Hochschule für Angewandte Wissenschaften Hamburg
University of Applied Sciences Hamburg
Fakultät Technik und Informatik
Department Informatik

Vorbemerkung zum 2.Aufgabenzettel und allen weiteren Aufgabenzetteln

- Sofern die Benutzung von Code-Templates eingefordert wird oder einfach nur Code-Templates zur Verfügung gestellt werden, finden Sie diese - wie Sie es ja auch schon vom 1. Aufgabenzettel kennen – an der üblichen Stelle bzw. unter
  https://users.informatik.haw-hamburg.de/~schafers/LOCAL/S17W_P1/CODE/LAB/Aufgabenzettel2_v*/
  Dies gilt nur, falls es auch Code-Templates gibt ;-)

Aufgaben A2.1 bis A2.3 (bzw. A2.6) Kleine Übungsaufgaben zu Kontrollstrukturen

Überlegen Sie sich zunächst, wie Sie die folgenden Aufgaben lösen wollen. Halten Sie Ihre Gedanken fest und bereiten Sie sich darauf vor, dass Sie auch die Idee Ihrer Lösung (Ihres Algorithmus) erklären sollen und dass dabei kein konkreter Java-Code "vorgelesen" werden darf. Beispiel für A2x6: Ihr kleines Geschwisterchen ist noch in Grundschule und kann sehr gut Mathe. Es will seinen Lehrer beeindrucken und in der Schule Zahlen in seine Primfaktoren zerlegen. Ihr kleines Geschwisterchen fragt Sie, wie das geht. Was erklären Sie ihm? (Ihr Geschwisterchen kann kein Java).

Achtung:
Bei den Lösungen zu diesen Aufgabenzettel wird ausdrücklich darauf Wert gelegt, dass die richtige Kontrollstruktur an der richtigen Stelle verwendet wird.

Weiterhin gilt, dass "Mittel" sinnvoll eingesetzt werden sollen. Dies gilt insbesondere für noch nicht in der Vorlesung besprochene "Dinge". Gehen Sie davon aus, dass noch nicht besprochene "Dinge" verboten sind oder fragen Sie im Vorfeld nach.

Teilweise wird eine Klasse ProposalForAlternativeTestFrame gestellt. Es nicht nur wichtig funktionierenden Code entwickeln zu können, sondern auch reproduzierbare Test entwickeln zu können, die belegen, dass der erstellte Code funktionstüchtig ist. Die Klasse ProposalForAlternativeTestFrame soll Ihnen die Möglichkeit geben, dass Erstellen eigener Tests zu üben.

Teilweise wird eine Klasse UnitTestFrame gestellt. Intention dieser Klasse ist Ihnen zu helfen sicherzustellen, dass Ihre Lösung korrekt ist. Aber Intention ist ausdrücklich **NICHT** Ihnen zu helfen Fehler zu finden.
U.U. brauchen Sie Unterstützung durch einen Betreuer um UnitTestFrame ausführen zu können.

Aufgabe A2.1 Notenpunkte in Schulnote umrechnen

Entpacken Sie das Code-Template `A2x1.zip` und lösen Sie die Aufgabe A2.1.
Nach dem Entpacken finden Sie 3 Klassen vor:

- GradeConverter ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.
- Für UnitTestFrame siehe einleitenden Text

Früher hatten/bekamen Sie (vermutlich) die Schulnoten 1,2,3,4,5 und 6.
Jetzt haben/bekommen Sie Notenpunkte 15, .., 0.
Rechnen Sie Notenpunkte in Schulnoten um.

1 NP	entspricht z.B. der Schulnote	"**5-**".
8 NP	entspricht z.B. der Schulnote	"**3**".
12 NP	entspricht z.B. der Schulnote	"**2+**".

Aufgabe A2.2 Fibonacci–Zahlen

Entpacken Sie das Code-Template `A2x2.zip` und lösen Sie die Aufgabe A2.2.
Nach dem Entpacken finden Sie 3 Klassen vor:

- FibonacciNumberPrinter ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.
- Für ProposalForAlternativeTestFrame siehe einleitenden Text.

Die Fibonacci-Reihe ist eine unendliche Folge von ganzen positiven Zahlen f_0, f_1, f_2, ...
(Siehe http://de.wikipedia.org/wiki/Fibonacci-Folge) Die ersten beiden Zahlen sind:

$f_0 = 0$,
$f_1 = 1$,

Jede weitere Zahl ist die Summe der beiden Vorgängerzahlen :

$f_n = f_{n-2} + f_{n-1}$ für n >= 2

Der Anfang der Fibonacci-Reihe lautet also: 0, 1, 1, 2, 3, 5, 8, ...
Geben Sie die ersten n Werte aus – n soll in einer Variablen **n** vom geeigneten Typ abgelegt werden.
Die ausgegeben Zahlen sind durch Komma zu trennen. Vor der ersten Zahl und nach der letzten Zahl darf jedoch kein Komma stehen.

Bedenke: Was passiert z.B. bei Eingaben/Startwerten wie 1, 0 oder gar -1 ?

Es sei bemerkt, dass -warum auch immer- die "ursprüngliche" Fibonacci-Reihe wie folgt beginnt:

$f_1 = 1$,
$f_2 = 1$,

Es steht Ihnen frei ob Sie die "ursprüngliche" oder die in Wikipedia beschriebene Fibonacci-Folge implementieren. Nur sagen Sie dies vor der Abnahme deutlich an und setzen Sie es konsequent um.

Aufgabe A2.3 Tannenbaum

Entpacken Sie das Code-Template `A2x3.zip` und lösen Sie die Aufgabe A2.3.
Nach dem Entpacken finden Sie 3 Klassen vor:

- FirPrinter ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.
- Für ProposalForAlternativeTestFrame siehe einleitenden Text.

Ergänzen Sie die Methode **`printFir()`** um Code, der einen Tannenbaum ausgibt (ok, es ist mehr ein Dreieck, aber mit etwas Fantasie, kann man einen Tannenbaum erkennen ;-) Die Ausgabe soll abhängig von einer vorgegebenen Höhe (**`height`**) sein. Sie sollen wie in den nachfolgenden Beispielen Sterne für den Tannenbaum und Punkte für die Luft ausgeben. Um in der Aufgabenstellung nicht bereits schon die Lösung vorzugeben, schauen Sie bitte auf die nachfolgenden Beispiele und entnehmen diesen, die Regeln, wie ein Tannenbaum aussehen soll. Die Farben dienen nur der besseren Lesbarkeit. Ihre Ausgabe soll bezogen auf die Farbe(n) "normal" sein.

Beispiel(e):

Ein Tannenbaum der Höhe 1:

```
*
```

Ein Tannenbaum der Höhe 2:

```
.*.
***
```

Ein Tannenbaum der Höhe 3:

```
..*..
.***.
*****
```

Ein Tannenbaum der Höhe 4:

```
...*...
..***..
.*****.
*******
```

Ein Tannenbaum der Höhe 5:

```
....*....
...***...
..*****..
.*******.
*********
```

Und entsprechend für größere Werte.

Freiwillige Zusatzaufgaben

Es folgen freiwillige Zusatzaufgaben. D.h. diese Aufgabe ist freiwillig ;-).
Wenn Sie diese freiwillige Zusatzaufgabe freiwillig lösen, dann haben Sie den "Gewinn", dass Sie mehr geübt haben und dass Sie Ihre Lösung für diese freiwillige Zusatzaufgabe im Labor besprechen können (sofern Zeit ist – Pflichtaufgaben haben Vorrang).

Freiwillige Zusatzaufgabe Z2.1 Sportmedaille

Entpacken Sie das Code-Template `Z2x1.zip` und lösen Sie die Aufgabe Z2.1.
Nach dem Entpacken finden Sie 2 Klassen vor:

- SportMedalComputer ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.

Schreiben Sie ein Programm zur (Sport-)Wettkampfauswertung. Nach einem Sportwettkampf müssen Medaillen vergeben werden. Junge Teilnehmer (max. 13 Jahre) bekommen

- eine Goldmedaille für 4000 oder mehr Punkte
- eine Silbermedaille für 3000-3999 Punkte
- eine Bronzemedaille für weniger als 3000 Punkte

Ältere Teilnehmer (ab 14 Jahre aufwärts) bekommen

- eine Goldmedaille für 5000 oder mehr Punkte
- eine Silbermedaille für 4000-4999 Punkte
- eine Bronzemedaille für weniger als 4000 Punkte

Berechnen Sie für einen Teilnehmer mit gegebenen Alter (in Jahren) und gegebenen Punktestand welche Medaille er/sie bekommt.

Freiwillige Zusatzaufgabe Z2.2 Umrechnung Celsius zu Fahrenheit

Entpacken Sie das Code-Template `Z2x2.zip` und lösen Sie die Aufgabe Z2.2.
Nach dem Entpacken finden Sie 2 Klassen vor:

- TemperatureConverter ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.

Erstellen Sie eine 2-spaltige Tabelle, in der in der ersten Spalte die Temperatur in Celsius ausgegeben wird (0 – 100 Grad Celsius, mit der Schrittweite 5 Grad) und in der zweiten Spalte die entsprechende Gradzahl in Fahrenheit.
Zwischen Grad Celsius und Grad Fahrenheit gibt es die Umrechnung:

```
Grad-Fahrenheit = Grad-Celsius * 9/5 + 32.
```

Da mit dieser Aufgabe insbesondere die Effekte des Numeric Promotion *(das Finden eines „gemeinsamen“ Types der Operanden für das Ausführen einer Operation)* geübt werden sollen, ist das Ergebnis der Umrechnung in einer ganzzahligen Variablen abzuspeichern, die dann ausgegeben werden soll.
Aus der Aufgabenstellung folgt, dass nur ganzzahlige Werte umgerechnet werden sollen. Celsius ist also immer ganzzahlig.
Die Ausgabe soll wie folgt aussehen:

```
Temperatur-Umrechnungstabelle
=============================
C       F
---------
0      32
5      41
10     50
15     59
...
```

Achtung!:
Die Ausgabe soll kaufmännisch gerundet (bzw. entsprechend nachfolgender Forderung auf- bzw. abgerundet) sein. D.h. ein Nachkommawert < ,5 wird abgerundet und ein Nachkommawert ≥ ,5 wird aufgerundet
Dies sollen Sie selbst(!) sicherstellen – das Math-Package darf nicht verwendet werden.

Freiwillige Zusatzaufgabe Z2.3 Bitanzahl

Entpacken Sie das Code-Template `Z2x3.zip` und lösen Sie die Aufgabe Z2.3.
Nach dem Entpacken finden Sie 3 Klassen vor:

- RequiredBitSizeComputer ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.
- Für ProposalForAlternativeTestFrame siehe einleitenden Text.
- Für UnitTestFrame siehe einleitenden Text

Bestimmen Sie für einen beliebigen positiven[1] ganzzahligen Wert $\mathbf{w}\in\mathbf{N_0}$ vom Typ **`long`**, die Anzahl der Bits, die min. nötig sind um diese Zahl binär zu codieren (unsigned bzw. als "normale" Dualzahl). Oder genauer: Die Anzahl der Bits, die mindestens nötig sind um alle $\mathbf{n}\in\{ z\in\mathbf{N_0} \mid z\in[0,\mathbf{w}] \}$ binär zu codieren – also auch z.B. kein Zweierkomplement.
Die Berechnung muss korrekt sein - es sind keine Rundungsfehler erlaubt!

Beispiele:

Für den Wert	`0`	lautet das Ergebnis	`1` Bit.
Für den Wert	`1`	lautet das Ergebnis	`1` Bit.
Für den Wert	`2`	lautet das Ergebnis	`2` Bit.
Für den Wert	`255`	lautet das Ergebnis	`8` Bit.
Für den Wert	`256`	lautet das Ergebnis	`9` Bit.
Für den Wert	`140737488355328`	lautet das Ergebnis	`48` Bit.
Für den Wert	`281474976710655`	lautet das Ergebnis	`48` Bit.
Für den Wert	`9223372036854775807`	lautet das Ergebnis	`63` Bit.

[1] Positiv im für "Programmierer" üblichen Sinn (und mathematisch falsch ;-) also "≥0" bzw. einschließlich 0.

Freiwillige Zusatzaufgabe Z2.4 Primfaktorzerlegung

Entpacken Sie das Code-Template `Z2x4.zip` und lösen Sie die Aufgabe Z2.4.
Nach dem Entpacken finden Sie 3 Klassen vor:

- PrimeFactorPrinter ist ein Code-Template in dem Sie Ihre Lösung einbauen sollen
- TestFrame ist eine Möglichkeit Ihre Lösung anzustarten und zu testen.
- Für ProposalForAlternativeTestFrame siehe einleitenden Text.
- Beautician ist nötig für interne Dinge und braucht Sie nicht weiter zu interessieren. (Sie dürfen aber gerne reinschauen und lernen – jedoch ist der Code in einigen Teilen vermutlich für Sie unverständlich)

Berechnen Sie die Primfaktorzerlegung für eine positive Zahl, die in einer Variablen **`zahl`** vom Typ long abgelegt ist. Die Ausgabe soll wie im Beispiel dargestellt aussehen. Also:
<Wert> "=" {<Primfaktor> "*"} <Primfaktor>

Beispiele:

```
                  12 =  2*2*3
                  13 =  13
                  15 =  3*5
                  28 =  2*2*7
                 768 =  2*2*2*2*2*2*2*3
          1080664428 =  2*2*3*90055369
         51539607816 =  2*2*2*3*2147483659
         65498029444 =  2*2*37*12689*34877
      37513856612736 =  2*2*2*2*2*2*2*3*3*3*3*3*7*7*7*17*17*23*23*23
     950052134362500 =  2*2*3*3*3*5*5*5*5*5*13*13*23*23*23*37*37
9223372036854775549 =  9223372036854775549
9223372036854775643 =  9223372036854775643
9223372036854775673 =  175934777*52424950849
9223372036854775771 =  19*485440633518672409
9223372036854775777 =  584911*15768846947407
9223372036854775782 =  2*3*3*3*3*17*23*319279*456065899456065899
9223372036854775783 =  9223372036854775783
9223372036854775797 =  3*3074457345618258599
9223372036854775807 =  7*7*73*127*337*92737*649657
```

Achtung: Die Ausgabe schließt mit einem Primfaktor ab und nicht dem Multiplikationszeichen "*"
Bedenke: Was passiert z.B. bei Startwerten wie 2, 1, 0 oder -1 ?

Zusatz-Teil-Aufgabe:

Zerlegen Sie die Zahl 9223372036854775643 in ihre Primfaktoren.
In welches Problem laufen Sie (möglicherweise) dabei? (Dauert es vielleicht etwas lange ;-)
(Nur) für diese (Zusatz-Teil-)Aufgabe dürfen Sie auf *Math.sqrt()* zurückgreifen.
Da die Berechnung einer Wurzel **sehr aufwendig** ist, sollten Sie dies nicht zu oft tun.
Konkret: Für eine Primfaktorzerlegung sollte nur einmal ("am Anfang") eine Wurzel gezogen werden ;-)

Unterstützende Fragen:
Ein Zahl zerfalle in 2 Faktoren f1 und f2 bzw.: **`zahl = f1 * f2`**
Sofern f1 größer der Wurzel ist, was gilt dann für f2? Bzw.: **`f1 > wurzel`** ⇒ **…f2… ?**
Wie lassen sich die zugehörigen "Erkenntnisse" für diese (Zusatz-Teil-)Aufgabe nutzen?
Ist es kritisch, dass **`wurzel`** nur der ganzzahlige Anteil der Wurzel ist?

Syntax-Beispiel:

```
long wurzel = (long)( Math.sqrt(zuZerlegendeZahl) );  // ganzzahlige Anteil der Wurzel
```