# Java Algorithmen am Beispiel der DIN 1045-1

Dipl.-Ing. Richard Romberg
Lehrstuhl für Bauinformatik
Technische Universität München
Theresienstr. 90 • D - 80290 München
Telefon: 089 / 289 – 25065
Fax: 089 / 289 – 25051
E-mail: romberg@bv.tum.de

Dipl.-Ing. Christoph Schmidhuber
Lehrstuhl für Massivbau
Technische Universität München
Theresienstr. 90 • D – 80290 München
Tel.: 089 / 289 – 23029
Fax.: 089 / 289 – 23046
E-mail: schmidhuber@mb.bv.tum.de

# 1 Einleitung

Der wachsende Bedarf an plattformunabhängigen Programmen drückt sich in der vermehrten Verwendung der Entwicklungsumgebung Java aus. Der wesentliche Vorteil dieser Sprache liegt in der Erzeugung eines Zwischencodes (Bytecode), der erst während der Ausführung des erstellten Programms auf die jeweiligen spezifischen Hardwaregegebenheiten der verwendeten Plattform angepasst wird. Damit liefert Java einen wichtigen Beitrag zur vereinfachten parallelen Verwendung der beiden bedeutenden Betriebssysteme Windows und Unix (Linux). Um dem Leser einen ersten Einstieg in die Thematik der plattformunabhängigen Softwareentwicklung zu geben, werden mit diesem Beitrag die Grundlagen der Java Programmierung aufgezeigt.

In einem ersten Schritt werden anhand des vielfach zitierten „Hello World“-Programms grundlegende Elemente der Java-Programmierung erläutert. Im Unterschied zum "Hello World"-Programm, das als textorientiertes Programm entwickelt wurde, zeigt das zweite Beispiel die Implementierung eines Java-Programms mit grafischer Benutzeroberfläche. Abschließend werden in einem umfangreicheren dritten Beispiel mit Bezug zur Bemessungspraxis von Bauingenieuren fortführende Programmiertechniken dargestellt. Aufgrund der aktuellen Entwicklung in der Normung im Bereich des Bauingenieurwesens wird im Rahmen dieses Beispieles ein Programm zur Biegebemessung von Stahlbetonträgern nach DIN 1045-1 (05.00) entwickelt. Zu diesem Zweck werden vorab die mechanischen Grundlagen zur Bemessung eines Stahlbetonträgers nach DIN 1045-1 (05.00) dargestellt. Da jedoch in diesem Beitrag die Programmiertechniken vorrangig behandelt werden, sind einschränkende Randbedingungen für das Bemessungsverfahren nach DIN 1045-1 (05.00) festgelegt worden.

Um den Lehrcharakter dieses Beitrags herauszustellen, werden wichtige Abschnitte des Quelltextes herausgegriffen und detailliert besprochen.

# 2 Eine einfache Java Konsolen-Anwendung

Erfahrungsgemäß sind die Hürden für den unerfahrenen Programmierer bei der Implementierung der ersten Anwendung besonders hoch. Die folgenden Erläuterungen beschreiben deshalb detailliert die Erstellung einer Java-Anwendung von den ersten Schritten bis zum lauffähigen Programm. Dabei geht es um das obligatorische Programm zu Ausgabe eines Textes am Bildschirm: 'HelloWorld!'.

## 2.1 Voraussetzungen

Zur Entwicklung eines Java Programms ist das Java Development Kit (JDK) erforderlich. Es kann bei [2] bezogen werden und umfasst ca. 20 MB. Nach der Installation sollte das Verzeichniss **'C:\jdk1.2.2\bin'** in den Suchpfad (PATH Variable) für Programme eingefügt wurden. Bei Windows NT ist das Dienstprogramm **System** in den **Einstellungen, Systemsteuerung** zu starten. Im Reiter **Umgebung** gibt es eine Variable **path** bei Benutzervariablen im unteren Dialogbereich. Nach der Markierung der Variablen, kann im Edit-Feld folgender Text an das Ende des vorhandenen Textes hinzugefügt werden: **;C:\jdk1.2.2\bin**. Der Strichpunkt vor der Verzeichnisangabe ist wichtig, da er die verschiedenen Verzeichnisse untereinander trennt. Bei Windows95 und Windows98 muss die Datei **C:\autoexec.bat** geändert werden. Die Datei kann mit Hilfe eines Editors (z.B. Notepad oder Edit) geöffnet werden. Die Zeile mit der Variablen **PATH** sollte folgendes Aussehen haben:

PATH=C:\WINDOWS;C:\WINDOWS\COMMAND**;C:\JDK1.2.2\BIN**

Bei Windows NT werden die vorgenommenen Änderungen beim nächsten Start der Eingabeaufforderung gültig, bei Windows95/Windows98 nach einem Neustart oder durch Starten des Programms C:\autoexec.bat. Sinn der oben beschriebenen Einstellung ist die Vereinfachung des Starts von Java Tools aus einem beliebigen Verzeichnis ohne Angabe des Java Verzeichnisses.

Das Projektverzeichnis für die Anwendung ist: **C:\java\HelloWorld**.

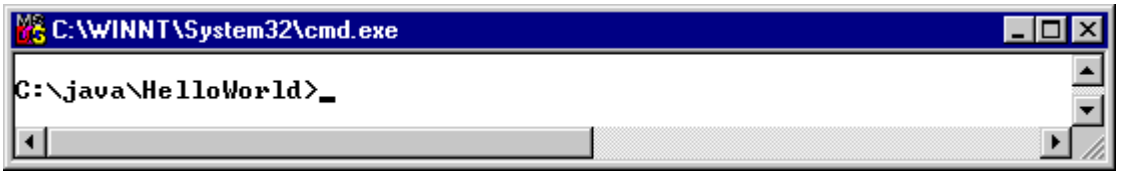


**Abbildung 2.1:** Projektverzeichnis

## 2.2 Implementierung der Konsolenanwendung

Das zu erstellende Programm wird den Gruß 'Hello, World!' am Bildschirm anzeigen. Zunächst wird mit Hilfe eines Editors der Java Quelltext (source code) in die Datei **HelloWorldApp.java** eingegeben (Abb. 2.2). Nach Speicherung der Datei wird mit dem Java Compiler **javac** der Quelltext in Java Bytecode übersetzt (Abb. 2.3). Der übersetzte (kompilierte) Bytecode ist zunächst für einen Rechner noch nicht ausführbar, sondern besteht aus abstrakten Prozessoranweisungen, die der Sprache die Unabhängigkeit von einer konkreten Plattform ermöglicht. So ist der Quellcode bzw. der kompilierte Bytecode für dieses Beispiel für alle verfügbaren Systeme (Windows, Linux, MaxOS, uvm.) identisch. Auf den Zielplattformen übersetzt die **Java Virtual Maschine** (Java VM) den Bytecode in die spezifischen Prozessorbefehle und führt diese aus. Die Java VM selbst ist die einzige plattformspezifische Anwendung in dieser Software-Entwicklungsumgebung. Diese muss einmal installiert und ggf. durch Updates auf den neuesten Stand gebracht werden. Die VM ist in der Datei **java.exe** realisiert (Abb. 2.4).

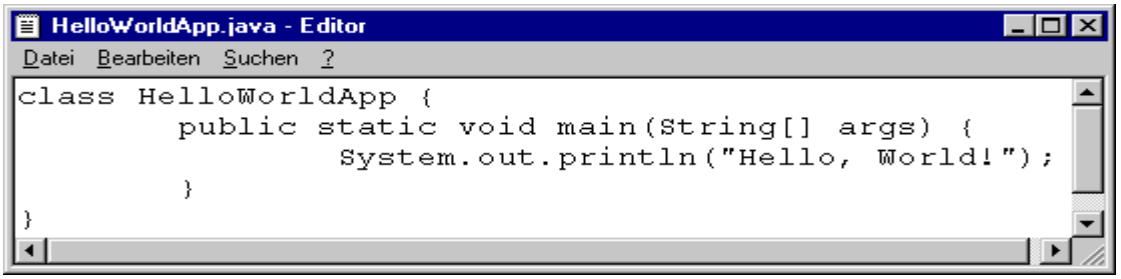

**Abbildung 2.2:** Eingabe des Quelltextes


**Abbildung 2.3:** Kompilieren des Quelltextes in Java Bytecode

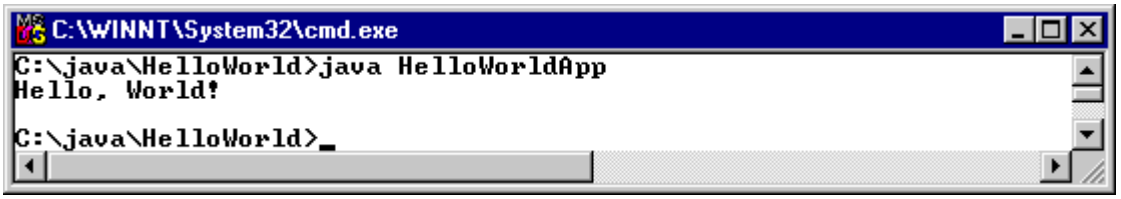


**Abbildung 2.4:** Ausführen der Anwendung

### 2.3 Java verwendet Bytecode

Mit der Einführung des Java Bytecode wurde Softwareentwicklung für unterschiedliche Plattformen deutlich vereinfacht. Der oft zitierte Satz "write once, run anywhere" trifft die zentrale Neuerung von Java auf den Punkt. Nachteilig wirkt sich die zusätzliche Softwareschicht jedoch auf die Ausführungsgeschwindigkeit von Programmen aus. Die Einbußen im Laufzeitverhalten können je nach Aufgabenstellung signifikant sein.

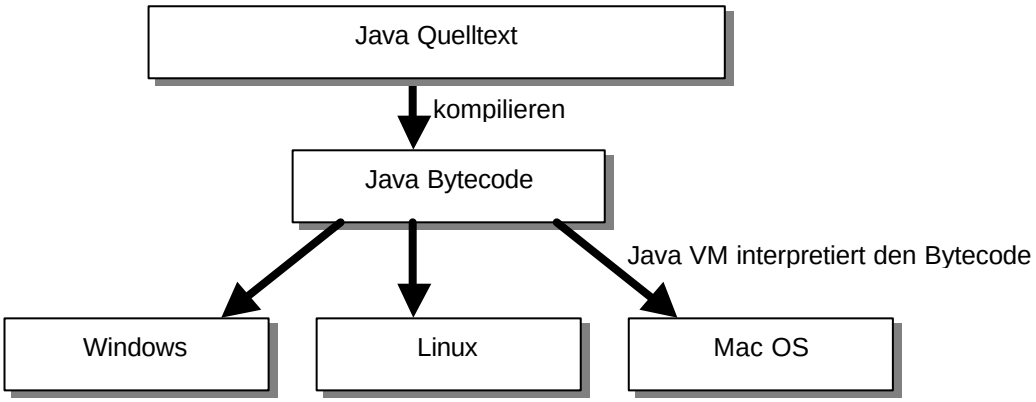


**Abbildung 2.5:** Abhängigkeiten von Quelltext und ausführbarem Programmcode

Eine sehr ausführliche Beschreibung zur Erstellung der Anwendung 'HelloWorld' ist unter [3] zu finden.

### 2.4 Details der Konsolenanwendung

Eine der wichtigsten Merkmale objektorientierter Programmiersprachen sind Klassen (classes). Sie definieren eine Einheit bezüglich einer Gruppe von Attribute und Methoden (Funktionen), die die Attribute manipulieren. Die Anzahl der in einem Projekt verwendeten Klassen ist dabei nicht begrenzt, lediglich die Bezeichnungen von Klassen müssen im Projekt eindeutig sein. Bei Java gibt es eine einfache Zuordnung zwischen Klassen und Dateien. Jede Java Quellcode-Datei enthält mindestens eine Java-Klasse. Der Name der Klasse entspricht dem Dateinamen. So ist im obigen Beispiel der Quellcode der Klasse 'HelloWorld' in der Datei 'HelloWorld.java' gespeichert. Werden weitere Klassen in einer Quellcode-Datei implementiert, so werden diese bei der Kompilierung in Dateien mit Namen geschrieben, die aus der Kombination von Haupt- und Nebenklasse gebildet werden. Angenommen in der Datei 'HelloWorld' wäre eine weitere Klasse (Nebenklasse) implementiert, würde der Bytecode der Nebenklasse in der Datei 'HelloWorld$Nebenklasse.class' abgelegt werden (siehe Listing 2.1).

```
1    public class HelloWorld
2    {
```

```
3       class Nebenklasse
4       {
5         // ... Quellcode
6       }
7     }
```

**Listing:** 2.1

Groß- und Kleinschreibung ist bei Java grundsätzlich zu beachten. Das gilt auch für die Windows-Betriebsysteme, da diese zwar Groß- und Kleinschreibung nicht beachten, jedoch Dateinamen in der vorgegebenen Schreibweise beibehalten (Case-erhaltend). Der Aufbau einer Klasse ist einfach und folgt immer demselben Schema:

```
1     public class HelloWorld
2     {
3       int i;
4       public static void main(String args[])
5       {
6         i=10;
7       }
8       public ausgabe
9       {
10      }
11    }
```

**Listing:** 2.2

Die Hauptklasse in einer Datei muss mit dem Schlüsselwort 'public' gekennzeichnet sein. Das bedeutet, dass von anderen Klassen im Projekt auf Funktionen aus dieser Klasse zugegriffen werden kann. Auf Funktionen von Nebenklassen kann von 'außerhalb' nicht zugegriffen werden. Die Definition der Klasse wird im nachfolgenden Block innerhalb der öffnenden geschweiften Klammer und der schließenden geschweiften Klammer durchgeführt (siehe Listing 3.2). Einrückungen des Quelltextes innerhalb von Blöcken helfen dabei die Übersicht über den Quelltext zu bewahren. Blöcke werden in Java nicht nur für die Definition von Klasse verwendet, sondern kommen in den verschiedensten Sprachkonstrukten zur Anwendung (siehe Listing: 2.2). Das Wesentliche an Blöcken ist, dass sie Quellcode-Abschnitte zusammenfassen und einem Java-Sprachkonstrukt zuordnen. Blöcke haben innerhalb einer Datei immer eine hierarchische Beziehung zueinander; so ist die Funktion 'main' ein Teil der Klasse 'HelloWorld'.
Neben der Definition von Klassen wie in Listing 2.2 gibt es noch sog. Objekte, die jedoch nur zur Laufzeit des Programms im Arbeitspeicher des Rechners existieren. Die Klassen stellen für Objekte Schablonen oder Baupläne dar, nach denen sie erzeugt werden. Jedes Objekt beansprucht selbst Arbeitsspeicher und reserviert für jede in der Klasse definierte Variable einen ausreichenden Speicherplatz. Von einer Klasse können beliebig viele Objekte erzeugt werden. Diese sind über einen eindeutigen Objekt-Namen ansprechbar. Objekte sind abstrakte Einheiten, die in der objektorientierten Programmierung mit dem Begriff der Datenkapselung in Zusammenhang stehen.
In Zeile drei aus Listing 2.2 wird gezeigt, wie Variablendeklarationen innerhalb von Klassen vorgenommen werden. Die Variable i ist damit in der gesamten Klasse 'HelloWorld' verwendbar. Eine besondere Bedeutung hat die Funktionen 'main'. Sie wird automatisch von der VM nach dem Laden der Java-Anwendung aufgerufen. Das Schlüssenwort 'static' wird verwendet, da die Funktion 'main' von der VM aufgerufen wird, bevor eine Objekt der Klasse 'HelloWorld' im Speicher angelegt wird. Im Parameter args[ ] wird eine Feld von Programmargumenten übergeben, die der Benutzer an der Konsole beim Programmaufruf angegeben hat.

# 3 Anwendung mit grafischer Benutzeroberfläche (GUI)

Nachdem in Abschnitt 2 vorbereitend eine Einführung in die Programmierung einer Konsolenanwendung besprochen wurde, wird im Folgenden ein Java-Programm mit einer grafischen Benutzeroberfläche (graphical user interface) entwickelt. Dazu leiten wir eine neue Klasse 'Bemess' von der Klasse 'Frame' ab. 'Frame' ist Bestandteil der JDK-Bibliothek und liefert die notwendige Funktionalität für das Erzeugen und Managen eines Fensters. Durch Ableiten – in der Literatur wird häufig das Wort 'Vererben' verwendet – werden alle Eigenschaften (Funktionen und Variablen) aus der Basisklasse (hier 'Frame') auf die neue Subklasse (hier 'Bemess') übertragen. Der Subklasse stehen dadurch nicht nur die eigenen Eigenschaften zur Verfügung, sondern auch die der Basisklasse. Hier eine Beispiel:

```
1    import java.awt.*;
2
3    public class Bemess extends Frame
4    {
5      public static void main(String args[])   // Main-Funktion
6      {
7        Bemess dlg = new Bemess();
8        dlg.setTitle("Bemessung nach DIN1045");
9        dlg.setSize(600,300);
10       dlg.show();
11     }
12     public Bemess()   // Konstruktor
13     {
14     }
15   }
```

**Listing:** 3.1

Das Schlüsselwort 'extends' definiert die Ableitung von der Klasse 'Frame'. Die Klasse 'Frame' ist in der Java-Bibliothek 'Abstract Window Toolkit' (AWT) enthalten. Damit die Klasse verfügbar ist, muss die entsprechende Bibliothek geladen werden. Die Anweisung 'import java.awt.*;' aus Zeile 1 führt die notwendigen Schritte dazu durch. Zwischen der Anweisung 'import' und dem Dateisystem auf der Festplatte gibt es einen einfachen Zusammenhang. Jede Angabe zwischen den Punkten definiert ein Verzeichnis. Ein Stern referenziert alle Dateien in einem Verzeichnis. Die 'import' Anweisung in Zeile 1 des Listings 3.1 lädt somit alle Dateien (Bibliotheken) aus dem Verzeichnis 'java\awt\'. Sofern nicht anderes definiert sind die wichtigsten Bibliotheken in der komprimierten Datei 'src.jar' im Verzeicnis 'C:\jdk1.2.2\' enthalten. Der Pfad in der 'import' Anweisung bezeichnet einen relativen Pfad in der Datei 'src.jar' (jar ist eine Abkürzung für Java Archive). Java Archive können auch vom Benutzer für eigene Anwendungen verwendet werden. Der zugrundegelegte Komprimierungsalgorithmus ist Zip-kompatibel.
In Zeile 7 wird das Bemessungs-Objekt nach Vorlage der Bemessung-Klasse mit 'new' angelegt. Die von 'new' zurückgegeben Referenz kann in einer entsprechenden Variablen für den späteren Zugriff aus das Objekt abgelegt werden. In Zeile 8 bis 10 werden Funktionen der Klasse über diese Referenz aufgerufen.


**Abbildung 3.1:** Anwendung mit grafischer Benutzerschnittstelle

# 4 Mechanischen Grundlagen zur Biegebemessung nach DIN 1045-1

## 4.1 Grundlegende Zusammenhänge

In diesem Abschnitt werden die Grundlagen der Biegebemessung von Stahlbetonträgern ohne Druckbewehrung nach DIN 1045-1 (05.00) beschrieben. Bei der Biegebemessung werden die einwirkenden und die widerstehenden Schnittgrößen miteinander verglichen. Dabei werden in der Bemessungspraxis in der Regel die Querschnittsabmessungen und die Betonfestigkeitsklasse vorgewählt und die erforderliche Biegebewehrung durch Gleichsetzen der Einwirkungen und der Bauteilwiderstände ermittelt. Die Bemessungsaufgabe erfordert dann die Berechnung des durch die Einwirkungen hervorgerufenen Dehnungszustandes und der daraus resultierenden inneren Schnittkräfte (siehe Abb. 4.1).

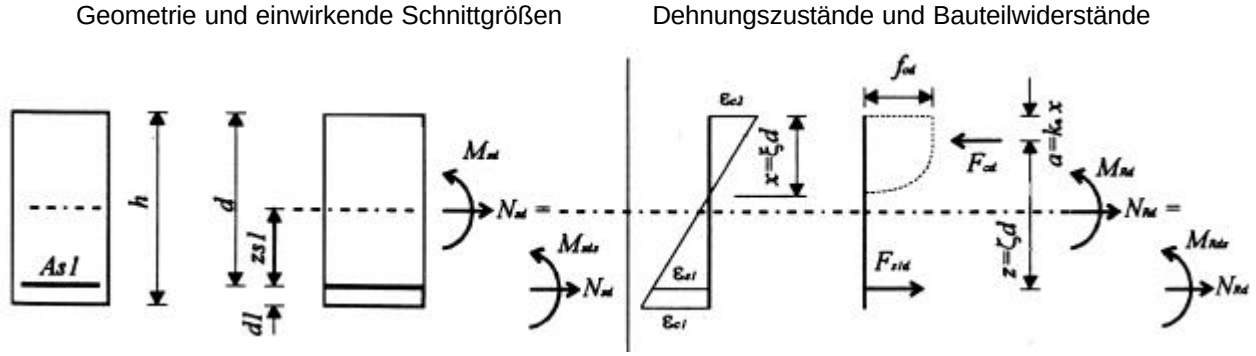


**Abbildung 4.1:** Grundlegende Zusammenhänge am Stahlbetonquerschnitt

Zur Vereinfachung der Berechnung werden die Schnittgrößen $M_{Sd}$ und $N_{Sd}$ auf die Achse der Biegezugbewehrung bezogen. Damit ergibt sich für die Einwirkungen folgender Zusammenhang:

$$M_{Sds} = M_{Sd} - N_{Sd} * z_{s1} \quad (4.1)$$

Die Bauteilwiderstände ergeben sich aus den inneren Schnittkräften im Querschnitt:

$$N_{Rd} = F_{cd} + F_{s1d} \quad (4.2)$$

$$M_{Rds} = F_{cd} * z \quad (4.3)$$

mit:

$$z = d - k_a * x \quad (4.4)$$

Die resultierende Betondruckkraft berechnet sich mit folgender Gleichung:

$$F_{cd} = b * x * \alpha_R * f_{cd} \quad (4.5)$$

Für die Zugkraft in der Biegezugbewehrung gilt

$$F_{s1d} = A_{s1} * \sigma_{s1d} \quad (4.6)$$

Dabei ist:

| | |
|---|---|
| $f_{cd}$ | Bemessungswert der Zylinderdruckfestigkeit des Betons |
| $\alpha_R$ | Völligkeitsbeiwert der Betondruckzone |
| d | statische Nutzhöhe |
| z | Innerer Hebelarm |
| x | Höhe der Betondruckzone |
| $k_a x$ | Abstand der Betondruckkraft vom gedrückten Rand |
| $\sigma_{s1d}$ | Bemessungswert der Betonstahlspannung |

### 4.2 Bemessung für Biegung ohne Druckbewehrung

Im Rahmen des hier entwickelten Bemessungsprogramms werden folgende vereinfachende Randbedingungen für die Biegebemessung festgelegt:

- Stahlbetonträger mit Rechteckquerschnitt
- Bemessung für Biegung ohne Normalkraft
- einlagige Betonstahlbewehrung in der Biegezugzone
- vollständig ausgenutzte Betondruckzone (Betonstauchung $\varepsilon_c$ = -3,5 ‰)

$$\Sigma N = 0 \qquad N_{Sd} = N_{Rd} \tag{4.7}$$

$$\rightarrow \qquad F_{cd} + F_{s1d} = 0 \tag{4.8}$$

$$\Sigma M_{s1} = 0 \qquad M_{Sds} = M_{Rds} \tag{4.9}$$

$$\rightarrow \qquad M_{Sds} - F_{cd} * z = 0 \tag{4.10}$$

mit $\mu_{Sds} = M_{Sds} / ( b * d^2 * f_{cd} )$ folgt:

$$\mu_{Sds} - \alpha_R * x/d * z/d = 0 \tag{4.11}$$

$$\mu_{Sds} - \alpha_R * \xi * \zeta = 0 \tag{4.12}$$

mit $\zeta = (1 - k_a * \xi)$ folgt:

$$\mu_{sds} - \alpha_R * \xi * (1-k_a * \xi) = 0 \tag{4.13}$$

**Lösung der Bemessungsaufgabe**

Die Bemessungsaufgabe ist gelöst, wenn ein $\xi$ gefunden werden kann, das die Gleichung (4.13) erfüllt. Die Lösung der quadratischen Gleichung kann mit folgendem Ausdruck angegeben werden:

$$? = \frac{1}{2k_a}\left(1 - \sqrt{1 - \frac{4k_a \mu_{sds}}{a_R}}\right) \tag{4.14}$$

Der Völligkeitsbeiwert $\alpha_R$ und der Beiwert $k_a$ ergeben sich für die Betonfestigkeitsklassen bis C50/60 unter Voraussetzung einer ausgenutzten Betondruckzone ($\varepsilon_c$ =-3,5 ‰) bei Verwendung des Parabel-Rechteck-Diagramms zu $\alpha_R$=0,81 und $k_a$ = 0,416. Damit sind die Dehnungsverhältnisse im Querschnitt einfach berechenbar.

Mit z = d – $k_a$ * x für den inneren Hebelarm berechnet sich die erforderliche Betonstahlbewehrungsmenge mit

$$A_{s1} = \frac{1}{s_{s1d}} * \frac{M_{Sds}}{z} \tag{4.15}$$

Die oben angegebenen Gleichungen zur Bestimmung der Betonstahlbewehrung sind im Zusammenhang mit den in diesem Abschnitt festgelegten Randbedingungen gültig.

# 5 Bemessungsprogramm nach DIN 1045-1

Das Fenster in Abbildung 3.1 ließ sich mit wenigen Java-Anweisungen erzeugen. Wesentlicher Schwachpunkt dabei ist, dass sich die Anwendung nicht regulär beenden lässt. Die Java Frame-Klasse implementiert nicht automatisch Funktionen, die Standard-Nachrichten behandeln (wie z.B. 'Anwendung beenden'). Im Bemessungsprogramm wird deshalb der Button 'buttonQuit' eingefügt, um damit die Anwendung zu beendet. Die Programmiertechnik, die dabei zu Einsatz kommen wird allgemein als Nachrichtenbehandlung bezeichnet. Nachrichten sind wesentliche Elemente von GUI gesteuerten Anwendungen, da der Programmablauf nicht mehr vollständig im eigenen Programmcode stattfindet – wie z.B. bei Konsolen-Anwendungen – sondern zum Teil in anderen Moduln. Die Oberfläche mit allen Fenstern und Steuerelementen (Button, Eingabefeld, etc.) verwaltet die Java VM. Führt der Benutzer eine Aktion aus – klickt er z.B. mit der Maus auf einen Button – so erzeugt die Java VM eine Nachricht `'ButtonXY wurde mit linker Maustaste angeklickt'`. Diese Nachricht kann die eigene Anwendung abfragen und entsprechend reagieren. Die Technik der Nachrichtenbehandlung wird nachfolgend am Beispiel des Steuerelements 'buttonQuit' in Listing 5.1 erläutert.

Zuerst wird ein Steuerelement vom Typ Button erzeugt und die Referenz auf das Button-Objekt in der Variablen 'buttonQuit' (Zeile 33) abgespeichert. Wichtig ist dabei, dass die Definition der Steuerelemente außerhalb jeder Funktion stattfindet. Somit ist die 'buttonQuit'-Referenz in der gesamten Klasse 'Bemess' verfügbar. In Zeile 19 und 20 werden Größe und Text festgelegt, Zeile 21 fügt das neue Steuerelement der Objektverwaltung des umgebenden Frame-Objekts hinzu. Funktionsaufrufe ('buttonQuit.setBounds' in Zeile 19) können in Java nur innerhalb von Funktionsdefinitionen stattfinden (z.B. Funktion 'Bemess'). Deshalb befinden sich die Zeilen 19-21 im Konstruktor der Klasse 'Bemess'. Dieser wird einmalig bei der Erzeugung des Objektes der Klasse 'Bemess' aufgerufen. Bis jetzt ist das Steuerelement 'buttonQuit' bei einem Programmlauf zwar sichtbar, eine Reaktion auf einen Mausklick bleibt jedoch immer noch aus.

Für die Nachrichtenbehandlung wird eine neue Klasse notwendig, die ein bestimmtes vorgegebenes Interface implementiert (Interface `ActionListener'` ist in der Bibliothek 'java.awt.event.*' gespeichert). Interfaces definieren bestimmte Funktionen. Klassen, die von diesen Interfaces abgeleitet werden, müssen diese ebenfalls implementieren. Die einzige Funktion des oben genannten Interfaces ist '`actionPerformed`'. Die Idee ist nun folgende: Klickt der Anwender auf den Button 'buttonQuit', so ruft die Java VM die Funktion '`actionPerformed`' aus der angegebenen Klasse (hier 'GUIAction') auf. Dazu muss noch die Verbindung zwischen dem Objekt 'buttonQuit' und der Nachrichtenbehandlungs-Klasse 'GUIAction' hergestellt werden. Dies geschieht in den Zeilen 26 bis 28. Die Funktion '`addActionListener`' registriert ein Objekt der Nachrichtenbehandlungs-Klasse 'GUIAction' im Steuerelement 'buttonQuit'. In der Funktion 'actionPerformed' (Zeile 38) kann jetzt auf die Nachricht `'ButtonXY wurde mit linker Maustaste angeklickt'` reagiert werden, hier beispielsweise mit '`System.exit(0)`' (Zeile 42). Das Listing 5.1 stellt nur die wesentliche Struktur des Bemessungsprogramms dar.

```
1    import java.awt.*;
2    import java.awt.event.*;
3    import java.math.*;
4
5    public class Bemess extends Frame
6    {
7      int nLeftCtrl = 5;
8      int nTopCtrl  = 30;
9
10     // Main-Funktion
11     public static void main(String args[])
12     {
13     }
14
15     // Konstruktor
16     public Bemess()
17     {
```

```
18        ...
19        buttonQuit.setBounds(nLeftCtrl+160, nTopCtrl+145, 104, 24);
20        buttonQuit.setLabel("beenden");
21        add(buttonQuit);
22        ...
23
24        ...
25        // -- Nachrichtenbehandlung --
26        GUIAction guiAction = new GUIAction();
27        buttonB.addActionListener(guiAction);
28        buttonQuit.addActionListener(guiAction);
29      }
30
31      // Deklaration der Steuerelemente
32      Button            buttonB     = new Button();
33      Button            buttonQuit  = new Button();
34      ...
35
36      class GUIAction implements java.awt.event.ActionListener
37      {
38        public void actionPerformed(java.awt.event.ActionEvent event)
39        {
40          Object object = event.getSource();
41          if(object == buttonQuit)
42           System.exit(0);
43          if(object == buttonB)
44           bemessen();
45        }
46      }
47
48      void bemessen()
49      {
50      }
51
52      double fck(String strBG)
53      {
54      }
55
56      double sigma_stahl(double es1)
57      {
58      }
59
60      double runden(double d, int iStellen)
61      {
62      }
63    }
```

**Listing:** 5.1

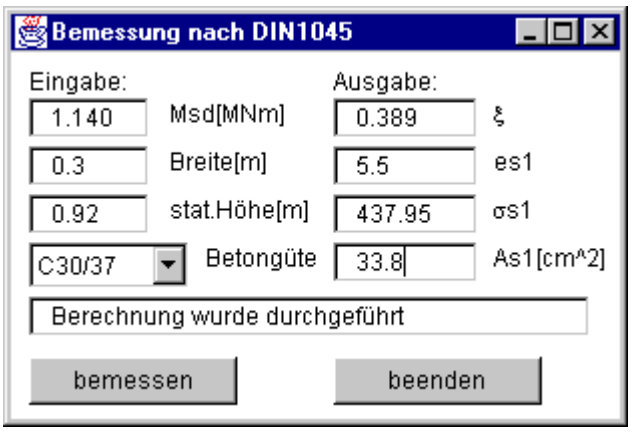


**Abbildung** 5.1**:** Das Bemessungsprogramm

Im Folgenden ist die zentrale Funktion zur Bemessung nach DIN 1045-1 (05.00) aufgelistet. Die vollständigen Quellcodes aller in diesem Artikel besprochenen Beispiele können von [4] geladen werden.

```
1       void bemessen()
2       {
3         String bg;           //Objekt für Betongüte
4         double b;            //Breite
5         double d;            //statische Nutzhöhe
6         double Msd;          //Bemessungswert des Biegemoments
7         double Msds;         //          - " -
8         double fck;          //charakteristische Zylinderdruckfest.
9         double fcd;          //Bemessungswert der Zylinderdruckfest.
10        double mysds;        //bezogenes Biegemoment
11        double ec2;          //Betondehnung
12        double alpha_R;      //Völligkeitsbeiwert
13        double k_a;          //Beiwert
14        double xi;           //bezogene Druckzonenhöhe
15        double xi_dach;      //Grenzwert der bezogenen Druckzonenhöhe
16        double es1;          //Betonstahldehnung
17        double sigma_s1;     //Betonstahlspannung
18        double As1;          //Betonstahlquerschnitt
19        double x;            //Druckzonenhöhe
20        double z;            //innerer Hebelarm
21
22        editStat.setText("");      //Statusfeld leeren
23        bg  = choiceBG.getSelectedItem().toString();
24        b   = Double.valueOf(editB.getText()).doubleValue();
25        d   = Double.valueOf(editSH.getText()).doubleValue();
26        Msd       = Double.valueOf(editMsd.getText()).doubleValue();
27        Msds      = Msd;
28        fck       = fck(bg);
29        fcd       = fck/1.5*0.85;
30
31        //Gültigkeit der Werte prüfen
32        if(b<=0.0 || d<=0.0 || Msd<=0.0)
33        {
34          editStat.setText("Eingabe-Werte ungültig");
35          return;
36        }
37
38        mysds     = Msds/(b*d*d*fcd);
39        ec2       = -0.0035;
40        alpha_R   = 0.81;
41        k_a       = 0.416;
42        xi_dach = ec2/(ec2-0.025); //Grenzwert für Randbedingung
43                                   //(ausgenutzte Betondruckzone)
44        if(1-4*k_a*mysds/alpha_R < 0)
45        {
46          editStat.setText("Bemessung nicht möglich");
47          return;
48        }
49
50        xi  = 1.0/2.0/k_a*(1-Math.sqrt(1-4*k_a*mysds/alpha_R));
51        x   = xi*d;
52        z   = d-k_a*x;
53        es1       = ec2*(1-1/xi);
54        sigma_s1= sigma_stahl(es1);
55        As1       = 1/sigma_s1*(Msds/z);
56        As1    *= 10000;      //vom m^2 nach cm^2
57
58        if(xi<xi_dach) //Betondruckzone nicht vollständig ausgenutzt
59        {
60          editStat.setText("Bemessung nicht möglich; Randbedingung!");
61          return;
62        }
```

```
63
64        //Ergebnisse in Feldern schreiben
65        edit_es1.setText(Double.toString(runden(es1*1000, 2)));
66        editXi.setText(Double.toString(runden(xi, 3)));
67        editSigma.setText(Double.toString(runden(sigma_s1, 2)));
68        editAs1.setText(Double.toString(runden(As1, 1)));
69        editStat.setText("Berechnung wurde durchgeführt");
70        return;
71     }
```

**Listing:** 5.2

# 6 Anmerkung

Ausgehend von der einfachen 'HelloWorld'-Anwendung wurden im Rahmen dieses Beitrags auch weiterführende Java-Programmiertechniken vorgeführt. Für eine vertiefte Einarbeitung in die Programmiersprache wird auf die im Literaturverzeichnis angegebenen Stellen verwiesen.

# 7 Literatur


[1] DIN 1045-1 (05.00): Tragwerke aus Beton, Stahlbeton und Spannbeton. Teil 1: Bemessung und Konstruktion. Entwurf Juli 2000

[2] JavaSoft: JDKTM 1.2.2 Documentation, Sun Microsystems, Inc., Palo Alto, USA 1999. http://java.sun.com/products/jdk/1.2/download-windows.html

[3] Your First Cup of Java (for Win32): Detailed Instructions for Your First Program, http://www.javasoft.com/docs/books/tutorial/getStarted/cupojava/win32.html

[4] Romberg, Richard: Lehrstuhl für Bauinformatik, TU München, http://www.inf.bauwesen.tu-muenchen.de/personen/richard/richard.htm

[5] Borkner-Delcarlo, Olav: Umstieg auf Java. Carl Hanser Verlag, München, 1998

[6] Flanagan, David:: Java in a Nutshell. O' Reilly & Associates, Inc, 1996

[7] Steyer, Ralph: JAVA 1.2 Kompendium. Markt&Technik, München, 1998