Technische Universität Darmstadt
Fachbereich Informatik
Prof. Dr. Johannes Fürnkranz

Allgemeine Informatik 2 im SS 2007

Übungsblatt 3

Bearbeitungszeit: 16.05. bis 22.05.2007

Aufgabe 1: Buchstaben zählen – verbesserte Version

In der letzten Übung haben Sie ein Programm geschrieben, das die Häufigkeit eines bestimmten Buchstabens in einer gegebenen Buchstabenmenge zählt. In dieser Übung werden Sie eine verbesserte Version des Programms schreiben: es wird jetzt die Häufigkeit aller Buchstaben gezählt, und als Texteingabemenge sind ganze Sätze erlaubt.

Eine Methode soll beispielsweise bei Übergabe des folgenden Satzes „**Buchstaben zaehlen macht total viel Spass“** die folgende Ausgabe liefern:

```
A: 5
B: 2
C: 2
E: 4
H: 3
I: 1
L: 3
M: 1
N: 2
O: 1
P: 1
S: 4
T: 4
U: 1
V: 1
Z: 1
```

Starten Sie BlueJ und erstellen Sie ein Projekt **uebung03**. Erstellen Sie dann eine Klasse **`LetterCounter2`** und löschen Sie im Quelltext alles zwischen **`public class LetterCounter2 {`** und der schließenden geschweiften Klammer.

Ihre Klasse soll folgendes enthalten:

- eine Methode **`public static void main(String[] args)`** und
- eine Methode **`public static int[] count(String[] text)`** und
- eine Methode **`public static void printCounterArray(int[] letterCounters)`**

In der Methode **`main`** wird die Methode **`count`** aufgerufen, welche die vorkommenden Buchstaben im **`String[] text`** zählt und zurückliefert. Dieses Ergebnis wird mittels der Methode **`printCounterArray`** auf dem Bildschirm ausgeben.

Beachten Sie folgende Hinweise:

- Zählen Sie die Buchstaben unabhängig davon, ob es Groß- oder Kleinbuchstaben sind – wandeln Sie dazu vor dem Zählen Klein- in Großbuchstaben um. Bitte suchen Sie in der **Java-API** (den Link finden Sie auf unserer Webseite im Bereich „Material“) in der Klasse **`String`** nach einer passenden Methode, die Sie auf ihren String anwenden können um ihn komplett in Großbuchstaben umzuwandeln.
- Sie dürfen davon ausgehen, dass die übergebenen Wörter nur aus Buchstaben, also Zeichen von A-Z bzw. a-z bestehen. Sie können also Umlaute, Ziffern und andere Zeichen ignorieren.
- Speichern Sie die Zählwerte in einem **`int`**-Array. Nutzen Sie dabei die Eigenschaft aus, dass **`char`**-Typen durch einen Zahlenwert repräsentiert werden. Da es bekannterweise 26 Buchstaben gibt, soll das Array 26 Elemente haben. Der Zähler von Buchstabe „A“ soll den Index 0 haben. Überlegen Sie, wie Sie das erreichen können.
- Um in den Wörtern die Buchstaben zu zählen, müssen Sie jedes Wort Buchstabe für Buchstabe durchgehen. Dazu können Sie sich entweder einzeln jedes Zeichen eines Strings als **`char`** zurückgeben lassen oder gleich den ganzen String in ein **`char`**-Array umwandeln und dieses dann durchlaufen. Für beide Möglichkeiten bietet die Klasse **`String`** Methoden (siehe **Java-API**).
- Berechnen Sie bei der Ausgabe den anzuzeigenden Buchstaben aus dem Array-Index. Dazu werden Sie einen **Typecast** nach **`char`** benötigen. Erinnern Sie sich hierfür an die zweite Vorlesung.
- Geben Sie nur die Anzahl der Buchstaben aus, die auch tatsächlich im Satz vorkommen (siehe Beispiel).

Achten Sie der Klasse und allen enthaltenen Methoden, auf angemessene JavaDoc-Kommentare mit den entsprechenden Variablen (**`@author, @version, @param, @return`**...)!

Um die Funktion Ihrer Klasse zu testen, müssen Sie sie zuerst wie in den letzten Übungen kompilieren. Wie sie bereits sicherlich bemerkt haben, wurden alle Methoden mit dem Schlüsselwort **`static`** deklariert. Aus diesem Grund müssen Sie kein Objekt der Klasse **`LettterCounter2`** erstellen. Sie können die Methoden direkt mit dem Rechtsklick auf eine Klasse ausführen. Rufen Sie so bitte die **`main`**-Methode auf. Diese verlangt einen **`String`**-Array als Eingabe, welcher wie folgt aussehen könnte:

{**`„Buchstaben“,„zaehlen“,„macht“,„total“,„viel“,„Spass“}`**

Achten Sie auf die genauen Syntax der Eingabe. Der ganze Satz muss in geschweiften Klammern stehen und jedes einzelne Wort in Anführungszeichen, getrennt durch ein Komma. Dies ist notwendig, da wir einen **`String`**-Array als Eingabe fordern.

Aufgabe 2: Caesar

Das Bedürfnis, Geheimnisse zu bewahren ist so alt wie die Menschheit selbst. Der Versuch, geschriebene Nachrichten vor den Augen anderer zu verbergen hat schon früh zur Entwicklung mehr oder minder erfolgreicher Verschlüsselungssysteme geführt. Der hier vorgestellte Chiffrieralgorithmus ist in unserer heutigen Zeit natürlich völlig überholt, eignet sich aber gut, das Prinzip deutlich zu machen.

Angeblich auf Julius Caesar geht die **Caesar-Chiffre** zurück. Sie basiert auf der zyklischen Rotation von Buchstaben um eine gegebene Anzahl Stellen im Alphabet. Die Anzahl der zu rotierenden Stellen wird als Schlüssel übergeben und ist (im wahren Leben) geheim zu halten.

Für das Wort (Klartext) **ANNA** und den Schlüssel 4 ergibt sich als Geheimtext (Schlüsseltext): **ERRE**.
Zum Entschlüsseln von **ERRE** benutzt man den Schlüssel -4.

Der Aufruf der **main**-Methode mit den Argumenten **(„7“,„ZUVERSCHLUESSELN“)** soll die folgende Ausgabe liefern: **GBCLYZJOSBLZZLSU**

Erstellen Sie eine neue Klasse. Schreiben Sie innerhalb der Klasse eine Methode **public static void main (String[] args)**, die einen Array aus Strings, in unserem Fall werden es genau 2 Elemente sein, erwartet. Das erste Element ist eine Zahl die den Verschiebungsschlüssel darstellt. Das zweite Element ist das zu verschlüsselnde Wort. Ausgeben soll das Programm den verschlüsselten Text.

Wandeln Sie zuerst das übergebene Wort in Großbuchstaben und danach den übergebenen Schlüssel in einen **int** um – benutzen Sie für letzteres die Methode **Integer.parseInt** (siehe die **Java-API** zur Klasse **Integer**).

Schreiben Sie dann eine Methode **public static string crypt(int key, String text),** die Sie von **der main-**Methode aus aufrufen. Legen Sie darin die Variablen **int p** und **int c** an. Außerdem benötigen wir einen leeren **StringBuffer result**, der die gleiche Länge hat wie **text** (siehe die **Java-API** zur Klasse **StringBuffer**, um den richtigen Konstruktor zu finden).

Hinweis: Sie haben in der Vorlesung den Unterschied zwischen einem **String**-Objekt und einem **Stringbuffer**-Objekt kennen gelernt. **String**-Objekte sind während der Laufzeit nicht veränderliche Objekte, während auf **Stringbuffer**-Objekten verschiedene Operationen ausgeführt werden können.

Gehen Sie dann **text** durch und tun Sie für jeden Buchstaben folgendes:

1. Bilden Sie den Buchstabens auf eine Zahl zwischen 0 und 25 ab und speichern Sie diese in **p**.
2. Berechnen Sie den Code des verschlüsselten Buchstabens **c** mit folgender Formel:
3. **c = (p + key) % 26**.
4. Falls **c** kleiner als 0 ist, addieren Sie 26 dazu (für negative Schlüssel).
5. Fügen Sie **result** das verschlüsselte Zeichen hinzu. Dazu müssen Sie es wieder auf einen **char** umwandeln (**Typecast**). Verwenden Sie die Methode **append(char c)** der Klasse **StringBuffer** (siehe **Java-API**) um den **char** dem **Stringbuffer result** anzuhängen.

Nach der Schleife geben Sie den **StringBuffer result** in Form eines **Strings** an die Aufrufstelle zurück. Sie können einen **StringBuffer** mittels der Methode **toString()**in einen **String** umwandeln.

Geben Sie zum Schluss den verschlüsselten Text aus.