technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWilngs und WiMas

Wintersemester 2018/19

Übungsblatt 8

Besprechung:
17.–21.12.2018
(KW 51)

Hinweise zum Tower-Defense-Spiel

Anstelle der ergänzenden Aufgaben können Sie ab diesem Übungsblatt auch die Aufgaben zu unserem Tower-Defense-Spiel bearbeiten. Diese Aufgaben finden Sie auf der Veranstaltungswebsite `https://tiny.cc/eini1819`.

Vorbereitende Aufgaben

Aufgabe 8.1: Manuelle Sortierung

Gegeben ist folgendes Array:

7	12	3	2	21	9

In der Vorlesung (Kapitel 5.1) haben Sie den Algorithmus **Selectionsort** zum Sortieren von Arrays kennengelernt. Sortieren Sie das Array nun aufsteigend nach diesem Algorithmus verfahrend. Notieren Sie dabei jede Tauschoperation:

Aufgabe 8.2: Debugging

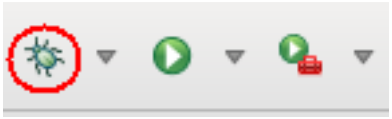
Lesen und verstehen Sie den Text über Debugging auf der nächsten Seite.

Debugging

Mit Hilfe eines sog. Debuggers können Sie ein Programm zur Laufzeit analysieren. Eclipse bietet eine sehr einfache Möglichkeit, dies umzusetzen.

Sie können mit Hilfe von sog. **Breakpoints** Zeilen im Code angeben, **vor** deren Zeilenausführung das Programm anhalten soll. Dazu müssen Sie links auf den Zeilenrand Ihres Quellcodes doppelklicken, so dass ein kleiner, blauer Punkt erscheint. Sie können den Breakpoint auf die selbe Art und Weise wieder entfernen.

Der Eclipse-Debugger kann dann durch das Klicken auf den kleinen, grünen Käfer neben dem Run-Button gestartet werden.



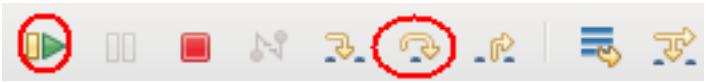
Debug-Button

Wenn Sie das Programm über den Debugger starten, wird Eclipse seine Ansicht wechseln wollen, sobald ein Breakpoint erreicht wird. Akzeptieren Sie diesen Vorschlag.

Ihr Quellcode wird in den unteren Teil des Bildschirms verschoben und in der oberen Hälfte des Bildschirms erscheinen Werkzeuge zum Analysieren Ihres Programmes. Zudem pausiert die Ausführung des Programmes am erreichten Breakpoint. Sie können sich in der oberen, rechten Tabelle alle lokalen Variablen mit ihren Werten ansehen.

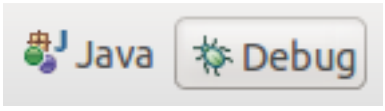
Im oberen Bereich ihrer Arbeitsumgebung können Sie durch das Klicken des grünen Pfeiles in Form eines „Abspielen“-Symboles den Programmfluss bis zum Erreichen des nächsten Breakpoints fortsetzen.

Durch Klicken des sich rechts daneben befindenden Buttons mit dem Tooltip `Step Over(F6)` können Sie das Programm genau eine Anweisung durchführen lassen, um so schrittweise den Ablauf des Programmes zu beobachten.



zum nächsten Breakpoint (links), schrittweise Ausführung (rechts)

Wenn Sie zurück zur normalen Editor-Ansicht wechseln wollen, gibt es in der oberen, rechten Ecke des Bildschirms einen dafür vorgesehenen Reiter.



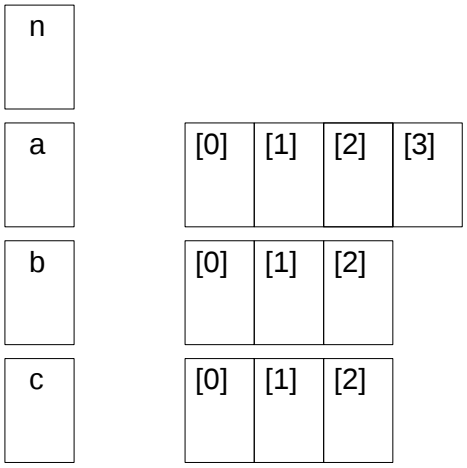
Umschalten der Editor-Ansicht

Präsenzaufgaben

Aufgabe 8.3: Speicherverwaltung für Felder

In dieser Aufgabe wollen wir uns mit der Speicherverwaltung im Zusammenhang mit Arrays beschäftigen. Betrachten Sie das untenstehende Programmfragment. Es deklariert und initialisiert eine ganzzahlige Variable **n** und drei Felder von ganzen Zahlen **a**, **b** und **c**. Gehen Sie den Programmtext zeilenweise durch und tragen Sie in das nebenstehende Diagramm ein, wie die einzelnen Speicherzellen im Laufe des Programms belegt und geändert werden.

```
1  int n;
2  int[] a;
3  a = new int[4];
4  for (int i = 0; i < a.length; i++) {
5      a[i] = i;
6  }
7  n = a.length - 1;
8  int[] b;
9  b = new int[n];
10 for (int i = 0; i < b.length; i++) {
11     b[i] = a[i] + 10;
12 }
13 int[] c = new int[b.length];
14 c = b;
```



Aufgabe 8.4: Arrays als Parameter

In dieser Aufgabe wollen wir lernen, Arrays als Parameter zu nutzen. Legen Sie dazu eine Klasse **UseArrays** an.

a) Schreiben Sie eine Funktion **printArray**, die den Inhalt eines übergebenen **int**-Array ausgibt.

b) Implementieren Sie eine Funktion mit dem Namen **swap**, die ein **int**-Array, sowie zwei Indizes entgegen nimmt und die Elemente im Array an den entsprechenden Indizes miteinander tauscht. Die Funktion besitzt aber keine Rückgabe.

c) Verwenden Sie anschließend die Funktion **swap** in einer **main**-Methode, um das erste mit dem letzten Element eines Arrays zu tauschen. Geben Sie das Array vor und nach dem Aufruf von **swap** mit der Funktion **printArray** aus.

d) Was fällt Ihnen auf, wenn Sie das Ergebnis mit den Erkenntnissen aus Aufgabe 6.5 vergleichen?

Aufgabe 8.5: Bibliothek erstellen

In dieser Aufgabe wollen wir eine Klasse programmieren, mit der Sie Arrays mit verschiedenen Eigenschaften erzeugen können. Erstellen Sie eine neue Klasse namens **ArrayGenerator** im Paket **blatt08**. Diese Klasse soll keine **main**-Methode erhalten! Implementieren Sie die folgenden beiden Funktionen in dieser Klasse.

a) Die Funktionen **generateAscendingArray** soll ein **int**-Array der Länge **length** erzeugen und es aufsteigend mit den Werten von 0 bis **length**-1 befüllen.

b) Die Funktion **generateRandomArray** soll ein zufällig gefülltes Array mit der Länge **length** erzeugen.

   Dafür müssen Sie die Klasse **Random** aus dem Paket **java.util** importieren. Verwenden Sie dafür: `import java.util.Random;` noch vor der Definition der Klasse. Anschließend können Sie mit `Random randomNumbers = new Random();` einen neuen Zufallsgenerator erzeugen und danach mit `randomNumbers.nextInt(n)` eine zufällige Zahl aus dem halboffenen Intervall $[0, n)$ erhalten. Verwenden Sie als n die Länge des erzeugten Arrays.

Aufgabe 8.6: Bibliothek verwenden

In dieser Aufgabe wollen wir nun die Funktionen der soeben definierten Klasse verwenden. Eine Klasse ohne **main**-Methode, wie Ihre **ArrayGenerator**-Klasse, nennt man auch **Bibliothek**. Bibliotheken besitzen meistens sehr hilfreiche Funktionen.

a) Verwenden Sie in der **main**-Methode der Klasse **UseArrays** die soeben von Ihnen definierten Funktionen, um ein aufsteigendes und ein zufälliges Array der Länge 32 auszugeben.

b) Schreiben Sie die Funktion **arrayMin**, die das Minimum eines übergebenen **int**-Arrays zurückgibt. Rufen Sie diese Funktion für die generierten Arrays auf.

c) Schreiben Sie die Funktion **average**, die den Mittelwert eines übergebenen **int**-Arrays berechnet und als **double** zurückgibt. Rufen Sie auch diese Funktion für die generierten Arrays auf.