PLC — CSCI-344
Erlang — Assignment

07/28/2019

1 Introduction

For this assignment you will be required to implement a small program in Erlang as outlined in class.

You will be implementing a very simple banking simulator. Your goal is to have a bank process service client processes.

Your program will start a "bank" process,that will handle the processing of bank transactions sent to it by various "clients", or client processes.

2 The Tasks

You must create a file called `prog4.erl`. All code must be in this file. It must run on the CS machines without error. To run on the CS machines:

- execute `erl` on the command line.
- in the interpreter execute `c(prog4).`
- in the interpreter execute `prog4:start().`

Your file must have the proper module declaration and an export statement to give access to only your `start()` method.

2.1 The Bank Process

The bank process will respond to requests of the client processes.

Details of the bank process:

- Will start with a random account balance between 2000 and 3000.
- Will start a random number of customer processes between 2 and 10.
- Will then sit in receive mode until it gets a message from a client.
- When it receives a message:
  - {`<client_id>, <number>`}: It process the number. Negative removes the amount from the account balance as long as there is enough to cover it. Positive adds the amount to the bank balance. The `client_id` is its process id. It then returns the amount requested, the account balance, and yes or no depending on if the transaction occurred in the form {`<amount>, <balance>, <yes/no>`}.
  - {`<client_id>, balance`}: It will return to the client the current balance.
  - `goodbye`: This will signify that a client is done. Once all clients are done the bank process will terminate. It does not get a `client_id` because it does not care what client terminated.
- It will return to receive mode after processing each message unless all clients have terminated.
- Once all clients have terminated, the bank will display the current balance and terminate.

2.2 The Client Processes

The client processes will be created by the bank process. It will create a random number of them.

Details of a client process:

- The client process will start with a list of random numbers; negative and positive. The values of the numbers can be between -100 and 100. The length of the list will be between 10 and 20.
- It will repeatedly send the next number in the list to the bank process in the form `{<client_id>, <number>}`. Then it will wait for the bank process to respond. It will print the response.
- It will then delay for a random number of milliseconds, between 500-1500, between messages.
- Every 5 message it will request the balance in the form `{their_id, balance}` and print the response.
- Once it has sent all of it's number it will send a final message of `goodbye` and terminate.

There is no specified output for this program, but you should make it meaningful for users. Users should understand what is going on.

Examples of meaningful output:

- "<pid> requested the balance from the bank"
- "<pid> received the balance of <balance>from the bank after a <failed/successful> transaction request of <amount>."

Basically the user should be able to track what is going on. The program should not run silently.

You must have a process called `start()` to start the simulation. This process will handle creating the bank and client processes.This will be the entry point used to test your program

3 Submission

Zip the source file for your program into a zip file called `pa4.zip` and submit to the proper myCourses' Assignment box. A penalty will be enforced if submission/naming rules are not followed.

Late Submissions will not be accepted!

Questions will not be answered within 24 hours of the due date.

4 Grading

This phase will be graded as follows:

- (40%) Proper bank process functionality

- (40%) Proper client process functionality
- (20%) Proper `start()` functionality that starts and executes the simulation as requested.

The following penalties will be enforced above and beyond any points lost above:

- A grade of 0 will be given if it does not run without errors; this includes infinite loops. If a section of code is causing errors comment it out. This will allow me to still see it and potentially award partial credit.
- A grade of 0 will be given if it does not run on the CS machines.
- -10% if the file is not named as requested.
- -10% if the zip file is not named as requested.
- -20% if the submission is not zipped.