Exercise 12

Exercise A

See implementation below.

```
// javac -cp scala.jar:akka-actor.jar ABC.java
// java -cp scala.jar:akka-actor.jar:akka-config.jar:. ABC

import java.util.Random;
import java.io.*;
import akka.actor.*;
import java.util.concurrent.ThreadLocalRandom;

// -- MESSAGES ------------------------------------------------
class StartTransferMessage implements Serializable {
    private static final long serialVersionUID =
ThreadLocalRandom.current().nextLong();
    public final ActorRef bank;
    public final ActorRef from;
    public final ActorRef to;

    public StartTransferMessage(ActorRef bank, ActorRef from, ActorRef to)
{
        this.bank = bank;
        this.from = from;
        this.to = to;
    }
}

class TransferMessage implements Serializable {
    private static final long serialVersionUID =
ThreadLocalRandom.current().nextLong();
    public int amount;
    public ActorRef from;
    public ActorRef to;

    public TransferMessage(int amount, ActorRef from, ActorRef to) {
        this.amount = amount;
        this.from = from;
        this.to = to;
    }
}

class DepositMessage implements Serializable {
    private static final long serialVersionUID =
ThreadLocalRandom.current().nextLong();
    public int amount = 0;

    public DepositMessage(int amount) {
```

```
        this.amount = amount;
    }
}

class PrintBalanceMessage implements Serializable {
    private static final long serialVersionUID =
ThreadLocalRandom.current().nextLong();
}

// -- ACTORS ------------------------------------------------
class AccountActor extends UntypedActor {
    private int balance = 0;

    public void onReceive(Object o) throws Exception {
        if (o instanceof DepositMessage) {
            this.balance += ((DepositMessage) o).amount;
        } else if (o instanceof PrintBalanceMessage) {
            System.out.println(String.format("Balance = %s",
this.balance));
        }
    }
}

class BankActor extends UntypedActor {
    public void onReceive(Object o) throws Exception {
        if (o instanceof TransferMessage) {
            var cast = (TransferMessage) o;
            var subtract = new DepositMessage(-cast.amount);
            var add = new DepositMessage(cast.amount);
            cast.from.tell(subtract, ActorRef.noSender());
            cast.to.tell(add, ActorRef.noSender());
        }
    }
}

class ClerkActor extends UntypedActor {
    public void onReceive(Object o) throws Exception {
        if (o instanceof StartTransferMessage) {
            for (var i = 0; i < 100; i++) {
                var cast = (StartTransferMessage) o;
                var amount =
ThreadLocalRandom.current().nextInt(Integer.MAX_VALUE);
                cast.bank.tell(new TransferMessage(amount, cast.from,
cast.to), ActorRef.noSender());
            }
        }
    }
}

// -- MAIN --------------------------------------------------
public class ABC {
    public static void main(String[] args) {
        final ActorSystem system = ActorSystem.create("ABCSystem");
```

```
        final ActorRef a1 =
system.actorOf(Props.create(AccountActor.class), "A1");
        final ActorRef a2 =
system.actorOf(Props.create(AccountActor.class), "A2");

        final ActorRef b1 = system.actorOf(Props.create(BankActor.class),
"B1");
        final ActorRef b2 = system.actorOf(Props.create(BankActor.class),
"B2");

        final ActorRef c1 = system.actorOf(Props.create(ClerkActor.class),
"C1");
        final ActorRef c2 = system.actorOf(Props.create(ClerkActor.class),
"C2");

        c1.tell(new StartTransferMessage(b1, a1, a2),
ActorRef.noSender());
        c2.tell(new StartTransferMessage(b2, a2, a1),
ActorRef.noSender());

        try {
            Thread.sleep(1000);
            System.out.println("Press return to inspect...");
            System.in.read();

            var print = new PrintBalanceMessage();
            a1.tell(print, ActorRef.noSender());
            a2.tell(print, ActorRef.noSender());

            Thread.sleep(10);
            System.out.println("Press return to terminate...");
            System.in.read();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            system.shutdown();
        }
    }
}
```

Exercise B

If we inspect, get a balance and sets a new balance based on the received balance, then we get race conditions e.g. lost updates since the balance on B can change while we are calculating the balance to set.