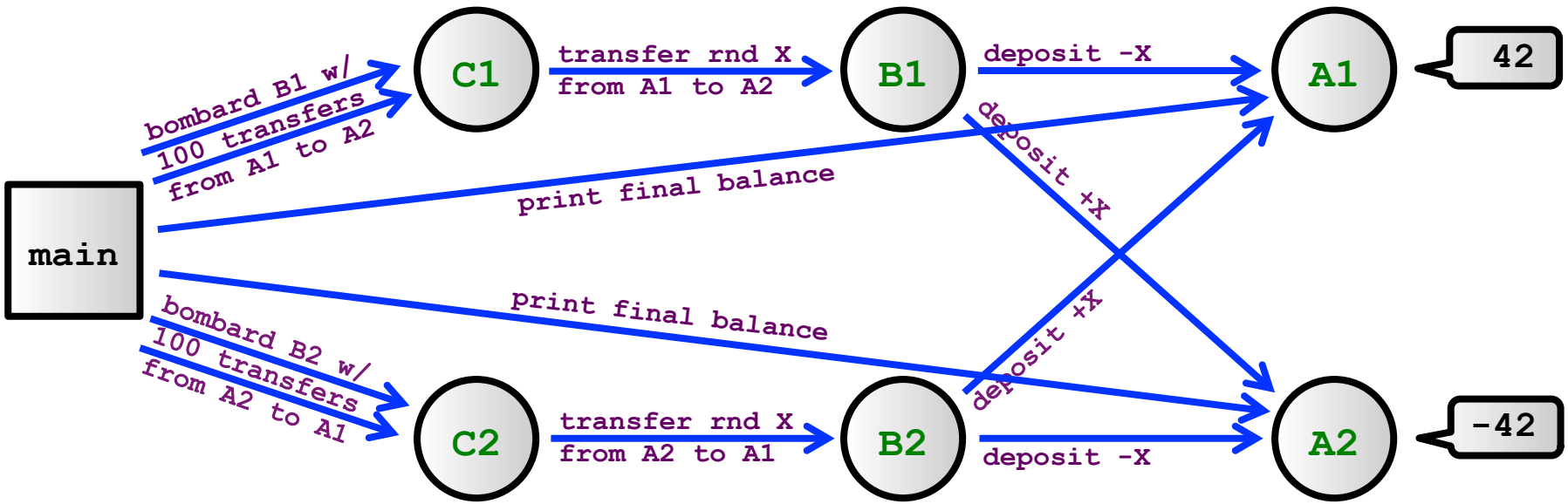
5) ABC (Clerk / Bank / Account)



5) ABC.erl

```
-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}.
```

5) ABC.java (Skeleton)

```
import java.util.Random;   import java.io.*;   import akka.actor.*;

// -- MESSAGES ------------------------------------------------
class StartTransferMessage implements Serializable { /* TODO */ }
class TransferMessage implements Serializable { /* TODO */ }
class DepositMessage implements Serializable { /* TODO */ }
class PrintBalanceMessage implements Serializable { /* TODO */ }

// -- ACTORS --------------------------------------------------
class AccountActor extends UntypedActor { /* TODO */ }
class BankActor extends UntypedActor { /* TODO */ }
class ClerkActor extends UntypedActor { /* TODO */ }

// -- MAIN ----------------------------------------------------
public class ABC { // Demo showing how things work:
    public static void main(String[] args) {
        final ActorSystem system = ActorSystem.create("ABCSystem");

        /* TODO (CREATE ACTORS AND SEND START MESSAGES) */

        try {
            System.out.println("Press return to inspect...");
            System.in.read();

            /* TODO (INSPECT FINAL BALANCES) */

            System.out.println("Press return to terminate...");
            System.in.read();
        } catch(IOException e) {
            e.printStackTrace();
        } finally {
            system.shutdown();
        }
    }
}
```

```
*** OUTPUT ***
Press return to inspect...
Press return to terminate...
Balance = 42
Balance = -42
```

MANDATORY HAND-IN!

a) Color ABC.erl
(according to color convention):
send, receive, msgs
actors, spawn, rest.
(try 2 B as consistent as possible)

b) Implement ABC.java
(as close to **ABC.erl** as possible)

c) Answer question:
What happens if we replace {deposit, ±Amount} w/ the msgs?: