Chapter 4: Kara Sokoban Game

Sokoban[1]

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 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[2] to describe the levels. To create our own levels any text editor can be used. The example with Kara would be a 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 **+**

[1] Source: http://de.wikipedia.org/wiki/Sokoban, 14.03.2011.
[2] ASCII is a character set containing the latin alphabet, arabic numbers and some other signs.

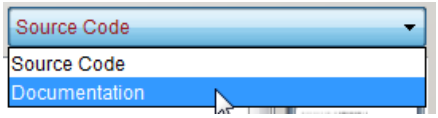
Programming Sokoban

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 scenario ***Kara 26 Sokoban*** from the folder ***scenarios-chapter-4***.

Now you see the usual Kara-world with a few extra classes. The extra classes should not disturb us, we will continue to deal primarily with the classes **`Kara`** and **`MyKara`**.



Double-click to open the editor for the (gray) **`KaraSokoban`** class. This class contains the methods that are available to **`MyKara`**. The methods of **`Kara`** are also inherited. In the top right corner in the editor change **'source code' to 'documentation'**. In the documentation you only see the comments for the class and methods. The most important thing for us will appear under the title *Method Summary*.

Kara has some of the same, and some new methods. Make yourself familiar with them and click for more detailed information on the blue links.

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.
Reminder: 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 really playable. The game already has a main menu, which must be unlocked as follows:

In the **`MyKara`** class you will find a constant ‚**`DEVELOPER_MODE`**‘. If you set this constant to **`false`**, and you Compile, the main menu starts. In the main menu you can enter the level password. So 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 in Greenfoot a bit easier with a little trick. Make sure the **`DEVELOPER_MODE`** *is set to true. Now you can place the actors in the Greenfoot world.*

When you are finished, press on a blank spot in the world with the right mouse button. Select the command **`generateASCIILevel()`***. 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 levels-file, the variable LEVEL_FILE can be modified in the MyKara class.

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 (CREATIVE): MY PICTURES

The pictures of Kara, mushroom, leaf, tree and background can be replaced. To select a different image, you can right-click on the classes and choose *set image*... . Most of the images are 28 x 28 pixels in size.

TASK 34 (FOR THE FAST): THE HIGHSCORE

In the class MyKara if the constant **`HIGHSCORE_ENABLED`** is set 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 checkes whether it has achieved a high score. If so, the high score should be added.

FURTHER IDEAS: SOUND

You could add sound to your Kara scenario with Greenfoot. Information on Sound can be found in the Greenfoot documentation under Class **`GreenfootSound`**.

I hope you had fun with the ladybug Kara.

If you would like to share your scenario with others, select the menu Scenario and choose Export. The export can either be made to the Greenfoot Gallery (online) or as an executable Java Program.
Note: Please do not add the source code if you publish it to the Greenfoot Gallery.