

Übung 06: Zeichen und Strings

Abgabetermin: 1. 12. 2009

Name: _____

Matrikelnummer: _____

Gruppe: G1 (Prähofer) G2 (Wolfinger) G3 (Wolfinger) G4 (Jahn)

Aufgabe	Pkte	Abzugeben	gelöst	Erreichte Pkte	Kommentar
Aufgabe 06.1	12	Lösungsidee Java-Programm Testfälle und Ergebnisse	<input type="checkbox"/>		
Aufgabe 06.2	12	Lösungsidee Java-Programm Testfälle und Ergebnisse	<input type="checkbox"/>		

Aufgabe 6.1: Verschlüsselung

Schon Cäsar wusste um die Bedeutung der Kryptographie und so geht die Geschichte, dass er einen Angriffsbefehl verschlüsselt an seine Befehlshaber sandte. Er verwendete dabei ein einfaches Verfahren, die so genannte Cäsar-Chiffre, an. Dabei wird das Alphabet um eine bestimmte Anzahl von Buchstaben verschoben, z.B. um 2 Zeichen. Dann bekommt ein 'c' die Bedeutung eines 'a' und ein 'd' die eines 'b' usw. Dabei dient die Anzahl der Zeichen, um die verschoben wird, als Schlüssel.

Siehe dazu auch folgende Darstellung:

```

Schlüssel:      2
Klartext:      a b c d e f g h i j k l m n o p q r s t u v w x y z
verschlüsselt: c d e f g h i j k l m n o p q r s t u v w x y z a b
  
```

Schreiben Sie ein Programm, mit dem man Texte verschlüsseln und wieder entschlüsseln kann. Buchstaben sollen nach dem Cäsar-Chiffre verschlüsselt werden. Alle anderen Zeichen werden unverändert übernommen.

Das Programm soll über einen Dialog wie im folgenden Beispiel gezeigt bedienbar sein. Man wählt zuerst die Operation aus: `c` für chiffrieren, `d` für dechiffrieren. Dann wird der Name der Output- bzw. Inputdatei eingegeben und schließlich der Schlüssel. Beim Chiffrieren kann man eine Zeile eingeben und diese wird chiffriert in die Outputdatei geschrieben. Beim Dechiffrieren werden die Daten aus der Inputdatei gelesen und in dechiffrierter Form auf der Konsole ausgegeben.

```

Bitte Operation eingeben (c oder d): c
Output-Datei: chiffre.txt
Schlüssel: 4
Textzeile: Dies ist eine Textzeile, die verschlüsselt werden soll.
  
```

```

Bitte Operation eingeben (c oder d): d
Input-Datei: chiffre.txt
Schlüssel: 4
Originaltext: Dies ist eine Textzeile, die verschlüsselt werden soll.
  
```

Hinweis: Verwenden Sie folgende Methoden:

```

boolean Character.isLetter(char ch)
boolean Character.isUpperCase(char ch)
boolean Character.isLowerCase(char ch)
  
```

Abzugeben ist:

- Die Lösungsidee: Beschreiben Sie den Algorithmus zur Ver- und Entschlüsselung von Texten.
- Das Java-Programm
- Testfälle: eingelesene Werte und Ergebnisse

Aufgabe 6.2: CSV-Printer

CSV (= Comma Separated Values) ist ein Dateiformat-Standard für Tabellen und andere strukturierte Daten. Das CSV-Format trennt Datensätze und deren Datenfelder durch Sonderzeichen. Datensätze werden üblicherweise durch Zeilenumbrüche getrennt, Datenfelder durch Strichpunkte, Doppelpunkte, Tabulatoren oder Leerzeichen. In dieser Übung trennen wir Datenfelder mit Strichpunkten (',').

Lesen Sie eine CSV-Datei zeilenweise ein und geben Sie diese formatiert am Bildschirm aus. Ein Format-String bestimmt das Format für die Ausgabe. Strichpunkte trennen Ausrichtung und Breite der Datenfelder. Pro Datenfeld beschreibt das erste Zeichen die Textausrichtung (L=Left, C=Center, R=Right) und die darauf folgende Zahl die Spaltenbreite. Folgendes Beispiel beschreibt eine Tabelle mit vier Spalten: Spalte 1 ist rechtsbündig und 7 Zeichen breit; Spalte zwei ist linksbündig und 20 Zeichen breit; usw.:

```
R7;L22;C27;C7
```

Implementieren Sie das Interpretieren des Format-Strings wie folgt:

- Teilen Sie den Format-String mit der Methode `split` aus der Klasse `String` in Teile für die einzelnen Spalten. Die `split`-Methode zerlegt eine Zeichenkette durch die Angabe eines Trennzeichens in Teil-Strings und gibt diese in einem `String-Array` zurück.
- Verwenden Sie die Methode `charAt` aus der Klasse `String` um das erste Zeichen der Teil-Strings zu lesen und die Spaltenausrichtung zu ermitteln. Mit der Methode `substring` aus der Klasse `String` lesen Sie den Teil-String der die Spaltenbreite beschreibt. Wandeln Sie den Teil-String für die Spaltenbreite mit der Methode `parseInt` aus der Klasse `Integer` in den `int` Datentyp um. Speichern Sie Ausrichtung und Breite der Spalten jeweils in ein `Array`. (Hinweis: Eine Funktion in Java kann nur einen Rückgabewert haben. Verwenden Sie ausschließlich lokale Variablen und Parameter und implementieren Sie jeweils eigene Methoden für die Ausrichtung und für die Spaltenbreite.)

Implementieren Sie das formatierte Ausgeben der Daten wie folgt:

- Lesen Sie die zu formatierenden Daten zeilenweise aus einer CSV-Datei mit der Methode `In.readLine`.
- Verwenden Sie die Klasse `StringBuilder` um eine Zeile zu formatieren. Ermitteln Sie in einer Schleife für jede Spalte deren Anfangs- und End-Position mit der Methode `indexOf` der Klasse `StringBuilder`. Übergeben Sie nun die `StringBuilder`-Referenz, Anfangs- und End-Position der Spalten, Spaltenausrichtung und Spaltenbreite an eine von Ihnen implementierte Methode `formatColumn`. Die Methode `formatColumn` formatiert richtet den Text der Zeile direkt im übergebenen `StringBuilder` aus.

```
void formatColumn(StringBuilder builder, int start, int end,
                  char alignment, int width) { ... }
```

- Geben Sie nach der Formatierung der Spalten die aktuelle Zeile auf der Konsole aus.

Beispiel:

Der Bildschirmdialog des Programms könnte wie folgt aussehen:

```
C:\GdP\06>java CsvFormatter
```

```
Dateiname: formul1.csv
Format-String: R7;L22;C27;C7
```

Platz	Fahrer	Nation	Siege
1.	Michael Schumacher	Deutschland	91
2.	Alain Prost	Frankreich	51
3.	Ayrton Senna	Brasilien	41
4.	Nigel Mansell	Vereinigtes Koenigreich	31
5.	Jackie Stewart	Vereinigtes Koenigreich	27
6.	Jim Clark	Vereinigtes Koenigreich	25
6.	Niki Lauda	Oesterreich	25
8.	Juan Manuel Fangio	Argentinien	24
9.	Nelson Piquet	Brasilien	23
10.	Damon Hill	Vereinigtes Koenigreich	22

Hinweise:

- Gehen Sie davon aus, dass der Format-String so gewählt wird, dass alle Werte in den Spalten Platz haben.
- Richten Sie die Überschrift gleich wie alle anderen Zeilen aus.
- **WICHTIG:** Lösen Sie die Aufgabe **OHNE** der Methode `String.format`.

Freiwillige Zusatzfunktionen (kein Extrapunkte):

- Vertikale und horizontale Trennlinien.
- Abstand bei links- und rechtsbündiger Ausgabe zum Spaltenrand.

Abzugeben ist:

- Die Lösungsidee: Beschreiben Sie wie die Methode `formatColumn` Spalten an die jeweilig gewünschte Breite und Ausrichtung anpasst.
- Das Java-Programm
- Testfälle: eingelesene Werte und Ergebnisse