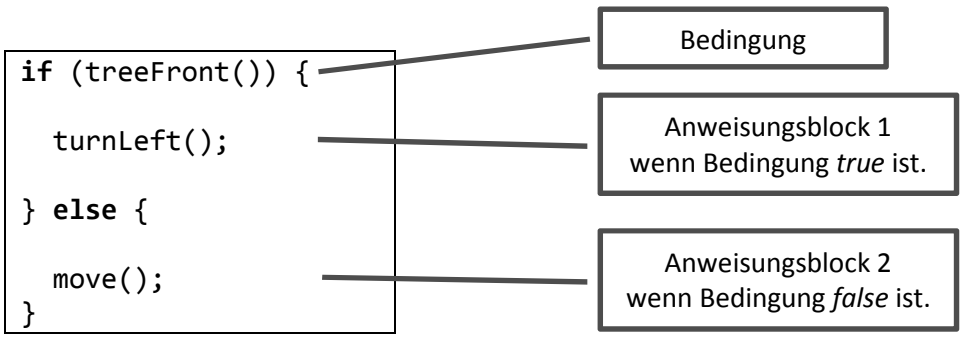
Kapitel 2: Programmfluss steuern

Bedingte Anweisungen in Dart[1]



Hinweis: Der `else`-Teil (Anweisungsblock 2) kann, wenn er nicht benötigt wird, weggelassen werden.

Beschreiben Sie mit Worten, was die folgenden Codebeispiele bewirken.

a)

```
if (onLeaf()) {

  removeLeaf();

} else {

  putLeaf();
}
move();
```

Nimmt ein Blatt weg, wenn es eines hat, legt

ein Blatt hin, wenn keines dort ist (Invertieren).

b)

```
if (onLeaf()) {

      move();
}
```

Wenn er auf einem Blatt ist, geht er einen Schritt vorwärts.

[1] Basiert auf: Thomas Kempe und David Tepasse, Informatik 1 - Softwareentwicklung mit Greenfoot und BlueJ, 1. Aufl (Paderborn: Schöningh, 2010), S. 36.

Verschachtelung

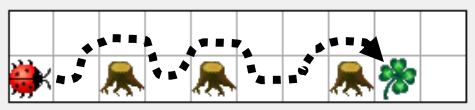
Bedingte Anweisungen können auch ineinander verschachtelt sein. Beschreiben Sie, was beim Ausführen des Programms in der gegebenen Kara Welt passiert. Zeichnen Sie ein entsprechendes Flussdiagramm.

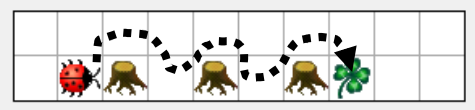
```
if (treeLeft()) {
  if (onLeaf()) {
    removeLeaf();
    move();
  } else {
    move();
  }
} else {
  move();
}
```

Nimmt das Blatt auf, wenn links ein Baum steht.

AUFGABE 04

- Starten Sie das **`scenario04a()`** aus dem Projekt ***dart-kara-chapter-2***. In MyKara ist die Methode `goAroundTree()` bereits programmiert und ein Teil der `act()`-Methode ist vorbereitet.
- Programmieren Sie den Ablauf so, dass Kara das Kleeblatt in allen Welten mit den folgenden Eigenschaften erreicht:
  - Das Kleeblatt liegt immer gerade vor ihm – er muss nur um die Bäume herumlaufen.
  - Es stehen nie zwei Bäume nebeneinander.
- Zwei mögliche Welten sehen Sie unten abgebildet.
- Für diese Aufgabe haben Sie **verschiedene Welten** zur Verfügung (a, b und c). Das gleiche Programm soll also mit **`scenario04a`**, **`scenario04b`**, **`scenario04c`** funktionieren.





WEITERE AUFGABEN...

Logische Operatoren

Unser Kara kann nun schon mehr als einfach nur simple Befehle abzuarbeiten. Er kann durch die Benutzung von Bedingungen auf die jeweiligen Ergebnisse eines Sensors unterschiedlich reagieren. Es sollte Kara aber natürlich auch möglich sein, auf zwei oder mehrere Sensoren gleichzeitig zu reagieren.

Die folgende Tabelle zeigt die drei wichtigsten logischen Operatoren in Dart:

Operator	Beschreibung	Beispiel	
&&	**und**	`treeFront() && onLeaf()`	Ist nur erfüllt (`true`) , wenn beide Aussagen erfüllt sind, d.h. wenn Kara vor einem Baum und auf einem Blatt steht.
||	**oder**	`treeFront() || onLeaf()`	Ist dann erfüllt (`true`), wenn entweder die eine oder die andere Aussage oder beide erfüllt sind.
!	**nicht**	`!treeFront()`	Ändert einen Ausdruck von `true` in `false` und umgekehrt. Diese Aussage wäre also dann erfüllt (`true`), wenn Kara nicht vor einem Baum steht.

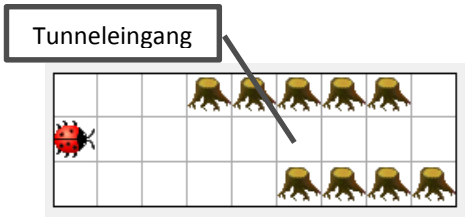
Ein Beispiel in Dart würde wie folgt aussehen:

```
if (treeLeft() && onLeaf()) {
      // Mache etwas…
}
```

oder auch kombiniert:

```
if (treeLeft() && !treeRight()) {
      // Mache etwas…
}
```

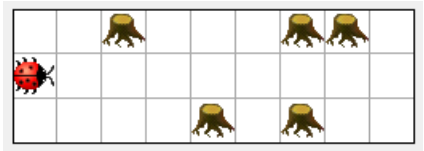
Aufgabe 05: Angst vor Tunnel



Kara hat etwas Angst vor Tunneln. Er soll auf jedem Feld überprüfen, ob es ein Tunneleingang ist (d.h. ob es auf beiden Seiten Bäume hat). Ist dies der Fall, so lässt er vor Schreck gleich ein Kleeblatt fallen.

Laden Sie ***scenario05…***, schreiben Sie das Programm und testen Sie es mit allen drei Welten.

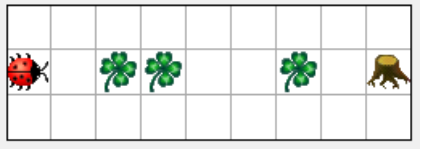
Aufgabe 06: Blatt beim Baum



Nun soll Kara wieder geradeaus gehen und überall dort ein Blatt legen, wo entweder links oder rechts oder auf beiden Seiten ein Baum steht.

Laden Sie ***scenario06…*** und schreiben Sie das Programm dazu.

Aufgabe 07: Blätter legen bis zum Baum



Kara soll vorwärts laufen und dabei überall ein Blatt legen, wo keines ist. Wenn er beim Baum angelangt ist, soll er nichts mehr machen.

Laden Sie ***scenario07…***, schreiben Sie das Programm und testen Sie es.

Weitere Aufgaben folgen…

Schleifen

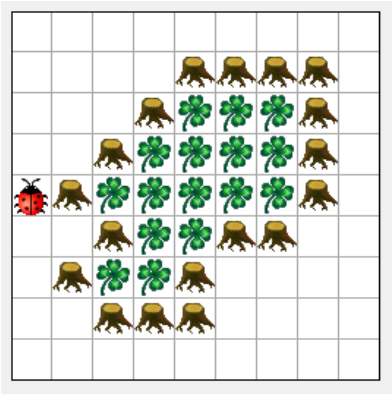
Fortsetzung folgt mit folgenden Themen:

- while-Schleife
- for-Schleife
- Weitere Aufgaben zu Bedingten Anweisungen
- Aufgaben zu Schleifen

ZUSATZAUFGABE 08 (SCHWIERIG): KARA ALS WÄCHTER

Kara will einen Wald bewachen. Er soll aussen am Waldrand entlang laufen. Nach 1000 Schritten soll er schliesslich erschöpft aufgeben.

Testen Sie Ihr Programm mit dem **`scenario08a`** und **`scenario08b`**.



Hinweis zu Kara:

- 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.
- Einige Kara-Übungen basieren auf Unterlagen von Horst Gierhardt, http://www.swisseduc.ch/informatik/karatojava/javakara/material/, abgerufen Februar 2011.