hochschule mannheim

Fakultät für Informatik
Prof. Dr. Peter Knauber

Bachelor-Studiengang Informatik
Übungen zur Vorlesung “Grundlagen der Informatik” (GDI), WS 2009/2010

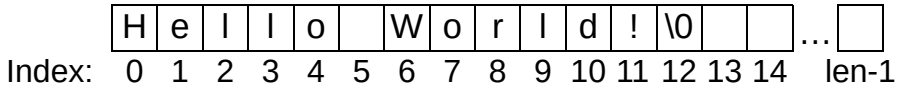
Übungsblatt 7: Methoden, Arrays Ausgabe am: 25.11.2009
2.12.2009

Aufgabe 1 **4 + 4 + 10 = 18 Punkte**

Strings (wie sie in Java verfügbar sind) können als *Array of char* verwaltet werden. Das Array bekommt eine feste Länge (zum Beispiel 256 Zeichen), das Ende des gerade darin gespeicherten Strings ist durch das Zeichen ’\0’ gekennzeichnet. (So werden Strings zum Beispiel in der Programmiersprache C repräsentiert.)

Beispiel

Der String “Hello World!” wird in einem Array of char der Länge len wie folgt gespeichert



Das Null-Byte ’0’ an der Index-Position 12 zeigt an, dass der String keine weiteren (sinnvollen) Zeichen mehr enthält, sondern 12 Zeichen lang ist; unabhängig von der tatsächlichen Länge des Arrays (das viel länger sein kann).

Zur Aufgabe:

Gegeben ist das folgende Interface (den Quelltext mit den Erklärungen zu der Methode finden Sie auch im Web und im Wiki):

```
public interface StringAsCharArray {

  boolean stricmp( char[] s1, char[] s2 );

}
```

Implementieren Sie die beschriebenen Operationen in Java, indem Sie Character-Arrays anstelle der in Java verfügbaren Strings benutzen. Sie sollen wie in C auch durch ein Null-Zeichen abgeschlossen werden:

a) Erstellen Sie eine Klasse *IterativeStringImplementation*, welche die Methode dieses Interface’ **iterativ** implementiert. Rekursion ist (auch in Hilfsmethoden) **nicht** erlaubt.

b) Erstellen Sie eine Klasse *RecursiveStringImplementation*, welche die Methode dieses Interface’ **rekursiv** implementiert. Schleifen sind (auch in Hilfsmethoden) **nicht** erlaubt.

c) Implementieren Sie eine geeignete Umgebung, in der die Methode im Rahmen des Testats getestet werden kann. Es soll möglich sein, verschiedene “Strings” (am Besten zeichenweise) zur Laufzeit Ihres Programms einzugeben, mit denen dann die Methoden Ihrer String-Implementierungen aufgerufen werden; deren Auswirkungen und das Ergebnis sol-

len dann ausgegeben werden.
Denken Sie beim Testen an Sonderfälle!

Bemerkungen

- Es sind keine Methoden der Java-Bibliotheken erlaubt außer den in der Vorlesung behandelten.
- Die Methoden dürfen intern keine Java-Strings und keine zusätzlichen Arrays (sondern nur die als Parameter übergebenen Character-Arrays) anlegen oder verwenden.
- Keine der implementierenden Methoden gibt etwas auf die Konsole aus, das ist ausschließlich Aufgabe der Testumgebung (Teilaufgabe c).
- Sie dürfen eigene Hilfsmethoden je nach Bedarf verwenden.
- Dokumentieren Sie explizit in jedem(!) Fall, was bei "fehlerhaften" Eingaben passieren wird.

Aufgabe 2 **4 + 6 + 6 + 6 = 22 Punkte**

a) Erstellen Sie ein Interface *PrimzahlBerechnung* mit einer einzigen Methode *calculatePrime*, die eine ganze Zahl n als Parameter erhält und ein passend dimensioniertes(!) Array mit ganzen Zahlen als Ergebnis liefert, das alle Primzahlen enthält, die kleiner oder gleich der Obergrenze n sind. Schreiben Sie im Interface einen entsprechenden Kommentar zu dieser Methode.

b) Korrigieren Sie bei Bedarf Ihre Klasse *PrimzahlenMitSchleifen* von Übungsblatt 3, Aufgabe 1.
Erstellen Sie eine Kopie im lokalen Paket und ändern Sie die Klasse so, dass sie das Interface aus Teilaufgabe 1 implementiert. Achtung:
Die überarbeitete Methode soll die Primzahlen nur berechnen, nicht ausgeben!

c) Korrigieren Sie bei Bedarf Ihre Klasse *SiebDesEratosthenes* von Übungsblatt 3, Aufgabe 2.
Erstellen Sie eine Kopie im lokalen Paket und ändern Sie die Klasse so, dass sie das Interface aus Teilaufgabe 1 implementiert. Achtung:
Die überarbeitete Methode soll die Primzahlen nur berechnen, nicht ausgeben!

d) Erstellen Sie eine neue Klasse *VergleichenderPrimzahlBerechner* mit einer *main*-Methode, in der Sie die maximale Zahl n eingeben (wie für Übungsblatt 3), jeweils ein Objekt der Klassen *PrimzahlenMitSchleifen* und *SiebDesEratosthenes* erzeugen und dann mit n nacheinander die beiden neuen *calculatePrime*-Methoden dieser Objekte aufrufen, so dass zwei mal (auf unterschiedliche Art und Weise) die Primzahlen bis n berechnet werden. Benutzen Sie die Iterator-Schleife, um die Inhalte der Arrays (in der *main*-Methode) in einer Zeile so auszugeben, dass man die Ergebnisse der beiden Berechnungen vergleichen kann.

Zusatzaufgabe 3 **5 Punkte**

Erstellen Sie eine neue Klasse *PrimeTimer*, die in einer Schleife zehnmal jede der beiden Primzahl-Implementierungen aus Aufgabe 2 mit einer relativ hohen Obergrenze n (z.B. 1000) aufruft. Stoppen Sie dabei mit Hilfe der beiden Methoden *initTimer* und *getDurationInMillis* aus *MakeItSimple* die jeweils benötigte Zeit und geben Sie den Durchschnittswert für jede Methode aus.

Achtung: Geben Sie die berechneten Primzahlen nicht aus, weil die Ausgabe wesentlich länger dauern würde als die Berechungen.

Aufgabe 4 **10 Punkte**

Erstellen Sie in Java ein Interface für eine Bibliothek bzw. ein Bibliotheksprogramm. Überlegen Sie, welche Aufgaben anstehen, welche Parameter jeweils dafür benötigt werden (z.B. Bücher, Euro-Beträge etc.) und welche Rückgaben sinnvoll sind. Tipp:
Schreiben Sie eine main-Methode, die Ihre Interface-Methoden aufruft und versetzen Sie sich dabei in die Rolle desjenigen, der die Aufgabe tatsächlich erledigt haben möchte. Überlagen Sie dann, welche Parameter notwendig sind und welches Ergebnis Sie erwarten.

Kommentieren Sie alle Methoden; orientieren Sie sich an den Kommentaren des String-Interface' im Web.

Hinweise zum gesamten Übungsblatt

Die Aufgaben sind in Eclipse zu bearbeiten.
Legen Sie für die Bearbeitung dieses Übungsblattes ein Paket (engl. Package) namens *uebung07* an, in dem Sie alle Klassen ablegen.

Aufgabe 3 ist wieder eine freiwillige Zusatzaufgabe: Sie können damit Ihr Punktekonto verbessern, *falls nötig*.