Michael Kölling

Einführung in Java mit Greenfoot

2., aktualisierte Auflage

Objektorientierte Einführung mit Spielen und Simulationen

ALWAYS LEARNING

PEARSON

Was uns noch bleibt, ist, die Würmer zu zählen und zu prüfen, ob wir acht erreicht haben. Dies muss jedes Mal erfolgen, wenn wir einen Wurm fressen. Deshalb suchen wir nach unserer Methode **lookForWorm**, die den Code zum Fressen der Würmer enthält. Hier fügen wir eine Codezeile hinzu, die die Wurmzählung inkrementiert:

```
wormsEaten = wormsEaten + 1;
```

Wie immer bei einer Zuweisung wird zuerst die Seite rechts des Zuweisungssymbols ausgewertet (**wormsEaten + 1**). Wir lesen also den aktuellen Wert von **wormsEaten** und addieren dazu eine 1. Anschließend weisen wir diesen Wert wieder der Variablen **wormsEaten** zu. Als Folge wird die Variable um 1 inkrementiert.

Jetzt benötigen wir noch eine **if**-Anweisung, die prüft, ob wir schon acht Würmer gefressen haben, und in diesem Fall den Sound abspielt und die Ausführung anhält. Listing 4.6 enthält den vollständigen Code der Methode **lookForWorms**. Die hier verwendete Sounddatei (*fanfare.wav*) liegt bereits im *sounds*-Ordner deines Szenarios, sodass sie einfach abgespielt werden kann.

Listing 4.6: Würmer zählen und prüfen, ob wir gewonnen haben.

```
/**
 * Prüft, ob wir auf einen Wurm gestoßen sind.
 * Wenn ja, wird er gefressen. Wenn nein, passiert nichts.
 * Wenn wir 8 Würmer gefressen haben, haben wir gewonnen.
 */
public void lookForWorm()
{
    if ( isTouching(Worm.class) )
    {
        removeTouching(Worm.class);
        Greenfoot.playSound("slurp.wav");

        wormsEaten = wormsEaten + 1;
        if (wormsEaten == 8)
        {
            Greenfoot.playSound("fanfare.wav");
            Greenfoot.stop();
        }
    }
}
```

Übung 4.25 Füge den zuvor besprochenen Code in dein Szenario ein, teste ihn und stelle sicher, dass er funktioniert.

Übung 4.26 Als zusätzliche Kontrolle kannst du den Objektinspektor für dein Krabben-Objekt öffnen (wähle dazu einfach INSPIZIEREN aus dem Kontextmenü der Krabbe), bevor du mit dem Spielen anfängst. Lass den Inspektor geöffnet, während du spielst, und beobachte die Variable **wormsEaten**.

4.16 Weitere Ideen

Das Szenario *little-crab-5* im Szenarien-Ordner des Buchkapitels ist eine Version des Projekts, das alle hier besprochenen Erweiterungen beinhaltet.

Wir werden uns jetzt von diesem Szenario verabschieden und einem anderen Beispiel zuwenden, auch wenn es noch viele naheliegende (oder weniger naheliegende) Möglichkeiten für Verbesserungen gibt. So könntest du zum Beispiel

- verschiedene Bilder für den Hintergrund und die Akteure verwenden;
- weitere Arten von Akteuren einführen;
- dich nicht automatisch vorwärtsbewegen, sondern nur, wenn die Taste ↑ gedrückt wird;
- ein Spiel entwickeln, das von zwei Spielern gespielt wird, indem du eine zweite tastaturgesteuerte Klasse einführst, die auf andere Tasten reagiert;
- festlegen, dass für jeden gefressenen Wurm (oder zu beliebigen Zeiten) neue Würmer auftauchen
- und dir noch vieles mehr überlegen.

Übung 4.27 Das Bild der Krabbe ändert sich beim Laufen sehr schnell, was die Krabbe ein wenig hyperaktiv wirken lässt. Vielleicht wäre es netter, wenn sich das Bild der Krabbe nur bei jedem zweiten oder dritten **act**-Schritt ändern würde. Versuche einmal, dies zu implementieren. Dazu könntest du einen Zähler hinzufügen, der in der **act**-Methode inkrementiert wird. Jedes Mal, wenn dieser Zähler 2 (oder 3) erreicht, ändert sich das Bild und der Zähler wird wieder auf 0 zurückgesetzt.

Zusammenfassung der Programmiertechniken

In diesem Kapitel haben wir uns mit einer Reihe von wichtigen neuen Programmierkonzepten vertraut gemacht. Wir haben gesehen, wie Konstruktoren dazu genutzt werden können, um Objekte zu initialisieren – Konstruktoren werden immer ausgeführt, wenn ein neues Objekt erzeugt wird.

Wir haben Instanzvariablen und lokale Variablen kennengelernt. Instanzvariablen (auch *Zustandsfelder* genannt) werden – zusammen mit Zuweisungsanweisungen – verwendet, um Informationen in Objekten zu speichern, auf die man später zugreifen kann. Lokale Variablen werden eingesetzt, um Informationen für eine kurze Zeit zu speichern – innerhalb einer einzelnen Methodenausführung –, sie werden verworfen, wenn die Methode endet.

Wir haben die Anweisung **new** verwendet, um neue Objekte im Quelltext zu erzeugen, und zum Schluss haben wir eine vollständige Version der **if**-Anweisung kennengelernt, die einen **else**-Teil umfasst, der ausgeführt wird, wenn die Bedingung nicht wahr ist.

Mithilfe all dieser Techniken können wir bereits eine ganze Menge Code schreiben.

Zusammenfassung der Konzepte

- Der **Konstruktor** einer Klasse ist eine besondere Art Methode, die immer automatisch ausgeführt wird, wenn ein Objekt dieser Klasse erzeugt wird.
- Java-Objekte können im Programmquelltext mithilfe des Schlüsselwortes **new** erzeugt werden.
- **Variablen** dienen zum Speichern von Informationen (Objekte oder Werte), die später benötigt werden.
- Variablen können durch das Schreiben einer **Variablendeklaration** erzeugt werden.
- Wir können Werte in Variablen mithilfe einer **Zuweisungsanweisung** (**=**) speichern.
- Variablen von **primitiven Typen** speichern Zahlen, boolesche Werte und Zeichen; Variablen von **Objekttypen** speichern Objekte.
- Objekte werden in Variablen gespeichert, indem eine **Referenz** auf das Objekt gespeichert wird.
- Greenfoot-Akteure verwalten ihr sichtbares Bild in Form eines Objekts vom Typ **`GreenfootImage`**.
- **Instanzvariablen** (auch **Zustandsfelder** genannt) sind Variablen, die zu einem Objekt gehören (und nicht zu einer Methode).
- **Lebenszeit von Instanzvariablen:** Instanzvariablen bestehen so lange, wie das Objekt existiert, zu dem sie gehören.
- **Lebenszeit von lokalen Variablen:** Lokale Variablen bestehen nur während einer einzigen Methodenausführung.
- Mithilfe des doppelten Gleichheitszeichens (**==**) können wir prüfen, ob zwei Dinge **gleich** sind.
- Die **`if/else`**-Anweisung führt ein Codefragment aus, wenn eine gegebene Bedingung wahr ist, und ein anderes Codefragment, wenn die Bedingung falsch ist.

Vertiefende Aufgaben

Dieses Mal machen wir mehr Übungen mit Methodenaufrufen, einschließlich einer neu geerbten **Actor**-Methode, und üben den Umgang mit sowie den Einsatz von Variablen.

Noch einmal Krabben

Übung 4.28 Die Hummer sollen ein bisschen gefährlicher werden. Die **Actor**-Klasse besitzt eine Methode namens **turnTowards**. Verwende diese Methode, damit sich die Hummer ab und an der Mitte des Bildschirms zuwenden. Experimentiere sowohl mit der Häufigkeit dieser Richtungswechsel als auch mit unterschiedlichen Laufgeschwindigkeiten für die Hummer und die Krabbe.

Übung 4.29 Statte die Krabbe mit einem Zeitzähler aus. Dazu kannst du eine **int**-Variable hinzufügen, die jedes Mal, wenn sich die Krabbe bewegt, hochgezählt wird. (Tatsächlich zählst du dann die **act**-Schritte.) Sollte dies eine lokale Variable oder eine Instanzvariable sein? Warum?

Übung 4.30 Führe dein Spiel aus. Sobald du einen Sieg geschafft hast (also acht Würmer zu fressen), untersuche dein Krabben-Objekt und prüfe, wie lange du gebraucht hast. Wie viele **act**-Schritte sind verbraucht?

Übung 4.31 Verschiebe deinen Zeitzähler aus der **Crab**-Klasse in die Klasse **CrabWorld**. (Es ist sinnvoller, dass die Welt anstatt eine einzelne Krabbe die Zeit verwaltet.) Die Variable ist einfach zu verschieben. Für das Verschieben der Anweisung, die die Zeit hochzählt, musst du eine **act**-Metode in der **CrabWorld**-Klasse definieren. **World**-Unterklassen können ebenso wie **Actor**-Unterklassen **act**-Methoden besitzen. Um eine neue **act**-Methode in **CrabWorld** zu erzeugen, kopiere einfach die Signatur der **act**-Methode der Krabbe und platziere hier deine Anweisung für den Zeitzähler.

Übung 4.32 Mache aus dem Zeitzähler deines Spiels einen Spielzähler. Das heißt: Initialisiere die Zeitvariable aus irgendeinen Wert (z.B. 500) und zähle bei jedem **act**-Schritt rückwärts (ziehe 1 vom Wert der Variable ab). Wenn der Zähler 0 erreicht, dann beende das Spiel mit einem Sound für „Zeit ist abgelaufen". Experimentiere mit unterschiedlichen Werten für die Spielzeit.

Übung 4.33 Untersuche die Methode **showText** in der **World**-Klasse. Wie viele Parameter hat sie? Welche sind es? Was gibt sie zurück? Was macht sie?

Übung 4.34 Zeige den Spielzähler auf dem Bildschirm an, indem du die **showText**-Methode verwendest. Du kannst dies in der **act**-Methode von **CrabWorld** machen. Dazu benötigst du eine Anweisung wie diese:

```
showText("Time left: "+ time, 100, 40);
```

wobei **time** der Name deiner Zählervariable ist. (Beachte: diese Anweisung verwendet den Plus-Operator und einen Text-String, dies werden wir in ***Kapitel 5*** behandeln.)

Übungen mit springendem Ball

Übung 4.35 Erzeuge ein neues Szenario und in diesem eine Welt- und eine Akteur-Klasse mit Namen *Ball*. (Weise ihm ein Bild zu, das wie ein Ball aussieht.) Programmiere den Ball so, dass er sich mit konstanter Geschwindigkeit bewegt, und lasse ihn von den Rändern der Welt abprallen.

Übung 4.36 Programmiere deinen Ball so, dass er zählt, wie häufig er vom Rand abgeprallt ist. Führe dein Szenario aus, wenn der Objekt-Inspektor des Balls zum Testen geöffnet ist.

Übung 4.37 Programmiere dein Szenario so, dass am Start automatisch drei Bälle vorhanden sind.

Übung 4.38 Verändere deinen Initialisierungscode, sodass die drei Bälle an zufälligen Orten erscheinen.

Übung 4.39 Verändere den Code für das Abprallen vom Rand, sodass die Bälle in zufälligen Winkeln vom Rand abprallen.

EXKURS

1 Szenarien teilen

In diesem Abschnitt wollen wir keine neuen Programmiertechniken einführen, sondern einen kleinen Abstecher machen und besprechen, wie du das, was du entwickelt hast, mit anderen teilen kannst. Die „anderen" können dein Freund sein, der neben dir sitzt, oder ein anderer Greenfoot-Programmierer am anderen Ende der Welt.

E1.1 Dein Szenario teilen

Wenn du die Entwicklung eines Szenarios – sei es ein Spiel oder eine Simulation – abgeschlossen hast, möchtest du vielleicht auch andere daran teilhaben lassen. Diese Benutzer sollten das Spiel starten (und neu starten) können, sie benötigen jedoch keinen Zugriff auf das Klassendiagramm oder den Quelltext. Sie sollen das Spiel nicht ändern, sondern lediglich damit spielen.

Damit ein Programm nach dem Teilen wie gewünscht funktioniert, ist es wichtig, dass alle Akteure, die zu Beginn des Spiels auf dem Bildschirm angezeigt werden sollen, automatisch erzeugt werden. Benutzer haben keine Möglichkeit, Objekte interaktiv zu erzeugen.

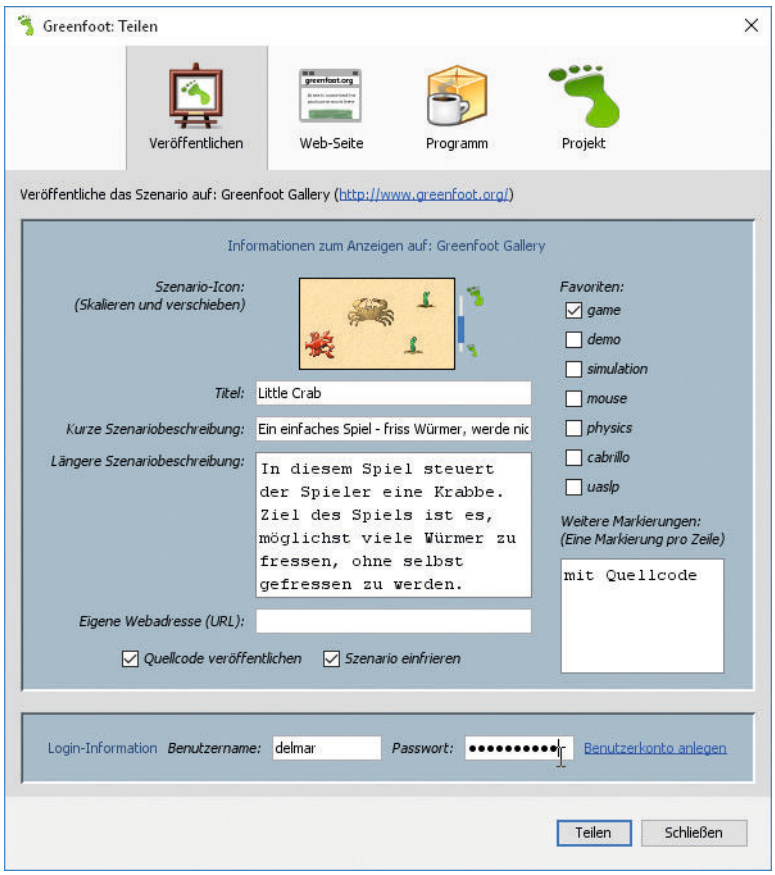
In Greenfoot musst du dazu das Szenario *teilen*. Hierfür steht dir der Button TEILEN oben rechts im Hauptfenster von Greenfoot zur Verfügung. Damit rufst du ein Dialogfeld auf, in dem du zwischen vier Optionen wählen kannst: VERÖFFENTLICHEN, WEB-SEITE, PROGRAMM und PROJEKT.

E1.2 Auf der Greenfoot-Webseite veröffentlichen

Die häufigste Art und Weise, dein Szenario zu teilen, ist, es auf der Greenfoot-Website zu veröffentlichen. Die Greenfoot-Website ist eine öffentliche Website (zu erreichen unter der Adresse *http://greenfoot.org/home)*, die es Greenfoot-Benutzern erlaubt, ihre Greenfoot-Szenarien hochzuladen. Wenn du dein Szenario auf der Greenfoot-Website teilst, dann wird es für die ganze Welt öffentlich – jeder mit Internetzugang kann es sehen und ausführen.

Das Dialogfeld TEILEN (Abbildung E1.1) zeigt oben die Adresse an. Klick den Link ruhig an und schau mal, was es bereits auf dieser Website gibt. Es ist sicher am besten, wenn du erst einmal die Site besuchst.

Abbildung E1.1
Auf der Greenfoot-Webseite veröffentlichen.



Auf der Greenfoot-Website kann jeder Szenarien anschauen und ausführen. Doch wenn du eine Bewertung abgeben, Kommentare hinterlassen oder deine Szenarien hochladen möchtest, musst du ein Konto auf dieser Site einrichten. Das geht jedoch schnell und ist ganz einfach.

Nachdem du dein Konto eingerichtet hast, kannst du ganz einfach über das Dialogfeld in Abbildung E1.1 dein Szenario hochladen. Über das Dialogfeld kannst du ein Symbol, eine Beschreibung und Tags hinzufügen, die dein Szenario identifizieren.

Wenn du deinen Quelltext zur Verfügung stellen möchtest (über das Kontrollkästchen QUELLCODE VERÖFFENTLICHEN), wird dein kompletter Quelltext auf die Greenfoot-Site kopiert, von wo jeder sie herunterladen und lesen kann, um vielleicht daraus seine eigene Version deines Szenarios zu erstellen.

Du kannst deine veröffentlichten Szenarien später ändern und verbessern, indem du sie einfach unter dem gleichen Titel erneut exportierst.

Das Veröffentlichen deiner Szenarien auf der Greenfoot-Site ist eine gute Möglichkeit, um Feedback von anderen Benutzern zu erhalten: Vorschläge für Ergänzungen und Kommentare, was möglich ist und was nicht. Die Greenfoot-Site ist außerdem ein guter Ausgangspunkt, um Ideen für weitere Funktionalität zu sammeln und um zu lernen, wie etwas umgesetzt wird. Es gibt dort ein Diskussionszentrum, wo du Fragen stellen und beantworten oder Programmiertechniken besprechen kannst. Oder halte Ausschau nach Szenarien mit Quelltext, lade den Quelltext herunter und schau, wie andere Programmierer ihre Klassen implementiert haben.

E1.3 In eine Webseite exportieren

Die zweite Exportoption erlaubt es dir, dein Szenario in deine eigene Webseite aufzunehmen (Abbildung E1.2). Du kannst einen Speicherort und einen Namen für deinen Export auswählen, dann legt Greenfoot einen Ordner mit dem von dir gewählten Namen an. In diesem Ordner erzeugt Greenfoot eine Webseite (im HTML-Format) und konvertiert dein Szenario in ein *Applet*, das in dieser Webseite läuft. Ein Applet ist eine Version deines Java-Programms, das eingebettet in einer Webseite in einem Webbrowser laufen kann.

Konzept

Ein **Applet** ist ein kleines Java-Programm, das auf einer Webseite in einem Webbrowser laufen kann.

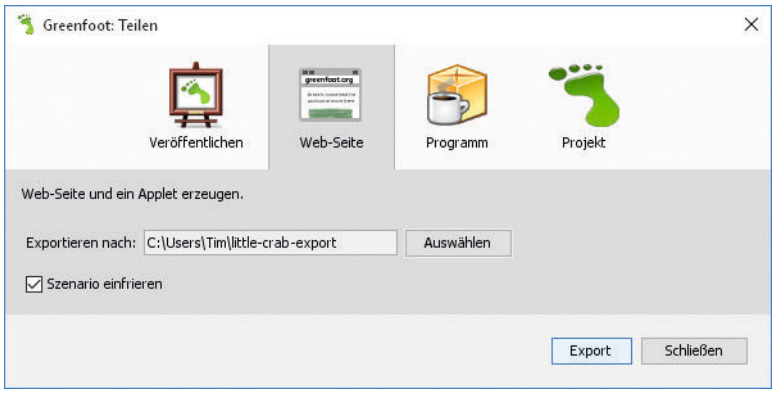


Abbildung E1.2
In eine Webseite exportieren.

Du kannst dein Szenario ausführen, indem du die erzeugte Webseite in einem Webbrowser öffnest.

Diese Webseite kannst du zunächst nur auf deinem eigenen Computer aufrufen. Wenn du möchtest, dass dein Applet auch auf einer Webseite steht, die für andere sichtbar ist, brauchst du Zugang zu einem Webserver, um es zu veröffentlichen.

Die Option SZENARIO EINFRIEREN regelt, ob die Akteure in der Welt vor dem Starten der Anwendung verschoben werden können und ob der ACT-Button und der Geschwindigkeitsregler angezeigt werden sollen. Bei einem Spiel wirst du in der Regel die Option SZENARIO EINFRIEREN aktivieren, während du für Simulationen oder eher experimentelle Szenarien diese Markierung sicher lieber entfernst, damit die Benutzer mehr experimentieren können.

E1.4 In ein Programm exportieren

Konzept

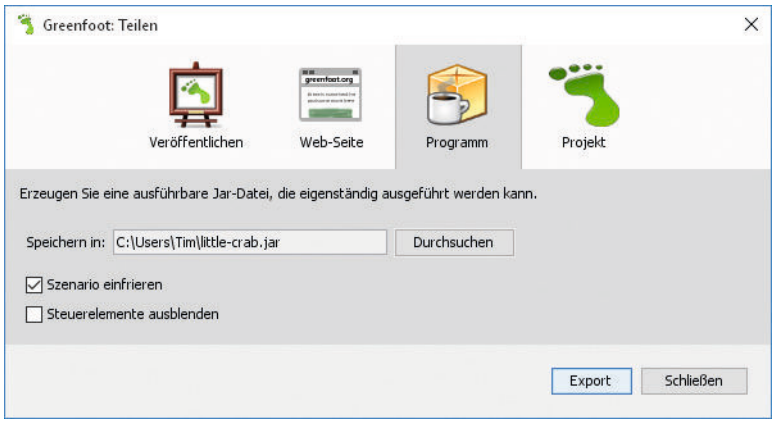
Eine **Jar-Datei** ist eine Datei mit dem Suffix *jar*, die alle Java-Klassen enthält, die zu einer Java-Anwendung gehören.

Die nächste Exportoption ist der Export in eine Anwendung. Eine Anwendung ist ein unabhängiges Programm, das der Benutzer lokal auf seinem Rechner ausführen kann.

Wähle hierfür in deinem TEILEN-Dialogfeld die Option PROGRAMM. Anschließend kannst du einen Speicherort und einen Namen für das ausführbare Szenario angeben, das du damit erzeugst (Abbildung E1.3).

Auf diese Weise erzeugst du eine *ausführbare Jar-Datei*. Diese Datei trägt den Suffix *.jar* (kurz für *Java Archive*) und kann auf vielen verschiedenen Betriebssystemen ausgeführt werden (sofern Java auf diesem Rechner installiert wurde). Zum Ausführen reicht ein Doppelklick auf die Jar-Datei.

Abbildung E1.3
Ein Szenario in ein Programm exportieren.



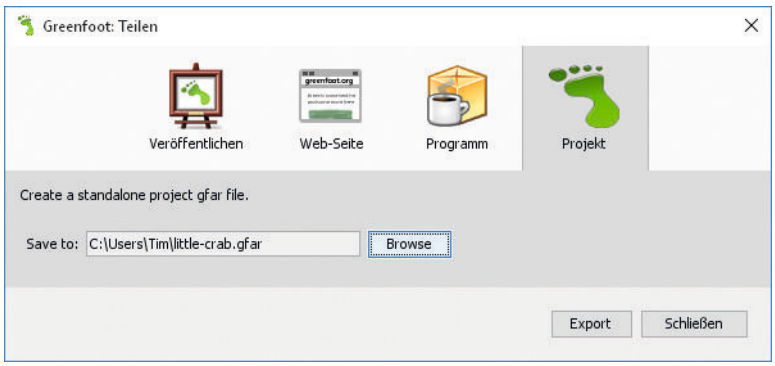
E1.5 In ein Greenfoot-Archiv exportieren

Konzept

Eine **gfar-Datei** ist eine einzelne Datei, die ein Greenfoot-Szenario enthält. Es kann mit Greenfoot geöffnet werden.

Die letzte Exportmöglichkeit ist ein Export an ein Greenfoot-Archiv, das aus einer einzelnen Datei besteht (Abbildung 1.4). Es erzeugt eine Datei mit der Endung *.gfar* (*Greenfoot Archive*). *gfar*-Dateien enthalten das gesamte Greenfoot-Szenario, einschließlich aller Dateien aus dem Szenario-Ordner. Wenn man darauf einen Doppelklick ausführt, dann klappt der normale Greenfoot-Szenario-Ordner auf, Greenfoot wird gestartet und öffnet das Szenario.

Abbildung E1.4
Export in ein Greenfoot-Archiv.



gfar-Dateien machen es leichter, Greenfoot-Szenarien auf andere Computer zu übertragen. Man kann sie zum Beispiel einfach an eine E-Mail anhängen.

Zusammenfassung der Konzepte

- Ein **Applet** ist ein kleines Java-Programm, das auf einer Webseite in einem Webbrowser laufen kann.
- Eine **Jar-Datei** ist eine Datei mit dem Suffix *jar*, die alle Java-Klassen enthält, die zu einer Java-Anwendung gehören.
- Eine **gfar-Datei** ist eine einzelne Datei, die ein Greenfoot-Szenario enthält. Sie kann mit Greenfoot geöffnet werden.

Copyright