EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

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
Düsselberger 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

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

E-Mail: Hardy@DirkHardy.de

Im Sommer 2019

Verlag Europa-Lehrmittel

E-Mail: Info@Europa-Lehrmittel.de

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*.

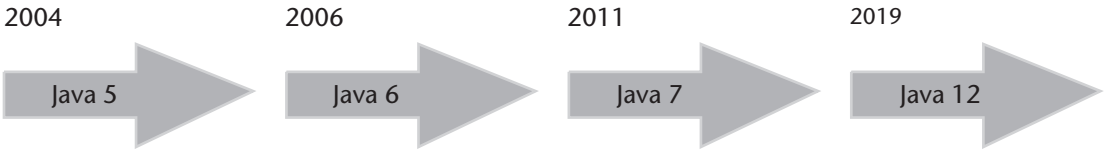
Die folgende Grafik zeigt den zeitlichen Verlauf der Java-Technologie-Entwicklung:

1992 Eine Gruppe von Entwicklern um den Programmierer *James Gosling* entwirft die objektorientierte Programmiersprache *OAK*, den Vorläufer von Java.

1995 *Sun Microsystems* stellt die erste Version von Java vor. Parallel dazu wird ein eigener Browser entwickelt (*HotJava*).

1996 Java wird in den Browser *Netscape Navigator* integriert und verbreitet sich dadurch rasant. Ebenso wird das ***JDK*** (*Java Development Kit*) veröffentlicht.

1998 Java 2 wird vorgestellt. Das JDK heißt nun ***J2SDK*** (*Java 2 Software Development Kit*).



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

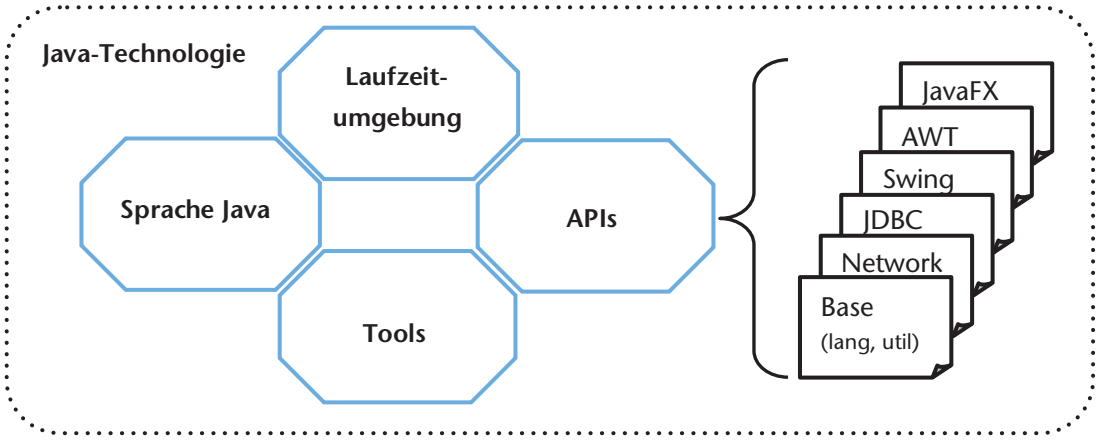
Der große Vorteil von Java ist die Plattformunabhängigkeit. Der Java-Quellcode wird nicht in nativen Code (*Maschinencode*) übersetzt, sondern in eine Art Zwischencode (*Bytecode*). Dieser Bytecode kann dann auf allen Plattformen ausgeführt werden, die über eine entsprechende Java-Laufzeitumgebung verfügen. 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 (***A**pplication **P**rogramming **I**nterfaces*) 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.

Beispiel:

Die konstanten Werte erhalten eine zusätzliche Information (in diesem Beispiel eine Zeichenkette).

Die konstanten Werte werden ebenfalls mit Kommata getrennt, aber der letzte Wert wird mit einem Semikolon abgeschlossen.

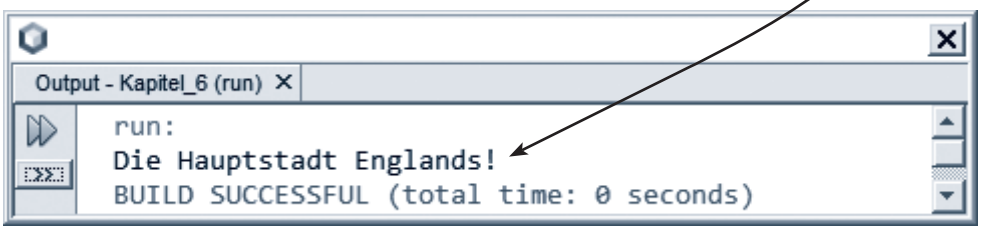
Der private Konstruktor speichert die zusätzliche Information zu dem konstanten Wert in einem privaten Attribut.

Durch die öffentliche Methode `gibBeschreibung()` ist die Zusatzinformation zu dem Wert abrufbar.

Der private Konstruktor sorgt für die Zuweisung der Zusatzinformation für den Wert `LONDON`.

```
enum Hauptstaedte {

  BERLIN ("Die Hauptstadt Deutschlands!"),
  LONDON ("Die Hauptstadt Englands!"),
  PARIS ("Die Hauptstadt Frankreichs!");


  private String beschreibung;


  private Hauptstaedte(String beschreibung) {
        this.beschreibung = beschreibung;
  }

  public String gibBeschreibung() {
        return beschreibung;
  }
}

public class WeitereElemente {
  public static void main(String[] args) {

        Hauptstaedte stadt;


        stadt = Hauptstaedte.LONDON;

        System.out.println(stadt.gibBeschreibung());
}
```

Nach dem Starten sieht die Bildschirmausgabe dann so aus:



Beispiel:

Optional kann ein Interface `public` sein (wie eine Klasse).

Ein Interface wird angelegt.

```
interface Ausgebbar {

    public abstract void ausgeben(Object obj);
}

interface Eingebbar {

    public abstract Object eingeben() throws IOException ;

}
```

Ein weiteres Interface wird angelegt.

Ein Interface deklariert beliebig viele abstrakte Methoden. Die Modifizierer `public` und `abstract` können auch weggelassen werden.

Mit dem Schlüsselwort `implements` werden beliebig viele Schnittstellen implementiert.

```
class Konsolenprogramm implements Ausgebbar, Eingebbar {
```

Die Methode `ausgeben()` der Schnittstelle `Ausgebbar` muss nun implementiert werden.

```
    @Override
    public void ausgeben(Object obj){
        System.out.println("Inhalt: " + obj.toString());

    }
```

Ebenso muss die Methode `eingeben()` der Schnittstelle `Eingebbar` implementiert werden.

```
    @Override
    public Object eingeben() throws IOException {

        BufferedReader einlesen  =
        new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Bitte eingeben: ");
        return (Object)(einlesen.readLine());
    }
}
public class Schnittstellen {
    public static void main(String[] args) throws IOException {
```

Ein Objekt der Klasse `Konsolenprogramm` wird instanziiert.

```
        Konsolenprogramm con = new Konsolenprogramm();

        //Alternativ:  Eingebbar con = new Konsolenprogramm();
        //Alternativ:  Ausgebbar con = new Konsolenprogramm();

        con.ausgeben(con.eingeben());

    }
}
```

Das Objekt nutzt die implementierten Methoden.

Von Schnittstellen können keine Objekte instanziiert werden, aber sie können als Verweise auf Objekte von Klassen dienen, die das Interface implementiert haben.

Beispiel:

Es wird ein zweidimensionales Array angelegt.

Die erste Dimension festlegen – entspricht der Anzahl der Zeilen einer Tabelle.

Die zweite Dimension festlegen – entspricht der Anzahl der Spalten einer Tabelle.

```
int [][] tabelle = new int[ 3 ][ 4 ];
```

Entsprechend der Dimension werden leere Klammern gesetzt.

Das zweidimensionale Array ist als Tabelle vorstellbar. Der Zugriff auf ein Element des Arrays erfolgt dann mit einem **Doppelindex**. Beispielsweise ist das Element aus Zeile 0 und Spalte 1 so ansprechbar:

```
tabelle[ 0 ][ 1 ] = 3
```

	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0		**3**		
Zeile 1				
Zeile 2				

Wie gewohnt kann ein mehrdimensionales Array auch direkt initialisiert werden. Damit sind die Dimensionen ebenso festgelegt:

```
int [][] tabelle = {   {5,3,7,2} ,
                       {8,6,9,1} ,
                       {2,7,3,4}     };
```

Drei Zeilen mit jeweils vier Spalten

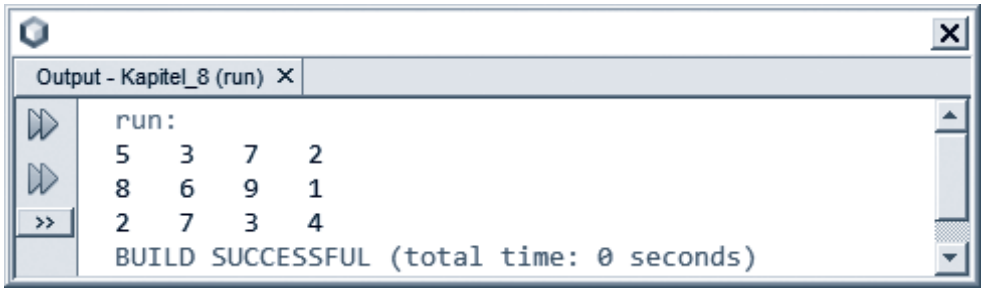
	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	5	3	7	2
Zeile 1	8	6	9	1
Zeile 2	2	7	3	4

Die Dimensionen können ebenfalls mit dem konstanten Attribut `length` abgefragt werden, wie das folgende Beispiel zeigt:

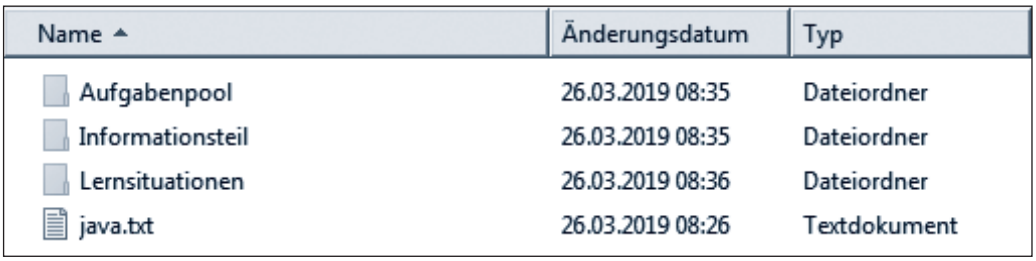
```
for (int i=0; i < tabelle.length; i++) {
      for (int j=0; j < tabelle[0].length; j++) {
            System.out.print(tabelle[i][j] + "   ");
      }
      System.out.println();
}
```

Die Methode `print()` erzeugt keinen Zeilenumbruch nach der Ausgabe.

Die Bildschirmausgabe zeigt das zweidimensionale Array:



Zum Vergleich hier ein Ausschnitt aus der Explorer-Ansicht:

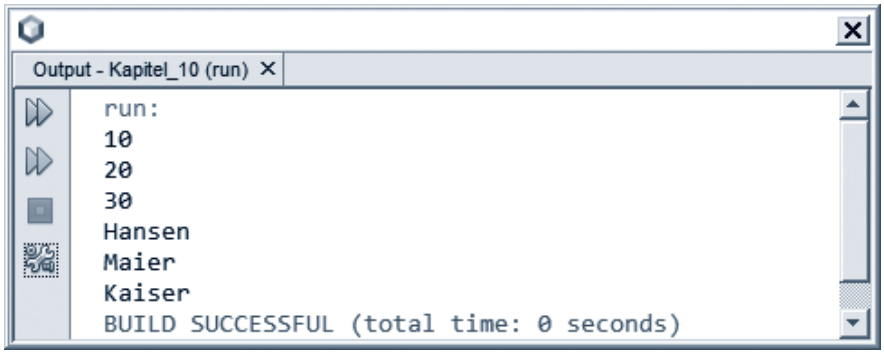


Hinweis:

Alle Datei- und Verzeichnisoperationen können fehlschlagen, weil beispielsweise die angegebene Datei nicht vorhanden ist oder ein Verzeichnis wegen mangelnder Rechte nicht angelegt werden darf. Deshalb sollten Dateioperationen immer mithilfe des Exception Handling abgesichert werden, denn dadurch können Fehler kontrolliert abgefangen werden. Das Exception Handling wird im nächsten Kapitel ausführlich behandelt.

```
for (int i : intListe) {
        System.out.println(i);
}

for (Person p : personenListe) {
        System.out.println(p);
}
```

Nach dem Starten sieht die Bildschirmausgabe so aus:



Hinweis:

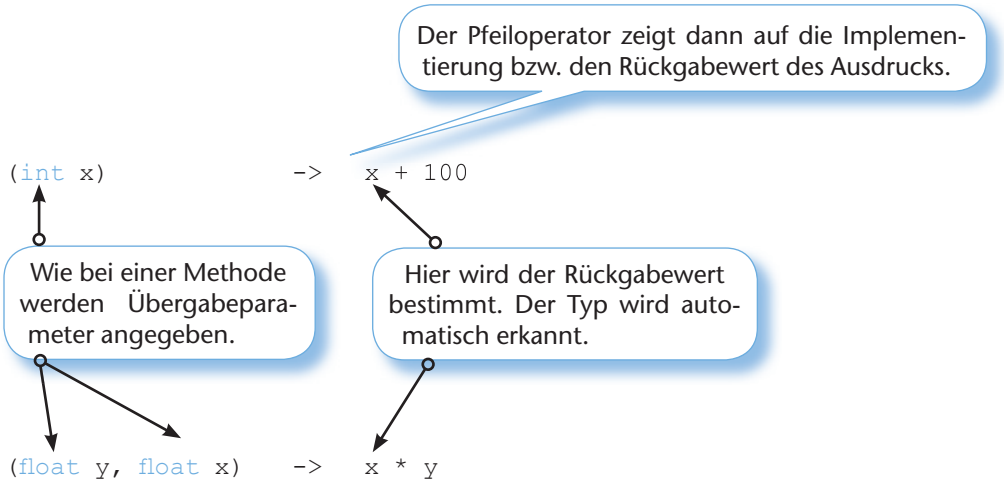
Durch die Festlegung des Typs bei der Instanziierung der Liste sind dann auch nur Elemente dieses Typs zugelassen. Der folgende Versuch, ein beliebiges Element hinzuzufügen, scheitert dann auch:

```
intListe.add(new Person("Kaiser"));
```

FEHLER:
Der `intListe` können nur Integerwerte zugewiesen werden!

10.3 Lambda-Ausdrücke

Aus der funktionalen Programmierung wurde in vielen Programmiersprachen das Konzept der Lambda-Ausdrücke übernommen, das auf dem Prinzip der anonymen Methoden basiert. Anonyme Methoden haben keinen Namen, verfügen aber wie normale Methoden über einen Rückgabewert und über Übergabeparameter. Diese Methoden werden in der Regel dort definiert, wo sie gerade gebraucht werden. Das macht den Einsatz dieser Methoden sehr flexibel. Durch den Einsatz einer speziellen Syntax entstehen dann aus diesen anonymen Methoden die so genannten Lambda-Ausdrücke. Die folgenden Beispiele zeigen die typische Verwendung solcher Lambda-Ausdrücke:



```
    public int GetAnzahl() {
        return zaehler;
    }

    private double Berechnung() {
        return (werte[9] - werte[0]) * konstante;
    }


    public static void main(String[] args) {

        int[] werte_1 = { 2, 4, 3, 7, 8, 3, 2, 6, 8, 5 };
        int[] werte_2 = { 3, 5, 2, 1, 8, 6, 4, 9, 7, 10 };

        Klassendiagramm obj_1 = new Klassendiagramm(werte_1);

        obj_1 = new Klassendiagramm(werte_2);


        Klassendiagramm obj_2 = new Klassendiagramm(werte_2);
        obj_1.AusgabeWerteSortiert();
        obj_2.AusgabeBerechnung();
        Klassendiagramm.AusgabeAnzahlObjekte();
    }
}
```

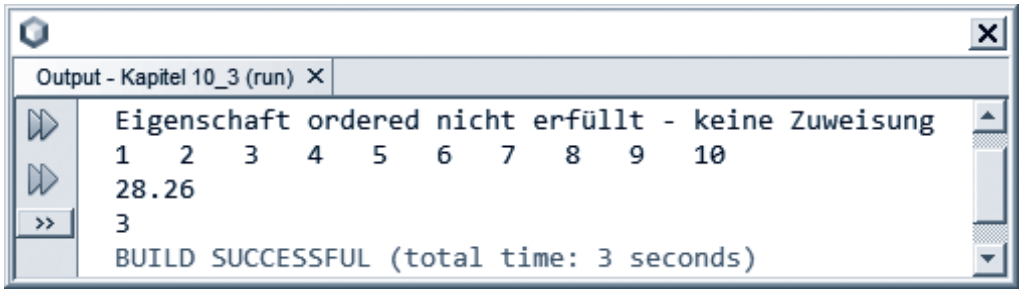
Private Methode zur Berechnung

Instanziierung von Objekten. Das erste Objekt mit `werte_1`-Array wird an dieser Stelle verweislos und vom ***garbage-collector*** gelöscht.

Instanziierung eines weiteren Objektes

Aufruf der statischen Methode

Nach dem Starten erscheint folgende Ausgabe auf dem Bildschirm:



Das erste Array (`werte_1`) fällt durch die „`ordered`"-Prüfung und wird nicht zugewiesen. Die zweite Zuweisung (`werte_2`) ist korrekt und das Attribut wird sortiert ausgegeben. Die Berechnung erfolgt ebenfalls korrekt. Es fällt allerdings auf, dass die Methode `AusgabeAnzahlObjekte` als Anzahl 3 ausgibt, obwohl nur zwei Objekte einen Verweis haben. Das liegt daran, dass der garbage-collector noch nicht aufgeräumt hat. Mit den folgenden Anweisungen wird der garbage-collector explizit aufgerufen (`System.gc()`) und nach einer kurzen Wartezeit (`Thread.sleep(3000)`) hat er dann auch seinen Dienst erledigt und es sind nur noch zwei Objekte im Speicher.

11.3.2 Einen Clientbereich hinzufügen

Die obigen Beispiele zeigen, wie in den Clientbereich des Fensters geschrieben werden kann. Das Problem bei dieser Vorgehensweise sind die zugrunde liegenden Koordinaten. Sie beginnen nicht in der linken oberen Ecke des Clientbereiches, sondern beziehen sich auf das komplette Fenster. Das führt zu Komplikationen bei der genauen Berechnung von Ausgaben, die beispielsweise von der linken oberen Ecke des Clientbereiches starten sollen. Aus diesem Grund ist es sinnvoll, eine Art neuen Clientbereich zu definieren und in das Fenster einzubetten. Dieser Clientbereich basiert auf der Klasse `Canvas` (deutsch: Leinwand). Diese Klasse bietet eine Art leere Fläche, die zum Zeichnen oder auch zur Ereignisbehandlung genutzt werden kann. Dem Fenster muss diese neue Leinwand nur hinzugefügt werden und anschließend füllt sie den ganzen Clientbereich aus. Die `paint`-Methode wird dann wie gewohnt überschrieben.

Das folgende Beispiel zeigt die Definition einer `Clientbereich` -Klasse innerhalb der Fenster-Klasse. Sie hat den Vorteil, dass die eingebettete Klasse nur in der umgebenden Klasse genutzt werden kann. Es wäre allerdings auch möglich, die `Clientbereich` -Klasse außerhalb zu definieren. Damit könnte sie vielen Fensterklassen zur Verfügung stehen.

```
class Fenster extends Frame {

      public Fenster() {
            this.setSize(200, 100);
            this.setLocation(400, 300);

            this.add (new Clientbereich());
      }



      class Clientbereich extends Canvas {

            @Override
            public void paint(Graphics g) {


                  int schrifthoehe = this.getFont().getSize();
                  g.drawString("Hallo Canvas", 0, schrifthoehe);
            }
      }
}
```

Mit der Methode `add` wird der Clientbereich hinzugefügt und in das Fenster eingebettet.

Die `Clientbereich`-Klasse als **innere Klasse** definieren und von der `Canvas`-Klasse ableiten

Die `paint`-Methode wie gewohnt überschreiben

Über die Methode `getSize` die exakte Höhe der Schriftart erhalten

Die Ausgabe der Zeichenkette genau in die linke obere Ecke des Clientbereiches setzen

Nach dem Starten wird der Text dann in dem Clientbereich ausgegeben:

Der neue Clientbereich!

11.3.3 Einfache Grafikausgabe

Mit den Methoden der Klasse `Graphics` können im Clientbereich beliebige grafische Elemente gezeichnet werden. Neben dem Zeichnen von einfachen grafischen Elementen wie Linien und Rechtecken können auch komplexe Formen wie Polygone oder Kreisausschnitte gezeichnet werden. Das folgende Beispiel zeigt die Verwendung einiger einfacher Methoden der Klasse `Graphics`:

```
        DefaultMutableTreeNode unterknoten =
              new DefaultMutableTreeNode("Unterknoten");
        wurzel.add(unterknoten);

        DefaultMutableTreeNode unterUnterknoten =
              new DefaultMutableTreeNode("Unter-Unterknoten");
        unterknoten.add(unterUnterknoten);

        baumansicht.addTreeSelectionListener(

              new TreeSelectionListener() {

                 @Override
                 public void valueChanged(TreeSelectionEvent e) {

                    String pfad =
                       baumansicht.getSelectionPath().toString();

                    selektion.setText(pfad);

                    baumansicht.expandPath(e.getNewLeadSelectionPath());
                    }
                 } );

        this.getContentPane().add(baumansicht);
        this.getContentPane().add(selektion);
    }
}
```

Einen Ereignisempfänger hinzufügen

Einen `TreeSelection-Listener` anlegen

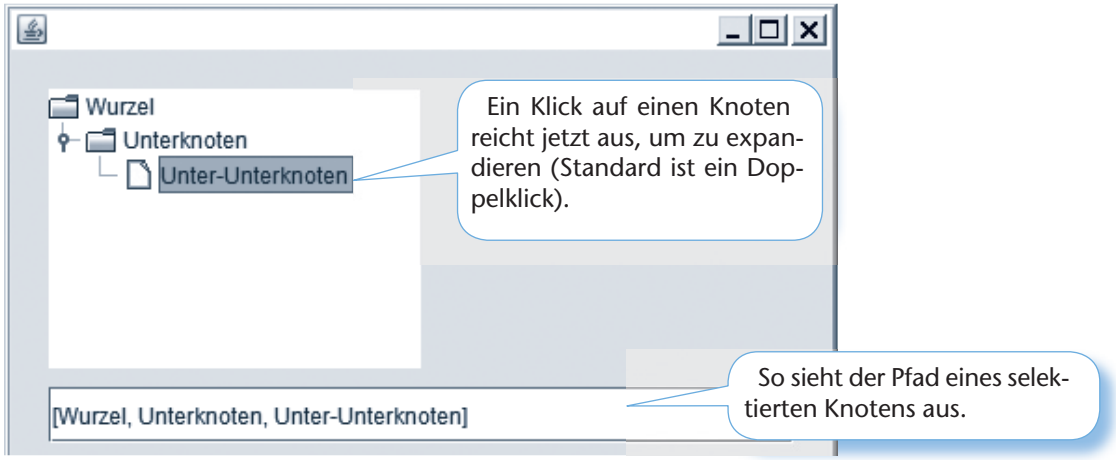
Die Ereignismethode `valueChanged` implementieren

Mit der Methode `getSelec-tionPath` den Pfad des selek-tierten Knotens wählen

Den Knoten über den Pfad expandieren lassen

Alternativ kann der selektierte Pfad auch mit der Methode des Übergabeparameters `e` vom Typ `TreeSelectionEvent` gewählt werden.

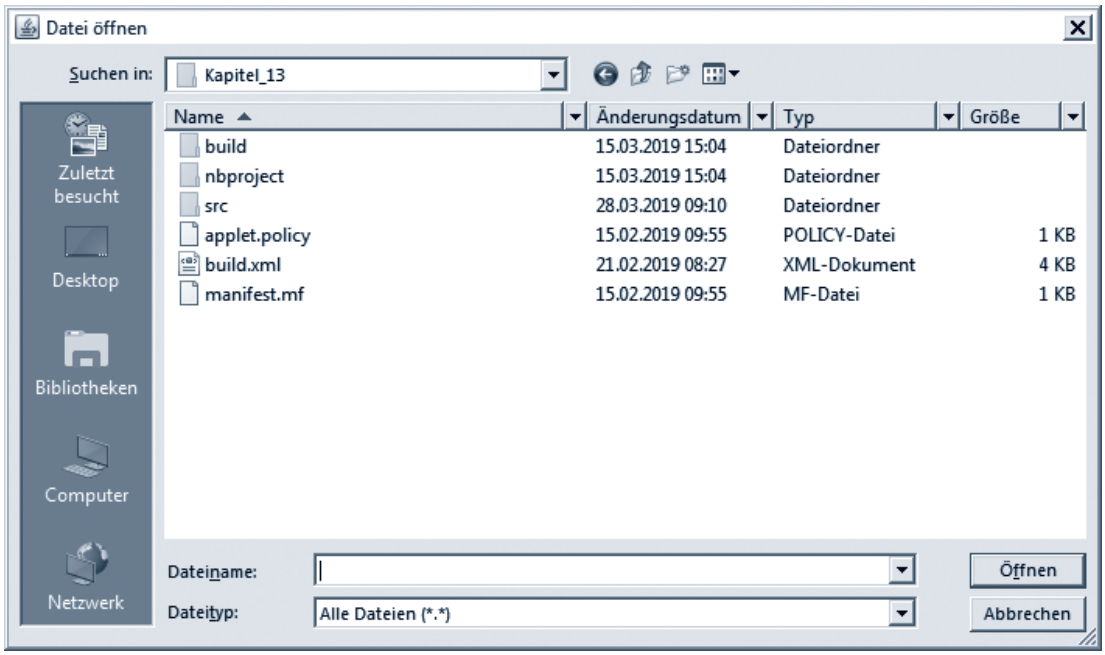
Nach dem Starten erscheint folgendes Fenster:



Ein Klick auf einen Knoten reicht jetzt aus, um zu expandieren (Standard ist ein Doppelklick).

So sieht der Pfad eines selektierten Knotens aus.

Nach dem Starten wird sofort der Dialog angezeigt (hier der Windows-Standarddialog):



Hinweis:

Mit den Methoden **`getDirectory`** und **`getFile`** können das gewählte Verzeichnis und die gewählte Datei des Benutzers abgefragt werden.

Variante 2: Swing-Klassen-Dialoge

Einen Verweis vom Typ `JFileChooser` anlegen. Die anderen Dialoge werden mit statischen Methoden realisiert.

```
class StandardSwingDialoge extends JFrame {

        private JFileChooser dateiDialog;

        public StandardSwingDialoge() {
```

Die statische Methode `showConfirmDialog` öffnet einen Dialog, der mit verschiedenen Optionen versehen werden kann (hier werden drei Optionen angezeigt). Dazu werden Konstanten aus der Klasse `JOptionPane` benutzt (hier `YES_NO_CANCEL_OPTION`). Die Methode gibt einen Wert zurück, der ebenfalls mit den Konstanten abgefragt werden kann.

```
        if      (    JOptionPane.showConfirmDialog(
                     this,
                     "Treffen Sie eine Auswahl:",
                     "Auswahl",
                     JOptionPane.YES_NO_CANCEL_OPTION)

                     ==  JOptionPane.YES_OPTION
                ) {
                     JOptionPane.showMessageDialog(
                     this,
                     "Sie haben 'JA' gewählt");

        }
```

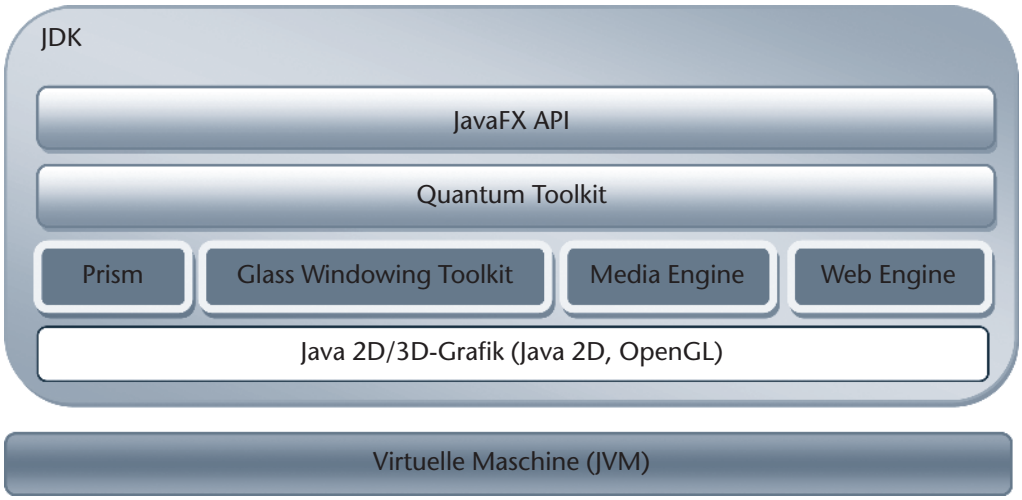
Die statische Methode `showMessageDialog` öffnet einen einfachen Dialog mit einer Meldung und einem OK-Button. Das ist vergleichbar mit einer `MessageBox` in anderen Programmiersprachen.

14 JavaFX-Anwendungen entwickeln

14.1 Grundkonzept von JavaFX

Die Entwicklung von GUI-Anwendungen wurde in den vorherigen Kapiteln mit den Frameworks AWT und Swing dargestellt. Für einfache Anwendungen ist diese Art immer noch aktuell, aber die Herausforderungen von modernen plattformunabhängigen Anwendungen sind mit diesen Mitteln nicht immer zu bewältigen. Eine Antwort darauf ist das ***JavaFX***-Framework, mit dem solche modernen Anwendungen (vor allem im Bereich Medien) entwickelt werden können. Im Prinzip ist es vergleichbar mit der .NET-Technologie und dem Wechsel von Windows-Forms-Anwendungen zu WPF-Anwendungen. ***Oracle*** hat ***JavaFX*** deshalb auch komplett neu entwickelt und ***JavaFX*** arbeitet nicht ohne Anpassungen mit den bekannten AWT/Swing-Klassen zusammen. ***JavaFX*** wird inzwischen von dem Open-Source-Projekt ***OpenJFX*** weiterentwickelt.

Die folgende Grafik zeigt den Aufbau von JavaFX:



Die Grundlage jeder JavaFX-Anwendung ist ebenfalls die virtuelle Maschine (JVM), die für die Ausführung des Java-Bytecodes verantwortlich ist.

Das Java Development Kit (JDK) stellt die Komponenten und Tools für die Entwicklung der JavaFX-Anwendungen bereit. Dabei sind bereits bekannte Komponenten wie ***Java 2D*** und die neue spezifischen Komponenten für JavaFX wie das ***Quantum Toolkit*** enthalten.

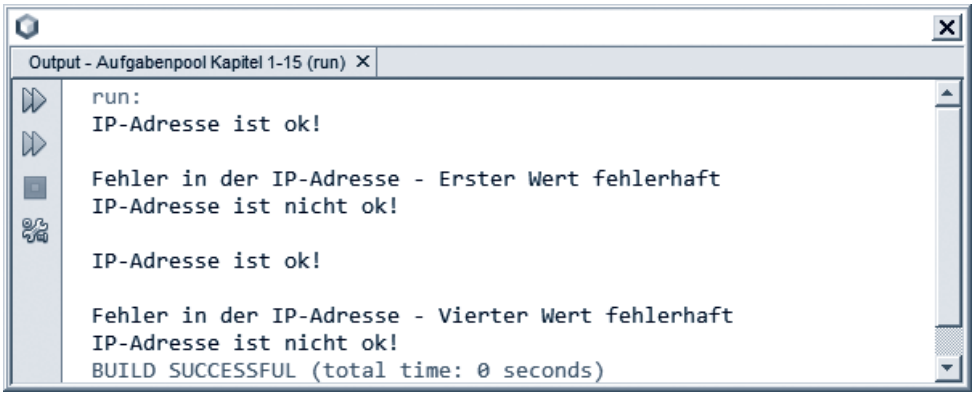
Die spezifischen Komponenten werden nun kurz erläutert:

- ***Prism*** ist eine neue Rendering Engine, die die Grafikhardware direkt anspricht. Ist das nicht möglich, so werden die Anforderungen softwaretechnisch mit Java 2D simuliert.
- Das ***Glass Windowing Toolkit*** stellt Funktionalitäten wie die Fensterverwaltung oder Ereignisverwaltung zur Verfügung. Es dient als Schnittstelle, um JavaFX mit dem Betriebssystem zu verbinden.
- Audio und Video werden von der ***Media Engine*** unterstützt.
- Die ***Web Engine*** bettet Web-Inhalte in JavaFX-Applikationen ein.
- Das ***Quantum Toolkit*** verbindet alle Komponenten (***Prism, Glass Windowing Toolkit, Media Engine und Web Engine***) und dient als übergeordnete Schnittstelle für die ***JavaFX API***.

Testprogramm:

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

      IPAdresse iP = new IPAdresse();


      if (iP.setIP("12.111.222.123")==true)
            System.out.println("IP-Adresse ist ok!");
      else System.out.println("IP-Adresse ist nicht ok!");

      System.out.println();
      if (iP.setIP("..0.000") == true)
            System.out.println("IP-Adresse ist ok!");
      else System.out.println("IP-Adresse ist nicht ok!");

      System.out.println();
      if (iP.setIP("012.1.10.000") == true)
            System.out.println("IP-Adresse ist ok!");
      else System.out.println("IP-Adresse ist nicht ok!");

      System.out.println();
      if (iP.setIP("123.12.0.") == true)
            System.out.println("IP-Adresse ist ok!");
      else System.out.println("IP-Adresse ist nicht ok!");
}
```

Nach dem Starten sieht die Bildschirmausgabe so aus:



Hinweis:

Nutzen Sie intensiv solche Methoden wie `substring` oder `indexOf` der Klasse `String`.

Aufgabe 6.4

Für den Mathematikunterricht einer Berufsschule soll eine Klasse entwickelt werden, die einige nützliche Methoden und ein Attribut zur Verfügung stellt. Implementieren Sie dazu die folgenden statischen Methoden und ein statisches Attribut in einer Klasse `Mathematik`:

- ***Potenz***-Methode: Diese Methode soll eine Variable vom Typ `double` mit einem angegebenen Exponenten potenzieren. Die berechnete Potenz soll zurückgegeben werden.
- ***Fakultät***-Methode: Diese Methode soll einen Integerwert übernehmen, die Fakultät berechnen und zurückgeben.

Hinweis:

Die Fakultät einer natürlichen Zahl ist das Produkt aller Zahlen von 1 bis zu dieser Zahl.

Beispiel: 5! (! heißt Fakultät) = 1 · 2 · 3 · 4 · 5 = 120

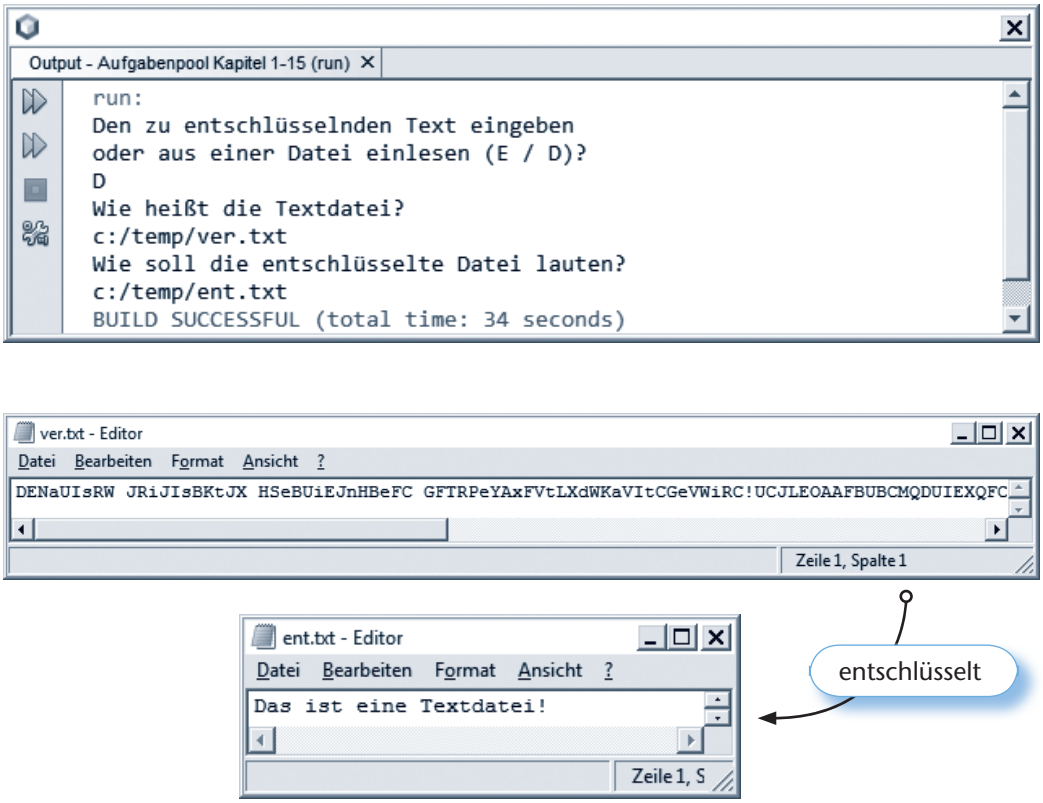
Hinweis:

Mit der folgenden Programmzeile können zufällige Werte erzeugt werden.

```
int zufallsZahl = (int) (Math.random() * 100);
```

Die Methode `random` liefert eine nichtnegative ganze Zufallszahl zwischen 0 (inkl.) und 1 (exkl.). Durch die Multiplikation mit 100 und die Konvertierung in eine Integerzahl liegt der Zufallswert zwischen 0 und 99.

Beispiel: Eine Datei entschlüsseln



Aufgabe 9.5

Entwerfen Sie eine Klasse `Vokabel` in Java, die Vokabelpaare (Deutsch – Englisch und Deutsch – Spanisch) darstellen und abfragen soll. Die Klasse soll die Vokabeln in zwei `Hashmaps` speichern. Die Vokabeln sollen im Konstruktor der Klasse aus der Datei *„Vokabel.txt"* eingelesen werden (siehe Hinweise).

Für die Behandlung von Vokabeln sollen folgende Methoden zur Verfügung stehen:

- **Ausgeben von Vokabeln**
  `public void ausgeben()`

  Die Methode zeigt alle Vokabeln formatiert auf dem Bildschirm an.

- **Abfragen von Vokabeln**
  `public int abfragen (String sprache)`

  Es wird eine Vokabelabfrage gestartet. Je nach Übergabeparameter wird eine Abfrage von Deutsch – Englisch oder Deutsch – Spanisch gestartet. Anschließend soll per Zufallsgenerator die Vokabelabfrage stattfinden. Hierbei werden die richtigen Antworten gezählt und zurückgegeben. Jede Vokabel darf nur einmal bzw. muss genau einmal abgefragt werden.

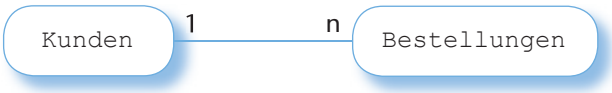
Schreiben Sie zusätzlich in der Hauptmethode ein Auswahlmenü. Der Benutzer soll auswählen können, ob die Vokabeln in Deutsch – Englisch oder Deutsch – Spanisch abgefragt werden.

Die zugrunde liegenden Tabellen sehen so aus:

Kundentabelle:

	ID	Name
	Filtern	Filtern
1	1	Maier
2	3	Kaiser
3	4	Franzen
4	5	Knoblauch
5	6	Laufer
6	7	Koenig

Beziehung der Tabellen:



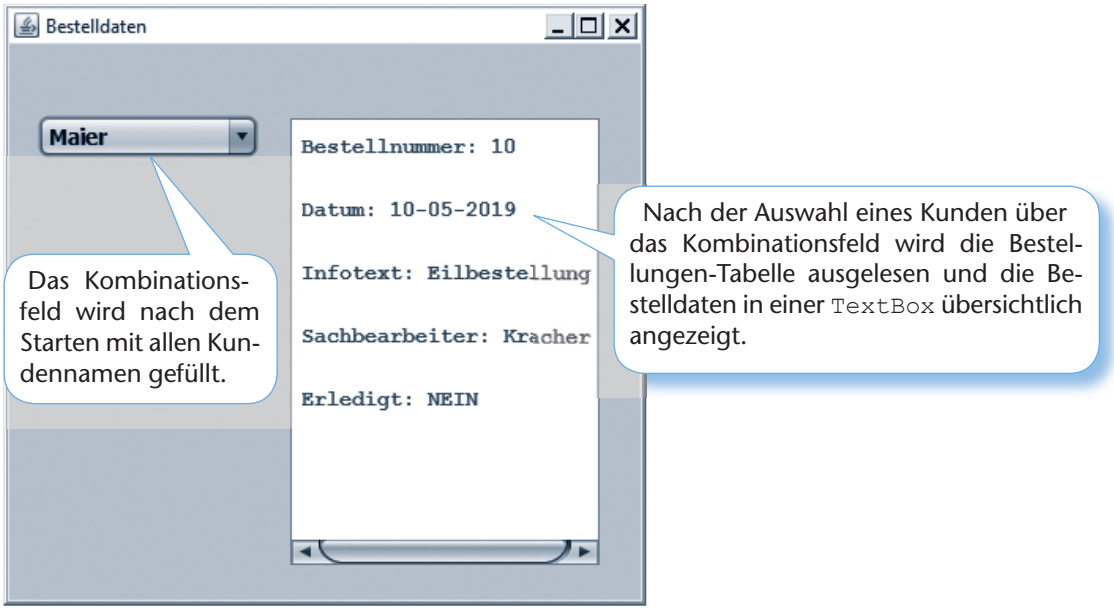
Bestellungen-Tabelle:

	Kunden_ID	Bestellnummer	Datum	Infotext	Sachbearbeiter	Erledigt
	Filtern	Filtern	Filtern	Filtern	Filtern	Filtern
1	1	10	10-05-2019	Eilbestellung	Kracher	true
2	3	11	10-06-2019	gefährliche Fr...	Klauber	true
3	4	12	10-07-2019	guter Kunde	Hütter	false

Die Bestellungen-Tabelle hat einen Fremdschüssel `Kunden_ID`, der die „1:n“ – Beziehung der beiden Tabellen umsetzt.

Aufgabenstellung:

Legen Sie die beiden Tabellen in einer geeigneten Datenbank an (beispielsweise *SQLite*) und füllen Sie die Tabellen mit den entsprechenden Werten. Implementieren Sie dann eine Java-Anwendung, die auf die Datenbank zugreift und die Tabellen ausliest. Die Oberfläche der Anwendung sollte so aussehen:



Lernsituation 1:

Erstellen einer Präsentation mit Hintergrundinformationen zu der Sprache Java (in Deutsch oder Englisch)

Ausgangssituation:

Sie haben die Ausbildung zum Fachinformatiker bei der mittelständischen Softwareentwicklungsfirma **ProSource** begonnen. Unter anderem führt die Firma Inhouse-Schulungen in verschiedenen IT-Bereichen durch.

Es ist eine Schulung in der Programmiersprache Java geplant. Da die Java-Entwickler der Firma unter Zeitdruck stehen, ist die Vorbereitung der Schulung problematisch. Sie erhalten deshalb den Auftrag, den einführenden Informationsteil der Schulung zu gestalten. Dieser Teil soll ungefähr 15 Minuten in Anspruch nehmen. Neben historischen Daten sollen die interessanten Aspekte der Sprache ansprechend vorgestellt werden – beispielsweise die Einordnung der Sprache Java bezüglich strukturierter und objektorientierter Sprachen. Die Präsentation soll auch in den ausländischen Niederlassungen der Firma genutzt und deshalb auch parallel in englischer Sprache vorbereitet werden.

Arbeitsschritte in Einzel- oder Partnerarbeit:

Planung:

Legen Sie das Präsentationsmittel fest (PowerPoint-Präsentation, Handouts usw.).

Denken Sie über den Umfang der Präsentation nach (Zeitrahmen 15 Minuten).

Informieren Sie sich über die Hintergründe von Java mithilfe des Informationsteils dieses Buches und weiterer Quellen wie dem Internet.

Wenn es möglich ist, dann arbeiten Sie fächerübergreifend im Deutsch- und/oder Englischunterricht an der Präsentation weiter.

Durchführung:

Gestalten Sie die Folien ansprechend, ohne sie zu überfrachten. Formulieren Sie die Folientexte kurz und aussagekräftig. Halten Sie die Präsentation entweder in Deutsch oder in Englisch, wenn der fächerübergreifende Unterricht stattfinden konnte.

Kontrolle:

Führen Sie die Präsentation vor Ihrem Partner oder einer anderen Lerngruppe vor. Der Partner bzw. die Zuhörer beobachten die Präsentation unter Einbeziehung des unten angegebenen Kriterienkatalogs, der im Anschluss die Grundlage für die kritische Auseinandersetzung bietet.