Prof. J. Wolff von Gudenberg
Prof. R. Kolla
Prof. F. Puppe
Prof. D. Seipel

Universität Würzburg
Lehrstuhl für Informatik II
Prof. Dr. J. Wolff von Gudenberg
Am Hubland, 97074 Würzburg

Dipl.-Inform. Gregor Fischer <fischer@informatik.uni-wuerzburg.de>

Einführung in die Informatik

für Hörer alle Fakultäten

(WS-03/04)

Übungsblatt 5

Ausgabe: Do. 2003-11-20

Abgabe: Do. 2003-11-27

Besprechung: Di./Do. 2003-12-02/04

Beachten Sie auch für dieses Aufgabenblatt die Hinweise auf Übungsblatt 4 zur Abgabe der Lösungen mit Praktomat unter:

https://jop.informatik.uni-wuerzburg.de/infohaf/praktomat.cgi

Aufgabe 5.1: StringBufferTools (2 Punkte)

Schreiben Sie ein Programm mit der Hauptklasse `StringBufferTools` mit folgenden Methoden:

- `public static void replaceFirst(StringBuffer sb, String oldString, String newString)`

  Die Methode `replaceFirst` ersetzt **das erste Auftreten** des als Parameter übergebenen Strings `oldString` durch den zweiten (als Parameter übergebenen) String `newString` im StringBuffer `sb`.

- `public static void replaceFirstFrom(StringBuffer sb, int fromIndex, String oldString, String newString)`

  Die Methode `replaceFirstFrom` ersetzt **das erste Auftreten ab der Position `fromIndex`** des als Parameter übergebenen Strings `oldString` durch den zweiten (als Parameter übergebenen) String `newString` im StringBuffer `sb`.

- `public static void replaceAll(StringBuffer sb, String oldString, String newString)`

  Die Methode `replaceAll` ersetzt **alle Auftreten** des als Parameter übergebenen Strings `oldString` durch den zweiten (als Parameter übergebenen) String `newString` im StringBuffer `sb`.

- `public static void main(String[] args)`

  Eine Main-Methode, die die Funktionalität der anderen Methoden demonstriert. Dies wird nicht automatisch geprüft, Sie sollten jedoch trotzdem eine sinnvolle Demonstration implementieren, da das bei der manuellen Abschlusskorrektur berücksichtigt wird. Darüber hinaus dient es für Sie selbst zur Überprüfung Ihrer Methoden.

Hinweis: Verwenden Sie Methoden aus den Klassen `StringBuffer` und evtl. `String`. Die Dokumentation dazu finden Sie unter:

StringBuffer:	http://java.sun.com/j2se/1.4.2/docs/api/java/lang/StringBuffer.html
String:	http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html
Java Doku. allgemein:	http://java.sun.com/j2se/1.4.2/docs/api/

Beispiel: Nachfolgend finden Sie einige Beispielaufrufe. Dabei wird der StringBuffer `sb` immer mit dem Wert "banana" angenommen. Die rechte Seite zeigt, welchen Wert der StringBuffer anschliessend haben muss.

Aufruf	Wert von *sb* nach dem Aufruf
replaceFirst(sb, "an", "at")	batana
replaceFirstFrom(sb, 3, "an", "at")	banata
replaceAll(sb, "an", "at")	batata

Aufgabe 5.2: ArrayTools (4 Punkte)

Schreiben Sie ein Programm mit der Hauptklasse `ArrayTools` mit folgenden Methoden:

- `public static double max(double[] doubleArray)`

  Diese Methode `max` bestimmt das Maximum der Zahlen im Array `doubleArray` und liefert den gefundenen (maximalen) Wert zurück.

- `public static double max(double[] doubleArray, int start, int end)`

  Diese Methode `max` bestimmt das Maximum der Zahlen des Arrays `doubleArray` im Bereich von `start` (inklusiv) bis `end` (inklusiv) und liefert den gefundenen (maximalen) Wert zurück.

  D.h für das Array d (`{ 5, 3, 4, 2, 7, 9 }`) liefert `max(d, 1, 4)` den Wert `7`.

- `public static int maxPosition(double[] doubleArray)`

  Diese Methode `maxPosition` bestimmt die Position des Elements des Arrays `doubleArray` mit dem maximalen Wert und liefert die gefundene Position zurück.

- `public static int maxPosition(double[] doubleArray, int start, int end)`

  Diese Methode `maxPosition` bestimmt die Position des Elements des Arrays `doubleArray` im Bereich von `start` (inklusiv) bis `end` (inklusiv) mit dem maximalen Wert und liefert die gefundene Position zurück.

- `public static void sort(double[] doubleArray)`

  Die Methode `sort` sortiert das Array `doubleArray`. Dabei soll zunächst das Maximum des Arrays gefunden werden und dieses dann mit dem letzten Element des Array vertauscht werden. Anschliessend sortiert man das restliche Array ohne das letzte Element auf gleich Weise.

- `public static int sum(int[] intArray)`

  Die Methode `sum` bestimmt die Summe über alle Werte des Arrays `intArray`.

- `public static double average(int[] intArray)`

  Die Methode `average` bestimmt das arithmetische Mittel (Durchschnittswert) aller Werte des Arrays `intArray`.

- `public static void main(String[] args)`

  Eine Main-Methode, die die Funktionalität der anderen Methoden demonstriert. Dies wird nicht automatisch geprüft, Sie sollten jedoch trotzdem eine sinnvolle Demonstration implementieren, da das bei der manuellen Abschlusskorrektur berücksichtigt wird. Darüber hinaus dient es für Sie selbst zur Überprüfung Ihrer Methoden.

Aufgabe 5.3: Wurzeln schlagen (4 Punkte)

Schreiben Sie ein Programm mit der Hauptklasse `IterativeWurzel` mit folgenden Methoden:

- `public static double wurzel(double x)`

  Die Methode `wurzel` bestimmt die Quadratwurzel von `x` nach untem angegebenen Verfahren und liefert das Ergebnis der Berechnung zurück.

- `public static void main(String[] args)`

  Eine Main-Methode, die die Funktionalität der anderen Methoden demonstriert. Dies wird nicht automatisch geprüft, Sie sollten jedoch trotzdem eine sinnvolle Demonstration implementieren, da das bei der manuellen Abschlusskorrektur berücksichtigt wird. Darüber hinaus dient es für Sie selbst zur Überprüfung Ihrer Methoden.

Iterative Wurzelberechnung durch Intervallschachtelung

Um die Quadratwurzel von `x` zu berechnen kann man sich folgender Idee bedienen: Statt die Wurzel selbst zu berechnen, kann man auch die Nullstelle der Funktion $f(z) = z^2 - x$ berechnen. Der zur Nullstelle gehörige z-Wert ist dann der Wert der gesuchten Quadratwurzel, da aus $f(z) = z^2 - x = 0$ folgt: $z = \sqrt{(x)}$.

Die Nullstelle findet man nun mittels Intervallschachtelung:

- Gegeben ist ein Intervall `[a,b]` in dem die gesuchte Nullstelle liegt. Für den ersten Durchlauf sind sinnvolle Startwerte zu wählen.
- Da die betrachtete Funktion `f` monoton wachsend ist, können wir davon ausgehen, dass gilt: `f(a) < 0` und `f(b) > 0`
- Man halbiert nun das Intervall an der Stelle `m = (a+b)/2`.
- Ist `f(m) > 0`, so liegt die Nullstelle in der linken Hälfte des Intervalls (`[a,m]`) und man wiederholt das Verfahren für dieses Intervall (d.h. mit `a` und `m` als Intervallgrenzen).
  Ist `f(m) < 0`, so liegt die Nullstelle in der rechten Hälfte (`[m,b]`), und man wiederholt das Verfahren mit diesem Intervall.
  Für den (unwahrscheinlichen) Fall `f(m) = 0` hat man die Nullstelle gefunden und kann das Ergebnis zurückliefern.
- Da das Ergebnis im Intervall liegen muss, kann man den maximalen Fehler der Berechnung durch die Breite des Intervalls abschätzen. Ist das Intervall z.B. nur noch 0.01 gross, so kann der Fehler maximal 0.01 betragen. Wenn man den Mittelwert des Intervalls als Ergebniss wählt sogar nur 0.005.
- Man wiederholt das Verfahren also so lange, bis das Intervall klein genug für die gewünschte Brechnungsgenauigkeit ist.

Stellen Sie Ihr Programm so ein, dass es das Ergebnis bis auf mindestens 4 Nachkommstellen genau berechnet, d.h. mit einem maximalen Fehler von 0.00005.