Aufgabe 3.1

Ziffern werden im Morse-Alphabet durch jeweils 5 Zeichen folgendermaßen dargestellt:

```
0    -----        1    .----        2    ..---
3    ...--        4    ....-        5    .....
6    -....        7    --...        8    ---..        9    ----.
```

Schreiben Sie ein Java-Programm, das 5 einzelne char-Zeichen über die Tastatur einliest und – falls es sich um eine Morse-Ziffer handelt – den entsprechenden Dezimalwert auf dem Bildschirm ausgibt.

Hinweis: Speichern Sie die einzelnen char Zeichen in einem String ab.

Nutzen Sie zum Einlesen der Zeichen die Klasse Scanner.

Beispiel zum Einlesen einer int-Zahl:

```
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
```

Mehr Informationen finden Sie unter:
http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

Aufgabe 3.2

Schreiben Sie ein Java-Programm, das zunächst die Eingabe eines positiven int-Wertes max und eines positiven int-Wertes div über die Tastatur erwartet. Anschließend soll das Programm alle Zahlen zwischen 0 und max, die sich ohne Rest durch div dividieren lassen, auf den Bildschirm ausgeben.

Beispiel:

Eingabe: max: 30

div: 9

Ausgabe: 0

9

18

27

Aufgabe 3.3 (Arrays)

Eine Funktion f(int[],int[]), die zwei eindimensionale Arrays als Parameter übergeben bekommt und überprüft, ob die darin gespeicherten Werte jeweils gleich sind. Schreiben Sie Methoden für Integer und für String Arrays.

Beispiel: f([1,4,3], [1, 4, 3]) == true

Aufgabe 3.4 - Wechselkursrechner

Schreiben Sie ein Java-Programm, das einen Betrag in Euro erfasst und in Dollar umgerechnet ausgiebt.

Einen Aktuellen Wechselkurs finden Sie hier auf finanzen.net

Aufgabe 3.5 - Datumsprüfung

Fragen Sie vom Benutzer eine Datumseingabe in drei Schritten ab. Im ersten Schritt soll der Benutzer den Tag, im zweiten Schritt den Monat und im dritten Schritt das Jahr eingeben (jeweils als Ganzzahlen). Das Programm soll ausgeben, ob es sich bei den drei eingegebenen Zahlen für Tag, Monat und Jahr um ein gültiges Datum handelt. Wird ein ungültiges Datum erkannt, soll der Grund für den Fehler in der Konsole ausgegeben werden.

Beispielsweise ist zu prüfen, ob für den Monat eine Zahl zwischen 1 und 12 eingegeben wurde. Weiterhin muss geprüft werden, ob es sich um einen Monat mit 28, 29 (Schaltjahr), 30 oder 31 Tagen handelt. Außerdem soll geprüft werden, ob das Datum nach dem 15.10.1582 liegt (Beginn des gregorianischen Kalenders). Ergänzen Sie selbst weitere nötige Datumsprüfungen.

Ein Jahr ist ein Schaltjahr ist, wenn:

- die Jahreszahl durch 4 teilbar ist
- Ausnahme 1: Ist die Jahreszahl durch 100 teilbar, so liegt kein Schaltjahr vor.
- Ausnahme 2: Ist die Jahreszahl durch 400 teilbar, so liegt immer ein Schaltjahr vor, d.h. die Jahre 2000, 2400 und 2800 sind beispielsweise Schaltjahre.

Aufgabe 3.6 - Notenliste

Schreiben Sie ein Programm, das es dem Benutzer erlaubt, eine Notenliste eines Kurses einzugeben. Für jeden Kursteilnehmer soll der Name und die Note erfasst werden. Nach jeder Eingabe eines Namens und einer zugehörigen Note soll der Benutzer gefragt werden:

```
Weiteren Kursteilnehmer erfassen (0/1)?
```

Antwortet der Benutzer mit 1, so werden ein weiterer Name und eine weitere Note abgefragt. Antwortet der Benutzer mit 0, so wird die Notenliste ausgegeben. Hierbei soll aber jede Note um den Wert 1 verringert werden (Achtung: es gibt nur ganzzahlige Noten zwischen 1 und 6). Achten Sie darauf, dass gültige Noten ein- und ausgegeben werden. Außerdem sollen die Teilnehmer bei der Ausgabe automatisch durchnummeriert werden. Beispielsweise sollte das Programm diese Ausgabe für einen Kurs mit drei Teilnehmern erzeugen:

```
1. Max Mustermann: 3.0
2. Sepp Maier:     1.0
3. Marta Wagner:   2.0
```

Der Kurs kann beliebig viele Teilnehmer haben.

Hinweis: Nutzen Sie für die Abfrage von Strings aus der Command-Line diesen Quellcode:

```
Scanner sc = new Scanner(System.in);
String input = sc.nextLine();
```

Aufgabe 3.7 - Primfaktorzerlegung

Schreiben Sie ein Java-Programm, das die Primfaktoren einer natürlichen Zahl berechnet. Die Ausgabe soll in einem bestimmten Format erfolgen (vgl. Beispiele unten). Dabei sollte

- „*“ für die Multiplikation und
- „^“ für die Potenz

stehen. Die Rechenzeit des Programms spielt keine Rolle.

Beispiele:

```
Geben Sie eine Zahl ein: 10
Die Primfaktorzerlegung von 10 ist: 2*5

Geben Sie eine Zahl ein: 256
Die Primfaktorzerlegung von 256 ist: 2^8

Geben Sie eine Zahl ein: 6534
Die Primfaktorzerlegung von 6534 ist: 2 * 3^3 * 11^2
```

```
Geben Sie eine Zahl ein:6936
Die Primfaktorzerlegung von 6936 ist: 2^3 * 3 * 17^2

Geben Sie eine Zahl ein:37
Die Primfaktorzerlegung von 37 ist: 37
```

Hintergrundinformationen zur Primfaktorzerlegung:
http://de.wikipedia.org/wiki/Primfaktorzerlegung

Hier können Sie testen, ob Ihr Programm korrekte Ergebnisse liefert:
http://www.mathepower.com/primfaktor.php

Aufgabe 3.8 - Caesar Verschlüsselung

Aufgabe 3.8.1

Schreiben Sie ein Java-Programm, das einen Text mit Hilfe der Caesar-Verschlüsselung chiffriert. Bei dieser einfachen (und unsicheren) Verschlüsselung wird jeder Buchstabe des Klartexts auf einen Geheimtextbuchstaben abgebildet. Diese Abbildung ergibt sich, indem man die Zeichen eines geordneten Alphabets um eine bestimmte Anzahl zyklisch nach rechts verschiebt (rotiert). Die Anzahl der verschobenen Zeichen bildet den Schlüssel, der für die gesamte Verschlüsselung unverändert bleibt.

Eine mögliche Zuordnung der Klartextbuchstaben zu den Geheimtextbuchstaben ist beispielsweise:

```
Klar:    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
Geheim:  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
c
```

Der Schlüssel ist in diesem Fall 23. Hier wird der Buchstabe a (Klartext) durch den Buchstaben d (Geheimtext), der Buchstabe b durch den Buchstaben e, usw. bei der Verschlüsselung ersetzt. Das Programm fordert den Benutzer im ersten Schritt zur Eingabe des Schlüssels (eine Zahl zwischen 1 und 26) auf. Anschließen kann der Benutzer den zu verschlüsselnden Klartext eingeben. Das Programm liefert den mit Hilfe der Caesar-Chiffre verschlüsselten Geheimtext. Zur Vereinfachung sollen dabei nur die oben dargestellten 26 Buchstaben (keine Sonderzeichen, Leerzeichen, Umlaute, usw.) verschlüsselt werden.

Hinweis: Weitere Informationen zur Caesar-Verschlüsselung finden Sie unter http://de.wikipedia.org/wiki/Caesar-Verschlüsselung

Aufgabe 3.8.2 (Caesar -Entschlüsselung)

Schreiben Sie ein Java-Programm, das in der Lage ist, einen verschlüsselten Text aus Aufgabe 10 zu entschlüsseln. Dazu gibt der Benutzer zunächst den Schlüssel (eine Zahl zwischen 1 und 26) ein. Anschließend gibt der Benutzer den Geheimtext ein. Das Programm zeigt den entschlüsselten Klartext an, sofern der korrekte Schlüssel eingegeben wurde.

Aufgabe 3.9 - Roulette

Schreiben Sie ein Java-Programm, das ein vereinfachtes Roulette-Spiel simuliert, in dem nur auf Zahlen (full number) oder auf Rot bzw. Schwarz gesetzt werden kann. Der Benutzer verfügt über ein Startkapital in Höhe von 1.000 Euro. Vor jedem Spieldurchlauf wird der Benutzer aufgefordert, seinen Einsatz zu nennen und entweder auf eine bestimmte Zahl oder auf rote bzw. schwarze Zahlen zu setzen. Nachdem der Benutzer seinen Einsatz und seine Spielentscheidung eingegeben hat („rien ne va plus“), bestimmt das Programm zufällig die Gewinnzahl. Hat der Benutzer direkt auf die Gewinnzahl gesetzt, wird ihm der 35-fache Einsatz ausgezahlt. Hat er auf die richtige Farbe gesetzt, wird ihm der 2-fache Einsatz ausgezahlt. In allen anderen Fällen verliert der Benutzer den Einsatz. Das Spiel wiederholt sich solange der Benutzer über Kapital verfügt.

Weitere Informationen: http://de.wikipedia.org/wiki/Roulette

Aufgabe 3.10 - Römische Zahlendarstellung

Schreiben Sie ein Java-Programm, das eine natürliche Zahl in der command line abfragt. Das Programm gibt diese Zahl in der römischen Zahlendarstellung aus. Die eingegebene Zahl soll kleiner als 1000 sein.

Weitere Informationen zu den römischen Zahlen:
http://de.wikipedia.org/wiki/Römische_Zahlen

Aufgabe 3.11 - Zahlensystem

Aufgabe 3.11.1

Schreiben Sie ein Java-Programm, das eine eingegebene natürliche Zahl in Binärdarstellung ausgibt.

Aufgabe 3.11.2

Schreiben Sie ein Java-Programm, das eine eingegebene natürliche Zahl in Hexadezimaldarstellung ausgibt.