a) Coloured Erlang file

```
-module(helloworld).
-export([start/0,account/1,bank/0,clerk/0]).
%% -- BASIC PROCESSING -------------------
n2s(N) -> lists:flatten( %% int2string io_lib:format("~p", [N])). %% HACK!
random(N) -> random:uniform(N) div 10.
%% -- ACTORS -----------------------------
account(Balance) ->
	receive
		{deposit,Amount} ->
			account(Balance+Amount) ;
		{printbalance} ->
			io:fwrite(n2s(Balance) ++ "\n")
	end.
bank() ->
	receive
		{transfer,Amount,From,To} ->
			From ! {deposit,-Amount},
			To ! {deposit,+Amount},
			bank()
	end.
ntransfers(0,_,_,_) -> true;
ntransfers(N,Bank,From,To) ->
	R = random(100),
	Bank ! {transfer,R,From,To},
	ntransfers(N-1,Bank,From,To).
clerk() ->
	receive
		{start,Bank,From,To} ->
			random:seed(now()),
			ntransfers(100,Bank,From,To),
			clerk()
	end.
```

```
start() ->
        A1 = spawn(helloworld,account,[0]),
        A2 = spawn(helloworld,account,[0]),
        B1 = spawn(helloworld,bank,[]),
        B2 = spawn(helloworld,bank,[]),
        C1 = spawn(helloworld,clerk,[]),
        C2 = spawn(helloworld,clerk,[]),
        C1 ! {start,B1,A1,A2},
        C2 ! {start,B2,A2,A1},
        timer:sleep(1000),
        A1 ! {printbalance},
        A2 ! {printbalance}.
```

b) Implement ABC.java

See src ABC.java

c) Answer question

This introduces a race condition in the system, since the Balance returned to the banker could be changed between his inspection and setting the new balance. This is be course we are introducing multiple messages, which tries to perform an atomic operation.

If the engineer insisted on this design, the Account would have to be locked until the call returned with the new balance.