Allgemeine Hinweise zu dieser Übung:

Die in den folgenden Aufgaben zu entwickelnden Programme sollen beliebige Wiederholungen zulassen. Die Programme sollen gezielt verlassen werden können. Fehleingaben sollten möglichst erkannt werden.

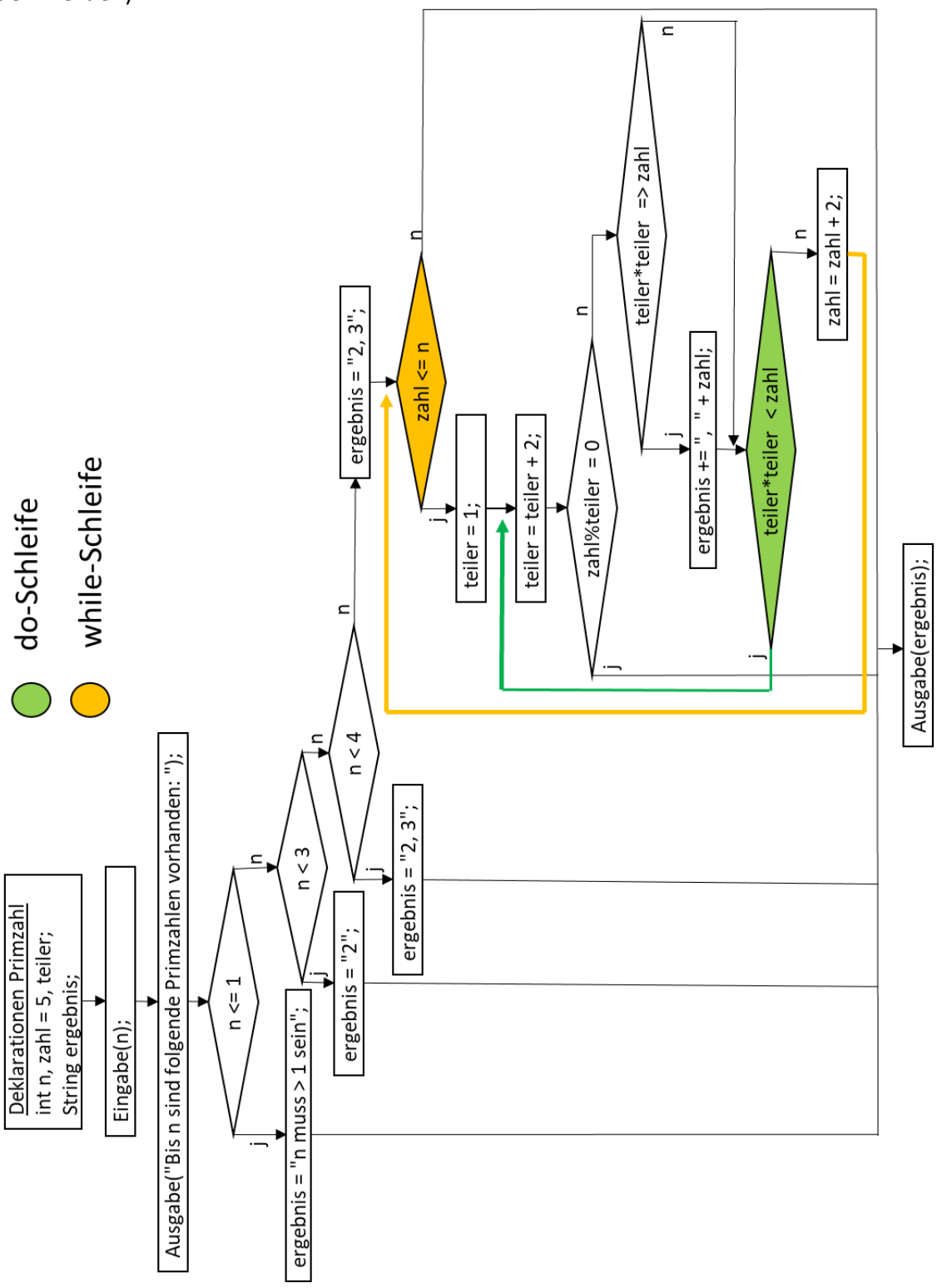
1. Aufgabe

Schreiben Sie ein Java-Konsolprogramm, welches bei der Eingabe von einzelnen Ziffern zwischen 0 und 9 diese in die entsprechenden Worte umsetzt (null, …, neun). Das Programm soll sich robust gegenüber Fehleingaben verhalten. Möglicher Beispieldialog:

```
Zahl> 5
die Zahl 5 als Wort: fünf
weiter(j), ende(e)> j
Zahl> 13
die Zahl 13 als Wort: -->Fehler, ungültige Zahl: 13
weiter(j), ende(e)> j
Zahl> 3
die Zahl 3 als Wort: drei
weiter(j), ende(e)> e
*** E N D E ***
```

2. Aufgabe

Schreiben Sie ein Java-Konsolprogramm, welches das folgende Struktogramm für die Berechnung von Primzahlen umsetzt (alle Primzahlen bis zum eingelesenen Endwert n sollen gefunden werden):



Dialogbeispiel:

```
N> 75
Bis 75 sind folgende Primzahlen vorhanden:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73
weiter(j), beenden(e)> j
```

```
N> 50
Bis 50 sind folgende Primzahlen vorhanden:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
weiter(j), beenden(e)> e
*** E N D E ***
```

3. Aufgabe

Schreiben Sie ein Java-Konsolprogramm, welches eine Basis und einen Exponenten einliest. Berechnen Sie das Ergebnis

$$\text{Ergebnis} = \text{Basis}^{\text{Exponent}}$$

ohne Verwendung der Math-Klasse (speziell nicht Math.pow()). Die Basis soll vom Typ **double**, der Exponent vom Typ **int** (+/-) sein. Geben Sie das Ergebnis aus. Beachten Sie die mathematische Bedeutung, wenn der Exponent 0 oder negativ ist. Dialogbeispiel:

```
Basis> 23
Exponent> 4
23,000000 potenziert mit 4 ist 279841,000000
weiter(j), ende(e)> j
Basis> 1.8
Exponent> 17
1,800000 potenziert mit 17 ist 21859,115597
weiter(j), ende(e)> j
Basis> 12
Exponent> -5
12,000000 potenziert mit -5 ist 0,000004
weiter(j), ende(e)> j
Basis> 12
Exponent> 0
12,000000 potenziert mit 0 ist 1,000000
weiter(j), ende(e)> e
*** E N D E ***
```

4. Aufgabe

a) Entwickeln Sie ein Programm, das den Anwender auffordert, zwei Zahlen einzugeben und danach eines der vier Zeichen +, -, / und *. Je nach eingegebenem Zeichen sollen die beiden Zahlen addiert, subtrahiert, dividiert oder multipliziert werden. Das Ergebnis der jeweiligen Rechenart soll auf den Bildschirm ausgegeben werden. Entwickeln Sie dazu eine Prozedur, welcher die zwei Zahlen und die gewählte Operation übergeben werden. Diese berechnet und gibt das Ergebnis aus.
Bei Eingabe einer unzulässigen Operation erfolgt eine Fehlermeldung.

b) Erstellen Sie eine weitere Programmvariante. Ersetzen Sie die Prozedur durch eine Funktion, welche das berechnete Ergebnis zurückgibt. Erstellen Sie dazu eine Funktion, die den Operator (+, -, *, /) auf Korrektheit testet. Alternativ zu „/“ soll auch die Eingabe von „:“ möglich sein. Falls die eingegebene Operation nicht korrekt ist, erfolgt eine Fehlermeldung und die Eingabe der Operation wird erneut angefordert. Die Ergebnisausgabe erfolgt dann in der `main`-Methode des Programms.

Dialogbeispiele:

Variante a	Variante b

`Argument 1> 23`
`Operation(+, -, *, /)> *`
`Argument 2> 11`
`23,000000 * 11,000000 = 253,000000`
`weiter(j), ende(e)>`
	`Argument 1> 3.4`
	`Operation(+, -, *, /, :)> :`
	`Argument 2> 1.8`
	`3,400000 : 1,800000 = 1,888889`
	`weiter(j), ende(e)> j`
	`Argument 1> 6`
	`Operation(+, -, *, /, :)> #`
	`Argument 2> 4`
	`unzulässige Operation: #`
	`weiter(j), ende(e)>`

5. Aufgabe

Modifizieren Sie Aufgabe 1 so, dass eine beliebige positive ganze Zahl eingelesen und deren Ziffern in Worten ausgegeben werden. Hinweis: machen Sie aus der Lösung der Aufgabe 1 eine Methode Ihres Programms. Dialogskizze:

```
Zahl> 752434
sieben fünf zwei vier drei vier
weiter(j), ende(e)>
```

6. Aufgabe optional

Modifizieren Sie Aufgabe 1 so, dass bei Eingabe der Wörter `null, … neun`, die korrespondierende Ziffer zurückgegeben wird. Verwenden Sie dazu einen **switch.**

7. Aufgabe

Modifizieren Sie das Programm der Übung 4/Aufgabe 5 folgendermaßen:

- Es sollen nun beliebige `long`-Zahlen eingebbar sein.
- Es soll auswählbar sein, ob die Zahl gedreht oder die Quersumme berechnet wird.
- Für das Zerlegen, Drehen und Berechnen der Quersumme sollen nun drei Funktionen verwendet werden:
  - `static int[] zerlege(long zahl){ . . . }`
    //long-Zahl wird übergeben, ein Feld mit ihren Ziffern zurückgegeben
  - `static long drehe(int[] ziffern){ . . . }`
    //Ziffern-Feld wird übergeben, daraus eine gedrehte long-Zahl rekonstruiert
  - `static long quersumme(int[] ziffern{ . . . }`
    //Feld wird übergeben, daraus die Summe berechnet