## Übung 5.1 (String Code Snippets)

Implementiere die folgenden Methoden als `static`. Alle sind nach dem gleichen Schema aufgebaut, den Methoden werden 2 Strings übergeben. Im ersten (längeren) wird ein zweiter (kürzerer) String gesucht und an der gefundenen Stelle im längeren String diverse Aktionen ausgeführt (siehe Angabe). Der sich ergebende neue String wird als Rückgabewert verwendet.

a) Schreibe eine Methode `toUpper` mit der ein beliebiges Wort in einem beliebigen Text auf Großbuchstaben geändert werden kann. Anwendung:

```
String s="Wo sind die Klassen?"
String r=toUpper(s, "sind");
sout(r);
// Wo SIND die Klassen?
```

b) Implementiere eine Methode `count` die zählt wie oft ein Wort in einem Text vorkommt. Hinweis verwende die `indexOf` Variante mit 2 Parameter.

c) Implementiere eine Methode `killWord` das das erste Vorkommen eines Worts aus einem Text löscht.

d) Implementiere eine Methode `killWords` mit einem Algorithmus der alle Vorkommen eines Worts aus einem Text löscht.

## Übung 5.2 (Zähler)

Mit Hilfe einer Klasse `Zaehler` soll Anzahl und Bezeichnung verwaltet werden. Beispiele:

- 3 Dosen,
- 5 Stueck,
- 9 Äpfel

Aufgaben:

a) Lege eine Klasse an mit den Instanzvariablen `anzahl` (`int`) und `bezeichnung` (`String`). Definiere einen passenden Konstruktor.

b) Schreibe eine Ausgabemethode `print` die die Anzahl gefolgt von der Bezeichnung ausgibt. Wobei die Bezeichnung immer mit großem Anfangsbuchstaben und ansonsten klein ausgegeben werden soll. Für die Korrektur der Schreibweise soll die Klasse eine eigene Methode enthalten die z.B. `"biRNeN"` in `"Birnen"` umwandelt.

c) Mit der Methode `einsmehr` soll die Anzahl um eins erhöht werden können.

d) Schreibe eine weitere `print` Methode mit einem String als Parameter. Mit diesem String kann die gewünschte Ausgabe vorgegeben werden und enthält Platzhalter für die gewünschte Position von Anzahl `%n` und Bezeichnung `%b`.

> Beispiel
>
> `"Die Anzahl der %b im Korb ist %n"`
>
> ergibt in der Ausgabe:
>
> `"Die Anzahl der Äpfel im Korb ist 42"`

teste auch: `"Es sind %n %b im Korb"`

## Übung 5.3 (substring)

Schreibe eine (`static`) Methode mit der gleichen Funktionalität wie `substring`. Hinweis: Verwende `charAt` und den `+` Operator, siehe folgendes Beispiel.

```
String s="abcdefg";
String r=""; // leerer String
r=r+s.charAt(2);
r=r+s.charAt(3);
```

## Übung 5.4 (Rechtsbündiges Ausgeben)

Schreibe ein Programm das mehrere Strings unterschiedlicher Länge rechtsbündig untereinander ausgeben kann. Beispiel:

```
      Ich
    finde
Informatik
     toll
```

Definiere dafür eine Methode

`String rightAlign(String str,int n)`

die den String und die Breite als Parameter übergeben bekommt. Rückgabewert ist ein String der durch Anfügen von Leerzeichen auf die gewünschte Länge gebracht wurde. Rufe diese Methode vier mal auf und gib die Ergebnisse aus, sodass obige Ausgabe entsteht.

## Übung 5.5 (Longest common prefix)

Schreibe eine (`static`) Methode die zwei Strings übergeben bekommt und zurück gibt wie viele Buchstaben am Anfang übereinstimmen.

```
p r e f e t c h
0 1 2 3 4 5 6 7
p r e f i x
```

## Übung 5.6 (Longest common suffix)

Schreibe eine (`static`) Methode die zwei Strings übergeben bekommt und zurück gibt wie viele Buchstaben am Ende übereinstimmen.

```
    0 1 2 3 4 5
    a b c x y z
e f g h i x y z
0 1 2 3 4 5 6 7
```

## Übung 5.7 (ROT13)

ROT13 dient dazu einen Text vorübergehend unlesbar zu machen, z.B. weil es sich um die Lösung eines Rätsels handelt. Dabei wird jeder Buchstabe eines Strings um 13 Stellen im Alphabet verschoben. Wird ROT13 zweimal angewandt so entsteht wieder das ursprüngliche Wort. Schreibe ein Programm das von einem String die ROT13 Kodierung ausgibt. Verwende nur Großbuchstaben. Bsp.: `"FSST"` ⟶ `"SFFG"`.

Hinweise: ASCII Code der Großbuchstaben `A = 65 ... 90 = Z`. Wenn `c` eine `char` Variable ist dann ergibt `(int)c` den ASCII Code. Umgekehrt kann mit `(char)i` der ASCII Code in der `int` Variable `i` wieder in ein `char` gewandelt werden.

## Übung 5.8 (Zahlentext)

Nimm einen String der nur aus Großbuchstaben besteht und erzeuge daraus einen String aus Zahlen die durch Leerzeichen getrennt sind. Jede Zahl entspricht der Position im Alphabet. Bsp.: `"ABCXYZ"` ⟶ `"1 2 3 24 25 26"`.

Schreibe weiters ein Programm das den zweiten String wieder in einen lesbaren Text wandelt.