Benutzungshandbuch

Java-Kara-Solist-Simulator

Version 2.0 (25.08.2010)

Dietrich Boles

Universität Oldenburg

Inhaltsverzeichnis

1 Einleitung

Programmieranfänger haben häufig Schwierigkeiten damit, dass sie beim Programmieren ihre normale Gedankenwelt verlassen und in eher technisch-orientierten Kategorien denken müssen, die ihnen von den Programmiersprachen vorgegeben werden. Gerade am Anfang strömen oft so viele inhaltliche und methodische Neuigkeiten auf sie ein, dass sie das Wesentliche der Programmierung, nämlich das Lösen von Problemen, aus den Augen verlieren.

1.1 Das Kara-Modell

Das Kara-Modell ist mit dem Ziel entwickelt worden, dieses Problem zu lösen. Mit dem Kara-Modell wird Programmieranfängern ein einfaches, aber mächtiges Modell zur Verfügung gestellt, mit dessen Hilfe Grundkonzepte der (imperativen) Programmierung auf spielerische Art und Weise erlernt werden können. Programmierer entwickeln so genannte „Kara-Programme“, mit denen sie einen virtuellen Marienkäfer namens Kara durch eine virtuelle Landschaft steuern und bestimmte Aufgaben lösen lassen.

Prinzipiell ist das Kara-Modell programmiersprachenunabhängig. Zum praktischen Umgang mit dem Modell wurde im Java-Kara-Simulator jedoch bewusst die Programmiersprache Java als Grundlage gewählt. Java – auch als „Sprache des Internet“ bezeichnet – ist eine moderne Programmiersprache, die sich in den letzten Jahren sowohl im Ausbildungsbereich als auch im industriellen Umfeld durchgesetzt hat.

Das Kara-Modell wurde an der ETH Zürich entwickelt und ist nun in die schweizerische Bildungsplattform SwissEduc integriert. Viele Informationen rund um das Modell stehen auf der Kara-Website zur Verfügung:

http://www.swisseduc.ch/informatik/karatojava/

1.2 Der Java-Kara-Solist-Simulator

Beim Kara-Modell 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 ist mit dem Java-Kara-Solist-Simulator möglich. Dieser stellt eine Reihe von Werkzeugen zum Erstellen und Ausführen von Kara-Programmen zur Verfügung: einen Editor zum Eingeben und Verwalten von Kara-Programmen, einen Compiler zum Übersetzen von Kara-Programmen, einen Territoriumsgestalter zum Gestalten und Verwalten von Territorien, einen Interpreter

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

Der vorliegende Simulator wird dabei zur Abgrenzung zum Original Java-Kara-Simulator, der über die Kara-Website geladen werden kann, als Java-Kara-Solist-Simulator bezeichnet.

1.3 Voraussetzungen

Zielgruppe des Kara-Modells sind Schüler oder Studierende ohne Programmiererfahrung, die die Grundlagen der (imperativen) Programmierung erlernen und dabei ein wenig Spaß haben wollen. Kenntnisse im Umgang mit Computern sind wünschenswert. Der Kara-Simulator ist dabei kein Lehrbuch sondern ein Werkzeug, das das Erlernen der Programmierung unterstützt. Auf gute Lehrbücher zum Erlernen der Programmierung wird in Kapitel 8 (Literatur zum Programmieren lernen) hingewiesen.

1.4 Anmerkungen

Der Kara-Solist-Simulator wurde mit dem Tool „Solist“ (Version 2.0) 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 Kara-Simulators ist die Bühne das Territorium, auf dem der Marienkäfer als Solist (es gibt nur einen Marienkäfer) agiert. Requisiten sind Pilze, Baumstümpfe und Kleeblätter. Wenn Ihnen also beim Umgang mit dem Kara-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 Änderungen von Version 2.0 gegenüber Version 1.0

Wesentliche Änderung gegenüber Version 1.0 des Java-Kara-Solist-Simulators ist, dass es nun im Programm-Menü bzw. in der Toolbar spezielle Menü-Items zum Neu-Erstellen, Laden und Speichern von Kara-Programmen gibt.

1.6 Aufbau des Benutzerhandbuch

Das Benutzungshandbuch ist in 8 Kapitel gegliedert. Nach dieser Einleitung wird in Kapitel 2 die Installation des Kara-Simulators beschrieben. Kapitel 3 stellt ausführlich das Kara-Modell vor. Eine Übersicht über den Aufbau von Kara-Programmen gibt Kapitel 4. Wie Sie Ihr erstes Kara-Programm zum Laufen bringen, erfahren Sie kurz und knapp in Kapitel 5. Dieses enthält eine knappe Einführung in die Elemente und die Bedienung des Kara-Simulators. Sehr viel ausführlicher geht dann Kapitel 6 auf die einzelnen Elemente und die Bedienung des Kara-Simulators ein. Kapitel 7 demonstriert an einigen Beispielprogrammen die Programmierung mit dem Marienkäfer. Kapitel 8 enthält letztendlich Hinweise zu guter Begleitliteratur zum Erlernen der Programmierung mit Java.

2 Installation

2.1 Voraussetzungen

Voraussetzung zum Starten des Kara-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 Kara-Solist-Simulator kann von der Website www.programmierkurs-java.de/solist kostenlos herunter geladen werden. Er befindet sich in einer Datei namens *karasolistsimulator -2.0.zip*. Diese muss zunächst entpackt werden. Es entsteht ein Ordner namens *karasolistsimulator -2.0* (der so genannte *Simulator-Ordner*), in dem sich eine Datei *simulator.jar* befindet. Durch Doppelklick auf diese Datei wird der Kara-Simulator gestartet. Alternativ lässt er sich auch durch Eingabe des Befehls `java –jar simulator.jar` in einer Console starten.

Beim ersten Start sucht der Kara-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 Kara-Simulator arbeitet. In der aktuellen Version wird allerdings nur deutsch unterstützt.

3 Das Kara-Modell

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.

Das Kara-Modell ist ein spezielles didaktisches Modell zum Erlernen der Programmierung. Im Kara-Modell nimmt ein virtueller Marienkäfer die Rolle des Computers ein. Diesem Marienkäfer können ähnlich wie einem Computer Befehle erteilt werden, die dieser ausführt.

Ihnen als Programmierer werden bestimmte Aufgaben gestellt, die sie durch die Steuerung des Marienkäfers zu lösen haben. Derartige Aufgaben werden im Folgenden Marienkäfer- oder Kara-Aufgaben genannt. Zu diesen Aufgaben müssen Sie in der Programmiersprache Java Programme - Kara-Programme genannt -- entwickeln, die die Aufgaben korrekt und vollständig lösen. Die Aufgaben werden dabei nach und nach komplexer. Zum Lösen der Aufgaben müssen bestimmte Programmierkonzepte eingesetzt werden, die im Kara-Modell inkrementell eingeführt werden.

Die Grundidee des Kara-Modells ist ausgesprochen einfach: Sie als Programmierer müssen einen (virtuellen) Marienkäfer durch eine (virtuelle) Landschaft steuern und ihn gegebene Aufgaben lösen lassen.

3.1 Kara-Modell

Kara, der Marienkäfer, lebt in einer rechteckigen Welt, die aus einzelnen Feldern besteht. Er kann sich nur von Feld zu Feld bewegen. Verlässt der Käfer die Welt an einer Seite, so betritt er sie auf der gegenüberliegenden Seite wieder. Auf den Feldern gibt es verschiedene Arten von Objekten (siehe auch Abbildung 3.1):

- Unbewegliche Baumstümpfe: Kara kann Felder mit einem Baumstumpf nicht betreten. Auf einem Feld mit einem Baumstumpf kann kein anderer Gegenstand liegen.
- Kleeblätter: Der Marienkäfer kann beliebig viele Kleeblätter aufnehmen, und er hat einen unerschöpflichen Vorrat an Blättern zum Ablegen. Kara kann auf einem Kleeblatt stehen, aber auf einem Feld kann höchstens ein Kleeblatt liegen.

- Verschiebbare Pilze: Der Käfer kann nicht auf einem Feld stehen, das von einem Pilz belegt ist. Ein Pilz kann auf einem Kleeblatt stehen. Läuft der Käfer in einen Pilz, so verschiebt er ihn geradeaus ins nächste Feld. Er ist aber zu schwach, um gleichzeitig zwei Pilze zu verschieben.

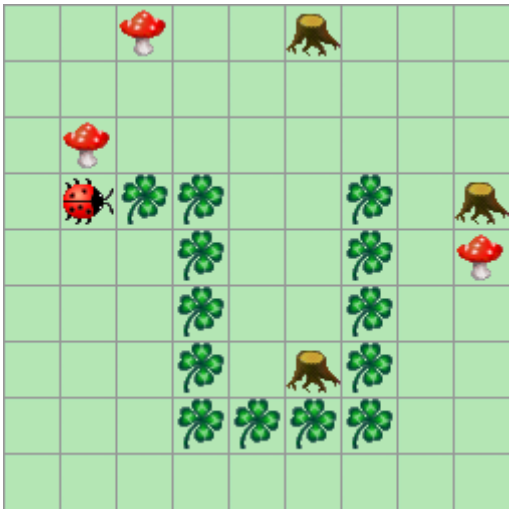


Abb. 3.1: Kara-Landschaft

3.2 Kara-Befehle

Kara kennt die folgenden Befehle:

- `void move();` : Schritt vorwärts
- `void turnLeft();` : 90°-Linksdrehung
- `void turnRight();` : 90°-Rechtsdrehung
- `void putLeaf();` : Kleeblatt hinlegen
- `void removeLeaf();` : Kleeblatt aufnehmen

sowie die folgenden Testbefehle:

- `boolean treeFront()` : Kara vor Baum?
- `boolean treeLeft()` : Baum links von Kara?
- `boolean treeRight()` : Baum rechts von Kara?
- `boolean onLeaf()` : Kara auf Kleeblatt?
- `boolean mushroomFront()` : Kara vor Pilz?

4 Kara-Programme

Eine schöne Einführung in die Entwicklung von Kara-Programmen gibt das Skript von Gerhard Bitch, das über folgende URL erreichbar ist:

http://www.swisseduc.ch/informatik/karatojava/javakara/docs/bitsch-gerhard-skript-java-javakara.pdf

Allerdings gibt es folgende Unterschiede zwischen dem Original-Java-Kara-Simulator und dem hier vorliegenden Kara-Solist-Simulator zu beachten:

- Original-Java-Kara-Programme beginnen immer mit einer import-Anweisung. Die entfällt im Kara-Solist-Simulator.
- Programme des Original-Java-Kara-Simulators müssen in einer Klasse definiert werden, deren Name identisch ist mit dem Namen der Datei (plus .java), in der sie abgespeichert werden, z.B. „`public class FindeBaum extends JavaKaraProgram`“. Dahingegen beginnen Kara-Programme im Light-Simulator immer mit „`public class Solist extends Marienkaefer`“. Abspeichern ist hier nicht notwendig.

Das folgende Programm des Original-Java-Kara-Simulators

```
import javakara.JavaKaraProgram;

public class FindeBaum extends JavaKaraProgram {

  // hier können Sie eigene Methoden definieren

  public void myProgram() {
    // hier kommt das Hauptprogramm hin, zB:
    while (!kara.treeFront()) {
      kara.move();
    }
  }
}
```

Das in einer Datei namens „FindeBaum.java“ abgespeichert werden muss, wird im Kara-Solist-Simulator wie folgt umgesetzt:

```
public class Solist extends Marienkaefer {

  // hier können Sie eigene Methoden definieren

  public void myProgram() {
    // hier kommt das Hauptprogramm hin, zB:
    while (!kara.treeFront()) {
      kara.move();
```

```
    }
  }
}
```

Einige weitere Anmerkungen:

- Anstelle der Prozedur „`public void myProgram`“ kann im Kara-Solist-Simulator auch eine Prozedur „`void main`“ als Start-Prozedur definiert werden.
- Kara-Befehle können im Kara-Solist-Simulator auch ohne den Präfix „`kara.`“ aufgerufen werden, also bspw. „`move();`“ anstelle von „`kara.move();`“.
- Werden neue Prozeduren oder Funktionen definiert, so können diese auch als neue Befehle für Kara aufgefasst werden. Ihr Aufruf ist also auch über das Präfix „`kara.`“ möglich.

5 Ihr erstes Kara-Programm

Nachdem Sie den Kara-Simulator gestartet haben, öffnet sich ein Fenster, das in etwa dem in Abbildung 5.1 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 Kara-Programme ein und im Simulationsbereich führen Sie Kara-Programme aus.

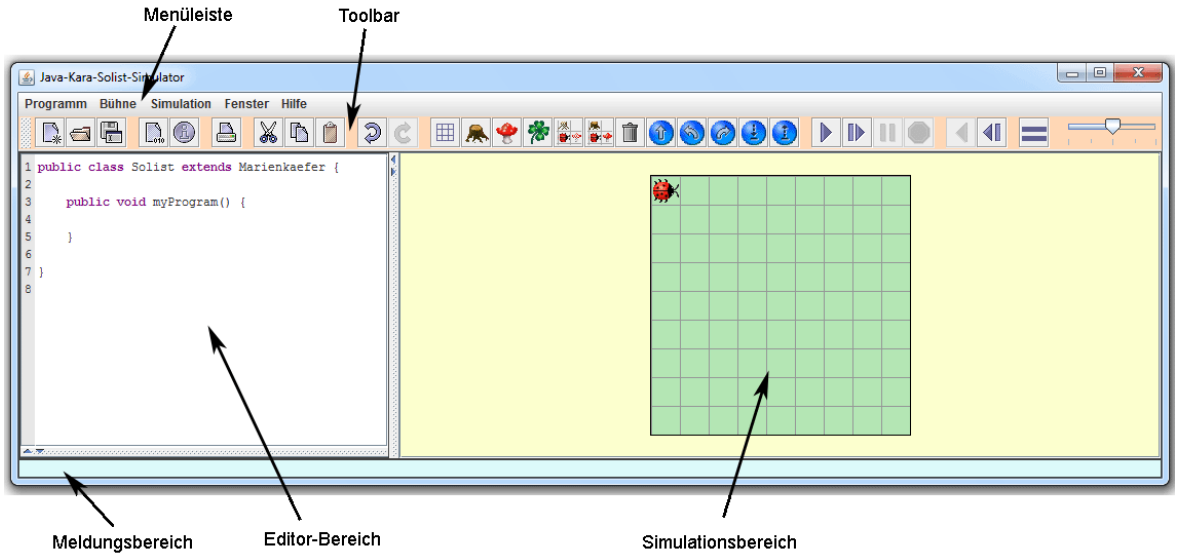


Abb. 5.1: Kara-Simulator nach dem Öffnen

Um den Marienkäfer ein wenig näher kennen zu lernen, empfehle ich Ihnen, als erstes zunächst die Kara-Befehle einmal auszuprobieren. Wie das geht, wird in Abschnitt 5.1 erläutert. Anschließend wird in den darauf folgenden Abschnitten im Detail beschrieben, was Sie machen müssen, um Ihr erstes Kara-Programm zu schreiben und auszuführen. Insgesamt müssen/können fünf Stationen durchlaufen werden:

- Gestaltung eines Territoriums
- Eingeben eines Kara-Programms
- Compilieren eines Kara-Programms
- Ausführen eines Kara-Programms
- Debuggen eines Kara-Programms

5.1 Ausprobieren der Kara-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 Marienkäfer kennt.

Abb. 5.2: Befehlsfenster

Sie können die jeweiligen Befehle ausführen, indem Sie den Maus-Cursor auf den entsprechenden Button verschieben und diesen anklicken. Klicken Sie bspw. auf den Button „void move()“, dann hüpft der Marienkäfer in seiner aktuellen Blickrichtung eine Kachel nach vorne (oder es erscheint eine Fehlermeldung, wenn der Marienkäfer bspw. vor einem Baumstumpf steht).

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 Simulationsbeereich ausgegeben, sondern immer der jeweilige Endzustand nach Abarbeitung der Funktion.

5.2 Gestaltung eines Territoriums

Nun wollen wir ein Territorium aufbauen, in dem unser erstes Programm ablaufen soll. Das geschieht im Simulationsbereich. Hier wird das Territorium dargestellt. Zur Gestaltung des Territoriums müssen Sie die Buttons 12 – 23 (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.

Zunächst werden wir die Größe des Territoriums anpassen. Klicken Sie dazu auf den Button „Territoriumsgröße ändern“ (12. Button von links). Es erscheint eine Dialogbox, in der Sie die gewünschte Anzahl an Reihen und Spalten eingeben können. Um die dort erscheinenden Werte (jeweils 9) ändern zu können, klicken Sie mit der Maus auf das entsprechende Eingabefeld. Anschließend können Sie den Wert mit der Tastatur eingeben. Nach der Eingabe der Werte klicken Sie bitte auf den OK-Button. Die Dialogbox schliesst sich und das Territorium erscheint in der angegebenen Größe.

Nun werden wir den Marienkäfer, der immer im Territorium sitzt, umplatzieren. Dazu klicken wir den Marienkäfer an und ziehen (man spricht auch von „draggen“) ihn mit gedrückter Maustaste, auf die gewünschte Kachel. Dann lassen wir die Maustaste los.

Nun wollen wir auf einigen Kacheln Baumstümpfe platzieren. Hierzu dient der Button „Baumstumpf setzen“ (13. Button von links). Wenn Sie ihn mit der Maus anklicken, erscheint am Maus-Cursor ein Baumstumpf-Symbol. Bewegen Sie die Maus nun über die Kachel, auf der ein Baumstumpf platziert werden soll, und klicken Sie dort die Maus. Auf der Kachel erscheint nun ein Baumstumpf-Symbol. Baumstümpfe können nicht auf Kacheln platziert werden, auf denen sich bereits andere Elemente befinden.

Wenn Sie die Shift-Taste Ihrer Tastatur drücken und gedrückt halten und anschließend den Button „Baumstumpf setzen“ der Toolbar anklicken, haben Sie die Möglichkeit, mehrere Baumstümpfe im Territorium zu platzieren, ohne vorher jedes Mal erneut den Button anklicken zu müssen. Solange Sie die Shift-Taste gedrückt halten und eine Kachel anklicken, wird dort ein Baumstumpf platziert.

Pilze werden ähnlich wie Baumstümpfe auf Kacheln platziert. Nutzen Sie dazu den „Pilz setzen“-Button (14. Button von links). Pilze können nicht auf Kacheln platziert werden, auf denen bereits ein anderer Pilz existiert oder auf der sich der Marienkäfer befindet.

Analog zu Baumstümpfen und Pilzen werden Kleeblätter im Territorium platziert, und zwar mit dem „Kleeblatt setzen“-Button (15. Button links). Kleeblätter können nicht auf Kacheln platziert werden, auf denen bereits ein anderes Kleeblatt oder ein Baumstumpf existiert.

Genauso wie der Marienkäfer können auch Baumstümpfe, Pilze und Kleeblätter durch Anklicken und Bewegen der Maus bei gedrückter linker Maustaste im Territorium umplatziert werden.

Mit den Buttons 16 und 17 kann die Weltsicht festgelegt werden. Wenn Sie Button 16 der Toolbar anklicken („Karas Weltsicht“), werden nur Objekte, die Kara sieht, in voller Farbe sichtbar dargestellt. Wenn Sie Button 17 anklicken („globale Weltsicht“) sind alle Objekte des Territoriums voll sichtbar.

Möchten Sie bestimmte Kacheln im Territorium wieder leeren, so aktivieren Sie den „Kachel löschen“-Button (18. Button von links). Klicken Sie anschließend auf die Kacheln, die geleert werden sollen. Der Button ist so lange aktiviert, bis er erneut oder ein anderer Gestaltungsbutton angeklickt wird.

Über die Buttons 19 bis 23 können Sie interaktiv die Befehle move, turnLeft, turnRight, putLeaf und removeLeaf ausführen und damit den Zustand des Marienkäfers und den Zustand des Territoriums verändern.

Über das Menü „Bühne“ können Territorien auch in Datei gespeichert und später wieder geladen werden.

So, jetzt wissen Sie eigentlich alles, was notwendig ist, um das Territorium nach Ihren Wünschen zu gestalten.

5.3 Eingeben eines Kara-Programms

Nun begeben wir uns in den Editor-Bereich. Dort werden wir unser erstes Kara-Programm schreiben.

Unser erstes Programm soll bewirken, dass der Marienkäfer solange nach vorne läuft, bis er auf einen Baum trifft. Wir klicken in den Editor-Bereich und tippen dort wie in einem normalen Editor bzw. Textverarbeitungsprogramm, wie Microsoft Word, die entsprechenden Kara-Befehle ein, so dass letztlich folgendes im Eingabebereich steht (wenn Sie den Simulator neu starten, steht dieses Programm bereits im Editor-Bereich):

```
public class Solist extends Marienkaefer {

  // hier können Sie eigene Prozeduren und
  // Funktionen definieren

  public void myProgram() {
    // hier kommt das Hauptprogramm hin, z.B:
    while (!kara.treeFront()) {
      kara.move();
    }
  }

  // hier können Sie ebenfalls eigene Prozeduren und
  // Funktionen definieren

}
```

Das ist unser erstes Kara-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, …).

5.4 Compilieren eines Kara-Programms

Nachdem wir unser Kara-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 5.3).

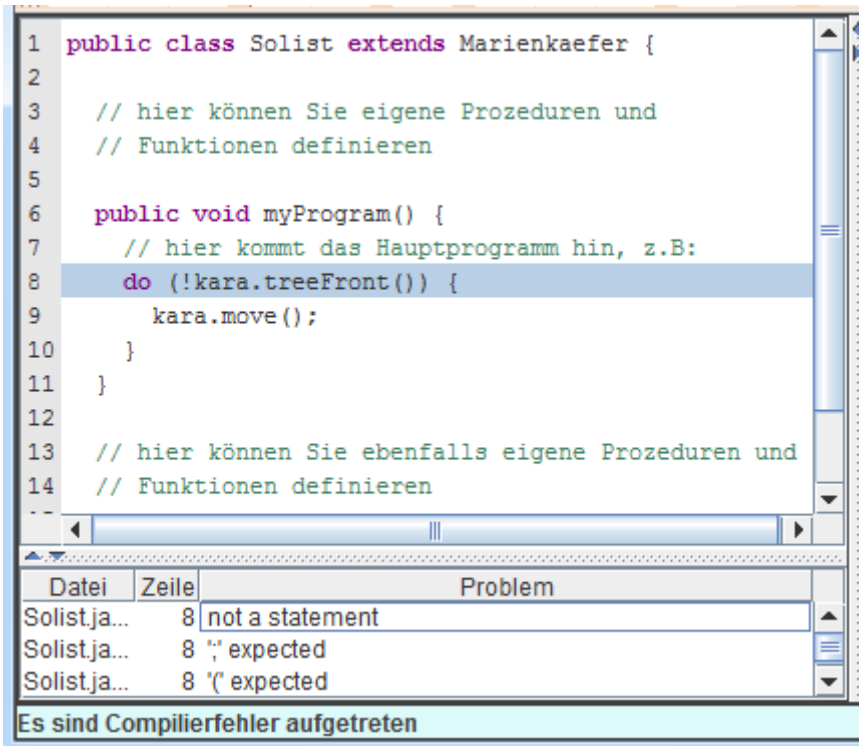


Abb. 5.3: Fehlermeldungen des Compilers

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!

5.5 Ausführen eines Kara-Programms

Nach dem erfolgreichen Compilieren ist es endlich soweit: Wir können den Marienkäfer bei der Arbeit beobachten. Macht er 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 (24. Button von links) starten wir das Programm. Wenn Sie bis hierhin alles richtig gemacht haben, sollte der Marienkäfer loslaufen und wie im Programm beschrieben, den nächsten Baum suchen. Herzlichen Glückwunsch zu Ihrem ersten Kara-Programm!

Wollen Sie die Programmausführung anhalten, können Sie dies durch Anklicken des „Simulation pausieren“-Buttons (26. Button von links) erreichen. Der Marienkäfer pausiert so lange, bis Sie wieder den „Simulation starten/fortsetzen“-Button (24. Button von links) anklicken. Dann fährt der Marienkäfer mit seiner Arbeit fort. Das Programm vorzeitig komplett abbrechen, können Sie mit Hilfe des „Simulation beenden“-Buttons (27. Button von links).

Wenn Sie ein Programm mehrmals hintereinander im gleichen Territorium ausführen wollen, können Sie mit dem „Rücksetzen“-Button (28. Button von links) den Zustand des Territoriums wieder herstellen, der vor dem letzten Ausführen des Programms Bestand hatte. Eine komplette Zurücksetzung des Territoriums auf den Zustand beim Öffnen des Kara-Simulators ist mit dem „Komplett zurücksetzen“-Button möglich (29. 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 Marienkäfer seine Arbeit. Je weiter Sie ihn nach rechts verschieben, umso schneller flitzt der Marienkäfer durchs Territorium.

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

5.6 Debuggen eines Kara-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 (30. 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 ist 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. 5.4).

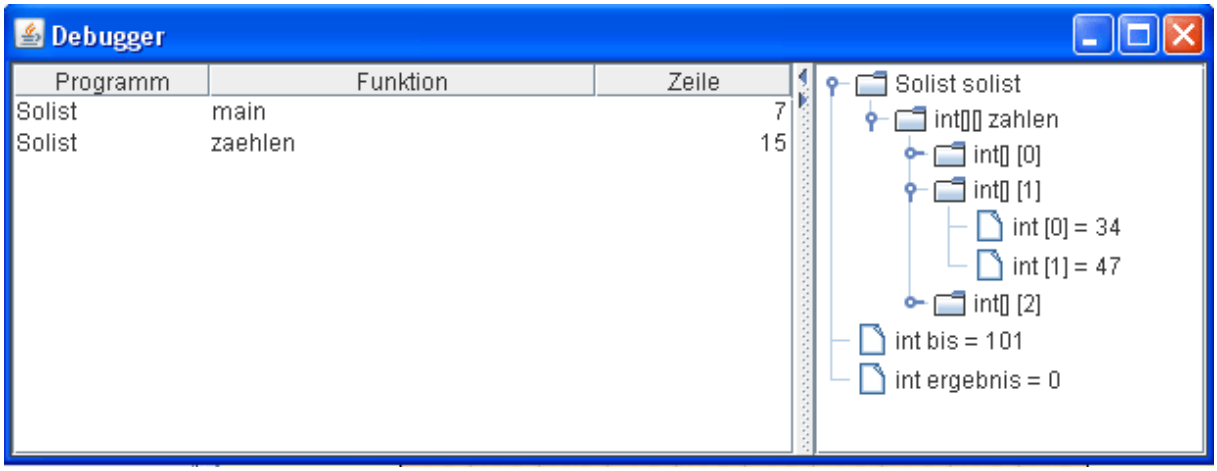


Abb. 5.4: Debugger-Fenster

Jetzt können Sie über den „Schritt“-Button (25. 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 Prodzedur-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.

5.7 Zusammenfassung

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

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

6 Bedienung des Kara-Simulators

Im letzten Abschnitt haben Sie eine kurze Einführung in die Funktionalität des Kara-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 Kara-Simulator starten, öffnen sich ein Fenster mit dem Titel „Kara-Solist-Simulator“. Abbildung 6.1 skizziert die einzelnen Komponenten des Fensters.

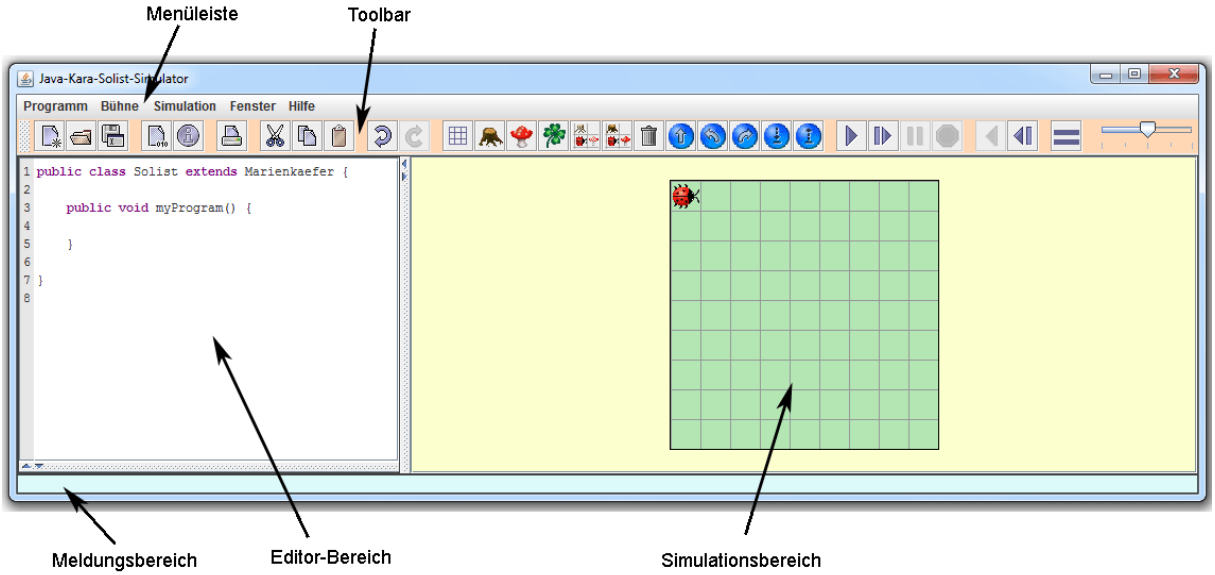


Abb. 6.1: Komponenten des Simulator-Fensters

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 Kara-Programmen erfolgt im Editor-Bereich links und die Ausführung von Kara-Programmen wird im Simulationsbereich (auch „Bühne“ genannt) visualisiert.

Als Hauptfunktionsbereiche des Kara-Simulators lassen sich identifizieren:

- Verwalten und Editieren von Kara-Programmen
- Compilieren von Kara-Programmen
- Verwalten und Gestalten von Territorien

- Interaktives Ausführen von Kara-Befehlen
- Ausführen von Kara-Programmen
- Debuggen von Kara-Programmen

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

6.1 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.

6.1.1 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.

6.1.2 Tooltipps

Als *Tooltipps* werden kleine Rechtecke bezeichnet, die automatisch auf dem Bildschirm erscheinen, wenn man den Maus-Cursor auf entsprechende Objekte platziert (siehe Abbildung 6.2). In den Tooltipps werden bestimmte Informationen ausgegeben.

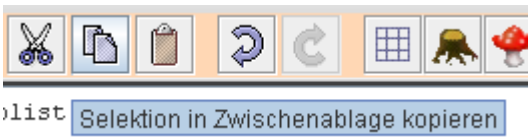


Abb. 6.2: Tooltipp

6.1.3 Button

Buttons sind Objekte der Benutzungsoberfläche, die man anklicken kann und die daraufhin eine bestimmte Aktion auslösen (siehe Abbildung 6.3). 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 Tooltipps zugeordnet, die die zugeordnete Aktion beschreiben.

Abb. 6.3: Buttons

6.1.4 Menü

Menüs befinden sich ganz oben in einem Fenster in der so genannten *Menüleiste* (siehe Abbildung 6.4). 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. 6.4: 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.

6.1.5 Toolbar

Direkt unterhalb der Menüleiste ist die so genannte *Toolbar* angeordnet (siehe Abbildung 6.5). Sie besteht aus einer Menge an Graphik-Buttons, die Alternativen zu den am häufigsten benutzten Menü-Items der Menüs darstellen.

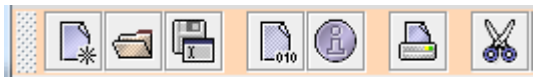


Abb. 6.5: Toolbar

6.1.6 Popup-Menü

Popup-Menüs sind spezielle Menüs, die bestimmten Elementen auf dem Bildschirm zugeordnet sind (siehe Abbildung 6.6). 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.

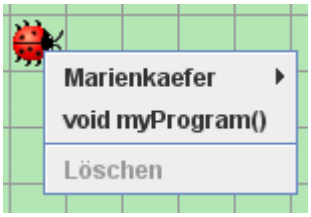


Abb. 6.6: Popup-Menü

6.1.7 Eingabefeld

Eingabefelder dienen zur Eingabe von Zeichen (siehe Abbildung 6.7). 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.

6.1.8 Dialogbox

Beim Auslösen bestimmter Aktionen erscheinen so genannte *Dialogboxen* auf dem Bildschirm (siehe Abbildung 6.7). 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 Dialoxbox 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.

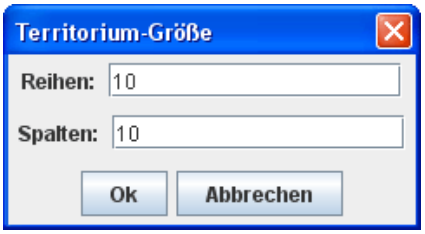


Abb. 6.7: Dialogbox mit Eingabefeldern

6.1.9 Dateiauswahl-Dialogbox

Dateiauswahl-Dialogboxen sind spezielle Dialogboxen, die zum Speichern und Öffnen von Dateien benutzt werden (siehe Abbildung 6.8). Sie spiegeln im Prinzip das Dateisystem wider und enthalten Funktionalitäten zum Verwalten von Dateien und Ordnern.

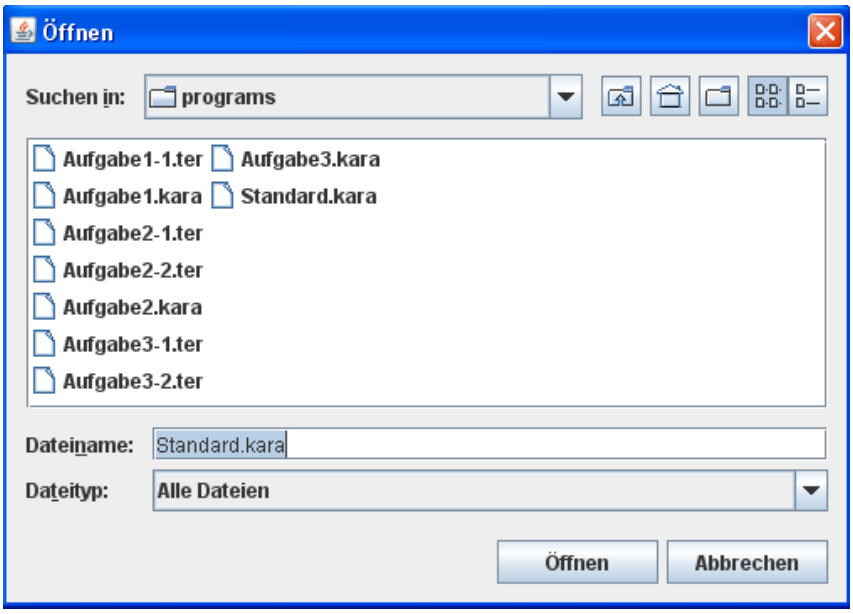


Abb. 6.8: Dateiauswahl-Dialogbox

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.

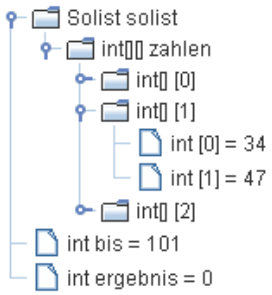
Dateiauswahl-Dialogboxen stellen jedoch noch zusätzliche Funktionalitäten bereit. Durch Doppelklick auf einen Ordner kann man in den entsprechenden Ordner wechseln. Es werden dann anschließend die Dateien und Unterordner dieses Ordners im mittleren Bereich angezeigt. Um zu einem übergeordneten Ordner 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.

6.1.10 Elementbaum

Ein *Elementbaum* repräsentiert Elemente und strukturelle Beziehungen zwischen ihnen, bspw. die Ordner und Dateien des Dateisystems (siehe Abbildung 6.9).



Abbi. 6.9: Elementbaum mit Verzeichnissen und Dateien

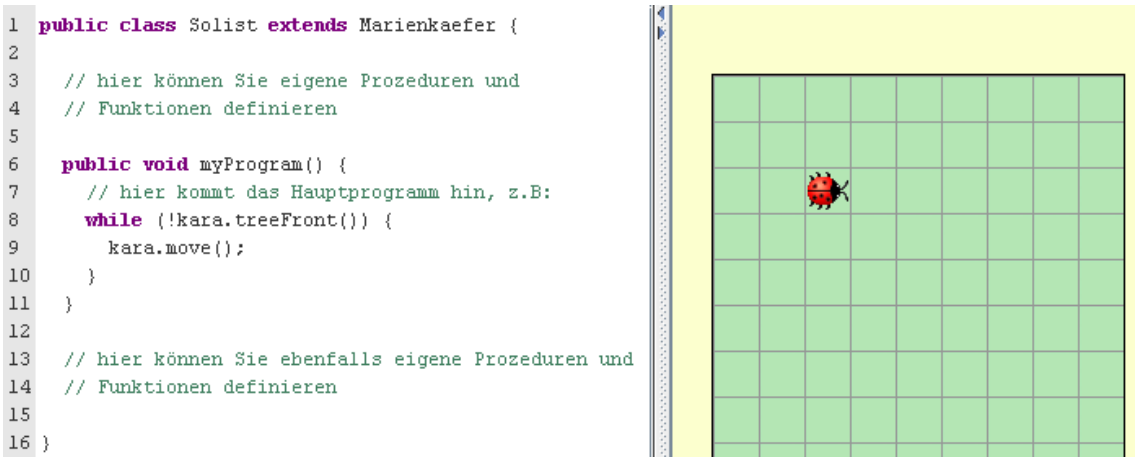
Unterschiedliche Elementtypen werden dabei durch unterschiedliche Symbole dargestellt, hinter denen entsprechende Bezeichnungen erscheinen. Durch Anklicken

der Symbole auf der linken Seite kann man Strukturen öffnen und schließen, d.h. Unterstrukuren 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.

6.1.11 Split-Pane

Eine Split-Pane ist ein Element, das aus zwei Bereichen und einem Balken besteht. (siehe Abbildung 6.10). 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.



Abbi. 6.10: Split-Pane

6.2 Verwalten und Editieren von Kara-Programmen

Das Schreiben von Programmen bzw. genauer gesagt das Schreiben des Sourcecodes von Programmen bezeichnet man als *Editieren*. Im Kara-Simulator dient der Editor-Bereich zum Editieren von Kara-Programmen (siehe Abbildung 6.11)

```
1  public class Solist extends Marienkaefer {
2
3    // hier können Sie eigene Prozeduren und
4    // Funktionen definieren
5
6    public void myProgram() {
7      // hier kommt das Hauptprogramm hin, z.B:
8      while (!kara.treeFront()) {
9        kara.move();
10     }
11   }
12
13   // hier können Sie ebenfalls eigene Prozeduren und
14   // Funktionen definieren
15
16 }
17
```

Abb. 6.11: Editor-Bereich des Kara-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 Tooltipp, der die Funktionalität des Buttons anzeigt. Die für das Editieren von Programmen wichtigen Buttons der Toolbar werden in Abbildung 6.12 skizziert.

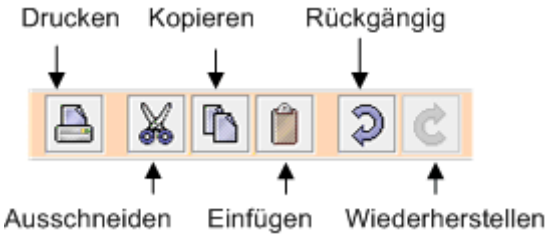


Abb. 6.12: Editor-Buttons der Toolbar

6.2.1 Schreiben eines neuen Kara-Programms

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

6.2.2 Ändern des aktuellen Kara-Programms

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

6.2.3 Löschen des aktuellen Kara-Programm

Komplett löschen können Sie das aktuelle Kara-Programm des Editor-Bereichs, indem Sie den kompletten Sourcecode mit der Maus selektieren und dann in der Toolbar den „Ausschneiden“-Button (7. Button von links) drücken. Alternativ können Sie in der Toolbar den „Neu“-Button anklicken (1. Button von links).

6.2.4 Abspeichern des aktuellen Kara-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 in der Toolbar den „Speichern“-Button (3. Button von links) anklicken. Es öffnet sich eine Dateiauswahl-Dialogbox, über die Sie die gewünschte Datei auswählen können.

6.2.5 Öffnen eines einmal abgespeicherten Kara-Programms

Möchten Sie ein einmal abgespeichertes Kara-Programm wieder in den Editorbereich laden, können Sie dies über den „Laden“-Button der Toolbar (2. Button von links) tun. Nach dem Anklicken des Buttons 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.

6.2.6 Drucken eines Kara-Programms

Über den „Drucken“-Button (6. 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.

6.2.7 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 (7. 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 (8. 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 (9. 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 (10. 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 (11. 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.

6.3 Compilieren von Kara-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 Kara-Simulator der „Compilieren“-Button (4. Button der Toolbar von links) bzw. das Menü-Item „Compilieren“ im „Programm“-Menü (siehe Abbildung 6.13).

An der Farbe des Compilier-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) compiliert haben. Erscheint der Button in einer neutralen Farbe, ist kein Compilieren notwendig.

6.3.1 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.

6.3.2 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 6.13). 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 Fehlermeldungsbereich schließt sich dann automatisch wieder.

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.

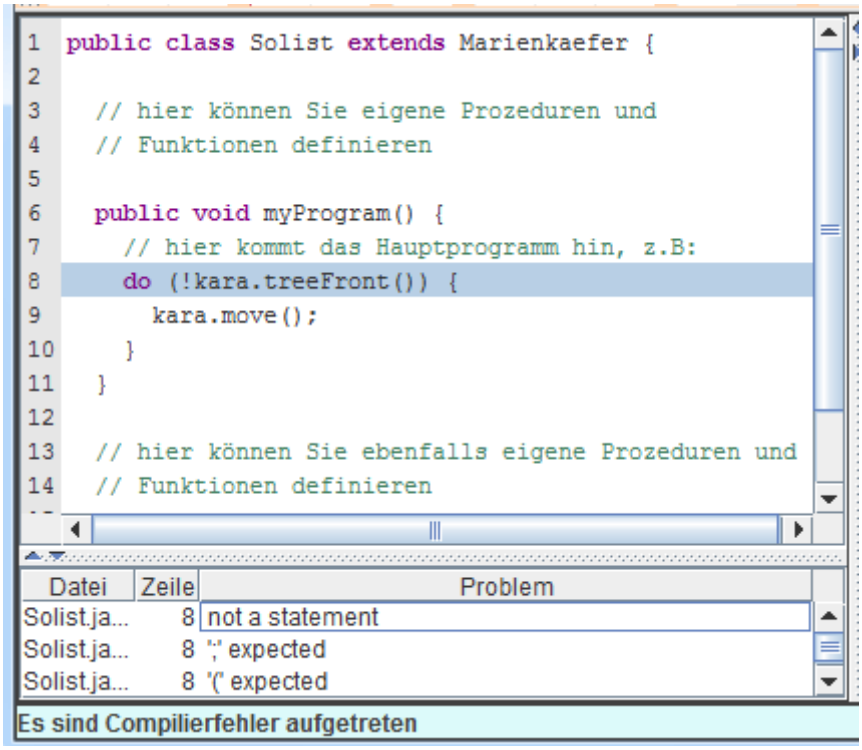


Abb. 6.13: Fehlermeldungen des Compilers

6.4 Verwalten und Gestalten von Territorien

Das Territorium umfasst standardmäßig 9 Reihen und 9 Spalten. Der Marienkäfer steht auf der Kachel ganz oben links, also der Kachel mit den Koordinaten (0/0). Er schaut nach Osten.

In der Toolbar dienen die Buttons 12 – 23 von links zum Gestalten des Territoriums. Über das Menü „Bühne“ können Territorien verwaltet werden (siehe auch Abbildung 6.14).

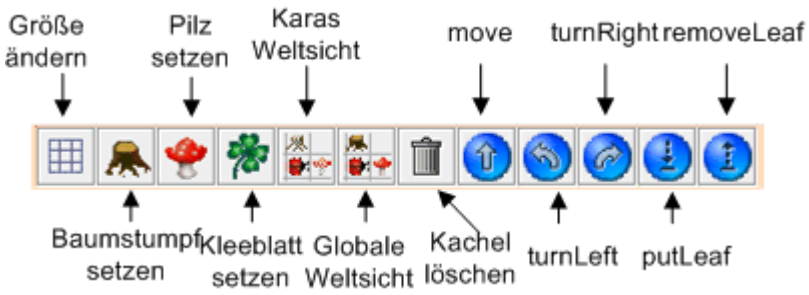


Abb. 6.14: Territorium-Buttons der Toolbar

Normalerweise sollten Sie ein Territorium vor der Ausführung eines Programms gestalten. Es ist jedoch auch möglich, während der Programmausführung noch Umgestaltungen vorzunehmen.

6.4.1 Verändern der Größe des Territoriums

Durch Anklicken des „Größe ändern“-Buttons (12. Toolbar-Button von links) können Sie die Größe des Territoriums verändern. Es öffnet sich eine Dialogbox mit zwei Eingabefelder, in denen Sie die gewünschte Reihen- und Spaltenanzahl eingeben können. Nach Drücken des OK-Buttons schließt sich die Dialogbox und das Territorium erscheint in der angegebenen Größe.

Achtung: Wenn Sie das Territorium verkleinern, werden Objekte, die sich dann außerhalb des Territoriums befinden würden, gelöscht. Steht der Marienkäfer beim Verkleinern auf einer Kachel außerhalb der neuen Territoriumsgröße, wird er auf die Kachel (0/0) versetzt (sich u.U. dort befindende Baumstümpfe oder Pilze werden zuvor gelöscht).

6.4.2 Umplatzieren des Marienkäfers im Territorium

Um den Marienkäfer im Territorium auf eine andere Kachel zu platzieren, klicken Sie ihn mit der Maus an und ziehen ihn bei gedrückter Maustaste auf die gewünschte Kachel.

6.4.3 Setzen der Blickrichtung des Marienkäfers

Um die Blickrichtung des Marienkäfers zu ändern, klicken Sie bitte den „turnLeft“-Button (20. Toolbar-Button von links) an. Bei jedem Klick auf diesen Button dreht sich der Marienkäfer um 90 Grad linksum. Analog funktioniert der „turnRight“-Button (21. Button der Toolbar von links) für eine Rechtsdrehung um 90 Grad.

6.4.4 Platzieren von Baumstümpfen auf Kacheln des Territorium

Um auf einer Kachel des Territoriums einen Baumstumpf zu platzieren, klicken Sie in der Toolbar den „Baumstumpf setzen“-Button (13. Toolbar-Button von links). Klicken Sie anschließend die entsprechende Kachel an. Der Baumstumpf wird dort platziert (insofern sich dort nicht ein anderes Objekt befindet).

Um nicht für jedes Platzieren eines Baumstumpfes erneut den „Baumstumpf setzen“-Button anklicken zu müssen, können Sie auch folgendes tun: Drücken Sie die Shift-Taste der Tastatur und Klicken bei gedrückter Shift-Taste den „Baumstumpf setzen“-Button an. Solange Sie nun die Shift-Taste gedrückt halten, können Sie durch Anklicken einer Kachel des Territoriums dort einen Baumstumpf platzieren.

Baumstümpfe können auch im Territorium umplaziert werden. Klicken Sie dazu die entsprechende Kachel an und ziehen Sie den dortigen Baumstumpf bei gedrückter Maustaste auf die gewünschte Kachel.

6.4.5 Platzieren von Pilzen auf Kacheln des Territorium

Um auf einer Kachel des Territoriums einen Pilz zu platzieren, klicken Sie in der Toolbar den „Pilz setzen“-Button (14. Toolbar-Button von links). Klicken Sie anschließend die entsprechende Kachel an. Der Pilz wird dort platziert (insofern sich dort nicht ein anderer Pilz, ein Baumstumpf oder der Marienkäfer befindet).

Um nicht für jedes Platzieren eines Pilz erneut den „Pilz setzen“-Button anklicken zu müssen, können Sie auch folgendes tun: Drücken Sie die Shift-Taste der Tastatur und Klicken bei gedrückter Shift-Taste den „Pilz setzen“-Button an. Solange Sie nun die Shift-Taste gedrückt halten, können Sie durch Anklicken einer Kachel des Territoriums dort einen Pilz platzieren.

Pilze können auch im Territorium umplaziert werden. Klicken Sie dazu die entsprechende Kachel an und ziehen Sie den dortigen Pilz bei gedrückter Maustaste auf die gewünschte Kachel.

6.4.6 Platzieren von Kleeblättern auf Kacheln des Territorium

Um auf einer Kachel des Territoriums ein Kleeblatt zu platzieren, klicken Sie in der Toolbar den „Kleeblatt setzen“-Button (15. Toolbar-Button von links). Klicken Sie anschließend die entsprechende Kachel an. Das Kleeblatt wird dort platziert (insofern sich dort kein anderes Kleeblatt oder ein Baumstumpf befindet).

Um nicht für jedes Platzieren eines Kleeblatts erneut den „Kleeblatt setzen“-Button anklicken zu müssen, können Sie auch folgendes tun: Drücken Sie die Shift-Taste der Tastatur und Klicken bei gedrückter Shift-Taste den „Kleeblatt setzen“-Button an. Solange Sie nun die Shift-Taste gedrückt halten, können Sie durch Anklicken einer Kachel des Territoriums dort ein Kleeblatt platzieren.

Kleeblätter können auch im Territorium umplaziert werden. Klicken Sie dazu die entsprechende Kachel an und ziehen Sie das dortige Kleeblatt bei gedrückter Maustaste auf die gewünschte Kachel.

6.4.7 Löschen von Kacheln des Territorium

Um einzelne Kacheln des Territoriums zu löschen, d.h. gegebenenfalls vorhandene Pilze, Kleeblätter bzw. Baumstümpfe zu entfernen, müssen Sie zunächst in der Toolbar den „Kachel löschen“-Button (18. Toolbar-Button von links) anklicken. Dadurch aktivieren Sie die Kachel-Löschen-Funktion. Sie erkennen dies daran, dass der Hintergrund des Buttons nun dunkler erscheint. Solange die Funktion aktiviert ist, können Sie nun durch Anklicken einer Kachel diese Kachel löschen.

Eine Deaktivierung der Kachel-Löschen-Funktion ist durch erneutes Anklicken des „Kachel löschen“-Buttons möglich. Eine Deaktivierung erfolgt automatisch, wenn ein anderer der Buttons zum Territorium-Gestalten gedrückt wird.

Eine weitere Möglichkeit, Kleeblätter, Pilze oder Baumstümpfe im Territorium wieder zu löschen, besteht darin, über den entsprechenden Elementen mit Hilfe der rechten Maustaste ein Popup-Menü zu aktivieren und das darin erscheinende Item „Löschen“ anzuklicken.

6.4.8 Ändern der Weltsicht

Mit den Buttons 16 und 17 kann die Weltsicht festgelegt werden. Wenn Sie Button 16 der Toolbar anklicken („Karas Weltsicht“), werden nur Objekte, die Kara sieht, in voller Farbe sichtbar dargestellt. Andere Objekte erscheinen transparent. Wenn Sie Button 17 anklicken („globale Weltsicht“) sind alle Objekte des Territoriums voll sichtbar.

6.4.9 Kara-Befehle

Über die Buttons 19 bis 23 können Sie interaktiv die Befehle move, turnLeft, turnRight, putLeaf und removeLeaf ausführen und damit den Zustand des Marienkäfers und den Zustand des Territoriums verändern.

6.4.10 Abspeichern eines Territoriums

Sie können einmal gestaltete Territorien in einer Datei abspeichern und später wieder laden. Zum Abspeichern des aktuellen Territoriums 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 Territorium gespeichert werden soll.

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

6.4.11 Wiederherstellen eines abgespeicherten Territoriums

Abgespeicherte Territorien 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 Territorium ist wiederhergestellt.

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

6.5 Interaktives Ausführen von Kara-Befehlen

Sie können einzelne Kara-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 6.15).

6.5.1 Befehlsfenster

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

Beim Anklicken mit der Maus werden die entsprechenden Befehle jeweils ausgeführt. Die Ausführung erfolgt dabei ohne Anzeige von Zwischenzuständen. D.h. wird bspw. eine Prozedur `turn` dadurch definiert, dass dreimal der Befehl `turnLeft` aufgerufen wird, und wird diese Prozedur `turn` im Befehlsfenster aktiviert, dreht sich der Marienkäfer tatsächlich nur einmal um 180 Grad um.

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

Abb. 6.15: Befehlsfenster

6.5.2 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.

6.5.3 Rückgabewerte von Funktionen

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

6.5.4 Befehls-Popup-Menü

Alternativ zur Benutzung des Befehlsfensters ist es auch möglich, über ein Popup-Menü die Kara-Befehle interaktiv auszuführen. Sie können dieses Popup-Menü durch Drücken der rechten Maustaste oberhalb des Marienkäfers im Territorium aktivieren (siehe Abbildung 6.16).

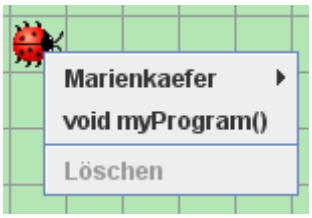


Abb. 6.16: Befehls-Popup-Menü

6.6 Ausführen von Kara-Programmen

Ausgeführt werden können (erfolgreich compilierte) Kara-Programme mit Hilfe der in Abbildung 6.17 skizzierten Buttons der Toolbar. Alle Funktionen sind darüber hinaus auch über das Menü „Simulation“ aufrufbar.

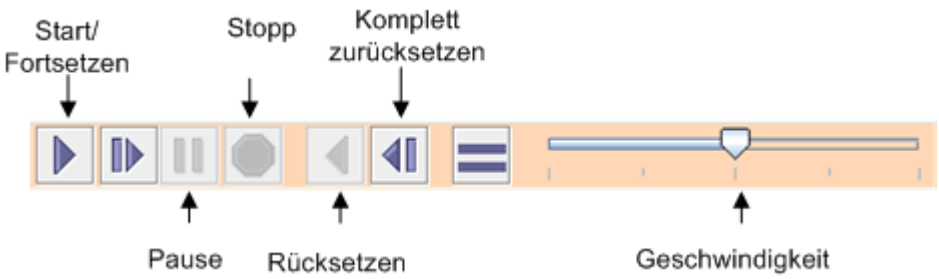


Abb. 6.17: Simulationsbuttons der Toolbar

6.6.1 Starten eines Kara-Programms

Bevor ein Kara-Programm ausgeführt werden kann, muss es erfolgreich compiliert worden sein. Gestartet werden kann das aktuelle Kara-Programm dann durch Anklicken des „Start/Fortsetzen“-Buttons (24. Toolbar-Button von links).

Nach dem Starten eines Kara-Programms wird der Marienkäfer im Territorium aktiv und tut das, was das Programm ihm vorgibt. Während des Ausführens eines Kara-Programms wird der Editor-Bereich ausgegraut, d.h. es können während der Ausführung eines Programms keine Änderungen am Sourcecode durchgeführt werden.

6.6.2 Stoppen eines Kara-Programms

Die Ausführung eines Kara-Programms kann durch Anklicken des „Stopp“-Buttons (27. Button der Toolbar von links) jederzeit abgebrochen werden.

6.6.3 Pausieren eines Kara-Programms

Möchten Sie ein in Ausführung befindliches Programm (kurzfristig) anhalten, können Sie dies durch Anklicken des „Pause“-Buttons (26. Button der Toolbar von links) tun. Wenn Sie anschließend auf den „Start/Fortsetzen“-Button klicken, wird die Programmausführung fortgesetzt.

6.6.4 Während der Ausführung eines Kara-Programms

Treten bei der Ausführung eines Programms Laufzeitfehler auf, z.B. wenn ein Marienkäfer gegen einen Baumstumpf rennt, wird eine Dialogbox geöffnet, die eine entsprechende Fehlermeldung enthält. Nach dem Anklicken des OK-Buttons in der Dialogbox wird das Kara-Programm beendet. Weiterhin öffnet sich das Konsolen-Fenster, in dem ebenfalls die Fehlermeldung ausgegeben wird.

Während der Ausführung eines Kara-Programms ist es durchaus noch möglich, das Territorium umzugestalten, also bspw. neue Pilze und Kleeblätter zu platzieren.

6.6.5 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 flitzt der Marienkäfer durchs Territorium.

6.6.6 Wiederherstellen eines Territoriums

Beim Testen eines Programms recht hilfreich ist der „Rücksetzen“-Button (28. Button der Toolbar von links). Sein Anklicken bewirkt, dass das Territorium in den Zustand zurückversetzt wird, den es vor dem letztmaligen Start eines Programms inne hatte.

Über den „Komplett Zurücksetzen“-Button (29. Button der Toolbar von links) ist eine Rücksetzung des Territoriums in den Zustand möglich, der beim Öffnen des Kara-Simulators gültig war. Sollte es irgendwann einmal bei der Benutzung des Kara-Simulators zu unerklärlichen Fehlern können, ist es mit Hilfe dieses Buttons möglich, den Kara-Simulator zu reinitialisieren.

6.7 Debuggen von Kara-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 Kara-Simulators ermöglicht während der Ausführung eines Kara-Programms das Beobachten des Programmzustands. Sie können sich während der Ausführung eines Kara-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 6.18).

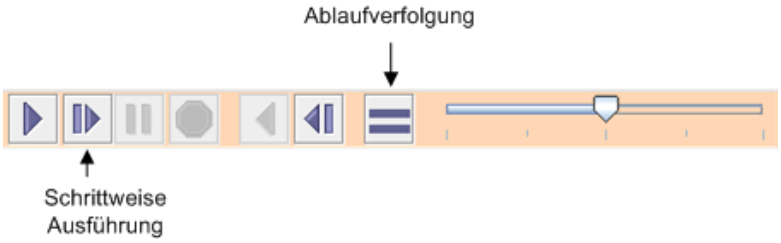


Abb. 6.18: Debugging-Buttons der Toolbar

6.7.1 Beobachten der Programmausführung

Durch Anklicken des Buttons „Ablaufverfolgung“ (30. 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 6.19). Dieses können Sie auch über das Menü „Fenster“ sichtbar bzw. unsichtbar machen.

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.

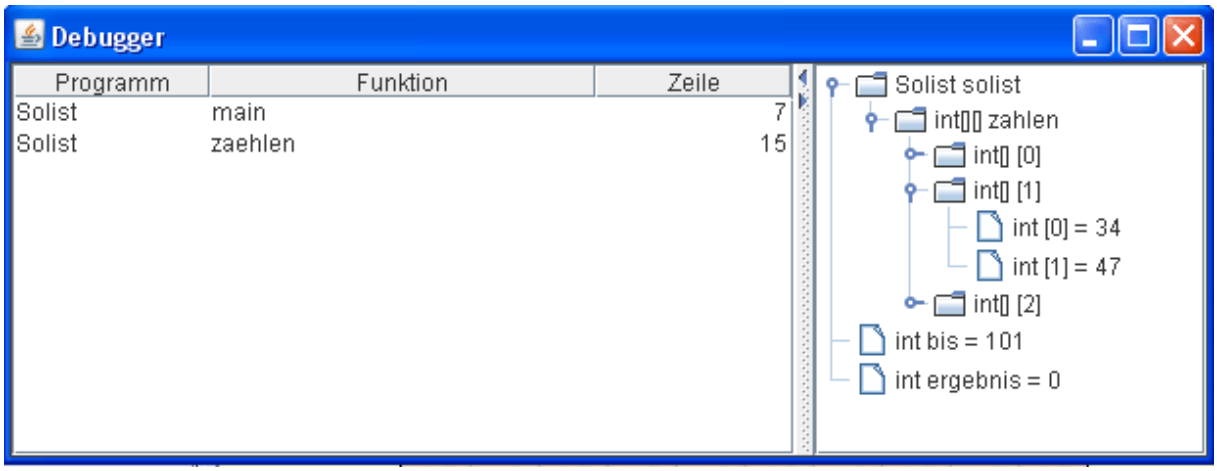


Abb. 6.19: Debugger-Fenster

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 `myProgram`- bzw. `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.

6.7.2 Schrittweise Programmausführung

Mit dem Toolbar-Button „Schrittweise Ausführung“ (25. 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.

6.7.3 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 6.20). 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.

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.

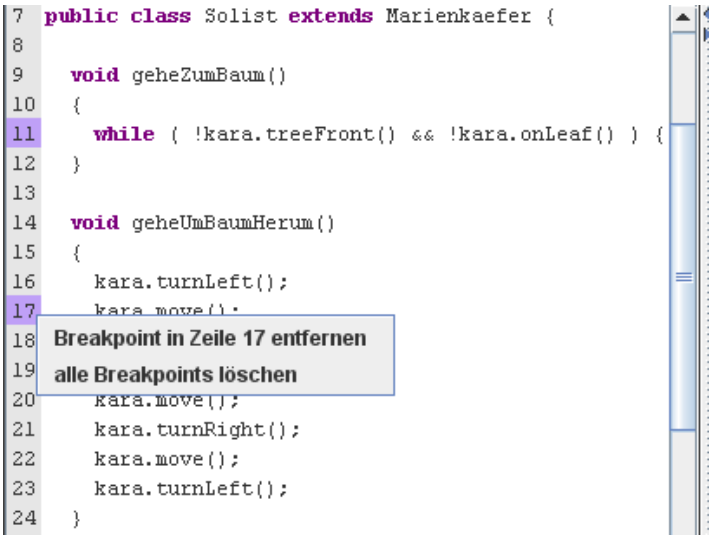


Abb. 6.20: Breakpoints

6.7.4 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.

6.8 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 6.20).

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

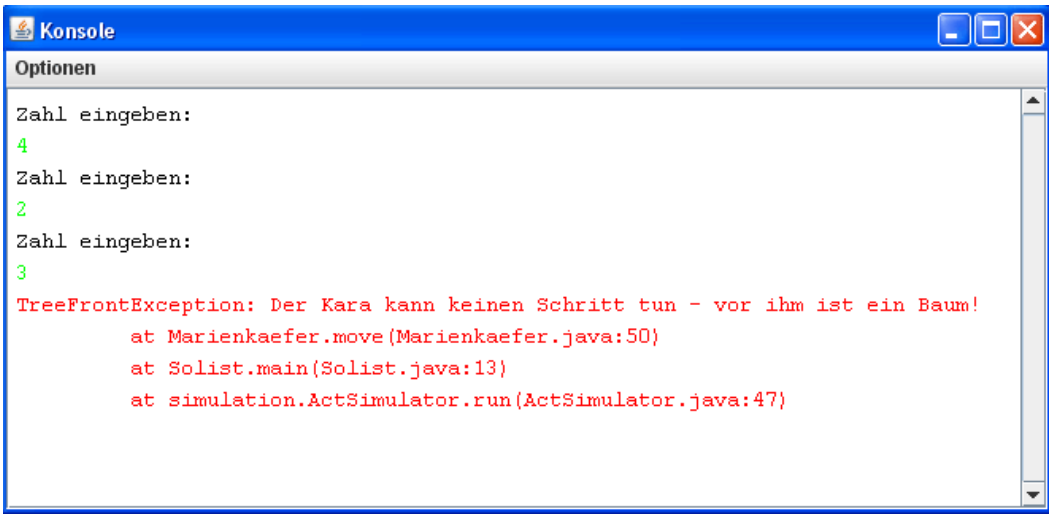


Abb. 6.20: Konsole

Ein Eingabebefehl via `System.in` blockiert das Kara-Programm so lange, bis der Nutzer eine entsprechende Eingabe in der Konsole getätigt und im Allgemeinen durch Drücken der Enter-Taste abgeschlossen hat. Die folgende Funktion `readInt` erwartet bspw. vom Nutzer die Eingabe einer Zahl in der Konsole und liefert den entsprechenden Wert an das Kara-Programm. Wenn der Nutzer in der Konsole den Wert 4 eingibt, hüpft der Marienkäfer vier Felder nach vorne (insofern er nicht gegen einen Baumstumpf rennt).

```
void main() {
    System.out.println("Zahl eingeben: ");
    int zahl = readInt();
    while (zahl > 0) {
        vor();
        zahl = zahl - 1;
    }
}

// Einlesen eines int-Wertes
int readInt() {
    try {
        java.io.BufferedReader input =
            new java.io.BufferedReader(
                new java.io.InputStreamReader(System.in));
            String eingabe = input.readLine();
       return new Integer(eingabe);
      } catch (Throwable exc) {
       return 0;
      }
}
```

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.

7 Beispielprogramme

Dem Kara-Solist-Simulator sind drei Beispielprogramme beigefügt, die Sie über das Menü „Programm“ laden können (Menü-Item „Laden…“). Weitere Aufgaben inkl. Musterlösungen finden Sie bspw. auf der Website von Kara.

7.1 Kleeblatt suchen

Aufgabe (von http://www.gierhardt.de/informatik/info11/javakara.html): Kara soll ein Kleeblatt finden, das sich in der gleichen Zeile (oder Spalte) befindet wie er selbst. Zwischen ihm und dem Kleeblatt können Bäume stehen, wobei nie zwei Bäume direkt nebeneinander stehen.

Dateiname: Aufgabe1.kara

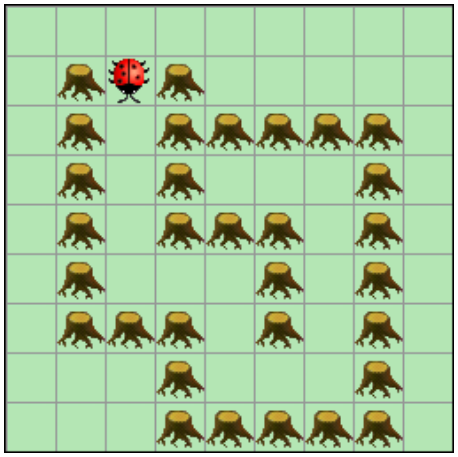
Beispiel-Territorien: Aufgabe1-1.ter und Aufgabe1-2.ter

```
public class Solist extends Marienkaefer {

  void geheZumBaum()
  {
    while ( !kara.treeFront() && !kara.onLeaf() ) { kara.move(); }
  }

  void geheUmBaumHerum()
  {
    kara.turnLeft();
    kara.move();
    kara.turnRight();
    kara.move();
    kara.move();
    kara.turnRight();
    kara.move();
    kara.turnLeft();
  }


  public void myProgram()
  {
    while (!kara.onLeaf())
      {
        geheZumBaum();
        if (!kara.onLeaf()) { geheUmBaumHerum(); }
```

```
      }

  }
}
```

7.2 Labyrinth

Aufgabe (von http://www.gierhardt.de/informatik/info11/javakara.html): Kara sucht das Ende eines einfachen Labyrinths bestehend aus Bäumen, wobei keine Löcher in den Baumreihen auftreten. Das Ende des Labyrinths ist eine "Sackgasse".



Dateiname: Aufgabe2.kara

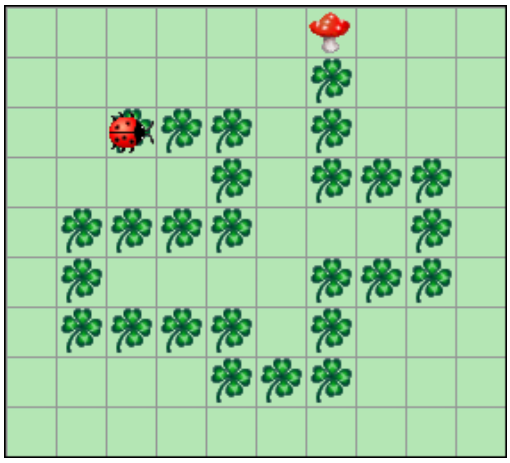
Beispiel-Territorien: Aufgabe2-1.ter und Aufgabe2-2.ter

```
public class Solist extends Marienkaefer {

  void einenSchrittWeiter()
  {
   if (!kara.treeFront())
        { kara.move(); }
   else {
         if (!kara.treeLeft())
              {
                kara.turnLeft();
                kara.move();
              }
         else {
                kara.turnRight();
                kara.move();
              }
        }
  }
```

```
  public void myProgram()
  {
    while ( !(kara.treeFront() && kara.treeLeft() && kara.treeRight() ) )
      {
        einenSchrittWeiter();
      }
  }
}
```

7.3 Pacman

Aufgabe (von http://www.gierhardt.de/informatik/info11/javakara.html): Kara spielt Pacman: Er steht auf dem ersten Kleeblatt einer langen Spur von Kleeblättern, die vor einem Pilz endet. Er soll alle Kleeblätter auffressen.



Dateiname: Aufgabe3.kara

Beispiel-Territorien: Aufgabe3-1.ter und Aufgabe3-2.ter

```
public class Solist extends Marienkaefer {

  void turnAround()
  {
    kara.turnLeft();
    kara.turnLeft();
  }

  void moveBack()
  {
    turnAround();
    kara.move();
    turnAround();
  }

  boolean vorneGehtsWeiter()
```

```
  {
    if (kara.treeFront()) { return false; }
    else { // jetzt muss man nachschauen
           kara.move();
           if (kara.onLeaf())
                {
                  moveBack();
                  return true;
                }
           else {
                  moveBack();
                  return false;
                }
         }
  }

  boolean linksGehtsWeiter()
  {
    if (kara.treeLeft()) {return false;}
    else { // jetzt muss man nachschauen
           kara.turnLeft();
           kara.move();
           if (kara.onLeaf())
                {
                  moveBack();
                  kara.turnRight();
                  return true;
                }
           else {
                  moveBack();
                  kara.turnRight();
                  return false;
                }
         }
  }


  void naechstesKleeblattSuchen()
  { if (!vorneGehtsWeiter())
         { if (linksGehtsWeiter()) { kara.turnLeft(); }
           else { kara.turnRight(); }
         }
    kara.move();
  }


  public void myProgram()
  {
    while ( !kara.mushroomFront() )
      {
        kara.removeLeaf();
        naechstesKleeblattSuchen();
      }
    kara.removeLeaf();

  }
}
```

8 Literatur zum Erlernen der Programmierung

Zum Kara-Modell gibt es viele Informationen und Materialien auf der Kara-Website: http://www.swisseduc.ch/informatik/karatojava/ Trotzdem rate ich dringend dazu, beim Erlernen der Programmierung mit Java auch Lehrbücher zu nutzen.

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. Die drei folgenden Bücher kann ich insbesondere für Programmieranfänger empfehlen:

- Boles, D.: „Programmieren spielend gelernt mit dem Java-Hamster-Modell“, Vieweg+Teubner.
- Ratz, D., Scheffler, J., Seese, D. und Wiesenberger, J.: „Grundkurs Programmieren in Java“, Hanser-Verlag.
- Heinisch, C., Müller, F. und Goll, J..: „Java als erste Programmiersprache“, Vieweg+Teubner.