Kapitel 4: Kara Sokoban Spiel

# Sokoban[[1]](#footnote-1)

Sokoban (倉庫番, japanisch „Lagerhausverwalter“) ist ein Computerspiel, das von Hiroyuki Imabayashi entwickelt und 1982 unter der Firma „Thinking Rabbit“ erstmals für verschiedene Computersysteme veröffentlicht wurde. Einige der Original-Levels tauchten später in den beiden boxxle-Versionen für den NINTENDO GameBoy wieder auf.

## Spielprinzip

In einem einfachen Spielprinzip gilt es, mit einer Spielfigur alle Objekte – meistens sind es Kisten – nacheinander auf die dafür vorgesehenen Zielfelder zu bewegen, wobei die Kisten von der Spielfigur nur geschoben und nicht gezogen werden können und auch nicht mehrere Kisten zugleich geschoben werden können. Neben dem blossen Bestehen der Levels ist eine weiterführende Herausforderung, die dafür nötigen Schritte zu minimieren.

Bei unserem Kara Sokoban müssen die Pilze auf die Zielfelder (Kleeblätter) geschoben werden.

## Beschreibung der Levels

Viele Sokoban-Spiele verwenden zur Beschreibung der Levels ein einfaches ASCII-Format[[2]](#footnote-2), welches den Austausch erleichtert. Zur Erstellung von eigenen Levels kann dabei jeder beliebige Texteditor verwendet werden. Das Beispiel mit Kara würde wie folgt aussehen:

**####**

**# .#**

**# ###**

**# @ #**

**# $ #**

**# ###**

**####**

Dabei werden die Elemente mit den folgenden Symbolen dargestellt

* Bäume durch **#**
* Kara durch **@**
* ein Kleeblatt als **.**
* die Pilze als **$**
* ein auf einem Kleeblatt stehender Pilz als **\***
* Kara auf einem Kleeblatt als **+**

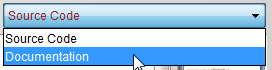
# Sokoban programmieren

Damit wir mit Kara Sokoban spielen können, müssen wir das Verhalten von Kara programmieren. Der Spieler (Sie!) sollten in der Lage sein, Kara mit den Pfeiltasten zu steuern:

## Kara mit Pfeiltasten steuern

Öffnen Sie anschliessend das Szenario ***Kara 26 Sokoban*** aus dem Ordner ***scenarios-chapter-4***.

Nun sehen Sie die gewohnte Kara-Welt mit ein paar zusätzlichen Klassen. Die zusätzlichen Klassen sollen uns nicht stören, wir werden uns weiterhin vor allem mit den Klassen Kara und MyKara beschäftigen.

Öffnen Sie mit einem Doppelklick den Editor für die (graue) Klasse Kara. Diese Klasse beinhaltet die Methoden, die uns beim Programmieren von MyKara zur Verfügung stehen. Am besten stellen Sie oben rechts im Editor von ‚Source Code‘ auf ‚Dokumentation‘. So sehen Sie nur die Kommentare für die Klasse und die Methoden. Das für uns Wichtigste erscheint dann unter dem Titel ‚Method Summary‘.

Kara besitzt zum Teil die gleichen, zum Teil auch neue Methoden. Machen Sie Sich damit vertraut und klicken Sie für ausführlichere Informationen auf die blauen Links.

Suchen Sie nun die Methode, mit welcher wir Informationen über gedrückte Tasten erhalten.

Die entsprechende Methode hat nicht void oder boolean als Rückgabetyp, sondern String. Ein String ist ein Text (z.B. ein Wort oder ein Satz), der in doppelten Anführungszeichen steht. Die folgenden Beispiele sind alles String-Beispiele:

* „Ich bin ein String“
* „Hallo“
* „a“

In unserem Fall ist der zurückerhaltene String der Name der Taste, die zuletzt gedrückt wurde. Diesen String können wir mit folgender Zeile in eine Variable speichern:

String key = getKey();

Schreiben Sie diesen Code in die act()-Methode vom (roten) MyKara!

Strings können mit einer speziellen Methode namens equals() verglichen werden. Um darauf zu reagieren, wenn eine bestimmte Taste gedrückt wurde, müssen wir folgende if-Bedingung hinzufügen:

if (key.equals(„left“))

{

// Taste ‚links‘ wurde gedrueckt -> mache etwas…

}

Aufgabe 26: Kara mit den Pfeiltasten steuern

🡪 Vervollständigen Sie den Code in der act()-Methode so, dass Kara auf alle vier Pfeiltasten reagiert und sich in die entsprechende Richtung bewegt. Benutzen Sie dabei die Methoden, welche Sie in der Dokumentation vom (grauen) Kara finden! Zum Testen drücken Sie auf den Run-Knopf!

## Hindernisse für Kara

Was passiert, wenn Sie in einen Baum oder Pilz hineinfahren?

Aufgabe 27: Kara vor den Bäumen schützen

Fangen Sie den Fehler so ab, dass sich Kara nicht bewegt, wenn er vor einem Baum steht.  
*Zur Erinnerung: Damit Sie den gleichen Code nicht immer wiederholen müssen, empfiehlt es sich eine Methode dafür zu schreiben!*

Aufgabe 28: Kara bewegt Pilze

Schliesslich soll ja Kara die Pilze stossen. Programmieren Sie Kara so, dass er einen Pilz stossen kann. Achten Sie darauf, dass am Schluss keine Fehlermeldungen mehr entstehen.  
*Hinweis: Falls Sie irgendwo steckengeblieben sind, können Sie auf den Knopf ‚Retry Level‘ drücken!*

Aufgabe 29: Arbeit erledigt

Nachdem Kara seine Arbeit erledigt hat, müsste er eine neue Aufgabe erhalten, d.h. der nächste Level müsste geladen werden. Kara hat eine Methode, mit der er überprüfen kann, ob ein Level erledigt ist. Finden Sie heraus, welche Methode es ist und rufen Sie sie an der richtigen Stelle in Ihrem Programm auf.

Anschliessend sollten Sie in den nächsten Level gelangen. Im Moment haben Sie vier Levels, die Sie spielen können. Wie weitere Levels hinzugefügt werden können, erfahren Sie später.

Aufgabe 30: Kara zählt die Schritte

Am unteren Bildschirmrand finden Sie eine Anzeige für die ‚Moves‘, die jedoch noch nicht funktioniert. Zum Setzen dieser Zahl hat Kara auch eine Methode. Mit dem folgenden Befehl kann die Anzahl beispielsweise auf 3 gesetzt werden:

setNumberOfMoves(3);

Erweitern Sie Ihr Programm so, dass es die Anzahl Schritte zählt. (Dafür werden Sie eine Variable ausserhalb der Methode act() brauchen. Dies ist dann eine sogenannte Objektvariable, da sie im ganzen Objekt verfügbar ist.)

Aufgabe 31: Das Spiel richtig spielen

Nun ist unser Sokoban-Spiel richtig spielbar. Das Spiel besitzt noch ein Hauptmenu, welches erst wie folgt freigeschaltet werden muss:

In der Klasse MyKara finden Sie eine Konstante ‚DEVELOPER\_MODE‘. Wenn Sie diese Konstante auf false setzen, wird bei einem erneuten ‚Compile‘ direkt das Hauptmenu gestartet.

Im Hauptmenu können Sie Level Passwörter eingeben. So können Sie Ihr Spiel in dem Level fortsetzen, wo Sie das letzte Mal verblieben sind.

Aufgabe 32: Eigene Levels erstellen

Die Levels werden aus der Datei ‚LEVELS.txt‘ gelesen. Suchen Sie diese Datei und öffnen Sie sie in einem beliebigen Texteditor. Versuchen Sie einen zusätzlichen Level zu erstellen.

*Tipp 1: Damit das Erstellen von Levels etwas einfacher geht, können Sie es in Greenfoot machen. Dafür stellen Sie am besten wieder den ‚DEVELOPER\_MODE‘ auf true. Nun können Sie auf der grünen Wiese Kara, die Bäume, die Pilze und die Kleeblätter platzieren. Wenn Sie fertig sind, drücken Sie auf einer leeren Stelle in der Welt mit der rechten Maustaste. Wählen Sie den Befehl ‚*generateASCIILevel()*‘. Nun sollte der gezeichnete Level in der Konsole im ASCII-Format ausgegeben werden. Dies können Sie nun direkt in die Level-Datei hineinkopieren.*

*Tipp 2: Die Datei mit den Levels können natürlich auch untereinander ausgetauscht werden. Damit Sie mehrere Level-Dateien haben können, kann in der Klasse* MyKara *die Variabel ‚LEVEL\_FILE‘ geändert werden.*

*Tipp 3: Für zusätzliche Levels können Sie Sich auf folgenden Internetseiten inspirieren lassen:*

* <http://users.bentonrea.com/~sasquatch/sokoban/> *(unter Microban-Levels schauen, die anderen sind sehr gross!)*
* <http://www.sourcecode.se/sokoban/levels.php> *(wenn man auf das ‚T‘ klickt, erhalten Sie die Levels in dem gewünschten ASCII-Text-Format)*

Aufgabe 33 (für Kreative): Eigene Bilder

Die Bilder für Kara, Pilze, Hintergrund und co. können ausgewechselt werden. Um ein anderes Bild auszuwählen können Sie auf die Klassen Rechtsklicken und ‚set Image…‘ drücken. Die Bilder für die Felder sind 28 x 28 Pixel gross.

Aufgabe 34 (für Schnelle): Die Highscore

Wenn Sie in der Klasse MyKara die Konstante ‚HIGHSCORE\_ENABLED‘ auf true stellen, dann können die Highscores geführt werden. Dazu wird im Hauptmenu ein zusätzlicher Knopf eingeschaltet.

Es gibt immer drei Highscores pro Level. Kara besitzt Methoden, um die Highscore zu ändern.

Versuchen Sie Ihr Programm so zu erweitern, dass geprüft wird, ob man eine Highscore erreicht hat. Falls ja, soll die Highscore hinzugefügt werden.

Weitere Ideen: Sound

Sie können mit Greenfoot auch Sound zu Ihrem Spiel hinzufügen. Informationen dazu finden Sie in der Greenfoot Dokumentation unter der Klasse GreenfootSound.

Ich hoffe, Sie hatten Spass mit dem Käfer.

Falls Sie das Szenario mit anderen teilen möchten, können Sie es unter dem Menu Scenario | Export … entweder auf die Greenfoot Gallery (online) stellen oder es als ausführbares Java-Programm exportieren.   
*Achtung: Bitte den Source Code nicht in der Greenfoot Gallery veröffentlichen!*

1. Quelle: <http://de.wikipedia.org/wiki/Sokoban>, abgerufen am 14.03.2011. [↑](#footnote-ref-1)
2. ASCII ist ein Zeichensatz mit lateinischem Alphabet, den zehn arabischen Zahlen sowie einigen Satz- und Steuerzeichen [↑](#footnote-ref-2)