Caesar-Verschlüsselung

Das folgende übernehmen wir aus **http://de.wikipedia.org/wiki/Verschiebechiffre** (bzw. aus **http://de.wikipedia.org/wiki/Monoalphabetische_Substitution**)

„Die **Caesar-Verschlüsselung** ist ein besonders einfacher Sonderfall einer einfachen monoalphabetischen Substitution. Zum Zwecke der Verschlüsselung wird dabei jeder Buchstabe des lateinischen Standardalphabets um eine bestimmte Anzahl von Positionen zyklisch verschoben (rotiert). Die Anzahl bestimmt den Schlüssel, der für die gesamte Verschlüsselung unverändert bleibt. Es ist eine der einfachsten (und unsichersten) Formen einer Geheimschrift."
„Die **monoalphabetische Substitution** ist ein Verschlüsselungsverfahren, bei dem jeder Buchstabe oder jedes Zeichen durch ein anderes Zeichen nach Vorgabe eines einzigen (lat.: *mono*) Alphabets ersetzt wird. Ein Beispiel ist die folgende Verschlüsselung.

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
Geheimtext:	U F L P W D R A S J M C O N Q Y B V T E X H Z K G I

Aus dem Klartext „der affe ist dumm" wird hier der Geheimtext „PWV UDDW STE PXOO". Der Klartext lässt sich wieder aus dem Geheimtext rekonstruieren indem man dort die Buchstaben in der zweiten Zeile durch die der ersten Zeile ersetzt."

Beispielprogramm (unvollständig!):

```
public class Caesar {

  public static void main(String[] args) {
    String Text = "Die Caesar-Verschlüsselung ist ein besonders einfacher Sonderfall einer \n";
    Text = Text + "einfachen monoalphabetischen Substitution. Zum Zwecke der Verschlüsselung wird\n";
    Text = Text + "dabei jeder Buchstabe des lateinischen Standardalphabets um eine bestimmte \n";
    Text = Text + "Anzahl von Positionen zyklisch verschoben (rotiert). Die Anzahl bestimmt \n";
    Text = Text + "den Schlüssel, der für die gesamte Verschlüsselung unverändert bleibt. \n";
    Text = Text + "Es ist eine der einfachsten (und unsichersten) Formen einer Geheimschrift.\n";
    //Text = Text + " \n";


  }


}
```

Ziel (!):

Jok Igkygx-Bkxyinrüyykratm oyz kot hkyutjkxy kotlginkx Yutjkxlgrr kotkx kotlginkt sutugrvnghkzoyinkt Yahyzozazout. Fas Fckiqk jkx Bkxyinrüyykratm coxj jghko pkjkx Hainyzghk jky rgzkotoyinkt Yzgtjgxjgrvnghkzy as kotk hkyzosszk Gtfgnr but Vuyozoutkt feqroyin bkxyinuhkt (xuzokxz). Jok Gtfgnr hkyzossz jkt Yinrüyykr, jkx lüx jok mkygszk Bkxyinrüyykratm atbkxätjkxz hrkohz. Ky oyz kotk jkx kotlginyzkt (atj atyoinkxyzkt) Luxskt kotkx Mknkosyinxolz.

Tipps:

1) Schreibe eine Methode encrypt (Signatur: **`encrypt(String text, int n)`**), der Parameter n steht für die Anzahl an Buchstaben, um die verschoben wird. Aufruf im Beispiel: **`encrypt( Text,6 );`**
2) So ermittelt man die Länge des Strings: **`int laenge = text.length();`**

3) Jetzt schreibst du erst mal eine Schleife, in der du den Text einmal Zeichen für Zeichen durchgehst und ausgibst. ZEICHEN FÜR ZEICHEN! Das Zeichen an der 7. Stelle bekommst du so: `char zeichen = text.charAt(6);`
   Und dann: `System.out.print( zeichen );`
4) Jedem Zeichen ist aufgrund einer Kodierung (z. B. ASCII ⇨ WWW!) eine Zahl zugeordnet, also funktioniert der folgende Befehl tatsächlich:

```
int konvert = (int) zeichen;
```

   Test: `System.out.print( konvert );`

➢ **Lösung für das Problem „Konvertieren einer Binärzahl in eine Dezimalzahl“:**

Die Lösung gibt’s im WWW: **http://www.wspiegel.de/upl/Konvert.html**

Die Lösung diskutieren wir im Unterricht!

➢ **Lösung für das Problem „Zeitrechnung“:**

Die Lösung gibt’s im WWW: **http://www.wspiegel.de/upl/Zeitrechnung.html**

Die Lösung macht intensiven Gebrauch von der Division mit Rest: wenn man 135 durch 60 teilt, bleibt ein Rest von 15, in Java: `rest = 135 % 60`

Dabei ist `%` der sogenannte Modulo-Operator, der den Rest bei einer ganzzahligen Division liefert. (Ganzzahlige Division in Java: `135 / 60` ⇨ Ergebnis: **2**)

Aufgaben

1. Nimm das Beispiel-Programm **`Caesar.java`** und versuche die vier Schritte oben in ein lauffähiges Programm umzusetzen. Nach jedem Schritt **testen**, testen, testen . . .
2. Schau mal im Internet nach der ASCII-Kodierung (alternativ: Unicode)
3. Verschlüsseln (**`encrypt`**) ist nur die eine Richtung! Irgendwann möchte man gerne wieder entschlüsseln: gesucht ist also eine Methode **`decrypt`** . . .

Du findest dieses Aufgabenblatt unter **http://upl.wspiegel.de/java_05.pdf**