HashMap

Einführung

Eine `HashMap` kann unter Java mit einer `ArrayList` verglichen werden. Jedoch mit einem wichtigen Unterschied: Die „Schubladen/Fächer“, die in einer `ArrayList` (wie auch bei einem statischen Array) fortlaufend von 0 ab durchnummeriert sind, können bei einer `HashMap` beliebig benannt werden. Das bedeutet, der *Index* muss keine Zahl sein, sondern kann irgendein Objekt sein, beispielsweise Buchstaben.
Hinweis: hierfür muss das Paket `import java.util.HashMap;` eingebunden werden!

```
1 HashMap<Character, Integer> zaehler = new HashMap<>();
2 zaehler.put('a',5);
3 System.out.println( "a: " + zaehler.get('a') );
```

Listing 1: eine HashMap anlegen

Hierbei werden in den spitzigen Klammern 2 Datentypen angegeben: der erste Datentyp gibt an, womit wir unsere Schubladen bezeichnen wollen. Der zweite Datentyp gibt – wie bei einer `ArrayList` – an, welche Daten wir in den Schubladen speichern wollen. Im Beispiel wird eine `HashMap` angelegt, deren Index ein einzelner Buchstabe (`Character`) ist, und in die wir Zahlen ablegen können.
Ähnlich wie bei einer `ArrayList` können wir mit der Methode `get('a')` den Wert abrufen (s. Zeile 3), der im Fach mit dem Index *a* abgelegt wurde. Das Ablegen funktioniert mit der Methode `put('a',5)`;
Die Abfrage *aller* Einträge ist mit einer `HashMap` allerdings etwas komplizierter, da wir nicht mehr einfach durchnummerierte Fächer haben.
Hinweis: hierfür muss das Paket `import java.util.Map;` eingebunden werden!

```
1 for(Map.Entry e : zaehler.entrySet()) {
2     System.out.println(e.getKey() +": " + e.getValue());
3 }
```

Listing 2: eine HashMap durchlaufen

Mit einer sogenannten „for-each“-Schleife können wir jeden Eintrag durchgehen. Hierbei wird bei jedem Schleifendurchlauf der nächste verfügbare Eintrag in der Variablen `e` vom Datentyp `Map.Entry` abgespeichert. Mit `e.getKey()` kann man dann den Index (*engl. key*), also die „Beschriftung“ des Faches abfragen, mit `e.getValue()` bekommen wir den im Fach gespeicherten Wert.

1. Aufgabe

Erstelle ein neues Paket `ab32` mit einer Klasse `Zaehlen` (inklusive `main`-Methode).
Lasse dann mithilfe der Methode von letztem Arbeitsblatt eine Textdatei einlesen, gehe dann den Text mit einer `for`-Schleife durch und zähle, wie oft jedes Zeichen in dem Text vokommt. Lasse anschließend die Häufigkeit aller Zeichen auf der Konsole ausgeben.
Hinweis: du kannst die `Reader.readFile()` Methode von letztem Mal benutzen, indem du die Klasse mit `import ab32.Reader;` einbindest!

2. Aufgabe

Programmiere eine einfache „rot13“-Verschlüsselung: hierbei sollen die Buchstaben jeweils um 13 Stellen „verschoben“ werden, d. h. aus `a` wird ein `n`, aus `b` wird `o` usw.
Betrachte hierzu nochmals die ASCII-Tabelle. Verschoben werden sollen dabei nur die Buchstaben (d. h. ASCII-Werte 65 bis 90 für die Großbuchstaben, sowie 97 bis 122 für die Kleinbuchstaben)
Für die Verschiebung musst du also zum ASCII-Wert 13 hinzuaddieren und das Ergebnis bei Bedarf (d. h. wenn der Wert über 90 ist) an das „untere“ Ende des Bereiches setzen, so dass aus einem `u` ein `h` wird.

- Programmiere hierfür eine `static`-Methode `verschluesseln(String s)` welche den übergebenen String „verschlüsselt“ und wieder zurückgibt.
- Ergänze anschließend die Methode um einen weiteren Parameter `int weite`, welcher angibt, wie weit die Buchstaben verschoben werden sollen (also anstatt fest um immer 13 Positionen zu verschieben).
- Programmiere zusätzlich eine Methode `entschluesseln(String s,int weite)`