

# Chapter 4: Kara Sokoban Game

## Sokoban<sup>1</sup>

Sokoban (倉庫番, Japanese “Warehouse Manager”) is a computer game developed by Hiroyuki Imabayashi in 1982 under the name “Thinking Rabbit” and was published on different computer systems.

## Gameplay

In a simple game the principle task is to have a character move successively all objects - usually they are boxes - to the designated target areas. The boxes can only be moved and cannot be pulled by the character, nor can several boxes be simultaneously pushed. Besides mastering a level it is a continuing challenge to minimize the necessary steps.

In our Kara Sokoban the mushrooms must be pushed on the target fields (leaves).



## Description of Levels

Many Sokoban games use a simple ASCII format<sup>2</sup> to describe the levels. To create our own levels any text editor can be used. The example with Kara would be as follows:



```
####
#  . #
#   ###
# @   #
#   $  #
#   ###
####
```

The different elements are represented by the following symbols

- Tree by #
- Kara by @
- Leaf by .
- Mushroom by \$
- A mushroom on a leaf by \*
- Kara on a leaf by +

<sup>1</sup> Source: <http://de.wikipedia.org/wiki/Sokoban>, 14.03.2011.

<sup>2</sup> ASCII is a character set containing the latin alphabet, arabic numbers and some other signs.


## Programming Sokoban

So that we can play with Kara Sokoban, we have to program the behavior of Kara. The player (you!) should be able to control Kara with the arrow keys:

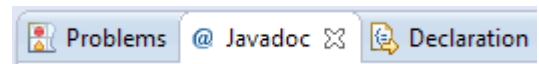
### Kara Control with Arrow Keys

Open the *scenario26...* from the project *scenarios-chapter-4*.

Our class is now called `MyKaraSokoban` and inherits not from the `Kara` class directly but from `KaraSokoban`. This can be seen by the statement `MyKaraSokoban extends KaraSokoban`. Through inheritance we can access all methods from `KaraSokoban`.

Hold down the **Ctrl-Key** and click on `KaraSokoban`. This follows the name like a link and opens the corresponding class. To find out where this class is actually located you can click on the yellow arrows in the package explorer (  ).

Familiarize yourself with the new methods by looking at the



(blue) comments. To view the comments nicely formatted you can use the Javadoc window below.

Now find the method by which we receive information about pressed keys.

The appropriate method has a return type that is not **void** but **String**. A **String** is a text (e.g. a word or phrase) that is enclosed in double quotation marks. The following examples are all **String** examples:

- "I am a string"
- "Hello"
- "a"

In our case, the resulting **String** is the name of the key that was pressed last is returned back to the calling program. We can save this **String** with the following line into a variable:

```
String key = getKey();
```

Write this code in the **act ()** method from the (red) `MyKara`!

Strings can be (compared) by a special method called **equals ()**. In order to react when a certain button is pressed, we must add the following if statement:

```
if (key.equals("left")) {  
    // Left key was pressed -> do something  
}
```

### **TASK 26: CONTROLLING KARA WITH ARROW KEYS**

→ Complete the code in the **act ()** method so that Kara responds to all four arrow keys and moves in that direction. Use methods found in the documentation of the (gray) `Kara`! To test, press the **Run button**!

## Obstacles for Kara

What happens when you drive into a tree or a mushroom?

### **TASK 27: PROTECT KARA FROM THE TREES**

Fix the error so that Kara does not move when he stands before a tree.

*Remember: You should not repeat code. It is recommended that you write a method that you can call more than once!*

### **TASK 28: KARA PUSHES MUSHROOMS**

At the end, Kara is obviously supposed to push the mushrooms. Program Kara so that he can move a mushroom. Make shure that there are no error messages showing any more.

*Note: If you are stuck somewhere, you can press the button 'Retry Level'*

### **TASK 29: WORK DONE**

After Kara has done his work, he would get a new task, i.e. the next level should be loaded. Kara has a way with which he can verify whether a level is completed. Find out which method it is and call it in the right place in your program.

Then you should be taken to the next level. Right now you have four levels that you can play. More levels will be added later.

### **TASK 30: KARA COUNTS THE STEPS**

At the bottom of the screen you will find an indicator for moves, however, it does not work. To set this number Kara also has a method. The following command will set the number to 3, for example:

```
setNumberOfMoves(3);
```

Extend your program so that it counts the number of steps. For this you will need to declare a variable outside the **act()** method. This is called an instance variable, because it is available for the entire class.

### **TASK 31: PLAY THE GAME**

Now our Sokoban game is playable. The game already has a main menu which can be unlocked as follows:

In the class **MyKaraSokoban** you will find a call to `,setDeveloperMode()` ' inside the `main()` method. If you set the value inside the braces to **false** the main menu is enabled.

In the main menu you can enter the level password. Thus, you can continue your game at the level where you have left the last time.

**TASK 32: DESIGN YOUR OWN LEVELS**

The levels are read from the file **Levels.txt**. Locate the file in the scenario and open it in a text editor. Try to create an additional level.

*Tip 1: You can create levels a bit easier with a little trick. Make sure the developer mode is set to true. Now you can place the actors in the world.*

*When you are finished, press on a blank spot in the world with the right mouse button. Select the command 'Print World Setup To Console'. This will write a level in ASCII format on the console. This you can copy directly into the level file.*

*Tip 2: The file with the levels can also be exchanged with others. In order that you can have several different level-files, the level filename may be modified inside the main method.*

*Tipp 3: For additional levels you can take inspiration from the following websites:*

- <http://users.bentonrea.com/~sasquatch/sokoban/> (look at Microban levels; the others are very large!)
- <http://www.sourcecode.se/sokoban/levels.php> (if you click on the 'T', you get the desired level in the ASCII text format)

**TASK 33 (FOR THE CREATIVE): MY PICTURES**

The pictures of Kara, mushroom, leaf, tree and background can be replaced. To choose a different image replace the image in the "images" subfolder. The new image has to have the same name as the original. The images are 28 x 24 pixels in size.

**TASK 34 (FOR THE FAST): THE HIGHSCORE**

If you set the Method call `setHighscoreEnabled()` to **true** then the high scores are displayed. In the main menu, an additional button is turned on.

There are always three high scores for each level. Kara has methods to change the highscore.

Try expanding your program so that it checks whether a new high score is achieved. If so, the high score should be added.

---

I hope you had fun learning to program with the ladybug Kara.

If you would like to share your scenario with others, you can do so with Eclipse by choosing the menu *File / Export ... / Java – Runnable Jar File*. Then you will need to select the correct *Launch Configuration*. It is probably the last launch configuration (if you have just launched the game before). The resulting jar file can be run on any system that has Java installed by double-clicking on the jar file.