Laborübung
Klasse: 1BHIF
Datum: 09.01.25

1) Implementiere die folgenden Methoden der Klasse **Berechnungen** und teste sie.

xHochN (float x) … lies mit Hilfe der Scanner Klasse eine ganze Zahl n ein und berechnet x^n. Achtung! Die Variable n kann auch einen negativen Werte erhalten und dann verhält sich die Rechnung anders:

$$x^{-n} = \frac{1}{x^n}$$

folge1(int a, int b) bekommt 2 Parameter. Die Parameter müssen positiv sein (wandle sie allenfalls in positive Zahlen um. Achte weiters darauf, dass der erste Parameter grösser ist als der zweite – anderenfalls müssen die beiden getauscht werden. Die Methode gibt entsprechend der Parameterwerte folgende Zahlen/Zeichenfolgen auf dem Bildschirm aus:

folge1(50,40); gibt folgendes auf dem Bildschirm aus:
-1/50 +3/48 -9/46 +27/44 -81/42 +243/40

folge1(47,40); gibt folgendes auf dem Bildschirm aus:
-1/47 +3/45 -9/43 +27/41

folge1(2,6); gibt folgendes auf dem Bildschirm aus:
-1/6 +3/4 -9/2

a) Was gibt *folge1(55,33)* aus?

b) Analysiere das Verhalten der Methode (kurze Beschreibung als Kommentar im Sourcecode)

c) Implementiere die Methode

ziffernsumme(int zahl) berechnet die Ziffernsumme der Zahl. Die Berechnung erscheint am Bildschirm, das Resultat wird zurückgegeben.
z.B.:
ziffernsumme(234)→2+3+4 -> return: 9
ziffernsumme(17)→1+7 -> return: 8

Wird eine negative Zahl übergeben, so wird diese zu einer positiven Zahl umgewandelt

void falling(int startHoehe, int zeit) berechnet die Fallgeschwindigkeit eines Handys:

Ein Handy wird von einer Stelle (Klippe, Donauturm, Ballon…) in die Tiefe geworfen. Die Schwerkraft wirkt auf das Handy und lässt es immer schneller werden. Die Entfernung zwischen dem Handy und der Stelle, an der es fallen gelassen wurde, beträgt nach x Sekunden

$(1/2)*G*x^2$ Meter (x ist die Anzahl an Sekunden, die das Handy fällt,
G ist eine Konstante: 9.80665).

Nach 0 Sekunden hat das Handy 0 Meter zurückgelegt.

Nach 1 Sekunde hat das Handy (1/2) * 9.80665 * $(1)^2$ Meter zurückgelegt, das sind 0.5 * 9.80665 * 1 = 4.903325 Meter.

Nach 2.0 Sekunden hat das Handy (1/2)* 9.80665 * $(2)^2$ Meter zurückgelegt, das sind 0.5 * 9.80665 * 4 = 19.6133 Meter.

.... usw .....

Hierbei soll so lange laufend die Zeit und die Entfernung von der "Wurf"-Stelle ausgegeben werden, bis entweder die Zeit abgelaufen ist oder das Handy am Boden aufgeschlagen ist.

Beispiele:

Aufruf **falling(500,3)** führt zu folgender (beispielhafter) Ausgabe:

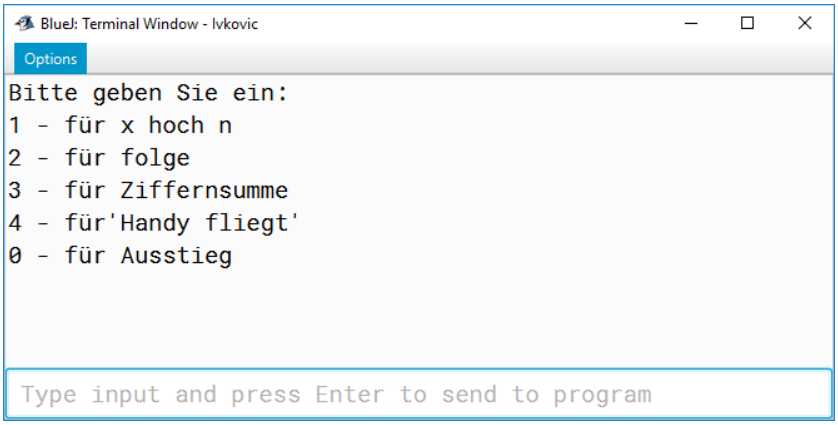
```
          Sekunden  ->  Entfernung (Abwurf-Höhe: 500)
          --------  ->  ----------------------------
             0      ->       0.0
             1      ->       4.903325
             2      ->       19.6133
             3      ->       44.129925

Versuchs-Abbruch; verbleibende Resthöhe: 455.870075 Meter
```

Aufruf **falling(100,5)** führt zu folgender (beispielhafter) Ausgabe:

```
          Sekunden  ->  Entfernung (Abwurf-Höhe: 100)
          --------  ->  ----------------------------
             0      ->       0.0
```

```
1       ->      4.903325
2       ->      19.6133
3       ->      44.129925
4       ->      78.4532
5       ->      Handy kaputt
```

menu() gibt dem Benutzer ein Menü zur Auswahl. Je nach Eingabe kann der Benutzer eine der implementierten Methoden aufrufen. Falls die Methode Parameter braucht, werden diese auch von ihm verlangt (verwende dafür die entsprechende Methode der *Scanner* Klasse).
Das Menu ermöglicht dem Benutzer so oft Methoden aufzurufen wie er möchte – erst bei der Eingabe von 0 steigt er aus dem Menü (=> der Schleife) aus. Ungültige Werte, die nicht im Menü vorgesehen sind (z.B. 8 oder 10), werden vom Programm ignoriert. D.h. der Benutzer wird wieder aufgefordert ein Zahl einzugeben,



2) Die Ungenauigkeit der Gleitkommazahlendarstellung führt bei einigen simplen Berechnungen zu Problemen. Z. B ergibt nachfolgender Code bei seiner Ausführung eine Endlosschleife. Gehe mit dem Debugger durch diese Endlosschleife und schreibe die Werte der Variable d der ersten 10 Durchläufe in ein Worddokument mit deinem Nachnamen. Sinn der Übung ist vor allem zu lernen wie man den Debugger verwendet!

```
public void test()
{
        double d = 0.0;
        while(d != 1.0)          // Solange d ungleich 1
        {
                d += 0.1;        // erhöhe d um 0.1
        }
}
```

Abzugeben ist die Klasse ***Berechnungen.java*** – nachdem du sie eingehend getestet hast und das Worddokument von Übung 2!