Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 8

Bearbeitungszeit: 02.06. bis 08.06.2008

Aufgabe 1: Rekursion: Addition und Multiplikation

Klar, Zahlen addieren und multiplizieren geht in Java ganz einfach mit den Operatoren **+** und *****. Das ist uns heute allerdings zu einfach, wir wollen ja Rekursion üben.

Schreiben Sie daher eine Klasse Numbers, die zwei statische Methoden **`int add(int a, int b)`** und **`int mult (int a, int b)`** enthält.

Diese sollen rekursiv die Summe bzw. das Produkt zweier Zahlen berechnen. Benutzen dürfen Sie daher nur folgende Teile von Java:

- **`if / else`**
- Addition von 1 (**`x + 1`**)
- Subtraktion von 1 (**`x – 1`**)
- Funktionsaufrufe Ihrer Funktionen **`add(…)`** und **`mult(…)`**
- **`return`**

Fangen Sie mit der Methode **`add(…)`** an. Überlegen Sie zuerst, in welchem Fall Sie direkt das Ergebnis zurückgeben können (Rekursionsanker) und wie Sie andere Fälle schrittweise auf den einfachen Fall zurückführen können (rekursiver Aufruf).

Auch in der Methode **`mult(…)`** benötigen Sie einen einfachen Fall als Rekursionsanker – vielleicht der Fall, dass einer der Faktoren 0 ist? Anschließend führen Sie die Multiplikation auf wiederholte Addition zurück.

Der Einfachheit halber dürfen Sie davon ausgehen, dass alle Summanden bzw. Faktoren ≥ 0 sind.

Aufgabe 2: Rekursion: Palindrome

Ein Palindrom ist ein Wort/Satz, das/der von vorne und von hinten gelesen gleich lautet. Bekannte Palindrome sind z.B. die Namen „Anna“ oder „Otto“, die Wörter „Lagerregal“ oder „Reliefpfeiler“ oder der Satz „Ein Esel lese nie“.

Schreiben Sie eine Klasse **`Words`**, die eine statische Methode **`boolean palindrom(String txt)`** enthält. Diese soll rekursiv feststellen, ob der Parameter **`txt`** ein Palindrom ist, wobei Leerzeichen ignoriert werden sollen. Auch die Groß- und Kleinschreibung soll egal sein.

Gehen Sie wie folgt vor:

a) Wandeln Sie zuerst den Parameter **`txt`** in Großbuchstaben (Methode **`toUpperCase()`** aus der Klasse **`String`**) um und entfernen Sie Leerzeichen am Anfang und Ende (Methode **`trim()`**).

b) Wenn **txt** jetzt nur noch höchstens einen Buchstaben lang ist, ist das Ergebnis der Methode **true** – ein Buchstabe ist immer ein Palindrom.

c) Andernfalls: Wenn der erste Buchstabe von **txt** sich vom letzten unterscheidet, ist das Ergebnis der Methode **false** – das Wort ist sicher kein Palindrom.

d) Wenn diese beiden Fälle ausgeschlossen sind, ist nicht sicher, ob das Wort ein Palindrom ist – wir testen daher weiter, indem wir die Methode rekursiv aufrufen, aber dabei von txt den ersten und letzten Buchstaben abschneiden (Methode **substring(…)** – siehe Java-API)!

   Beispiel: Die Methode wird mit dem String „**h o CH**“ aufgerufen. Zuerst wird dieser in „**H O CH**“ umgewandelt. Der String ist lang genug und die ersten beiden Buchstaben sind gleich, also ist das Ergebnis der Methode vom rekursiven Aufruf mit dem String „ **O C**“ abhängig. Nachdem das erste Leerzeichen entfernt wurde, ist klar, dass das Ergebnis der Methode **false** ist, da sich „**O**“ und „**C**“ unterscheiden. „**h O CH**“ ist also kein Palindrom.

e) Testen Sie Ihre Methode mit den oben angegebenen Palindromen und anderen Strings.

Aufgabe 3: Rekursion: Anagramme

Ein Anagramm ist ein Wort/Satz, das/der sich aus einem anderen durch simples Vertauschen der Buchstaben erzeugen lässt, z.B. kann „Naherholungsgebiet“ aus „Hungerlohnabsteige“ entstehen. Weitere bekannte Anagramme sind „eleven plus two“ ⇔ „twelve plus one“ und „desperation“ ⇔ „a rope ends it“.

Erweitern Sie die Klasse **Words** aus Aufgabe 2 um eine statische Methode **boolean anagramm(String t1, String t2)**. Diese soll rekursiv feststellen, ob **t1** und **t2** ein Anagrammpaar bilden. Auch hier sollen Leerzeichen und Groß-/Kleinschreibung ignoriert werden.

a) Wandeln Sie zuerst beide Parameter in Großbuchstaben um und entfernen Sie Leerzeichen am Anfang und Ende.

b) Wenn **t1** ein leerer String ist, ist das Ergebnis der Methode **false** – es sei denn, **t2** ist auch leer: dann ist das Ergebnis **true**.

c) Andernfalls: Verwenden Sie die Methode **indexOf(…)** der Klasse **String**, um den ersten Buchstaben von **t1** in **t2** zu suchen (siehe Java-API). Speichern Sie das Ergebnis in einer Variablen.

d) Wenn der erste Buchstabe nicht gefunden wurde (Ihre Variable ist **-1**), ist das Ergebnis der Methode **false** – **t2** kann sicher kein Anagramm von **t1** sein.

e) Wenn **t1** jetzt nur noch höchstens einen Buchstaben lang ist, ist das Ergebnis der Methode **true** – es sei denn, **t2** ist länger als ein Buchstabe, dann ist das Ergebnis **false**.

f) Nachdem diese Sonderfälle (bzw. Rekursionsanker!) abgedeckt sind, hängt das Ergebnis der Methode von einem rekursiven Aufruf ab. Rufen Sie also die Methode rekursiv auf, schneiden Sie aber dabei von **t1** den ersten Buchstaben ab und entfernen Sie diesen auch aus **t2** (die Fundstelle haben Sie ja zwischengespeichert). Sie benötigen auch hier wieder die Methode **substring(…)** aus der Klasse **String**.

g) Machen Sie sich klar, warum und wie die Methode funktioniert, besonders im Normalfall von rekursiven Aufrufen. Nutzen Sie dazu gerne Papier und Stift – in der Klausur haben Sie auch nichts anderes!

h) Testen Sie Ihre Methode mit den oben angegebenen Anagrammen und anderen Strings.