



Die Schafe können schon eine ganze Menge. Viel mehr lernt so ein Schaf auch nicht. Daher wenden wir uns nun dem Bauern zu und bringen ihm einiges bei...

## Neue Sensoren ...

**ZIEL:** Methoden mit boolschen Rückgabewerten erstellen können. Parameter verwenden können, um Methoden Zusatzinformationen mitzugeben.

Unsere Figuren haben nur eine beschränkte Anzahl von Methoden, die die Umwelt der Figur wahrnehmen (z.B. `istVorneFrei()`, `istVorne("Wasser")`). In diesem Arbeitsblatt lernst, du neue (komplexere) Sensoren selbst zu erstellen. Schau dir dazu die Methode `istAufLeeremAcker()` des `AB5_Bauer` an. Im Gegensatz zu den Methoden, die du bisher implementiert hast, gibt diese eine Antwort zurück (wie z.B. auch `istVorneFrei()`).

### Aufgaben:

1. Teste die Methode `istAufLeeremAcker()` an den verschiedenen Bauern. Wie beantwortet der Bauer diese Anfrage? Welche Antwortmöglichkeiten gibt es?
2. Analysiere nun den Quelltext zur Methode `istAufLeeremAcker()`: Wo tauchen die Antwortmöglichkeiten (`true` und `false`) im Quelltext auf? Welcher Befehl sorgt dafür, dass eine Antwort zurückgegeben wird? Wie unterscheidet sich der Methodenkopf, d.h. die mit `public ...` beginnende Zeile, von den bisherigen Methodenköpfen?

```
public boolean istAufLeeremAcker() {  
    if(istAuf("Acker") && !istAuf("Tomate")) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Alle bisherigen Methoden wurden mit `public void methodeName() {...}` programmiert. `Void` steht dabei für leer/nichts. `Void` steht an der Stelle, an der der sogenannte Rückgabotyp steht. Es wird also nichts zurückgegeben. Möchte man nun mit `true/false` antworten, muss man den Rückgabewert als `boolean`<sup>1</sup> deklarieren. Mit `return true/false;` kann man dann den gewünschten Wert zurückgeben. `Return` beendet außerdem das Unterprogramm. Es werden also keine Befehle mehr nach dem Ausführen eines `Return` ausgeführt.

### Aufgaben:

3. **Erntebereit:** Reife Tomaten kann man mit `ernte("Tomate")` ernten. Damit dabei kein Fehler passiert, muss man testen, ob man überhaupt auf einer Tomate steht und ob diese reif ist. Den Reifegrad kann man mit `pruefe("Tomate")` bestimmen. Hat er den Wert 100, kann die Tomate geerntet werden. Implementiere die Methode `boolean istAufReiferTomate()`, die diesen Test vornimmt. Hinweis 1: Dieser Test soll die Tomate nicht ernten. Tests sollten eigentlich nie Änderungen vornehmen, da der Benutzer das nicht erwartet. Hinweis 2: In AB4 Aufg. 3 wird erklärt, wie man zwei Bedingungen verknüpfen kann.
4. **Gießen:** Die Tomaten wachsen leider nicht so leicht wie Gras. Tomaten haben einen sehr hohen Wasserbedarf. Daher muss man sie mit `benutze("Giesskanne")` gießen. Vorher sollte man aber prüfen, ob man auf einer Tomate steht und sie Wasser benötigt (`pruefe("Tomate")` liefert dann den Wert -1). Implementiere die Methode `boolean istAufDurstigerTomate()`, die diesen Test vornimmt.
5. **Out of Power:** Implementiere eine Methode `istEnergieSchwach()`. Diese soll `true` zurückgeben, wenn mindestens einer der Energiewerte (Hunger und Durst) des Bauern auf weniger als 40 Energiepunkte gesunken ist. Ansonsten gibt sie `false` zurück. Teste deine Methode an den unterschiedlichen Bauern.

<sup>1</sup> Nach George Boole, der sich mit dem mathematischen Teilbereich Logik beschäftigt hat. Die Logik beschäftigt sich u.A. mit Aussagen, die nur die Wahrheitswerte wahr oder falsch annehmen können.



## Parameter

Viele Methoden benötigen zusätzliche Informationen, um korrekt arbeiten zu können. Du hast z.B. schon die Methoden `pruefe` oder `istAuf` kennen gelernt. Beide erwarten einen Text als Parameter. Ihr Methodenkopf sieht folgendermaßen aus:

```
public boolean istAuf(String name) {...}
```

```
public int pruefe(String name) {...}
```

Man gibt also in der runden Klammer hinter dem Methodennamen die Parameter an. Dabei wird zuerst der Typ genannt (hier `String`) und dann der Name des Parameters (hier `name`). Diesen Parameter kann man dann überall in der Methode nutzen. Dabei wird nur noch der Name angegeben und nicht mehr der Typ.

6. **Besitztümer:** Der Bauer kann Gegenstände mit sich herumtragen. Mit `getAnzahl(String name)` kann man testen, wie viele Gegenstände eines bestimmten Typs er bei sich hat. Untersuche die Methode `hatGegenstand` des `AB5_Bauer`. Welchen Parameter erwartet diese Methode? Teste die Methode an den Bauern. Wie wirkt es sich aus, dass die Methode einen Parameter hat? Wo wird dieser Parameter innerhalb der Methode genutzt?
7. **Gießbereit:** Implementiere eine Methode `boolean kannGiessen()`, die testet, ob der Bauer eine Gießkanne ("Giesskanne") hat und sie gefüllt ist, d.h. das Prüfen der Gießkanne einen Wert größer als 0 ergibt.  
Hinweis: Hier musst du noch keinen eigenen Parameter definieren, sondern siehst wie man eine Methode mit Parameter nutzt.
8. **Blick nach links:** Implementiere eine Methode `boolean istLinks(String name)`, die testet, ob links vom Bauern ein bestimmter Gegenstand ist. Der Name dieses Gegenstandes ist der Parameter der Methode. Da der Bauer nur die Methode `istVorne(String name)` kennt, musst du den Bauern zunächst nach links drehen. Am Ende soll der Bauer wieder so stehen wie am Anfang.  
Hinweis: Befehle nach der `return` Anweisung werden nicht mehr ausgeführt. An welcher Stelle muss man den Bauern wieder zurückdrehen?
9. **Positionslauf:** Implementiere eine Methode `laufeBisY(int y)`, die den Bauern so lange vorwärts gehen lässt, wie die durch den Parameter angegebene y-Koordinate (`getY()`) noch nicht erreicht ist. Du kannst davon ausgehen, dass die angegebene y-Koordinate vor dem Bauern liegt und er sich nicht dafür drehen muss. Teste deine Methode. Welche Art von Parameter erwartet sie?  
Für Experten: Drehe den Bauer zusätzlich zunächst in die richtige Richtung.

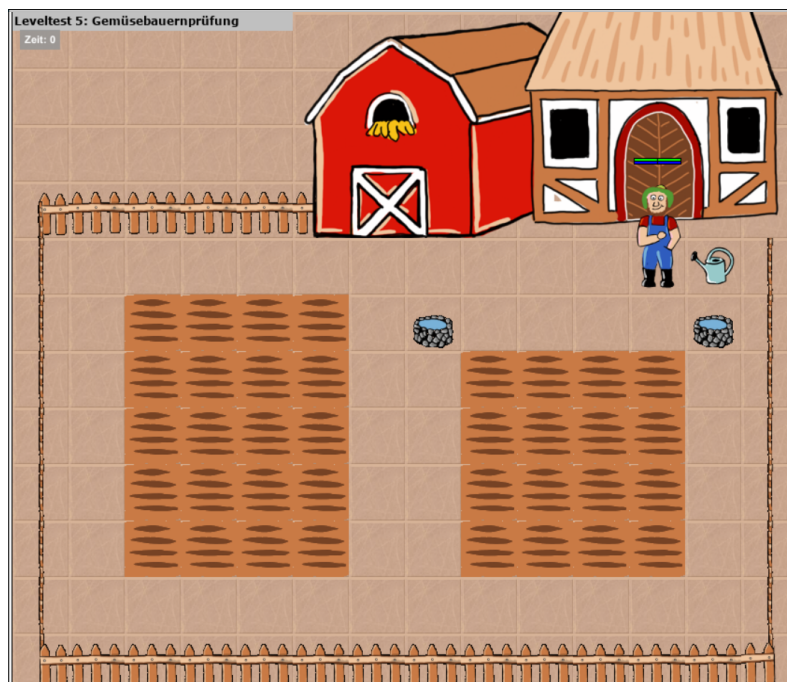


## Leveltest 5: Gemüsebauernprüfung

Dein Bauer ist bereit: Er hat sich für die Gemüsebauerprüfung angemeldet. Seine Aufgabe ist es, Tomaten anzubauen und mindestens 20 Tomaten zu ernten. Je schneller er diese Aufgabe erledigt, desto besser.

Er hat am Anfang vier Tomaten. Diese kann er pflanzen, indem er sie auf einem leeren Ackerfeld ablegt. Danach muss er fleißig gießen, bis die Tomaten herangereift sind. Sind sie schön rot, kann er sie ernten. Tomatenpflanzen produzieren bei ausreichender Wasserversorgung immer wieder neue Tomaten, es müssen keine neuen gepflanzt werden.

So viel Arbeit ist natürlich anstrengend. Daher muss der Bauer immer mal wieder eine Vesperpause in seinem Haus machen, um seine Energiereserven aufzufüllen.



### Hilfreiche Methoden:

<code>void ablegen(String name)</code>	der Bauer legt einen Gegenstand aus seinem Inventar ab.
<code>void aufnehmen()</code>	der Bauer hebt einen Gegenstand vom Boden auf.
<code>void ernte(String name)</code>	der Bauer kann "Tomate" und "Korn" ernten, wenn es reif ist.
<code>void benutze(String name)</code>	der angegebene Gegenstand wird benutzt. Benutzt man eine Gießkanne ( <code>benutze("Giesskanne")</code> ) vor einem Wasserfeld, wird sie gefüllt. Benutzt man sie auf einem Acker, dann gießt man den Boden damit.
<code>void vesperpause()</code>	der Bauer macht eine Pause im Haus. Er muss dazu direkt vor der Tür zum Haus stehen.
<code>int getAnzahl(String name)</code>	gibt die Anzahl der Gegenstände vom Typ name zurück, die der Bauer mit sich herumträgt.