Cours MOOC EPFL d'initiation à la programmation illustré en Java

Third assignment
Loops

J. Sam & J.-C. Chappelier

1 Exercise 1 — Playing with numbers

Description

Let's consider the folowing operations on a given integer:

- if the number is divisible by 3, 4 is added to it;
- if it is not, but is divisible by 4, it is divided by 2;
- if it is divisible neither by 3 nor by 4, 1 is substracted from it.

These operations are repeated successively until the result is zero. Concretely, if we start from an integer n_0, the operations are applied to n_0 which gives the number n_1; then if n_1 is different from zero, the operations are applied to n_1 and son on, until we reach the number n_k with value 0. For example if we start from 7, the following suite of number is generated: 6, 10, 9, 13, 12, 16, 8, 4, 2, 1 and0. The operations are repeated $k = 11$ times. If we start from 1, the value zero is immediately reached and the number of repetitions is then $k = 1$. You are asked to write a program which displays how many times the operations must be repeated until zero is reached. The number of repetitions must be displayed for any starting number comprised between two integers read from the keyboard. For example, for all starting numbers lying between 1 and 7, your program must produce the following output:

```
1 -> 1
2 -> 2
3 -> 12
4 -> 3
5 -> 4
```

```
6 -> 10
7 -> 11
```

In this display, 1 and 7 (in the first column) are the limits read from the keyboard. The format `i -> j` means that if we start with the number `i`, `j` repetitions are required to reach 0; for example 11 repetitions are needed if we start from 7. If the integer read from the keyborad are 99 and 100, the output of your program must be:

```
99 -> 18
100 -> 17
```

Notice: testing if a number `n` is divisible by `p` amounts to testing if `n % p` is zero.

Procedure

Download the source code `Suite.java` available at the course webpageand complete it.

WARNING: you should modify neither the beginning nor the end of the program, only add your own lines as indicated. It is therefore very important to respect the following procedure (the points 1 and 3 concern only Eclipse users):

1. remove automated formatting of the code in Eclipse:

   `Window > Preferences > Java > Editor > Save Actions` (and untick the formatting option if it's on);

2. save the downloaded file as `Suite.java` (respect the upper case). If you work with Eclipse, save it to `projectFolderUsedForThatExercise/src/`;

3. refresh the Eclipse project where the file is stored (right click on the project > "refresh") in order to take into account the new file;

4. write your code between these two provided comments:

```
/*****************************************
 * Completez le programme a partir d'ici.
 *****************************************/

/*****************************************
 * Ne rien modifier apres cette ligne.
 *****************************************/
```

5. save and test your program to be sure that it works properly, for example using the values given below;

6. upload the modified file (still named `Suite.java`) in "OUTPUT submission" (not in "Additional"!).

2 Exercise 2 — Parachuting jump

2.1 Introduction

The focus here is to write a program that allows to calculate the parameters of the fall of a skydiver.

Download the source code `Parachutiste.java` available at the course webpageand complete it.

WARNING: you should modify neither the beginning nor the end of the program, only add your own lines as indicated. It is therefore very important to respect the following procedure (the points 1 and 3 concern only Eclipse users):

1. remove automated formatting of the code in Eclipse:

   `Window > Preferences > Java > Editor > Save Actions` (and untick the formatting option if it's on);

2. save the downloaded file as `Parachutiste.java` (respect the upper case). If you work with Eclipse, save it to `projectFolderUsedForThatExercise/src/`;

3. refresh the Eclipse project where the file is stored (right click on the project > "refresh") in order to take into account the new file;

4. write your code between these two provided comments:

```
/*****************************************
 * Completez le programme a partir d'ici.
 *****************************************/

/*****************************************
 * Ne rien modifier apres cette ligne.
 *****************************************/
```

5. save and test your program to be sure that it works properly, for example using the values given below;

6. upload the modified file (still named `Parachutiste.java`) in "OUTPUT submission" (not in "Additional"!).

2.2 Modeling the skydiver

The skydiver will be represented in the program by his mass, his falling speed, his acceleration, his altitude and the surface of his body exposed to the resistance of the air (this one will vary when the parachute opens).

In the downloaded program:

1. start by declaring in the method `main` a constant `g` of value 9.81, and two variables of the type `double` (that will be modified when the parachute opens): `v0`, initialized with 0 and `t0`, initialized with 0.

2. define next the variables necessary for the description of the skydiver as given above, apart from the mass that we have already defined (look at the beginning of the method `main` in the downloaded file): `vitesse` for the speed, `hauteur` for the altitude, `accel` for the acceleration and `t` for the time.

   We will initialize the surface of the skydiver at 2.0 m^2, his altitude with the value of `h0`, his speed with the value of `v0` and his acceleration with the value of `g`.

   Lastly define a variable `t` initialized with the value of `t0`.

3. Print the initial values as they were defined right above using *strictly* the following line (also provided in the downloaded file):

   ```
   System.out.printf("%.0f, %.4f, %.4f, %.5f\n",
                     t, hauteur, vitesse, accel);
   ```

   with the initial values given above, a mass of 80 kg and a starting altitude of 39'000 m, the program prints at this stage:

   ```
   0, 39000.0000, 0.0000, 9.81000
   ```

2.3 Free fall

In order to calculate the evolution of the athlete in free fall we will need the two following expressions:

- s which is the surface of the athlete divided by his mass;
- a "term" named q and having value $q = \exp(-s \times (t - t_0))$, where t represents the current time and t_0 the initial time of the fall, initialized with 0 in the previous question.

  Notice: the function exp is simply written `Math.exp` in Java, for example: `Math.exp(x)`.

The evolution of the athlete in free fall is therefore expressed as follows:

$$v(t) = \frac{g}{s} \times (1 - q) + v_0 \times q$$
$$h(t) = h_0 - \frac{g}{s} \times (t - t_0) - \frac{v_0 - g/s}{s} \times (1 - q)$$
$$a(t) = g - s \times v(t)$$

where v is the speed of the athlete, h his altitude, a his acceleration, $g = 9.81$, and v_0, h_0 and t_0 correspond to the three variables defined in the previous question [1].

We ask you to complete your previous program in order to calculate the evolution of the fall of the athlete as it was initialized in the previous question: do the calculation, second to second (which means adding every time `1` to the time `t`), as long as the athlete has not reached the ground, which means as long as his altitude `h` is positive.

Print the statistics of the athlete at every second at every second respecting the format of the previous question.

Test your program with a mass of 80 kg and a starting altitude of 39'000 m; it should give the following results:

```
0, 39000.0000, 0.0000, 9.81000
1, 38995.1356, 9.6884, 9.56779
2, 38980.7030, 19.1376, 9.33156
3, 38956.9382, 28.3535, 9.10116
...
137, 426.3065, 379.6277, 0.31931
138, 46.5205, 379.9430, 0.31142
```

[1] They can subsequently change, so even if some of them are zero for now, it is important to keep explicitly all of them in the calculated formulas.

2.4 Speed of sound and speed limit

You are now asked to extend your previous program so that:

5. when the speed of the athlete exceeds the speed of sound (343 m/s), the program prints (additionally, but *only once*) the following message:

   ```
   ## Felix depasse la vitesse du son
   ```

   This message should be printed *BEFORE* the information regarding the time, the altitude, the speed and the acceleration :

   ```
   ...
   82, 20498.5770, 341.8844, 1.26289
   ## Felix depasse la vitesse du son
   83, 20156.0663, 343.1317, 1.23171
   ...
   ```

6. as long as his acceleration is lower than 0.5 m/s^2, the program prints (additionally, but *only once*) the following message:

   ```
   ## Felix a atteint sa vitesse maximale
   ```

   This message should be printed *BEFORE* the information regarding the time, the altitude, the speed and the acceleration:

   ```
   ...
   119, 7199.1595, 372.3690, 0.50078
   ## Felix a atteint sa vitesse maximale
   120, 6826.5422, 372.8636, 0.48841
   ...
   ```

For testing: with the previous values (80 kg and 39'000 m), the speed of the sound is reached after 83 s and the maximum speed ($\simeq$ 372 m/s) after 120 s as shown in the two examples above.

2.5 Opening of the parachute

We finally ask you to extend one last time your previous program so that when the altitude of the athlete is smaller than 2500 m, the program changes the value of the surface of the athlete from 2.0 m^2 (before the opening of the parachute) to 25.0 m^2 (after the opening of the parachute). You must also change the "initial conditions" `t0`, `v0` and `h0` with the current values of the athlete (so that the equations of the evolution will be correct for the rest of the fall).

Additionally, your program must print the following message:

```
## Felix ouvre son parachute
```

This message should be printed *BEFORE* the information regarding the time, the altitude, the speed and the acceleration:

```
...
131, 2698.0264, 377.5607, 0.37098
## Felix ouvre son parachute
132, 2320.2818, 377.9270, 0.36182
133, 1991.2751, 284.9225, -79.22827
...
```

Note that the acceleration becomes negative *two* lines after this message is printed.

For testing: with the previous values (80 kg and 39'000 m), the parachute is open after 132 s and the simulation terminates after 170 s:

```
...
131, 2698.0264, 377.5607, 0.37098
## Felix ouvre son parachute
132, 2320.2818, 377.9270, 0.36182
133, 1991.2751, 284.9225, -79.22827
...
170, 18.4814, 31.3944, -0.00075
```

Finaly, pay attention where you add the line(s) to "open the parachute"; in particular, the skydiver does not start jumping with a parachute already open!..