EXERCICE 3

ATM 3

For this programming exercise, you will write a new program based on the program from exercise 2.

In exercise 2, the ATM managed several accounts by distributing the data needed to represent each client account across several arrays.

In this exercise, you will use classes and files to improve the management of client accounts with a single array.

Class Account

Declare and then use an *Account* class whose header is :

class Account(pinclient : String = "INTRO1234", balanceclient : Double = 1200.0)

And which contains the following attributes:

- *Balance* type *Double*, which represents the amount of the available balance on the client's account
- *pin* of type *String*, which represents the value of the client's pin code

And the following methods ***whose appropriate signature(header) you must determine***:

- *deposit*: The method should read the currency (CHF or EUR), the amount to be deposited - checking that it is divisible by 10 - and add the amount to be deposited, if necessary converted from EUR to CHF, to the amount of the client's account balance.
- *withdraw*: This method shall read out the currency (CHF or EUR), the amount to be withdrawn - checking that it is divisible by 10 and below the limit - and subtract the withdrawal amount, if necessary converted from EUR to CHF, from the client's account balance. It shall also allocate and distribute the denominations for the withdrawal.
- *changepin*: The method must read the new pin code from the keyboard - checking that it contains at least 8 characters - and then update the client's pin code.

Account array

Replace the two arrays Array *accounts* and *codepin* from exercise 2 with a single ArrayBuffer array of objects of the class *Accounts*: *clientaccounts*.

The client ID remains the same as in exercise 2. Each client therefore has an integer identifier. The value of this identifier also corresponds to the client's account in the array *clientaccounts*. Thus, the client with identifier 0 has the account: *clientaccounts(0)*, the client with identifier 1 has the account: *clientaccounts(1)*...

The behaviour of the program with the user is identical to that of exercise 2. Repeat the same operation except for this minor change:

- When the user enters a client ID value that is negative or greater than or equal to nbclients, the program stops.

CSV text file of accounts

The account data of the clients will now be saved in a text file in CSV format: *clients.csv*. You will find a ready-made file containing the data of several client accounts in the replit project Exercise 3 in the section Exercises. You can read the content in the replit editor by clicking on the file *clients.csv* in the left-hand file panel.

Each line of the file represents a client account, with two values separated by a comma. The first value corresponds to the pin code and the second to the amount of the client account balance.

The program will systematically start by reading this file and initializing the clientaccounts array with the data from the csv file.

The variable nbclients from exercise 2 is no longer needed, the number of clients is determined by the number of rows in the clients.csv file.

The data of the client with ID 0 will be stored in the first line of the file. The data of the client with ID 1 will be stored in the second line of the file and so on until the client with ID nbclients-1.

When the program ends, all data for all clients must be systematically saved in the file: clients.csv. This is the case whether or not the data was modified during the program session.

Reminder: The program terminates when the user enters an invalid client ID.

For reading from and saving to the file, declare and use the following two methods:

- *record(clients : Array[Accounts]) : Unit*
- *read() : Array[Accounts]*

Exception handling

Use the exception handling so that the program does not crash at runtime if the file does not exist. If the clients.csv does not exist, the program should simply terminate with the message: "Cannot load client account data. ATM temporarily out of service".