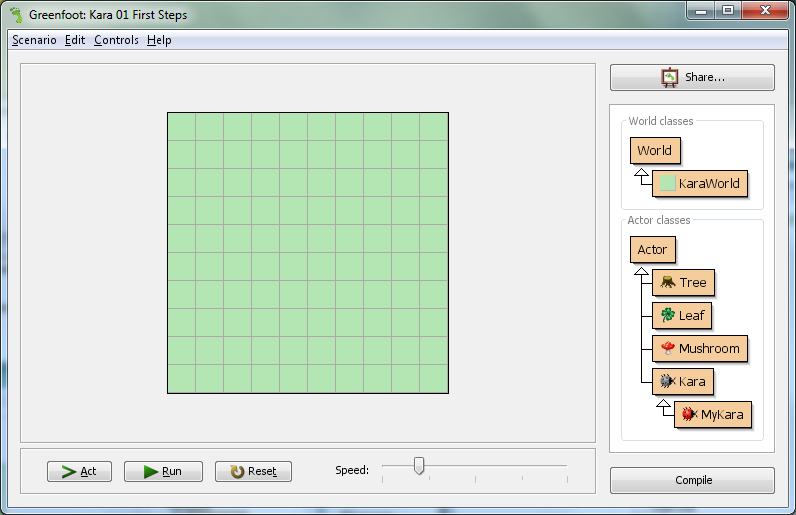
Chapter 1: Getting Started with Greenfoot[[1]](#footnote-1)

Start Greenfoot.

(If you open Greenfoot for the first time, a dialog box that asks what you want. Click on *choose scenario*.)

D:\Programmieren\greenfoot-szenarien\_Praktikum Wettingen\Kara 00 - Erste Schritte\images\kara_red.pngOpen the scenario ***Kara 01 First Steps*** from the folder ***scenarios-chapter-1***.

It now appears in the main Greenfoot window similar to the following:



Class   
diagram

Greenfoot controls

World

The Greenfoot window consists of three main areas and a few additional buttons. These three main areas are:

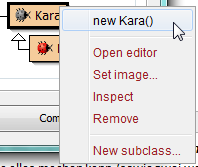
* The *world*: The largest area is called world. This is the area when the program runs, we can see what happens. In the Kara scenario it is a green meadow with gridlines.
* The *class diagram*: The area to the right contains the classes that are available to us. First, we will mainly need the beetle Kara.
* *Greenfoot 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.

# 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 of Kara**. Kara 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).

In Greenfoot, we create new objects as follows: Right click on the class of Kara |menu item *new Kara ()*. Then move the object anywhere in the world and click where you wish to place it.





***Task 1:***

Create an object of (the **gray**!) **Kara**. Create several objects of the leaves.  
*Tip: To create multiple objects of the same class, press the Shift[[2]](#footnote-2) key before you click.*

# 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:

1. What does the **move()** method do?

Kara moves ahead one field.

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

turnLeft() or turnRight()

1. 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:

1. 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.

1. Add an additional tree to the world. What method can you call to check if Kara is facing a tree?

treeFront()

1. 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?

|  |  |
| --- | --- |
| a) | b) |
| x: 0  y: 0  rotation: 0 | x: 1  y: 2  rotation: 180 |

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

# Running Programs in Greenfoot

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



**red** (MyKara)

From now on we will work with the **red Kara** (**MyKara** class)!  
In this class we will write our programs.

Task 5:

1. Place a **MyKara** object into your world. What new method do you have available?

act()

1. What does this method do?

Move, turn right, move

1. What happens if you press the **Act button** in the Greenfoot controls?

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

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

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

1. What methods can be found when you *right-click on MyKara* and then *inherited from Kara*? Can you use them?

All methods from Kara. They can directly be used by MyKara

# 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 editor: *Right-click on the class MyKara* then choose *Open editor* (or just use 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 lowest 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 finished making changes, close the editor window. You will notice that the class is shown with gray stripes. This is an indication that the class has been modified and now needs to be **compiled**. Compiling is a process of translation: the source code of the class is translated into machine code that can run on the computer.

The classes in Greenfoot are translated using the button **Compile** (bottom right of the main window). Once the classes are translated, the stripes disappear and we can create objects again.

**The Translation Process (Compiling)**



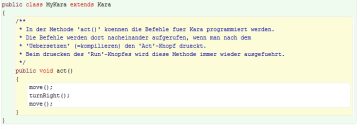
Human readable

source code

Machine readable   
code

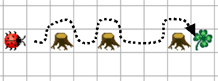


Compiler



Task 6:

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:

Create the world shown on the right:

*Tip: So that you don’t have to recreate the world after each compile you can save the world by right-clicking on the World and calling saveWorldSetupToFile(). Then choose a new filename.  
In order for your world to actually be loaded you must change the constant* ***WORLD\_SETUP\_FILE*** *in KaraWorld:*

*public static final String WORLD\_SETUP\_FILE = "****[MyName]****.txt";*

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() {**

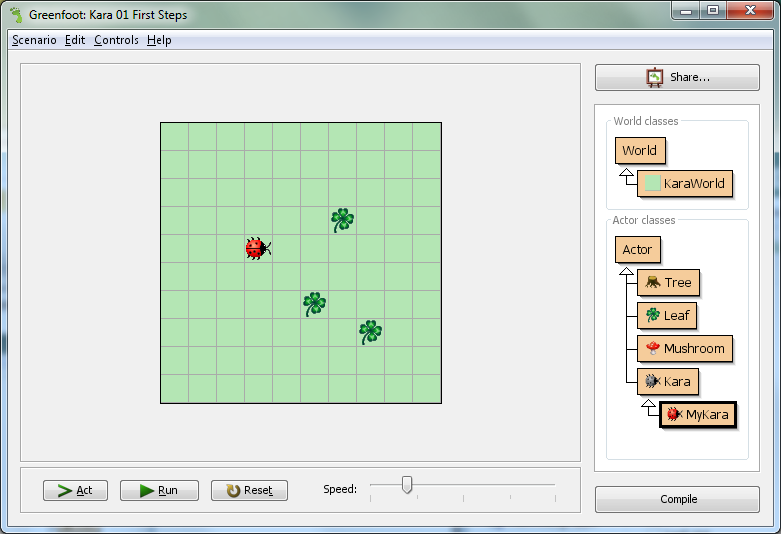
**}**

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.

# Learning Summary of Chapter 1

1. Name the following elements in the image:  
   *World, Class diagram, compile button, a class, an object*



1. Return type
2. What is special about a method with a **void** return type?

Performs an action but does not return a result.

1. What is special about a method with a **boolean** return type?

Returns a result of the type boolean.

1. What values can **boolean** have?

true and false

1. What is the difference between **move()** and **act()** in our Kara scenario?

move() moves Kara one step forward. In act() we can call any method we want.

1. What is compiling? Why is it necessary?

Compiling is the translation of source code into machine code so that the computer can understand it.

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. [↑](#footnote-ref-1)
2. If the mouse, for example For Mac OS computers, has only one button: Simultaneously press the Ctrl key together with the mouse button! [↑](#footnote-ref-2)