Java für IT-Berufe

Mit App-Entwicklung

Bearbeitet von
Dirk Hardy

1. Auflage 2015. Taschenbuch. 283 S. Paperback
ISBN 978 3 8085 8560 3
Format (B x L): 21 x 29,7 cm
Gewicht: 739 g

schnell und portofrei erhältlich bei

beck-shop.de
DIE FACHBUCHHANDLUNG

Die Online-Fachbuchhandlung beck-shop.de ist spezialisiert auf Fachbücher, insbesondere Recht, Steuern und Wirtschaft. Im Sortiment finden Sie alle Medien (Bücher, Zeitschriften, CDs, eBooks, etc.) aller Verlage. Ergänzt wird das Programm durch Services wie Neuerscheinungsdienst oder Zusammenstellungen von Büchern zu Sonderpreisen. Der Shop führt mehr als 8 Millionen Produkte.

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("      - Applets");
        System.out.println("      - DB-Anbindung");
        System.out.println("      - Android Apps");

        System.out.println("Aufgabenpool");

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

2. Auflage

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

Europa-Nr.: 85535

Verfasser:
Dirk Hardy, 46049 Oberhausen

Verlagslektorat:
Alexander Barth

2. Auflage 2015

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 978-3-8085-8560-3

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

©2015 by Verlag Europa-Lehrmittel, Nourney, Vollmer GmbH & Co. KG, 42781 Haan-Gruiten
http://www.europa-lehrmittel.de
Satz: Reemers Publishing Services GmbH, 47799 Krefeld
Umschlaggestaltung: braunwerbeagentur, 42477 Radevormwald
Druck: Konrad Triltsch Print und digitale Medien GmbH, 97199 Ochsenfurt-Hohestadt

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, die Anbindung von Datenbanken und die **Entwicklung von Android Apps (neu in der 2. Auflage)** 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.

Als Entwicklungswerkzeug wird in diesem Buch **NetBeans IDE 8.0.1** 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 2015

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*.

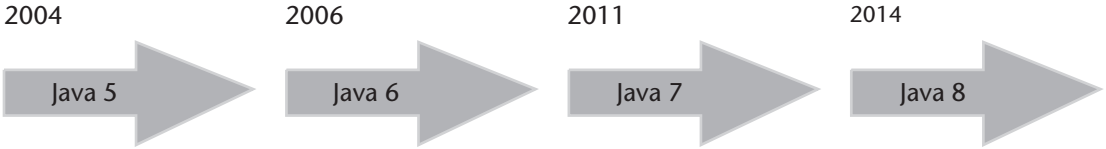
Die folgende Grafik zeigt den zeitlichen Verlauf der Java-Technologie-Entwicklung:

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

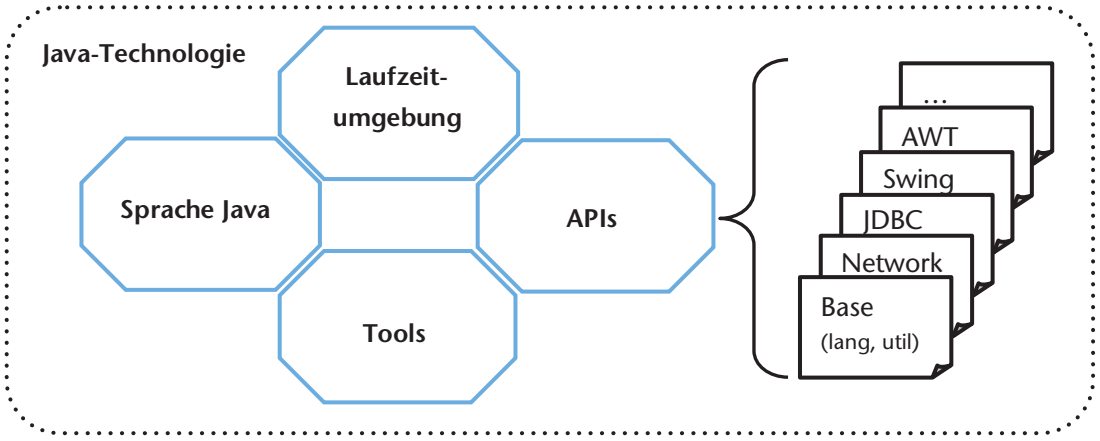
Der große Vorteil von Java ist die Plattformunabhängigkeit. Der Java-Quellcode wird nicht in nativen Code (*Maschinencode*) übersetzt, sondern in eine Art Zwischencode (*Bytecode*). Dieser Bytecode kann dann auf allen Plattformen ausgeführt werden, die über eine entsprechende Java-Laufzeitumgebung verfügen.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

1.1.4 Kompilierung von Java-Programmen

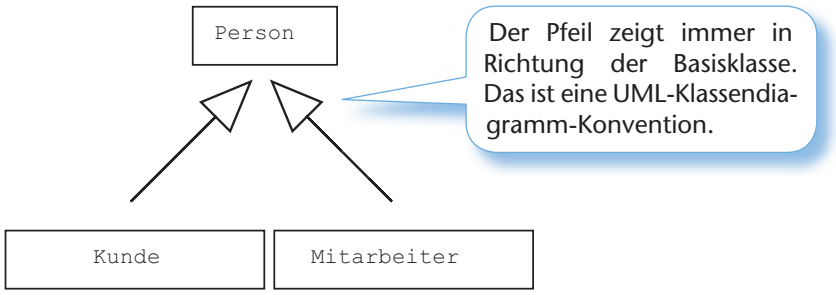
Der Java-Quellcode wird nicht mehr direkt in eine ausführbare Datei, sondern in eine Art Zwischencode (Bytecode) übersetzt. Dieser Zwischencode wird dann von der Java-Laufzeitumgebung ausgeführt. Dabei übersetzt eine sogenannte *virtuelle Maschine* ***JVM*** (**J**ava **V**irtual **M**achine) den Zwischencode 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.

7 Vererbung in Java

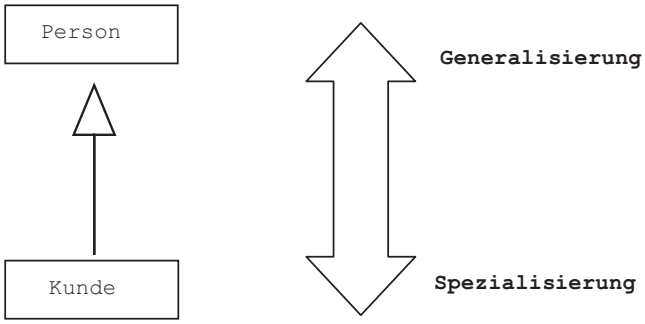
Das Konzept der Vererbung ist ein zentrales Thema in der OOP. Durch Vererbung können einerseits Situationen aus der „realen" Welt besser in die Programmiersprache umgesetzt werden, andererseits kann bereits existierender Programmcode (in Form von Klassen) wiederverwendet werden. Dadurch ergeben sich mehr Effizienz und Sicherheit in der Softwareentwicklung durch bereits vorhandenen und geprüften Programmcode. Bei der Vererbung spricht man von einer sogenannten **Ist-Beziehung**.

Beispiel:

Die Basisklasse `Person` vererbt an die Klassen `Kunde` und `Mitarbeiter`. Der Kunde bzw. der Mitarbeiter sind eine Person (Ist-Beziehung). Der Kundenklasse bzw. Mitarbeiterklasse stehen nun alle Elemente der Basisklasse zur Verfügung (mit gewissen Einschränkungen, siehe dazu später). Wenn beispielsweise die Personenklasse ein Attribut `name` hat, so erbt sowohl die Kunden- als auch die Mitarbeiterklasse dieses Attribut.



Die Klassen `Kunde` bzw. `Mitarbeiter` sind eine spezielle Klasse `Person`. Die Klasse `Person` ist eine Verallgemeinerung der Klasse `Kunde` bzw. `Mitarbeiter`. Aus diesem Grund spricht man auch von Generalisierung und Spezialisierung.



Hinweis:

Die Klasse, die vererbt (`Person`), wird in der Regel Basisklasse, Oberklasse oder auch Superklasse genannt. Die Klasse, die erbt (`Kunde`), wird abgeleitete Klasse, Unterklasse oder auch Subklasse genannt.

7.1 Die Vererbung in Java

7.1.1 Die einfache Vererbung

Solange eine Klasse immer nur von einer anderen Klasse erbt, spricht man von **einfacher Vererbung**. Die einfache Vererbung bedeutet aber nicht, dass nicht mehrere Klassen hintereinander erben können. Die folgenden Beispiele sind einfache Vererbungen.

Nach dem Starten sieht die Ausgabe dann so aus:

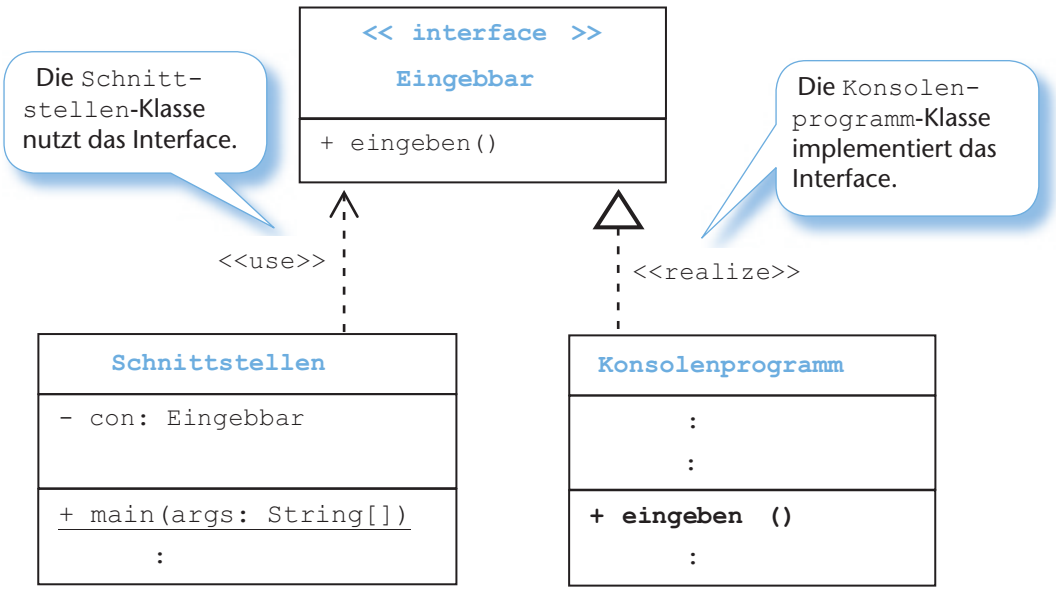
Hinweise:

Die Namen für Interfaces werden in der Regel so gewählt, dass sie ausdrücken, für welchen Zweck das Interface implementiert werden kann. In dem obigen Beispiel wurden die Interfaces `Eingebbar` und `Ausgebbar` genannt. Damit sollen Objekte von Klassen, die die genannten Interfaces implementieren, über diese Eigenschaften verfügen – also etwas einlesen und ausgeben können. Die Java-Bibliotheken bieten eine Vielzahl von Interfaces an, die für die entsprechenden Zwecke implementiert werden können. Beim Thema Arrays und Sortieren wird beispielsweise das Interface `Comparable` noch einmal näher beleuchtet.

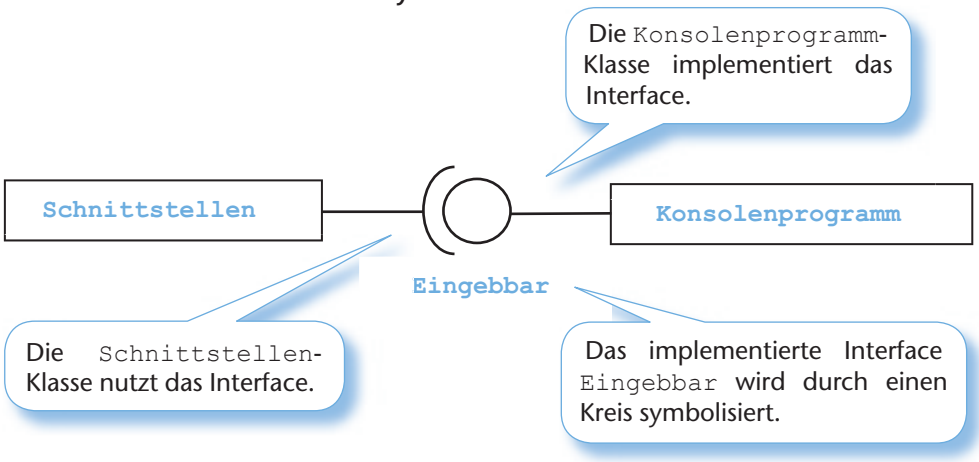
Darstellung von Interfaces in der UML

Die Darstellung der Vererbung von Klassen in der UML wurde zu Beginn des Kapitels vorgestellt. Die Implementierung von Interfaces erfolgt in der Regel mit anderer Symbolik. Das folgende Beispiel zeigt die UML-Darstellung für das obige Beispiel:

Beispiel:



Alternativ mit der ***Ball and Socket*-Symbolik:**



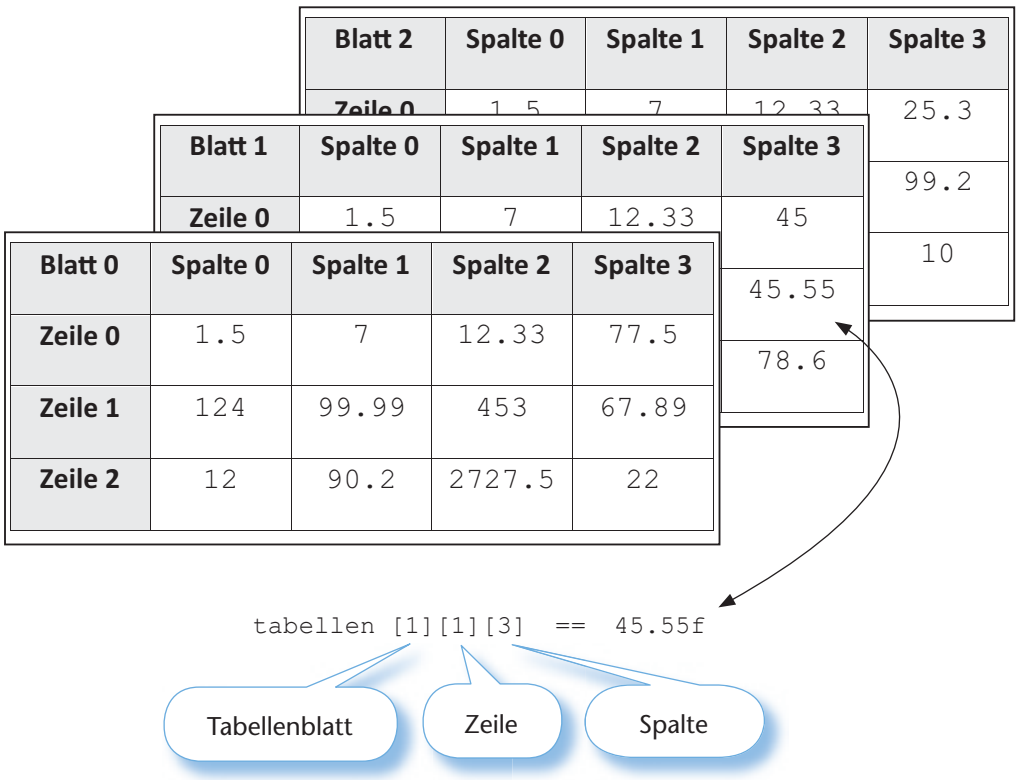
Dreidimensionale Arrays kann man sich als eine Sammlung von Tabellenblättern vorstellen, die hintereinander angeordnet sind.

Beispiel:

Es wird ein dreidimensionales Array angelegt.

```
float  [][][] tabellen = new float [3] [3] [4];
```

Das dreidimensionale Array ist dann so vorstellbar:



Nach der dritten Dimension hört die menschliche Vorstellungskraft auf. Die mehrdimensionalen Arrays mit mehr als drei Dimensionen sind dann auch nicht mehr so konkret vorstellbar wie beispielsweise die Tabellen, aber dennoch sind sinnvolle Einsätze dieser Arrays denkbar.

Beispiel eines fünfdimensionalen Arrays:

Für ein psychologisches Experiment werden drei unterschiedliche Gruppen mit jeweils 15 Probanden festgelegt. Jeder Proband erhält 10 Fragebögen mit jeweils 12 Fragen. Jede Frage hat 3 Antworten, die angekreuzt werden können. Ein Array, das diesen Versuch widerspiegelt, könnte so aussehen:

```
boolean [][][][][] experiment = new boolean [3][15][10][12][3];
```

Es soll nun die Antwort des 5. Probanden aus Gruppe 2 für den 7. Fragebogen und die 4. Frage gespeichert werden. Die drei Antworten waren: `Ja , Nein , Ja`.

Der Einfachheit halber wird ein „Ja" mit dem booleschen Wert `true`, ein „Nein" mit dem booleschen Wert `false` gespeichert.

```
experiment [1][4][6][3][0] = true;
experiment [1][4][6][3][1] = false;
experiment [1][4][6][3][2] = true;
```

Arrays von Arrays

Die mehrdimensionalen Arrays sind eigentlich nichts anderes als Arrays von Arrays mit konstanter Größe. In Java ist es aber möglich, die zweite oder höhere Dimension nicht festzulegen, sondern variabel zu gestalten. Damit erhält man ein mehrdimensionales Array, welches verschieden große *Unterarrays* besitzt. Ein einfaches Beispiel soll das Prinzip verdeutlichen.

10 Fortgeschrittene Themen in Java

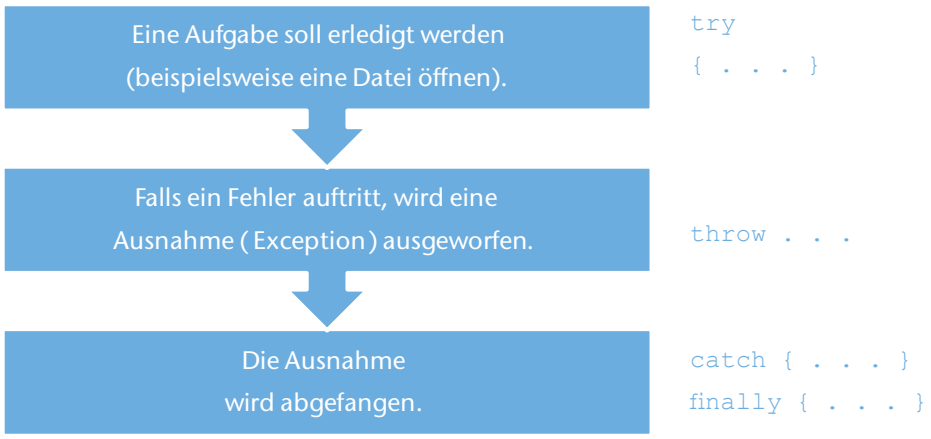
10.1 Ausnahmen – Exceptions

Das Abfangen von Fehlern ist eine wichtige Aufgabe in der Programmierung. Oftmals können Fehler durch Rückgabewerte von Methoden identifiziert werden. Der Nachteil dieser Vorgehensweise ist, dass es dem Programmierer selbst überlassen bleibt, ob er die Rückgabewerte bzw. Fehler auswertet und darauf reagiert oder nicht.

Mögliche Fehlerquellen sind:

- Über den reservierten Bereich eines Arrays schreiten
- Division durch Null
- Eingabe von nicht erwarteten Zeichen über die Tastatur
- Fehler bei Dateioperationen
- Fehler bei Datenbankzugriffen

Die Ausnahmebehandlung in Java hilft dabei, diese Probleme zu bewältigen. Dabei wird die Fehlerbehandlung vom eigentlichen Programmcode separiert. Die folgende Abbildung zeigt den schematischen Ablauf einer Ausnahmebehandlung:



10.1.1 Versuchen und Auffangen (try and catch)

Die Ausnahmebehandlung startet mit dem sogenannten `try`-Block. Innerhalb dieses Blocks steht der Programmcode, der möglicherweise einen Fehler verursachen kann – deshalb das Schlüsselwort `try` für einen Versuch. In dem folgenden Beispiel soll eine Zahl über die Tastatur eingelesen werden. Wenn der Benutzer allerdings Buchstaben statt Zahlen eingibt, dann wird eine Ausnahme „geworfen".

Beispiel:

Die bereits bekannte Initialisierung der System-Ausnahmebehandlung für IO-Fehler

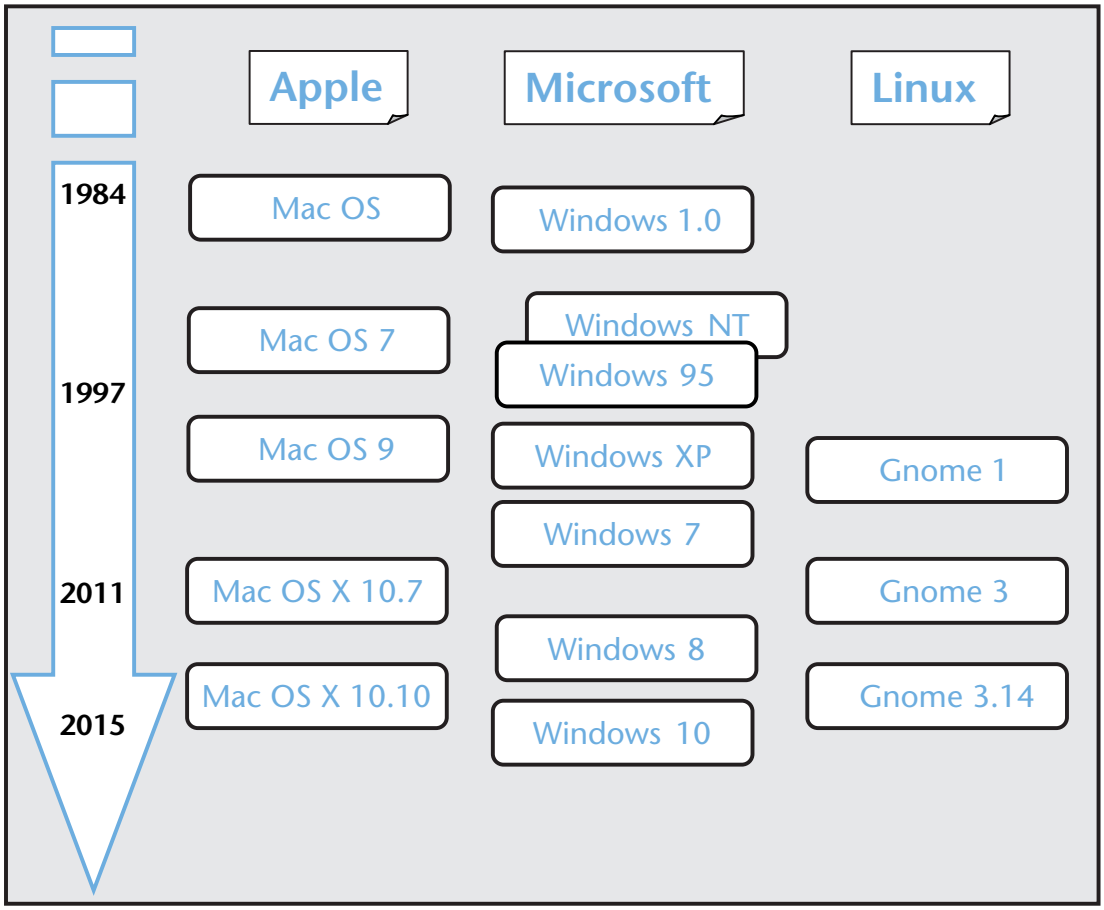
```
public class Ausnahmen {
   public static void main(String[] args) throws IOException {


   int x;
   BufferedReader einlesen  =     new BufferedReader(new
                                  InputStreamReader(System.in));
```

11 GUI-Programmierung mit dem Abstract Window Toolkit AWT

11.1 GUI-Programmierung

11.1.1 Historische Entwicklung der GUI-Programmierung

In den Anfängen des Computers fand die Kommunikation mit dem Computer über Lochkarten statt. In den 60er-Jahren kamen dann die ersten PC-ähnlichen Computer auf den Markt, die über eine Tastatur und eine Bildschirmanzeige verfügten, welche manchmal nur drei Zeilen anzeigen konnte. Später wurden die Computer dann mit Röhrenmonitoren ausgestattet, die eine Anzeige von mehreren Zeilen und Spalten erlaubten. Diese Anzeige ähnelt der Konsolenausgabe, die bislang in diesem Buch genutzt wurde. In den 70er-Jahren wurden dann die ersten Versuche einer grafischen Benutzeroberfläche **GUI** (**G**raphical **U**ser **I**nterface) gestartet. In den meisten Fällen scheiterten diese GUIs an mangelnder Rechenleistung der Computer. Erst in den 80er-Jahren wurde mit dem *Apple Macintosh* ein Computer vorgestellt, der sowohl erschwinglich war als auch über genug Rechenleistung verfügte, um die grafische Benutzeroberfläche *Mac OS* zu unterstützen. Diese GUI hatte viele Aspekte, die dem heutigen Benutzer sehr vertraut sind – vor allem auf die Benutzung einer Maus kann heutzutage nicht mehr verzichtet werden. Die Firma Microsoft stellte kurz danach die erste Version von Windows vor, die aber erst einige Jahre später mit den Versionen *Windows 95* und der parallel entwickelten *Windows NT-Linie* zu großem Erfolg kam. Die folgende Abbildung zeigt die Entwicklung einiger grafischer Benutzeroberflächen in zeitlichem Zusammenhang (inkl. der Linux-GUI *Gnome*):

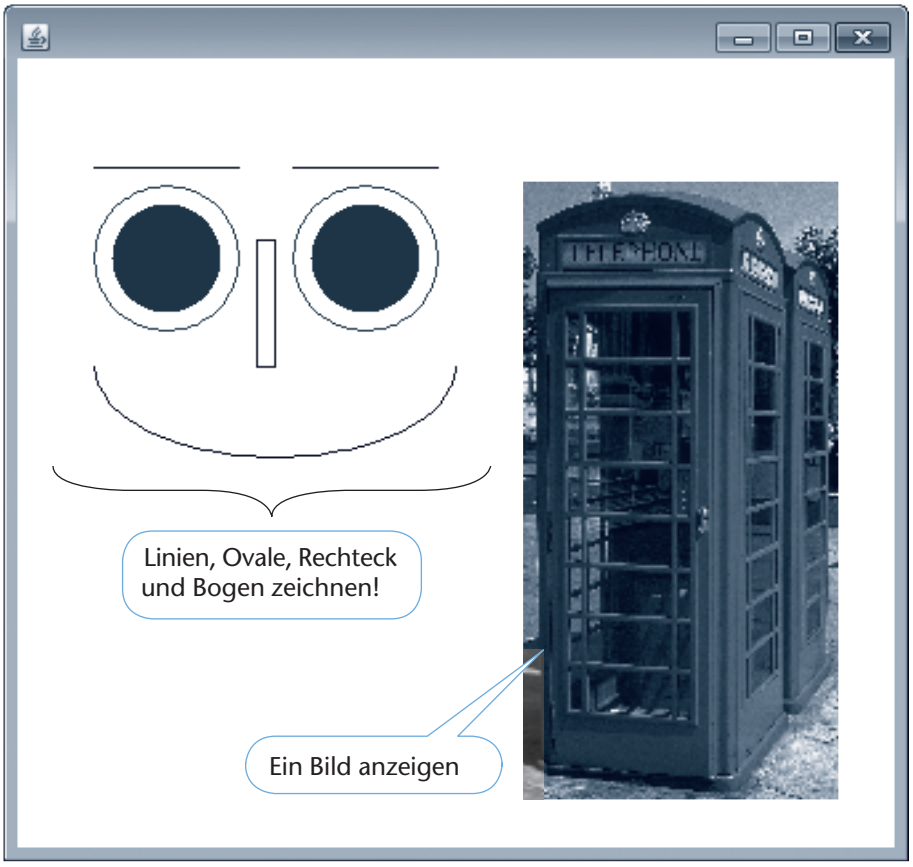


Mit der Methode `getImage` eines `Toolkit`-Objektes wird ein Bild geladen.

```
        Image bild = getToolkit().getImage("c:/temp/telefon.jpg");
        g.drawImage(bild, 300, 100, this);

    }
}
}
```

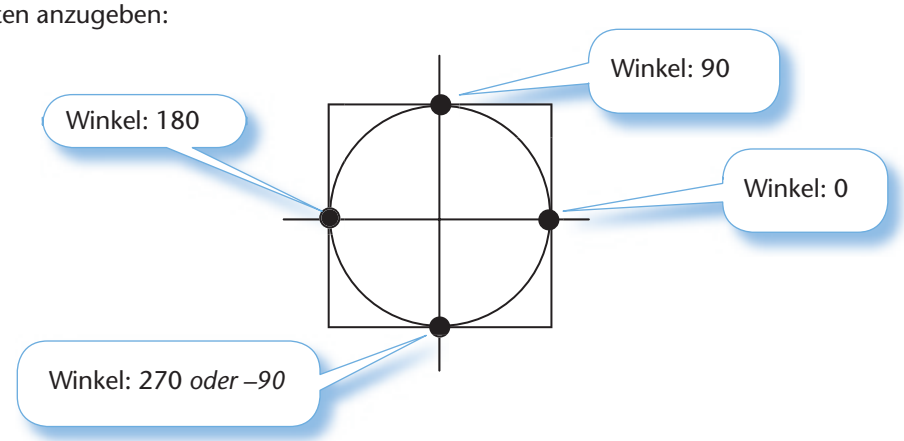
Mit der Methode `drawImage` das Bild anzeigen

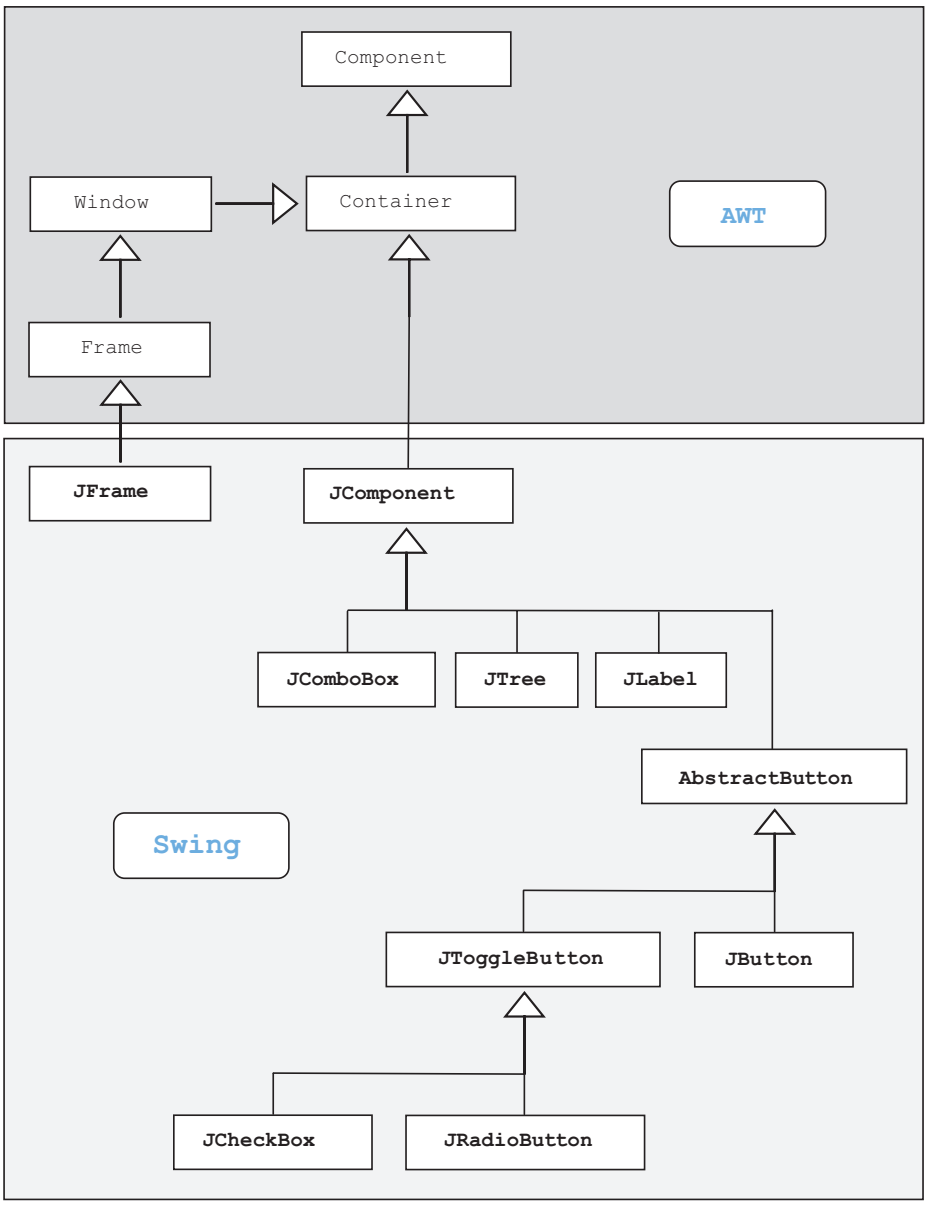
Nach dem Starten sieht das Programm so aus:



Hinweis:

Bei der Methode für den Bogen werden ein Startwinkel und ein Distanzwinkel angegeben, die den Bogen innerhalb des Rechtecks definieren. Der Startwinkel beginnt dabei in der Mitte der rechten Seite des umgebenden Rechtecks mit Null. Nach unten sind die Winkel mit negativen Werten anzugeben:





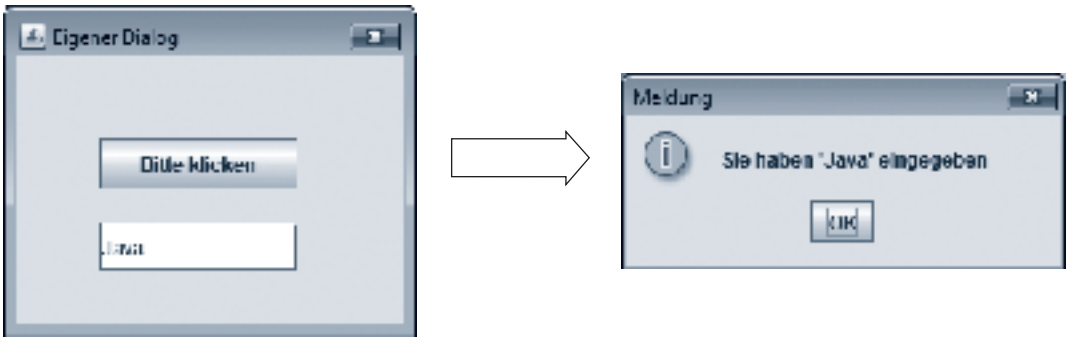
12.2.2 Swing-Steuerelemente

Das Swing-Paket verfügt über deutlich mehr Steuerelemente als das AWT. Vor allem solche Elemente wie die Baumansicht (`JTree`) oder eine Tabellenansicht (`JTable`) sollten bei der heutigen Gestaltung von Oberflächen nicht fehlen. Einige wichtige Steuerelemente sind in der folgenden Tabelle aufgeführt:

Steuerelement	Aussehen	Beschreibung
`JButton`	Button	Auf dieses Steuerelement kann der Benutzer klicken und löst damit in der Regel eine Aktion aus.
`JCheckBox`	CheckBox	Dieses Steuerelement dient zum Setzen eines kleinen Hakens, um beispielsweise eine Option zu wählen.
`JRadioButton`	RadioButton	Dieses Steuerelement dient zum Setzen einer Option und wird meistens in einer Gruppe benutzt. In der Gruppe kann immer nur ein Element gesetzt sein.
`JComboBox`	ComboBox	Dieses Steuerelement ist ein Kombinationsfeld aus einer Drop-down-Liste und einem Eingabefeld.

```
        :
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Nach dem Starten erscheint die Swing-Variante:

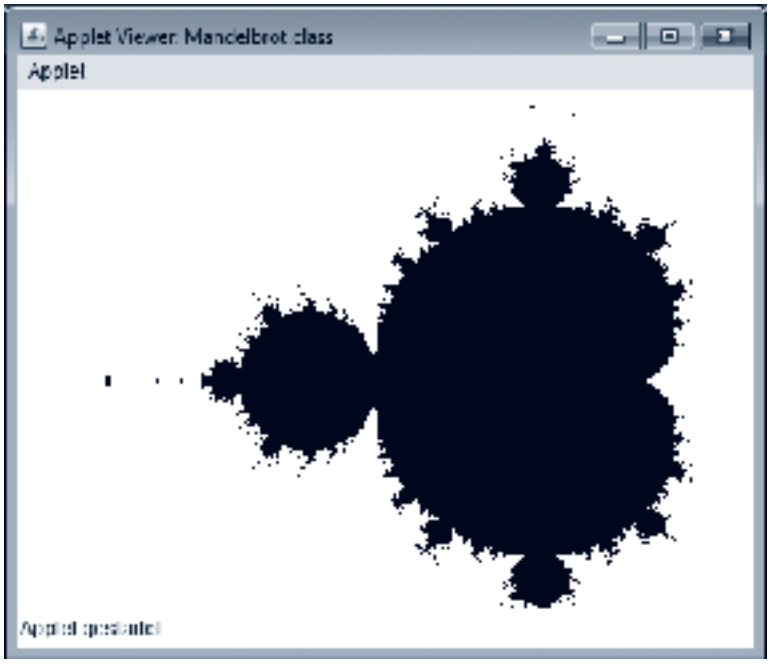


13.3 Applets erstellen

13.3.1 Grundlagen von Applets

Die meisten Internetnutzer kennen Applets, wenn vielleicht auch nicht unter diesem Namen. Applets sind nämlich Java-Programme, die in einem Browser ausgeführt werden. Dabei werden sowohl HTML-Seite als auch der Applet-*Bytecode* von einem Server geladen und im Browser angezeigt. Der Browser muss allerdings das Java-*Plug-in* installiert haben. Mit Applets steht ein mächtiges Werkzeug für die Web-Programmierung zur Verfügung. Aus Sicherheitsgründen gibt es aber folgende sinnvolle Einschränkungen für Applets:

- Applets werden vor der Ausführung vom Browser auf Sicherheitsrisiken geprüft. Das Laden eines Applets dauert deshalb etwas länger als bei einem gewöhnlichen Java-Programm.
- Das Applet darf nicht auf lokale Speichermedien (beispielsweise die Festplatte) zugreifen.
- Auch der Zugriff auf andere Ressourcen im Netzwerk wird blockiert. Das Applet darf nur auf den Server zugreifen, von dem es geladen wurde (um beispielsweise weitere Daten zu erhalten).



Das Applet zeigt eine sogenannte **Mandelbrot-Menge** (auch **Apfelmännchen** genannt). Dieses interessant aussehende Bild entsteht über eine relativ einfach definierte *Rekursion* und kann daher hervorragend mit den Grafikmethoden von Java umgesetzt werden.

Hinweis:

Das obige *Mandelbrot*-Applet wurde aus ***NetBeans*** mithilfe des *Applet Viewer* gestartet. Das vereinfacht den Entwicklungsprozess von Applets. Das Starten über eine HTML-Seite wird später dargestellt.

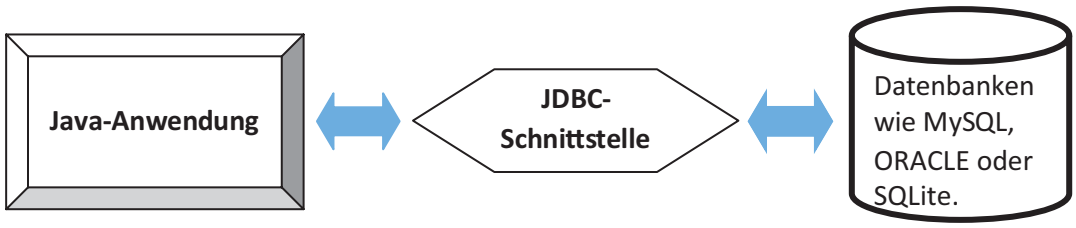
15 Datenbankanbindung

15.1 Datenbankzugriff mit Java

Das Speichern von Daten kann eine Anwendung natürlich mithilfe von Dateioperationen selbst durchführen. Für wenige Daten ist das wahrscheinlich auch die beste Wahl bei der Entwicklung einer Anwendung, weil sie damit relativ unabhängig ist. Wenn allerdings viele Daten (oder Datensätze) zu speichern sind und die Daten zusätzlich einen komplizierten Aufbau haben, dann ist die Speicherung in einer Datenbank in Betracht zu ziehen. Der große Vorteil bei einer Datenbankanbindung ist die Unabhängigkeit der Anwendung von der technischen Umsetzung der Datenspeicherung. Das erledigt die Datenbank im Hintergrund. Auch das Ändern oder Löschen von Daten ist bequem durch einige Datenbankbefehle (SQL-Befehle) zu realisieren. Die Abfragesprache **SQL** (Structured Query Language) spielt hierbei eine wichtige Rolle. Bei den folgenden Ausführungen werden deshalb auch grundlegende Kenntnisse in SQL vorausgesetzt.

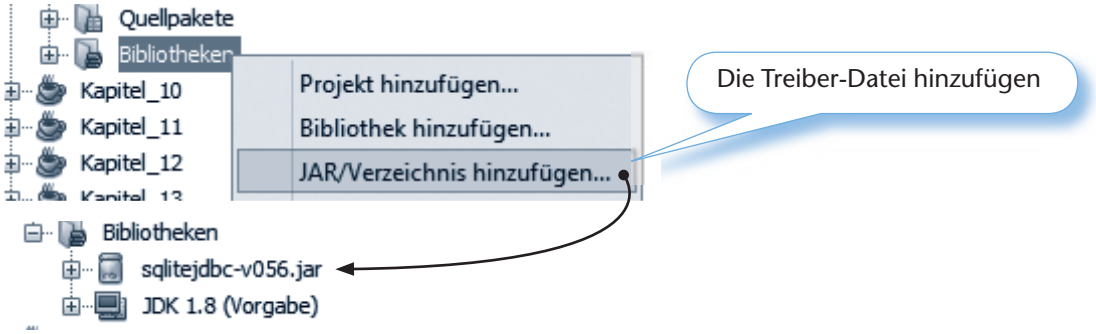
15.1.1 Datenbankanbindung mit JDBC

Java bietet eine Vielzahl von Klassen, um die Anbindung an eine Datenbank zu realisieren. Diese Klassen sind unter dem Oberbegriff **JDBC** (Java Database Connectivity) gesammelt. Für die meisten Datenbanken existieren sogenannte JDCB-Schnittstellen, die den Datenbankzugriff aus einer Java-Anwendung in die entsprechenden Befehle der Datenbank umwandeln. Damit ist es für den Java-Programmierer im Prinzip egal, mit welcher Datenbank im Hintergrund gearbeitet wird – der Zugriff ist gleich. Die folgende Abbildung zeigt das Grundprinzip dieses Zugriffs:



Im Folgenden wird der Zugriff auf eine SQLite-Datenbank vorgestellt. Das Prinzip ist aber übertragbar auf andere relationale Datenbanken wie beispielsweise MySQL- oder Oracle-Datenbanken. Dabei muss das Paket `java.sql` importiert werden. In diesem Paket sind alle relevanten Klassen, um auf eine Datenbank zuzugreifen.

Nach dem Download des gewünschten Treibers (beispielsweise *sqlitejdbc-vxx.jar*) wird die Datei vom Typ Java-Archive in das Projekt integriert:



Nach dem erfolgreichen Hinzufügen des Treibers kann die Verbindung mit der Klasse `Class` erzeugt werden:

```
String datenbank = "jdbc:sqlite:/Pfad/Datenbank";
```

Den Pfad und die Datenbankdatei angeben

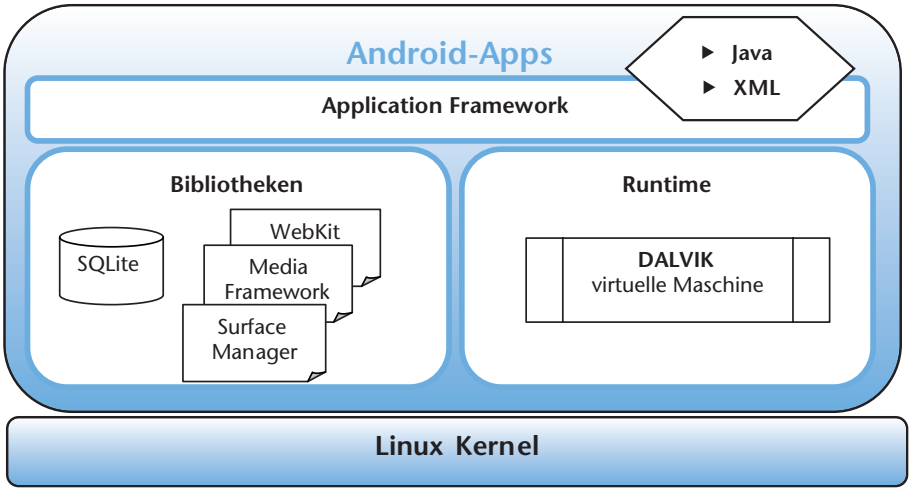
16 Android App-Entwicklung

16.1 Android-Apps Grundlagen

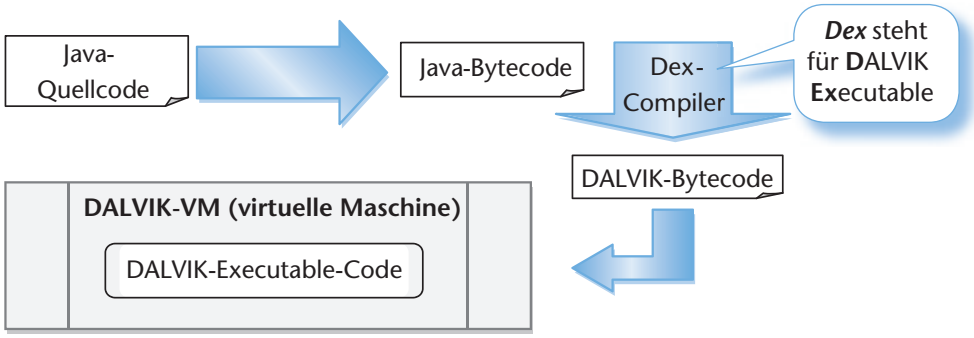
Android ist ein Betriebssystem für mobile Geräte wie Smartphones oder Tablets. Android basiert auf Linux, wurde aber für den Einsatz auf mobilen Geräten modifiziert. Die Firma Google hat dieses Betriebssystem entwickelt, stellt es aber kostenfrei zu Verfügung. Damit können Anbieter von mobilen Geräten Anpassungen für die jeweiligen Geräte durchführen. Ebenso können auch private Anwender das System nutzen, um es zu modifizieren. Anwendungen für Android werden ***Apps*** (für Applikationen) genannt und können auf den mobilen Geräten sehr einfach installiert werden. Google bietet einen Online-Shop (Play-Store) an, um Apps entweder kostenfrei zu beziehen oder über verschiedene Bezahlsysteme (wie PayPal oder Kreditkarte) zu kaufen. Damit stehen jedem Nutzer Millionen von Apps zur Verfügung.

Android-Apps werden in Java entwickelt. Für die Entwicklung werden verschiedene Bibliotheken eingebunden und letzendlich wird die App in einer speziellen virtuellen Maschine (DALVIK) ausgeführt. Eine Android-App liegt als eine ****.apk-Datei*** (für Android-Package) vor. Diese Datei kann auch ohne Online-Systeme wie den Play-Store auf ein mobiles Gerät (beispielsweise über USB[1]) kopiert und installiert werden.

Die folgende Übersicht zeigt den strukturellen Aufbau einer Android-App:

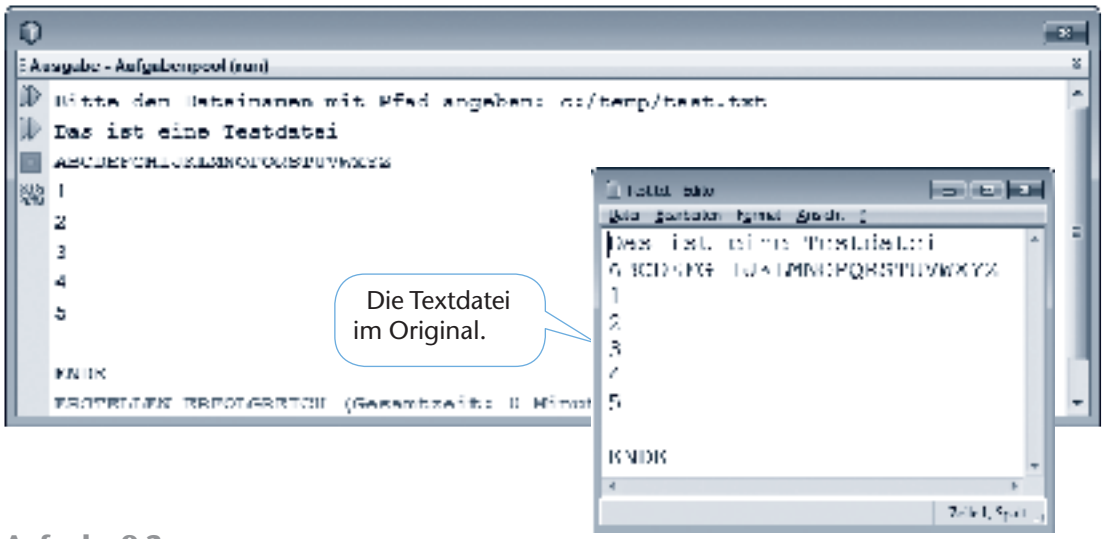


Der Ablauf einer App-Erstellung ähnelt der Erstellung einer Java-Anwendung, wie die folgende Grafik zeigt:



1 Das Kopieren einer apk-Datei über USB erfordert in der Regel, dass in den Einstellungen die Installation anderer Apps als Play-Store-Apps zugelassen wird. Danach kann über einen Dateiexplorer die apk-Datei zur Installation ausgewählt werden.

Nach dem Starten könnte das Programm so aussehen:



Aufgabe 9.2

Schreiben Sie ein Java-Programm, welches eine Kopie einer beliebigen Datei anfertigt. Dazu soll der Benutzer den Namen der zu kopierenden Datei und den Namen der Kopie angeben.

Nach dem Starten könnte das Programm so aussehen:

Hinweis:

Das binäre Lesen und Schreiben kann mit den Klassen `FileInputStream` und `FileOutputStream` umgesetzt werden:

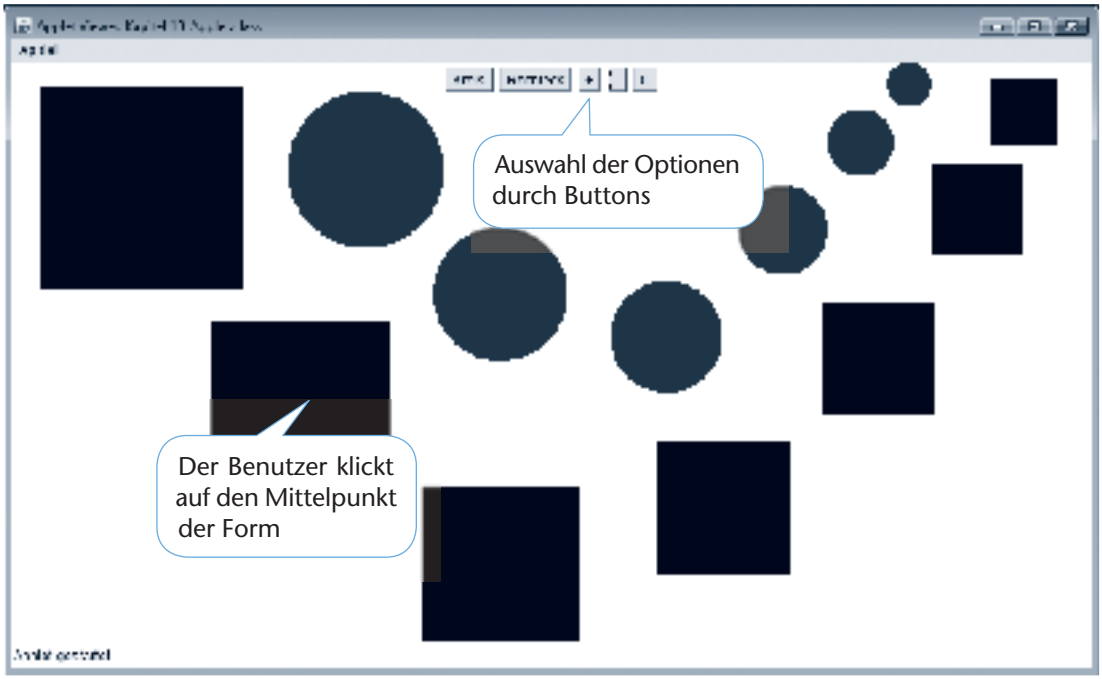
```
FileInputStream ein = new FileInputStream("Quelldateiname");
FileOutputStream aus = new FileOutputStream("Zieldateiname");

byte [] b = new byte[1];

lesen.read(b);

schreiben.write(b);
```

Ein einelementiges Array zum binären Einlesen

Dateiname mit Pfadangabe

Aufgabe 13.2

Entwickeln Sie ein Java-Applet, das Kreise und Rechtecke in den Clientbereich zeichnen kann. Der Benutzer klickt dazu mit der Maus in den Clientbereich und anschließend wird eine der Formen gezeichnet. Die Kreise werden in Blau und die Rechtecke in Schwarz gezeichnet. Mithilfe der Buttons kann der Benutzer auswählen, welche Form gezeichnet werden soll. Durch das Klicken auf den Plus-Button vergrößert sich die nächste zu zeichnende Form (entsprechend der Minus-Button). Mit dem `c`-Button (*clear*) werden alle Formen gelöscht. Die Formen sollen in einer Liste gespeichert werden (beispielsweise `ArrayList`), so dass die `paint`-Methode jederzeit in der Lage ist, den kompletten Applet-Inhalt korrekt darzustellen.



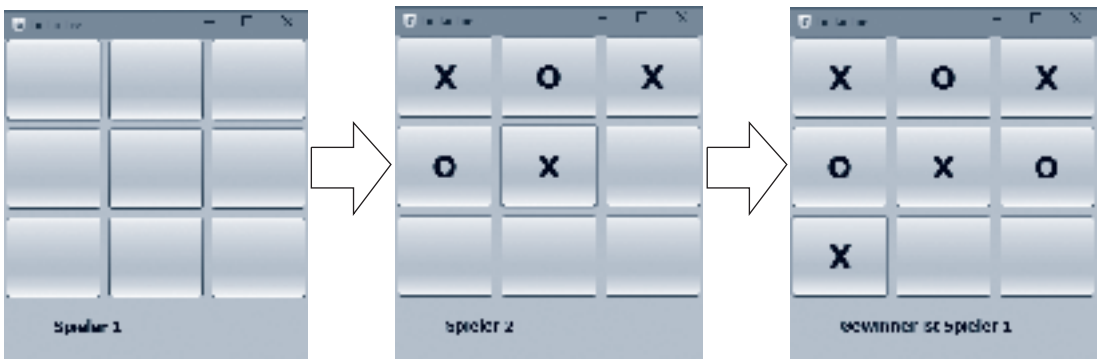
Hinweis:

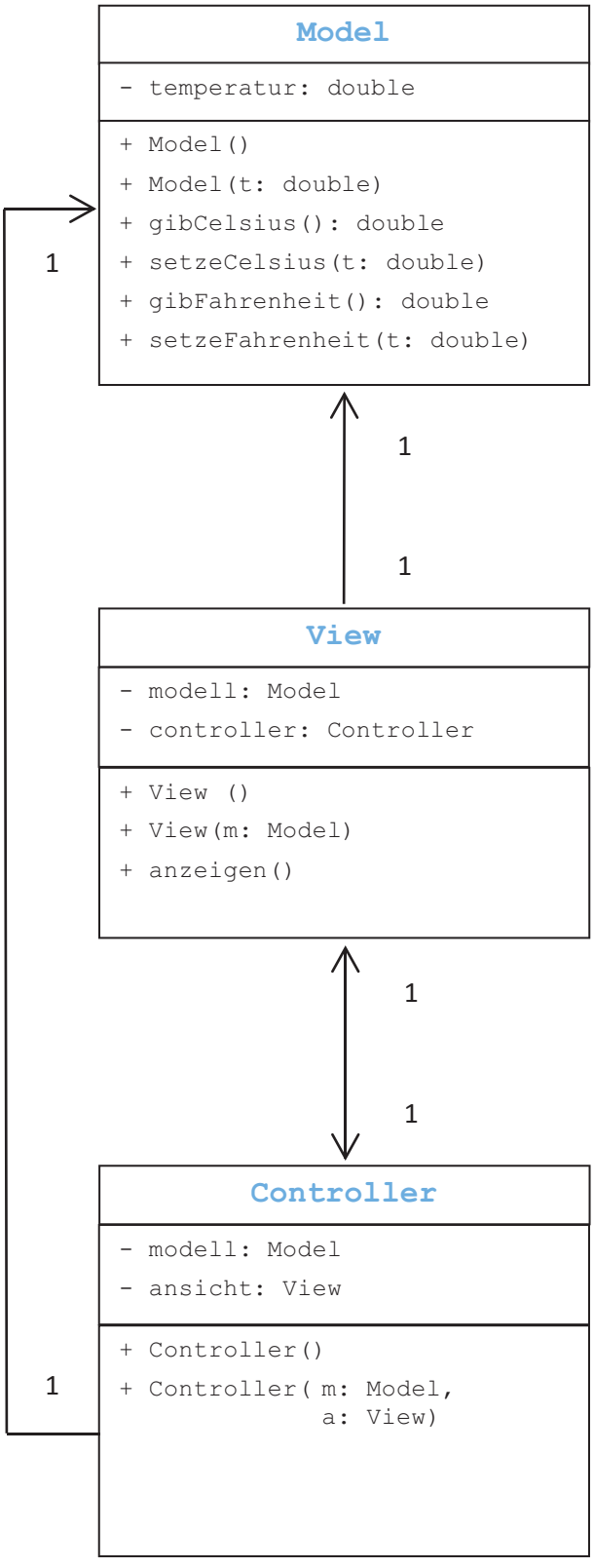
Entwerfen Sie Klassen für die Formen, die alle von einer abstrakten Basisklasse abgeleitet sind. Damit wird das Speichern der Formen in einer `ArrayList` vereinfacht. Die Klassen sollten eine Methode haben, die die Form auf den Bildschirm zeichnet. Diese Methode ist in der abstrakten Basisklasse angelegt und wird in den vererbten Klassen überschrieben.

14 Aufgaben zum NetBeans GUI Builder

Aufgabe 14.1

Schreiben Sie eine Java-Anwendung, die das Spiel ***Tic-Tac-Toe*** aus Aufgabe 11.4 mithilfe des GUI-Builders umsetzt. Legen Sie dazu neun Buttons mit der *Palette* an und weisen Sie jedem Button einen Ereignisempfänger zu. Die Logik des Spiels kann prinzipiell aus der Aufgabe 11.4 übernommen werden. Die Kreise und Kreuze werden nun einfach mit einer entsprechend großen Schrift als Beschriftung des Buttons dargestellt („**X**“ oder „**O**“).





Die Fachklasse `Model`:

Dieses Modell speichert eine Temperatur in Celsius. Die Methoden sorgen dafür, dass diese Temperatur auch in Fahrenheit angegeben werden kann.

Umrechnung:

Celsius = (Fahrenheit – 32) / 1,8

Die Ansichtsklasse `View`:

Die Ansichtsklasse erhält einen Verweis auf das Modell und erzeugt einen `Controller`. Die Methode `anzeigen()` holt die Daten vom Modell und zeigt sie entsprechend an.

Die Steuerungsklasse `Controller`:

Die Steuerung erhält Verweise auf das Modell und auf die Ansichtsklasse, von welcher sie erzeugt wurde.

Die Steuerungsklasse sorgt für Interaktion mit dem Benutzer und ruft entsprechende Methoden beim Modell oder der Ansicht auf.

Lernsituation 6:

Entwicklung einer App, um Sudokus zu lösen

Ausgangssituation:
Ein bekannter Verlag für Rätselhefte möchte seine Produkte mit kostenfreien Android-Apps aufwerten. Zu den verschiedenen Rätselheften (Kreuzworträtsel, Sudokus usw.) sollen über eine Plattform wie *Google play* kostenfreie Apps angeboten werden. Damit sollen die Kunden stärker an die Produkte gebunden werden. Die Firma **ProSource** erhält von dem Verlag den Auftrag eine App zu entwickeln, die Sudokus **selbstständig** lösen kann. In einer ersten Variante sollen 4x4-Sudokus gelöst werden.

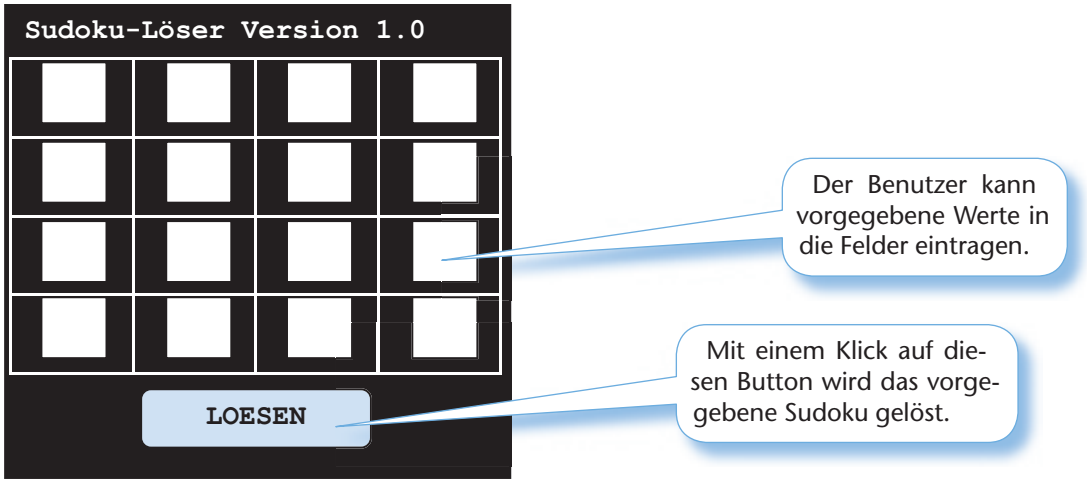
Als Auszubildender der Firma erhalten Sie nun den Auftrag diese App zu entwickeln.

Arbeitsschritte in Einzel- oder Partnerarbeit:

Planung:

Von dem Verlag wurde ein Layout entworfen, das als Grundlage für die Entwicklung dienen soll:

Layoutentwurf:



Beispiel einer Anwendung: