- Goll / Heinisch: Java als erste Programmiersprache

Bücher zu Java und NetBeans

- Joel Murach, Michael Urban: Murach's beginning Java with NetBeans
- Geertjan Wielenga: Beginning NetBeans IDE
- David Salter: Mastering NetBeans

Tutorials im Web

- http://docs.oracle.com/javase/tutorial/
- http://www.gailer-net.de/tutorials/java/
- http://www4.fh-swf.de/media/java.pdf
- https://www.youtube.com/channel/UCB6-E_RGh9necQ5YOclkpag/playlists
- https://www.uni-trier.de/fileadmin/urt/doku/java/v70/Java7.pdf
- http://www.java-tutorial.org/
- http://www.vogella.com/java.html
- https://beginnersbook.com/2013/05/java-introduction/

Code-Beispiele im Web

- http://www.javabeginners.de/
- http://www.programmieraufgaben.ch/
- http://www2.inf.fh-bonn-rhein-sieg.de/~sweil2m/Tutorium/Tutorium1.pdf

Online-Kurse

- http://codingbat.com/java. Der Code wird zum Testen direkt in den Browser eingegeben.
- https://open.hpi.de/?locale=de bietet Online-Kurse zu bestimmten Zeiten an.
- https://www.learnjavaonline.org/

Java-Technologie

- Programmiersprache Java.
- Java-Plattform bestehend aus einer Laufzeitumgebung (Java Runtime Environment) und den Standard-Klassenbibliotheken.

Programmiersprache „Java“

- Anlehnung an die Syntax der Programmiersprache C++.
- Objektorientierte Programmiersprache. Daten und die dazugehörigen Aktionen werden in ein Objekt zusammengefasst.
- Speichervergabe und -freigabe erfolgt über das Laufzeitsystem.
- Implementierung einer Fehlerbehandlung (Expection-Handling).
- Plattformunabhängig. Der Quellcode wird in einen Zwischencode übersetzt.

Anwendungsgebiete

- Client-Server-Programmierung.
- Java-Applet.
- Entwicklung von Apps für Android.

Java-Plattform

- Die Programmiersprache Java. Java 9 ist am 21.09.2017 erschienen.
- Werkzeuge wie der Java-Compiler.
- Java-Virtuelle Maschine.
- Klassenbibliothek.

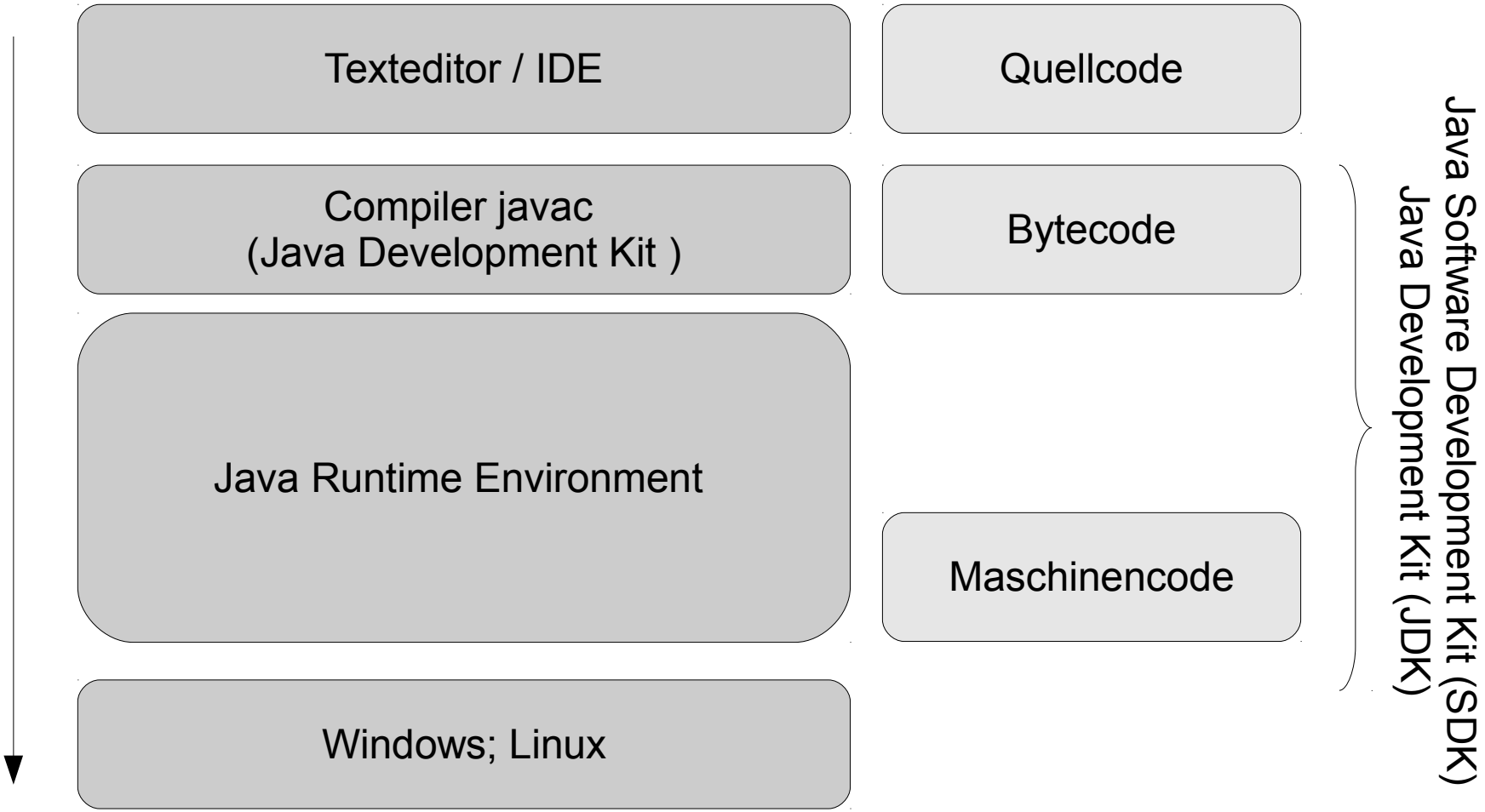
… werden gebündelt in der

- Standard Edition (Java SE). Desktopanwendungen. Lesen und Schreiben von Dateien und vieles mehr. Anwendung in diesem Kurs.
- Enterprise Edition (Java EE). Erweiterte Java SE. Mehrbenutzerumgebungen. Arbeiten mit relationalen Datenbanken. Webfrontend.
- Micro Edition (Java ME). Schnittstelle für Mobiltelefone. In der Version 1.3 eingefroren.
- Java Card für Smartcards mit einem eingeschränkten Sprachumfang.
- Siehe http://www.oracle.com/technetwork/java/api-141528.html.

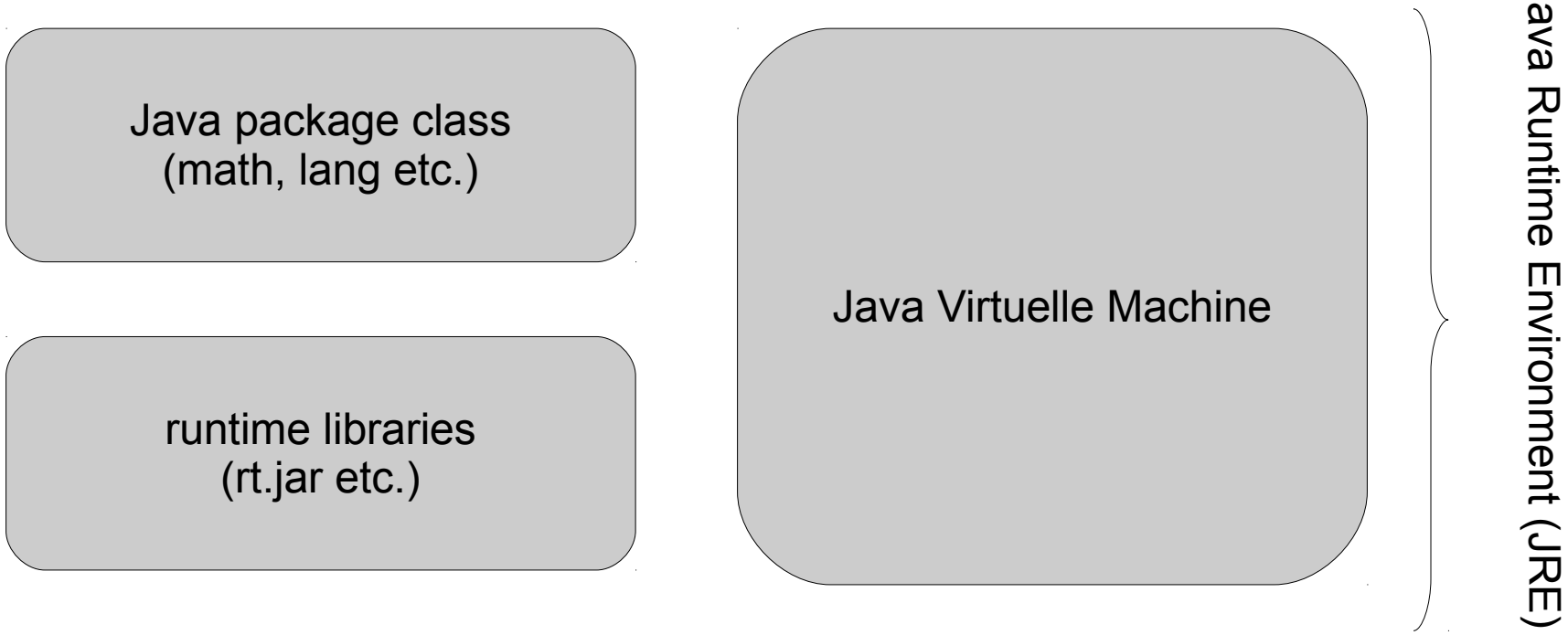
Java Development Kit (JDK)

- Java Software Development Kit (SDK)
- Enthält eine Java-Plattform.
- Wird für die Entwicklung von Java-Programmen benötigt.
- Download unter http://www.oracle.com/technetwork/java/javase/downloads/index.html.
- Alternative: https://developers.redhat.com/products/openjdk/overview/

Architektur



Java Runtime Environment (JRE)



Entwicklungsumgebungen (IDE)

- Grafische Oberfläche zur Entwicklung eines Programms.
- Arbeitsumgebung zur Programmierung.
- Bündlung von einer Laufzeitumgebung und eines Texteditors für eine oder mehrere Programmiersprachen.
- Prüfung des geschriebenen Codes auf Syntaxfehler.

Liste von IDEs im Netz

- https://blog.idrsolutions.com/2015/03/the-top-11-free-ide-for-java-coding-development-programming/
- http://www.javaworld.com/article/3114167/development-tools/choosing-your-java-ide.html

BlueJ

- Entwicklung für den Anfänger in der Programmierung.
- Start an der Monash University in Melbourne, Australien.
- Freie Software.
- Download unter http://www.bluej.org/.

IntelliJ IDEA

- Entwicklung des tschechischen Softwarehauses JetBrains.
- Die „Community Edition“ ist frei.
- Als weitere Programmiersprachen werden PHP und JavaScript unterstützt.
- Zusätzlich zu den Java-APIs ist die Android API implementiert.
- Download unter http://www.jetbrains.com/idea/.

Eclipse

- OpenSource.
- 2001 Gründung des Eclipse Projects durch IBM. 2004 Gründung der Eclipse Foundation zur Weiterentwicklung von Eclipse.
- Erweiterbarkeit durch PlugIns.
- Als weitere Programmiersprachen werden C /C++, PHP und JavaScript unterstützt.
- Zusätzlich zu den Java-APIs ist die Android API implementiert.
- Download unter http://www.eclipse.org/.

Oracle JDeveloper

- Seit 2005 frei verfügbar.
- Einbindung in die Software-Palette von Oracle.
- Sammlung von Oracle-Werkzeugen in einer IDE.
- Erweiterbarkeit durch PlugIns.
- Als weitere Programmiersprachen wird JavaScript und PHP unterstützt.
- Download unter http://www.oracle.com/technetwork/developer-tools/jdev/overview/index-094652.html.

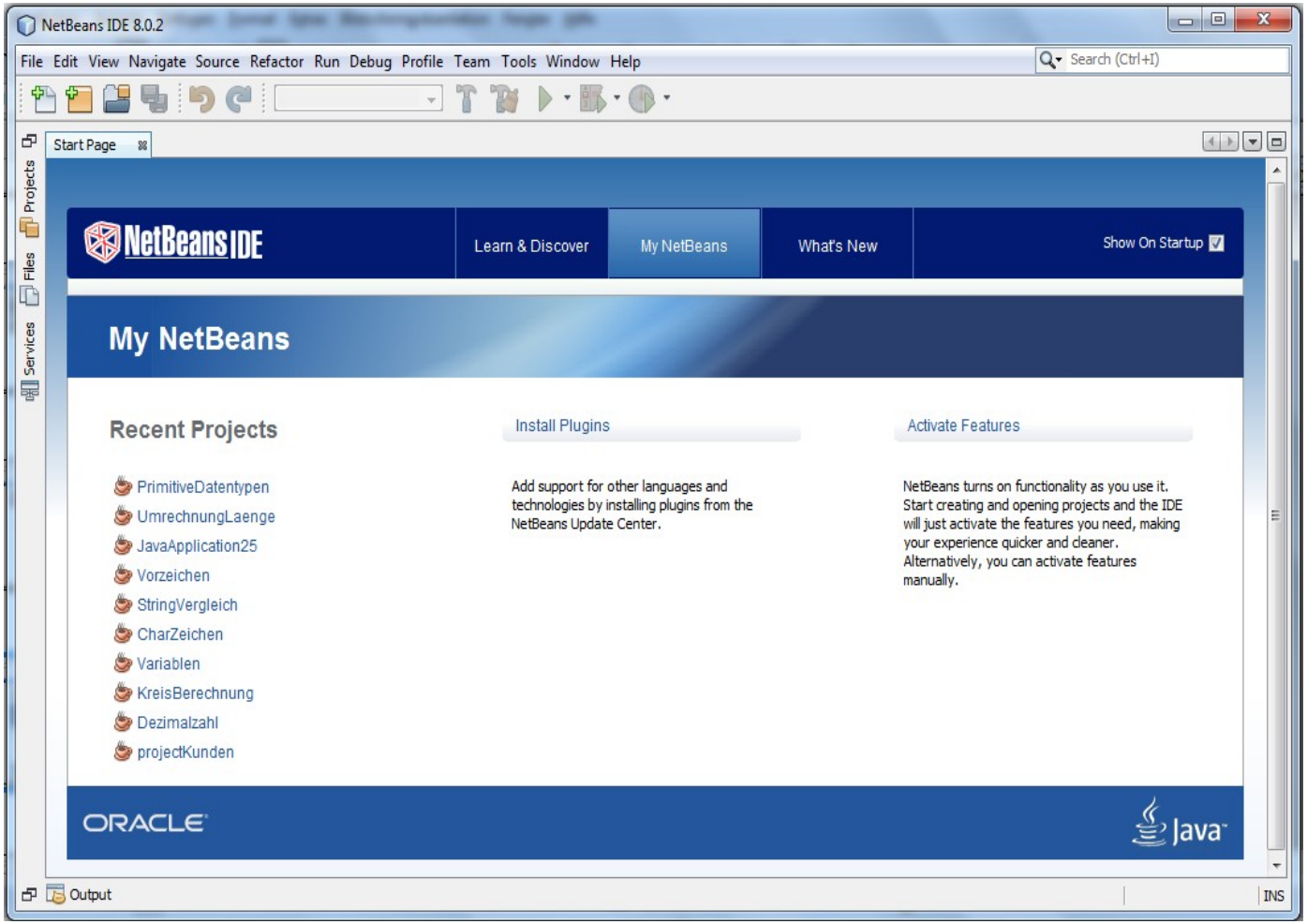
NetBeans ...

- 1996 entwickelt von Studenten aus der Tschechischen Republik. 1999 Übernahme durch Sun Microsystems. 2010 Übernahme durch Oracle.
- Als weitere Programmiersprachen werden C /C++, PHP und JavaScript unterstützt..
- Download unter https://netbeans.org/.

Start von NetBeans

- Icon auf dem Desktop.
- Windows 8 und höher: *Suchen …*
- Im Installationspfad von NetBeans: *bin\netbeans64.exe.*

Startseite

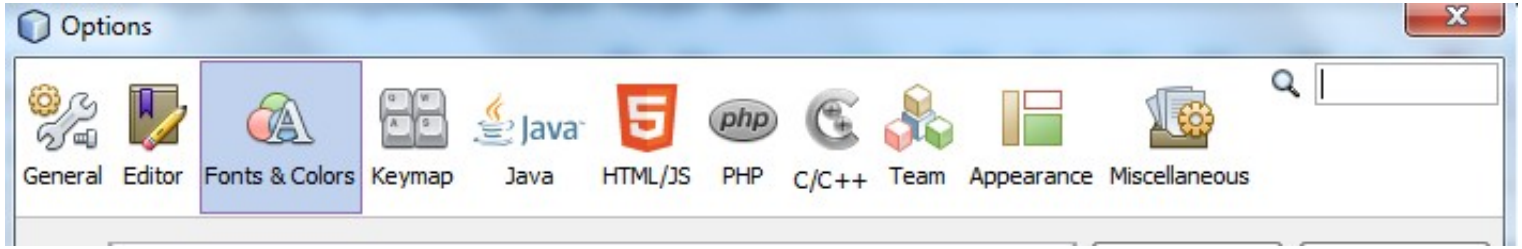

NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Start Page
Projects
Files
Services
NetBeans IDE
Learn & Discover
My NetBeans
What's New
Show On Startup
My NetBeans
Recent Projects
PrimitiveDatentypen
UmrechnungLaenge
JavaApplication25
Vorzeichen
StringVergleich
CharZeichen
Variablen
KreisBerechnung
Dezimalzahl
projectKunden
Install Plugins
Add support for other languages and technologies by installing plugins from the NetBeans Update Center.
Activate Features
NetBeans turns on functionality as you use it. Start creating and opening projects and the IDE will just activate the features you need, making your experience quicker and cleaner. Alternatively, you can activate features manually.
ORACLE
Java
Output
INS

Hinweis

- Aktivierung des Kontrollkästchens *Show on startup*. Die Startseite wird beim Starten des Editors automatisch angezeigt.
- Deaktivierung des Kontrollkästchens *Show on startup*. Die Startseite wird beim Starten des Editors nicht mehr angezeigt.

IDE anpassen

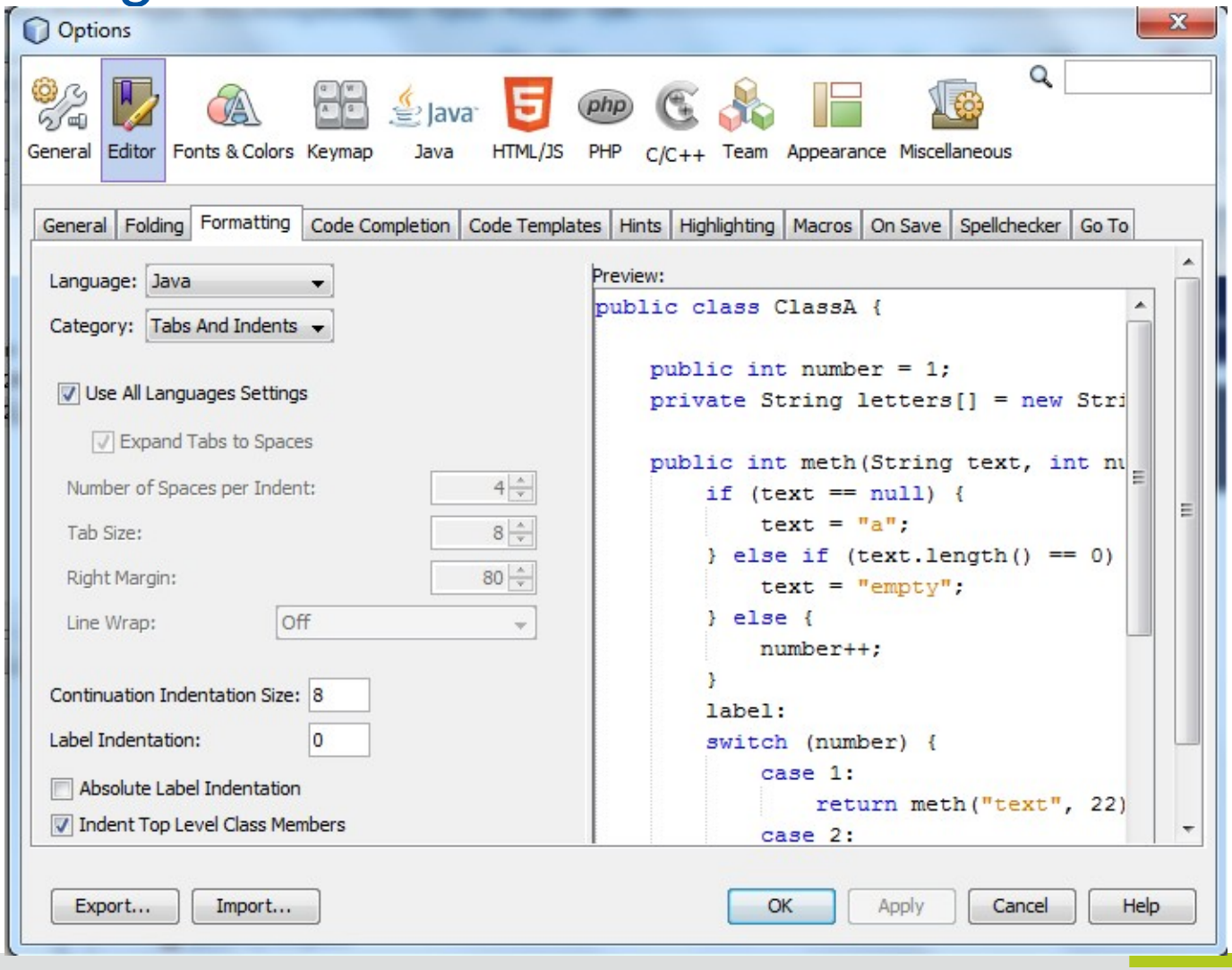
- *Tools – Options*.



Möglichkeiten

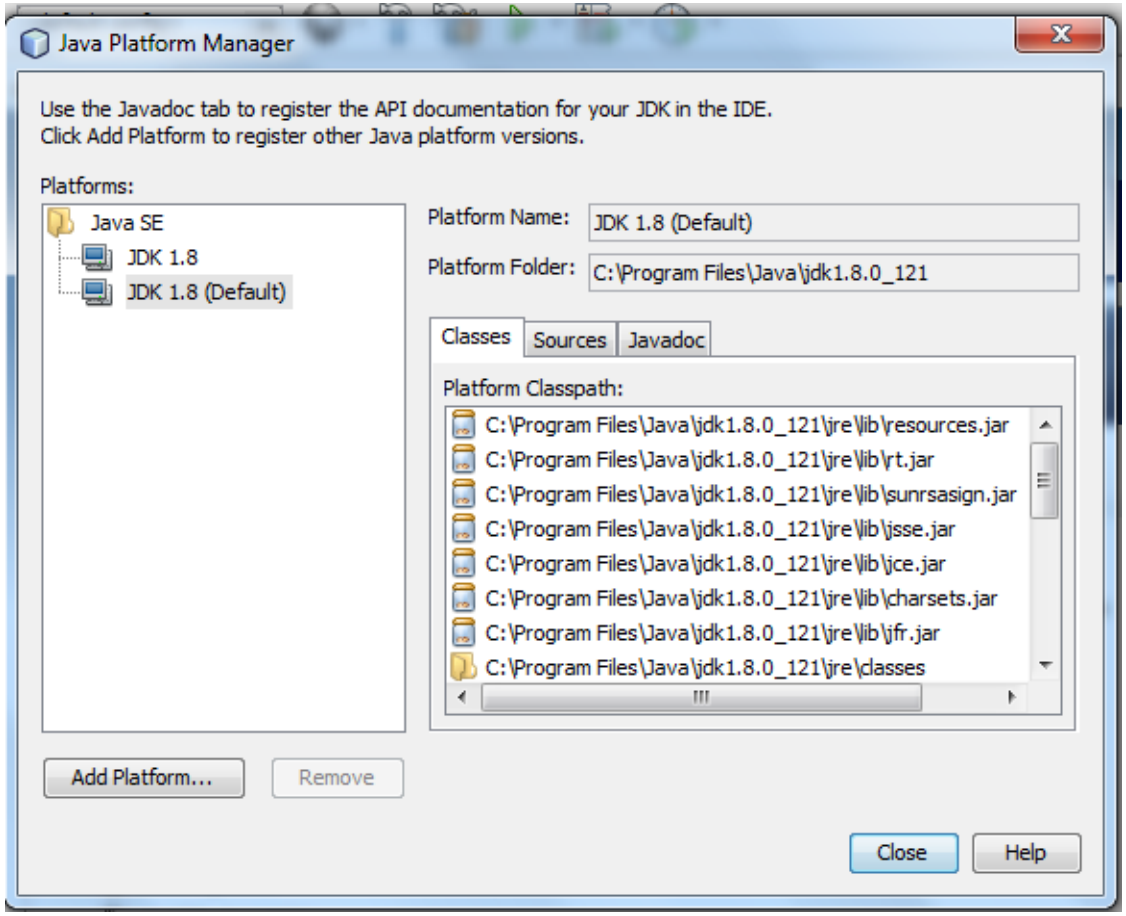
- Die Symbole am oberen Rand stellen die verschiedenen Kategorien dar.
- *General*. Einstellungen für die Programmierung im Web.
- *Editor*. Automatische Ergänzung von Schlüsselwörtern etc. Einstellung von Einrückungen im Codefenster.
- *Fonts & Color*. Welche Schriftart und -farbe wird im Codefenster genutzt?
- *Keymap*. Welche Shortcuts werden genutzt?
- *Java*. Einstellungen zum Debugger etc.

Einstellungen für den Editor



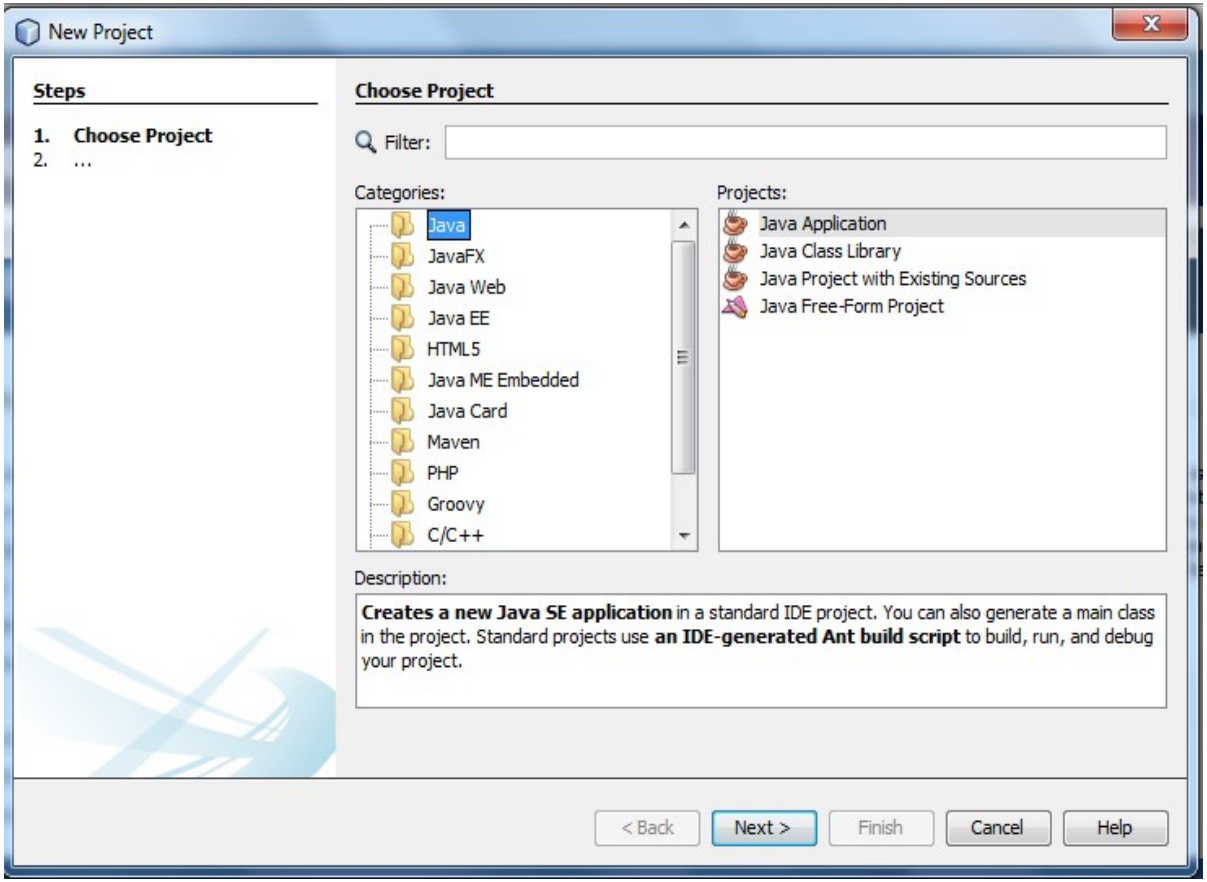
Plattform-Manager in NetBeans

- *Tools – Java Platforms*.



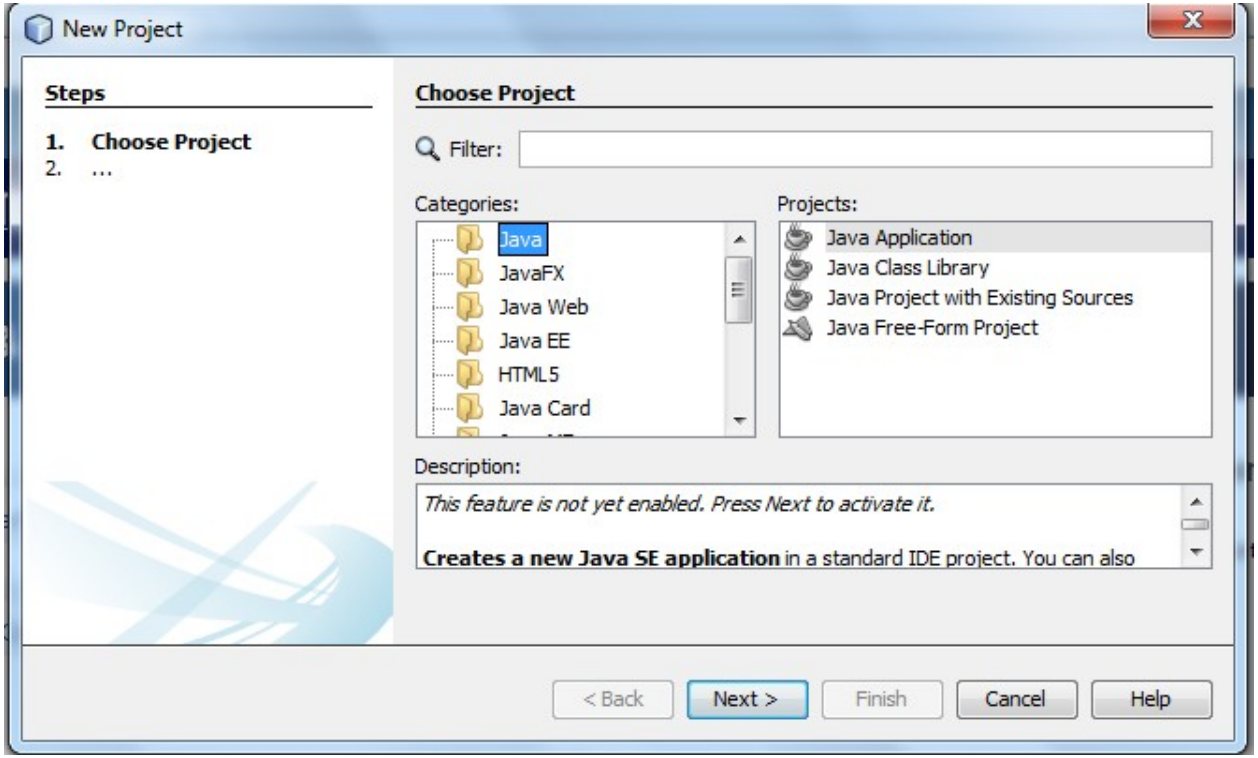
Projekt anlegen

- *File – New Project*.

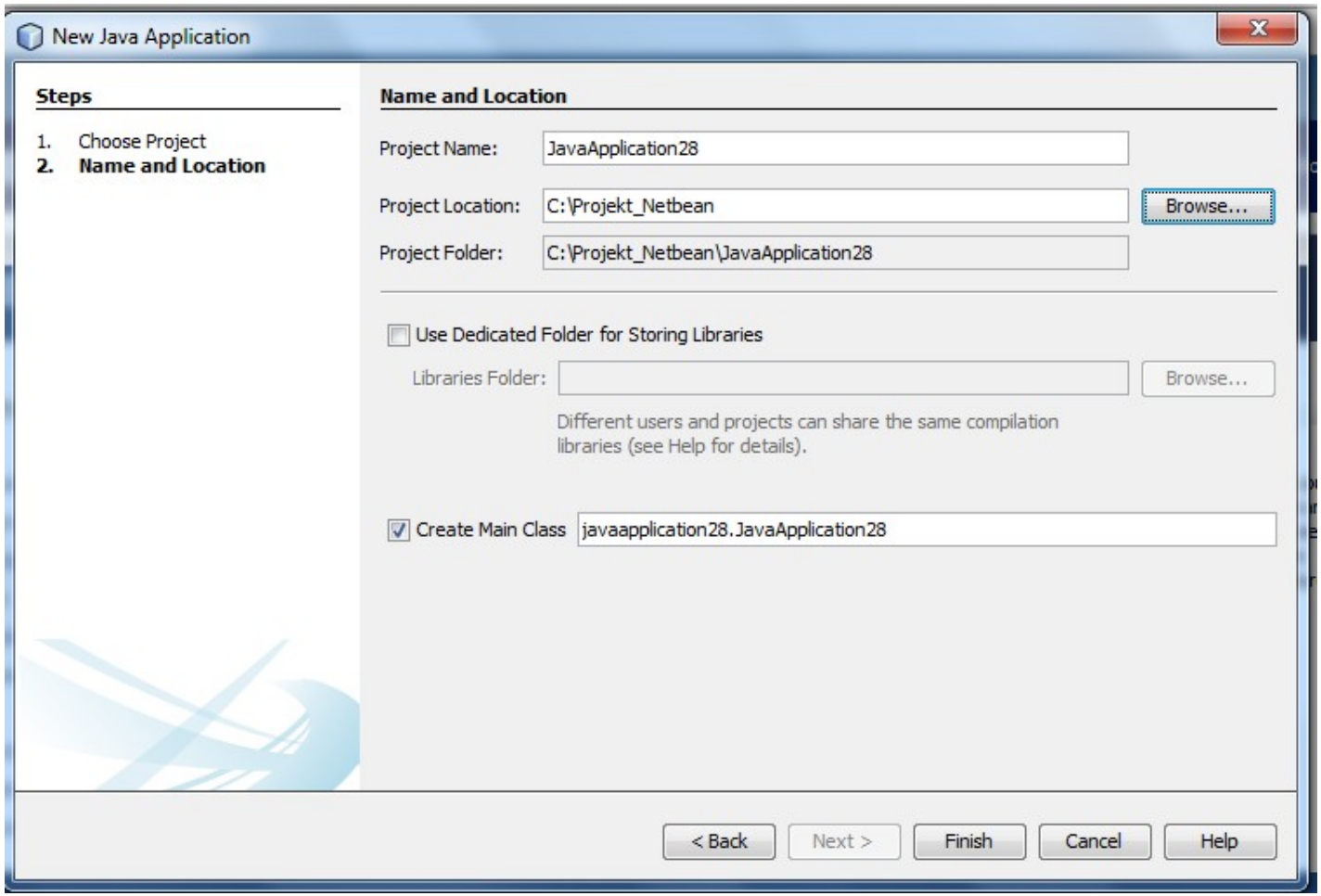


1. Schritt: Auswahl einer Projektkategorie

- *Categories* „Java“. *Projects* „Java Application“.
- Das Grundgerüst für eine Java-Anwendung wird automatisiert erstellt.



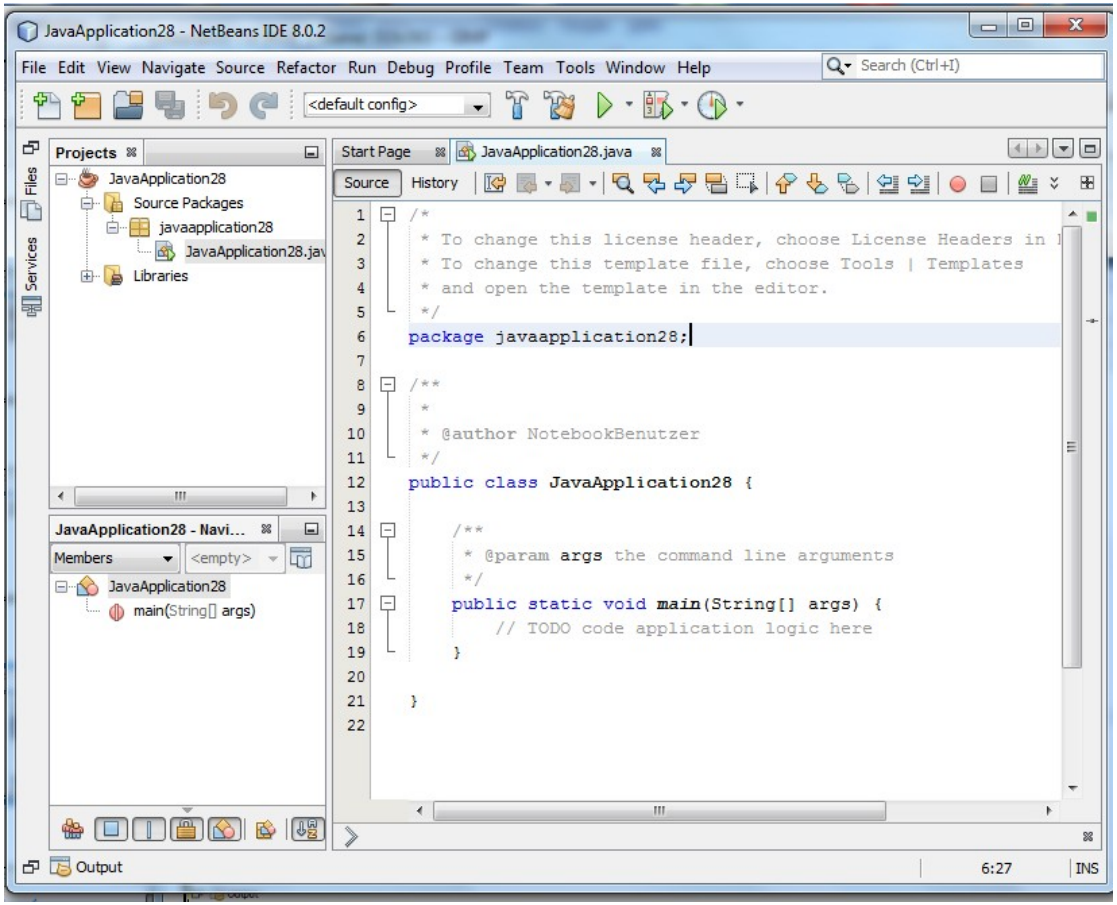
2. Schritt: Name und Speicherort des Projekts



Projekt schließen

- *File – Close Project (ProjectName)*.

Benutzeroberfläche



Aufbau der Benutzeroberfläche

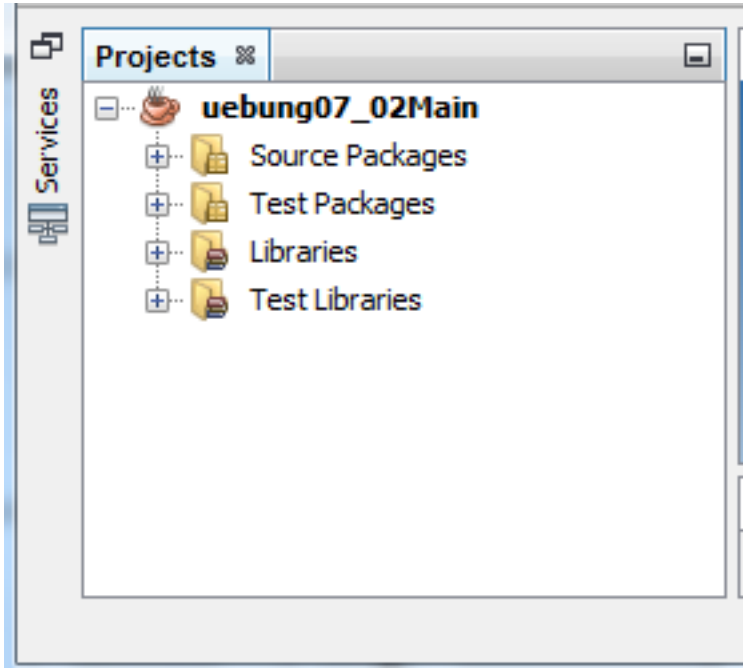
- In der Titelleiste wird der Name des Projekts angezeigt.
- Menüleiste.
- Symbolleiste für die wichtigsten Befehle. Mit Hilfe von *View* – *Toolbars* können Symbolleisten ein- oder ausgeblendet werden.
- Diverse Fenster für den Code und Informationen zu dem Projekt.

Fenster in der IDE

- Fenster bündeln mit Hilfe von Registerkarten die verschiedenen Informationen.
- Ein Fenster besitzt mindestens eine Registerkarte.

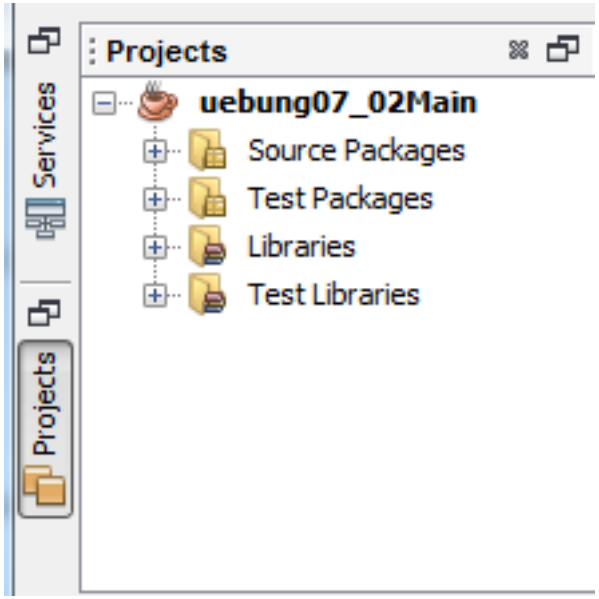
An den linken Rand angedockte Fenster

- Mit Hilfe der Schaltfläche am oberen rechten Rand des Fensters, kann dieses auf die Titelleiste minimiert werden.
- Die Titelleiste wird am linken Rand der IDE angezeigt.



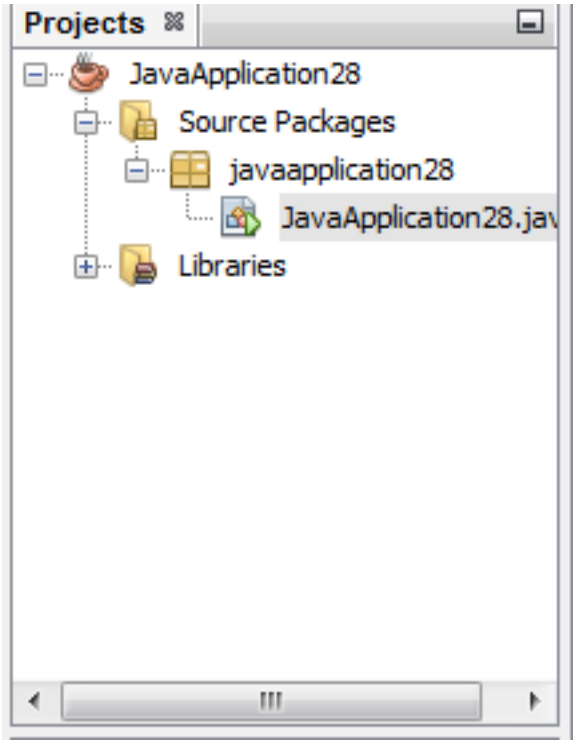
Minimiertes Fenster

- Der Mauszeiger liegt über einem Titel am linken Rand der IDE. Das Fenster wird eingeblendet.
- Mit Hilfe des Kreuzes in der Titelleiste kann das Fenster geschlossen werden.
- Das Symbol am rechten Rand dockt ein Fenster an. Das Fenster wird dauerhaft angezeigt.



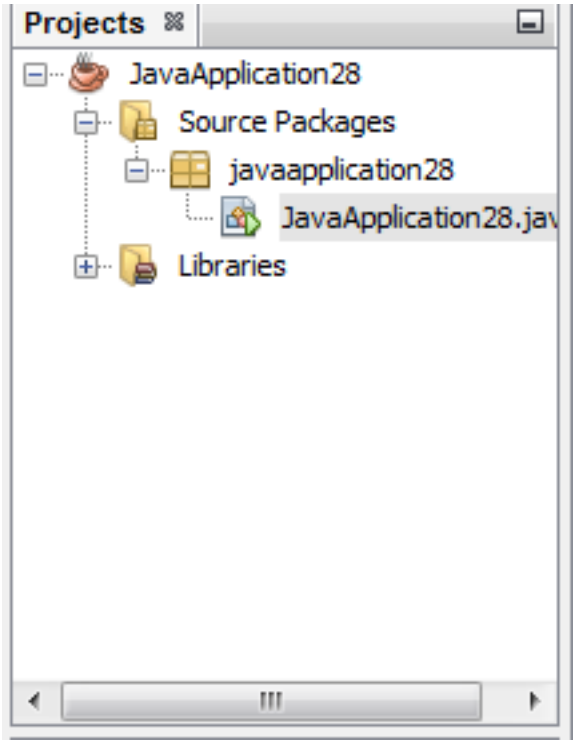
Registerkarten in einem Fenster

- Am oberen Rand der Registerkarte wird der Name und die Schließen-Schaltfläche angezeigt
- Mit Hilfe von Drag & Drop kann eine Registerkarte von einem Fenster in ein anderes Fenster verschoben werden.
- Mit Hilfe des Menüs *Window* können die verschiedenen Registerkarten geöffnet werden. Die Einträge für die Registerkarten haben ein entsprechendes Symbol am linken Rand.



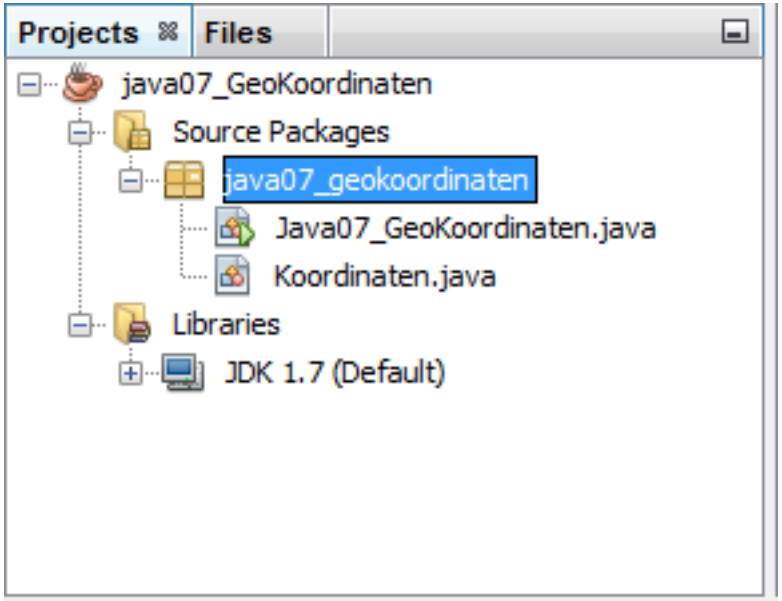
„Ordner“ öffnen oder schließen

- Klick auf das Pluszeichen: Ein Ordner wird geöffnet. Der Inhalt wird angezeigt.
- Klick auf das Minuszeichen: Ein Ordner wird geschlossen.



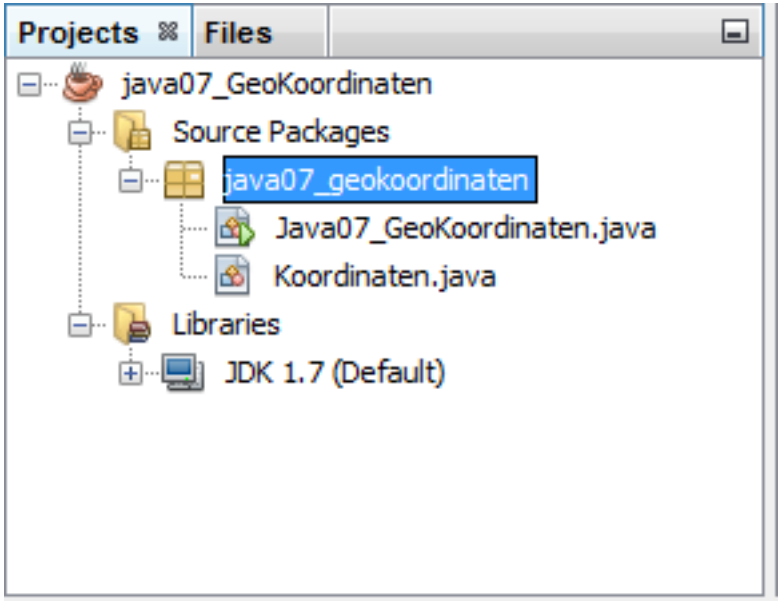
Projekt-Explorer

- Die Registerkarte *Projects* bildet die Struktur eines Java-Projektes in NetBeans ab.
- Als Wurzel wird der Name des Projekts angezeigt.
- In dem geöffneten Wurzel-Ordner werden die Ordner *Source Packages* und *Libraries* angezeigt.



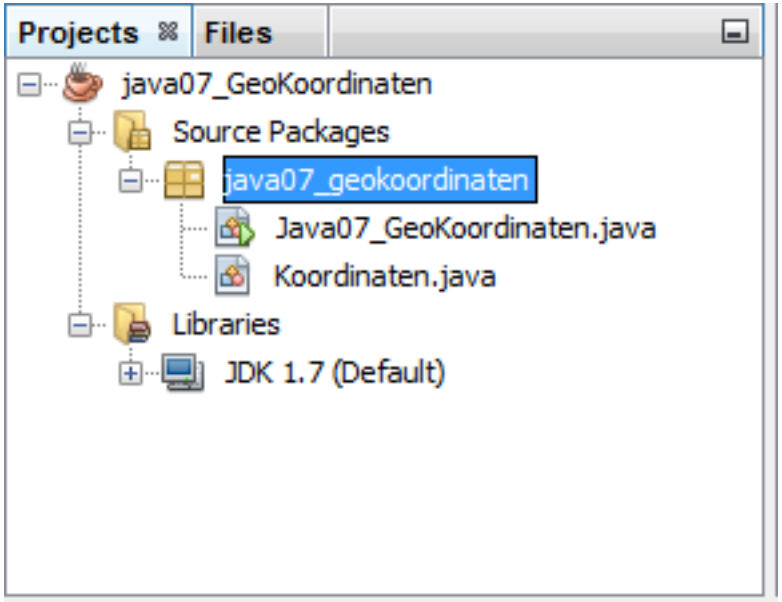
Projekt-Explorer – Ordner Source Packages

- Ablage des Programmcodes. Jede Code-Datei, die die Programmiersprache Java nutzt, hat die Endung „.java“.
- Code-Dateien können zu Paketen gebündelt werden. Jeder Unterordner des Ordners *Source Packages* symbolisiert ein Paket.



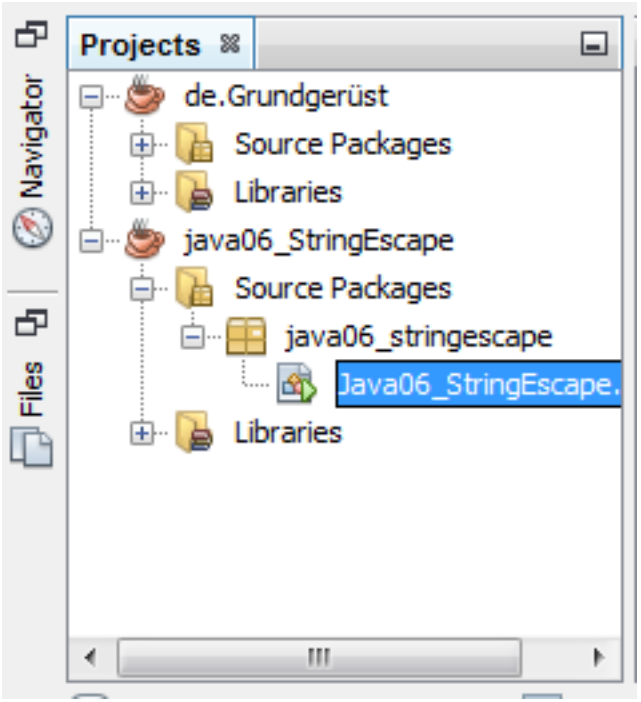
Projekt-Explorer – Ordner Libraries

- Eingebundene Bibliotheken.
- In jedem Projekt ist mindestens die Bibliothek „JDK 1.8“ als Standard-Plattform für Java eingebunden.
- Mit Hilfe des Menüs *Tools* – *Java Platform* kann die Plattform geändert werden.



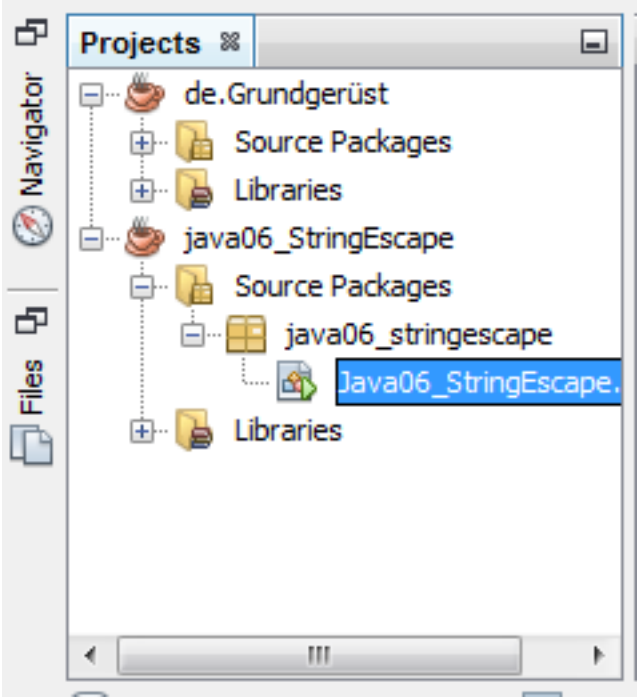
Anzeige von mehreren Projekten

- Durch einen Klick auf den Projektnamen wird ein Projekt aktiviert. Das aktive Projekt wird kompiliert und gestartet.
- Mit einem rechten Mausklick auf einen Projekttyp das Kontextmenü zu dem Projekt geöffnet.
- *File – Close Project* schließt das aktive Projekt.
- Mit Hilfe des Befehls *Delete* im Kontextmenü eines Projektes wird das aktive Projekt gelöscht.



Schließen von Projekten

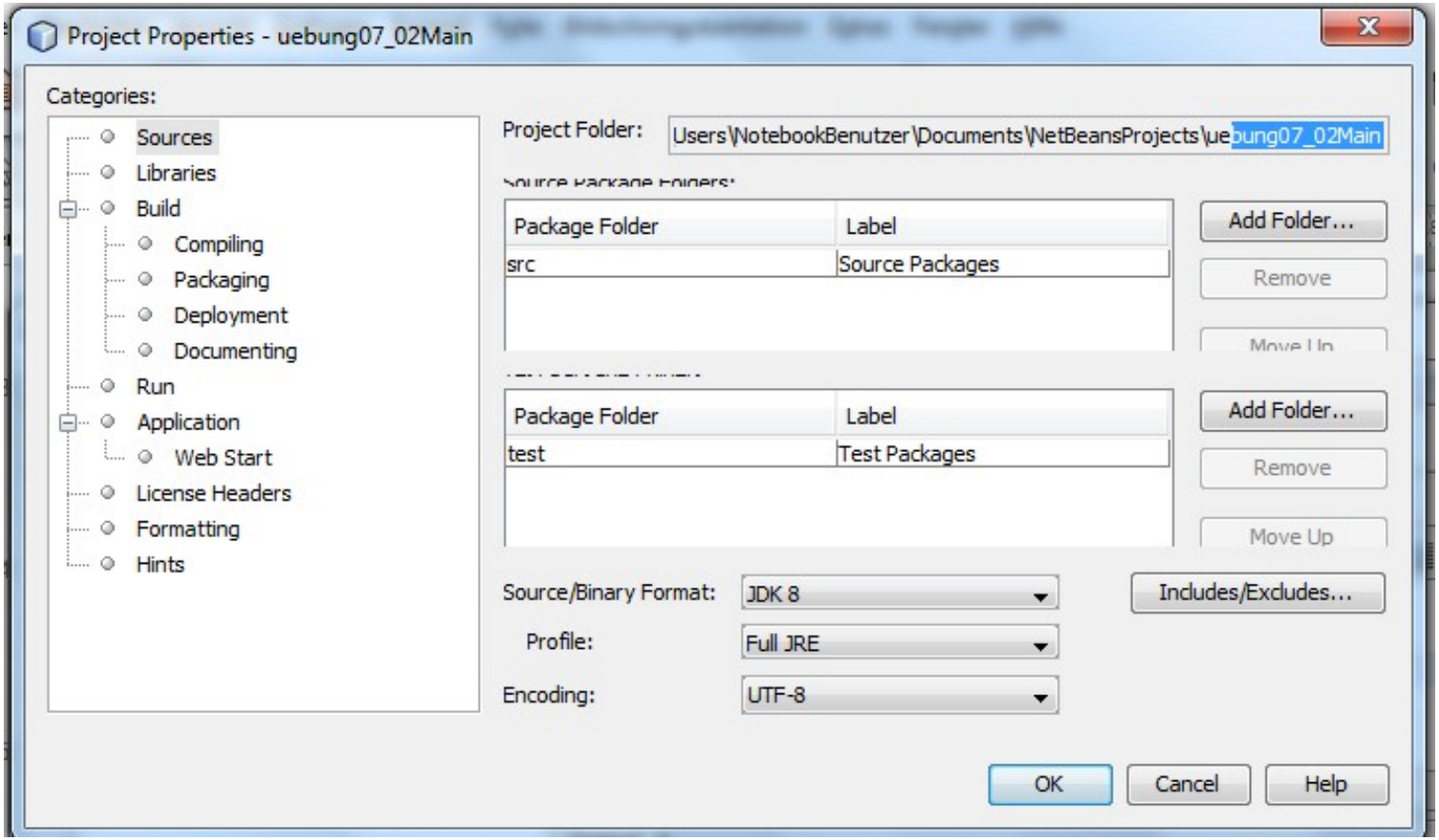
- *File – Close Project.*
- Rechtsklick auf das Wurzelverzeichnis. Menüelement *Close.*



Optionen zu einem Projekt

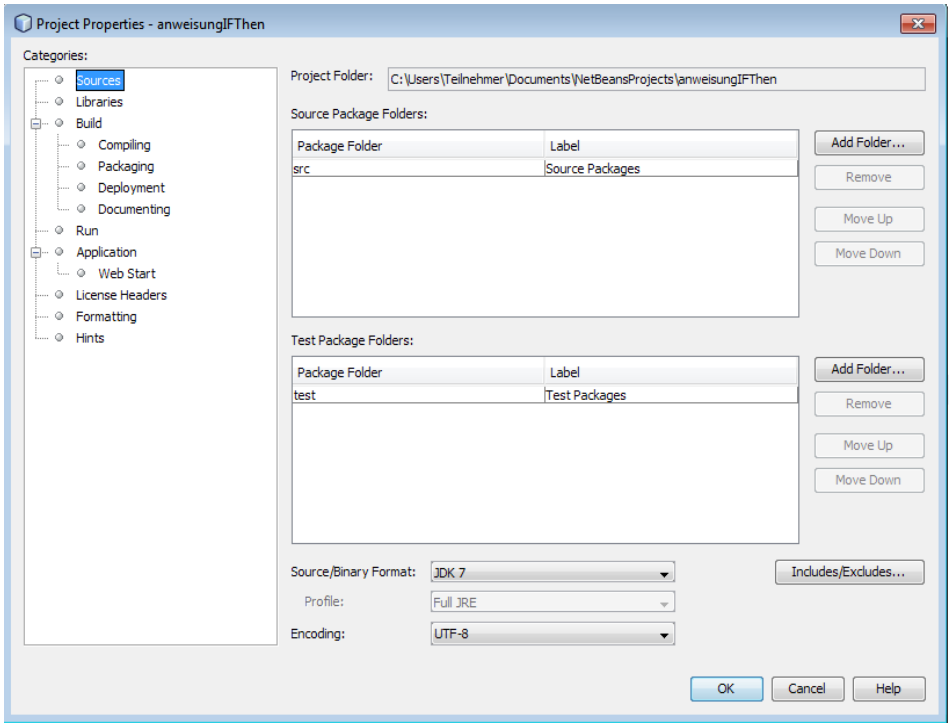
- Rechtsklick auf den Projektnamen. Menü *Properties*.
- Andere Möglichkeit: *File – Project Properties*.
- In der Liste *Categories* können die verschiedenen Kategorien der Einstellungsmöglichkeiten gewählt werden. In der Abhängigkeit der gewählten Kategorie werden die Einstellungen zu dem Projekt rechts angezeigt.

Beispiel


Project Properties - uebung07_02Main
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Project Folder:
Users\NotebookBenutzer\Documents\NetBeansProjects\uebung07_02Main
Package Folder
Label
src
Source Packages
Add Folder...
Remove
Move Up
Package Folder
Label
test
Test Packages
Add Folder...
Remove
Move Up
Source/Binary Format:
JDK 8
Includes/Excludes...
Profile:
Full JRE
Encoding:
UTF-8
OK
Cancel
Help

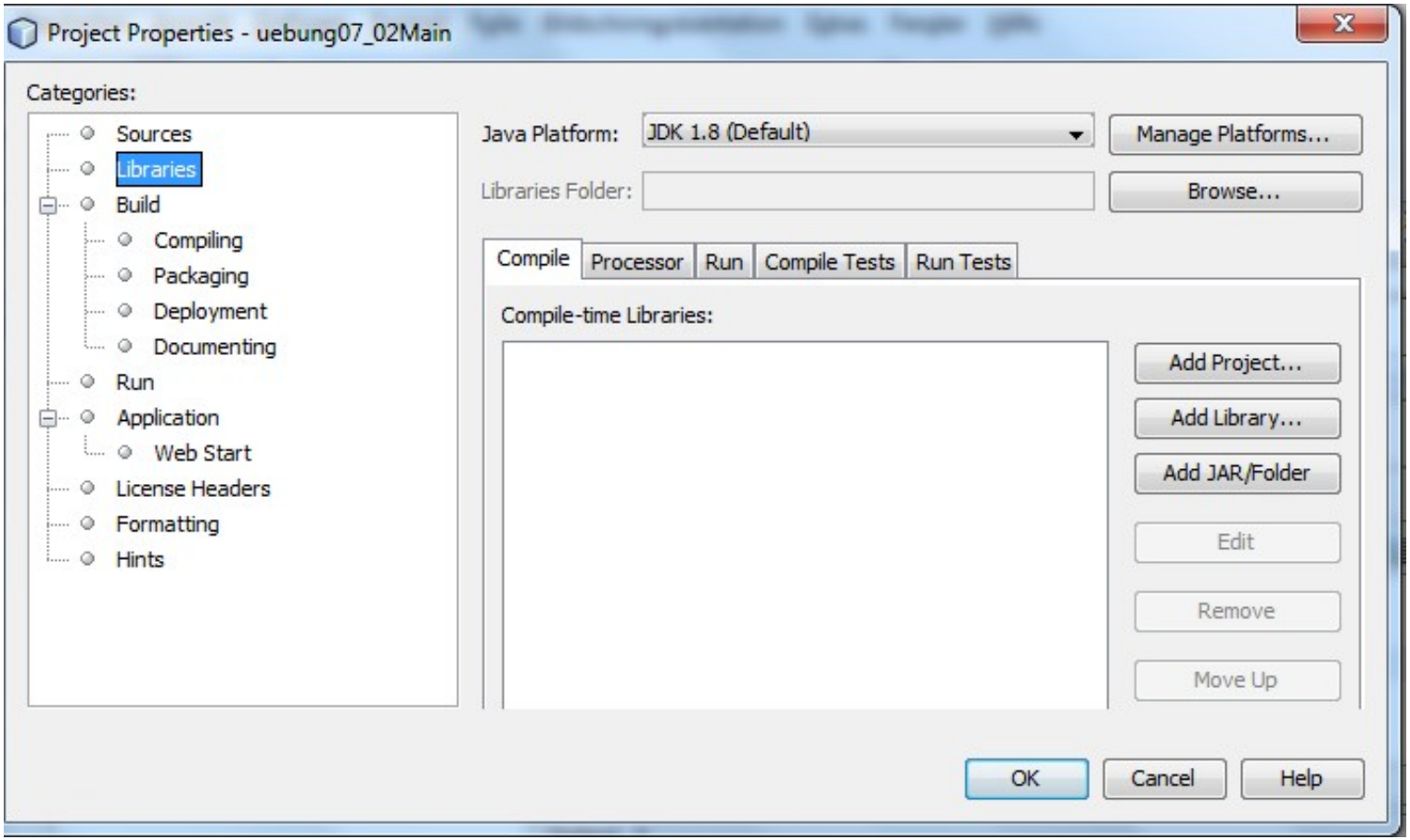
Kategorie „Sources“

- Wo werden die Quelldateien gespeichert?
- In welcher Java-Version wird die Quelldatei geschrieben?
- Welche Zeichenkodierung wird genutzt?



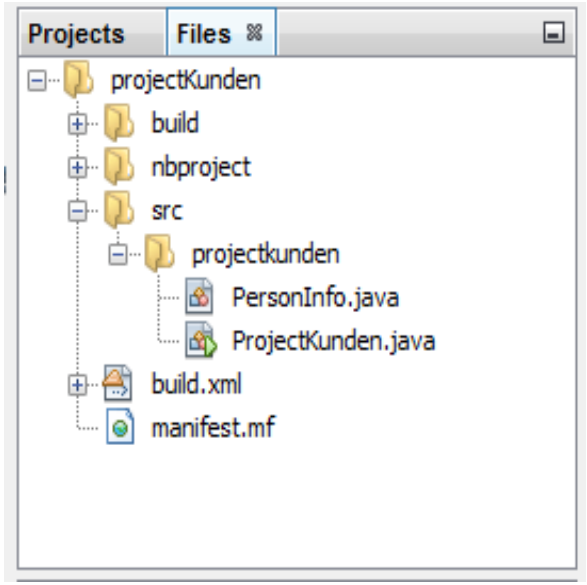
Kategorie „Libraries“

- Welche Java-Version kann in dem Projekt genutzt werden?



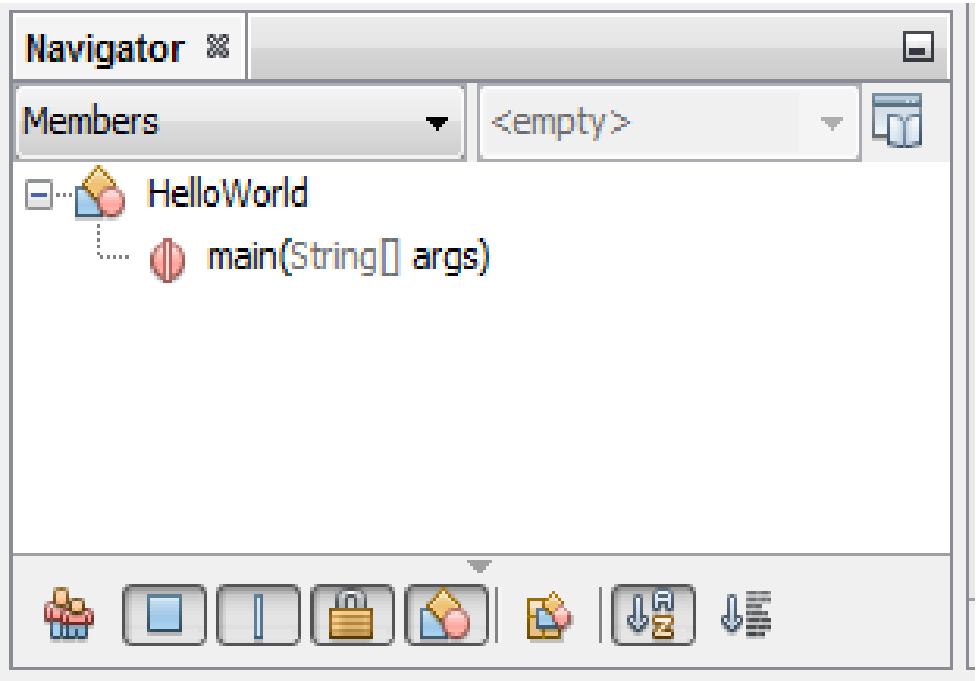
File-Explorer

- Die Registerkarte *Files* zeigt die Dateistruktur des gewählten Projektes an.
- Im Ordner *src* befindet sich der Programmcode.
- Das JAR-Archiv (das kompilierte Projekt) wird im Ordner *build* abgelegt.

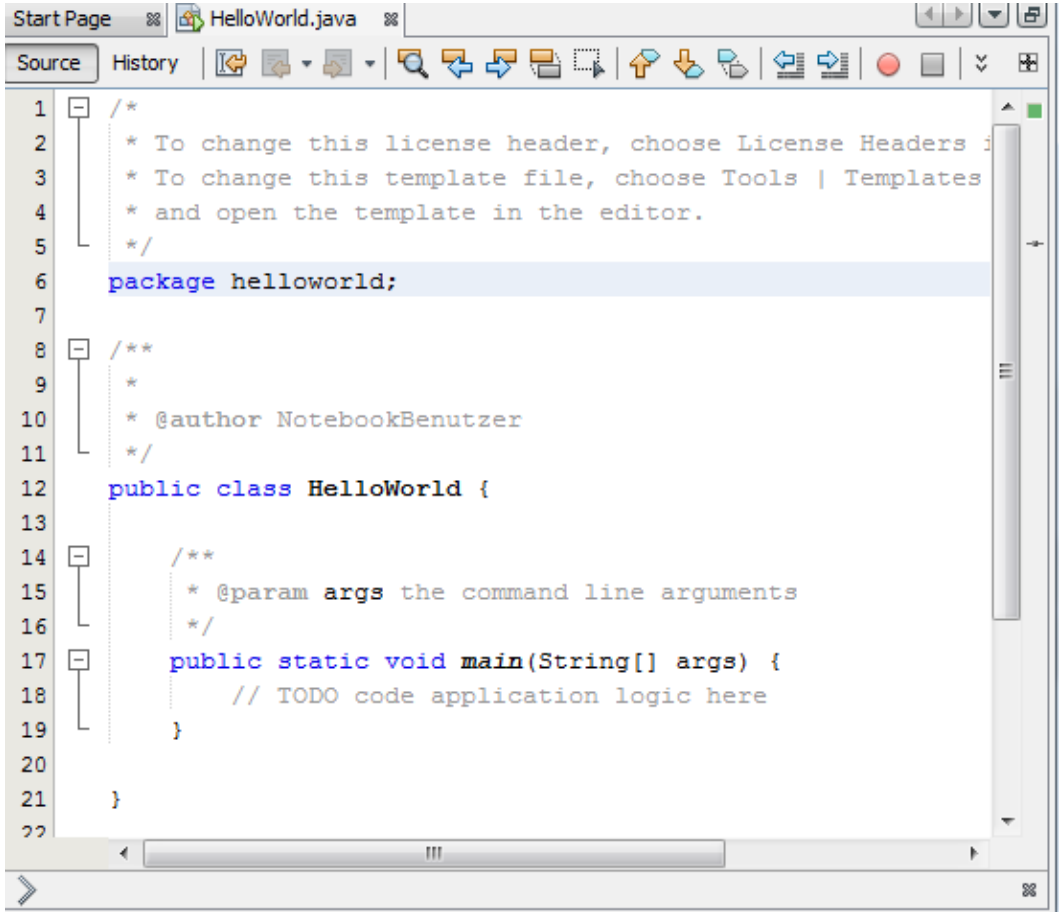


Navigator

- Strukturierte Darstellung des Codes.
- Als Wurzel wird der Name des Bauplans (der Klasse) genutzt.
- Darunter werden die Methoden der Klasse angezeigt.



Code-Fenster in NetBeans

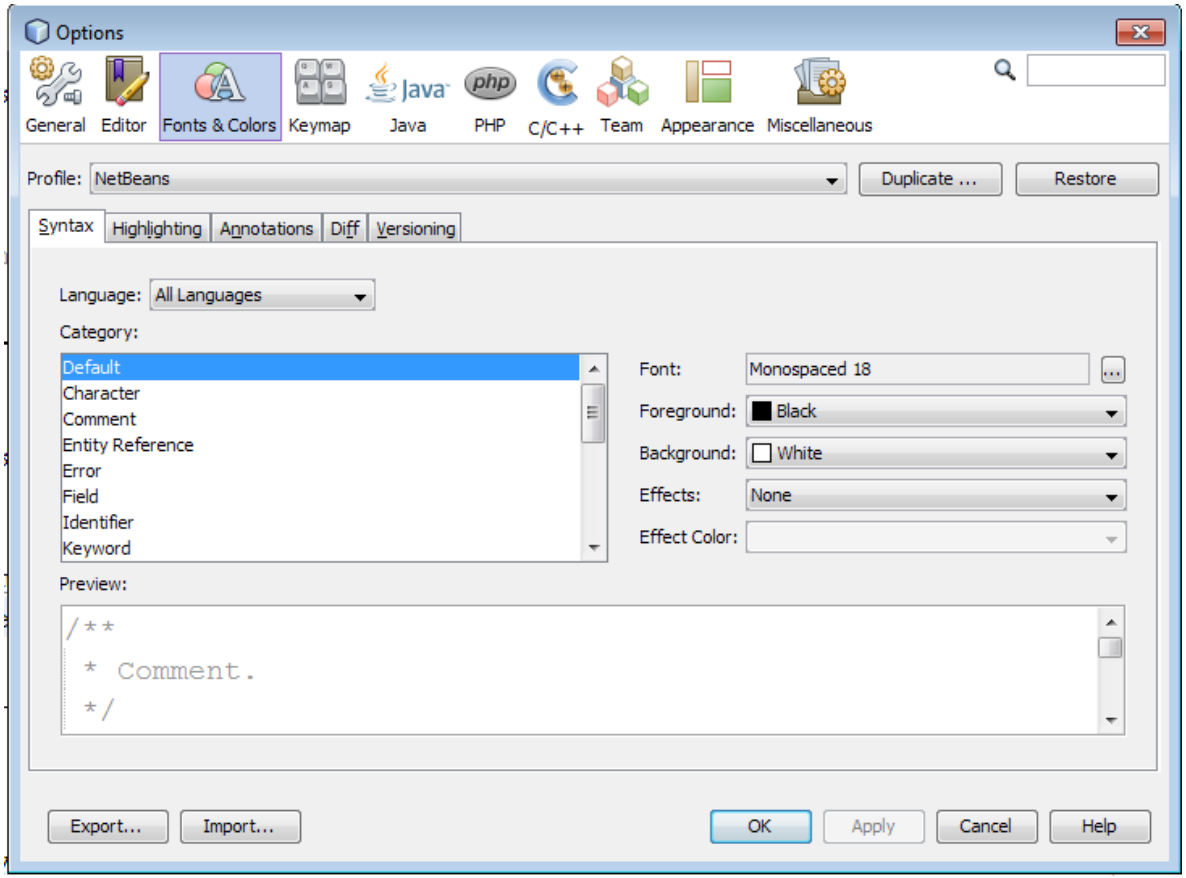


Codefenster

- Die, im Projekt-Explorer ausgewählte Datei wird angezeigt.
- Struktur einer Code-Datei in der Programmiersprache Java.
- Für jede Datei wird eine Registerkarte angelegt.

Schriftart und -farbe im Codefenster

- *Tools – Options*. *Fonts & Colors*.



Grundgerüst in „NetBeans“

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;
/**
 *
 * @author Benutzer
 */
public class HelloWorld {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

Kommentare

- Graue Schriftfarbe im Code-Fenster.
- Erläuterungen zu dem Code.
- Hilfe für den Entwickler.

Beispiele für Kommentare

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
// TODO code application logic here
```

```
/**
 *
 * @author Benutzer
 */
```

„Lizenz-Kopf“

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

- Lizenzierung von eigenen Code mit Hilfe eines mehrzeiligen Kommentars.
- *File – Project Properties*. *Categories*: „Licence Header“.

Mehrzeilige Kommentare

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

- Beginn des Kommentars `/*`.
- Ende des Kommentars `*/`.
- Die Zeilen dazwischen werden vom Compiler überlesen.

Einzeilige Kommentare

```
// TODO code application logic here
```

- Beginn des Kommentars `//`.
- Der Kommentar endet mit der Zeile.
- Einzeilige Kommentare beschreiben häufig den Code rechts vom Kommentar oder die Zeile direkt darunter.
- In diesem Beispiel ist der Kommentar ein Platzhalter für die Beschreibung der Aktivität an dieser Position

Kommentare für „Javadoc“

```
/**
 *
 * @author Benutzer
 */
```

- Aus dem Quellcode werden mit Hilfe des Tools Javadoc HTML-Dokumentationsdateien erzeugt.
- Beginn des Kommentars `/**`. Ende des Kommentars `*/`.
- Tags in Javadoc-Kommentare beginnen mit dem Add-Zeichen. Das Tag `@autor` beschreibt den Autor des Codes.
- Siehe http://www.oracle.com/technetwork/articles/java/index-137868.html

Anweisungen in Java

```
package helloworld;
```

- Anweisungen beschreiben Arbeitsschritte. Der Compiler übersetzt jeden Arbeitsschritt in Zwischencode. Der Zwischencode wird von der Java Runtime ausgeführt.
- Jede Anweisung in Java endet mit einem Semikolon.

Blöcke von Anweisungen

```
public class HelloWorld {
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

- Zusammenfassung von Anweisungen zu Codeblöcken.
- Strukturierung eines Programms in verschiedene Abschnitte.
- Ein Block beginnt mit den geschweiften Klammern und endet mit den geschweiften Klammern.

Schlüsselworte in Java

```
package helloworld;
```

- In der IDE „NetBeans“ werden Schlüsselworte standardmäßig mit einer blauen Schriftfarbe gekennzeichnet.
- In dem Skript zu dem Kurs werden die Schlüsselworte fett dargestellt.
- Alle Schlüsselworte bilden den Sprachumfang einer Programmiersprache ab. Schlüsselwörter sind reservierte Wörter der Programmiersprache Java.
- Siehe https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Packages

```
package helloworld;
```

- Das Schlüsselwort **package** zeigt die Definition eines Pakets in in einem Java-Programm an.
- In der ersten Zeile eines Java-Programms muss ein Paket definiert werden.
- Pakete sortieren Klassen (Baupläne) mit Hilfe von Ordner.
- Auf Betriebssystemebene werden Pakete als Verzeichnisse abgelegt. Zum Beispiel wird der Inhalt dieses Pakets in dem Pfad projektname/src/helloworld abgelegt.

Hinweise

- Bei der Anlage des Projekts wird der Paketname aus dem Textfeld *Project Name* gebildet.
- Der Paketname nutzt nur Kleinbuchstaben.
- Benutzerdefinierte Namen bestehen aus Buchstaben von A..Z oder a..z, den Zahlen 0..9 und dem Unterstrich.
- Projektnamen können eine Ordner-Hierarchie abbilden. Zum Beispiel bildet der Paketname `de.helloworld` die Verzeichnisstruktur de/helloword ab.

Klassen

```
public class HelloWorld {


}
```

- Jedes Java-Programm enthält mindestens eine Klasse.
- Klassen sind Baupläne für Objekte. Die Objekte können aus der realen Welt kommen.
- Klassen enthalten Attribute, die die Objekte beschreiben. Mit Hilfe von Methoden verändern die Klassen die Attribute.

Klassenkopf

```
public class HelloWorld {

}
```

- Jede Klasse wird durch das Schlüsselwort **`class`** gekennzeichnet.
- Jede Klasse hat einen eindeutigen Namen. In diesem Beispiel wird die Klasse `HelloWorld` implementiert.
- Das Schlüsselwort **`public`** regelt den Zugriff auf die Klasse von außen her. Die Klasse ist öffentlich. Jeder kann auf die Klasse zugreifen.

Methoden

```
public class HelloWorld {
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

- Beschreibung einer Aktivität.
- Zusammenfassung von zusammengehörigen Anweisungen.

Methode „main“

```
public class HelloWorld {
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

- Jedes Java-Programm muss eine Methode `main` besitzen.
- Start-Punkt der Java-Anwendung.
- Die Signatur (der Methoden-Kopf) entspricht immer dem oben angegebenen Beispiel.

Signatur der Methode

```
public class HelloWorld {
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

- Die Methode wird mit Hilfe des Namens `main` aufgerufen.
- Die Methode muss von außen aufgerufen werden. Die Methode ist als öffentlich (**`public`**) deklariert.
- Die Methode gibt keinen Wert an den Aufrufer zurück (**`void`**).
- Die Methode wird aufgerufen, ohne ein Objekt von der Klasse zu bilden (**`static`**).

Datentyp der Methode

```
public class HelloWorld {
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

- Jede Methode ist von einem bestimmten Typ.
- Der Typ der Methode legt fest, ob die Methode einen Wert an den Aufrufer zurück gibt oder nicht.
- Die Methode `main` gibt keinen Wert an die Kommandozeile zurück. Die Methode ist vom Typ **`void`**.

Parameter der Methode

```
public class HelloWorld {
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

- Einer Methode können Startwerte übergeben werden.
- Die Parameterliste folgt direkt im Anschluss an den Namen. Die runden Klammern kennzeichnen den Beginn und Ende der Liste.
- Das Feld `[] args` enthält die Kommandozeilen-Parameter vom Typ `String`. Das Feld kann beliebig viele Parameter enthalten.

Hinzufügung von Code

- Die Kommentar-Zeile in der Methode wird markiert und gelöscht.
- Eine leere Zeile wird mit Hilfe der Return-Taste eingefügt.
- Mausklick in die leere Zeile. Die Einfügemarke wird eingeblendet.
- An der Position der Einfügemarke wird zum Beispiel der Code `System.out.println("Hello World");` eingegeben.

Erläuterung

```
System.out.println("Hello Word");
```

- Mit Hilfe des Punktoperators wird eine Hierarchie von Klassen abgebildet. In diesem Beispiel liegt in dem „Ordner“ `System` der „Unterordner“ `out`.
- Der Punktoperator verbindet eine Klasse (`out`) und eine Methode (`println`). Die Methode entspricht einem „Dokument“ in dem „Ordner“ `out`.

Klasse System und out

```
System.out.println("Hello Word");
```

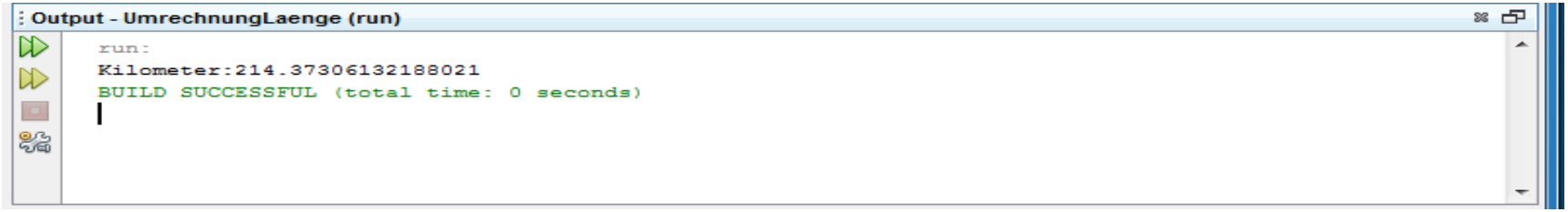
- Beide Klassen sind in der Klassenbibliothek von Java definiert.
- Die Klasse `System` symbolisiert die Standardeingabe und -ausgabe in Java.
- Die Klasse `out` nutzt die Standardausgabe. In diesem Kurs das Fenster output von NetBeans.

Methode println()

```
System.out.println("Hello Word");
```

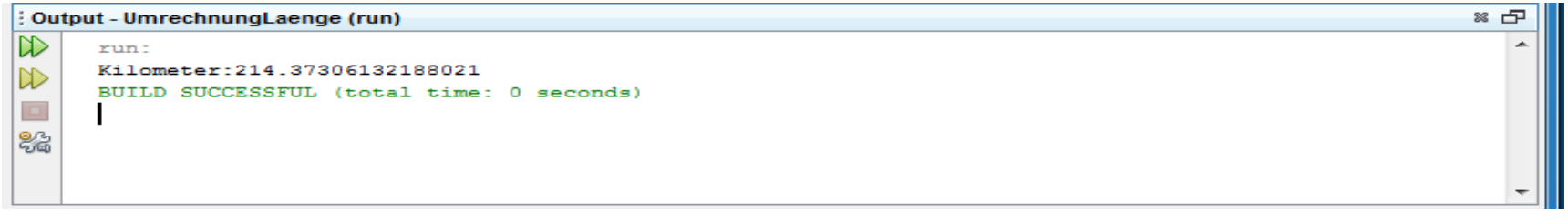
- Jede Methode ist in einer bestimmten Klasse definiert. Die Methode `println` ist in der Klasse `System.out` definiert.
- Die Methode `println` druckt eine Zeile aus. Die Zeile wird immer mit einem Zeilenvorschub beendet.
- Die auszudruckende Zeile wird als Parameter in den runden Klammern der Methode übergeben. In diesem Beispiel wird der String `Hello World` zum Ausdruck übergeben.
- Strings beginnen und enden immer mit den Anführungszeichen.

Standardausgabe in NetBeans



- Fenster *Output* am unteren Rand von NetBeans.
- Simulation einer Konsole wie zum Beispiel die MS Eingabeaufforderung etc.
- Ausgabe von Syntaxfehlern bei der Kompilierung des Programms.

… öffnen



- Nach Ausführung des Programms wird das Fenster automatisch geöffnet.
- Klick auf die Schaltfläche *Output* am unteren Rand der Anwendung NetBeans.

Kompilieren und ausführen von Projekten

- <F6>.
- *Run Project* (grüner Pfeil) in der Symbolleiste.
- *Run – Run Project*.

Kompilieren von Projekten

- <F11>.
- Klick mit der rechten Maustaste auf das Projekt. *Build.*
- *Run – Build Project*.

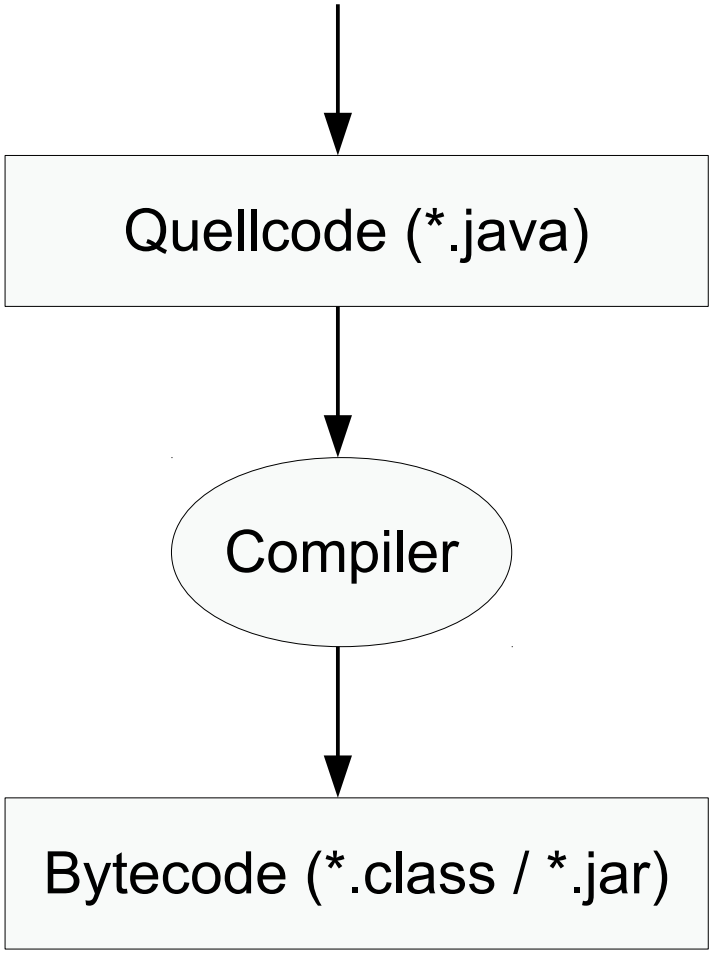
Nochmals Kompilieren

- <UMSCHALT>+<F11>.
- Klick mit der rechten Maustaste auf das Projekt. *Clean and Build.*
- *Run – Clean and Build Project*.

Ausführung stoppen

- *Run – Stop Build / Run*.
- Zum Beispiel: Die Ausführung wird beim Durchlaufen einer Endlosschleife gestoppt. Der Fehler wird behoben. Anschließend kann das Programm neu gestartet werden.

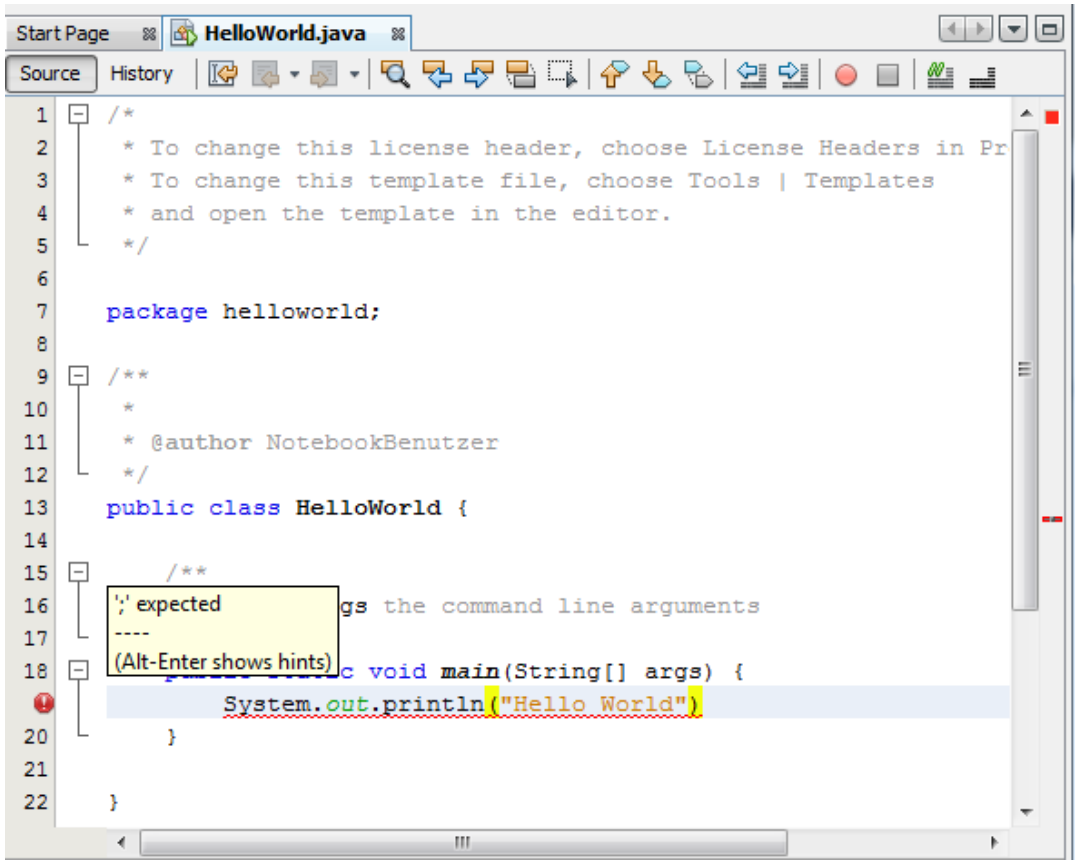
Kompilierung



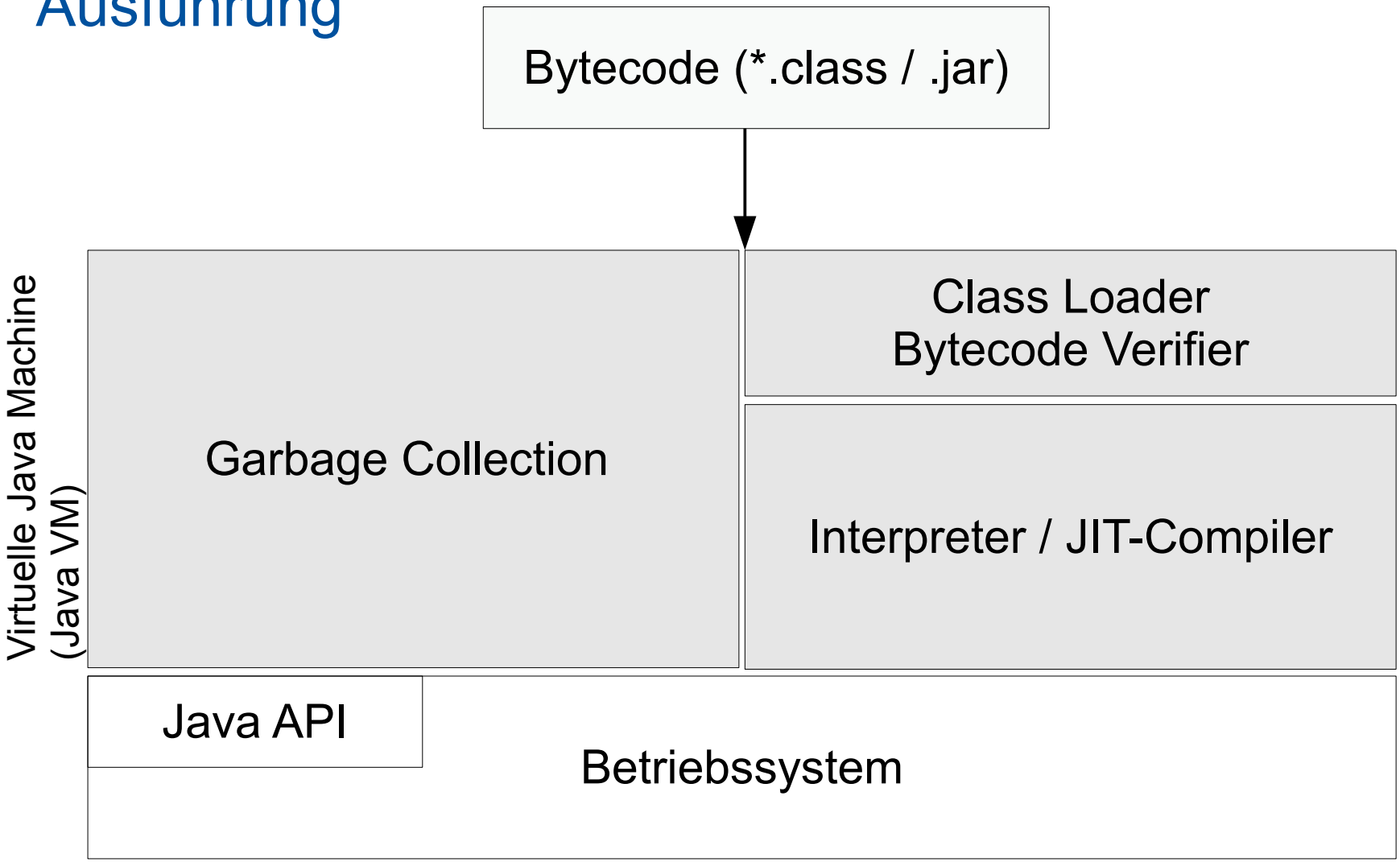
Compiler (javac)

- Überprüfung des Codes auf Korrektheit.
- Übersetzung von lesbaren Code in ein ausführbares Format.
- Syntax-Prüfung.

Syntaxfehler in NetBeans



Ausführung


Bytecode (*.class / .jar)
Virtuelle Java Machine (Java VM)
Garbage Collection
Class Loader
Bytecode Verifier
Interpreter / JIT-Compiler
Java API
Betriebssystem

Java Virtual Machine (JVM)

- Simulation eines bestimmten Betriebssystem.
- Schnittstelle zwischen dem Java-Bytecode und dem ausführenden Rechner.
- Interpretiert den Bytecode und führt das Java-Programm aus.
- Auslösung von Exceptions bei Laufzeitfehlern.

Garbage Collection

- Verwaltung von Objekten.
- In Java wird jedes Objekt referenziert. Falls auf ein Objekt keine Referenz mehr vorhanden ist, wird das Objekt automatisch entfernt.

Class Loader und Verifier

- Der Class Loader lädt den Bytecode sowie die benötigten Klassen von überall her.
- Der Verifier liest den Code und überprüft diesen auf Typsicherheit und Korrektheit. Beim Auftreten von Fehlern wird der Code nicht ausgeführt.

JIT-Compiler

- Übersetzung von Bytecode in Maschinencode des jeweiligen Betriebssystems zur Laufzeit.
- Vorteil: Beschleunigung von Programmen bei der Ausführung.