

Übung 2

Wiederholung JAVA, Arrays und Strings

1. Ermitteln Sie den Momentan-Speicherbedarf in Bytes für ein „Schachprogramm“, das alle möglichen Schachfiguren-Positionen (6 verschiedene Figuren je Spieler bei 2 Spielern) auf einem Schachbrett mit 8x8 Feldern für eine Schachpartie mit maximal 100 Zügen speichern kann. Dabei werden die Schachregeln und die Tatsache, dass es jede Figur nur in einer begrenzten Anzahl in einem Spiel geben kann, und pro Feld nur maximal eine Figur aufgestellt werden kann, außer acht gelassen.

Hinweis 1: Ob eine bestimmte Figur auf einem Feld steht, könnte durch ein Boolean (vereinfacht angenommen, ein Bit) festgelegt werden.

Hinweis 2: Ein Byte besteht aus 8 Bit und kann daher Werte zwischen 0 und (2^8-1) annehmen.

2. Schreiben Sie in JAVA eine Funktion (!), die ein Polynom n -ten Grades der Form

$$P(x) = \sum_{i=0}^n a_i x^i = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3 + \dots + a_n \cdot x^n$$

an der Stelle x berechnet, wobei die Koeffizienten $a_0 \dots a_n$ als **double** []-Array und die Variable x als **double** an die Funktion übergeben werden. Das Ergebnis soll ebenfalls vom Typ **double** sein.

Hinweis 1: Erstellen Sie zunächst die **Signatur** der Funktion, d.h. den Funktionskopf, überlegen Sie sich geeignete Namen für die Variablen und Arrays und verwenden Sie diese dann im Funktionsrumpf.

Hinweis 2: Wenn Sie nicht wissen, was ein „Polynom“ ist (Mathematik...), können Sie es hier nachschlagen: <http://de.wikipedia.org/wiki/Polynom>.

Es ist aber für die Lösung der Aufgabe unerheblich.

3. Schreiben Sie ein JAVA-Programm, das
 - (a) die als Kommandozeilenparameter **args[0]...args[args.length-1]** übergebenen Zahlen von Datentyp **String** nach Datentyp **int** konvertiert,
 - (b) in ein **int** []-Array der Größe **args.length** speichert,
 - (c) das Array aufsteigend nach Größe der Zahlen sortiert,
 - (d) die Array-Elemente der Reihe nach am Bildschirm ausgibt.

Tipp: In der Beispielklasse **Eingabe.java** finden Sie die Konvertierungsfunktionen zur Zahlenumwandlung.

4. **String**-Vergleiche

- (a) Programmieren Sie folgende Code-Sequenz in einem Java-Testprogramm, und überprüfen Sie sich, ob und warum es zu unterschiedlichen Ergebnissen in den Zeilen 5 und 6 kommt.

1	String s1 = new String("123");
2	String s2 = new String("123");
3	String s3 = "123";
4	String s4 = "123";
5	System.out.println(s1 == s2);
6	System.out.println(s3 == s4);

- (b) Wie kann man **Strings** zuverlässig auf Gleichheit des Inhalts überprüfen?