## Usage

All functions are packed into class Admin. Usage:

make

see <test.sh> for more details.

# Specification

The program Admin takes an arbitrary number of command-line arguments, and computes various things, depending on the parameter values and specified in the following subsections. In case you can not handle a case, then print out NOT IMPLEMENTED.

Furthermore four public functions have to be provided.

Your test-cases should of course include at least the given cases.

To avoid errors, copyand-paste those examples to the command-line, and carefully check the output.

## General requirements:

1. For the formatting of the output, the given examples have to be (carefully) obeyed. Especially there must not be trailing spaces (invisible spaces at the end of a line).
2. Besides the input-output-behaviour of the (whole) program, now also functions (static methods) have to be provided, which shall be usable from any client (except if stated otherwise), and which must precisely obey the signature and the input-output relation as specified.
3. These functions must also be used in the main-function to solve the tasks.
4. Furthermore the class Person is given (in the usual code-directory), which must not be changed, and which is to be used.
5. No explanations for Person are given — it is your task to read the code and to understand what this class is doing.

## 2.1 The four functions

All these functions are public, and thus must be callable under any circumstances, without ever throwing (raising) an exception (note the possibilities of null). The order of arguments follows the order in the text.

### 2.1.1 change\_income

Takes a Person and an int, increasing the income of Person for positive values and decreasing for negative values. Returned is a boolean, which is false if and only if the income is increased in such a way as to cause int to overflow.

### 2.1.2 equal\_persons

Takes an array of Person’s, returning a boolean, which is true if and only if there are two different slots in the array, where the persons are equal according to the definition in Person.

Note that the appropriate facilities in Personmust be used (this is part of the contract!).

### 2.1.3 sum\_income

Takes an array of Person’s, returning the sum of the incomes, which must always be correct (under all circumstances)

### 2.1.4 longest\_name

Takes an array of Person’s, returning the longest name. For the precise definition note that if a string is returned, then this string must occur in the array! If there are several longest strings, then the first is to be taken.

## 2.2 Zero command-line arguments

This just prints the message inside of Person.

> java Admin  
ehpXdT3tie

As always, nothing is additionally printed, except that the line is completed by end-of-line. And, as always for this coursework, the appropriate functionality of Person must be used (so that when Person is updated, then your code adapts automatically!).

## 2.3 One command-line argument

The argument is taken as name of a person, and the person is printed:

> java Admin XYZ  
XYZ: 0

To emphasise again: You need to use the facilities in Person for that. So that when for example the default income or the way a person is printed is changed, then your code changes automatically.

## 2.4 Two command-line arguments

The arguments are taken as name and income of a person, and the person is printed:

> java Admin XYZ 77  
XYZ: 77

As with the first coursework, when the argument is not an int, then the appropriate exception must be thrown, which is not caught by the program (and thus the program aborts with the Java error-message).

## 2.5 Three command-line arguments

> java Admin XYZ 77 -7  
XYZ: 70  
> java Admin XYZ 77 12  
XYZ: 89  
> java Admin XYZ 77 2147483647  
Failure.

The first two arguments are interpreted as before, while the third argument is taken as argument for the change\_income function (which must be used), where after the change the person is printed.

## 2.6 Otherwise

If we have four or more arguments, then each consecutive pair is taken as specifying a person, where in case of an odd number of arguments the final argument is taken as just a name (to be used with the default income). Three pieces of information are then printed: Whether there are two equal persons, the sum of incomes, and the longest name:

> java Admin XYZ 77 kkk -9  
false  
77  
XYZ  
> java Admin a 1 b 2 a  
true  
3  
a  
> java Admin a 1 b 2 a 3  
true  
6  
a  
> java Admin a 1 b 2 a 3 kk -1  
true  
6  
kk

The appropriate functions must be used.