Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 3

Bearbeitungszeit: 28.04. bis 04.05.2008

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)`**
- eine Methode **`public static int[] count(String text)`**
- eine Methode **`public static void print(int[] a)`**

In der Methode **`main`** wird überprüft, ob der Parameter **`args`** genau ein Element enthält. Wenn ja, wird dieses (**`args[0]`**) an die Methode **`count`** übergeben, welche die vorkommenden Buchstaben im **`String text`** zählt und zurückliefert. Dieses Ergebnis wird mittels der Methode **`print`** 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.
- Zählen Sie nur Buchstaben, also Zeichen von A – Z bzw. a – z. Ignorieren Sie alle anderen Zeichen (im Beispiel oben Leer- und Ausrufezeichen).
- 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 des Buchstabens „A“ soll den Index 0 haben, der des Buchstabens „Z“ den Index 25. Überlegen Sie, wie Sie das erreichen können.
- Um im **`String text`** die Buchstaben zu zählen, müssen Sie ihn 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**).
- Bei der Ausgabe soll ja nicht nur die Anzahl, sondern jeweils auch der zugehörige Buchstabe ausgegeben werden. Rekonstruieren Sie diesen aus dem Array-Index – dazu werden Sie einen **Typecast** nach **`char`** benötigen (siehe Skript).
- 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 ein **`String`**-Array als Eingabe, welches wie folgt aussehen könnte:

`{“Buchstaben zaehlen macht total viel Spass”}`

Achten Sie auf die genaue Syntax der Eingabe. Der String muss nicht nur in Anführungszeichen stehen, sondern auch in geschweiften Klammern, da die **`main`**-Methode keinen **`String`**, sondern ein **`String`**-Array als Parameter erwartet.

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**.

Erstellen Sie eine neue Klasse **`CaesarChiffre`**. Schreiben Sie innerhalb der Klasse eine Methode **`public static String crypt(int key, String text)`**. **`key`** stellt dabei den Schlüssel dar, **`text`** den Klartext.

Deklarieren Sie in der Methode die Hilfsvariablen **`int plain`** sowie **`int cipher`** und initialisieren Sie beide mit **`0`**. Außerdem benötigen wir einen **`String result`**, der mit **`""`** initialisiert wird.

Wandeln Sie außerdem den **`String text`** komplett in Großbuchstaben um.

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

1. Bilden Sie den Buchstaben auf eine Zahl zwischen 0 und 25 ab und speichern Sie diese in **`plain`**. Sie können davon ausgehen, dass Text nur Buchstaben zwischen A und Z enthält.
2. Berechnen Sie den Code des verschlüsselten Buchstabens **`cipher`** mit folgender Formel: **`cipher = (plain + key + 26) % 26`** (die Addition von 26 ist für negative Schlüssel notwendig).
3. Fügen Sie **`result`** das verschlüsselte Zeichen hinzu. Dazu müssen Sie es wieder in einen **`char`** umwandeln (**Typecast**) und dann einfach auf den **`String result`** „addieren".

Nach der Schleife geben Sie den **`String result`** als Ergebnis zurück.

Nach dem Kompilieren der Klasse können Sie mit einem Rechtsklick darauf direkt die Methode **`crypt`** aufrufen. Wenn Sie als Parameter **7** und **"ZUVERSCHLUESSELN"** übergeben, sollte sich ein Fenster mit dem Ergebnis **"GBCLYZJOSBLZZLSU"** öffnen.