

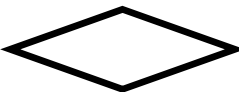


Chapter 2: Program Flow

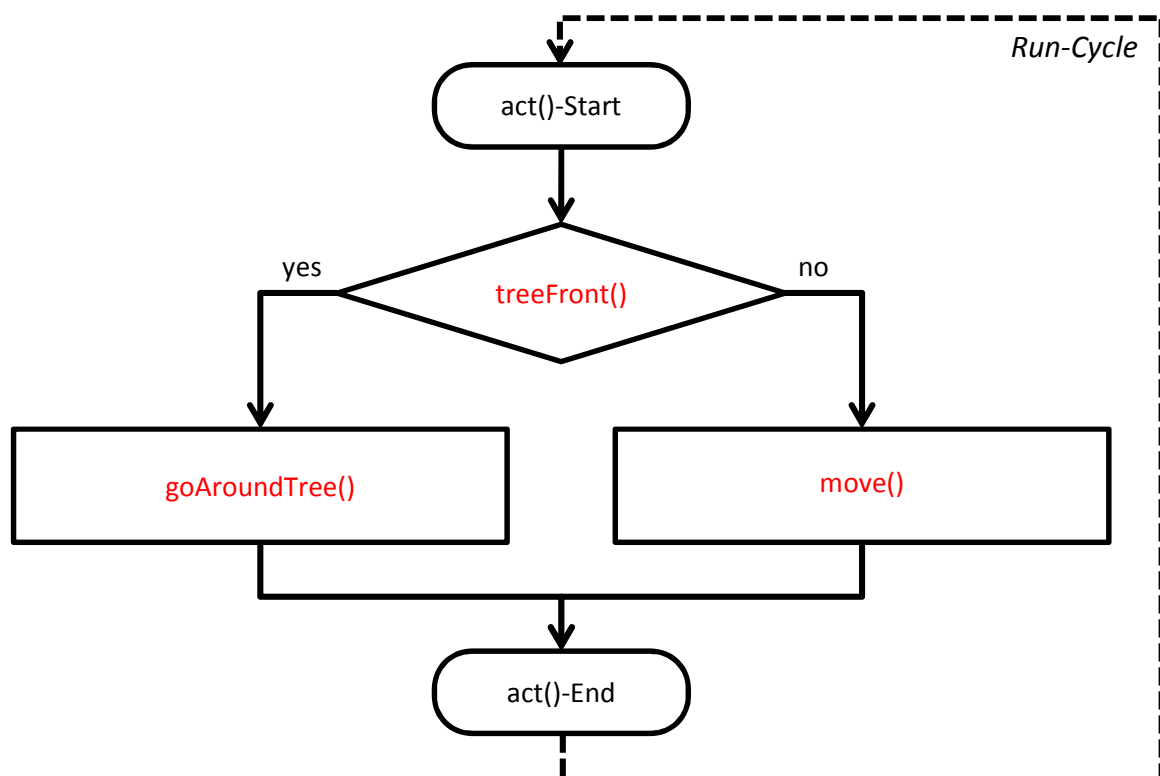
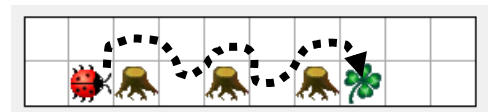
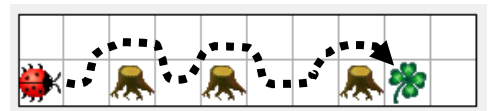
The Flowchart

In order to solve difficult tasks in programming, it is often helpful to outline the program flow in a flowchart. The following symbols are used in flowcharts:

Symbol	Meaning
	Start / Stop
	Activity
	Decision / Condition

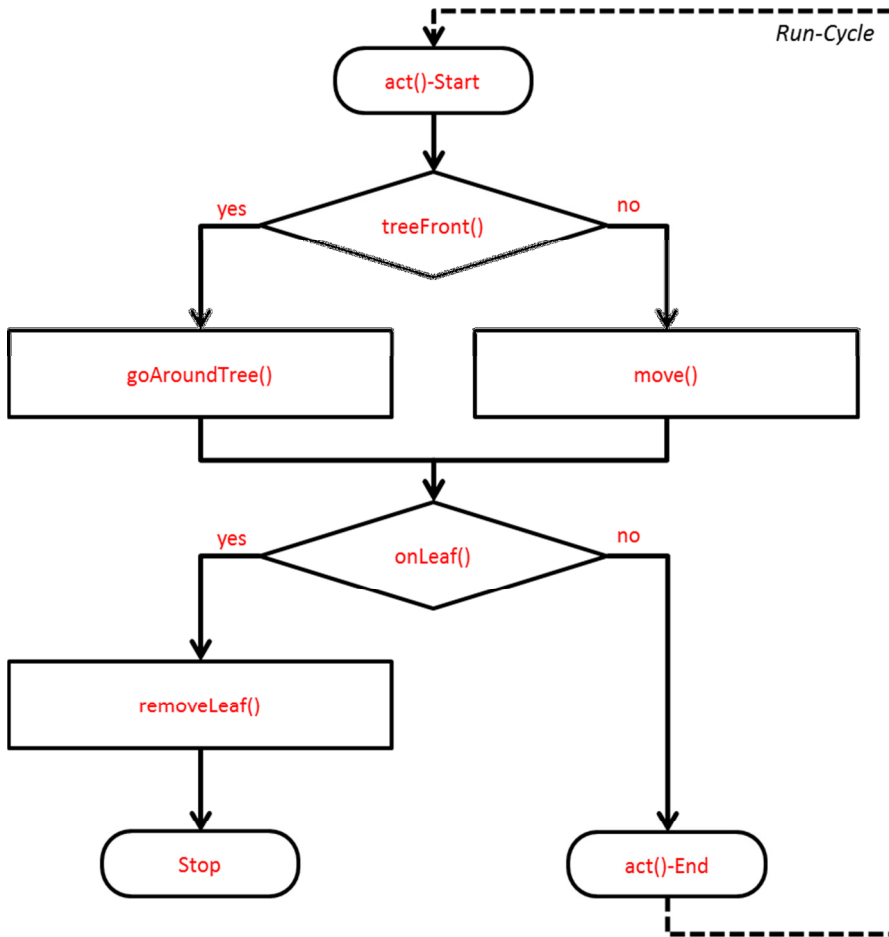
TASK 9: AROUND TREE II

- a) Complete the flow chart so that Kara reached the leaf in all worlds with the following properties:
- The leaf is always right in front of him - he only needs to walk around the trees.
 - There are never two trees standing side by side



- b) Draw an extended diagram so that Kara picks up the leaf at the end if he finds it. On the right you find all the available methods of Kara as a help. Use the method `stop()` when the program is at the end.

Possible solution:



Kara

Actions:

`move()`
`turnLeft()`
`turnRight()`
`putLeaf()`
`removeLeaf()`

Sensors:

`treeFront()`
`treeLeft()`
`treeRight()`
`onLeaf()`
`mushroomFront()`

Conditional Statements in Java¹

```

if (treeFront())
    turnLeft();
else {
    move();
}
  
```

Condition

Block 1, executed if the conditions is **true**.

Block 2, executed if the conditions is **false**.

Note: The `else` part (block 2) may be omitted if it is not needed.

¹ Based on: Thomas Kempe and David Tepassee, Informatik 1 - Softwareentwicklung mit Greenfoot und BlueJ, 1. Ed. (Paderborn: Schöningh, 2010), P. 36.

TASK 10:

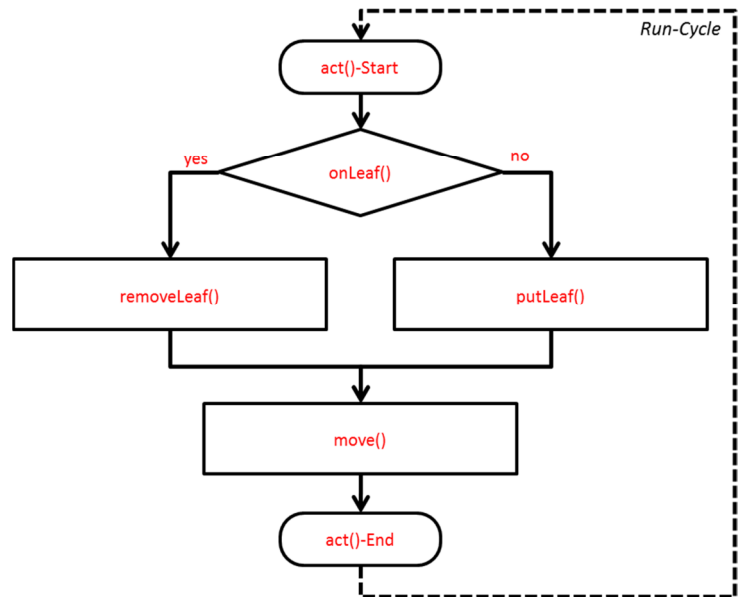
First describe with words what the following code does. Then sketch each of these as a flowchart.

a)

```
if (onLeaf()) {  
    removeLeaf();  
} else {  
    putLeaf();  
}  
move();
```

Removes a leaf if there is one,

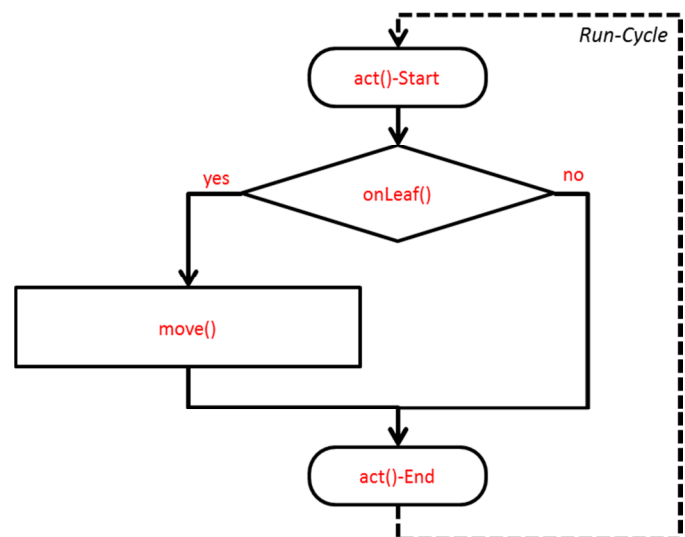
puts a leaf if there is no leaf



b)

```
if (onLeaf()) {  
    move();  
}
```

If Kara is on a leaf he makes one step forward



TASK 11: NESTED CONDITIONS

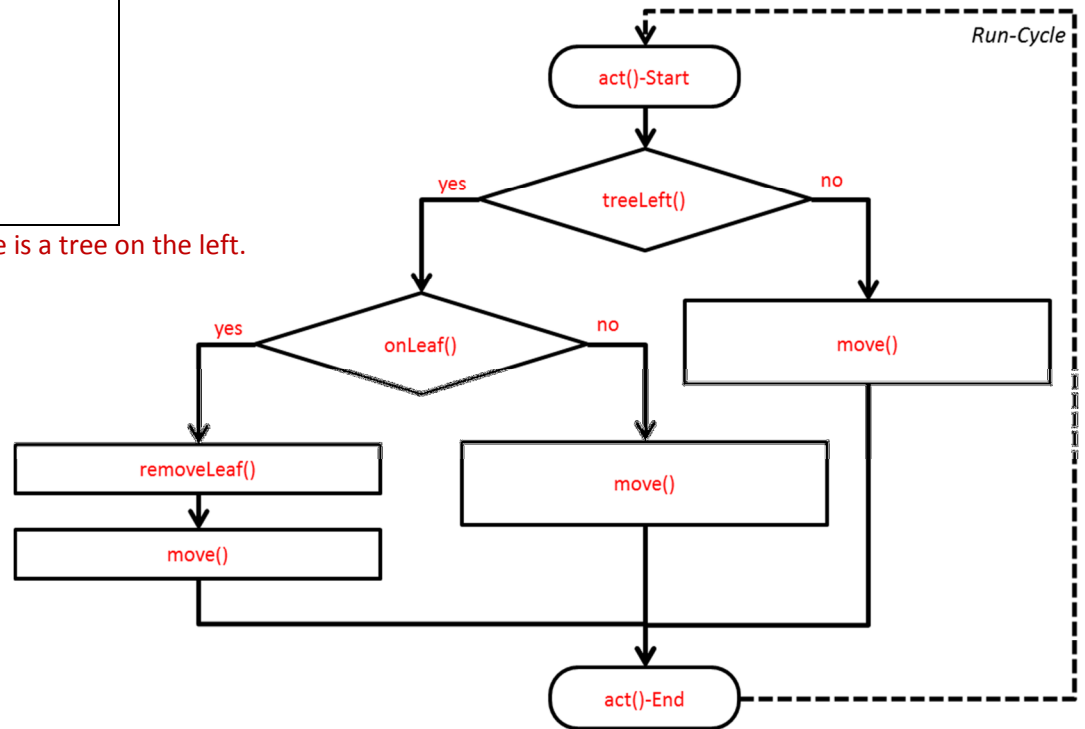
- a) Conditional statements can be nested. Describe what happens when you run the program on the specified KaraWorld. Draw a corresponding flowchart.

```

if (treeLeft()) {
    if (onLeaf()) {
        removeLeaf();
        move();
    } else {
        move();
    }
} else {
    move();
}

```

Removes the leaf if there is a tree on the left.

**Implementing**

Now you can implement the tasks that you have drawn on a computer. You should proceed as follows:

To TASK 9:

- Open **scenario09...** from the project **scenarios-chapter-2**. In this scenario, the method `goAround-Tree()` is already programmed and part of the `act()` method is prepared.
- Now program in the `act()` method what you have drawn in Task 9b as a flowchart.
- In this scenario, you have **several Worlds** (a, b and c).
- Your program should work in any of those worlds **without error messages**.

To TASK 11:

- Open the **scenario11...** in Greenfoot and write the program as outlined in task 11a. Modify the program so that Kara only picks up the leaf if no tree is on his side.

Logical Operations

Our Kara can already do more than just execute simple commands. He will react differently based on a test. It should also be possible for Kara to react simultaneously on two or more tests.

The following table shows the three main logical operators in Java:

Operator	Description	Example	
&&	and	treeFront() && onLeaf()	Is only satisfied (true) if both statements are true, i.e. if Kara is facing a tree <u>and</u> is on a leaf.
	or	treeFront() onLeaf()	Is met (true) if either one <u>or</u> the other <u>or</u> both statements are true.
!	not	!treeFront()	Changes an expression of true to false and vice versa. This statement would be satisfied (true) if Kara is <u>not</u> facing a tree.

An example in Java would look like this:

```
if (treeLeft() && onLeaf()) {
    // Do something ...
}
```

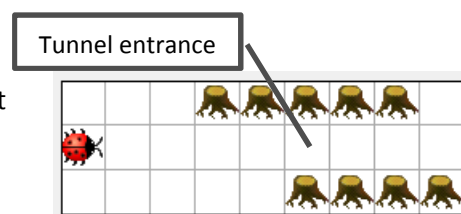
or combined:

```
if (treeLeft() && !treeRight()) {
    // Do something ...
}
```

TASK 12: AFRAID OF TUNNEL

Kara has a little fear of tunnels. He should check on every field whether it is a tunnel entrance (i.e. whether it has trees on both sides). If so, he instantly lets a leaf fall because of the shock.

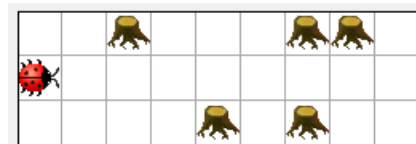
Load the **scenario12...**, write the program and test it with all three worlds.



TASK 13: LEAF AT TREE

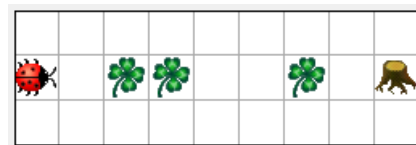
Now let Kara go straight and lay a leaf anywhere where there is a tree on its left or right or on both sides.

Load the **scenario13...** and write the program.



TASK 14: PUT LEAF TRACK

Kara is running straight ahead and lays a leaf anywhere where there is none. When he reaches the tree he will do nothing (even if the Step button is pressed again).



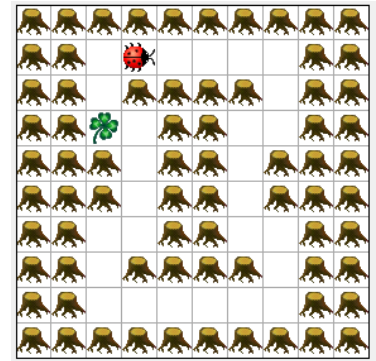
Load the **scenario14...**, write the program and test.

TASK 15: ROUND TRIP

Each field in the tour has exactly two empty neighboring fields. One empty field always lays behind Kara which is the field he came from.

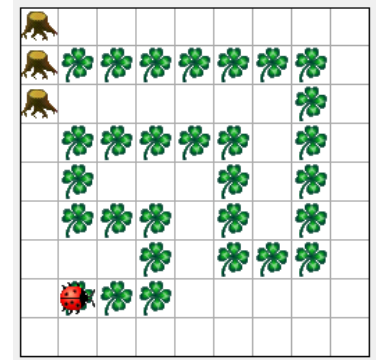
Load the **scenario15...** and write a program for it. Test your program in all three worlds.

Tip: Imagine what must be done each time the Step button is pressed. Draw a flowchart as a help to find the solution.

**ADDITIONAL TASK 16 (DIFFICULT): KARA PLAYS PACMAN**

Kara plays Pacman: He is on the first of a long trail of leafs, ending in front of a tree. He picks up all leafs and stops in front of the trees.

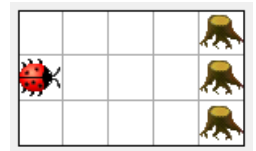
As a better overview write some parts of the program in its own methods.

**Loops**

Kara Kara can now react to rules set by us in different situations. He is not yet capable of **repeating** a specified set of instructions. To execute an instruction block multiple times, loops are used.

As an example, we want to do the following:

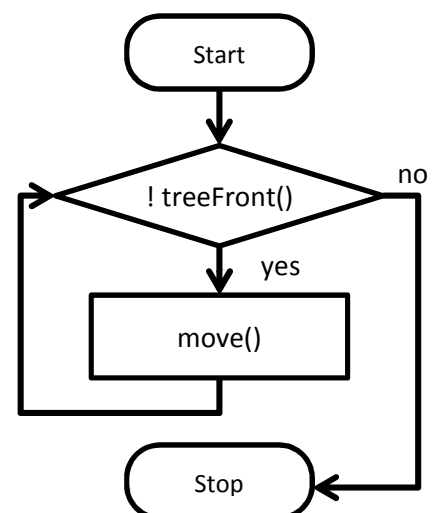
Kara will move forward until he hits a tree.



In the flow chart you can see that `move()` is executed repeatedly, as long as no tree stands in front of Kara.

Die notation in Java:

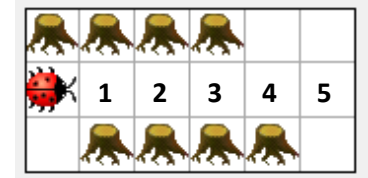
```
while (!treeFront()) {
    move();
}
```



TASK 17:

Given the following situation: Kara stands in front of a tunnel.

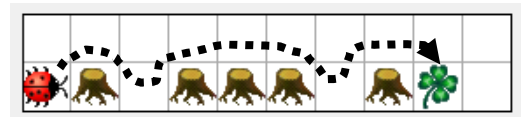
Describe what each of the following loops does, and how many steps Kara takes.



Code	Beschreibung	Anzahl Schritte
<pre>while (treeLeft()) { move(); }</pre>	Move as long as there is no tree on the left.	4
<pre>while (treeRight()) { move(); }</pre>	Move as long as there is no tree on the right.	0
<pre>while (treeLeft() treeRight()) { move(); }</pre>	Move as long as there is a tree either on the right or on the left side.	5
<pre>if (treeLeft()) { move(); } while (treeLeft() && treeRight()) { move(); }</pre>	First, move if there is a tree on the left side. Then move as long as there is a tree on the right and on the left side.	4
<pre>while (!treeFront) { if (treeLeft()) { move(); } }</pre>	<p>As long as there is no tree in front of Kara If there is a tree on the left, make a step</p> <p>Warning: never ending loop!</p>	4

TASK 18: AROUND TREE III

This is a similar exercise as in task 9: Kara is to find a leaf that lies ahead of him. Now, there can be any number of trees in a row.



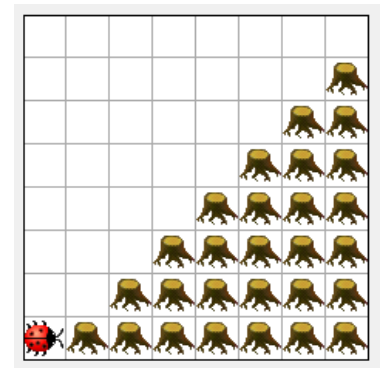
- Load the **scenario18...** and improve the method `goAroundTree()` so that Kara can walk around several trees. Test your program in all available worlds.
- Modify `act()` so that you can just press the Step button once. Kara should then automatically run around the trees until he reached the leaf. In the end he will eat it again.

TASK 19: CLIMBING UP

Kara shall climb arbitrarily long stairs.

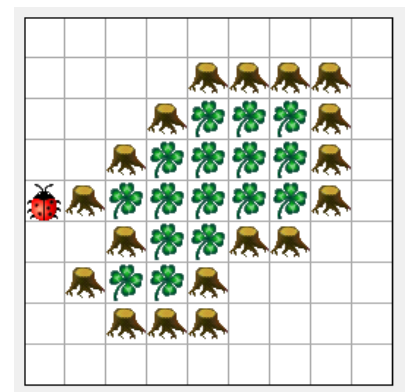
Write a method `oneStepUp()` to make Kara climb a single step. You need to figure out how Kara knows if he still has to climb a step or if he reached the top.

Note: The Solution should work with pressing only once on the Step button.

**ADDITIONAL TASK 20 (DIFFICULT): KARA AS GUARD**

Kara wants to guard the forest. He is endlessly walking along outside the forest. To help yourself, you can draw a flowchart.

Note: For an infinite loop we can press the Run button.

**Note about Kara:**

- Ideas and concepts were developed by Jürg Nievergelt, Werner Hartmann, Raimond Reichert et al., <http://www.swisseduc.ch/informatik/karatojava/>, February 2011.
- Some Kara exercises are based on material by Horst Gierhardt, <http://www.swisseduc.ch/informatik/karatojava/javakara/material/>, February 2011.