	Algorithmen - Modulo		AnPr	V 1.0
	Name	Klasse	Datum	

1 Allgemeines zu „Modulo“

Für alle positiven Zahlen gilt, dass „Modulo“ der Rest einer ganzzahligen Division ist:

$$17 : 6 = 2\ Rest\ 5$$

In Programmiersprachen wird dies wie folgt notiert:

```
public static void doModulo() {
  int dividend = 17;
  int divisor = 6;
  int quotient = dividend / divisor;
  int modulo = dividend % divisor;
  System.out.println(dividend + " : " + divisor + " = " + quotient
                     + " Rest " + modulo);
}
```

Im negativen Bereich müsste (mathematisch korrekt) würde `-17 mod 6` die 1 ergeben, da Modulo eigentlich berechnet, „wie viel zum nächstgrößeren Vielfachen“ fehlt. -17 ist somit 6 * (-3) + **1**. Bei allen gängigen Programmiersprachen (außer Python) ist die implementierte Funktionalität eigentlich der tatsächliche „Rest der ganzzahligen Division“. `-17 % 6` ergibt hier die **-5**. Da wir uns für diese Übung nur im positiven Bereich bewegen, ist diese Unterscheidung aber nicht notwendig.

Modulo können wir in der Programmierung für diverse Algorithmen verwenden.

2 Prüfung auf gerade/ungerade

Eine Zahl, welche bei der Division durch 2 einen Rest aufweist ist immer ungerade:

```
public static boolean isOdd(int i) {
  return (i % 2) == 1;
}
```

3 Extraktion von 10er Stellen

Weiterhin zum „ausschneiden“ von Informationen. Wollen wir bspw. die Einerstelle aus einer Zahl extrahieren, hilft uns hier Modulo:

```
public static int getEiner(int i) {
  return i % 10;
}
```

4 Extraktion von Informationen

Die Extraktion von 10er Stelle können wir noch weitertreiben – wir können in Zahlen codierte Informationen extrahieren. Um den „Sinn“ dieses Ansatzes zu verstehen, müssen wir nun erstmal einen Code schaffen. Hierzu folgendes Beispiel. Wir haben eine Uhrzeit im Format „hh:mm:ss“. Wir wollen diese Zeit in einer einzigen `int` Variablen hinterlegen. Die kleinste Einheit ist die Sekunde. Also ist es sinnvoll, sowohl die Minuten, als auch die Stunden in Sekunden umzurechnen und diese zu addieren. Ein möglicher Code würde wie folgt aussehen:

```
public static int getSeconds(int h, int m, int s) {
  int code = h;
  code *= 60; // Umrechnung Stunden in Minuten
  code += m; // Addieren der Minuten
```

```
  code *= 60; // Umrechnung der Minuten in Sekunden
  code += s; // Addieren der Sekunden
  return code;
}
```

Um nun wieder zu dem ursprünglichen Format zurückzukommen, müssen wir die Informationen als 60er Pakete extrahieren:

```
public static String getTime(int code) {
  int s = code % 60;
  code /= 60;
  int m = code % 60;
  code /= 60;
  int h = code;
  return h + ":" + m + ":" + s;
}
```

Nun kann man das Programm zur Erzeugung des Codes auch anders interpretieren.
1.: Übernehme die Stunden als Information.
2.: „Schaffe Platz für 60 neue Zustände". Wenn wir die Minuten addieren wollen, wird dies eine Zahl zwischen 0-59 sein. Dazu multiplizieren wir den Code mit 60. Das Ergebnis wird also garantiert größer 59 sein.
3.: Addiere die Minuten
4.: „Schaffe Platz für 60 neue Zustände" – das sind die Sekunden, welche ebenfalls einen Wert zwischen 0 und 59 haben werden.
5.: Addiere die Sekunden.

Die Interpretation, dass wir „Platz schaffen" ermöglicht es uns nun, unterschiedliche Werte zu codieren. Sehen wir uns hierfür folgenden Code an:

```
public static int getCodeFromDate(int y, int m, int d){
  int code = y;
  code *= 12;
  code += m - 1;
  code *= 31;
  code += d - 1;
  return code;
}
```

Wir übernehmen das Jahr als Information. Nun benötigen wir Platz für die Monate. Hier haben wir 12 verschiedene Zustände, also multiplizieren wir mit 12. Danach addieren wir die Monate. Da wir bei den Monaten jedoch die Werte 1-12 haben, jedoch bei der 0 beginnen müssen, ziehen wir die 1 ab und kommen auf 0-11. Bei den Tagen ist der maximale Wert die 31, also multiplizieren wir mit 31. Schreiben Sie nun den Code für die Rückcodierung:

```
public static String getDate(int code) {
    return d + "." + m + "." + y;
```

5 Aufgabenstellung

In einer Fabrik muss die Qualitätskontrolle getrackt werden. Hierzu wird nach jeder Prüfung eine Kontrollziffer generiert, welche folgende Informationen enthält:

- Das Datum
- Das Kürzel des Kontrolleurs. Dieses besteht immer aus zwei Kleinbuchstaben a-z
- Die letzte Ziffer ist eine Prüfziffer, welche die Einerstelle der Quersumme der oberen beiden Punkte darstellt.

Erstellen Sie folgende Methoden:

```
String getCode(String date, String empl) {…}
```

Sie soll bspw. bei folgendem Aufruf:

```
String sCode = getCode("21.10.2023", "ba");
```

den String `5089300061` zurückgeben.

Für die Rückcodierung soll folgende Methode umgesetzt werden:

```
String getValues(String sCode) {…}
```

Welche bei dem Aufruf

```
String sValues = getValues("5089300061");
```

Folgenden String zurückgeben soll:

```
21.10.2023 e:ba  (code valid)
```

Für den Fall, dass die Checksumme nicht korrekt ist, soll anstatt „`code valid`“ die Nachricht „`code not valid`“ enthalten sein.

Rechnen Sie vorher aus, was der größtmöglich Wert für die Codierung (ohne Prüfziffer) sein kann und tragen Sie ihn hier ein. Basierend darauf müssen Sie den Datentyp für die interne Berechnung des Codes festlegen: