



"Mein Feld ist größer als deins", "meine Tomaten sind schöner als deine!": Schon immer gab es Konkurrenz unter Nachbarn. Auch unsere Bauern sind da nicht besser und wollen ihre Feldergröße vergleichen. Außerdem steht die Kornbauernprüfung an...

Lokale Variablen und Zählschleifen...

ZIEL: Lokale Variablen als Zwischenspeicher für jeweils einen Wert kennen und einsetzen können. Lokale Variable von Objektvariablen unterscheiden können. Zählschleifen einsetzen können.

Lokale Variable als Kurzzeitgedächtnis

Der AB7_Bauer hat zwei ähnliche Methoden: `bestimmeAnzahlTomatenAttribut()` und `bestimmeAnzahlTomatenLokaleVariable()`. Beide lassen den Bauern bis zum nächsten Hindernis laufen und dabei Tomatenpflanzen zählen.

Aufgaben:

1. Rufe beim Bauern auf dem Tomatenfeld zunächst in jeder Spalte die erste Methode auf. Du kannst ihn mit `umkehren(boolean linksrum)` umkehren und in die nächste Spalte laufen lassen.
2. Führe einen Reset durch. Wiederhole die Aufgabe 1 mit der zweiten Methode. Was macht die zweite Methode anders?

Manchmal müssen sich Objekte bestimmte Werte nur vorübergehend merken, um eine Aufgabe zu erfüllen, z.B. wenn die Werte nur innerhalb einer Methode benötigt werden, nach deren Abschluss nicht mehr. Es war beispielsweise im Leveltest von AB6 nicht notwendig, dass sich der Bauer die Zahl der Körner in einer Reihe dauerhaft merkt. Es musste sich das nur merken, bis er die Anzahl auf 4 Körner aufgefüllt hatte.

Bisher hatten wir Attribute der Objekte als Gedächtnis. Diese sind aber nur für Werte gedacht, die die Figuren sich dauerhaft merken sollen. Sie stellen das "Langzeitgedächtnis" dar. Du sollst dir beispielsweise auch dauerhaft merken, dass es keine "if-Schleifen" gibt. Die Koordinaten der Position des Bauernhofs kannst du aber nach Beendigung der Aufgabe wieder vergessen.

Man könnte auch das „Kurzzeitgedächtnis“ mit Attributen realisieren, würde dann aber sehr viele Attribute bekommen, die immer nur kurzfristig zum Einsatz kämen. Das fördert nicht gerade die Lesbarkeit. Daher verwenden wir sogenannte lokale Variablen, deren Geltungsbereich nur eine Methode (oder auch nur eine Schleife) ist. Man verwendet sie fast genauso wie Attribute.

Aufgaben:

3. Vergleiche die Quelltexte der beiden Methoden, die die Tomatenpflanzen zählen. Welche Variable wird jeweils zum Zählen benutzt? Wo wird diese Variable deklariert (erzeugt), initialisiert (auf ihren Startwert gesetzt) und benutzt?

4. **Debugging:** Lasse dir mit INSPECT die Attribute anzeigen. Warum sieht man hier nur den `zaehler1`? Die lokalen Variablen sieht man nur im Debugger. Setze dazu einen Breakpoint, indem du auf die Zeilennummer klickst, an der das Programm unterbrochen werden soll (hier Zeile 32). Wenn du dann die Methode aufrufst, hält das Programm am Stoppschild an und kann mit "Schritt über" Schritt für Schritt ausgeführt werden. Du siehst, dass die lokale Variable erst entsteht, wenn sie das erste Mal benutzt

The screenshot shows the AB7_Bauer class with two methods. The first method, `bestimmeAnzahlTomatenAttribut`, uses a static attribute `zaehler1`. The second method, `bestimmeAnzahlTomatenLokaleVariable`, uses a local variable `anzahl2`. The variable inspector on the right shows the state of the program. Under 'Attribute', it lists `protected ArrayList<Greenfoot>` and `private ArrayList<String> orig`. Under 'Lokal Variablen', it lists `private int anzahl1 = 0` and `int anzahl2 = 0`. Arrows point from the text 'Attribute' and 'Lokal Variablen' to their respective sections in the inspector.



wird.

Vergleich Attribut - Lokale Variable:

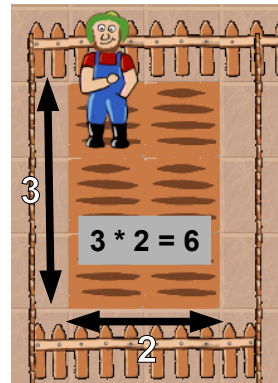
	Attribut	Lokale Variable
Deklaration	Vor den Methoden: <pre>public class AB7 Bauer { private int zaehler1; ... //Methoden ... }</pre> → mit private!	Innerhalb der Methode: <pre>public class AB7_Bauer { ... public int zaehle() { int zaehler2; ... } }</pre> → ohne private!
Initialisierung	Im Konstruktor der Klasse: <pre>public AB7 Bauer { zaehler1 = 0; ... }</pre>	Direkt nach der Deklaration: <pre>public int zaehle { int zaehler2; zaehler2 = 0; ... }</pre>
Verwendung	<pre>public int zaehle { zaehler1++; ... }</pre>	<pre>public int zaehle { zaehler2++; ... }</pre>

Hinweis: Um Attribute von lokalen Variablen unterscheiden zu können, kann man bei Attributen (außer bei der Deklaration) immer ein `this` voranstellen: z.B. `this.zaehler1 = 0;`

Aufgaben:

5. **Feldlänge 1:** Implementiere die Methode `int bestimmeLaenge()`, die dem Bauer ermöglicht, die Länge seines Feldes zu bestimmen. Dazu soll er einfach geradeaus bis zum nächsten Hindernis laufen und die Anzahl der Felder in einer lokalen Variable zählen und als Ergebnis zurückgeben.
Hinweis: beachte, dass es ein Feld mehr ist als man Schritte machen muss.

6. **Feldlänge 2:** Implementiere die Methode `int bestimmeLaenge2()`: Dekлариere drei lokale Variablen `y1`, `y2`, `y_diff`. Weise `y1` den Wert des Aufrufs von `getY()` zu. Lasse den Bauern bis zum nächsten Zaun laufen. Weise nun `y2` den Wert des Aufrufs von `getY()` zu. Berechne die Differenz von `y2` und `y1` und speichere sie in `y_diff`. Gib mit Hilfe von `y_diff` die Länge des Feldes zurück.
Was passiert, wenn der Bauer nach rechts schaut und du die beiden Methoden aufrufst?



7. **Fläche:** Implementiere eine Methode, die die Fläche eines rechteckigen Feldes (Breite * Höhe) bestimmt und zurück gibt. Du kannst davon ausgehen, dass der Bauer in einer Ecke des Feldes steht und in die passende Richtung schaut. Nutze eine der Methoden aus Aufgabe 5+6 (ggf. auch mehrfach), gerne darfst Du auch noch weitere Methoden formulieren und hier aufrufen, nutze sinnvolle Namen.
8. **Lästiges Unkraut:** Implementiere eine Methode, die den Blumenbauer die Anzahl der Felder mit Gras in seinem einzeiligen Blumenbeet zählen lässt.
Hinweis: Beachte, dass auch auf dem ersten und letzten Feld Gras wachsen kann.
9. **Blumenbeet:** Implementiere die Methode `int anzahlSchritteBisBlume3()`, die den Blumenbauer die Anzahl der Schritte bis zur dritten Blume zählen lässt und diese Anzahl zurückgibt.
Hinweis: Du brauchst zwei lokale Variablen. Eine für die Anzahl der Schritte und eine für die Anzahl der schon gefundenen Blumen.



Zählschleifen

Bisher hast du `while`-Schleifen kennengelernt. Mit ihnen kann man alles programmieren, aber manchmal geht es kompakter. Sehr häufig muss man wissen, in welchem Schleifendurchgang man ist. Macht man es zum ersten, zweiten oder dritten Mal...

Dazu braucht man eine lokale Variable, die zählt, in welchem Durchgang man ist. Dabei beginnen die Informatiker in der Regel mit 0 zu zählen.

While-Schleife	For-Schleife
<pre> 1 public void gehe4 { 2 int zaehler; 3 zaehler = 0; 4 while(zaehler < 4) { 5 einsVor(); 6 zaehler++; 7 } 8 }</pre>	<pre> public void gehe4 { for(int zaehler=0; zaehler<4; zaehler++) { einsVor(); } }</pre>

10. Vergleiche die beiden Varianten der Zählschleife. Markiere jeweils, an welcher Stelle die Zählvariable:

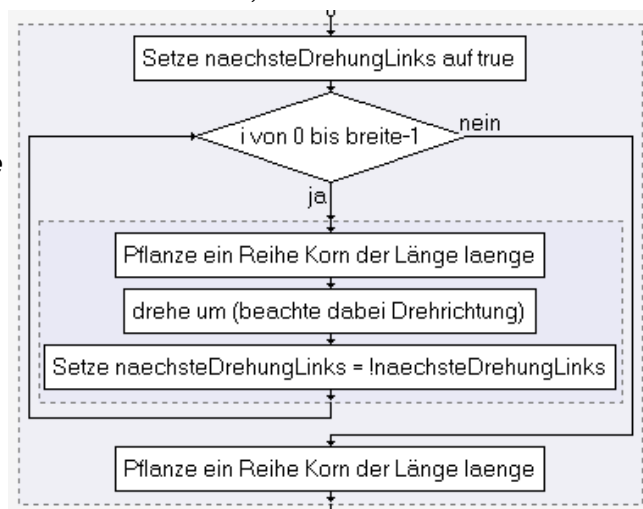
- a) deklariert b) initialisiert c) erhöht
wird und
d) die Schleifenbedingung steht.

11. **Pflanze Korn:** Die Methode `pflanzeKorn(int anzahl)` lässt den Bauern eine bestimmte Anzahl von Getreidepflanzen säen. Implementiere eine Methode `pflanzeKorn2(int anzahl)`, die diese Aufgabe mit einer `for`-Schleife statt einer `while`-Schleife löst.

12. Verbessere beide Methoden so, dass sie auch funktionieren, wenn bis direkt an den Zaun gepflanzt werden soll, d.h. der Bauer steht am Ende auf dem letzten bepflanzten Kornfeld.

13. **Drehwurm:** Implementiere eine Methode `drehwurm(int anzahl)`, die den AB7_Bauern zunächst links herum und dann rechts herum drehen lässt. Der Parameter `anzahl` gibt dabei an, wie viele komplette 360°-Drehungen er in beide Richtungen machen soll. Verwende eine `for`-Schleife.

14. **Kornfeld:** Implementiere eine Methode `kornfeld(int laenge, int breite)`, das den Bauer eine Fläche von `laenge x breite` mit Korn bepflanzen lässt. Setze dazu das abgebildete Flussdiagramm um. Verwende dabei die Methode `pflanzeKorn`. Mache dir klar, was die Zeile `naechsteDrehungLinks = !naechsteDrehungLinks` macht.

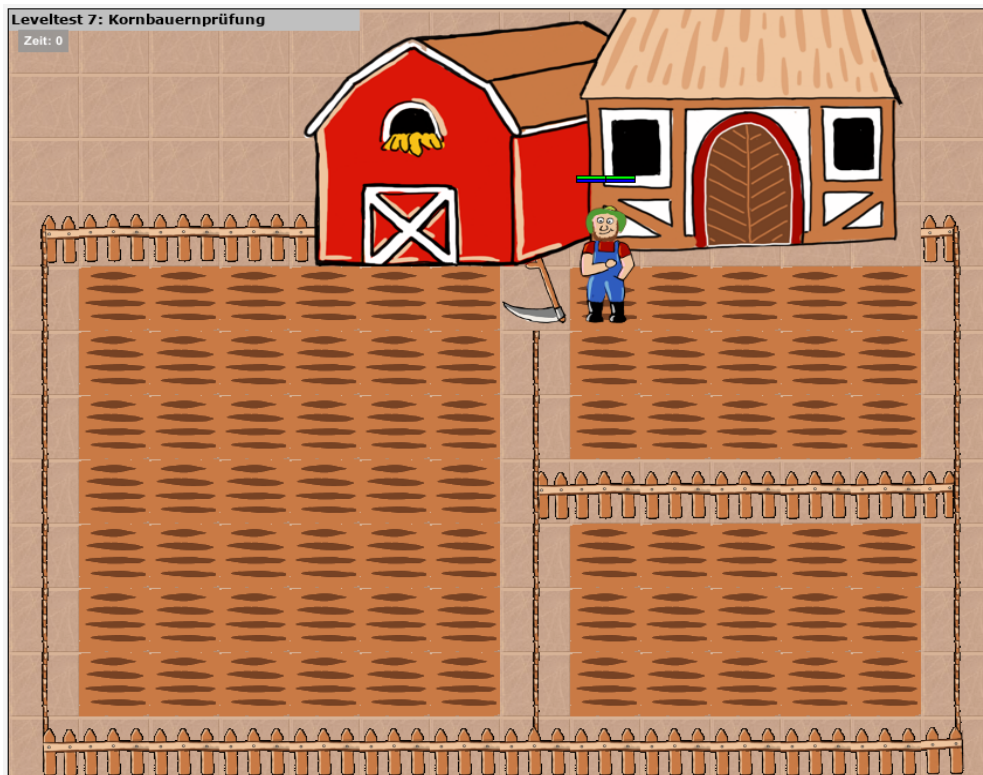




Leveltest 7: Kornbauernprüfung

Du musst dich als Kornbauer bewähren. Deine Aufgabe ist es 300 Körner (`getAnzahl("Koerner")`) in deinem Inventar zu haben. Du startest mit 70 Körnern, die du auf deinem Feld säen kannst. Sobald die Kornpflanzen reif sind, kannst du sie mit einer Sense wieder ernten (`ernte("Korn")`) und erhältst pro Kornpflanze fünf Körner. Zwischendrin musst du natürlich Vesperpause machen gehen, um nicht zu verhungern und zu verdursten.

Dein Bauer startet in der linken oberen Ecke des Feldes. Die Sense liegt auf dem Feld links daneben. Die Tür zum Hauptgebäude ist immer bei den Koordinaten (10 | 3). Du kannst davon ausgehen, dass das Korn vollständig gereift ist, wenn du das ganze Feld eingesät hast und wieder zum Ausgangspunkt zurückgekehrt bist.



Tipps: Bestimme zunächst die Länge und Breite deines Feldes. Laufe dann zurück zum Ausgangspunkt. Laufe dabei an deinem Haus vorbei und gehe etwas essen (`vesperpause()` auf Feld 10/4 aufrufen).

Verwende dann die Methode `kornfeld`, um Korn zu säen. Laufe wieder zum Startpunkt zurück. Essen nicht vergessen.

Modifiziere die Methode `kornfeld(int laenge, int breite)` so, dass sie entweder Korn pflanzt oder Korn erntet.