Functional Programming Assignment 1, element 2: Accounting program, code

CP50065E

Alessandro Cuccaro

2021

# Introduction

In the short report I will explain the functionalities of the code written, including eight screenshots showing all the application functions used to create this text-based accounting program in Haskell.

First, we define the data types that we will help us defining our program, we then proceed to create lists such as **AccountList** and **TransactionList.**

We start to populate the list to check the parameters.

Graphical user interface, text, application, email

Description automatically generated

We will call our main function by the command **main** operating on the **ContactList’s** data

Text

Description automatically generated

Our program operation includes **deposit, list withdraw and quit**.

First let’s check the list we populated before with some data.

Text, email

Description automatically generated

The data has been recorded and showed as requested by the application, let’s proceed now to deposit an amount of money

Text

Description automatically generated

Using **handle** we are able to input data and processing it using pattern machine. Now let’s see if the deposit transaction has been recorded inside our program.

Graphical user interface, text

Description automatically generated

The program acknowledged the operation and saved it inside our **TransactionList,** with the **displayTransaction** function we can display all the transactions made until now.

Let’s use the withdraw function now.

Graphical user interface, text, email

Description automatically generated

The function takes in consideration the currency so it can perform the ratio exchange automatically.  
Now we control inside our **TransactionList** and see if the transaction has been recorded.

A picture containing text

Description automatically generated

Now let’s test if the currency exchange works when we deposit money inside our program, it should be able to automatically convert the money to GBP. I will test it depositing 50 USD.

Graphical user interface, text

Description automatically generated

As we see the value of the 4th transaction as automatically changed in GBP resulting the value to change.

I will also include all the source code with my comments and explanation of the code.

-- data types

**import** **System.IO**

**import** **Data.List**

**type** **Name** **=** **String**

**type** **AccountNumber** **=** **String**

**type** **ID** **=** **Int**

**type** **Value** **=** **Float**

**type** **Currency** **=** **String**

**type** **TransactionAmount** **=** **Float**

**type** **ReferenceNumber** **=** **Int**

**type** **Comment** **=** **String**

**type** **Account** **=** (**ID**, **Name**, **AccountNumber**, **Currency**, **Value**)

**type** **AccountList** **=** [**Account**]

**type** **Transaction** **=** (**AccountNumber**, **TransactionAmount**, **ReferenceNumber**, **Comment**)

**type** **TransactionList** **=**[**Transaction**]

-- Initially populated list

**myList** **::** **AccountList**

**myList** **=** [(**1**, "Alessandro", "21381701", "GBP", **100.00**)]

**bank** **::** **TransactionList**

**bank** **=** [("21381701", **50.00**, **1**, "Rent")]

-- "ui": part of main input processing; prompts user for a command,

-- then hands command and contact list to handle function.

-- Contact list must be retained as data (no global variables)

-- input: ContactList (data to operate on)

**main** **=** ui bank

**ui** **::** **TransactionList** **->** **IO** ()

**ui** ts **=** **do**

putStrLn "Welcome to bank, what operation would you like to do? [deposit, list , withdraw, quit]"

cmd **<-** getLine

handle cmd ts

-- "handle": part of main input processing; uses pattern matching to handle input commands

-- input: String (command), TransactionList (data to operate on)

**handle** **::** **String** **->** **TransactionList** **->** **IO** ()

**handle** "quit" ts **=** **do**

putStrLn "Goodbye!"

**handle** "list" ts **=** **do**

displayTransaction ts

ui ts

**handle** "deposit" ts **=** **do**

putStrLn "Please enter account number: "

newAccountNumber **<-** getLine

putStrLn "Please enter a value: "

interimValue **<-** getLine

**let** newValue **=** read interimValue **::** **Float**

putStrLn "Please enter a comment: "

comment **<-** getLine

putStrLn "Please enter currency [USD, EUR, GBP]"

curr **<-** getLine

**let** finalValue **=** depositexchange newValue curr

ui ((newAccountNumber, finalValue, (length ts) + **1**, "Deposit: " ++ comment)**:** ts)

**handle** "withdraw" ts **=** **do**

putStrLn "Please enter account number: "

newAccountNumber2 **<-** getLine

putStrLn "Please enter a value: "

interimValue2 **<-** getLine

**let** newValue2 **=** read interimValue2 **::** **Float**

putStrLn "Please enter a comment: "

comment2 **<-** getLine

putStrLn "Please enter currency [USD, EUR, GBP]"

curr2 **<-** getLine

**let** finalValue2 **=** exchange newValue2 curr2

ui ((newAccountNumber2, finalValue2, (length ts) + **1**, "Withdrawl: " ++ comment2)**:** ts)

**handle** x ts **=** **do**

putStrLn (x ++ " is not recognised.")

ui ts

-- "displayTransaction" prints transactions list on the screen

-- input: TransactionList

**displayTransaction** **::** **TransactionList** **->** **IO** ()

**displayTransaction** **[]** **=** **do**

putStrLn "List of Transaction:"

putStrLn "All final values in GBP"

putStrLn "=================="

**displayTransaction** ((accnumber, tamount, ref, comment)**:**ts) **=** **do**

displayTransaction ts

putStrLn (" Reference: " ++ (show ref))

putStrLn (" Account no: " ++ (show accnumber))

putStrLn (" Value: " ++ (show tamount))

putStrLn (" comment: " ++ comment)

putStrLn "=================="

-- This converts the deposit value into the fixed rate currency exchange

**depositexchange** **::** **Float** **->** **String** **->** **Float**

**depositexchange** x "USD" **=** x / **1.2**

**depositexchange** x "EUR" **=** x / **1.25**

**depositexchange** x "GBP" **=** x \* **1**

-- This converts the value into the fixed rate currency exchange

**exchange** **::** **Float** **->** **String** **->** **Float**

**exchange** x "USD" **=** x \* **1.2**

**exchange** x "EUR" **=** x \* **1.25**

**exchange** x "GBP" **=** x \* **1**