Kapitel 1: Die ersten Schritte[[1]](#footnote-1)

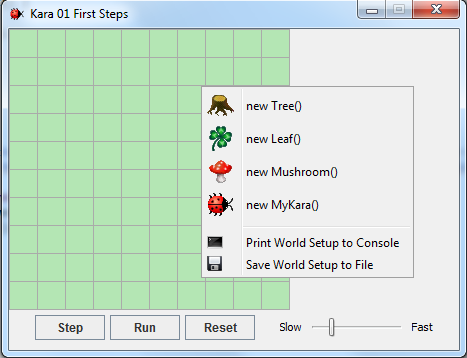
D:\Programmieren\greenfoot-szenarien\_Praktikum Wettingen\Kara 00 - Erste Schritte\images\kara_red.pngStarten Sie Eclipse.

Importieren Sie das Eclipse-Projekt ***scenarios-chapter-1***.

Gehen Sie in den Unterordner ***src/scenario01***. Das Szenario enthält eine Datei ***MyKara.java***.Dateien mit der Endung *.java* beinhalten Programmcode.   
Öffnen Sie nun also die Datei ***MyKara***.

In dieser Datei hat es zwei Blöcke, genannt Methoden, mit Namen **act()** und **main(…)** und ein paar **blaue Kommentare** zur Beschreibung. Die main-Methode braucht es, damit der Computer weiss, wo das Programm beginnt. Uns wird vorläufig aber vor allem die **act()**-Methode beschäftigen.

Zum Starten drücken wir in Eclipse auf den „Run“-Knopf: 

Es erscheint nun das Szenario ähnlich folgender Abbildung:

Kontextmenu   
(Rechtsklick)

Steuerung

Welt

* Die *Welt*: Der grösste Bereich wird Welt genannt. Dies ist der Bereich, in dem das Programm ausgeführt wird und in dem wir verfolgen können, was passiert. Im Kara-Szenario ist die Welt eine grüne Wiese mit Gitterlinien.
* Das *Kontextmenu*: Mit dem Kontextmenu können wir Akteure in die Welt setzen.
* Die *Steuerung*: Die Buttons *Step*, *Run*, und *Reset* und der Schieberegler für die Geschwindigkeit ganz unten dienen der Programmsteuerung. Darauf werden wir später zu sprechen kommen.

# Klassen und Objekte

Für unsere Projekte verwenden wir die Programmiersprache ***Java***. Java ist eine sogenannte *objektorientierte* Sprache. Für objektorientierte Programmierung sind die Konzepte von Klassen und Objekten von grundlegender Bedeutung.

Betrachten wir einmal die ***Klasse*** **MyKara**. Die ***Klasse*** **MyKara** steht für das allgemeine Konzept eines Käfers – sie ist sozusagen wie ein Bauplan, aus dem wir einzelne Käfer erstellen können. Die erzeugten Käfer werden als ***Objekte*** (oder auch Instanzen) bezeichnet.

Mit der Maus erstellen wir neue Objekte wie folgt: Rechte Maustaste[[2]](#footnote-2) auf die Wiese und Menüpunkt *new MyKara()* auswählen. Die Objekte können auch mit der Maus verschoben werden.





***Aufgabe 1:***

Erzeugen Sie ein Objekt von Kara. Erzeugen Sie mehrere Objekte von den Kleeblättern (Leaf).

# Mit Objekten interagieren

Zum Interagieren mit Objekten in der Welt klicken wir sie mit der rechten Maustaste an, um das ***Objektmenu*** aufzurufen. Das Objektmenu von **Kara** zeigt uns, was dieser spezielle Kara alles machen kann. Diese Operationen werden in Java ***Methoden*** genannt.

Aufgabe 2:

1. Was bewirkt die Methode **move()**?

Kara geht ein Feld nach vorne.

1. Platzieren Sie zwei Karas in Ihrer Welt und sorgen Sie dafür, dass sie sich anschauen. Welche Methode brauchen Sie dazu?

turnLeft() oder turnRight()

1. Testen Sie die weiteren Methoden. In welche zwei Arten können die Methoden unterteilt werden?

Methoden mit void: führen nur eine Aktion aus.

Methoden mit boolean: Öffnen ein Fenster mit dem Methodenergebnis.

# Rückgabetypen

Das Wort vor einer Methodenbezeichnung wird auch als *Rückgabetyp* bezeichnet. Es teilt uns mit, was die Methode zurückliefert, wenn wir sie aufrufen. Das Wort **void** bedeutet in diesem Falle „nichts“ – sie liefert also nichts zurück und führt nur eine Aktion aus.

Wird anstelle von **void** irgendetwas anderes angegeben, wissen wir, dass die Methode uns ein Resultat zurückliefert und auch von welchem Typ dieses Resultat ist. Der Typ **boolean** kennt zwei mögliche Werte: **true** (wahr) oder **false** (falsch). Dann gibt es noch eine Reihe weiterer Typen, auf die wir später eingehen werden.

Aufgabe 3:

1. Rufen Sie die Methode **onLeaf()** für einen Kara auf. Liefert Sie immer **false** zurück? Oder gibt es auch Situationen, in denen sie **true** zurückliefert?

Wenn Kara auf einem Blatt steht, liefert die Methode **true** zurück.

1. Setzen Sie zusätzlich einen **Baum** (Tree) in die Welt. Mit welcher Methode können Sie überprüfen, ob Kara vor einem Baum steht?

treeFront()

1. Was passiert, wenn Sie Kara mittels der Methode move() in einen Baum fahren lassen?

Kara reklamiert dass er sich nicht bewegen kann.

# Objektzustand

Mittels *Rechtsklick | Inspect (Inspizieren)* auf einem Kara-Objekt können wir den Zustand des Objektes ermitteln.

Aufgabe 4:

Welchen Zustandswerte hat Kara in den folgenden Situationen (beachten Sie nur die Koordinaten und die Drehung):

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

Beachte: Das erste Feld oben links hat die Koordinaten (0, 0) und NICHT (1, 1)

# Ausführen von Programmen

Bis jetzt haben wir immer nur mit Mausklicks mit den Objekten interagiert. Es gibt aber noch eine weitere Möglichkeit, nämlich indem wir Programme schreiben.

Aufgabe 5:

1. Platzieren Sie ein Objekt von **MyKara** in Ihre Welt. Was macht die Methode act()?

Schritt vorwärts, Drehung nach rechts, Schritt vorwärts

1. Was passiert, wenn Sie in der Steuerung (unten links) auf den „**Step-Button**“ drücken?

Das Gleiche wie vorher. Die act()-Methode wird aufgerufen.

1. Klicken Sie auf den „**Run-Button**“. Was geschieht? (Versuchen Sie mal die Geschwindigkeit zu verstellen)

act() wird immer wieder aufgerufen, bis man auf „Pause“ drückt.

# Quelltext

Vorhin haben wir die act()-Methode kennen gelernt. Nun wollen wir uns anschauen, wie das Verhalten dieser Methode programmiert wurde. Dazu müssen wir den ***Quelltext*** im Eclipse-Editor öffnen:

Öffnen Sie die Klasse **MyKara** mit Doppelklick.

Der Quelltext ist in Java geschrieben und enthält alle Details zu einer Klasse und ihren Objekten. Für den Moment interessiert uns nur der Teil, wo die Methode **act()** definiert ist. Hier sollten die drei Befehle: *move(),* *turnRight()* und *move()* untereinander stehen. Sie können nun diese Befehle verändern und erweitern.

Wenn Sie eine Änderung vornehmen werden sie feststellen, dass oben im Tab ein kleiner Stern erscheint. Das bedeutet, dass die Änderung noch nicht gespeichert wurde. Speichern Sie nun die Änderung (ctrl-s).

Bei jedem Speichern wird die Klasse in Eclipse automatisch neu ***übersetzt***(oder fachsprachlich: ***kompiliert***). Das Kompilieren ist ein Übersetzungsprozess: Der Quelltext der Klasse wird in ***Maschinencode*** übersetzt, den der Computer ausführen kann.

**Der Übersetzungsprozess (Kompilieren)**



Quelltext

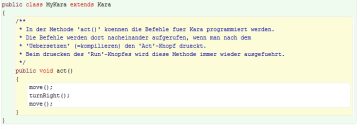
(für Menschen lesbar)

Maschinencode

(für Maschinen lesbar)



Compiler

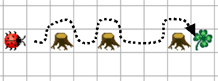


Aufgabe 6:

Gehen Sie in das **Szenario 06** unter *src/scenario06\_putting\_leaf.* Öffnen Sie im **Szenario 06** die Datei   
MyKara.java.

Ändern Sie den Inhalt der act()-Methode so, dass Kara zuerst einen Schritt macht, dann ein Kleeblatt legt und wieder einen Schritt macht. (Am Anfang der Klasse finden Sie als Kommentar alle Aktionen, die Kara durchführen kann.)  
*Beachten Sie*: Nach jedem Befehl muss ein Strichpunkt stehen!

Aufgabe 7:

****Öffnen Sie das **Szenario 07**. Nun sollte bei einem Start von MyKara die Welt automatisch wie rechts abgebildet erscheinen:

*Hinweis: Die Welt wird jeweils aus der Textdatei* ***WorldSetup.txt*** *geladen. In der main()-Methode innerhalb MyKara wird dieser Dateiname angegeben.*

Schreiben Sie jetzt ein Programm, welches Kara auf dem angegebenen Weg zum Kleeblatt führt (siehe Grafik). Er muss dabei um die Bäume herumlaufen. Beim Kleeblatt angekommen, soll er es aufheben.

# Kara erhält neue Methoden

Aufgabe 8:

Wenn Sie Aufgabe 7 korrekt gelöst haben, so wird Ihr Programm wahrscheinlich **drei gleiche Teile** enthalten, nämlich für das Herumgehen um jeden Baum. Dies können wir zur besseren Übersicht noch etwas erweitern, indem wir eine neue Methode einführen. Unterhalb der act()-Methode erstellen wir eine neue Methode:

**public void goAroundTree() {**

**}**

Schreiben Sie zwischen die geschweiften Klammern der Methode die Befehle, die es braucht, um um den Baum zu kommen.

Benutzen Sie nun innerhalb der act()-Methode die Methode *goAroundTree()* für jeden der drei Bäume.

1. Die Arbeitsblätter dieses Kapitels basieren auf dem Buch „Einführung in Java mit Greenfoot“ von Michael Kölling (2010).

   Ideen und Konzepte von Kara wurden entwickelt von Jürg Nievergelt, Werner Hartmann, Raimond Reichert et al., http://www.swisseduc.ch/informatik/karatojava/, abgerufen Februar 2011. [↑](#footnote-ref-1)
2. Wenn die Maus, z.B. bei Mac OS-Rechnern, nur eine Taste hat, gleichzeitig mit der Maustaste die Ctrl-Taste drücken! [↑](#footnote-ref-2)