

Java für IT-Berufe

```
package java_it_berufe;

public class Java_IT_Berufe {

    public static void main(String[] args) {

        System.out.println("Informationsteil:");
        System.out.println("    - Einführung Java");
        System.out.println("    - GUI-Programmierung");
        System.out.println("    - DB-Anbindung");

        System.out.println("Aufgabenpool");

        System.out.println("Lernsituationen");
    }
}
```

3. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsseldorf Str. 23 · 42781 Haan-Gruiten

Europa-Nr.: 85535

Verfasser:

Dirk Hardy, 46049 Oberhausen

3. Auflage 2019

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Behebung von Druckfehlern untereinander unverändert sind.

ISBN 978-3-8085-8597-9

Alle Rechte vorbehalten. Das Werk ist urheberrechtlich geschützt. Jede Verwertung außerhalb der gesetzlich geregelten Fälle muss vom Verlag schriftlich genehmigt werden.

© 2019 by Verlag Europa-Lehrmittel, Nourney, Vollmer GmbH & Co. KG, 42781 Haan-Gruiten
<http://www.europa-lehrmittel.de>

Satz: Reemers Publishing Services GmbH, 47799 Krefeld

Umschlag: braunwerbeagentur, 42477 Radevormwald

Umschlagfotos: Tomy Badurina-fotolia.com, DeVlce-fotolia.com, fp-fotolia.com, bilderbox-fotolia.com

Druck: Medienhaus Plump GmbH, 53619 Rheinbreitbach

Vorbemerkung

Die Java-Technologie wurde Anfang der 90er-Jahre entwickelt, um ein eigenständiges System aus einer modernen Programmiersprache und einer ausführenden Umgebung zu erhalten. Damit sollte eine plattformunabhängige Programmierung möglich sein, denn auf jeder Plattform (auch auf einer Kaffeemaschine) brauchte nur die ausführende Umgebung vorhanden zu sein.

Der Durchbruch gelang der Java-Technologie in Verbindung mit dem Internet in den späten 90er-Jahren. Die Web-Programmierung wurde durch Java und die entsprechenden Techniken (wie Applets) deutlich vorangetrieben. Heute sind Java-Programme in allen Bereichen zu finden: nicht nur in der Web-Programmierung, sondern auch als Desktopanwendungen, im *Mobile Computing* oder auch als eingebettete Systeme (*Embedded Systems*).

Die Beschäftigung mit Java beinhaltet deshalb nicht nur das Erlernen einer objektorientierten Programmiersprache, sondern auch eine verstärkte Auseinandersetzung mit der Bandbreite der Java-Technologie – gerade für **Auszubildende in den IT-Berufen** ist das ein sehr wichtiger Aspekt.

Aufbau des Buches

Das vorliegende Buch möchte die Sprache **Java** möglichst anschaulich, praxis- und unterrichtsnah vermitteln. Damit verfolgt dieses Buch einen **praktischen Ansatz**. Es ist die Ansicht des Autors, dass gerade in der schulischen Ausbildung der Zugang zu den komplexen Themen der Programmierung verstärkt durch anschauliche und praktische Umsetzung vorbereitet werden muss. Anschließend können allgemeine und komplexe Aspekte der Programmierung oder auch der Softwareentwicklung besser verstanden und umgesetzt werden.

Das Buch ist in **drei Teile** gegliedert.

Der **erste Teil** des Buches dient als **Informationsteil** und bietet eine **systematische Einführung in die Sprache Java sowie in die Grundlagen der Java-Technologie**. Ein Einstieg in die GUI-Programmierung mit den klassischen Bibliotheken AWT und Swing, aber auch ein Einstieg in die moderne JavaFX-Technik sowie die Anbindung von Datenbanken runden diesen Informationsteil ab.

Der **zweite Teil** des Buches ist eine **Sammlung von Übungsaufgaben**. Nach der Erarbeitung der entsprechenden Kenntnisse aus dem Informationsteil können die Aufgaben aus diesem Teil zur weiteren Auseinandersetzung mit den Themen dienen und durch verschiedene Schwierigkeitsgrade auch die Differenzierung im Unterricht ermöglichen.

Der **dritte Teil** des Buches beinhaltet **Lernsituationen** basierend auf dem Lernfeld „Entwickeln und Bereitstellen von Anwendungssystemen“ aus dem Rahmenlehrplan für die IT-Berufe (speziell Fachinformatiker-Anwendungsentwicklung). Lernsituationen konkretisieren sich aus den Lernfeldern und sollen im Idealfall vollständige Handlungen darstellen (Planen, Durchführen, Kontrollieren). Aus diesem Grund werden die Lernsituationen so angelegt, dass neben einer Planungsphase nicht nur die Durchführung (Implementation des Programms) im Blickpunkt steht, sondern auch geeignete Testverfahren zur Kontrolle des Programms bzw. des Entwicklungsprozesses in die Betrachtung einbezogen werden. Die Lernsituationen können aber auch als **Projektideen** verstanden werden.

Das Buch ist für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen (speziell Fachinformatiker), aber auch von den informationstechnischen Assistenten genutzt werden. Ebenso vorstellbar ist der Einsatz des Buches in der gymnasialen Oberstufe sowie zu den Übungen der Basisvorlesungen der Fachhochschulen mit informationstechnischen Studiengängen.

Als Entwicklungswerkzeug wird in diesem Buch **Apache NetBeans 10.0** genutzt. Diese Entwicklungsumgebung ist kostenfrei als Download im Internet verfügbar.

Für Anregungen und Kritik zu diesem Buch sind wir Ihnen dankbar (gerne auch per E-Mail).

Dirk Hardy

Im Sommer 2019

E-Mail: Hardy@DirkHardy.de

Verlag Europa-Lehrmittel

E-Mail: Info@Europa-Lehrmittel.de

Vorbemerkung	3
Aufbau des Buches	3
Teil 1 Einführung in Java	11
1 Einführung in die Java-Technologie.....	13
1.1 Die Java-Technologie.....	13
1.1.1 Entstehung der Java-Technologie.....	13
1.1.2 Eigenschaften der Java-Technologie.....	14
1.1.3 Die Komponenten der Java-Technologie.....	14
1.1.4 Kompilierung von Java-Programmen	14
1.2 Die Sprache Java.....	15
1.2.1 Entwicklung der Sprache Java	15
1.2.2 Eigenschaften der Sprache Java	15
1.2.3 Schlüsselworte in Java.....	16
1.2.4 Prozedurale, strukturierte und objektorientierte Programmierung unter Java	16
1.2.5 Bestandteile eines Java-Programms.....	17
2 Das erste Java-Programm	19
2.1 Ein Java-Projekt anlegen	19
2.2 Das erste Java-Programm	22
2.2.1 Das Java-Grundgerüst.....	22
2.2.2 Pakete	22
2.2.3 Die Klasse HalloWelt und die Hauptmethode main	23
2.2.4 Die Ausgabe auf dem Bildschirm	24
2.2.5 Wichtige Regeln eines Java-Programms.....	25
2.3 Grundlegende Konventionen in Java	25
2.3.1 Bezeichner (Namen) in Java.....	25
2.3.2 Trennzeichen.....	26
2.3.3 Kommentare in Java	27
2.4 Datentypen und Variablen.....	29
2.4.1 Variablen in Java	29
2.4.2 Elementare Datentypen	30
2.4.3 Deklaration einer Variablen	30
2.4.4 Operationen auf den elementaren Datentypen.....	31
2.4.5 Konstante Variablen.....	33
3 Aus- und Eingabe in Java	34
3.1 Ausgabe in Java	34
3.1.1 Ausgabe von Variablen	34
3.2 Eingabe über die Konsole	36
3.2.1 Zeichenketten einlesen	36
3.2.2 Konvertierung der Eingabe	36
4 Operatoren in Java.....	39
4.1 Arithmetische Operatoren	39
4.1.1 Elementare Datentypen und ihre arithmetischen Operatoren	39
4.1.2 Der Modulo-Operator	40
4.1.3 Inkrement- und Dekrementoperatoren	40
4.2 Relationale und logische Operatoren.....	41
4.2.1 Relationale Operatoren.....	41
4.2.2 Logische Operatoren	42
4.3 Bit-Operatoren und weitere Operatoren	43
4.3.1 Logische Bit-Operatoren.....	43
4.3.2 Bit-Schiebeoperatoren	44
4.3.3 Typumwandlung mit cast-Operatoren	44
4.3.4 Zuweisung und gekoppelte Zuweisung	45
4.4 Rang von Operatoren	46

5	Selektion und Iteration	48
5.1	Die Selektion	48
5.1.1	Darstellung der Selektion mit einem Programmablaufplan.....	48
5.1.2	Die einseitige Selektion mit der if-Anweisung.....	49
5.1.3	Die zweiseitige Selektion mit der if-else-Anweisung	49
5.1.4	Verschachtelte Selektionen mit if und if-else.....	51
5.1.5	Mehrfachselektion mit switch	52
5.2	Fuß-, kopf- und zählergesteuerte Iterationen.....	55
5.2.1	Die do-while-Schleife.....	55
5.2.2	Die while-Schleife	57
5.2.3	Die for-Schleife	58
5.2.4	Abbruch und Sprung in einer Schleife.....	60
6	Das Klassenkonzept in Java	61
6.1	Die erste Klasse in Java.....	63
6.1.1	Aufbau einer Klasse in Java.....	63
6.1.2	Werttypen und Verweistypen.....	65
6.2	Methoden in Java.....	65
6.2.1	Aufbau einer Methode.....	65
6.2.2	Rückgabewert einer Methode.....	67
6.2.3	Lokale Variablen	68
6.2.4	Übergabeparameter einer Methode.....	69
6.2.5	Überladen von Methoden.....	73
6.2.6	Zusammenfassende Hinweise zu Methoden	74
6.3	Weitere Elemente von Klassen.....	75
6.3.1	Konstruktoren und der Destruktor	75
6.3.2	Der this-Verweis.....	79
6.3.3	Statische Klassenelemente	80
6.3.4	Konstante Klassenelemente	81
6.4	Aufzählungstypen	81
6.4.1	Einfache Aufzählungen	81
6.4.2	Klassen von Aufzählungen	82
7	Vererbung in Java	84
7.1	Die Vererbung in Java	84
7.1.1	Die einfache Vererbung	84
7.1.2	Umsetzung der Vererbung in Java.....	85
7.1.3	Zugriff auf Attribute.....	87
7.1.4	Finale Klassen	88
7.2	Polymorphismus	88
7.2.1	Die Klasse Object.....	88
7.2.2	Zuweisungen innerhalb von Vererbungshierarchien.....	90
7.2.3	Überschreiben von Methoden	91
7.3	Abstrakte Basisklassen.....	94
7.3.1	Eine abstrakte Basisklasse.....	94
7.4	Interfaces in Java.....	95
7.4.1	Aufbau eines Interfaces.....	95
8	Arrays in Java	98
8.1	Ein- und mehrdimensionale Arrays.....	98
8.1.1	Eindimensionale Arrays.....	98
8.1.2	Die for each-Schleife.....	100
8.1.3	Mehrdimensionale Arrays	102
8.1.4	Arrays kopieren.....	105
8.1.5	Arrays von Objekten	107
8.1.6	Übergabe von Arrays an Methoden	108
8.2	Sortieren von Arrays	110
8.2.1	Das Sortieren durch Auswahl	110
8.2.2	Statische Sortiermethode sort.....	112
8.2.3	Das Interface Comparable	113

8.3	Besondere Array-Klassen	114
8.3.1	Die Klasse ArrayList	114
8.3.2	Die Klasse HashMap	115
9	Dateioperationen in Java	118
9.1	Lesen und Schreiben von Dateien	119
9.1.1	Sequenzielles Lesen und Schreiben	119
9.1.2	Direkter Zugriff in Dateien	121
9.2	Textdateien lesen und schreiben	123
9.2.1	Textdateien mit dem PrintWriter schreiben	123
9.2.2	Textdateien mit dem Scanner lesen	124
9.3	Serialisierung von Objekten	125
9.4	Methoden der Klasse File	128
9.4.1	Methoden der Klasse File	128
9.4.2	Verzeichnisse auflisten	129
10	Fortgeschrittene Themen in Java	131
10.1	Ausnahmen – Exceptions	131
10.1.1	Versuchen und Auffangen (try and catch)	131
10.1.2	System-Exceptions	133
10.1.3	Der finally-Block	135
10.1.4	Ausnahmen werfen	136
10.1.5	Eigene Exception-Klassen erstellen	137
10.2	Generische Programmierung	138
10.2.1	Generische Methoden	138
10.2.2	Generische Klassen	139
10.2.3	Generische Listenklassen benutzen	140
10.3	Lambda-Ausdrücke	141
10.4	Java und UML	143
10.4.1	Das Klassendiagramm	143
10.4.2	Darstellung der Attribute im Klassendiagramm	144
10.4.3	Darstellung der Methoden im Klassendiagramm	145
10.4.4	Umsetzung eines Klassendiagramms	147
10.4.5	Beziehungen zwischen Klassen	150
10.4.6	Die Assoziation	151
10.4.7	Die Aggregation	154
10.4.8	Die Komposition	157
11	GUI-Programmierung mit dem Abstract Window Toolkit AWT	159
11.1	GUI-Programmierung	159
11.1.1	Historische Entwicklung der GUI-Programmierung	159
11.1.2	Aufbau des AWT	160
11.1.3	Grundbegriffe der GUI-Programmierung	160
11.2	Das erste GUI-Programm	161
11.2.1	Die Klasse Frame nutzen	161
11.2.2	Eine eigene Frame-Klasse schreiben	162
11.3	Text- und Grafikausgabe	163
11.3.1	Das Paint-Ereignis und die erste Textausgabe	163
11.3.2	Einen Clientbereich hinzufügen	165
11.3.3	Einfache Grafikausgabe	165
11.3.4	Mehrzeilige Textausgabe	168
11.4	Ereignisgesteuerte Programmierung	170
11.4.1	Grundlage der ereignisgesteuerten Programmierung	170
11.4.2	Ereignisarten und Ereignisempfänger	171

12	Steuerelemente mit dem AWT und mit Swing-Klassen	175
12.1	Steuerelemente mit dem AWT	175
12.1.1	Einfache Steuerelemente	175
12.1.2	Steuerelemente benutzen	175
12.1.3	Auf Ereignisse reagieren	176
12.1.4	Beispielanwendung mit einfachen Steuerelementen	177
12.1.5	Mit dem Layoutmanager Steuerelemente anordnen	180
12.2	Steuerelemente mit Swing-Klassen	182
12.2.1	Grundlagen der Swing-Klassen	182
12.2.2	Swing-Steuerelemente	183
12.2.3	Einfache Swing-Steuerelemente einsetzen	184
12.2.4	Look and Feel	187
12.3.1	Die Baumansicht JTree	187
12.3.2	Anlegen von Knoten in einem JTree	188
12.3.3	Wichtige JTree-Methoden im Überblick	190
12.3.4	Auf JTree-Ereignisse reagieren	190
12.3.5	Tabellen mit JTable	192
12.3.6	Wichtige JTable-Methoden im Überblick	193
12.3.7	Auf JTable-Ereignisse reagieren	194
12.3.8	Steuerelemente mit Bildlaufleisten versehen	195
13	Menüs und Dialoge	198
13.1	Menüs mit dem AWT erstellen	198
13.1.1	Ein Menü erstellen	198
13.1.2	Auf Menü-Ereignisse reagieren	199
13.1.3	Ein Kontextmenü erstellen	200
13.1.4	Menüs mit den Swing-Klassen	201
13.2	Dialoge	203
13.2.1	Standarddialoge nutzen	203
13.2.2	Eigene Dialoge erstellen	206
14	JavaFX-Anwendungen entwickeln	210
14.1	Grundkonzept von JavaFX	210
14.2	Aufbau einer JavaFX-Anwendung	211
14.3	Container und Steuerelemente	213
14.4	Ereignisbehandlung	218
14.4.1	Ereignisbehandlung mit einer allgemeinen Methode	218
14.4.2	Ereignisbehandlung mit einer inneren Klasse	219
14.5	JavaFXML-Anwendungen	221
14.5.1	Die erste JavaFXML-Anwendung	221
14.5.2	Den Scene Builder einsetzen	223
14.5.3	Steuerelemente mit dem Controller verbinden	223
15	Datenbankanbindung	226
15.1	Datenbankzugriff mit Java	226
15.1.1	Datenbankanbindung mit JDBC	226
15.1.2	JDBC-Treiber laden und eine Verbindung aufbauen	226
15.1.3	Zugriff auf eine SQLite-Datenbank	227
15.1.4	Nicht-Select-Befehle absetzen	230
15.1.5	Metadaten ermitteln	232
15.2	Weitere Datenbanken ansprechen	233
15.2.1	Einen Treiber hinzufügen	233
15.2.2	Weitere Datenbanktreiber	234

Teil 2 Aufgabenpool 235**Aufgabenpool..... 236**

1	Aufgaben zur Einführung in die Java-Technologie.....	236
2	Aufgaben zum ersten Java-Programm	236
3	Aufgaben zur Ein- und Ausgabe in Java	237
4	Aufgaben zu Operatoren in Java	238
5	Aufgaben zur Selektion und Iteration	240
6	Aufgaben zum Klassenkonzept in Java.....	244
7	Aufgaben zur Vererbung in Java.....	247
8	Aufgaben zu Arrays in Java	250
9	Aufgaben zu Dateioperationen in Java.....	255
10	Aufgaben zu fortgeschrittenen Themen in Java	261
11	Aufgaben zur GUI-Programmierung mit dem AWT.....	266
12	Aufgaben zu Steuerelementen mit dem AWT oder den Swing-Klassen	268
13	Aufgaben zu Menüs und Dialogen	270
14	Aufgaben zu JavaFX-Anwendungen	271
15	Aufgaben zur Datenbankankbindung.....	273

Teil 3 Lernsituationen 275

Lernsituation 1:	Erstellen einer Präsentation mit Hintergrundinformationen zu der Sprache Java (in Deutsch oder Englisch)	276
Lernsituation 2:	Anfertigen einer Kundendokumentation für den Einsatz einer Entwicklungsumgebung in Java (in Deutsch oder Englisch).....	277
Lernsituation 3:	Entwicklung eines Verschlüsselungsverfahrens für ein internes Memo-System der Support-Abteilung einer Netzwerk-Firma.....	279
Lernsituation 4:	Planung, Implementierung und Auswertung eines elektronischen Fragebogens.....	280
Lernsituation 5:	Entwicklung einer Software zur Darstellung von Wetterdaten mit dem Model-View-Controller-Konzept.....	283
Lernsituation 6:	Entwicklung einer JavaFXML-Anwendung, um Sudokus zu lösen.....	286

Index 289

Teil Einführung in Java

1.1	Die Java-Technologie	13
1.2	Die Sprache Java	15
2.1	Ein Java-Projekt anlegen	19
2.2	Das erste Java-Programm	22
2.3	Grundlegende Konventionen in Java	25
2.4	Datentypen und Variablen	29
3.1	Ausgabe in Java	34
3.2	Eingabe über die Konsole	36
4.1	Arithmetische Operatoren	39
4.2	Relationale und logische Operatoren	41
4.3	Bit-Operatoren und weitere Operatoren	43
4.4	Rang von Operatoren	46
5.1	Die Selektion	48
5.2	Fuß-, kopf- und zählergesteuerte Iterationen	55
6.1	Die erste Klasse in Java	63
6.2	Methoden in Java	65
6.3	Weitere Elemente von Klassen	75
6.4	Aufzählungstypen	81
7.1	Die Vererbung in Java	84
7.2	Polymorphismus	88
7.3	Abstrakte Basisklassen	94
7.4	Interfaces in Java	95
8.1	Ein- und mehrdimensionale Arrays	98
8.2	Sortieren von Arrays	110
8.3	Besondere Array-Klassen	114
9.1	Lesen und Schreiben von Dateien	119
9.2	Textdateien lesen und schreiben	123
9.3	Serialisierung von Objekten	125
9.4	Methoden der Klasse File	128
10.1	Ausnahmen – Exceptions	131
10.2	Generische Programmierung	138
10.3	Lambda-Ausdrücke	141
10.4	Java und UML	143
11.1	GUI-Programmierung	159
11.2	Das erste GUI-Programm	161
11.3	Text- und Grafikausgabe	163
11.4	Ereignisgesteuerte Programmierung	170
12.1	Steuerelemente mit dem AWT	175
12.2	Steuerelemente mit Swing-Klassen	182

13.1	Menüs mit dem AWT erstellen.....	198
13.2	Dialoge	203
14.1	Grundkonzept von JavaFX	210
14.2	Aufbau einer JavaFX-Anwendung	211
14.3	Container und Steuerelemente.....	213
14.4	Ereignisbehandlung.....	218
14.5	JavaFXML-Anwendungen	221
15.1	Datenbankzugriff mit Java	226
15.2	Weitere Datenbanken ansprechen	233

1 Einführung in die Java-Technologie

1.1 Die Java-Technologie

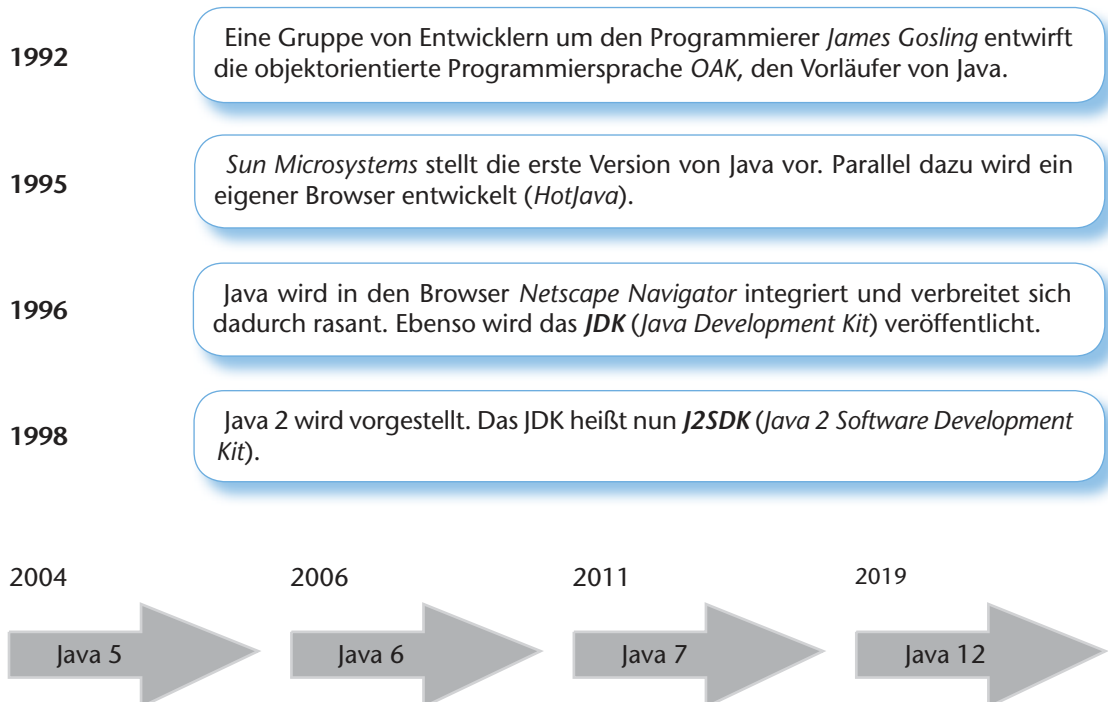
1.1.1 Entstehung der Java-Technologie

Anfang der 90er-Jahre wurde bei der Firma *Sun Microsystems* eine Programmiersprache entwickelt, die nicht nur auf PCs, sondern auf verschiedenen elektronischen Geräten (beispielsweise tragbaren Minicomputern oder auch in intelligenten Kaffeemaschinen) einsetzbar sein sollte. Diese Programmiersprache sollte *OAK* heißen. Allerdings war dieser Name geschützt, sodass sich die Entwickler einen neuen Namen einfallen lassen mussten: **JAVA**¹.

Die erste Version von Java wurde 1995 von *Sun Microsystems* vorgestellt. Die Sprache war internetfähig – sie konnte in einem bestimmten Browser (*HotJava*) ausgeführt werden. Die Firma *Netscape* schloss dann 1996 einen Vertrag mit *Sun Microsystems* und damit breitete sich Java über den berühmten Browser von *Netscape* (den *Netscape Navigator*) rasend schnell aus.

Inzwischen ist Java ein mächtiges Werkzeug zur Entwicklung von Internet-, aber auch Desktop-Anwendungen. Ebenso hat es einen großen Anteil an der Entwicklung im Bereich des *Mobile Computing*.

Die folgende Grafik zeigt den zeitlichen Verlauf der Java-Technologie-Entwicklung:



¹ Der Name Java soll auf den enormen Kaffeedurst der Entwickler zurückzuführen sein, denn Java ist in vielen Ländern (auch den Vereinigten Staaten) ein Synonym für Kaffee.

1.1.2 Eigenschaften der Java-Technologie

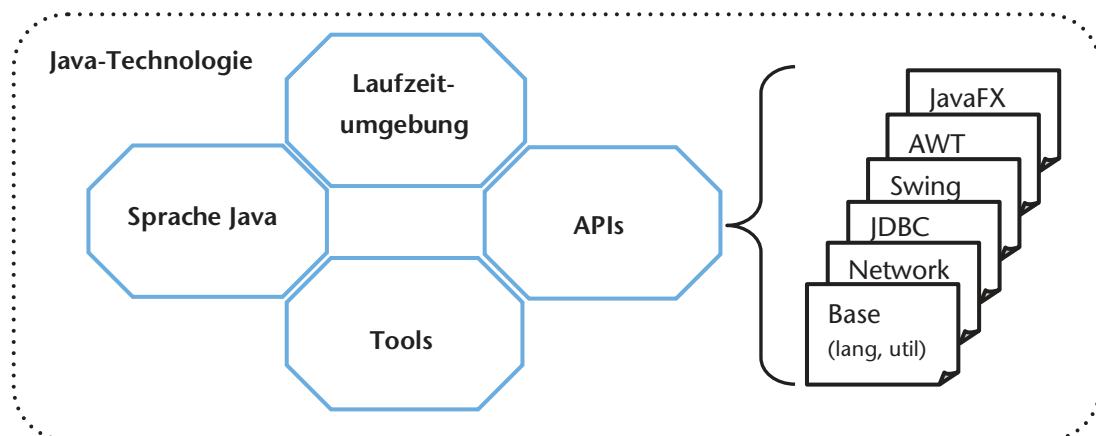
Der große Vorteil von Java ist die Plattformunabhängigkeit. Der Java-Quellcode wird nicht in nativen Code (*Maschinencode*) übersetzt, sondern in eine Art Zwischencode (*Bytecode*). Dieser Bytecode kann dann auf allen Plattformen ausgeführt werden, die über eine entsprechende Java-Laufzeitumgebung verfügen. Die wichtigsten Eigenschaften der Java-Technologie sind:

- **Objektorientierung:** Java ist eine vollständig objektorientierte Sprache.
- **Plattformunabhängigkeit:** Für die meisten Plattformen wurde eine Java-Laufzeitumgebung entwickelt, sodass von einer relativ großen Plattformunabhängigkeit gesprochen werden kann. Es gibt beispielsweise Laufzeitumgebungen für die Betriebssysteme Solaris (ein Unix-System), Linux, Windows und auch für Mac OS X – damit sind die wichtigsten Betriebssysteme bereits abgedeckt.
- **Sicherheit:** Java-Programme laufen kontrolliert in der Java-Laufzeitumgebung ab. Der sogenannte *garbage collector* sorgt beispielsweise für eine sichere Handhabung der Speicherfreigabe.
- **Moderne Anwendungsentwicklung:** Mit Java können moderne verteilte Systeme programmiert werden. Ebenso wird der Zugriff auf Datenbanken durch mächtige Bibliotheken unterstützt.

1.1.3 Die Komponenten der Java-Technologie

Die Java-Technologie besteht aus verschiedenen Komponenten, die dafür sorgen, dass die oben beschriebenen Eigenschaften umgesetzt werden können. Neben der eigentlichen Sprache Java und der Laufzeitumgebung gehören auch verschiedene APIs (*Application Programming Interfaces*) dazu. Das alles wird unter dem *Java Software Development Kit* (kurz **JDK**) zusammengefasst.

Die nächste Abbildung zeigt die Komponenten noch einmal im Überblick.



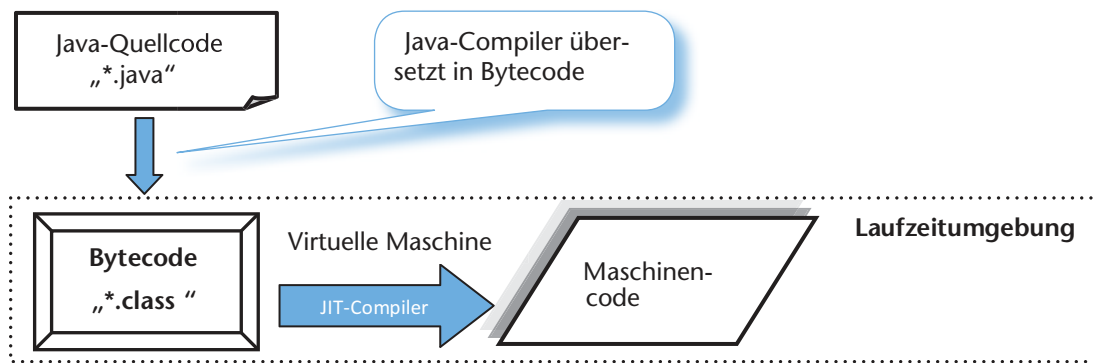
Mit den verschiedenen APIs können die meisten Anwendungen realisiert werden. Die einzelnen APIs sind dabei für die folgenden Bereiche verantwortlich:

- ▶ **Base (lang und util):** Eine Sammlung von Klassen für elementare Funktionalitäten wie Stringbehandlung, mathematische Operationen, formatierte Ausgaben oder Arraybehandlung.
- ▶ **Network:** Mithilfe dieser Klassen kann die Netzwerkprogrammierung umgesetzt werden, beispielsweise über TCP-Verbindungen und den Einsatz von Sockets.
- ▶ **JDBC (Java Database Connectivity):** Mit diesen Klassen werden Datenbanken angesprochen. Sie sind auch die Grundlage für verteilte Anwendungen.
- ▶ **Swing und AWT (Abstract Window Toolkit):** Diese Klassen stellen Komponenten zur Entwicklung von grafischen Benutzeroberflächen zu Verfügung. JavaFX: Mit diesem Framework ist eine moderne GUI-Entwicklung möglich.

1.1.4 Kompilierung von Java-Programmen

Der Java-Quellcode wird nicht mehr direkt in eine ausführbare Datei, sondern in eine Art Zwischencode (Bytecode) übersetzt. Dieser Zwischencode wird dann von der Java-Laufzeitumgebung ausgeführt. Dabei übersetzt eine sogenannte *virtuelle Maschine JVM* (**J**ava **V**irtual **M**achine) den Zwischencode in nativen Code, der dann auf der jeweiligen Plattform ausführbar ist. Die aktuellen virtuellen Maschinen basieren auf intelligenten *Just-in-time-Compilern* (**JIT**) wie dem *HotSpot*, die durch Optimierungen die Java-Programme sehr schnell ausführen können.

Die folgende Abbildung zeigt den schematischen Ablauf einer Kompilierung:

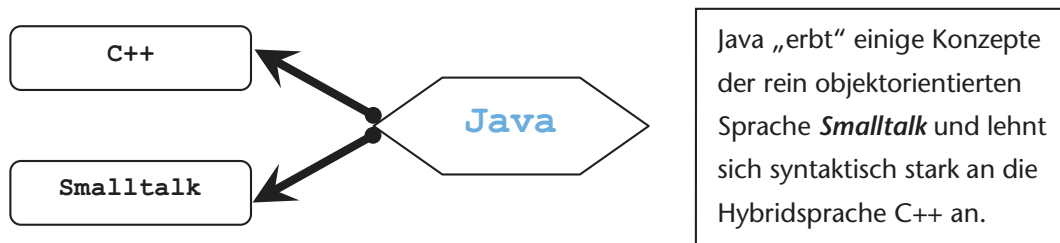


1.2 Die Sprache Java

1.2.1 Entwicklung der Sprache Java

Die Grundlage für Java wurde Anfang der 90er-Jahre durch ein Entwicklerteam um *James Gosling*² gelegt. Das Team arbeitete an einer neuen Technologie, die für beliebige Elektrogeräte eine Programmiersprache mit der entsprechenden Laufzeitumgebung zur Verfügung stellen sollte. Die erste Fassung dieser modernen und objektorientierten Sprache nannte *Gosling OAK*. Aus rechtlichen Gründen musste *OAK* später umbenannt werden. Es wurde der Name **Java** ausgewählt (Java ist nicht nur eine Insel, sondern steht auch für eine aromatische Kaffeesorte und wird in vielen Ländern als Synonym für Kaffee benutzt). Java ist syntaktisch stark an C++ angelehnt, hat aber einige Konzepte von C++ (wie die Mehrfachvererbung) nicht übernommen. Im Vergleich ist Java deshalb etwas einfacher zu erlernen als C++. Die Problematik der Speicherreservierung in C++ wurde in Java durch die Entwicklung eines **garbage collector** gelöst. Die Freigabe von reserviertem Speicher erfolgt dadurch automatisch und entlastet den Entwickler enorm.

Die Web-Programmierung wurde durch Java deutlich vorangebracht (Server-Pages, JavaFX oder Frameworks wie Spring etc.). Durch *Java Enterprise Edition-Anwendungen (Java EE-Anwendungen)* sind moderne **Three-Tier-Anwendungen**³ umsetzbar und mit *Java ME* (Java Platform Micro Edition) ist die moderne App-Entwicklung auf mobilen Endgeräten möglich.



Die Sprache **Smalltalk** war eine der ersten objektorientierten Programmiersprachen (neben *Simula-67*) und verfügte schon über solche Konzepte wie **MVC (Model View Controller)**, das auch in Java sehr erfolgreich eingesetzt wird. Auch das Konzept des **garbage collector** wurde bereits in Smalltalk angewendet.

1.2.2 Eigenschaften der Sprache Java

Die folgenden Eigenschaften zeichnen die Sprache Java aus:

- ▶ Moderne, objektorientierte Sprache
- ▶ „Etwas“ einfacher zu erlernen als C++ (Zeiger müssen nicht verwendet werden)
- ▶ Plattformunabhängiges Konzept
- ▶ Schnelle und effektive Softwareentwicklung (Desktop-Anwendungen, Web-Anwendungen) mit Unterstützung durch mächtige APIs bzw. Klassenbibliotheken
- ▶ Komfortable Anbindung von beliebigen Datenbanken

² James Gosling war von 1984 bis 2010 bei der Firma Sun Microsystems beschäftigt. Die letzten Jahre war er technischer Leiter der Forschungs- und Entwicklungsabteilung.

³ Eine **Three-Tier-Anwendung** verfügt über drei Schichten, die in der Regel auf drei verschiedenen Rechnersystemen installiert sind. Die Schichten bestehen aus einer Clientanwendung, einer Server-Anwendung und einer Datenbank.

1.2.3 Schlüsselworte in Java

Die Sprache Java hat einen Wortschatz von ungefähr 50 reservierten Worten – den sogenannten **Schlüsselworten**. Die Schlüsselworte bilden die Grundlage der Programme in Java. Die folgende Tabelle zeigt die Schlüsselworte von Java:

<code>abstract</code>	<code>continue</code>	<code>for</code>	<code>new</code>	<code>switch</code>
<code>assert</code>	<code>default</code>	<code>goto</code>	<code>package</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>case</code>	<code>enum</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>catch</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>strictfp</code>	<code>volatile</code>
<code>const</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>

Die Schlüsselworte `const` und `goto` können in Java (noch?) nicht benutzt werden. Sie sind reserviert und stehen deshalb nicht als Namen für Variablen oder Objekte zur Verfügung.

Die Bedeutungen der einzelnen Schlüsselworte werden Schritt für Schritt im Laufe dieses Informationsteils erklärt.

1.2.4 Prozedurale, strukturierte und objektorientierte Programmierung unter Java

In der Programmierung können verschiedene Paradigmen⁴ unterschieden werden. Es gibt Sprachen wie C, mit denen beispielsweise nur strukturiert (und auch prozedural) programmiert werden kann. Andere Sprachen wie C++ können sowohl strukturiert (und prozedural) als auch objektorientiert programmiert werden. Die Sprache Java ist hingegen eine rein objektorientierte Sprache. Trotzdem spielt die strukturierte Programmierung auch bei Java eine Rolle, denn innerhalb des objektorientierten Rahmens muss auch strukturiert programmiert werden.

Zum besseren Verständnis werden diese Begriffe kurz erläutert:

Strukturierte Programmierung

Die strukturierte Programmierung zeichnet sich durch Kontrollstrukturen wie die **Auswahl** (IF-ELSE) oder die **Wiederholungen** (FOR, WHILE usw.) aus. Damit erhält ein Programm eine nachvollziehbare Struktur. In den Anfängen der Programmierung war es üblich, Sprunganweisungen (GOTO) in einem Programm zu benutzen. Dadurch wird ein Programm sehr unübersichtlich und fehleranfällig. Strukturierte Programme sind hingegen übersichtlicher und besser wartbar.

Beispiel:

```
FÜR Var := 1 BIS 5 MIT SCHRITTWEITE 1
  SCHREIBE AUF BILDSCHIRM Var
```

```
1
2
3
4
5
```

Das Beispiel zeigt eine Wiederholung in sogenanntem Pseudocode⁵. Dieser Code beschreibt den Ablauf des Programmes, ohne allerdings auf eine spezielle Programmiersprache einzugehen. In dem Beispiel wird eine Variable `Var` so lange um 1 erhöht, bis der Wert 5 erreicht ist. Jeder Wert der Variablen wird dann auf dem Bildschirm ausgegeben.

Prozedurale Programmierung

Die prozedurale Programmierung teilt Programme in kleine Einheiten (Prozeduren oder Funktionen), die für bestimmte Aufgaben verantwortlich sind. Sind diese Prozeduren einmal geschrieben und getestet, können sie immer wieder benutzt werden – das spart Entwicklungszeit und führt auch zu einer besseren Lesbarkeit des Programms.

⁴ Paradigma kommt aus dem Griechischen und heißt so viel wie Muster oder Vorbild.

⁵ Pseudocode ist eine Art Sprache, mit der der Ablauf eines Programmes beschrieben wird. Pseudocode zeichnet sich dadurch aus, dass er näher an der natürlichen Sprache als eine Programmiersprache ist. Ein Programm, das in Pseudocode geschrieben ist, kann problemlos in jede Programmiersprache übersetzt werden.

Beispiel:

```

PROZEDUR Ausgabe
  SCHREIBE AUF BILDSCHIRM "Hallo"
ENDE

FÜR Var := 1 BIS 5 MIT SCHRITTWEITE 1
  AUFRUF Ausgabe

```

```

Hallo
Hallo
Hallo
Hallo
Hallo

```

Das Beispiel in Pseudocode zeigt eine Prozedur mit dem Namen *Ausgabe*. Diese Prozedur hat eine Anweisung, die das Wort „Hallo“ auf den Bildschirm schreibt. Die bereits bekannte Wiederholung aus dem Beispiel vorher läuft dann 5-mal und ruft jedes Mal die Prozedur *Ausgabe* auf. Damit steht 5-mal das Wort „Hallo“ auf dem Bildschirm.

Objektorientierte Programmierung

Die objektorientierte Programmierung möchte Objekte der realen Welt in einem Programm abbilden. Damit sollen Problemstellungen aus beliebigen Bereichen (Geschäftsprozesse, wissenschaftliche Untersuchungen usw.) geeigneter als mit den anderen Programmierparadigmen in Programme umgesetzt werden können.

Im Mittelpunkt der objektorientierten Programmierung steht die **Klasse**, aus der dann konkrete Objekte gebildet werden. Diese Objekte haben bestimmte Eigenschaften (Attribute) und sogenannte Methoden, mit denen diese Eigenschaften beispielsweise verändert werden können.

Beispiel:

```

KLASSE Kunde
  Name
  Telefon
ENDE

BILDE OBJEKT K1 VON Kunde
K1.Name    := "Maier"
K1.Telefon := "123456"

SCHREIBE AUF BILDSCHIRM K1.Name und K1.Telefon

```

```

Maier
123456

```

In dem Beispiel wird eine Klasse *Kunde* definiert. Von dieser Klasse können dann konkrete Objekte wie *K1* (für Kunde 1) gebildet werden. Die Eigenschaften des Objektes (Name, Telefon) können mit Werten belegt werden. In diesem Beispiel erhält das Objekt *K1* den Namen „Maier“ und die Telefonnummer „123456“. Anschließend werden Name und Telefon des Objektes auf den Bildschirm geschrieben.

1.2.5 Bestandteile eines Java-Programms

Ein Java-Programm besteht aus einer Folge von endlich vielen und eindeutigen Anweisungen⁶, die mithilfe der Schlüsselwörter und selbst gewählter Namen für bestimmte Elemente wie Klassen oder Objekte gebildet werden. Zusätzlich kann ein Java-Programm auch Anweisungen enthalten, die nicht zum eigentlichen Programm gehören, aber die Erstellung des Programms steuern. Das folgende Beispiel zeigt ein erstes einfaches Java-Programm:

```
package java_it_berufe;
```

Damit wird ein eigenes Paket mit dem Namen „java_it_berufe“ definiert.

```
import java.io.*;
```

Der import-Befehl sorgt für die Einbindung von Bibliotheken, in diesem Fall der Input-Output-Bibliotheken.

⁶ Eine endliche Folge von eindeutigen Anweisungen an den Computer nennt man **Algorithmus**.

Eine eigene Klasse
wird definiert.

```
public class Java_IT_Berufe {
```

Die main-Methode – entspricht klas-
sischerweise dem „Hauptprogramm“.

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

```
        System.out.println("Hallo Java!");
```

Ein Java-Befehl zur Ausgabe einer
Zeichenkette auf dem Bildschirm

```
    }  
}
```

In dem obigen Beispiel wird deutlich, dass auch ein einfaches Java-Programm schon einen relativ komplizierten Aufbau hat. Das liegt daran, dass Java eine vollständig objektorientierte Sprache ist und deshalb immer auch eine Klasse definiert werden muss. Dieser Aufbau wird nun in den folgenden Kapiteln Schritt für Schritt erläutert.

2 Das erste Java-Programm

2.1 Ein Java-Projekt anlegen

Die integrierte Entwicklungsumgebung *Apache NetBeans IDE 10.0* ist eine komfortable Umgebung, um Java-Programme zu entwickeln. Besonders erfreulich ist der Umstand, dass die Umgebung kostenfrei im Internet bereit steht. Ein Java-Programm besteht aus einer oder mehreren Quellcode-Dateien. Diese Dateien werden in einem Projekt organisiert. *NetBeans* bietet eine Vielzahl von Projekten an. Im Folgenden sind die wichtigsten aufgeführt:

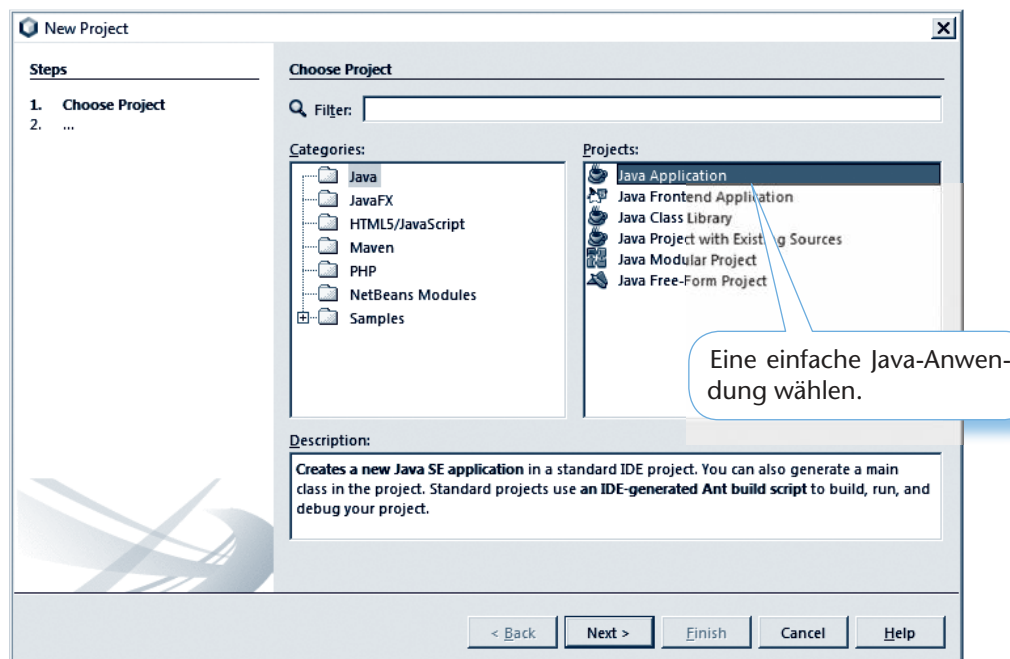
- Java-Anwendung (erstellt verschiedene Basis-Anwendungen)
- JavaFX-Anwendung (erstellt Vorlagen für JavaFX-GUI-Anwendungen)
- HTML5/Javascript (erstellt Vorlagen für Web-Anwendungen auf HTML5-Basis)
- Maven (ein Build-Tool: dient zur Verwaltung / Standardisierung von Java-Anwendung)

In diesem Buch sind hauptsächlich zwei Projektformen von Bedeutung: die **Java-Anwendung** und die **JavaFX-Anwendung**.

Die Java-Anwendung ist ausreichend, um eine einfache Ein- und Ausgabemöglichkeit für die ersten Java-Programme zu haben. In den späteren Kapiteln wird dann die JavaFX-Anwendung verwendet. Die einfache Java-Anwendung ist natürlich nicht so ansprechend wie ein Programm mit graphischer Benutzeroberfläche, aber, um die Grundlagen der Sprache zu erlernen, völlig ausreichend.

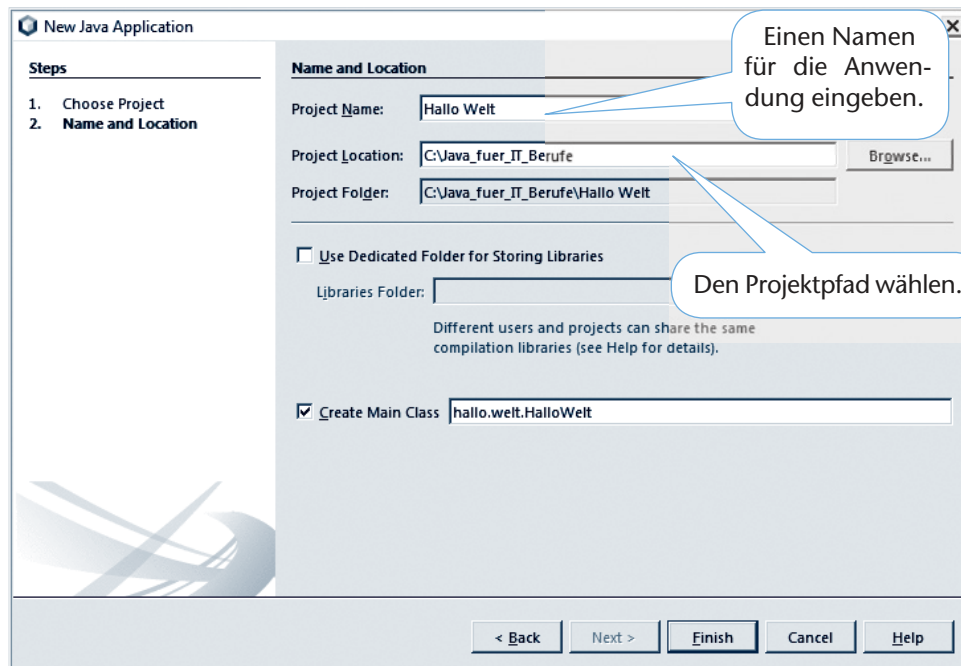
Anlegen eines neuen Projektes:

- Starten Sie *Apache NetBeans IDE 10.0*.
- Wählen Sie den Menüpunkt Datei → Neues Projekt (oder *File → New Project*¹).



¹ Hier wird die englische Version von *Apache Netbeans* genutzt, deshalb werden sowohl deutsche als auch englische Befehle angegeben.

Nach dem Bestätigen mit „Weiter“ oder „Next“ werden die Eingabe eines Namens und die Wahl eines Projektpfades erwartet.



Anschließend kann das durch einen Klick auf „Fertigstellen“ oder „Finish“ das Projekt erstellt werden.

