AB03 - Schleifen: Die Schafe machen etwas immer wieder...

Nicht jede Wiese ist gleich groß, der Stall ist nicht immer gleich weit weg. Trotzdem sollen die Schafe richtig reagieren. Daher müssen sie ihre Umwelt wahrnehmen und darauf reagieren.

> **Ziel:** Wiederholungen in Handlungen erkennen, als SOLANGE-Schleife formulieren und in Programmiersprache umsetzen können. Methoden mit Parametern benutzen können.

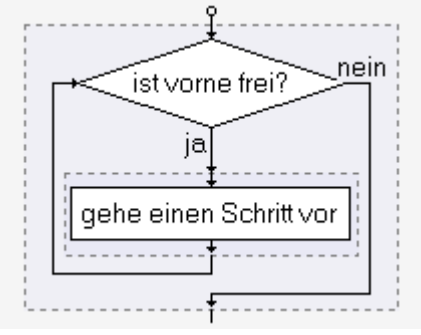
(A1) Wiese erkunden

Die Schafe sollen eine Runde auf ihrer Wiese drehen, egal wie groß diese Wiese ist. Dazu gibt es eine Methode `vorBisZaun()`. Teste diese Methode und lies dann im Quelltext vom `AB3_Schaf` nach, wie sie implementiert wurde. Setze diese Methode ein, um die Schafe eine Runde entlang des Zauns ihrer Wiese drehen zu lassen. Teste diese Methode an verschiedenen Schafen.

Der Auftrag `vorBisZaun()` muss korrekt ausgeführt werden, egal wie weit der Zaun entfernt ist. Daher muss das Schaf prüfen, wie lange es vorwärts gehen muss. Auf die Anfrage: `istVorneFrei()` liefert es die Antwort `true` bzw. `false`. Damit können wir diese **bedingte Wiederholung** so formulieren:

-- *normierte Sprache*	-- *Programmiersprache*

SOLANGE (vorne frei ist)
WDH:
ein Schritt vor
ENDE WDH
	`while (istVorneFrei())`
	`{`
	`einsVor();`
	`}`



Genau so steht es auch im Quelltext. Im runden Klammerpaar hinter dem Schlüsselwort while(...) steht die Bedingung, welche die Wiederholungen steuert. Solange diese Ausführungsbedingung gilt, werden alle Anweisungen im Schleifenblock zwischen { und } wiederholt.

Oder anders ausgedrückt: Die Ausführungsbedingung wird überprüft. Ist sie wahr, so wird jede Anweisung in der Blockklammer zwischen { und } ausgeführt. Danach wird wieder überprüft, ob die Ausführungsbedingung immer noch wahr ist. Die Anweisungen innerhalb der Blockklammer werden wieder ausgeführt. Und so weiter und so fort. Erst wenn die Ausführungsbedingung falsch ist, wird die Schleife beendet und die Befehle hinter der schließenden Klammer } ausgeführt. Innerhalb des wiederholten Vorgangs muss sich die Ausführungsbedingung verändern, damit die Wiederholungen schließlich aufhören und nicht endlos laufen.

(A2) Zum Gras

Jedes Schaf muss fressen und dafür muss es einen schönen Grasbüschel finden. Es gibt eine Methode `istAufGegenstand()`, die testet, ob eine Figur auf einem Gegenstand (also z.B. auch Gras) steht. Nutze diese Methode, um die Schafe bis zum nächsten Grasbüschel laufen zu lassen und danach das Gras zu fressen. Implementiere dies in der Methode `vorBisGrasUndFriss()`.
Hinweis: Es muss also so lange laufen, wie es (noch) nicht auf einem Grasbüschel steht. In Java hat `while(!Bedingung)` die Bedeutung "solange die Bedingung nicht erfüllt ist, mache...". Das Ausrufezeichen heißt also "nicht".

Teste die Methode an verschiedenen Schafen. Beschreibe, was passiert, wenn das Schaf schon auf einem Grasbüschel steht oder nirgends vor dem Schaf Gras zu finden ist.

<<< Zurück zu Level 2 **AB03** Weiter zu Level 4 >>>