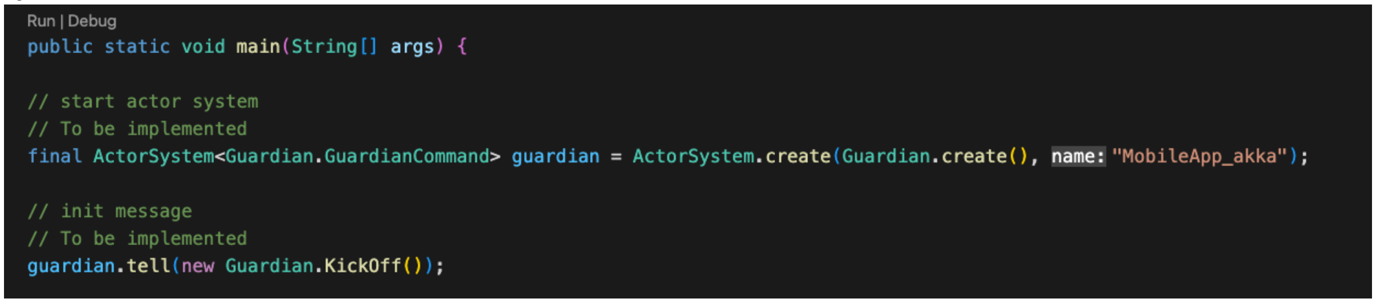
Exercise 11

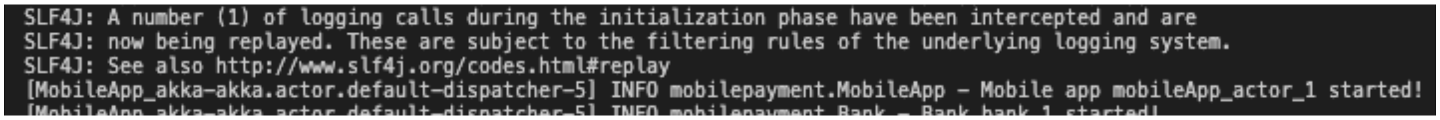
1. In my implementation, the guardian has 6 state, they're 2 mobile apps, 2 banks and 2 accounts.

   The message it must handle is KickOff.

   To start off the actor system, I create a new Guardian and send KickOff message to it to start the whole system.

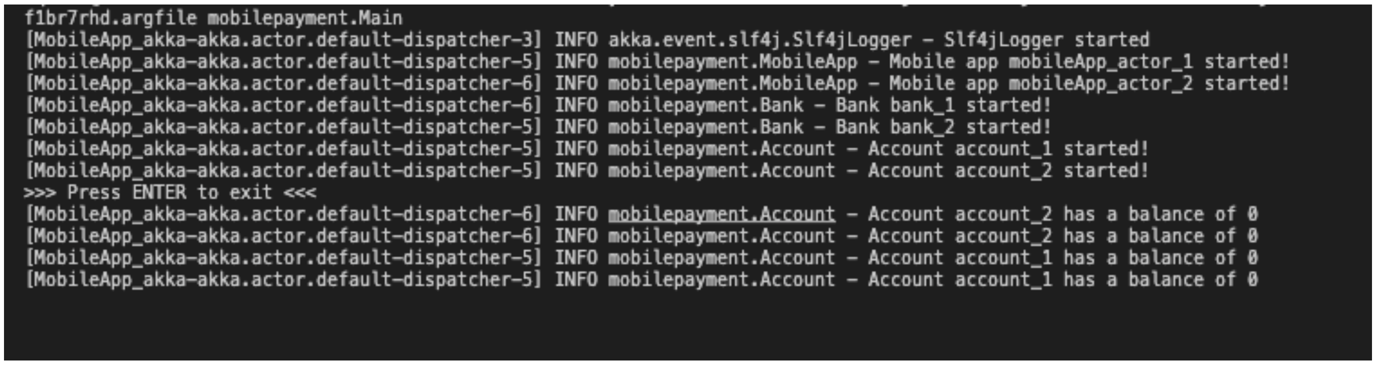


```
Run | Debug
public static void main(String[] args) {

// start actor system
// To be implemented
final ActorSystem<Guardian.GuardianCommand> guardian = ActorSystem.create(Guardian.create(), name: "MobileApp_akka");

// init message
// To be implemented
guardian.tell(new Guardian.KickOff());
```



```
SLF4J: A number (1) of logging calls during the initialization phase have been intercepted and are
SLF4J: now being replayed. These are subject to the filtering rules of the underlying logging system.
SLF4J: See also http://www.slf4j.org/codes.html#replay
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.MobileApp - Mobile app mobileApp_actor_1 started!
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.Bank - Bank bank_1 started!
```

2. For the Account actor, each Account has a state called 'balance', which is the balance of this Account, it is initialized to 0 for each account. The message it must handle is *Deposite* message send from Bank actor.
3. Bank actor doesn't maintain state in our implementation. The message it must handle with is *Transaction*, it will send 2 *Deposite* message to sender account and the receiver.
4. Mobile App also doesn't hold any state. The message it must handle with is *PaymentReq* which has 3 parameters: bank, Account_from and Account_to. It will make a random payment from Account_from to Account_to via the given bank.
5. finished.
6. finished. The mobileApp will generate 100 random amount, and then send 'Transaction' message to banks.
7. The balance printed is the one before all payments are made. I guess it is because this info printed before the message sent, and the code will print the most stable balance of each Account.



```
f1br7rhd.argfile mobilepayment.Main
[MobileApp_akka-akka.actor.default-dispatcher-3] INFO akka.event.slf4j.Slf4jLogger - Slf4jLogger started
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.MobileApp - Mobile app mobileApp_actor_1 started!
[MobileApp_akka-akka.actor.default-dispatcher-6] INFO mobilepayment.MobileApp - Mobile app mobileApp_actor_2 started!
[MobileApp_akka-akka.actor.default-dispatcher-6] INFO mobilepayment.Bank - Bank bank_1 started!
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.Bank - Bank bank_2 started!
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.Account - Account account_1 started!
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.Account - Account account_2 started!
>>> Press ENTER to exit <<<
[MobileApp_akka-akka.actor.default-dispatcher-6] INFO mobilepayment.Account - Account account_2 has a balance of 0
[MobileApp_akka-akka.actor.default-dispatcher-6] INFO mobilepayment.Account - Account account_2 has a balance of 0
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.Account - Account account_1 has a balance of 0
[MobileApp_akka-akka.actor.default-dispatcher-5] INFO mobilepayment.Account - Account account_1 has a balance of 0
```

8. I think there won't be a race condition in this case. Since each Thread(Actor) communicate with each other by sending message, and even if they send at the same time, the receive time counld be different, all of these messages will be stored in the receiver Actor's mailbox by some order. The order of these message may not be the same as they were sent, but there won't be a race condition.