# Spielerisches Erlernen der Programmierung mit dem Java-Hamster-Modell

Dietrich Boles

Department für Informatik, Abteilung Informationssysteme
Universität Oldenburg, Fakultät II
Escherweg 2
D-26121 Oldenburg
boles@informatik.uni-oldenburg.de

**Abstract:** Das Java-Hamster-Modell ist ein spezielles didaktisches Modell zum spielerischen Erlernen der Programmierung. Programmieranfänger lernen die grundlegenden Programmierkonzepte und den Programmentwurf kennen, indem sie so genannte Hamster-Programme entwickeln, mit denen sie virtuelle Hamster durch virtuelle Landschaften steuern und dabei bestimmte Aufgaben lösen lassen. Die Programmierkonzepte werden schrittweise eingeführt und anhand vieler Beispiele motiviert und demonstriert. Das Modell besteht aktuell aus fünf Teilen, die neben der Vermittlung der Konzepte der imperativen, objektorientierten und parallelen Programmierung in grundlegende Algorithmen und Datenstrukturen und in die objektorientierte Modellierung einführen.

## 1 Einleitung

Heftig diskutiert wird immer wieder die Frage, ob die Programmierung noch zur Informatik-Grundausbildung gehören soll. Für fast alles gibt es heutzutage bereits fertige Computer-Anwendungen. Reicht es nicht aus, den Umgang mit diesen zu erlernen, als zu wissen, wie sie intern funktionieren? Probleme einer solchen anwendungsorientierten Einführung in die Informatik sind die Kurzlebigkeit heutiger Anwendungen und die entstehende Abhängigkeit von den Produkten bzw. Anbietern. Gefragt ist daher eher Konzeptwissen (wie funktionieren Softwaresysteme, wie werden sie erstellt), das letztendlich den Transfer auf konkrete Anwendungen erlaubt.
Das Erlernen von Programmiersprachen und die Entwicklung von Programmen ist allerdings nicht trivial. Sicher gibt es Schüler, die sehr schnell damit zurechtkommen. Viele haben aber Probleme mit der Komplexität der Thematik und sind schnell frustriert. Das Hamster-Modell, das in diesem Artikel vorgestellt wird, ist genau für diese Zielgruppe entwickelt worden. Es ist ein spezielles didaktisches Modell, das Programmieranfängern einen spielerischen Zugang zu der doch eher technischen Welt der Programmierung bietet. Programmieranfänger erlernen die grundlegenden Konzepte der Programmierung sowie den Programmentwicklungsprozess, indem sie virtuelle Hamster durch eine virtuelle Landschaft steuern und dabei bestimmte Aufgaben lösen lassen.

Das Hamster-Modell ist aktuell in insgesamt fünf Teile gegliedert, die neben der Vermittlung der Konzepte der imperativen, objektorientierten und parallelen Programmierung in grundlegende Algorithmen und Datenstrukturen und in die objektorientierte

Modellierung einführen. Das Hamster-Modell reduziert die Komplexität eines Computers auf einen minimalen Satz von Befehlen, die den Hamstern erteilt werden können. Die Programmierkonzepte werden schrittweise und aufeinander aufbauend eingeführt und jeweils durch zahlreiche Beispiele demonstriert. Anhand vieler Aufgaben mit einfach zu verstehenden Aufgabenstellungen können die Programmieranfänger selbst überprüfen, ob sie den Stoff nicht nur verstanden haben, sondern auch praktisch umsetzen können. Für die Bearbeitung der Aufgaben existiert eine einfach zu bedienende Programmierumgebung, die die Erstellung und den Test von Programmen unterstützt und Programmausführungen, d. h. die Aktionen der Hamster, visuell auf dem Bildschirm darstellt. Mit diesen Eigenschaften trägt das Hamster-Modell Empfehlungen von Didaktikern, wie der Einstieg in die Programmierung erfolgen könnte, voll und ganz Rechnung [Br97, BOM99].

Prinzipiell ist das Hamster-Modell programmiersprachenunabhängig. Zum praktischen Umgang mit dem Modell wurde jedoch bewusst die Programmiersprache Java als Grundlage gewählt. Java ist eine moderne Programmiersprache, die sich in den letzten Jahren sowohl im Ausbildungsbereich als auch im industriellen Umfeld immer mehr durchgesetzt hat.

In den folgenden Abschnitten werden die einzelnen Komponenten des Java-Hamster-Modells kurz vorgestellt. Die Ausführungen orientieren sich dabei am didaktischen Aufbau des Modells. Zum Schluss erfolgt ein Vergleich mit ähnlichen Ansätzen und es werden Erfahrungen beim Einsatz des Modells geschildert. Weitere Informationen finden sich auf der Website www.java-hamster-modell.de.

## 2 Das Java-Hamster-Modell

Die Grundidee des Hamster-Modells ist ausgesprochen einfach: Programmierer müssen virtuelle Hamster durch eine vorgegebene virtuelle Landschaft steuern und sie gegebene Aufgaben lösen lassen. Dazu müssen sie entsprechende Programme - *Hamster-Programme* genannt - in der *Hamster-Sprache* entwickeln.[1]

Die Landschaft, in der die Hamster leben, wird durch eine gekachelte Ebene repräsentiert. Die Größe der Landschaft, d. h. die Anzahl der Kacheln, ist dabei flexibel. Auf einzelnen Kacheln können ein oder mehrere Körner liegen oder sie können durch Mauern blockiert sein. Im Territorium befindet sich immer mindestens ein Hamster, auch *Standard-Hamster* genannt.[2] Hamster können in vier unterschiedlichen Blickrichtungen (Nord, Süd, West, Ost) auf nicht blockierten Kacheln stehen. Sie können prinzipiell beliebig viele Körner im Maul haben. Abbildung 1 zeigt ein typisches Hamster-Territorium inklusive Legende.

---

[1] Die Hamster-Sprache ist fast deckungsgleich mit der Programmiersprache Java. Es gibt aus didaktischen Gründen lediglich einen kleinen Unterschied: Das Hauptprogramm (main-Prozedur) muss nicht innerhalb einer Klasse definiert werden.

[2] Im objektorientierten Modell können weitere Hamster erzeugt werden.

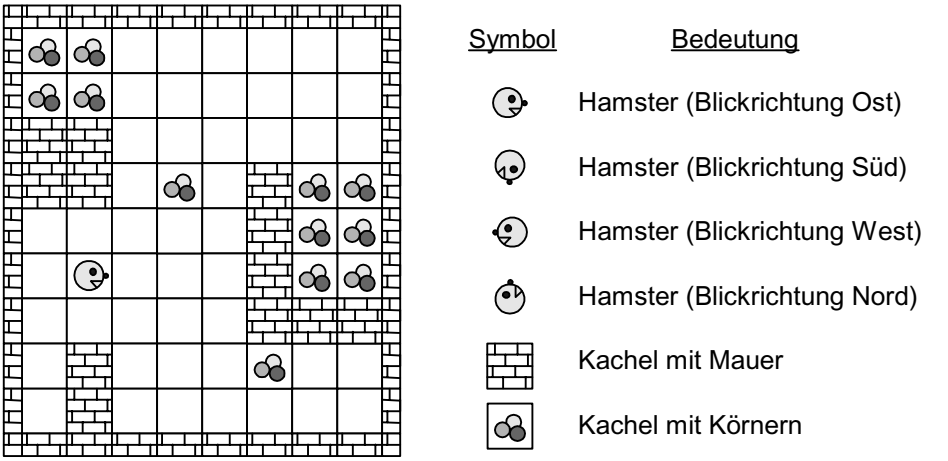


Abbildung 1: Hamster-Territorium und Legende

Mit Hilfe bestimmter Befehle kann ein Programmierer Hamster durch ein gegebenes Hamster-Territorium steuern:

- `vor`: Hüpfe eine Kachel in Blickrichtung nach vorne.
- `linksUm`: Drehe dich um 90 Grad nach links.
- `nimm`: Nimm von der Kachel, auf der du gerade stehst, ein Korn auf.
- `gib`: Lege auf der Kachel, auf der du gerade stehst, ein Korn aus deinem Maul ab.

Probleme können dabei auftreten, wenn ein Hamster durch den `vor`-Befehl gegen eine Mauer rennt oder versucht, ein Korn von einer leeren Kachel zu fressen oder ein Korn abzulegen, obwohl er gar keins im Maul hat. Um derartige Laufzeitfehler verhindern zu können, existieren die drei Testbefehle `vornFrei`, `kornDa` und `maulLeer`.

## 3 Der Hamster-Simulator

Programmieren lernt man nicht durch Lesen, sondern durch Üben. Zu diesem Zweck gibt es den so genannten *Hamster-Simulator*. Er stellt eine Reihe von Programmierwerkzeugen zur Verfügung: einen Editor zum Eingeben und Verwalten von Hamster-Programmen, einen Compiler zum Übersetzen von Hamster-Programmen, einen Territoriumsgestalter zum Gestalten und Verwalten von Hamster-Territorien, einen Interpreter zum Ausführen von Hamster-Programmen und einen Debugger zum Testen von Hamster-Programmen.

Der Hamster-Simulator wurde funktional und bedienungsmäßig bewusst an professionelle Entwicklungsumgebungen für Java (z. B. Eclipse) angelehnt, um einen späteren Umstieg auf diese zu erleichtern. Abbildung 2 zeigt den Simulator in Aktion.

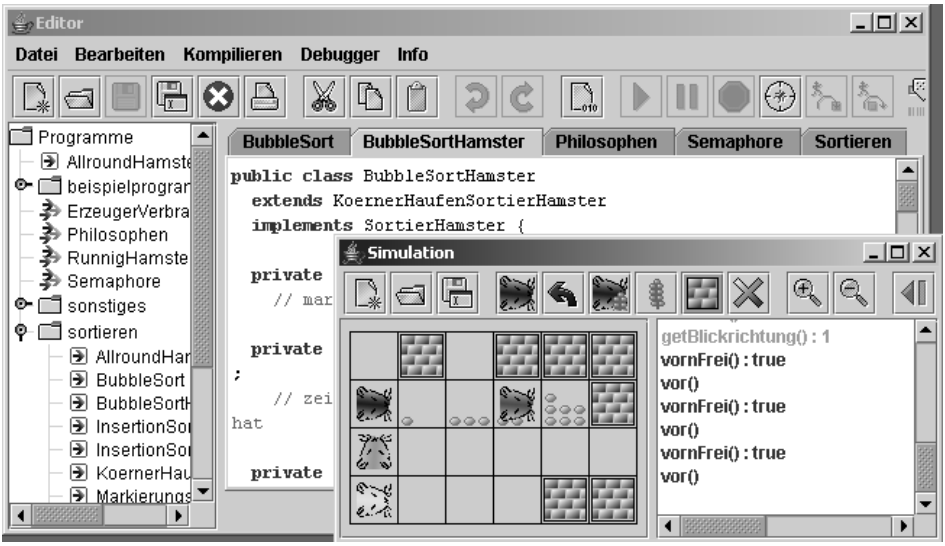


Abbildung 2: Der Hamster-Simulator

# 4 Imperative Programmierung spielend gelernt

Im ersten Teil des Java-Hamster-Modells werden die wesentlichen Konzepte der imperativen Programmierung schrittweise eingeführt [Bo02]. Dabei existiert lediglich ein einzelner Hamster.

**Anweisungen und Programme:** In der imperativen Programmierung bestehen Programme aus Anweisungen, die in der angegebenen Reihenfolge hintereinander ausgeführt werden. Das folgende Hamster-Programm löst das Problem, den Standard-Hamster in dem in Abbildung 1 skizzierten Territorium zwei Körner fressen zu lassen.

```
void main() {
  vor(); vor(); nimm(); // erstes Korn
  linksUm();
  vor(); vor(); nimm(); // zweites Korn
}
```

**Prozeduren und boolesche Funktionen:** Mit Hilfe von Prozeduren können dem Hamster neue Befehle und mit Hilfe von booleschen Funktionen neue Testbefehle beigebracht werden.

**Kontrollstrukturen:** Bis hierhin sind die Territorien zu den gestellten Hamster-Aufgaben immer fest vorgegeben. Durch Einführung von Kontrollstrukturen (`if`-, `while`- und `do`-Anweisungen) wird es nun möglich, Programme für mehrere Hamster-Terri-

torien mit bestimmten Charakteristiken zu entwickeln; zum Beispiel: Der Hamster soll in einem beliebigen rechteckigen Territorium ohne innere Mauern alle Körner sammeln.

**Programmentwurf:** Gegeben ist ein Problem, wie entwirft man ein korrektes und verständliches Programm, das dieses Problem löst? Das ist die zentrale Frage, der anschließend nachgegangen wird. Vorgestellt wird der Top-Down-Entwurf (prozedurale Zerlegung), bei dem das Problem sukzessive in weniger komplexe Teilprobleme zerlegt wird, die dann jeweils in überschaubaren Prozeduren bzw. Funktionen gelöst werden.

**Variablen und Ausdrücke:** Die meisten Lehrbücher für Programmiersprachen beginnen mit Typen, Variablen, Operationen und Ausdrücken. Viele Programmieranfänger werden aber durch die Komplexität dieses Themengebietes und seine vielen syntaktischen Feinheiten bereits abgeschreckt. Beim Java-Hamster-Modell erfolgt die Einführung erst an dieser Stelle. Über Variablen bekommt der Hamster ein Gedächtnis und er lernt rechnen. Typen sind im Java-Hamster-Modell zunächst auf die Typen `int` (Zahlen) und `boolean` (Wahrheitswerte) beschränkt.

**Verallgemeinerung des Funktionskonzeptes:** Nachdem der Typ-Begriff behandelt wurde, kann nun das Funktionskonzept verallgemeinert werden. Funktionen berechnen und liefern Werte eines bestimmten Typs. Mit Hilfe von Parametern lässt sich ihre Flexibilität erhöhen.

**Rekursion:** Durch das visuelle Feedback sind Hamster-Programme außerordentlich gut dafür geeignet, das Prinzip rekursiver Funktionen zu veranschaulichen. Die Kornsuche eines Hamsters in einem Labyrinth ist beispielsweise eine komplexe Aufgabe, die sich relativ einfach rekursiv lösen lässt.

## 5 Objektorientierte Programmierung spielend gelernt

Aufbauend auf den imperativen Sprachkonzepten werden im zweiten Teil des Java-Hamster-Modells die grundlegenden Konzepte der objektorientierten Programmierung vorgestellt [BB04]. Kennzeichen dieses objektorientierten Hamster-Modells ist, dass mehrere Hamster erzeugt und durch das Territorium gesteuert werden können, um gemeinsam gegebene Hamster-Aufgaben zu lösen.

**Hamster(objekte):** Objekte sind Instanzen so genannter *Klassen*. Im objektorientierten Hamster-Modell existiert eine vordefinierte Klasse `Hamster`, die die vier Hamster-Befehle und die drei Hamster-Testbefehle als Methoden definiert. Von der Klasse `Hamster` können Hamster(-Objekte) erzeugt werden. Objektvariablen, die Referenzen auf Hamster-Objekte speichern, entsprechen Namen für die Hamster. Im folgenden objektorientierten Hamster-Programm wird auf der Kachel in Reihe 1 und Spalte 2 mit Blickrichtung Osten und 3 Körnern im Maul ein Hamster namens *paul* erzeugt, der anschließend bis zur nächsten Mauer läuft und dabei - falls vorhanden - auf jeder Kachel ein Korn frisst.

```
void main() {
      Hamster paul = new Hamster(1, 2, Hamster.OST, 3);
      while (paul.vornFrei()) {
        paul.vor();
        if (paul.kornDa()) paul.nimm();
      }
    }
```

**Vererbung und Klassendefinition:** Im objektorientierten Hamster-Modell lässt sich der Befehlsvorrat der Hamster durch das Konzept der Vererbung erweitern, indem von der Klasse `Hamster` neue Klassen abgeleitet werden. Diese erben automatisch alle Methoden der Klasse `Hamster` und können weitere definieren.

**Arrays:** Arrays, die eigentlich zur imperativen Programmierung gehören, sind in Java als Objekte realisiert und werden daher - zusammen mit der `for`-Schleife - erst im objektorientierten Hamster-Modell eingeführt. Arrays erlauben zum einen das Anlegen von Hamster-Kolonnen und zum anderen eine Erweiterung des Hamster-Gedächtnisses, um beispielsweise ein Abbild des Hamster-Territoriums (Matrix mit Körnern und Mauern) abzuspeichern.

**Ein- und Ausgabe:** Um Hamster-Programme noch flexibler gestalten zu können, wird der Befehlsvorrat der Hamster um Lese- und Schreibbefehle sowie den Typ `String` erweitert. Damit können die Hamster nun zur Laufzeit mit dem Benutzer eines Programms kommunizieren. Im Hamster-Simulator wird die Kommunikation über Dialogboxen realisiert.

**Interfaces, Polymorphie und dynamisches Binden:** Mit Hilfe von Interfaces in Zusammenhang mit Polymorphie und dem dynamischen Binden von Methoden lassen sich generische Frameworks entwickeln. Das Prinzip und der Nutzen dieses Konzeptes lässt sich sehr schön anhand von Spiele-Frameworks demonstrieren, bei denen die Hamster mit verschiedenen Spielstrategien gegeneinander oder gegen Menschen spielen.

**Fehlerbehandlung mit Exceptions:** Anstatt mithilfe der drei vordefinierten Testbefehle die Hamster vor ihrem Tod zu bewahren, wenn sie beispielsweise gegen eine Mauer rennen, können auch Exceptions abgefangen werden, die die entsprechenden Grundbefehle liefern.

**Zugriffsrechte und Pakete:** Unter dem Motto „Die Hamster haben auch eine Privatsphäre“ werden anschließend Zugriffsrechte und das Paket-Konzept eingeführt. Zugriffsrechte regulieren den Zugriff auf Attribute und Methoden einer Klasse. Pakete dienen dazu, Klassen zu Klassenbibliotheken zusammenzufassen und anderen Programmierern zur Verfügung zu stellen.

# 6 Parallele Programmierung spielend gelernt

Ein Problem von Hamster-Programmen mit mehreren zusammenarbeitenden Hamstern ist, dass die Lösungen zum Teil künstlich wirken. Der Grund hierfür liegt darin, dass der Programmierer die Hamster explizit steuern und koordinieren muss. Im dritten Teil des Java-Hamster-Modells[3] werden die Hamster selbstständig und müssen sich selbst koordinieren. Erreicht wird dies durch Nutzung des Thread-Konzeptes von Java, mit dem parallele Programme entwickelt werden können.

**Threads:** Die vordefinierte Klasse `Hamster` ist von der Klasse `Thread` der Java-Klassenbibliothek abgeleitet und ermöglicht somit auf einfache Art und Weise die Definition und Erzeugung selbstständiger Hamster.

**Kommunikation zwischen Hamstern:** Um gemeinsam gegebene Probleme zu lösen, müssen selbstständige Hamster miteinander kommunizieren, d. h. Daten austauschen können. Da sich Java-Threads einen virtuellen Adressraum teilen, kann dies über gemeinsam zugreifbare Objekte geschehen.

**Mehrseitige Synchronisation:** Die Nutzung gemeinsamer Ressourcen durch mehrere Threads kann zu unerwarteten Zuständen bzw. Fehlern führen. Führt beispielsweise ein selbstständiger Hamster die Anweisung `if (kornDa()) nimm();` aus, kann es zu einem Fehler kommen, da bei einem Thread-Wechsel nach dem Testbefehl später evtl. gar kein Korn mehr auf der Kachel liegt. Zur Vermeidung derartiger Probleme ist es in Java möglich, Aufrufe von Methoden mittels der `synchronized`-Anweisung zu synchronisieren.

**Einseitige Synchronisation:** Wenn mehrere selbstständige Hamster gemeinsam ein Problem lösen, müssen sie manchmal während ihrer Aktionen auf andere Hamster warten, bis eine bestimmte Bedingung erfüllt ist und sie von diesen darüber informiert werden. Hierzu stellt Java allen Objekten die Methoden `wait` und `notify` zur Verfügung.

**Deadlocks:** Die Hamster könnten „verhungern", wenn sie alle auf die Erfüllung bestimmter Bedingungen warten und keiner mehr aktiv ist. Solche Situationen werden *Deadlocks* genannt. Sie gilt es natürlich zu vermeiden bzw. zu erkennen.

**Realisierung und Visualisierung klassischer Synchronisationsprobleme:** Die Konzepte der parallelen Programmierung werden anhand zahlreicher klassischer Synchronisationsprobleme, die auf die Hamster-Welt übertragen werden, demonstriert und visualisiert. Beispielsweise sitzen beim „Hamster-Philosophen-Problem" mehrere Philosophen-Hamster um einen Tisch und teilen sich mit ihren Nachbarn die Gabeln, die durch Körner repräsentiert werden. Zum Essen werden immer zwei Gabeln benötigt. Um nicht zu verhungern, gilt es, sich mit seinen Nachbarn abzustimmen.

---

[3] Dietrich Boles: Parallele Programmierung spielend gelernt mit dem Java-Hamster-Modell. Online-Buch unter www.java-hamster-modell.de

## 7 Algorithmen und Datenstrukturen spielend gelernt

Die Meinungen, inwieweit heutzutage noch klassische Algorithmen und Datenstrukturen, wie Sortier-, Such-, Baum-, Graphen-, Kompressions- oder Mustererkennungsalgorithmen, in der Programmierausbildung vorgestellt und analysiert werden sollen, gehen auseinander. Wir halten dies jedoch für sehr wichtig. Sicher gibt es heutzutage Klassenbibliotheken, in denen alle bedeutenden Algorithmen und Datenstrukturen fertig implementiert zur Verfügung gestellt werden. Aber durch das Kennenlernen der zugrunde liegenden Konzepte können Programmieranfänger Kenntnisse und Erfahrungen beim Algorithmenentwurf sammeln und diese nutzen, wenn sie selbst Algorithmen zur Lösung bestimmter Probleme entwerfen müssen.

An dieser Stelle setzen wir im vierten Teil des Java-Hamster-Modells[4] an. Wir nutzen das Modell und insbesondere seine Visualisierung, d. h. das Hamster-Territorium und die Aktionen der Hamster, um mit entsprechenden Hamster-Programmen die Konzepte und Funktionsweisen wichtiger Algorithmen und Datenstrukturen zu demonstrieren. Das Java-Hamster-Modell wird also zur so genannten *Algorithmenvisualisierung* oder *Algorithmenanimation* eingesetzt.

## 8 Objektorientierte Modellierung spielend gelernt

Zum Vermitteln bzw. Erlernen der wichtigsten Konzepte der imperativen, objektorientierten und parallelen Programmierung ist das Java-Hamster-Modell hervorragend geeignet. Bei der objektorientierten Softwareentwicklung mit dem der Programmierung vor geschalteten Phasen der Analyse und dem Entwurf stößt es jedoch an seine Grenzen. Hauptgrund hierfür ist, dass Hamster-Probleme nicht wirklich komplex genug sind, um die entsprechenden Vorgehensweisen und Konzepte an geeigneten Beispielen demonstrieren zu können.
Nichtsdestotrotz besitzt die objektorientierte Softwareentwicklung bzw. Modellierung heutzutage eine immense Bedeutung und sollte daher bereits frühzeitig neben der eigentlichen Programmierung erlernt werden, um die Gefahr einer Hacker-Mentalität[5] bei den Programmieranfängern zu minimieren.

Diesem Aspekt versuchen wir im fünften Teil des Java-Hamster-Modells[6] gerecht zu werden. Wir haben nämlich eine Problemklasse entdeckt, die nicht zu umfangreich, aber komplex genug und sehr motivierend für eine ordentliche objektorientierte Modellierung ist, nämlich die Entwicklung von Spielprogrammen, wie Schach, Reversi oder 4-Gewinnt (siehe auch [Bo03]). Spielfeld ist hierbei jeweils das Hamster-Territorium, Hamster oder Körner repräsentieren die Spielfiguren. Spieler sind entweder Menschen, denen jeweils ein Hamster zugeordnet ist, der für sie Spielzüge ausführt, oder Hamster von spe-

---

[4] Dietrich Boles: Algorithmen und Datenstrukturen spielend gelernt mit dem Java-Hamster-Modell. Online-Buch unter www.java-hamster-modell.de
[5] direkt eintippen, ohne vorher zu konzipieren
[6] Dietrich Boles: Objektorientierte Modellierung spielend gelernt mit dem Java-Hamster-Modell. Online-Buch unter www.java-hamster-modell.de

ziellen Hamster-Klassen, die intelligente Spielstrategien implementieren. Anhand dieser Problemklasse zeigen wir, wie vor der eigentlichen Programmierung eine ordentliche objektorientierte Analyse und ein objektorientierter Entwurf durchgeführt werden. Grundlage ist dabei ein allgemein gültiges Framework für 2-Personen/Hamster-Strategiespiele, das bei der Umsetzung der jeweiligen Spiele konkretisiert werden muss.

Ausgesprochen wichtig bei der objektorientierten Softwareentwicklung sind so genannte *Entwurfsmuster*. Ein Entwurfsmuster beschreibt eine in der Praxis bewährte, generische Lösung für ein häufig wiederkehrendes Entwurfsproblem und stellt damit eine wieder verwendbare Vorlage zur Problemlösung dar. Die gängigsten Entwurfsmuster (Abstrakte Fabrik, Beobachter, ...) lassen sich ganz hervorragend an dem Hamster-Spiele-Beispiel demonstrieren. Zusätzlich zeigen wir ihren Einsatz und ihren Nutzen aber auch noch an weiteren kleineren Hamster-Beispielen.

## 9 Andere Programmierlernumgebungen

Die Entwicklung von speziellen Programmiersprachen und Programmierumgebungen zum Erlernen der Programmierung reicht weit zurück. Am bekanntesten sind die Schildkröten-Graphik mit LOGO [Pa80] und „Karel the Robot“ [PRS95]. Karel war auch Vorbild für das Hamster-Modell, das seinen Ursprung bereits in den 1980er Jahren hatte [Am87]. Grundlage war damals die Programmiersprache ELAN. Auf diesen Modellen aufbauend sind in den letzten Jahren eine Reihe von Nachfolgern entwickelt worden. Einen schönen Überblick über derartige Programmierlernumgebungen enthält [RNH04]. In diesem Buch wird mit „Kara dem Marienkäfer“ auch ein alternativer Ansatz vorgestellt, bei dem Programme nicht in einer textuellen Programmiersprache verfasst, sondern graphisch als endliche Automaten erstellt werden.
Die Hauptvorteile aller dieser Modelle und Lernumgebungen sind, dass die Befehlssätze stark eingeschränkt sind und sich an bekannten (spielerischen) Dingen der realen Welt und nicht an technischen Feinheiten von Computern orientieren. Weiterhin werden die Auswirkungen der Befehle mit graphischen Mitteln auf dem Bildschirm veranschaulicht, ohne dass sich die Programmieranfänger mit der Komplexität der Programmierung graphischer Benutzungsoberflächen herumschlagen müssen.

An dieser Stelle soll und kann keine Bewertung erfolgen, welche dieser Programmierlernumgebungen die beste ist. Jede hat ihre ganz speziellen didaktischen Merkmale und Vorteile. Die wesentlichen Vorteile des Java-Hamster-Modells sind sein Umfang (imperative, objektorientierte und parallele Programmierung), der in diesem Artikel vorgestellte didaktische Aufbau sowie die ausführlichen Begleitbücher mit vielen Beispielen und Aufgaben. Als weiteren Vorteil sehen wir auch die Wahl von Java als zugrunde liegende Programmiersprache. Programmieranfänger lernen mit dem Hamster-Modell gleichzeitig diese moderne Programmiersprache kennen und neben dem Lösen von Hamster-Problemen können Programmieranfänger auch bereits frühzeitig „echte“ Probleme mit Java lösen.

## 10 Erfahrungen und Fazit

Das Java-Hamster-Modell wird seit nunmehr 9 Jahren bei der Programmierausbildung von Studienanfängern an der Universität Oldenburg eingesetzt. Durch die dabei gewonnenen Erfahrungen hat es sich inkrementell weiterentwickelt. Ob sein Einsatz den Lernerfolg der Studierenden tatsächlich verbessert hat, ist zwar kaum messbar. Die Meinungen und Rückmeldungen der Studierenden als auch vieler anderer Nutzer sind jedoch fast ausnahmslos positiv.

Dabei sei jedoch anzumerken, dass das Hamster-Modell insbesondere für solche Schüler und Studierenden gedacht ist, die sich mit dem Erlernen der Programmierung schwer tun. Denjenigen Programmieranfängern, die keine Probleme haben, wird es durch die geringe Komplexität der Aufgaben schnell langweilig. Sie wollen größere Anwendungen mit graphischen Oberflächen oder Java-Applets entwickeln. Aus diesem Grund und der Erfahrung, dass auch die erst genannte Gruppe mal echte und nicht nur Hamster-Probleme lösen will, sollte das Java-Hamster-Modell nicht ausschließlich, sondern nur motivierend und begleitend zur „richtigen“ Java-Programmierung eingesetzt werden.

## Literaturverzeichnis


[Am87] Ambros, W.: Der Hamster: Programmieren in einer Modellwelt. Metzler, 1987.

[BB04] Boles, D.; Boles, C.: Objektorientierte Programmierung spielend gelernt mit dem Java-Hamster-Modell. Teubner, 2004.

[Bo02] Boles, D.: Programmieren spielend gelernt mit dem Java-Hamster-Modell. Teubner, 2. Auflage, 2002.

[Bo03] Boles, D.: Programmierkurse für Anfänger und Fortgeschrittene. In K. Dittrich, W. König, A. Oberweis, K. Rannenberg und W. Wahlster, Hrsg., Informatik 2003, P-35 of Lecture Notes in Informatics (LNI) - Proceedings, Köllen Druck+Verlag GmBH, Bonn, 2003; S. 45 - 49.

[BOM99] du Boulay, B.; O'Shea, T.; Monk, J.: The black box inside the glass box: Presenting computing concepts to novices. International Journal of Human-Computer Studies, 51(2), 1999; S. 265 – 277.

[Br97] Brusilovsky, P. et. all.: Mini-languages: A way to learn programming principles. Education and Information Technologies, 2(1), 1997; S. 65-83.

[Pa80] Papert, S.: Mindstorms, children, computers and powerful ideas. Basic Books, New York, 1980.

[PRS95] Pattis, R.; Robert, J.; Stehlik, M.: Karel the Robot: A Gentle Introduction to the Art of Programming. Wiley, 1995.

[RNH04] Reichert,R.; Nievergelt, J.; Hartmann,W.: Programmieren mit Kara: Ein spielerischer Zugang zur Informatik. Springer, 2004.