Kapitel 3 Aufbau eines SuM-Programms

In diesem Kapitel sollen Sie lernen:

- wie man ein SuM-Programm aus einer Vorlage erzeugt
- wie man in einem Programm Bibliotheksklassen zur Verfügung stellt
- wie man in einem Programm ein Objekt erzeugt
- wie man in einem Programm die Dienste eines Objekts benutzt
- wie man Objekte freigibt
- wie man ein Programm übersetzt und startet
- wie man ein Programm verändert
- wie man Fehlermeldungen beim Übersetzen auswertet

Nachdem Sie im letzten Kapitel Objekte "per Hand" erzeugt haben und ihre Dienste benutzt haben, sollen Sie in diesem Kapitel lernen, wie man diese Vorgänge automatisiert. Sie werden ein kleines Java-Programm schreiben, testen und verbessern. Dazu wird Ihnen eine Vorlage (Template) bereitgestellt, die die immer wieder benötigten Programmteile schon enthält. So müssen Sie nur die Vorlage an Ihre Wünsche anpassen.

3.1 Erzeugen einer Programm-Vorlage

Starten Sie BlueJ und erzeugen Sie ein neues Projekt mit dem Namen `Zwei`. Klicken Sie den Knopf `Neue Klasse`. Tragen Sie als Klassenname `MeinProgramm` ein und wählen Sie den Radioknopf `SuMKern` aus.

Abbildung 3.1: Erzeugung eines SuMKern-Programms

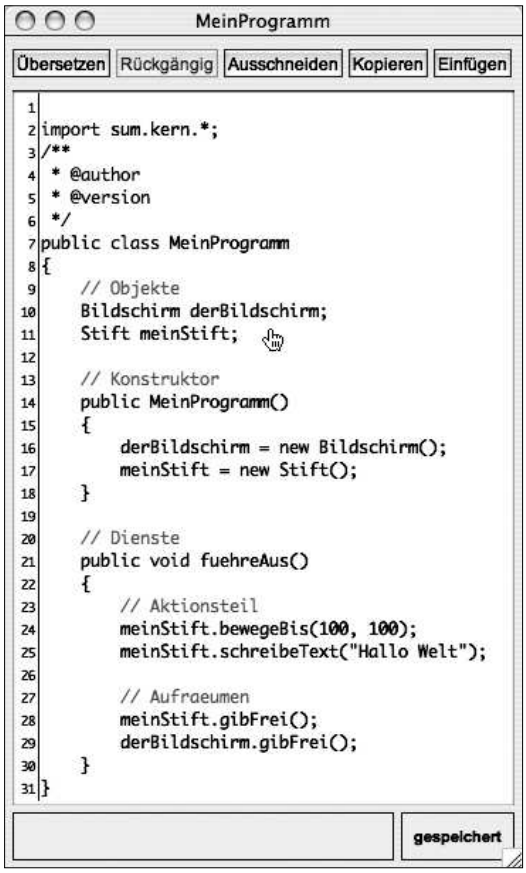
Bestätigen Sie mit Ok und im Projektfenster erscheint ein neues Symbol mit dem Namen `MeinProgramm`. Für Java ist auch ein Programm eine Klasse, deshalb beginnt `MeinProgramm` mit einem Großbuchstaben.

Abbildung 3.2:
Klassensymbol der
Klasse MeinProgramm

Das Symbol steht für die Klasse `MeinProgramm`. Die diagonale Schraffur bedeutet, dass diese Klasse noch übersetzt werden muss. Klicken Sie in den Knopf `Übersetzen` und die Schraffur verschwindet. Klicken Sie mit der rechten Maustaste (Mac ctrl-Klick) in das Symbol und ein Kontextmenü klappt auf. Wählen Sie `Bearbeiten`.

Abbildung 3.3:
Kontextmenü zur
Klasse MeinProgramm

Jetzt öffnet sich ein Fenster mit dem Programmtext.




```
1
2  import sum.kern.*;
3  /**
4   * @author
5   * @version
6   */
7  public class MeinProgramm
8  {
9      // Objekte
10     Bildschirm derBildschirm;
11     Stift meinStift;
12
13     // Konstruktor
14     public MeinProgramm()
15     {
16         derBildschirm = new Bildschirm();
17         meinStift = new Stift();
18     }
19
20     // Dienste
21     public void fuehreAus()
22     {
23         // Aktionsteil
24         meinStift.bewegeBis(100, 100);
25         meinStift.schreibeText("Hallo Welt");
26
27         // Aufraeumen
28         meinStift.gibFrei();
29         derBildschirm.gibFrei();
30     }
31 }
```

Abbildung 3.4:
Programmtext der
Klasse MeinProgramm

Dieses Fenster nennt man *Editor*. Der Editor dient dazu, Programmtext zu verändern. Im Prinzip ist der Editor eine einfache Textverarbeitung. Er kann aber auch mehr, z.B. werden bestimmte Worte wie `import` farbig hervorgehoben. In Kapitel 1 haben Sie in den Einstellungen `Zeilennummern anzeigen` ausgewählt, deshalb sehen Sie links die Zei-

lennummerierung.

Sie sehen, dass der Programmtext eine Struktur aufweist. Es gibt Einrückungen und Leerzeilen. Zu jedem Symbol geschweifte Klammer auf '{' gibt es das entsprechende Symbol '}'. Diese formale Struktur erhöht die Lesbarkeit eines Programms und dadurch wird es für eine andere Person einfacher, das Programm zu verstehen und eventuell zu verändern. Versuchen Sie von Beginn an, diese Struktur in Ihren Programmtexten zu erzeugen, die äußere Form ist ein nicht unwichtiges Bewertungskriterium für Programme.

3.2 Programmaufbau

Hinter das Wort `@author` sollten Sie Ihren Namen schreiben und hinter das Wort `@version` das aktuelle Datum. So kann man später gedruckte oder kopierte Texte einem Autoren zuordnen.

In Zeile 2 sehen Sie die sogenannte import-Anweisung. Sie sorgt dafür, dass im restlichen Programmtext die Klassen des Bibliothekspakets `sum.kern` zur Verfügung stehen. Das Sternsymbol * bedeutet dabei: alle Klassen des Pakets. Man hätte stattdessen auch zwei import-Anweisungen schreiben können:

```
import sum.kern.Bildschirm;
import sum.kern.Stift;
```

So werden nur bestimmte Klassen (Bildschirm und Stift) des Pakets importiert.

`public class MeinProgramm` bedeutet, dass `MeinProgramm` eine Klasse ist. Das Schlüsselwort `public` bedeutet, dass diese Klasse von anderen Stellen angesprochen werden kann, also öffentlich ist. Sie werden später den Dienst `fuehreAus` der Klasse `SuMProgramm` aufrufen. Die beiden geschweiften Klammern in Zeile 8 und 31 geben an, wo die Klassenbeschreibung beginnt und endet.

Jede Klasse besitzt einen *Konstruktor*. Das ist ein besonderer Dienst, der aufgerufen wird, wenn ein Objekt dieser Klasse erzeugt wird. Im Konstruktor der Klasse `MeinProgramm` werden die beiden Objekte `derBildschirm` und `meinStift` erzeugt. Man könnte den Konstruktor auch als *Initialisierungsteil* bezeichnen. Der Konstruktor ist öffentlich (`public`) und besitzt den gleichen Bezeichner wie die Klasse. Mehr über Konstruktoren werden Sie in Kapitel 6 erfahren.

Die Klasse `MeinProgramm` besitzt nur einen Dienst, den Auftrag `fuehreAus`. Auch dieser Dienst ist mit `public` als öffentlich markiert. Das Wort `void` bedeutet, dass es sich hier um einen Auftrag handelt. Die beiden geschweiften Klammern in Zeile 22 und 30 zeigen den Anfang und das Ende des Dienstes an.

Die beiden Schrägstriche in den Zeilen 9, 13, 20, 23 und 27 bedeuten, dass der Rest der Zeile ein Kommentar ist, der dem Leser das Verständnis des Programmtexts erleichtern soll. Allgemein gilt: **Sparen Sie nicht an Kommentaren!**

In Zeile 10 und 11 werden zwei Objekte deklariert. Dabei wird zuerst der Klassenname und danach der Objektname angegeben.

Im Konstruktor (Initialisierungsteil) werden die beiden Objekte erzeugt. Es wird jeweils

ein Konstruktor der entsprechenden Klasse mit dem Schlüsselwort `new` aufgerufen. Das Gleichheitszeichen "=" nennt man den *Zuweisungsoperator* und man liest ihn als *wird zu* (nicht gleich). Die mit `new` erzeugten Objekte werden also mit dem Zuweisungsoperator den entsprechenden Objektbezeichnern zugewiesen. Beachten Sie die Semikolons am Ende der Zeilen.

Im Aktionsteil werden dem Objekt `meinStift` zwei Aufträge geschickt (in Punktnotation). Beachten Sie die Anführungszeichen in Zeile 25. Einen solchen Parameter bezeichnet man als Zeichenkette bzw. `String`. Beachten Sie auch hier die Semikolons am Ende der Zeilen.

Guter Stil ist es, am Ende des Programms den benutzten Speicherbereich wieder freizugeben. Man schickt den Objekten die Anweisung `gibFrei()`.

Der Auftrag `derBildschirm.gibFrei()` sollte immer die **letzte** Anweisung in einem SuM-Kern-Programm sein. Sie sorgt dafür, dass das Bildschirmfenster stehen bleibt und als Titel *Das SuM-Programm ist beendet* erhält. Bei einem Klick auf das Fenster verschwindet es dann. Falls man diese Anweisung vergisst, verschwindet das Fenster sofort und Sie können sich die Zeichnung nicht mehr ansehen. Beachten Sie die Semikolons am Ende der Zeilen.

3.3 Programmausführung

Klicken Sie in den Knopf `Übersetzen` des Editors und das Programm wird erneut in den sogenannten Java-Bytecode übersetzt. Ein übersetztes Programm kann vom Rechner ausgeführt werden. Dazu schließen Sie den Editor und klicken mit der rechten Maustaste (Mac ctrl-Klick) auf das Klassensymbol `MeinProgramm`. Wählen Sie `new MeinProgramm()`. Es erscheint ein Dialog, um den Namen des neuen Objekts der Klasse `MeinProgramm` einzugeben. Benutzen Sie den vorgeschlagenen Namen `meinProg1`. Sie sehen unten in der Werkbank (engl. work bench), dass Sie ein neues Objekt erzeugt haben.

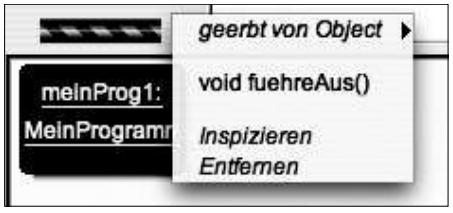


Abbildung 3.5:
Aufruf des Dienstes fuehreAus

Mit einem Rechtsklick auf dieses Objekt können Sie den Dienst `fuehreAus` aufrufen. Nach kurzer Zeit öffnet sich das Fenster `SuM-Programm`, das den Text "Hallo Welt" enthält. Wenn Sie in das Fenster klicken, verschwindet es wieder und auch das Objekt wird automatisch aus der Werkbank entfernt.

3.4 Fehlersuche

Ein großes Problem für Programmieranfänger in Java ist die Fehlersuche. Auch hierbei wird man von BlueJ sehr gut unterstützt. Dies sollen Sie an mehreren absichtlich erzeugten Fehlern kennenlernen.

Öffnen Sie dazu den Editor mit einem Doppelklick auf das Klassensymbol `MeinPro-`

gramm oder klicken Sie mit der rechten Maustaste (Mac ctrl-Klick) in das Klassensymbol und wählen Sie Bearbeiten. Ändern Sie in Zeile 2 das Wort import in Import. Java unterscheidet zwischen Groß- und Kleinschrift (im Gegensatz zu Pascal). Haben Sie bemerkt, dass die Farbe der Schrift sich geändert hat? Import ist kein Schlüsselwort für Java. Klicken Sie anschließend in den Knopf Übersetzen. Sie erhalten unten im Editor die Meldung:

Abbildung 3.6: Fehlermeldung beim Übersetzen

Diese Fehlermeldung bedeutet, das Schlüsselwort class oder interface wurde erwartet, aber Import wurde vorgefunden. Die Fehlermeldung ist knapp und nicht sehr aussagekräftig.

Klicken Sie jetzt in das Fragezeichen unten rechts im Editorfenster und Sie erhalten eine umfangreichere deutsche Fehlerbeschreibung:

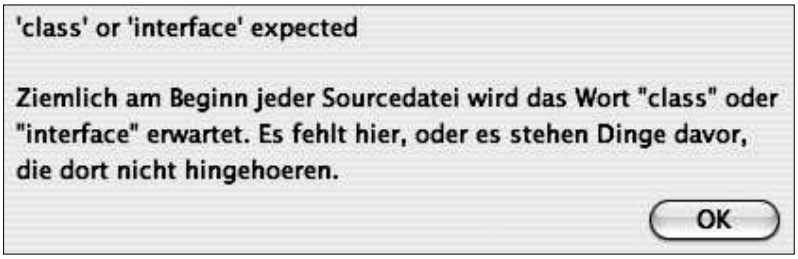


Abbildung 3.7: Ausführliche Fehlermeldung

Leider weist diese Meldung auch noch nicht auf den falschen Großbuchstaben hin, ist aber schon bedeutend aufschlussreicher. In den folgenden Übungen sollen Sie die Fehlermeldungen für anfängertypische Fehler untersuchen:

Übung 3.1 Ändern Sie sum.kern.* in sum.kerne.* und rufen Sie die ausführliche Fehlermeldung auf.

Übung 3.2 Entfernen Sie das Semikolon hinter sum.kern.* und rufen Sie die ausführliche Fehlermeldung auf.

Übung 3.3 Entfernen Sie eine geschweifte Klammer '{' und rufen Sie die ausführliche Fehlermeldung auf.

Übung 3.4 Entfernen Sie eine geschweifte Klammer '}' und rufen Sie die ausführliche Fehlermeldung auf.

Wenn Sie Dienste in ihrer Schreibweise verändern, z.B. gibFrei in gibtFrei, sehen Sie, dass es auch für bestimmte Fehler (noch) keine ausführlichen Fehlererläuterungen gibt.

Hinweis: Prüfen Sie bei Fehlermeldungen zuerst, ob Sie die Worte richtig geschrieben haben. Oft ist die Groß- Kleinschrift falsch, Worte werden getrennt statt zusammen geschrieben (gibFrei ist korrekt, gib Frei ist falsch). Oft wird ein Semikolon am Ende einer Zeile vergessen oder fälschlicherweise geschrieben.

Regel: Vor dem Zeichen '{' und nach dem Zeichen '}' darf **kein** Semikolon stehen.

3.5 Programmgesteuertes Zeichnen

Jetzt sollen Sie das vorgegebene Programm verändern und testen.

Übung 3.5 Ändern Sie den Programmtext so ab, dass die Zeichnung aus Abbildung 3.8 im Bildschirmfenster entsteht.

Hinweis: Versuchen Sie schrittweise vorzugehen und testen Sie zwischendurch die Ergebnisse. Vergessen Sie nicht den Stift hoch oder runter zu setzen. Sparen Sie nicht an Kommentaren!!!

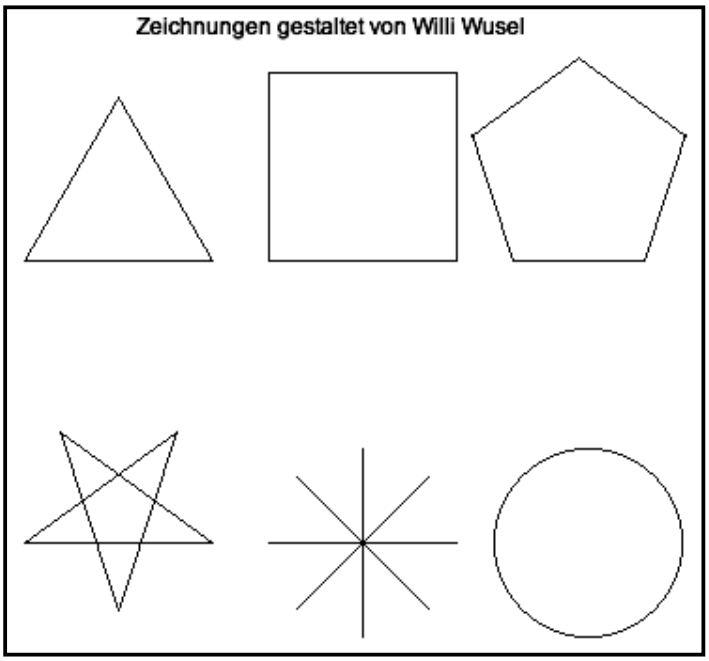


Abbildung 3.8: Programmgesteuertes Zeichnen

Hinweis: Manchmal passiert es, dass Sie das Programm erneut übersetzen wollen, während es noch im Hintergrund läuft. Sie können ein laufendes Programm immer beenden, indem Sie mit der rechten Maustaste (Mac ctrl-Klick) auf den Drehbalken (Barberpole) unten links im Projektfenster klicken.

Abbildung 3.9: Abbrechen eines laufenden Programms

3.6 Zusammenfassung

In diesem Kapitel haben Sie den Aufbau eines einfachen SuM-Kern-Programms kennen gelernt. Zu Beginn des Programm sollte der Autor / die Autoren und das Datum eingetragen werden. Die Bibliothek `sum.kern` muss importiert werden, * bedeutet dabei, dass alle Klassen des Pakets `sum.kern` importiert werden. Danach wird die Klasse `MeinProgramm` definiert. Die Klassendefinition endet mit der letzten Zeile 31, wie man an der geschweiften Klammer '}' erkennt. In dieser Klasse werden zuerst die sogenannten

Bezugsobjekte deklariert. Die Klasse enthält neben dem Konstruktor einen einzigen Dienst `fuehreAus`. Der Konstruktor enthält den Initialisierungsteil des Programms. Der Dienst `fuehreAus` enthält die beiden weiteren Teile eines jeden Programms: *Aktionsteil* und *Aufräumen*. Kommentare erkennen Sie an '//'. Bei den einzelnen Anweisungen wird die Punktnotation benutzt. Am Ende jeder Zeile folgt ein Semikolon ';'. Vor '{' und nach '}' darf kein Semikolon stehen. Die letzte Programmanweisung ist immer `derBildschirm.gib-Frei()`.

Sie haben auch gelernt, wie man ein Programm verändert, übersetzt, Fehler sucht, das Programm ausführt und zur Not abbricht. Vor der Programmausführung wird ein Programm übersetzt. Zuerst wird mit `new MeinProgramm()` ein Objekt der Klasse `MeinProgramm` erzeugt. Das Programm wird dann mit dem Aufruf des Dienstes `fuehreAus` des Objekts gestartet.

Programme kann man zur Not mit einem Rechtsklick auf den *Barberpole* abbrechen.

Programme kann man im Editorfenster verändern. Der Editor wird mit einem Rechtsklick oder einem Doppelklick auf das Klassensymbol im Projektfenster geöffnet.

Beim Übersetzen eines Programms werden eventuell Fehlermeldungen unten im Editor angezeigt. Durch Klick in das Fragezeichen erhält man eine ausführliche Fehlermeldung.

Neue Begriffe in diesem Kapitel

• **Programm-Vorlage (Template)** Viele Anweisungen kommen in jedem Programm vor. Deshalb bietet BlueJ die Möglichkeit diesen Text in eine Vorlage zu schreiben, die dann automatisch erzeugt wird. Dadurch vermeidet man Fehler und muss nicht soviel schreiben.

• **Übersetzen** Ein Javaprogramm kann erst ausgeführt werden, wenn es in den sogenannten Bytecode (für uns nicht lesbar) übersetzt wurde. Dieser Bytecode wird dann von einem Programm, der sogenannten Virtuellen Maschine (VM) ausgeführt.

• **Editor** Spezielles Textverarbeitungsprogramm mit Unterstützung der Javasyntax durch Färbung bestimmter Worte. Im Editor schreibt und ändert man Programme.

• **Einrücken** Allgemein übliche Methode zur optischen Verdeutlichung der Programmstruktur. Sollte unbedingt eingehalten werden.
• **Kommentar** Eine Kommentarzeile beginnt mit einem Doppelslash '//'. Kommentare werden bei der Programmübersetzung ignoriert und dienen nur der besseren Lesbarkeit. Sparen Sie nicht an Kommentaren!

• **Fehlermeldung** Beim Übersetzen werden oft Schreib- und Syntaxfehler erkannt. BlueJ versucht die passenden Fehlermeldungen auszugeben. Es gibt die Möglichkeit, Fehlermeldungen ausführlich anzuzeigen.

• **Barberpole** Balken unten links im Projektfenster. Wenn er rot schraffiert ist, zeigt er an, dass gerade ein Programm ausgeführt oder übersetzt wird. In Amerika war ein senkrechter Barberpole auf der Straße das Symbol für einen Friseurladen (barber shop).

Java-Bezeichner

• **import** Anweisung, um Klassen zur Verfügung zu stellen, die in einer Bibliothek im Bytecode zur Verfügung stehen. Der Quelltext ist dazu nicht notwendig.

• **public** öffentlich, Markierung für Klassen oder Dienste, die von "außen" benutzt werden sollen.

• **class** Schlüsselwort für die Deklaration einer Klasse.

• **new** Schlüsselwort zur Erzeugung eines neuen Objekts. Dahinter muss der Konstruktor einer Klasse aufgerufen werden.

• **'='** Zuweisung, wird als *wird zu* ausgesprochen. Der Bezeichner links erhält den Wert des Ausdrucks rechts.