



Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 1

Bearbeitungszeit: 14.04. bis 20.04.2008

Aufgabe 1: Java-Einstieg (Abschied von Karel J. Robot)

In diesem Semester werden wir die **KarelJIDE** nicht mehr verwenden! Stattdessen kommt als Entwicklungsumgebung **BlueJ** zum Einsatz. Diese Aufgabe wird Ihnen demonstrieren, was sich durch diesen Schritt im Quelltext verändern wird – denn nun sind nur noch reine Java-Konstrukte zugelassen. Von Robotern und den Vereinfachungen, die Karel mitgebracht hat (**task**, **loop**), verabschieden wir uns.

Auf unserer Webseite finden Sie im Übungsbereich auch eine Datei **uebung01.task**, die Sie sich bitte auch herunterladen.

Bevor wir mit **BlueJ** starten, werden wir zunächst das heruntergeladene KarelJ-Beispielprogramm so modifizieren, dass am Ende gültiger Java-Code steht. Dafür verwenden wir zunächst die Konsole und einen einfachen Editor, damit Sie den Unterschied zu einer Programmierumgebung kennen lernen.

- a) Öffnen Sie ein Terminal- bzw. Konsolen-Fenster.
- b) Geben Sie dort bitte den Befehl **gedit &** ein (statt früher **KarelJIDE**). Das **&** dient dazu, dass wir im Terminal noch weiterarbeiten können. Machen Sie sich zuerst einmal mit dem Editor vertraut.
- c) Öffnen Sie bitte das heruntergeladene Beispielprogramm **uebung01.task**.

Damit unser ehemaliges KarelJ-Programm zu einer echten Java-Applikation wird, sind folgende Schritte nötig:

- d) Speichern Sie nun die Datei (ohne etwas zu ändern!) mittels **File → Save As...** unter dem neuen Namen **Reverse.java** – achten Sie auf das große „R“! Das sollten Sie sich auch schon mal merken: in Java haben Dateien im Allgemeinen den gleichen Namen wie die Klasse, die sie beinhalten, ergänzt um die Erweiterung „.java“.
- e) Entfernen Sie in der Klassendefinition das **„extends Robot“** – die Karel-Werke werden für den weiteren Fortgang der Vorlesung keine Bedeutung mehr spielen.
- f) Entfernen Sie im Konstruktor die Zeile **„super(1, 1, 0, East)“**.
- g) Das Hauptprogramm darf nicht mehr außerhalb einer Klasse stehen. Markieren Sie es (von **„task {“** bis zum schließenden **„}“**), drücken Sie zum Ausschneiden **STRG+X** und fügen Sie es dann mit **STRG+V** nach der letzten Methode (die Stelle ist markiert) in der Klasse ein.
- h) Außerdem heißt das Hauptprogramm nicht mehr **„task“**, sondern **„public static void main(String[] args)“** – ändern Sie das also entsprechend. Statt vom „Hauptprogramm“ oder „Task“ reden wir daher ab jetzt auch von der **„main-Methode“**.

Was die Schlüsselwörter **public** und **static** bedeuten, erfahren Sie in der Vorlesung und/oder in einer späteren Übung.

- i) Um von KarelJ ganz Abschied zu nehmen, nutzen Sie in der **main**-Methode bitte den Kennzeichner **r** anstelle von **karel**.
- j) Speichern Sie das Programm mittels **File** → **Save**.

Nun haben wir unser Programm in ein echtes Java-Programm verwandelt. Natürlich wollen wir sehen, ob es trotzdem noch funktioniert. Einen „Execute“-Button wie in der **KarelJIDE** gibt es nicht, also müssen wir das Programm auf eine andere Art ausführen. Das geschieht in zwei Schritten.

Zuerst müssen wir das Programm in „Java Byte Code“ übersetzen, eine Form, die der Computer bzw. die „Java Virtual Machine“ versteht. Dieser Vorgang nennt sich „kompilieren“ – vielleicht erinnern Sie sich ja an die **KarelJIDE**-Statusmeldung „**## Compile program ##**“.

- k) Geben Sie zum Kompilieren im Terminal **javac Reverse.java** (mit „.java“!) ein.

Wenn Ihr Programm Fehler enthält, bekommen Sie genau die gleichen Fehlermeldungen, die früher die **KarelJIDE** angezeigt hat. Sie müssen dann Ihr Programm entsprechend ändern, speichern und erneut kompilieren.

Wenn gar keine Ausgabe (also insbesondere auch keine Fehlermeldung) erfolgt, ist das Programm erfolgreich kompiliert. Sie sollten nun mit dem Befehl „ls“ nicht nur die Datei **Reverse.java** finden können, sondern auch eine Datei **Reverse.class**.

- l) Diese führen Sie nun aus, indem Sie **java Reverse** eingeben (ohne „.class“!).

Im Terminal sollte nun der String „**Testausgabe**“ erscheinen.

Aufgabe 2: Die Programmierumgebung BlueJ

Wir wollen nun den einfachen Editor und das Terminal hinter uns lassen und uns der Entwicklungsumgebung **BlueJ** zuwenden. Bitte schließen Sie den Editor (gedit) aber noch nicht!

- a) Starten Sie BlueJ mit dem Kommando **bluej**.
- b) Erstellen Sie ein neues Projekt namens **uebung01** (**Project** → **New Project...**).
- c) Erstellen Sie über **New Class...** eine neue Klasse mit dem Namen **Reverse**. Mit einem Doppelklick auf das neu entstandene, schraffierte Symbol kommen Sie in die Quellcodeansicht. Löschen Sie nun bitte den gesamten vorgegebenen Quellcode, wir werden stattdessen mit dem Code aus Aufgabe 1 weiterarbeiten. Kopieren Sie also bitte aus dem noch geöffneten Editor-Fenster den kompletten Quellcode aus Aufgabe 1 in die leere Klasse **Reverse** in **BlueJ**.
- d) Speichern Sie den Quelltext ab und kompilieren Sie ihn (entweder per Rechtsklick auf das Klassensymbol und dann **Compile** oder über den Button **Compile** in der linken Symbolleiste von **BlueJ**). Die Schraffierung wird verschwinden und Sie haben eine kompilierte und ausführbare Klasse vorliegen.
- e) Mit einem Rechtsklick auf die Klasse können Sie nun die **main**-Methode direkt aufrufen. In einem neuen Fenster sollte nun der String „**Testausgabe**“ erscheinen.
- f) Bevor wir nun die eigentliche Funktionalität der Klasse implementieren, löschen Sie bitte aus der **main**-Methode die Zeile, die für den „**Testausgabe**“-String verantwortlich ist

Im Folgenden sollen Sie die Klasse nun so erweitern, dass beim Erzeugen eines Objekts der Klasse ein Array angelegt wird, das dann mit den Methoden **print()** und **reverse()** auf dem Bildschirm ausgegeben bzw. umgekehrt (gespiegelt) werden kann.

- a) Erweitern Sie den Konstruktor folgendermaßen:
 - Wenn der übergebene Parameter **int n** kleiner als 2 ist, soll **n** auf 2 geändert werden.
 - Das am Anfang der Klasse deklarierte Array **a** soll mit der Länge **n** erzeugt werden.
 - Erzeugen Sie einen Zufallsgenerator: legen Sie dazu ein Objekt **r** der Klasse **java.util.Random** an (der Konstruktor erwartet keine Parameter).
 - Benutzen Sie eine **for**-Schleife, um jedem Wert (**0** bis **n-1**) des Arrays **a** einen zufälligen Wert zwischen 1 und 99 (**r.nextInt(99) + 1**) zuzuweisen.
- b) Die Methode **print()** soll das Array in einer Zeile auf dem Bildschirm ausgeben. Geben Sie dazu zuerst das erste Element aus, dann in einer Schleife das zweite bis zum letzten (geben Sie aber vor jedem dieser Elemente ein Leerzeichen aus) und benutzen Sie dann **System.out.println()**, um in die nächste Zeile zu wechseln.
- c) Nun zum Kernstück der Klasse: in der Methode **reverse()** soll das Array umgekehrt bzw. gespiegelt werden. Vertauschen Sie mit Hilfe einer **for**-Schleife das erste Element mit dem letzten, das zweite mit dem vorletzten und so weiter. Überlegen Sie, bis wohin der Schleifenzähler laufen muss! Für die Vertauschung der Elemente benötigen Sie eine Hilfsvariable, legen Sie diese am besten nicht erst in der Schleife an, sondern davor.

Beispiel:

Array vor Umkehrung: **42 23 1 99 17 50 66 7 3 81**

Array nach Umkehrung: **81 3 7 66 50 17 99 1 23 42**

Aufgabe 3: Javadoc

In dieser Aufgabe sollen Sie sich nun mit dem in der Vorlesung vorgestellten **Javadoc**-Konzept auseinandersetzen. Auf der nächsten Seite dieses Aufgabenblattes sehen Sie eine beispielhafte Interfacebeschreibung der Musterlösung.

- a) Eignen Sie sich das notwendige Wissen (Variablen und Syntax von **Javadoc**) an, das Sie benötigen, um die Interfacebeschreibung (siehe Hinweis) Ihres Programms genauso aussehen zu lassen – und kommentieren Sie Ihr Programm dann entsprechend!

Sie können sich beispielsweise auf der Webseite <http://de.wikipedia.org/wiki/Javadoc> über den Javadoc-Standard informieren.

Hinweis: In der Editoransicht von **BlueJ** haben Sie die Möglichkeit zwischen den Ansichten **Implementation/Source Code** und **Interface/Documentation** zu wechseln. Im Interface/Documentation-Modus sehen Sie, wie sich Ihre Veränderung im Quellcode auf die Dokumentation Ihrer Klasse auswirkt.

Im Programmierprojekt wird Wert auf eine Javadoc-konforme Kommentierung gelegt werden!

```
public class Reverse extends java.lang.Object
```

Umkehren eines Arrays

Version:

2008-04-12

Author:

Owyn

Field Summary

(package private) int[]	a	Enthält das Array
-------------------------	-------------------	-------------------

Constructor Summary

Reverse (int n)	Erzeugt ein Array der Länge n, gefüllt mit zufälligen Zahlen
---------------------------------	--

Method Summary

static void	main (java.lang.String[] args)	main-Methode (ehemals "task")
(package private) void	print ()	Gibt das Array auf dem Bildschirm aus
(package private) void	reverse ()	Spiegelt das Array

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

a
int[] **a**
Enthält das Array

Constructor Detail

Reverse
public **Reverse**(int n)
Erzeugt ein Array der Länge n, gefüllt mit zufälligen Zahlen
Parameters:
n - Anzahl der Arrayelemente, wird auf 2 gesetzt, falls kleiner

Method Detail

main
public static void **main**(java.lang.String[] args)
main-Methode (ehemals "task")

print
void **print**()
Gibt das Array auf dem Bildschirm aus

reverse
void **reverse**()
Spiegelt das Array