

Chapter 1: First Steps¹


Start Eclipse.

Import the Eclipse project **scenarios-chapter-1**.

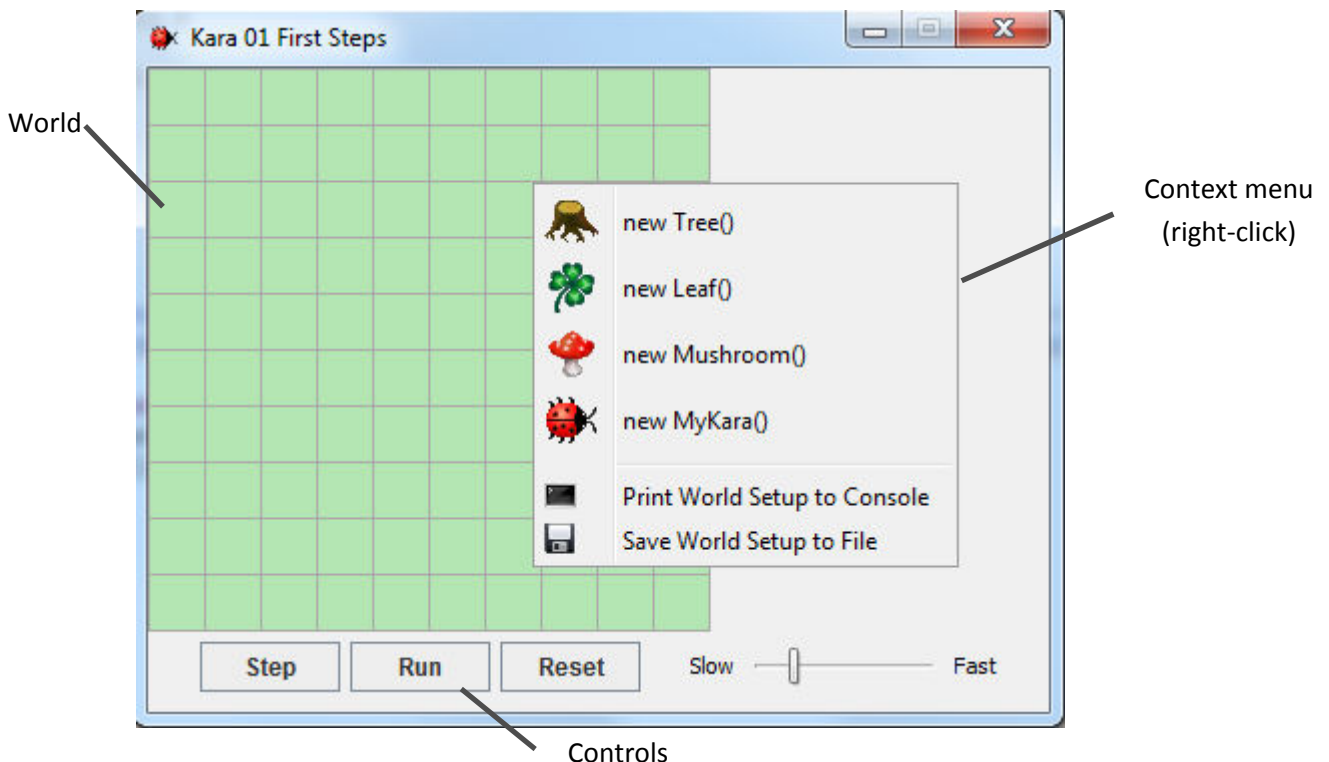
Go to the **src/scenario01...** subfolder. The scenario includes a file **MyKara.java**. Files ending in .java Include program code.

Now open the file **MyKara**.

In this file there are two blocks, called methods, with names **act()** and **main (...)** and a few blue comments. The main method is needed so that the computer knows where the program begins. For now, we'll only need the act() method.

To start, we press the "Run" button in Eclipse: 

A window opens and the scenario similar to the following figure should appear:



- The *world*: The largest area is called world. This is the area when the program runs, we can see what happens. In the Kara scenario it is a green meadow with gridlines.
- The *context menu*: With the context menu we can set new actors into the world.
- The *controls*: The buttons *Act*, *Run*, and *Reset* and the *speed slider* at the bottom serve the controlling of the program. We will come to that later.

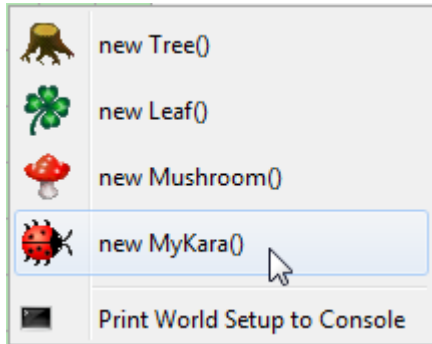
¹ The worksheets in this chapter are based on the book "Introduction to Java using Greenfoot" by Michael Kölling (2010). Ideas and concepts of Kara were developed by Jürg Nievergelt, Werner Hartmann, Raimond Reichert et al. <http://www.swisseduc.ch/informatik/karatojava/>, February 2011.

Classes and Objects

For our projects we use the **Java** programming language. Java is a so-called object-oriented language. For object-oriented programming, the concepts of classes and objects are of fundamental importance.

Consider the **class MyKara**. MyKara is the class for the general concept of a beetle - so to speak, it is like a blueprint from which we can create individual beetles. The produced beetles are called **objects** (or instances).

With the mouse we create new objects as follows: Right click on the green world and choose menu item *new MyKara ()*. Objects can also be moved around using the mouse.



TASK 1:

Create an object of **Kara**. Create several objects of the leaves.

Interacting with Objects

To interact with objects in the world, we click with the right mouse button to bring up the **object menu**. The object menu of Kara shows us what Kara can do. These operations are called **methods** in Java.

TASK 2:

a) What does the **move ()** method do?

Kara moves ahead one field.

b) Place two Karas in your world and make sure that they face each other. Which method do you need?

turnLeft() or turnRight()

c) Try the other methods. There are two types of methods. What are those types and what do they do?

Methods with void: Perform an action.

Methods with boolean: Open a window containing the methods result.

Return Types

The word in front of a method name is also referred to as the **return type**. It tells us what the method returns, if we call it. The word **void** in this case means “nothing” - they return nothing, just perform an action.

If instead of **void** anything else is stated, we know that our method returns a result and also what type of result. The **boolean** type has two possible values: **true** or **false**. Then there are a number of other types which we will discuss later.

TASK 3:

- a) Right-click on the Kara object and call the **onLeaf ()** method. Does it always return **false**? Or are there situations where it returns **true**?

If Kara is on a leaf this method returns true.

- b) Add a tree to the world. What method can you call to check if Kara is facing a tree?

treeFront()

- c) What happens if you call Kara's **move ()** method when Kara is facing a tree?



Kara complains that he can't move.

Object State

If we *right-click* on a Kara-object, then select *inspect*, we can determine the state of the object.

TASK 4:

What are the values of the coordinates and rotation in the following situations?

 <p>a)</p>	 <p>b)</p>
<p>x: 0 y: 0 rotation: 0</p>	<p>x: 1 y: 2 rotation: 180</p>

Note: The first field in the top left corner has the coordinates (0, 0) and NOT (1, 1)!

Running Programs

So far we have only interacted with the objects using mouse-clicks. But there is another possibility, namely by writing programs.

TASK 5:

a) Place a **MyKara** object into your world. What does the method `act()` do?

move, turn right, move

b) What happens if you press the **Step button**?

The same as before, the `act()` Method is called

c) Click on the **Run button**. What is happening? Try using the slider.

`act()` is called again and again until the pause button is clicked

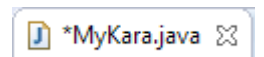
Source Code

We have just used the `act ()` method. Now let's see where the behavior of this method is programmed. For this we need to open the **source code** in the Eclipse editor:

Open the class **MyKara** with a double click.

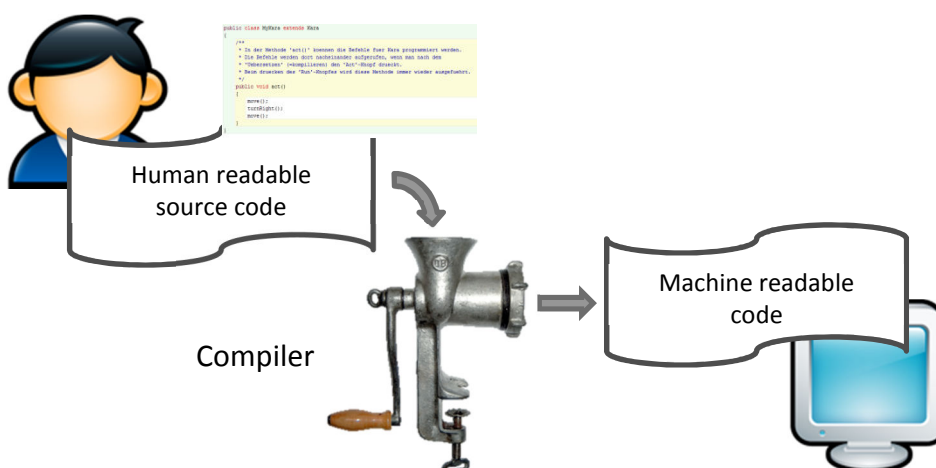
The source code is written in Java and contains all the details about a class and its objects. For the moment we are only interested in the part where the method `act ()` is defined. Here, the three commands: `move ()`, `turnRight ()` and `move ()` are called. You can now modify and extend these commands.

When you have made some changes you will notice that the class is shown with a little star in the tab. This is an indication that the class has been modified and has not been saved yet. Save the changes (ctrl-s).



Each time you save the class Eclipse automatically translates it (or technical terminology: compiles). The compiling is a process of translation: The source code of the class is translated into machine code that can be executed by the computer.

The Translation Process (Compiling)



TASK 6:

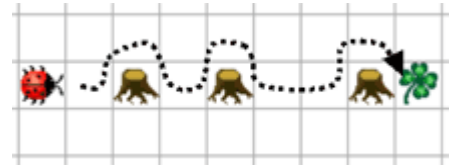
Go to the **Scenario 06** in *src/scenario06_putting_leaf*. Open in **Scenario 06** the file *MyKara.java*.

Change the content of the **act ()** method so that Kara makes a step first, then puts a leaf down and takes a step again. (At the beginning of the class you will find a comment describing all the actions that Kara can perform.)

Note: After each command we must include a semicolon!

TASK 7:

Open the **Scenario 07**. If you start *MyKara* you should automatically see the world as shown to the right:



Note: The world is loaded from the text file **WorldSetup.txt**. This file-name is defined in the *main()*-Method inside *MyKara*.

Now write a program that takes Kara to the leaf along the dotted line on the image above. He will have to run around the trees. Arriving at the leaf, he should pick it up.

Kara gets new Methods**TASK 8:**

If you have solved task 7 correctly your program should contain **three equal parts**, namely for walking around each tree. We can expand our program for clarity by introducing a new method. Below the **act ()** method we will create a new method:

```
public void goAroundTree() {  
  
}
```

Between the curly brackets of the method, write the commands to make it go around a tree.

Now use the new method **goAroundTree ()** inside the **act ()** method for each of the three trees.