

Benutzungshandbuch

Terminal-Simulator

Version 1.0 (23.10.2009)

Dietrich Boles

Universität Oldenburg

Inhaltsverzeichnis

1	Einleitung	7
1.1	Terminal	7
1.2	Der Terminal-Simulator	7
1.3	Voraussetzungen.....	8
1.4	Anmerkungen	8
1.5	Aufbau des Benutzerhandbuch	8
2	Installation.....	9
2.1	Voraussetzungen.....	9
2.2	Download, Installation und Start.....	9
3	Grundlagen der Terminal-Miniprogrammierwelt.....	10
3.1	Terminal	10
3.2	Befehle	11
3.3	Terminal-Programme.....	12
3.4	Beispielprogramm.....	12
3.5	Aufgaben	14
4	Ihr erstes Terminal-Programm	15
4.1	Ausprobieren der Terminal-Befehle.....	16
4.2	Gestaltung des Terminals.....	17
4.3	Eingeben eines Terminal-Programms	17
4.4	Compilieren eines Terminal-Programms	18
4.5	Ausführen eines Terminal-Programms	20

4.6	Debuggen eines Terminal-Programms.....	20
4.7	Zusammenfassung	22
5	Bedienung des Terminal-Simulators	23
5.1	Grundfunktionen	24
5.1.1	Anklicken	24
5.1.2	Tooltipps	24
5.1.3	Button	24
5.1.4	Menü.....	25
5.1.5	Toolbar.....	25
5.1.6	Popup-Menü	26
5.1.7	Eingabefeld	26
5.1.8	Dialogbox.....	26
5.1.9	Dateiauswahl-Dialogbox	27
5.1.10	Elementbaum	28
5.1.11	Split-Pane	28
5.2	Verwalten und Editieren von Terminal-Programmen	29
5.2.1	Schreiben eines neuen Terminal-Programms.....	30
5.2.2	Ändern des aktuellen Terminal-Programms.....	30
5.2.3	Löschen des aktuellen Terminal-Programm	30
5.2.4	Abspeichern des aktuellen Terminal-Programms	30
5.2.5	Öffnen eines einmal abgespeicherten Terminal-Programms	30
5.2.6	Drucken eines Terminal-Programms	31
5.2.7	Editier-Funktionen.....	31
5.3	Compilieren von Terminal-Programmen.....	32

5.3.1	Compilieren.....	32
5.3.2	Beseitigen von Fehlern	32
5.4	Verwalten und Gestalten von Terminalern	33
5.4.1	Größe des Terminals ändern	34
5.4.2	Hintergrundfarbe ändern.....	35
5.4.3	Schriftfarbe ändern	35
5.4.4	Löschen des Terminals.....	35
5.4.5	Abspeichern des Terminals	35
5.4.6	Wiederherstellen eines abgespeicherten Terminals	35
5.5	Interaktives Ausführen von Terminal-Befehlen.....	35
5.5.1	Befehlsfenster	36
5.5.2	Parameter	37
5.5.3	Rückgabewerte von Funktionen.....	37
5.6	Ausführen von Terminal-Programmen.....	37
5.6.1	Starten eines Terminal-Programms	37
5.6.2	Stoppen eines Terminal-Programms	38
5.6.3	Pausieren eines Terminal-Programms.....	38
5.6.4	Während der Ausführung eines Terminal-Programms.....	38
5.6.5	Einstellen der Geschwindigkeit	38
5.6.6	Wiederherstellen einer Zeichenfläche.....	38
5.7	Debuggen von Terminal-Programmen	38
5.7.1	Beobachten der Programmausführung	39
5.7.2	Schrittweise Programmausführung.....	40
5.7.3	Breakpoints	40

5.7.4	Debugger-Fenster	41
5.8	Konsole	42
6	Beispielprogramme	43
6.1	Treppe	43
6.2	Schneeflocke	44
6.3	Tannenbaum	45
6.4	Dreieck	46
6.5	Pfeil	47
7	Literatur zum Erlernen der Programmierung	49

1 Einleitung

Miniprogrammierwelten sind spezielle Entwicklungsumgebungen für Programmieranfänger. Sie bestehen aus einer überschaubaren Menge von Befehlen, die ein Programmieranfänger nutzen kann, um hiermit bestimmte ihm gestellte Aufgaben zu lösen. Die Ausführung der einzelnen Befehle bewirkt dabei jeweils eine bestimmte visuelle Änderung auf dem Bildschirm. Dadurch können Programmieranfänger sehr gut nachvollziehen, was ihr Programm tut. Fehler lassen sich sehr schnell entdecken und beheben.

1.1 Terminal

Im Terminal-Simulator wird ein Terminal simuliert. Mittels spezieller Befehle können hier Daten ausgegeben bzw. vom Benutzer eingegeben werden.

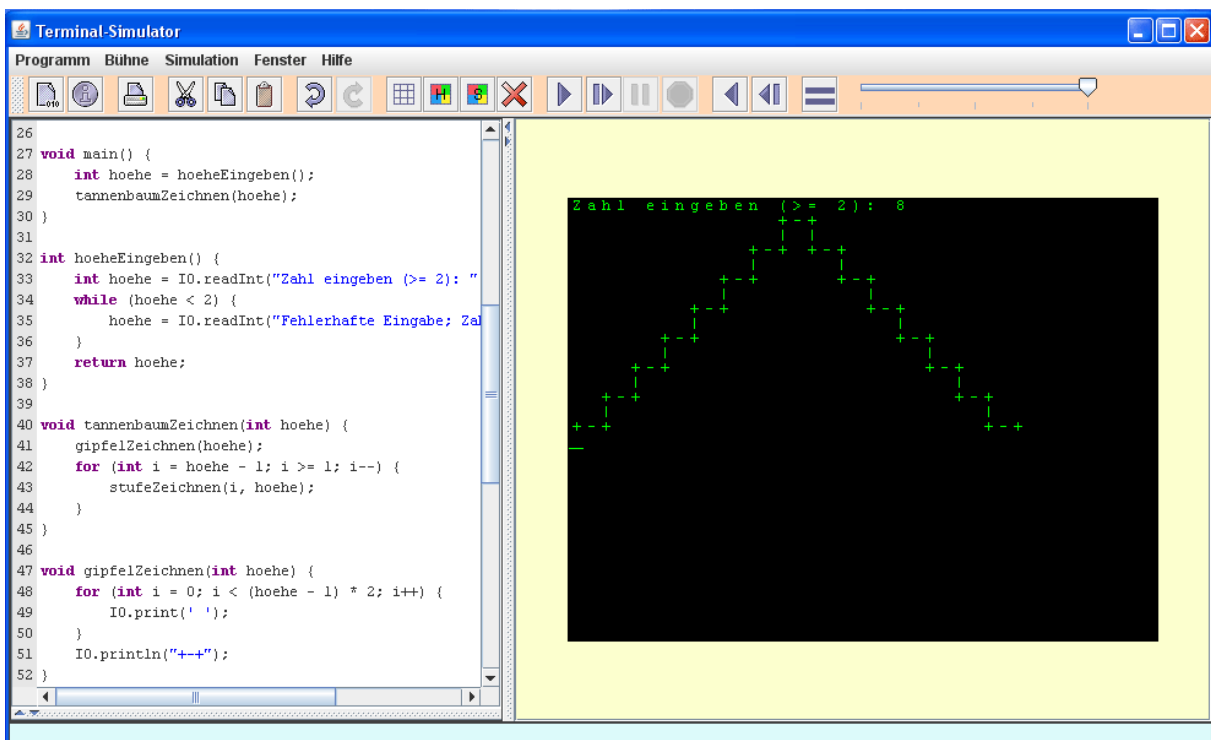


Abb. 1.1: Beispiel eines Terminals mit ausgegebenen Daten

1.2 Der Terminal-Simulator

Bei den Miniprogrammierwelten steht nicht so sehr das "Learning-by-Listening" bzw. „Learning-by-Reading“ im Vordergrund, sondern vielmehr das „Learning-by-Doing“, also das praktische Üben. Genau das unterstützt der Terminal-Simulator. Dieser stellt eine Reihe von Werkzeugen zum Erstellen und Ausführen von Terminal-Programmen zur Verfügung: einen Editor zum Eingeben und Verwalten von Terminal-Programmen, einen Compiler zum Übersetzen von Terminal-Programmen, einen Territoriumsgestalter zum Gestalten und Verwalten von Terminals, einen

Interpreter zum Ausführen von Terminal-Programmen und einen Debugger zum Testen von Terminal-Programmen. Der Terminal-Simulator ist einfach zu bedienen, wurde aber funktional und bedienungsmäßig bewusst an professionelle Entwicklungsumgebungen für Java (z.B. Eclipse) angelehnt, um Programmieranfängern einen späteren Umstieg auf diese zu erleichtern.

1.3 Voraussetzungen

Zielgruppe von Miniprogrammierwelten sind Schüler oder Studierende ohne oder mit nur wenig Programmiererfahrung, die die Grundlagen der (imperativen) Programmierung erlernen und dabei ein wenig Spaß haben wollen. Kenntnisse im Umgang mit Computern sind wünschenswert. Der Terminal-Simulator ist dabei kein Lehrbuch sondern ein Werkzeug, das das Erlernen der Programmierung unterstützt. Auf gute begleitende Lehrbücher zum Erlernen der Programmierung wird in Kapitel 6 (Literatur zum Programmieren lernen) hingewiesen.

1.4 Anmerkungen

Der Terminal-Simulator wurde mit dem Tool „Solist“ erstellt. Solist ist eine spezielle Entwicklungsumgebung für Miniprogrammierwelt-Simulatoren. Solist ist ein Werkzeug der Werkzeugfamilie des „Programmier-Theaters“, eine Menge von Werkzeugen zum Erlernen der Programmierung. Metapher aller dieser Werkzeuge ist die Theaterwelt. Schauspieler (im Falle von Solist „Solisten“) agieren auf einer Bühne, auf der zusätzlich Requisiten platziert werden können. Eine Aufführung entspricht der Ausführung eines Programms.

Im Falle des Terminal-Simulators ist die Bühne ein Terminal, in der ein (unsichtbarer) Schreiber – der Solist – agiert. Er schreibt Daten auf den Bildschirm und fordert den Benutzer zum Eingeben von Daten auf. Wenn Ihnen also beim Umgang mit dem Terminal-Simulator der Begriff „Solist“ begegnet, kennen Sie nun hiermit den Hintergrund dieser Namenswahl.

Mehr Informationen zu Solist finden Sie im WWW unter www.programmierkurs-java.de/solist.

1.5 Aufbau des Benutzerhandbuch

Das Benutzungshandbuch ist in 6 Kapitel gegliedert. Nach dieser Einleitung wird in Kapitel 2 die Installation des Terminal-Simulators beschrieben. Kapitel 3 erläutert die Grundlagen der Terminal-Miniprogrammierwelt. Wie Sie Ihr erstes Terminal-Programm zum Laufen bringen, erfahren Sie kurz und knapp in Kapitel 4. Dieses enthält eine knappe Einführung in die Elemente und die Bedienung des Terminal-Simulators. Sehr viel ausführlicher geht dann Kapitel 5 auf die einzelnen Elemente und die Bedienung des Terminal-Simulators ein. Kapitel 6 enthält letztendlich Hinweise zu guter Begleitliteratur zum Erlernen der Programmierung mit Java.

2 Installation

2.1 Voraussetzungen

Voraussetzung zum Starten des Terminal-Simulators ist ein Java Development Kit SE (JDK) der Version 6 oder höher. Ein Java Runtime Environment SE (JRE) reicht nicht aus. Das jeweils aktuelle JDK kann über die Website <http://java.sun.com/javase/downloads/index.jsp> bezogen werden und muss anschließend installiert werden.

2.2 Download, Installation und Start

Der Terminal-Simulator kann von der Solist-Website <http://www.programmierkurs-java.de/solist> kostenlos herunter geladen werden. Er befindet sich in einer Datei namens *terminalsimulator-1.0.zip*. Diese muss zunächst entpackt werden. Es entsteht ein Ordner namens *terminalsimulator-1.0* (der so genannte *Simulator-Ordner*), in dem sich eine Datei *simulator.jar* befindet. Durch Doppelklick auf diese Datei wird der Terminal-Simulator gestartet. Alternativ lässt er sich auch durch Eingabe des Befehls `java -jar simulator.jar` in einer Console starten.

Beim ersten Start sucht der Terminal-Simulator nach der JDK-Installation auf Ihrem Computer. Sollte diese nicht gefunden werden, werden Sie aufgefordert, den entsprechenden Pfad einzugeben. Der Pfad wird anschließend in einer Datei namens *solist.properties* im Simulator-Ordner gespeichert, wo er später wieder geändert werden kann, wenn Sie bspw. eine aktuellere JDK-Version auf Ihrem Rechner installieren sollten. In der Property-Datei können Sie weiterhin die Sprache angeben, mit der der Terminal-Simulator arbeitet. In der Version 1.0 wird allerdings nur deutsch unterstützt.

3 Grundlagen der Terminal-Miniprogrammierwelt

Computer können heutzutage zum Lösen vielfältiger Aufgaben genutzt werden. Die Arbeitsanleitungen zum Bearbeiten der Aufgaben werden ihnen in Form von Programmen mitgeteilt. Diese Programme, die von Programmierern entwickelt werden, bestehen aus einer Menge von Befehlen bzw. Anweisungen, die der Computer ausführen kann. Die Entwicklung solcher Programme bezeichnet man als Programmierung.

Die Terminal-Miniprogrammierwelt ist ein spezielles didaktisches Modell zum Erlernen der Programmierung. In der Terminal-Miniprogrammierwelt nimmt ein virtueller (unsichtbarer) Schreiber die Rolle des Computers ein. Diesem Schreiber können ähnlich wie einem Computer Befehle erteilt werden, die dieser ausführt.

Ihnen als Programmierer werden nun Aufgabe gestellt, bspw. bestimmte ASCII-Zeichnungen auf dem Terminal auszugeben, wobei Sie den virtuellen Schreiber notwendige Aktionen, wie das Platzieren von Zeichen, ausführen lassen können. Ihre Programme entwickeln Sie in der Terminal-Sprache - eine Programmiersprache, die fast vollständig der Programmiersprache Java entspricht.

3.1 Terminal

Die Ausgangssituation im Terminal-Simulator wird in Abbildung 3.1 skizziert. Standardmäßig beträgt die Größe des Terminals 30 Reihen und 40 Spalten. Die Zellen haben eine Größe von 12 Pixeln. Die Zellengröße bestimmt dabei die Schriftgröße. Anfangs ist das Terminal leer und der Cursor, der den virtuellen Schreiber repräsentiert, steht in Reihe 0 und Spalte 0, also ganz oben links.

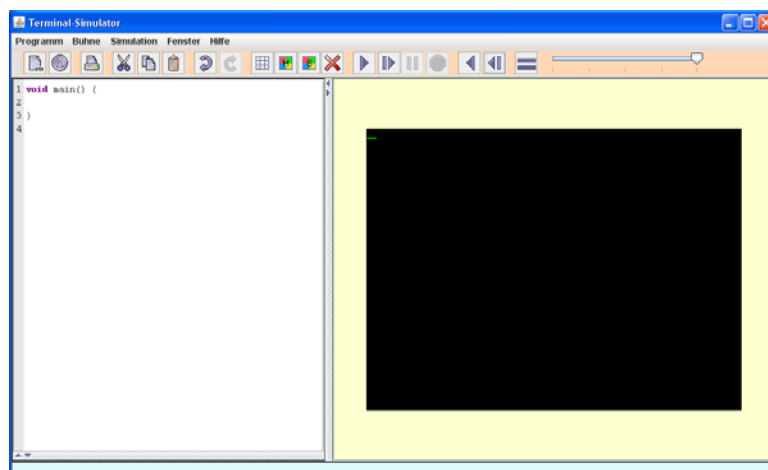


Abb. 3.1: Terminal-Simulator

3.2 Befehle

Der virtuelle Schreiber des Terminals kennt nur einige wenige Befehle. Sie können sich den Schreiber quasi als einen virtuellen Prozessor vorstellen, der im Gegensatz zu realen Computer-Prozessoren in der Lage ist, mit einem kleinen Grundvorrat an Befehlen u.a. Zeichen auf dem Terminal zu platzieren.

Die Befehle des Schreibers – im Folgenden Terminal-Befehle genannt – sind:

- `int getColumns()` : Liefert die Anzahl an Spalten des Terminals.
- `int getRows()` : Liefert die Anzahl an Reihen des Terminals.
- `void move(int row, int col)` : Verschiebt den Cursor an die angegebene Position erster Parameter = Reihe, zweiter Parameter = Spalte). Wenn ungültige Koordinaten übergeben werden, passiert nichts.
- `void clear()` : Löscht das Terminal. Die Cursorposition verändert sich nicht.
- `void print(char zeichen)` : Gibt das übergeben Zeichen aus.
- `void print(String str)` : Gibt die übergebene Zeichenkette aus.
- `void print(Object obj)` : Gibt eine String-Repräsentation des übergebenen Objektes aus.
- `void println(char zeichen)` : Gibt das übergeben Zeichen aus und verschiebt den Cursor an den Anfang der nächsten Zeile.
- `void println(String str)` : Gibt die übergebene Zeichenkette aus und verschiebt den Cursor an den Anfang der nächsten Zeile.
- `void println(Object obj)` : Gibt eine String-Repräsentation des übergebenen Objektes aus und springt in die nächste Zeile.
- `void println()` : Verschiebt den Cursor an den Anfang der nächsten Zeile.
- `String readString(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe einer Zeichenkette durch den Nutzer. Nach Drücken der Enter-Taste wird die eingegebene Zeichenkette als Wert geliefert.
- `int readInt(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines int-Wertes durch den Nutzer. Nach Drücken der Enter-Taste wird der eingegebene Wert geliefert. Wenn der Nutzer einen ungültigen int-Wert eingibt, wird 0 geliefert.
- `short readShort(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines short-Wertes durch den Nutzer. Nach Drücken der Enter-Taste wird der eingegebene Wert geliefert. Wenn der Nutzer einen ungültigen short-Wert eingibt, wird 0 geliefert.
- `long readLong(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines long-Wertes durch den Nutzer. Nach Drücken der Enter-Taste wird der eingegebene Wert geliefert. Wenn der Nutzer einen ungültigen long-Wert eingibt, wird 0L geliefert.
- `float readFloat(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines float-Wertes durch den Nutzer. Nach

Drücken der Enter-Taste wird der eingegebene Wert geliefert. Wenn der Nutzer einen ungültigen float-Wert eingibt, wird 0.0F geliefert.

- `double readDouble(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines double-Wertes durch den Nutzer. Nach Drücken der Enter-Taste wird der eingegebene Wert geliefert. Wenn der Nutzer einen ungültigen double-Wert eingibt, wird 0.0 geliefert.
- `char readChar(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines Zeichens durch den Nutzer. Nach Drücken der Enter-Taste wird das eingegebene Zeichen geliefert. Wenn der Nutzer kein Zeichen eingibt, wird '\0' geliefert.
- `boolean readBoolean(String prompt)` : Gibt die übergebene Zeichenkette aus und wartet auf die Eingabe eines boolean-Wertes (true, false) durch den Nutzer. Nach Drücken der Enter-Taste wird der eingegebene Wert geliefert. Wenn der Nutzer einen ungültigen boolean-Wert eingibt, wird 'false' geliefert.

3.3 Terminal-Programme

Terminal-Programme werden in der so genannten Terminal-Sprache geschrieben. Diese ist fast deckungsgleich mit der Programmiersprache Java. Die wichtigsten Unterschiede sind:

Anders als in Java bedarf es in der Terminal-Sprache keiner Klassen-Definition.

In der Terminal-Sprache gibt es keine Methode mit der Signatur `public static void main(String[] args)`. Stattdessen gibt es die so genannte main-Prozedur mit der Signatur `void main()`.

Terminal-Programme sind imperative, keine objektorientierten Programme. Es brauchen also keine Objekte erzeugt zu werden. Trotzdem müssen Prozeduren und Funktionen nicht als `static` deklariert werden.

Ein Terminal-Programm setzt sich aus der main-Prozedur sowie weiteren Prozeduren bzw. Funktionen zusammen, die vor oder nach der main-Prozedur definiert werden dürfen. Variablen können global, d.h. außerhalb von Prozeduren, oder lokal, d.h. innerhalb von Prozeduren, definiert werden. Der Gültigkeitsbereich von globalen Variablen erstreckt sich über das gesamte Programm, der Gültigkeitsbereich von lokalen Variablen ist auf den Prozedurrumpf beschränkt.

Der Aufruf bzw. Start eines Terminal-Programms bewirkt die automatische Ausführung der main-Prozedur.

3.4 Beispielprogramm

In folgendem Programm wird ein Tannenbaum auf das Terminal gemalt.

```
/*  
Bald ist Weihnachten und Sie möchten sich selbst mal eine
```

Freude machen, indem Sie einen ASCII-Weihnachtsbaum auf den Bildschirm Ihres Computers malen.

Aufgabe: Schreiben Sie ein Java-Programm, das den Benutzer zunächst auffordert, eine Zahl größer gleich 2 einzugeben und das anschließend einen entsprechend hohen Weihnachtsbaum der folgenden Form auf den Bildschirm ausgibt:

Höhe = 2:

```
  +-+
  | |
+-+ +-+
```

Höhe = 4:

```
      +-+
      | |
    +-+ +-+
    |   |
 +-+   +-+
 |     |
+-+     +-+
*/
```

```
void main() {
    int hoehe = hoeheEingeben();
    tannenbaumZeichnen(hoehe);
}

int hoeheEingeben() {
    int hoehe = readInt("Zahl eingeben (>= 2): ");
    while (hoehe < 2) {
        hoehe = readInt("Fehlerhafte Eingabe; Zahl eingeben (>= 2): ");
    }
    return hoehe;
}

void tannenbaumZeichnen(int hoehe) {
    gipfelZeichnen(hoehe);
    for (int i = hoehe - 1; i >= 1; i--) {
        stufeZeichnen(i, hoehe);
    }
}

void gipfelZeichnen(int hoehe) {
    for (int i = 0; i < (hoehe - 1) * 2; i++) {
        print(' ');
    }
    println("+-+");
}

void stufeZeichnen(int aktHoehe, int hoehe) {
    pipeZeileSchreiben(aktHoehe, hoehe);
    plusPipePlusZeileSchreiben(aktHoehe, hoehe);
}

void pipeZeileSchreiben(int aktHoehe, int hoehe) {
    for (int i = 0; i < aktHoehe * 2; i++) {
        print(' ');
    }
    IO.print('|');
    for (int i = 0; i <= (hoehe - aktHoehe - 1) * 4; i++) {
        print(' ');
    }
    println('|');
}
```

```

void plusPipePlusZeileSchreiben(int aktHoehe, int hoehe) {
    for (int i = 0; i < aktHoehe * 2 - 2; i++) {
        print(' ');
    }
    print("+--+");
    for (int i = 0; i <= (hoehe - aktHoehe - 1) * 4; i++) {
        print(' ');
    }
    println("+--+");
}

```

3.5 Aufgaben

Überlegen Sie sich bspw. andere Zeichnungen, die der virtuelle Schreiber auf das Terminal malen soll.

4 Ihr erstes Terminal-Programm

Nachdem Sie den Terminal-Simulator gestartet haben, öffnet sich ein Fenster, das in etwa dem in Abbildung 4.1 dargestellten Fenster entspricht. Ganz oben enthält es eine Menüleiste mit 5 Menüs, darunter eine Toolbar mit einer Reihe von Buttons und ganz unten einen Meldungsbereich, in dem Meldungen ausgegeben werden. Im linken Teil befindet sich der Editor-Bereich, im rechten Teil der Simulationsbereich. Im Editor-Bereich geben Sie Terminal-Programme ein und im Simulationsbereich führen Sie Terminal-Programme aus.

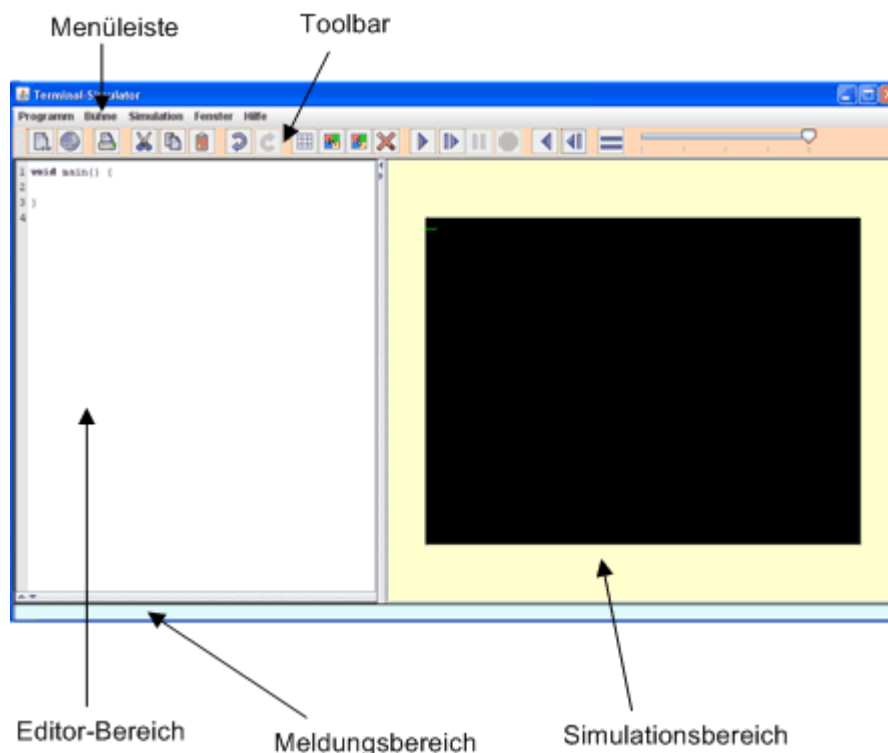


Abb. 4.1: Terminal-Simulator nach dem Öffnen

Um den Simulator ein wenig näher kennen zu lernen, empfehle ich Ihnen, als erstes zunächst die Terminal-Befehle einmal auszuprobieren. Wie das geht, wird in Abschnitt 4.1 erläutert. Anschließend wird in den darauf folgenden Abschnitten im Detail beschrieben, was Sie machen müssen, um Ihr erstes Terminal-Programm zu schreiben und auszuführen. Insgesamt müssen/können fünf Stationen durchlaufen werden:

- Gestaltung von Terminalern
- Eingeben eines Terminal-Programms
- Compilieren eines Terminal-Programms
- Ausführen eines Terminal-Programms
- Debuggen eines Terminal-Programms

4.1 Ausprobieren der Terminal-Befehle

Klicken Sie im Menü „Fenster“ das Menü-Item „Befehlsfenster“ an. Es öffnet sich ein Fenster mit dem Titel „Befehlsfenster“. In diesem Fenster erscheinen alle Befehle, die der virtuelle Schreiber kennt.

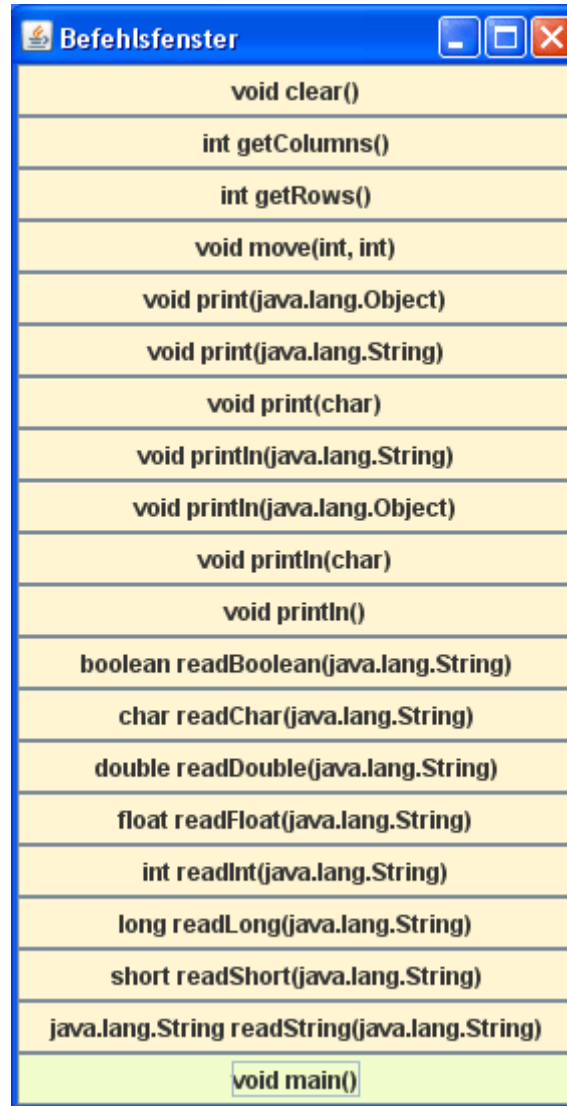


Abb. 4.2: Befehlsfenster

Sie können die jeweiligen Befehle ausführen, indem Sie den Maus-Cursor auf den entsprechenden Button verschieben und diesen anklicken. Besitzt ein Befehl Parameter, dann erscheint nach dem Anklicken des Befehls eine Dialogbox, in der die gewünschten Parameterwerte eingegeben werden können. Liefert der Befehl einen Wert, dann wird dieser ebenfalls in einer Dialogbox dargestellt.

Klicken Sie doch bspw. einfach mal den Befehl `void println(java.lang.String)` an und geben Sie in der Dialogbox die Zeichenkette „hallo“ ein. Nach dem Drücken auf den OK-Button schreibt der

virtuelle Schreiber an der aktuellen Cursor-Position die Zeichenkette „hallo“ auf das Terminal.

Wenn Sie Programme mit Prozeduren oder Funktionen schreiben und erfolgreich kompilieren, werden die Prozeduren und Funktionen übrigens auch im Befehlsfenster angezeigt und können interaktiv ausgeführt werden. Dabei werden keine Zwischenzustände im Simulationsbereich ausgegeben, sondern immer der jeweilige Endzustand nach Abarbeitung der Funktion.

4.2 Gestaltung des Terminals

Standardmäßig ist das im Simulationsbereich dargestellte Terminal 30 Reihen und 40 Spalten groß. Eine Zelle, deren Größe die Fontgröße bestimmt, misst 12 * 12 Pixel. Die Hintergrundfarbe ist schwarz, die Schriftfarbe grün. Sie können diesen Zustand jedoch ändern. Zur Gestaltung des Terminals müssen Sie die Buttons 9 – 12 (von links) in der so genannten *Toolbar* benutzen. Diese werden auch als *Gestaltungsbuttons* bezeichnet. Fahren Sie einfach mal mit der Maus über die einzelnen Buttons der Toolbar, dann erscheint jeweils ein Tooltipp, der beschreibt, wozu dieser Button dient.

Mit Hilfe des „Größe ändern“-Buttons (9. Toolbar-Button von links) können Sie die Größe des Terminals ändern. Nach dem Anklicken des Buttons öffnet sich eine Dialogbox, in der Sie die gewünschte Größe angeben können. Nach dem Drücken des OK-Buttons der Dialogbox wird im Simulationsbereich das aktuelle Terminal gelöscht und ein Terminal der entsprechenden Größe dargestellt.

Der „Hintergrundfarbe ändern“-Button (10. Toolbar-Button von links) dient zum Ändern der Hintergrundfarbe. Wenn Sie den Button anklicken, erscheint eine spezielle Dialogbox, in der Sie die gewünschte Farbe auswählen können. Auf analoge Art und Weise können Sie mit dem „Schriftfarbe ändern“-Button (11. Toolbar-Button von links) die Schriftfarbe anpassen.

Möchten Sie das Terminal löschen, so können Sie dies durch Anklicken des „Löschen“-Buttons (12. Toolbar-Button von links) tun.

Über das Menü „Bühne“ kann das aktuell dargestellte Terminal auch in einer Datei gespeichert und unter Umständen später wieder geladen werden.

4.3 Eingeben eines Terminal-Programms

Das Eingeben von Terminal-Programmen erfolgt im Editor-Bereich.

Wir klicken in den Editor-Bereich und tippen dort wie in einem normalen Editor bzw. Textverarbeitungsprogramm, wie Microsoft Word, die entsprechenden Terminal-Befehle ein:

```
void main() {
```

```
String name = readString("Wie heißt du?");

println("hallo " + name);

}
```

Das ist unser erstes Terminal-Programm.

Ihnen sicher von anderen Editoren bzw. Textverarbeitungsprogrammen bekannte Funktionen, wie „Ausschneiden“, „Kopieren“, „Einfügen“, „Rückgängig“ und „Wiederherstellen“ können Sie über das Menü „Programm“ bzw. die entsprechenden Buttons in der Toolbar ausführen (vierter bis achter Button von links).

Weiterhin gibt es im Menü „Programm“ zwei Menü-Items zum Speichern von Programmen in Dateien und zum wieder Laden von abgespeicherten Programmen. Bei ihrer Aktivierung erscheint eine Dateiauswahl-Dialogbox, in der Sie die entsprechende Auswahl der jeweiligen Datei vornehmen müssen. Achtung: Wenn Sie eine abgespeicherte Datei laden, geht der Programmtext, der sich aktuell im Editorbereich befindet, verloren (wenn er nicht zuvor in einer Datei abgespeichert wurde).

Im Menü „Programm“ finden Sie darüber hinaus weitere nützliche Funktionen (Schriftgröße ändern, Drucken, ...).

4.4 Compilieren eines Terminal-Programms

Nachdem wir unser Terminal-Programm geschrieben haben, müssen wir es compilieren. Der Compiler überprüft den Sourcecode auf syntaktische Korrektheit und transformiert ihn – wenn er korrekt ist – in ein ausführbares Programm. Zum Compilieren drücken Sie einfach auf den „Compilieren“-Button (erster Toolbar-Button von links oder „Compilieren“-Menü-Item im „Programm“-Menü). Compiliert wird dann das Programm, das gerade im Eingabebereich sichtbar ist. Es wird zuvor automatisch in einer (temporären) Datei (namens Solist.java) abgespeichert.

Wenn das Programm korrekt ist, erscheint eine Dialogbox mit der Nachricht „Compilierung erfolgreich“. Zur Bestätigung müssen Sie anschließend noch den OK-Button drücken. Das Programm kann nun ausgeführt werden. Merken Sie sich bitte: Immer, wenn Sie Änderungen am Sourcecode Ihres Programms vorgenommen haben, müssen Sie es zunächst neu kompilieren.

Wenn das Programm syntaktische Fehler enthält – wenn Sie sich bspw. bei der Eingabe des obigen Programms vertippt haben –, werden unter dem Editor-Bereich die Fehlermeldungen des Compilers eingeblendet. Diese erscheinen in englischer Sprache. Weiterhin wird die Zeile angegeben, in der der Fehler entdeckt wurde. Wenn Sie mit der Maus auf die Fehlermeldung klicken, springt der Maus-Cursor im Editor-Bereich automatisch in die angegebene Zeile (siehe Abbildung 4.3).

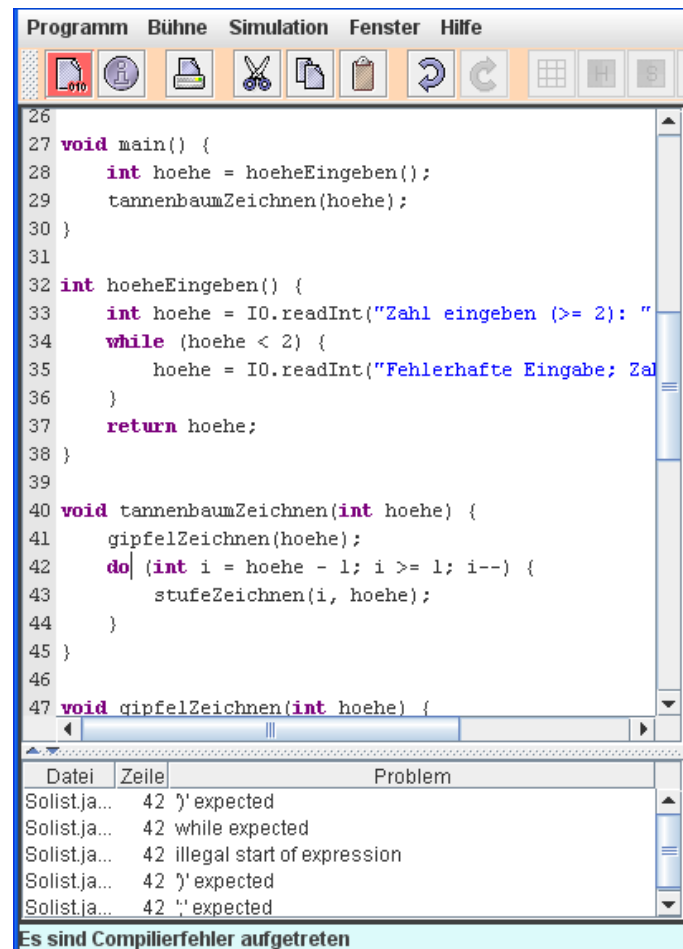


Abb. 4.3: Fehlermeldungen des Compilers

Vorsicht: Die Fehlermeldungen sowie die Zeilenangabe eines Compilers sind nicht immer wirklich exakt. Das Interpretieren der Meldungen ist für Programmieranfänger häufig nicht einfach und bedarf einiger Erfahrungen. Deshalb machen Sie ruhig am Anfang mal absichtlich Fehler und versuchen Sie, die Meldungen des Compilers zu verstehen.

Tipp: Arbeiten Sie die Fehler, die der Compiler entdeckt hat, immer von oben nach unten ab. Wenn Sie eine Meldung dann überhaupt nicht verstehen, compilieren Sie ruhig erst mal erneut. Häufig ist es (leider) so, dass der Compiler für einen einzelnen Fehler mehrere Fehlermeldungen ausgibt, was Anfänger leicht verwirren kann.

Nachdem Sie die Fehler korrigiert haben, müssen Sie das Programm erneut compilieren. Wiederholen Sie dies so lange, bis der Compiler die Meldung „Compilierung erfolgreich“ ausgibt. Erst dann können Sie das Programm ausführen!

4.5 Ausführen eines Terminal-Programms

Nach dem erfolgreichen Compilieren ist es endlich soweit: Wir können beobachten, wie Zeichen auf dem Terminal platziert werden. Macht der virtuelle Schreiber wirklich das, was wir ihm durch unser Programm beigebracht haben?

Zum Steuern der Programmausführung dienen die Buttons rechts in der Toolbar. Durch Anklicken des „Simulation starten“-Buttons (13. Button von links) starten wir das eben eingegebene und compilierte Programm. Wenn Sie bis hierhin alles richtig gemacht haben, sollten Sie aufgefordert werden, Ihren Namen einzugeben. Tun Sie dies und drücken dann die RETURN-Taste. Anschließend werden Sie dann von Ihrem Programm freundlich begrüßt. Herzlichen Glückwunsch zu Ihrem ersten Terminal-Programm!

Wollen Sie die Programmausführung anhalten, können Sie dies durch Anklicken des „Simulation pausieren“-Buttons (15. Button von links) erreichen. Der virtuelle Schreiber pausiert so lange, bis Sie wieder den „Simulation starten/fortsetzen“-Button (13. Button von links) anklicken. Dann fährt der Schreiber mit seiner Arbeit fort. Das Programm vorzeitig komplett abbrechen, können Sie mit Hilfe des „Simulation beenden“-Buttons (16. Button von links).

Wenn Sie ein Programm mehrmals hintereinander im gleichen Terminal ausführen wollen, können Sie mit dem „Rücksetzen“-Button (17. Button von links) den Zustand des Terminals wieder herstellen, der vor dem letzten Ausführen des Programms Bestand hatte. Eine komplette Zurücksetzung des Terminals auf den Zustand beim Öffnen des Terminal-Simulators ist mit dem „Komplett zurücksetzen“-Button möglich (17. Button von links).

Der Schieberegler ganz rechts in der Menüleiste dient zur Steuerung der Geschwindigkeit der Programmausführung. Je weiter Sie den Knopf nach links verschieben, umso langsamer erledigt der virtuelle Schreiber Arbeit. Je weiter Sie ihn nach rechts verschieben, umso schneller führt er Ihre Anweisungen aus.

Die Bedienelemente zum Steuern der Programmausführung finden Sie übrigens auch im Menü „Simulation“.

4.6 Debuggen eines Terminal-Programms

„Debuggen eines Programms“ eines Programms bedeutet, dass Sie bei der Ausführung eines Programms zusätzliche Möglichkeiten zur Steuerung besitzen und sich den Zustand des Programms (welche Zeile des Sourcecodes wird gerade ausgeführt, welche Werte besitzen aktuell die Variablen) in bestimmten Situationen anzeigen lassen können. Das ist ganz nützlich, wenn Ihr Programm nicht das tut, was es soll, und sie herausfinden wollen, woran der Fehler liegt.

Klicken Sie zum Debuggen zunächst den „Ablaufverfolgung aktivieren“-Button in der Toolbar an (19. Button von links). Es wird das so genannte Debugger-Fenster mit dem Titel „Debugger“ geöffnet. Im linken Bereich des Debugger-Fensters wird der

Prozedur-Stack angezeigt, das sind die aktuelle Prozedur sowie die Prozeduren, aus denen die Prozedur heraus aufgerufen wurde. Im rechten Bereich werden – falls vorhanden – die gültigen Variablen und ihre aktuellen Werte dargestellt (siehe Abb. 4.4).

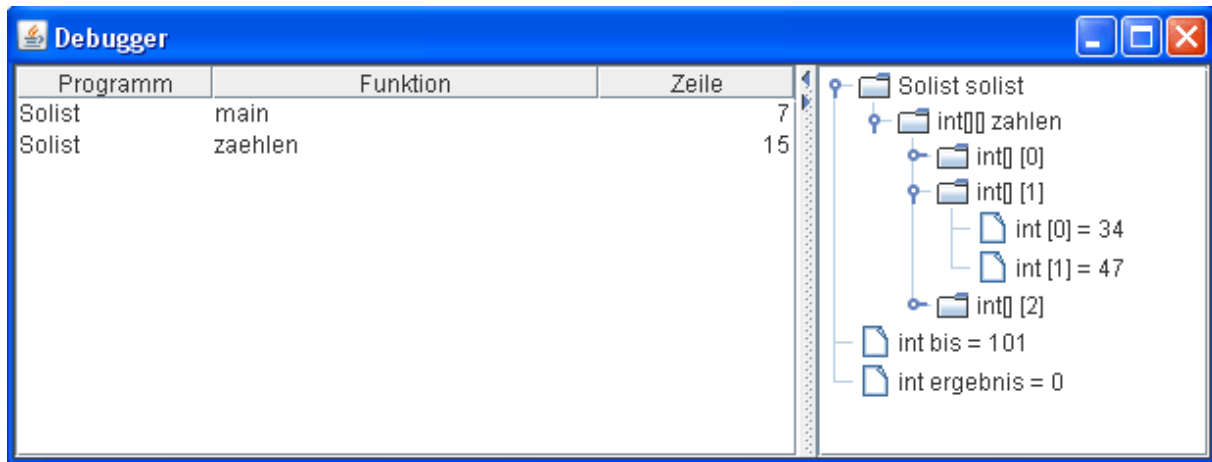


Abb. 4.4: Debugger-Fenster

Jetzt können Sie über den „Schritt“-Button (14. Toolbar-Button von links) jeden Befehl einzeln ausführen und bekommen im Editorbereich durch einen blauen Balken angezeigt, welcher Befehl bzw. welche Zeile als nächstes ausgeführt wird. Bei einem Prozeduraufruf wird dabei auch automatisch in die entsprechende Prozedur verzweigt. Im Debugger-Fenster wird zusätzlich der Prozedur-Stack angezeigt, d.h. die aktuelle Prozedur sowie die Prozeduren, aus denen die Prozedur heraus aufgerufen wurde.

Sie können zunächst auch einfach das Programm durch Anklicken des „Starten“-Buttons starten und beobachten. Wenn Sie dann den „Pause“-Button drücken, haben Sie anschließend ebenfalls die Möglichkeit der schrittweisen Ausführung ab der aktuellen Position.

Die Ablaufverfolgung kann übrigens jederzeit durch erneutes Klicken des „Ablaufverfolgung“-Buttons deaktiviert bzw. reaktiviert werden.

Wenn Sie möchten, dass der Programmablauf beim Erreichen einer bestimmten Zeile automatisch in den Pausezustand gelangt, können Sie vor oder während des Programmablaufs in der entsprechenden Zeile einen Breakpoint setzen. Führen Sie dazu im Editorbereich auf der entsprechenden Zeilennummer einen Doppelklick mit der Maus aus. Breakpoints werden durch eine violette Hinterlegung der Zeilennummer kenntlich gemacht. Durch erneuten Doppelklick oder über ein Popup-Menü, das oberhalb der Zeilennummern aktiviert werden kann, kann ein Breakpoint oder auch alle Breakpoints wieder gelöscht werden. Breakpoints nutzt man häufig dazu, dass man ein Programm bis zu einer bestimmten Stelle normal ablaufen lässt und ab dort dann die Möglichkeit der zeilenweisen Ausführung nutzt.

4.7 Zusammenfassung

Herzlichen Glückwunsch! Wenn Sie bis hierhin gekommen sind, haben Sie Ihr erstes Terminal-Programm erstellt und ausgeführt. Sie sehen, die Bedienung des Terminal-Simulators ist gar nicht so kompliziert.

Der Terminal-Simulator bietet jedoch noch weitere Möglichkeiten. Diese können Sie nun durch einfaches Ausprobieren selbst erkunden oder im nächsten Abschnitt im Detail nachlesen.

5 Bedienung des Terminal-Simulators

Im letzten Abschnitt haben Sie eine kurze Einführung in die Funktionalität des Terminal-Simulators erhalten. In diesem Abschnitt werden die einzelnen Funktionen des Simulators nun im Detail vorgestellt. Dabei wird sich natürlich einiges auch wiederholen.

Wenn Sie den Terminal-Simulator starten, öffnen sich ein Fenster mit dem Titel „Terminal-Simulator“. Abbildung 5.1 skizziert die einzelnen Komponenten des Fensters.

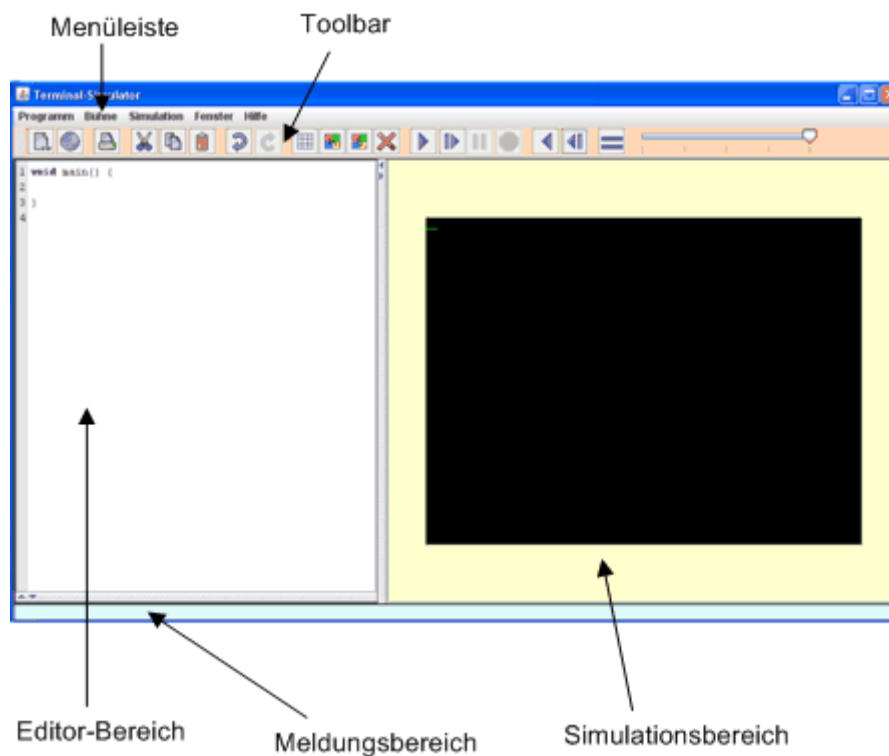


Abb. 5.1: Komponenten des Simulator-Fensters

Die Menüleiste oben im Fenster beinhaltet 5 Menüs. Darunter ist eine Toolbar mit Buttons platziert, über die die wichtigsten Funktionen der Menüs durch Anklicken eines Buttons schneller ausgeführt werden können. Ganz unten im Meldungsbereich werden wichtige Meldungen ausgegeben. Die Eingabe von Terminal-Programmen erfolgt im Editor-Bereich links und die Ausführung von Terminal-Programmen wird im Simulationsbereich (auch „Bühne“ genannt) visualisiert.

Als Hauptfunktionsbereiche des Terminal-Simulators lassen sich identifizieren:

- Verwalten und Editieren von Terminal-Programmen
- Compilieren von Terminal-Programmen
- Verwalten und Gestalten von Terminalern
- Interaktives Ausführen von Terminal-Befehlen

- Ausführen von Terminal-Programmen
- Debuggen von Terminal-Programmen

Bevor im Folgenden anhand dieser Funktionsbereiche der Simulator im Detail vorgestellt wird, werden zuvor noch einige Grundfunktionen graphischer Benutzungsoberflächen erläutert.

5.1 Grundfunktionen

In diesem Unterabschnitt werden einige wichtige Grundfunktionalitäten graphischer Benutzungsoberflächen beschrieben. Der Abschnitt ist für diejenigen von Ihnen gedacht, die bisher kaum Erfahrungen mit Computern haben. Diejenigen von Ihnen, die schon längere Zeit einen Computer haben und ihn regelmäßig benutzen, können diesen Abschnitt ruhig überspringen.

5.1.1 Anklicken

Wenn im Folgenden von „Anklicken eines Objektes“ oder „Anklicken eines Objektes mit der Maus“ gesprochen wird, bedeutet das, dass Sie den Maus-Cursor auf dem Bildschirm durch Verschieben der Maus auf dem Tisch über das Objekt platzieren und dann die – im Allgemeinen linke – Maustaste drücken.

5.1.2 Tooltips

Als *Tooltips* werden kleine Rechtecke bezeichnet, die automatisch auf dem Bildschirm erscheinen, wenn man den Maus-Cursor auf entsprechende Objekte platziert (siehe Abbildung 5.2). In den Tooltips werden bestimmte Informationen ausgegeben.



Abb. 5.2: Tooltipp

5.1.3 Button

Buttons sind Objekte der Benutzungsoberfläche, die man anklicken kann und die daraufhin eine bestimmte Aktion auslösen (siehe Abbildung **Fehler! Verweisquelle konnte nicht gefunden werden..3**). Buttons besitzen eine textuelle Beschreibung (z.B. „OK“) oder eine Graphik, die etwas über die Aktion aussagen. Sie erkennen Buttons an der etwas hervorgehobenen Darstellung. Graphik-Buttons sind in der Regel Tooltips zugeordnet, die die zugeordnete Aktion beschreiben.



Abb. 5.3: Buttons

5.1.4 Menü

Menüs befinden sich ganz oben in einem Fenster in der so genannten *Menüleiste* (siehe Abbildung 5.4). Sie werden durch einen Text beschrieben (Programm, Bühne, Simulation, ...). Klickt man die Texte an, öffnet sich eine Box mit so genannten *Menü-Items*. Diese bestehen wiederum aus Texten, die man anklicken kann. Durch Anklicken von Menü-Items werden genauso wie bei Buttons Aktionen ausgelöst, die im Allgemeinen durch die Texte beschrieben werden (Speichern, Kopieren, ...). Nach dem Anklicken eines Menü-Items wird die Aktion gestartet und die Box schließt sich automatisch wieder. Klickt man irgendwo außerhalb der Box ins Fenster, schließt sich die Box ebenfalls und es wird keine Aktion ausgelöst.

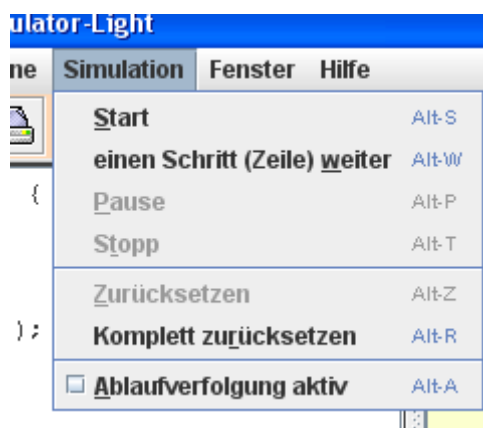


Abb. 5.4: Menüleiste und Menü

Häufig steht hinter den Menü-Items ein weiterer Text, wie z.B. „Strg-O“ oder „Alt-N“. Diese Texte kennzeichnen Tastenkombinationen. Drückt man die entsprechenden Tasten auf der Tastatur, wird dieselbe Aktion ausgelöst, die man auch durch Anklicken des Menü-Items auslösen würde.

Manchmal erscheinen bestimmte Menü-Items etwas heller. Man sagt auch, sie sind ausgegraut. In diesem Fall kann man das Menü-Item nicht anklicken und die zugeordnete Aktion nicht auslösen. Das Programm befindet sich in einem Zustand, in dem die Aktion keinen Sinn machen würde.

5.1.5 Toolbar

Direkt unterhalb der Menüleiste ist die so genannte *Toolbar* angeordnet (siehe Abbildung 5.5). Sie besteht aus einer Menge an Graphik-Buttons, die Alternativen zu den am häufigsten benutzten Menü-Items der Menüs darstellen.

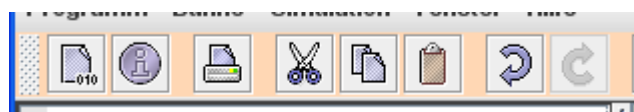


Abb. 5.5: Toolbar

5.1.6 Popup-Menü

Popup-Menüs sind spezielle Menüs, die bestimmten Elementen auf dem Bildschirm zugeordnet sind (siehe Abbildung 5.6). Man öffnet sie dadurch, dass man den Maus-Cursor auf das entsprechende Element verschiebt und danach die rechte Maustaste drückt. Genauso wie bei normalen Menüs erscheint dann eine Box mit Menü-Items.



Abb. 5.6: Popup-Menü

5.1.7 Eingabefeld

Eingabefelder dienen zur Eingabe von Zeichen (siehe Abbildung 5.7). Positionieren Sie dazu den Maus-Cursor auf das Eingabefeld und klicken Sie die Maus. Anschließend können Sie über die Tastatur Zeichen eingeben, die im Eingabefeld erscheinen.

5.1.8 Dialogbox

Beim Auslösen bestimmter Aktionen erscheinen so genannte *Dialogboxen* auf dem Bildschirm (siehe Abbildung 5.7). Sie enthalten in der Regel eine Menge von graphischen Objekten, wie textuelle Informationen, Eingabefelder und Buttons. Wenn eine Dialogbox auf dem Bildschirm erscheint, sind alle anderen Fenster des Programms für Texteingaben oder Mausklicks gesperrt. Zum Schließen einer Dialogbox, d.h. um die Dialogbox wieder vom Bildschirm verschwinden zu lassen, dienen in der Regel eine Menge an Buttons, die unten in der Dialogbox angeordnet sind. Durch Anklicken eines „OK-Buttons“ wird dabei die der Dialogbox zugeordnete Aktion ausgelöst. Durch Anklicken des „Abbrechen-Buttons“ wird eine Dialogbox geschlossen, ohne dass irgendwelche Aktionen ausgelöst werden.

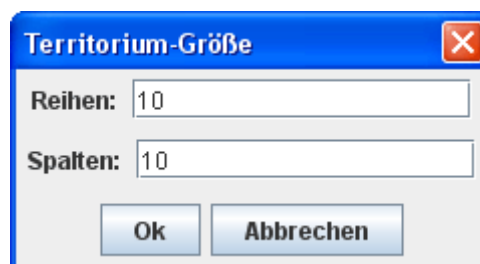


Abb. 5.7: Dialogbox mit Eingabefeldern

5.1.9 Dateiauswahl-Dialogbox

Dateiauswahl-Dialogboxen sind spezielle Dialogboxen, die zum Speichern und Öffnen von Dateien benutzt werden (siehe Abbildung 5.8). Sie spiegeln im Prinzip das Dateisystem wider und enthalten Funktionalitäten zum Verwalten von Dateien und Ordnern.

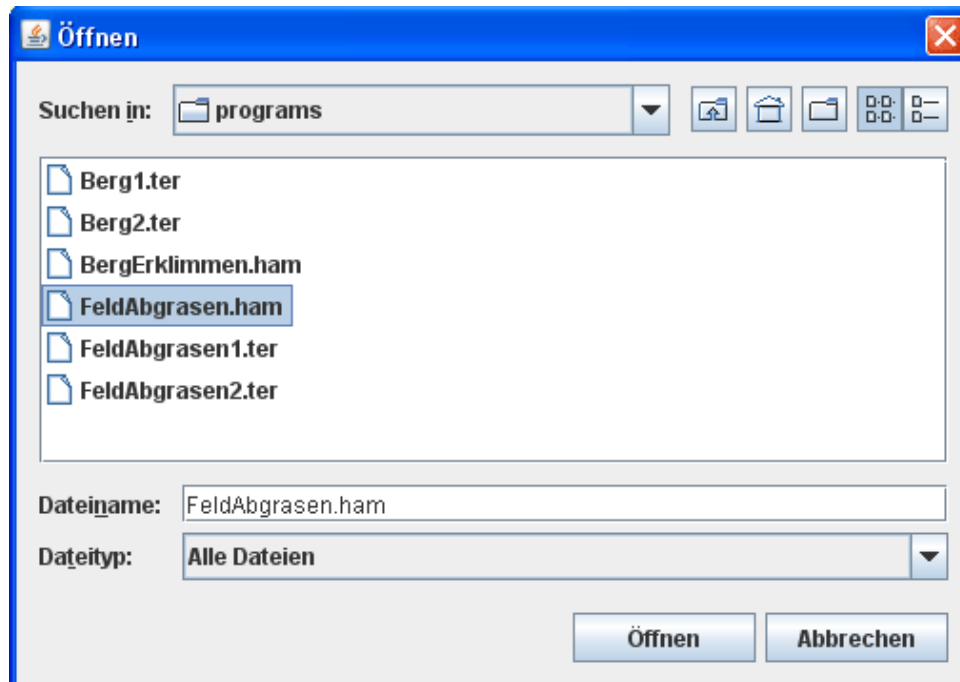


Abb. 5.8: Dateiauswahl-Dialogbox

Im mittleren Bereich einer Dateiauswahl-Dialogbox erscheinen alle Dateien und Unterordner des aktuellen Ordners. Sie sind durch unterschiedliche Symbole repräsentiert. Der eigentliche Zweck von Dateiauswahl-Dialogboxen ist – wie der Name schon sagt – die Auswahl einer Datei. Klickt man auf eine Datei, erscheint der Name automatisch im Eingabefeld „Dateiname“. Dort kann man auch über die Tastatur einen Dateinamen eingeben. Anschließend wird nach Drücken des OK-Buttons die entsprechende Datei geöffnet bzw. gespeichert.

Dateiauswahl-Dialogboxen stellen jedoch noch zusätzliche Funktionalitäten bereit. Durch Doppelklick auf einen Ordner kann man in den entsprechenden Ordner wechseln. Es werden dann anschließend die Dateien und Unterordner dieses Ordners im mittleren Bereich angezeigt. Um zu einem übergeordneten Ordner zurück zu gelangen, bedient man sich des Menüs „Suchen in“, in dem man den entsprechenden Ordner auswählen kann.

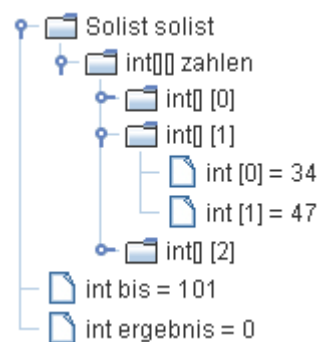
Neben dem „Suchen in“-Menü sind noch fünf Graphik-Buttons angeordnet. Durch Anklicken des linken Buttons kommt man im Ordnerbaum eine Ebene höher. Durch Anklicken des zweiten Buttons von links gelangt man zur Wurzel des Ordnerbaumes. Mit dem mittleren Button kann man im aktuellen Ordner einen neuen Unterordner

anlegen. Mit den beiden rechten Buttons kann man die Darstellung im mittleren Bereich verändern.

Möchte man einen Ordner oder eine Datei umbenennen, muss man im mittleren Bereich der Dateiauswahl-Dialogbox zweimal – mit Pause zwischendurch – auf den Namen des Ordners oder der Datei klicken. Die textuelle Darstellung des Namens wird dann zu einem Eingabefeld, in der man über die Tastatur den Namen verändern kann.

5.1.10 Elementbaum

Ein *Elementbaum* repräsentiert Elemente und strukturelle Beziehungen zwischen ihnen, bspw. die Ordner und Dateien des Dateisystems (siehe Abbildung 5.9).



Abbi. 5.9: Elementbaum mit Verzeichnissen und Dateien

Unterschiedliche Elementtypen werden dabei durch unterschiedliche Symbole dargestellt, hinter denen entsprechende Bezeichnungen erscheinen. Durch Anklicken der Symbole auf der linken Seite kann man Strukturen öffnen und schließen, d.h. Unterstrukturen sichtbar bzw. unsichtbar machen.

Den Ordnern und Dateien sind Popup-Menüs zugeordnet. Um diese zu öffnen, muss man zunächst den Ordner bzw. die Datei mit der Maus anklicken. Der Name wird dann durch einen blauen Balken hinterlegt. Anschließend muss man die rechte Maustaste drücken. Dann öffnet sich das Popup-Menü. Die Popup-Menüs enthalten bspw. Menü-Items zum Löschen und Umbenennen des entsprechenden Ordners bzw. der entsprechenden Datei.

5.1.11 Split-Pane

Eine Split-Pane ist ein Element, das aus zwei Bereichen und einem Balken besteht. (siehe Abbildung 5.10). Die beiden Bereiche können dabei links und rechts oder oberhalb und unterhalb des Balkens liegen. Wenn Sie den Balken mit der Maus anklicken und bei gedrückter Maustaste nach links oder rechts (bzw. oben oder unten) verschieben, vergrößert sich einer der beiden Bereiche und der andere verkleinert sich. Durch Klicken auf einen der beiden Pfeile auf dem Balken können Sie einen der beiden Bereiche auch ganz verschwinden lassen.

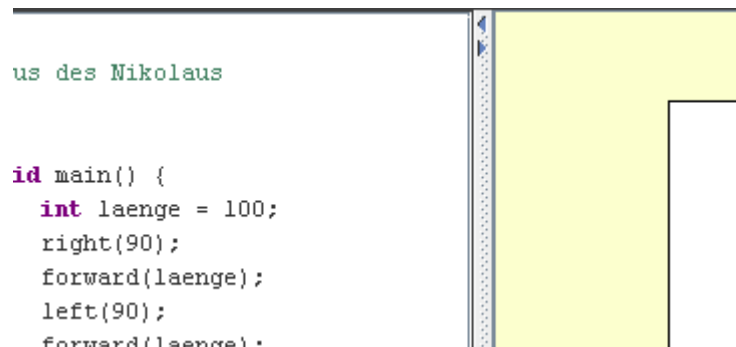


Abb. 5.10: Split-Pane

5.2 Verwalten und Editieren von Terminal-Programmen

Das Schreiben von Programmen bzw. genauer gesagt das Schreiben des Sourcecodes von Programmen bezeichnet man als *Editieren*. Im Terminal-Simulator dient der Editor-Bereich zum Editieren von Terminal-Programmen (siehe Abbildung 5.11)

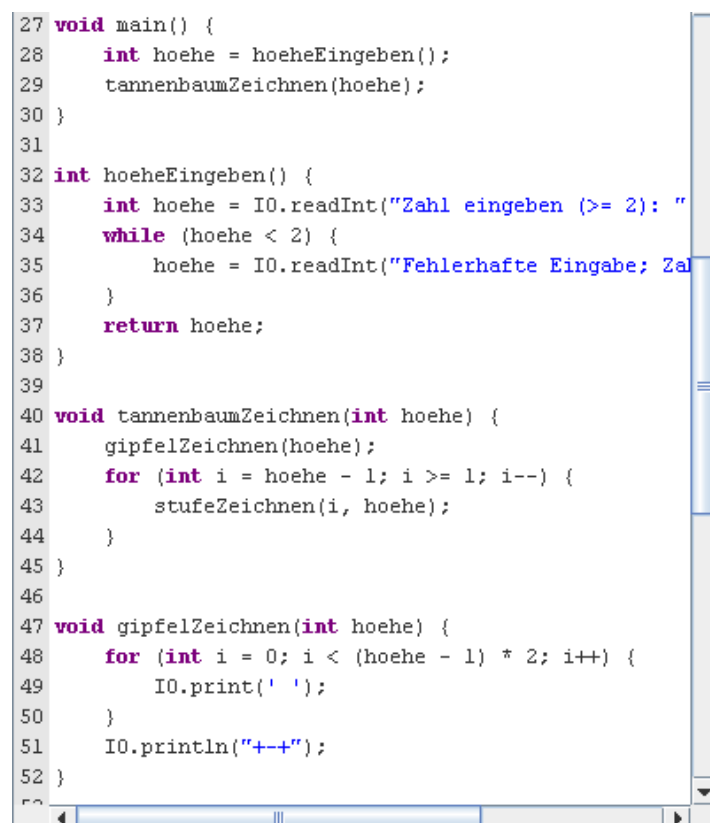


Abb. 5.11: Editor-Bereich des Terminal-Simulators

Im Editor-Bereich können Sie Programme eintippen. Für das Verwalten und Editieren von Programmen ist das Menü „Programm“ wichtig. Unterhalb der Menüleiste ist eine

spezielle Toolbar zu sehen, über die Sie die wichtigsten Funktionen der Menüs auch schneller erreichen und ausführen können. Schieben Sie einfach mal die Maus über die Buttons. Dann erscheint jeweils ein Tooltip, der die Funktionalität des Buttons anzeigt. Die für das Editieren von Programmen wichtigen Buttons der Toolbar werden in Abbildung 5.12 skizziert.

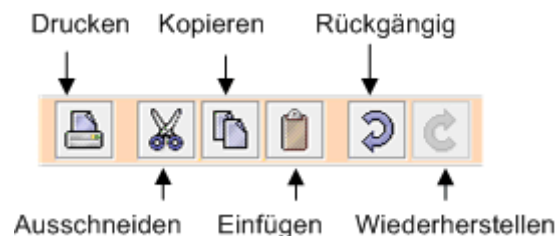


Abb. 5.12: Editor-Buttons der Toolbar

5.2.1 Schreiben eines neuen Terminal-Programms

Das Schreiben eines neuen Terminal-Programms ist durch das entsprechende Eintippen des Sourcecodes im Editor-Bereich möglich.

5.2.2 Ändern des aktuellen Terminal-Programms

Möchten Sie Teile des aktuellen Terminal-Programms ändern, klicken Sie im Editor-Bereich einfach an die entsprechende Stelle und fügen dort die entsprechenden Wörter ein oder löschen sie.

5.2.3 Löschen des aktuellen Terminal-Programms

Komplett löschen können Sie das aktuelle Terminal-Programm des Editor-Bereichs, indem Sie den kompletten Sourcecode mit der Maus selektieren und dann in der Toolbar den „Ausschneiden“-Button (4. Button von links) drücken.

5.2.4 Abspeichern des aktuellen Terminal-Programms

Normalerweise müssen Sie sich nicht um das Speichern des aktuellen Programms kümmern. Es wird automatisch vor dem Compilieren in einer internen Datei abgespeichert. Wenn Sie jedoch ein Programm explizit abspeichern möchten, können Sie im „Programm“-Menü das „Speichern unter...“-Menü-Item anklicken. Es öffnet sich eine Dateiauswahl-Dialogbox, über die Sie die gewünschte Datei auswählen können.

5.2.5 Öffnen eines einmal abgespeicherten Terminal-Programms

Möchten Sie ein einmal abgespeichertes Terminal-Programm wieder in den Editorbereich laden, können Sie dies über das Menü-Item „Laden...“ des

„Programm“-Menüs tun. Nach dem Anklicken des Items erscheint eine Dateiauswahl-Dialogbox, über die Sie die gewünschte Datei auswählen können.

Achtung: Beim Laden einer abgespeicherten Datei geht der aktuelle Inhalt des Editor-Bereichs verloren! Sie müssen ihn also gegebenenfalls vorher in einer Datei abspeichern.

5.2.6 Drucken eines Terminal-Programms

Über den „Drucken“-Button (3. Toolbar-Button von links) können Sie das aktuelle Programm des Editor-Bereichs drucken. Es öffnet sich eine Dialogbox, in der Sie die entsprechenden Druckeinstellungen vornehmen und den Druck starten können.

5.2.7 Editier-Funktionen

Im Editor-Bereich können Sie – wie bei anderen Editoren auch – über die Tastatur Zeichen eingeben bzw. wieder löschen. Darüber hinaus stellt der Editor ein paar weitere Funktionalitäten zur Verfügung, die über das „Programm“-Menü bzw. die entsprechenden Editor-Buttons in der Toolbar aktiviert werden können.

„Ausschneiden“-Button (4. Toolbar-Button von links): Hiermit können Sie komplette Passagen des Editor-Bereichs in einem Schritt löschen. Markieren Sie die zu löschende Passage mit der Maus und klicken Sie dann den Button an. Der markierte Text verschwindet.

„Kopieren“-Button (5. Toolbar-Button von links): Hiermit können Sie komplette Passagen des Editor-Bereichs in einen Zwischenpuffer kopieren. Markieren Sie die zu kopierende Passage mit der Maus und klicken Sie dann den Button an.

„Einfügen“-Button (5. Toolbar-Button von links): Hiermit können Sie den Inhalt des Zwischenpuffers an die aktuelle Cursorposition einfügen. Wählen Sie zunächst die entsprechende Position aus und klicken Sie dann den Button an. Der Text des Zwischenpuffers wird eingefügt.

„Rückgängig“-Button (7. Toolbar-Button von links): Wenn Sie durchgeführte Änderungen des Sourcecode – aus welchem Grund auch immer – wieder rückgängig machen wollen, können Sie dies durch Anklicken des Buttons bewirken.

„Wiederherstellen“-Button (8. Toolbar-Button von links): Rückgängig gemachte Änderungen können Sie mit Hilfe dieses Buttons wieder herstellen.

Die Funktionalitäten „Kopieren“ und „Einfügen“ funktionieren übrigens auch über einzelne Programme hinaus. Es ist sogar möglich, mit Hilfe der Betriebssystem-Kopieren-Funktion Text aus anderen Programmen (bspw. Microsoft Word) zu kopieren und hier einzufügen.

Die gerade aufgelisteten Funktionen finden Sie auch im „Programm“-Menü. Als zusätzliche Funktionalitäten werden dort angeboten:

Einrückung: Durch Anklicken des Menü-Items wird der Einrück-Modus aktiviert bzw. deaktiviert. Ist der Einrück-Modus aktiviert, werden beim Eingeben von Text im Editor-Bereich automatisch Einrückungen an die entsprechende Spalte vorgenommen, wenn Sie die Enter-Taste drücken.

Schriftgröße: Hierüber können Sie die Schriftgröße des Editor-Bereichs anpassen.

5.3 Compilieren von Terminal-Programmen

Beim Compilieren werden Programme – genauer gesagt der Sourcecode – auf ihre (syntaktische) Korrektheit überprüft und im Erfolgsfall ausführbare Programme erzeugt. Zum Compilieren von Programmen dient im Terminal-Simulator der „Compilieren“-Button (erster Button der Toolbar von links) bzw. das Menü-Item „Compilieren“ im „Programm“-Menü (siehe Abbildung 5.13).

An der Farbe des Compiler-Buttons können Sie erkennen, ob Compilieren aktuell notwendig ist oder nicht. Immer wenn Sie Änderungen im Editor-Bereich vorgenommen haben, erscheint der Button rot, und die Änderungen werden erst dann berücksichtigt, wenn Sie (erneut) kompiliert haben. Erscheint der Button in einer neutralen Farbe, ist kein Compilieren notwendig.

5.3.1 Compilieren

Wenn Sie den „Compilieren“-Button anklicken, wird das Programm, das gerade im Editor-Bereich sichtbar ist, in einer internen Datei (mit dem Namen „Solist.java“) abgespeichert und kompiliert.

Wenn Ihr Programm syntaktisch korrekt ist, erscheint nach ein paar Sekunden eine Dialogbox mit einer entsprechenden Meldung. Es wurde ein (neues) ausführbares Programm erzeugt.

5.3.2 Beseitigen von Fehlern

Wenn Ihr Programm Fehler enthält, öffnet sich unterhalb des Editor-Bereichs in einer Scroll-Pane ein neuer Bereich, der die Fehlermeldungen des Compilers anzeigt (siehe Abbildung 5.13). Es wurde kein (neues) ausführbares Programm erzeugt! Jede Fehlermeldung erscheint in einer eigenen Zeile. Jede Zeile enthält eine Beschreibung des (wahrscheinlichen) Fehlers sowie die entsprechende Zeile der Anweisung im Programm. Wenn Sie eine Fehlermeldung anklicken, wird die entsprechende Anweisung im Eingabebereich blau markiert und der Maus-Cursor an die entsprechende Stelle gesetzt. Sie müssen nun die einzelnen Fehler beseitigen und dann erneut speichern und compilieren, bis Ihr Programm keine Fehler mehr enthält. Der Fehlermeldungs-bereich schließt sich dann automatisch wieder.

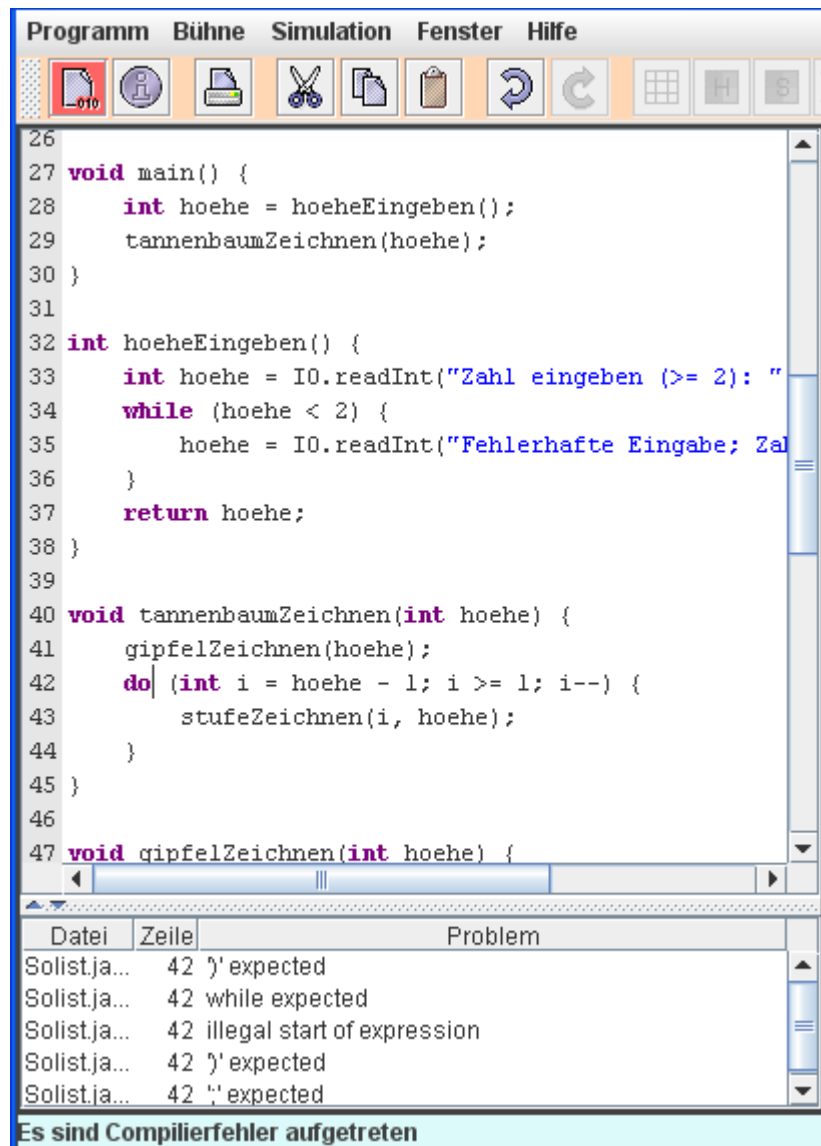


Abb. 5.13: Fehlermeldungen des Compilers

Achtung: Die Interpretation von Fehlermeldungen, die der Compiler ausgibt, ist nicht trivial. Die Meldungen sind nicht immer besonders präzise und oft auch irreführend. Häufig gibt der Compiler mehrere Fehlermeldungen aus, obwohl es sich nur um einen einzelnen Fehler handelt. Deshalb beherzigen Sie gerade am Anfang folgende Hinweise: Arbeiten Sie die Fehlermeldungen immer von oben nach unten ab. Wenn der Compiler eine große Menge von Fehlermeldungen liefert, korrigieren Sie zunächst nur eine Teilmenge und kompilieren Sie danach erneut. Bauen Sie – gerade als Programmieranfänger – auch mal absichtlich Fehler in Ihre Programme ein und schauen Sie sich dann die Fehlermeldungen des Compilers an.

5.4 Verwalten und Gestalten von Terminalern

Standardmäßig ist das im Simulationsbereich angezeigte Terminal 30 Reihen und 40 Spalten groß und leer. Der Cursor befindet sich an der Position (0 / 0), d.h. oben

links. Die Größe der einzelnen Zellen beträgt 12 * 12 Pixel. Die Zellengröße bestimmt die Schriftgröße.

In der Toolbar dienen die Buttons 9 – 12 von links zum Gestalten von Terminals. Über das Menü „Bühne“ können Terminals verwaltet werden (siehe auch Abbildung 5.14).

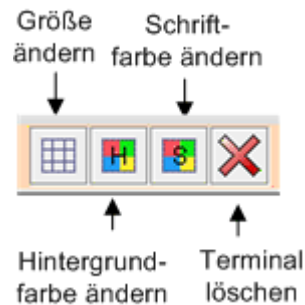


Abb. 5.14: Gestaltungsbuttons der Toolbar

Während ein Programm läuft oder pausiert, ist keine Umgestaltung des Territoriums möglich.

5.4.1 Größe des Terminals ändern

Mit Hilfe des „Größe ändern“-Buttons (9. Toolbar-Button von links) können Sie die Größe des Terminals ändern. Nach dem Anklicken des Buttons öffnet sich eine Dialogbox, in der Sie die gewünschte Größe (Anzahl an Reihen, Anzahl an Spalten, Zellengröße) angeben können. Nach dem Drücken des OK-Buttons der Dialogbox wird im Simulationsbereich das aktuelle Terminal gelöscht und ein Terminal der entsprechenden Größe dargestellt.

Der „Hintergrundfarbe ändern“-Button (10. Toolbar-Button von links) dient zum Ändern der Hintergrundfarbe. Wenn Sie den Button anklicken, erscheint eine spezielle Dialogbox, in der Sie die gewünschte Farbe auswählen können. Auf analoge Art und Weise können Sie mit dem „Schriftfarbe ändern“-Button (11. Toolbar-Button von links) die Schriftfarbe anpassen.

Möchten Sie das Terminal löschen, so können Sie dies durch Anklicken des „Löschen“-Buttons (12. Toolbar-Button von links) tun.

Mit Hilfe des „Größe ändern“-Buttons (9. Toolbar-Button von links) können Sie die Größe des Terminals ändern. Nach dem Anklicken des Buttons öffnet sich eine Dialogbox, in der Sie die gewünschte Größe angeben können. Erlaubt sind Werte zwischen 1 und 8. Nach dem Drücken des OK-Buttons der Dialogbox wird im Simulationsbereich ein Terminal der entsprechenden Größe dargestellt. Sollten bei einer Verkleinerung des Terminals bereits platzierte Damen außerhalb des Terminals liegen, so werden diese automatisch gelöscht.

5.4.2 Hintergrundfarbe ändern

Der „Hintergrundfarbe ändern“-Button (10. Toolbar-Button von links) dient zum Ändern der Hintergrundfarbe. Wenn Sie den Button anklicken, erscheint eine spezielle Dialogbox, in der Sie die gewünschte Farbe auswählen können. Nach der Farbwahl und dem Drücken des OK-Buttons erscheint das Terminal in der neuen Hintergrundfarbe.

5.4.3 Schriftfarbe ändern

Der „Schriftfarbe ändern“-Button (11. Toolbar-Button von links) dient zum Ändern der Schriftfarbe. Wenn Sie den Button anklicken, erscheint eine spezielle Dialogbox, in der Sie die gewünschte Farbe auswählen können. Nach der Farbwahl und dem Drücken des OK-Buttons erscheint das Terminal mit der neuen Schriftfarbe.

5.4.4 Löschen des Terminals

Möchten Sie das Terminal löschen, so können Sie dies durch Anklicken des „Löschen“-Buttons (12. Toolbar-Button von links) tun. Der Cursor wird in der linken oberen Ecke (Position (0 / 0)) platziert.

5.4.5 Abspeichern des Terminales

Sie können das aktuelle Terminal in einer Datei abspeichern und später wieder laden. Zum Abspeichern des aktuellen Terminales aktivieren Sie im Menü „Bühne“ das Menü-Item „Speichern unter...“. Es öffnet sich eine Dateiauswahl-Dialogbox. Hierin können Sie den Ordner auswählen und den Namen einer Datei eingeben, in die das aktuelle Terminal gespeichert werden soll.

Weiterhin ist es möglich, das aktuelle Terminal als Bild (gif- oder png-Datei) abzuspeichern. Eine entsprechende Funktion findet sich im „Bühne“-Menü.

5.4.6 Wiederherstellen eines abgespeicherten Terminals

Abgespeicherte Terminaler können mit dem „Laden“-Menü-Item des „Bühne“-Menüs wieder geladen werden. Es erscheint eine Dateiauswahl-Dialogbox, in der Sie die zu ladende Datei auswählen können. Nach dem Anklicken des OK-Buttons schließt sich die Dialogbox und das entsprechende Terminal wird wiederhergestellt.

Achtung: Der Zustand des Terminals, der vor dem Ausführen der Laden-Funktion Gültigkeit hatte, geht dabei verloren. Speichern Sie das Terminal daher gegebenenfalls vorher ab.

5.5 *Interaktives Ausführen von Terminal-Befehlen*

Sie können einzelne Terminal-Befehle oder selbst definierte Funktionen und Prozeduren bspw. zu Testzwecken auch interaktiv ausführen. Aktivieren Sie dazu im

Menü „Fenster“ den Eintrag „Befehlsfenster“. Es öffnet sich das so genannte Befehlsfenster (siehe Abbildung 5.15).

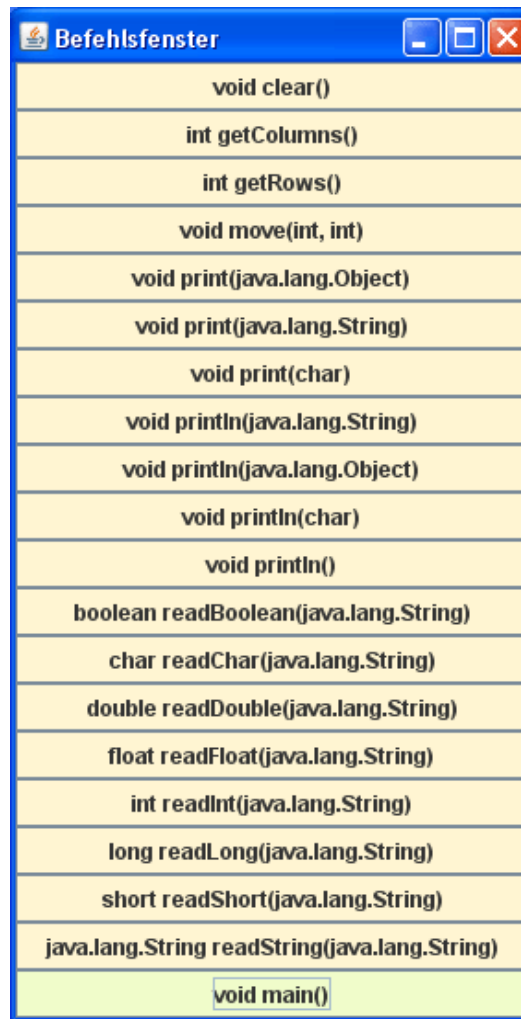


Abb. 5.15: Befehlsfenster

5.5.1 Befehlsfenster

Im Befehlsfenster werden die Terminal-Befehle (mit grünlichem Hintergrund) sowie die Prozeduren und Funktionen dargestellt, die im aktuellen Terminal-Programm beim letztmaligen erfolgreichen Compilieren definiert waren (mit orangenem Hintergrund).

Beim Anklicken mit der Maus werden die entsprechenden Befehle jeweils ausgeführt. Die Ausführung erfolgt dabei ohne Anzeige von Zwischenzuständen. D.h. wird bspw. die main-Prozedur angeklickt, wird das entsprechende Programm ohne Visualisierung der einzelnen Befehle „in einem Rutsch“, also sehr schnell ausgeführt.

Dauert die Ausführung eines Befehls mehr als 5 Sekunden (vermutlich enthält dann die Funktion eine Endlosschleife), wird der Befehl abgebrochen und der Terminal-Simulator wird komplett auf seinen Startzustand zurückgesetzt.

5.5.2 Parameter

Besitzt eine Prozedur oder Funktion Parameter, erscheint nach ihrer Aktivierung im Befehlsfenster eine Dialogbox, in der die entsprechenden aktuellen Parameterwerte eingegeben werden müssen. Hinweis: Aktuell wird nur die Eingabe von Werten der Java-Standard-Datentypen (`int`, `boolean`, `float`, ...) sowie die Eingabe von Zeichenketten (Strings) unterstützt.

5.5.3 Rückgabewerte von Funktionen

Bei der Ausführung von Funktionen wird der jeweils gelieferte Wert in einem Dialogfenster dargestellt.

5.6 Ausführen von Terminal-Programmen

Ausgeführt werden können (erfolgreich compilierte) Terminal-Programme mit Hilfe der in Abbildung 5.16 skizzierten Buttons der Toolbar. Alle Funktionen sind darüber hinaus auch über das Menü „Simulation“ aufrufbar.

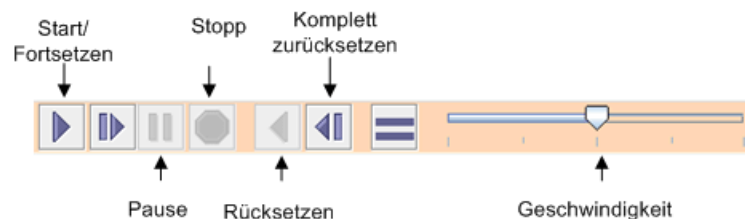


Abb. 5.16: Simulationsbuttons der Toolbar

5.6.1 Starten eines Terminal-Programms

Bevor ein Terminal-Programm ausgeführt werden kann, muss es erfolgreich compiliert worden sein. Gestartet werden kann das aktuelle Terminal-Programm dann durch Anklicken des „Start/Fortsetzen“-Buttons (13. Toolbar-Button von links).

Nach dem Starten eines Terminal-Programms wird der virtuelle Schreiber auf der Zeichenfläche aktiv und tut das, was das Programm ihr vorgibt. Während des Ausführens eines Terminal-Programms wird der Editor-Bereich ausgegraut, d.h. es können während der Ausführung eines Programms keine Änderungen am Sourcecode durchgeführt werden. Während ein Programm läuft oder pausiert ist, ist keine interaktive Umgestaltung des Terminals möglich.

5.6.2 Stoppen eines Terminal-Programms

Die Ausführung eines Terminal-Programms kann durch Anklicken des „Stopp“-Buttons (16. Button der Toolbar von links) jederzeit abgebrochen werden.

5.6.3 Pausieren eines Terminal-Programms

Möchten Sie ein in Ausführung befindliches Programm (kurzfristig) anhalten, können Sie dies durch Anklicken des „Pause“-Buttons (15. Button der Toolbar von links) tun. Wenn Sie anschließend auf den „Start/Fortsetzen“-Button klicken, wird die Programmausführung fortgesetzt.

5.6.4 Während der Ausführung eines Terminal-Programms

Treten bei der Ausführung eines Programms Laufzeitfehler auf, wird eine Dialogbox geöffnet, die eine entsprechende Fehlermeldung enthält. Nach dem Anklicken des OK-Buttons in der Dialogbox wird das Terminal-Programm beendet. Weiterhin öffnet sich das Konsolen-Fenster, in dem ebenfalls die Fehlermeldung ausgegeben wird.

5.6.5 Einstellen der Geschwindigkeit

Mit dem Schieberegler ganz rechts in der Toolbar können Sie die Geschwindigkeit der Programmausführung beeinflussen. Je weiter links der Regler steht, desto langsamer wird das Programm ausgeführt. Je weiter Sie den Regler nach rechts verschieben, umso schneller agiert der virtuelle Schreiber.

5.6.6 Wiederherstellen einer Zeichenfläche

Beim Testen eines Programms recht hilfreich ist der „Rücksetzen“-Button (17. Button der Toolbar von links). Sein Anklicken bewirkt, dass das Terminal in den Zustand zurückversetzt wird, den es vor dem letzten Start eines Programms inne hatte.

Über den „Komplett Zurücksetzen“-Button (18. Button der Toolbar von links) ist eine Rücksetzung des Terminals in den Zustand möglich, der beim Öffnen des Terminal-Simulators gültig war. Sollte es irgendwann einmal bei der Benutzung des Terminal-Simulators zu unerklärlichen Fehlern kommen, ist es mit Hilfe dieses Buttons möglich, den Terminal-Simulator zu reinitialisieren.

5.7 Debuggen von Terminal-Programmen

Debugger sind Hilfsmittel zum Testen von Programmen. Sie erlauben es, während der Programmausführung den Zustand des Programms zu beobachten und gegebenenfalls sogar interaktiv zu ändern. Damit sind Debugger sehr hilfreich, wenn es um das Entdecken von Laufzeitfehlern und logischen Programmfehlern geht.

Der Debugger des Terminal-Simulators ermöglicht während der Ausführung eines Terminal-Programms das Beobachten des Programmzustands. Sie können sich während der Ausführung eines Terminal-Programms anzeigen lassen, welche

Anweisung des Sourcecodes gerade ausgeführt wird und welche Werte die Variablen aktuell speichern. Die interaktive Änderung von Variablenwerten wird aktuell nicht unterstützt.

Die Funktionen des Debuggers sind eng mit den Funktionen zur Programmausführung verknüpft. Sie finden die Funktionen im Menü „Simulation“. Es bietet sich jedoch an, die entsprechenden Buttons der Toolbar zu verwenden. Neben dem „Start/Fortsetzen“- , dem „Pause“- und dem „Stopp“-Button gehören die zwei Buttons „Schrittweise Ausführung“ und „Ablaufverfolgung“ zu den Debugger-Funktionen (siehe auch Abbildung 5.17).

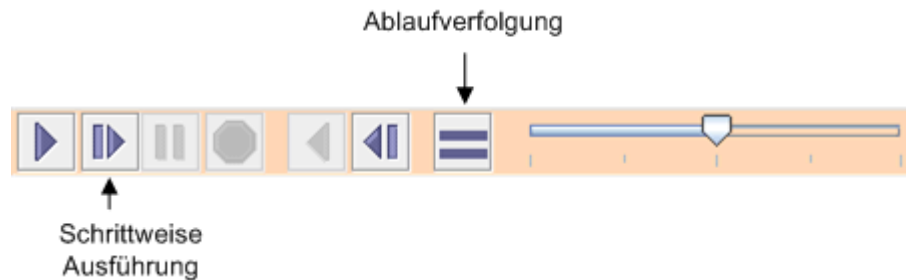


Abb. 5.17: Debugging-Buttons der Toolbar

5.7.1 Beobachten der Programmausführung

Durch Anklicken des Buttons „Ablaufverfolgung“ (19. Button der Toolbar von links) aktivieren bzw. (bei erneuten Anklicken) deaktivieren Sie die Ablaufverfolgung des Debuggers. Bei der Aktivierung öffnet sich dazu das Debugger-Fenster (siehe Abbildung 5.18). Dieses können Sie auch über das Menü „Fenster“ sichtbar bzw. unsichtbar machen.

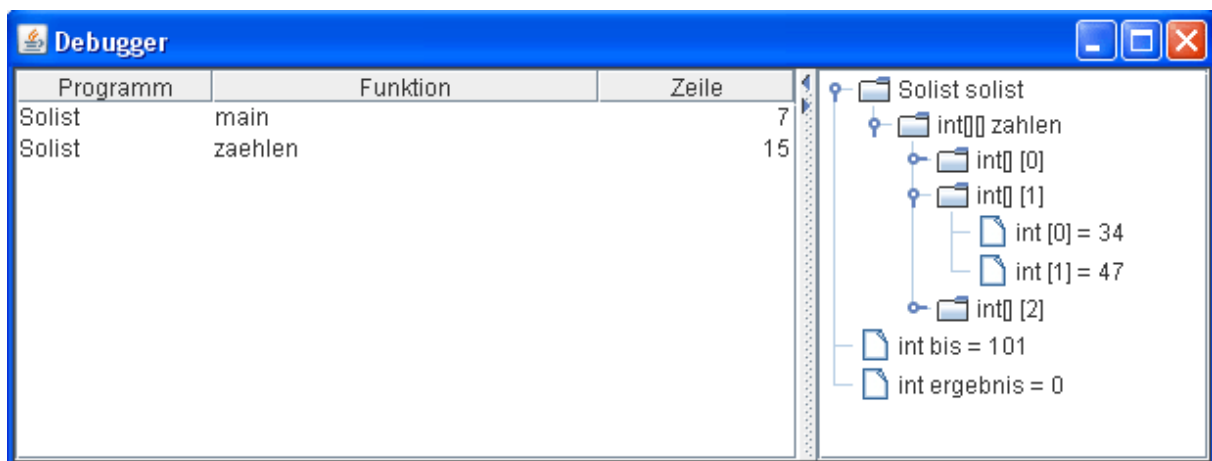


Abb. 5.18: Debugger-Fenster

Ist die Ablaufverfolgung aktiviert, wird bei der Ausführung des Programms im Editor-Bereich der Befehl (bzw. die entsprechende Zeile), der als nächstes ausgeführt wird, blau markiert. Bei einem Prozedur- bzw. Funktionsaufruf wird in die entsprechende

Funktion gesprungen. Weiterhin werden im Debugger-Fenster der aktuelle Stack der Funktionsaufrufe (Name der Funktion und aktuelle Position der Ausführung der Funktion) sowie die aktuelle Belegung der Variablen dargestellt.

Im linken Bereich des Debugger-Fensters werden Informationen zu den aktiven Funktionen angezeigt, und zwar jeweils der Name der Funktion und die aktuelle Position der Ausführung der Funktion. Ganz unten erscheint die aktuell aktive Funktion, darüber gegebenenfalls die Funktion, die diese Funktion aufgerufen hat, usw. Ganz oben steht also immer die `main`-Funktion.

Im rechten Bereich des Debugger-Fensters werden die aktiven Variablen und ihre aktuellen Werte angezeigt. Die Darstellung erfolgt dabei in einem Elementbaum, d.h. bei komplexen Variablen, wie Arrays, können Sie durch Anklicken des Symbols vor dem Variablennamen die Komponenten einsehen.

Auch während die Ablaufverfolgung aktiviert ist, können Sie die Programmausführung durch Anklicken des „Pause“-Buttons anhalten und durch anschließendes Anklicken des „Start/Fortsetzen“-Buttons wieder fortfahren lassen. Auch die Geschwindigkeit der Programmausführung lässt sich mit dem Schieberegler anpassen.

5.7.2 Schrittweise Programmausführung

Mit dem Toolbar-Button „Schrittweise Ausführung“ (14. Button der Toolbar von links) ist es möglich, ein Programm schrittweise, d.h. Anweisung für Anweisung auszuführen. Immer, wenn Sie den Button anklicken, wird die nächste Anweisung (bzw. Zeile) ausgeführt.

Sie können das Programm mit der schrittweisen Ausführung starten. Sie können jedoch auch zunächst das Programm normal starten, dann pausieren und ab der aktuellen Anweisung schrittweise ausführen. Eine normale Programmweiterführung ist jederzeit durch Klicken des „Start“-Buttons möglich.

5.7.3 Breakpoints

Wenn Sie das Programm an einer bestimmten Stelle anhalten möchten, können Sie in der entsprechenden Zeile einen so genannten Breakpoint setzen. Führen Sie dazu vor oder während der Programmausführung im Editor-Bereich mit der Maus einen Doppelklick auf die entsprechende Zeilennummer aus. Breakpoints werden durch violett hinterlegte Zeilennummern dargestellt (siehe Abbildung 5.19). Ein Doppelklick auf einen Breakpoint löscht den Breakpoint wieder. Über der Spalte mit den Zeilennummern lässt sich auch durch Klicken der rechten Maustaste ein Popup-Menü aktivieren, das als Funktionen das Setzen bzw. Entfernen von Breakpoints bzw. das gleichzeitige Löschen aller Breakpoints bietet.


```

31
32 int hoeheEingeben() {
33     int hoehe = IO.readInt("Zahl eingeben (>= 2): ");
34     while (hoehe < 2) {
35         hoehe = IO.readInt("Fehlerhafte Eingabe; Zahl eingeben (>= 2): ");
36     }
37     return hoehe;
38 }
39
40 void tannenbaumZeichnen(int hoehe) {
41     gipfelZeichnen(hoehe);
42     for (int i = hoehe - 1; i >= 1; i--) {
43         stufeZeichnen(i, hoehe);
44     }
45 }
46
47 void gipfelZeichnen(int hoehe) {
48     for (int i = 0; i < (hoehe - 1) * 2; i++) {

```

Abb. 5.19: Breakpoints

Wenn Sie ein Programm mit Breakpoints ausführen, wird die Ausführung jedesmal pausiert, wenn eine Zeile mit einem Breakpoint erreicht wird (unabhängig davon, ob die Ablaufverfolgung eingeschaltet ist). Durch Drücken des „Start/Fortsetzen“-Buttons kann die Ausführung des Programms fortgesetzt werden.

5.7.4 Debugger-Fenster

Das Debugger-Fenster wird automatisch bei Aktivierung der Ablaufverfolgung sichtbar gemacht. Über das Menü „Fenster“ lässt es sich jedoch auch explizit sichtbar bzw. unsichtbar machen. Das Fenster besteht aus einem rechten und einem linken Teil innerhalb einer Split-Pane. Der Inhalt des Debugger-Fensters wird nur dann aktualisiert, wenn die Ablaufverfolgung aktiviert ist.

Im linken Bereich des Fensters werden die aktuell aufgerufenen Funktionen und die jeweilige Zeilennummer dargestellt, in der sich die Programmausführung innerhalb der Funktion gerade befindet. Ganz oben steht dabei immer die main-Funktion, ganz unten die zuletzt aufgerufene Funktion.

Im rechten Bereich werden Variablen und deren aktuelle Werte dargestellt. Im Normalfall sind das genau die Variablen, die in der zuletzt aufgerufenen Funktion (also der im linken Bereich ganz unten stehenden Funktion) gültig sind. Die Darstellung erfolgt dabei in einem Elementbaum. Bei strukturierten Variablen kann durch Mausklick auf das Symbol vor dem Variablennamen die Struktur weiter geöffnet (bzw. wieder geschlossen) werden.

Im pausierten Zustand ist es möglich, sich auch die Werte lokaler Variablen anderer Funktionsinkarnationen anzuschauen. Das ist möglich, indem man im linken Bereich auf den entsprechenden Funktionsnamen klickt.

5.8 Konsole

Die Konsole ist ein zusätzliches Fenster, das bei bestimmten Aktionen automatisch geöffnet wird, sich aber über das Menü „Fenster“ auch explizit öffnen bzw. schließen lässt (siehe Abbildung 5.20).

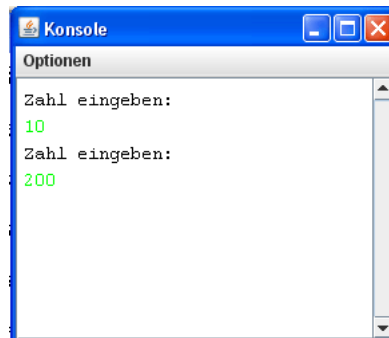


Abb. 5.20: Konsole

Die Konsole ist für die Java-Eingabe mittels `java.lang.System.in` und die Ausgabe mittels `java.lang.System.out` und `java.lang.System.err` zuständig. Wird in Ihrem Programm bspw. der Befehl `java.lang.System.out.println(„hallo“);` ausgeführt, wird die Zeichenkette `hallo` in der Konsole ausgegeben. `java.lang.System.out` und `java.lang.System.err` unterscheiden sich durch eine schwarze (out) bzw. eine rote (err) Ausgabe des entsprechenden Textes. Auch Fehlermeldungen des Terminal-Simulators erscheinen in der Konsole.

Ein Eingabebefehl via `java.lang.System.in` blockiert das Terminal-Programm so lange, bis der Nutzer eine entsprechende Eingabe in der Konsole getätigt und im Allgemeinen durch Drücken der Enter-Taste abgeschlossen hat.

Die Konsole enthält in der Menüleiste ein Menü namens „Optionen“. Hierin finden sich drei Menü-Items, über die es möglich ist, den aktuellen Inhalt der Konsole zu löschen, den aktuellen Inhalt der Konsole in einer Datei abzuspeichern bzw. die Konsole zu schließen.

Beachten Sie, dass Sie die verlängerte Schreibweise `java.lang.System` verwenden müssen, wenn Sie die Konsole für Ein- bzw. Ausgaben nutzen wollen. Die Kurzform `System` wird auf das virtuelle Terminal umgelenkt (siehe auch Abschnitt 6.2).

6 Beispielprogramme

Dem Terminal-Simulator sind fünf Beispielprogramme beigelegt, die Sie über das Menü „Programm“ laden können (Menü-Item „Laden...“).

6.1 Treppe

Dateiname: treppe.prog

```
/*
In dieser Aufgabe geht es darum, Treppen der
folgenden Form auf den Bildschirm zu zeichnen:
```

```
+--+
| |
+-----+
|      |
+-----+
|      |
+-----+
|      |
```

Schreiben Sie ein Java-Programm, das zunächst vom Benutzer einen positiven int-Wert erfragt. Dieser gibt die Höhe der Treppe an (in dem Beispiel 4). Das Programm soll daraufhin eine Treppe in der oben skizzierten Form auf den Bildschirm ausgeben.

```
*/
void main() {

    // Eingabe
    int hoehe = readInt("Hoehe (>0): ");
    while (hoehe <= 0) {
        hoehe = readInt("Fehler (Hoehe <= 0); Hoehe (>0): ");
    }

    // Ausgabe
    for (int r = 1; r <= hoehe; r++) {

        // erste Zeile
        print('+');
        for (int i = 1; i <= (r - 1) * 4 + 1; i++) {
            print('-');
        }
        println('+');

        // zweite Zeile
        print('|');
        for (int i = 1; i <= (r - 1) * 4 + 1; i++) {
            print(' ');
        }
        println('|');
    }
}
```

6.2 Schneeflocke

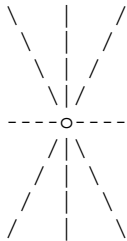
Den Terminal-Befehlen können Sie übrigens auch den Präfix `System.out` voranstellen.

Dateiname: `schneeflocke.prog`

```
/*
Bald ist Winter, und hoffentlich wird bald der
erste Schnee fallen. Sie sollen diesem Tag
vorgreifen und schon jetzt Schneeflocken auf
den Bildschirm zaubern.
```

Aufgabe: Schreiben Sie ein Java-Programm, das den Benutzer zunächst auffordert, eine Zahl größer gleich 0 einzugeben und das anschließend eine entsprechend große Schneeflocke der folgenden Form auf den Bildschirm ausgibt:

Größe = 4:



```
*/

void main() {
    int groesse = readInt("Groesse (>= 0):");
    while (groesse < 0) {
        System.out.println("Falsche Eingabe!");
        groesse = readInt("Groesse (>= 0):");
    }

    zeichneObereFlocke(groesse);
    zeichneMitte(groesse);
    zeichneUntereFlocke(groesse);
}

void zeichneObereFlocke(int groesse) {
    for (int r=0; r<groesse; r++) {
        zeichneZeichen(' ', r);
        System.out.print('\\');
        zeichneZeichen(' ', groesse-r-1);
        System.out.print('|');
        zeichneZeichen(' ', groesse-r-1);
        System.out.println('/');
    }
}

void zeichneMitte(int groesse) {
    zeichneZeichen('-', groesse);
    System.out.print('o');
    zeichneZeichen('-', groesse);
    System.out.println();
}
```

```

void zeichneUntereFlocke(int groesse) {
    for (int r=0; r<groesse; r++) {
        zeichneZeichen(' ', groesse-r-1);
        System.out.print('/');
        zeichneZeichen(' ', r);
        System.out.print('|');
        zeichneZeichen(' ', r);
        System.out.println('\n');
    }
}

void zeichneZeichen(char zeichen, int anzahl) {
    for (int a = 0; a < anzahl; a++) {
        System.out.print(zeichen);
    }
}

```

6.3 Tannenbaum

Den Terminal-Befehlen können Sie übrigens auch den Präfix `IO` voranstellen.

Dateiname: `tannenbaum.prog`

```

/*
Bald ist Weihnachten und Sie möchten sich selbst mal eine
Freude machen, indem Sie einen ASCII-Weihnachtsbaum auf
den Bildschirm Ihres Computers malen.

Aufgabe: Schreiben Sie ein Java-Programm, das den
Benutzer zunächst auffordert, eine Zahl größer gleich 2
einzugeben und das anschließend einen entsprechend hohen
Weihnachtsbaum der folgenden Form auf den Bildschirm
ausgibt:

Höhe = 2:
  +-+
  | |
+-+ +-+

Höhe = 4:
      +-+
      | |
    +-+ +-+
    |   |
  +-+   +-+
  |     |
+-+     +-+
*/

void main() {
    int hoehe = hoeheEingeben();
    tannenbaumZeichnen(hoehe);
}

int hoeheEingeben() {
    int hoehe = IO.readInt("Zahl eingeben (>= 2): ");
    while (hoehe < 2) {
        hoehe = IO.readInt("Fehlerhafte Eingabe; Zahl eingeben (>= 2): ");
    }
    return hoehe;
}

void tannenbaumZeichnen(int hoehe) {
    gipfelZeichnen(hoehe);
}

```

```

        for (int i = hoehe - 1; i >= 1; i--) {
            stufeZeichnen(i, hoehe);
        }
    }

void gipfelZeichnen(int hoehe) {
    for (int i = 0; i < (hoehe - 1) * 2; i++) {
        IO.print(' ');
    }
    IO.println("+--+");
}

void stufeZeichnen(int aktHoehe, int hoehe) {
    pipeZeileSchreiben(aktHoehe, hoehe);
    plusPipePlusZeileSchreiben(aktHoehe, hoehe);
}

void pipeZeileSchreiben(int aktHoehe, int hoehe) {
    for (int i = 0; i < aktHoehe * 2; i++) {
        IO.print(' ');
    }
    IO.print('|');
    for (int i = 0; i <= (hoehe - aktHoehe - 1) * 4; i++) {
        IO.print(' ');
    }
    IO.println('|');
}

void plusPipePlusZeileSchreiben(int aktHoehe, int hoehe) {
    for (int i = 0; i < aktHoehe * 2 - 2; i++) {
        IO.print(' ');
    }
    IO.print("+--+");
    for (int i = 0; i <= (hoehe - aktHoehe - 1) * 4; i++) {
        IO.print(' ');
    }
    IO.println("+--+");
}

```

6.4 Dreieck

Dateiname: dreieck.prog

```

/*
Schreiben Sie ein Java-Programm, das nach Eingabe
einer ungeraden Natürlichen Zahl basislaenge
ein Sternendreieck auf den Bildschirm ausgibt.
Die unterste Reihe soll dabei aus basislaenge
Sternchen bestehen.
Beispielausgabe für basislaenge == 9:

    *
   ***
  *****
 *****
*****
*/
void main() {

    // Eingabe
    int basislaenge = IO.readInt("Basislaenge (>0 und ungerade): ");
    while (basislaenge <= 0 || basislaenge % 2 == 0) {
        basislaenge = IO.readInt("Fehler; Anzahl (>0 und ungerade): ");
    }
}

```

```

// Ausgabe

// alle Reihen
for (int r = 1; r <= basislaenge / 2 + 1; r++) {

    // Leerzeichen
    for (int i = 1; i <= basislaenge / 2 - r + 1; i++) {
        IO.print(' ');
    }

    // Sterne
    for (int i = 1; i <= r * 2 - 1; i++) {
        IO.print('*');
    }
    IO.println();
}
}

```

6.5 Pfeil

Dateiname: pfeil.prog

```

/*
Schreiben Sie ein Java-Programm, das den Benutzer
zunächst auffordert, eine Zahl größer gleich 2
einzugeben und das, sobald der Benutzer eine
entsprechende Zahl eingegeben hat, einen
entsprechend großen Pfeil der folgenden Form
auf den Bildschirm ausgibt:

Zahl = 2:
\
 \
 /
 /

Zahl = 4:
\
 \
 \
 \
 /
 /
 /
*/
void main() {
    int laenge = IO.readInt("Laenge (>= 2): ");
    while (laenge < 2) {
        laenge = IO.readInt("Fehler; Laenge (>= 2): ");
    }

    // oberer Teil
    for (int i = 1; i <= laenge; i++) {
        // Leerzeichen
        for (int j = 1; j <= i - 1; j++) {
            System.out.print(" ");
        }
        System.out.println("\\");
    }

    // unterer Teil
    for (int i = 1; i <= laenge; i++) {
        // Leerzeichen
        for (int j = 1; j <= laenge - i; j++) {

```

```
        System.out.print(" ");
    }
    System.out.println("/");
}
}
```


7 Literatur zum Erlernen der Programmierung

Der Terminal-Simulator ist ein Werkzeug, das Programmierern beim praktischen Erlernen der (imperativen) Programmierung hilft. Der Befehlssatz ist minimal und es macht einfach Spaß visualisiert zu bekommen, was ein Programm bewirkt. Der Terminal-Simulator ist jedoch kein Lehrbuch. Um mit dem Terminal-Simulator programmieren zu lernen, sollten Sie sich ein begleitendes Lehrbuch anschaffen.

Wie bei der Terminal-Miniprogrammierwelt handelt es sich auch beim so genannten Java-Hamster-Modell um eine Miniprogrammierwelt. Zu dem Hamster-Modell gibt es ein Lehrbuch: **„Programmieren spielend gelernt mit dem Java-Hamster-Modell“** von Dietrich Boles, erschienen im Vieweg+Teubner-Verlag. In diesem Buch werden die grundlegenden Konzepte der Programmierung anhand des Java-Hamster-Modells vorgestellt. Das Buch geht langsam und schrittweise vor. Es enthält viele Beispiele und auch eine Reihe von Aufgaben. Dieses Buch kann uneingeschränkt auch als Begleitbuch für Terminal-Programmierer empfohlen werden. Das Buch ist dabei insbesondere für solche Programmieranfänger zu empfehlen, die sich beim Erlernen der Programmierung schwer tun. Mehr Informationen und auch Links zu Leseproben gibt es auf der Java-Hamster-Website www.java-hamster-modell.de.

Es gibt heutzutage unzählige Lehrbücher zu Java und es ist schwer zu sagen, für wen welches Buch das geeignetste ist. Viele Bücher stehen bei Amazon oder Google-Books zumindest auszugsweise online zur Verfügung und ich kann nur empfehlen, über diese Online-Angebote selbst einmal in die Bücher hineinzuschnuppern. Neben dem Java-Hamster-Buch kann ich die beiden weiteren Bücher insbesondere für Programmieranfänger empfehlen:

Ratz, D., Scheffler, J., Seese, D. und Wiesenberger, J.: „Grundkurs Programmieren in Java, Band 1“, Hanser-Verlag.

Heinisch, C., Müller, F. und Goll, J.: „Java als erste Programmiersprache“, Vieweg+Teubner.

Wenn Sie noch überhaupt keine Kenntnisse der Programmierung haben, empfehle ich Ihnen, sich Informationen zu den allgemeinen Grundlagen der Programmierung auf der folgenden Website durchzulesen (Leseprobe des Java-Hamster-Buches): <http://www-is.informatik.uni-oldenburg.de/~dibo/hamster/leseprobe/node3.html>.