Exercise 1 — BMI

The goal of this exercise is to create « patients » that have a weight and a height, and to compute their « Body Mass Index » (BMI)

The provided code does the following things:

- creates a patient,
- displays patient data and his BMI.

These two steps are repeated twice with different values.

A patient is characterized by a weight and a height. You will name the corresponding attributes respectively mass and height.

Methods specific to a patient are:

- a method init taking two double as parameters, to initialize the patient weight and his height;
- these data will be assigned to the patient only if they are positives; if not, weight and height will be initialized to zero; in order to simplify, there won't be any other check on these data;
- a method *display* that displays on the terminal the attributes of the patient respecting the following format:

  Patient: <weight> kg pour <height> m

  where <weight> is to be replaced by the weight of the patient and <height> by his height ; this output will be ended by a line break. The output of the decimal numbers will have a one-digit precision for the weight and the height.
- a getter returning the patient's weight;
- a getter returning the patient's height;
- a method *bmi* returning the BMI of the patient: his weight divided by the square of his height; if the height is zero, the method will return zero.

Execution example:

```
Patient: 74.5 kg pour 1.8 m
BMI: 24.3265306122449
Patient: 0.0 kg pour 0.0 m
```

Exercise 2 — Piggy bank

The goal of this exercise is to simulate a piggy bank in which we deposit and withdraw money. We wish to use it to pay some specific amounts.
The provided code creates a piggy bank and manipulates it (empty it, shake it, display its content etc.). This program also asks the user the budget he wishes to put in his/her holidays.

If the piggy bank contains enough money (this budget or more), it tells how much money would be left after the holidays. In the opposite case, it indicates the amount missing to go on holidays with the desired budget.
You are asked to provide the implementation of the class missing PiggyBank.
A piggy bank is characterized by the *amount* it contains. You will name this attribute with the name *amount* in your code.
Its specific procedures are:

- a getter getAmount returning the amount of the piggy bank
- a method display displaying the information about the piggy bank in the following format:
  "you are penniless" if the piggy bank is empty
  "You have: <amount> dollars in your piggy bank" in the opposite case
- method shake displaying in the terminal the message "Bing bing", followed by a line break, in the case where the piggy bank is not empty, and does display anything otherwise;
- a method fill adding a given amount, given as double parameter, in the piggy bank. Only the positive amounts will be accepted (otherwise we don't do anything);
- a method empty (re)initializing the amount to zero;
- a method *draw* allowing to withdraw in the piggy bank an amount given as parameter. If the amount is negative it will be ignored. If the amount passed as parameter is bigger than the amount in the piggy bank, the piggy bank is emptied. The *draw* method does not return anything.
- A method calculateBalance that returns the difference between the amount in the piggy bank and the budget we are willing to spend (a double). If the budget is negative (or zero), the method calculateBalance will return the piggy bank amount.

Execution examples:

```
You are penniless.
You are penniless.
Bing bing
You have: 550.0 dollars in your piggy bank
You have: 535.0 dollars in your piggy bank

Give the budget for your holiday:
450
You are enough rich to go on holiday
It will be remained 85.0 dollars for you in return
```

Or:

```
Give the budget for your holiday:
1250.0
You have lack of 715.0 dollars to go on holiday
```