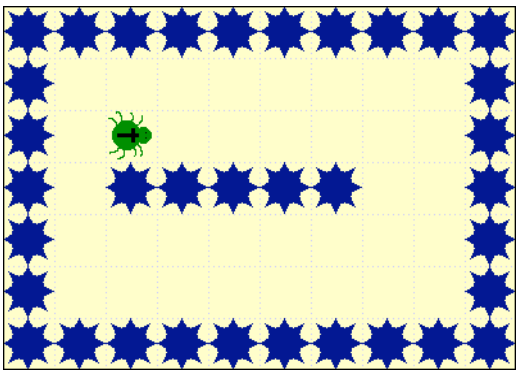
6 Die Taranteln werden flexibel — Programmschleifen

Konzepte in diesem Kapitel

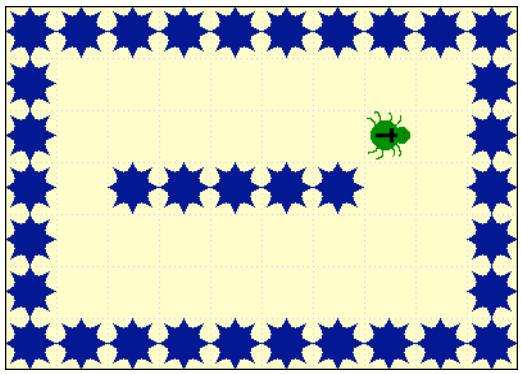
Algorithmik/Java: vorprüfende und nachprüfende Programmschleife

6.1 Die Tarantel wiederholt sich

Bisher funktioniert das Holen der Fliege nur in einer einzigen Welt. Liegt z.B. die Fliege nicht fünf, sondern sechs oder etwa drei Schritte entfernt, muss ein neues Programm geschrieben werden. Besser wäre es, man könnte der Tarantel den Befehl geben: „Lauf, *solange* du noch keine Fliege gefunden hast!“ Das erreicht man mit einer *Programmschleife*.
Ein Beispiel: Die Tarantel soll links an einer Reihe Hindernisse vorbei gehen und dann stehen bleiben. Die Reihe kann beliebig lang sein:



Anfang



Ende

Die Tarantel erhält dazu den Befehl:

umgangssprachlich	**Java**	**allgemeine Java–Syntax**

solange rechts ein Hindernis liegt
geh einen Schritt weiter
	`while(hindernisRechts()){`
	`  schritt();`
	`}`
		`while(`*Laufbedingung*`) {`
		Anweisungsblock
		`}`

Die Befehle in der Programmschleife werden solange wiederholt, wie die Sensormethode `hindernisRechts()` den Wert `true` (wahr) liefert.

Ein Ausrufezeichen kehrt den Wert der Sensormethode um. Dann läuft die Schleife, solange der Sensor `false` (falsch) liefert. Der Befehl „Gehe, solange keine Fliege am Platz liegt“ würde z.B. lauten:

```
while(!fliegeAmPlatz()) {
    schritt();
}
```

6.2 Tu ich's gleich — oder prüf' ich's erst?

Die bisher besprochene Form der Programmschleife nennt man die *vorprüfende Schleife*. Eine andere Form ist die *nachprüfende Schleife*, z.B.:

umgangssprachlich	**Java**	**allgemeine Java–Syntax**

führe aus
geh einen Schritt weiter
solange rechts ein Hindernis liegt
	`do {`
	`  schritt();`
	`}`
	`while(hindernisRechts());`
		`do {`
		Anweisungsblock
		`}`
		`while(`*Laufbedingung*`);`

Aufgabe 1: Vergleichen Sie die vorprüfende und die nachprüfende Schleife:

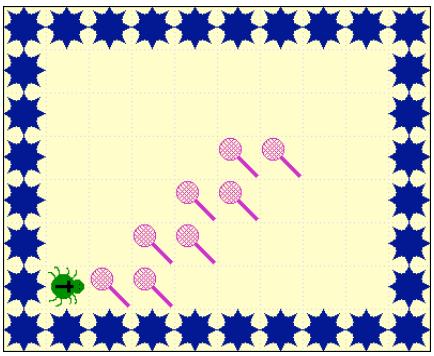
- Wann wird die Bedingung geprüft, wann die Befehle ausgeführt?
- Wie oft läuft die Schleife mindestens?
- In welchem Fall setzt man günstiger welche Schleife ein?

Aufgabe 2: Tragen Sie die allgemeine Java–Syntax der Schleifen in Ihr Arbeitsblatt ein. Formulieren und testen Sie dann folgende Schleifen vor- und nachprüfend:

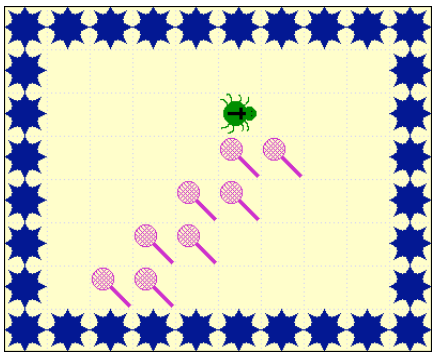
- Solange vorne ein Hindernis liegt, drehe dich nach links.
- Solange rechts kein Hindernis liegt, friss eine Fliege und geh einen Schritt.
- Solange am Platz eine Fliege liegt, friss sie, drehe dich nach links und geh einen Schritt.
- Solange vorne keine Klatsche lauert, geh einen Schritt.

Aufgabe 3: Schreiben Sie die Methoden `fliegeSuchen()` und `rueckWeg()` so um, dass sie für beliebige Strecken funktionieren. Testen Sie Ihr Programm mit verschieden weit entfernten Fliegen.

Aufgabe 4: Schreiben Sie die Methode `treppeSteigen()` für beliebig hohe Treppen aus Klatschen. Überlegen Sie dazu, wie die Tarantel erkennt, ob sie noch eine Stufe hochsteigen muss:



Anfang



Ende