# 18.3.2.1 Money mule example

## Description

As mentioned, the scenario for the `Accounts` being subject for observation, and for `Customer` and `alarm` to be observers is that the bank on regular basis go through all its account to find if there has been any issue with the transactions on accounts. This may be that the transactions form a suspicious pattern, e.g. because the account has been hacked.

As a simple example we consider the case of money mules. A money mule is someone who transfers (typically illegally acquired) money on behalf of someone else. Money mules can move money in various ways, including through bank accounts.

Some money mules know they are supporting criminal enterprises; others are unaware that they are helping criminals profit.

Money mules often receive a commission for their service, or they might provide assistance because they believe they have a trusting or romantic relationship with the individual who is asking for help.

Acting as a money mule is illegal, therefore the customer is also included as an observer.

For the purpose of simplicity we assume that money muling takes place in one day, i.e. there is a deposit of a certain amount and a withdraw of the same amount on the same day. This activity may not be money muling, but it is still worth being reported as an issue. In real life banking there has to be a pattern of this pair of `deposit`/`withdraw`.

`Account` will for this purpose have the method `checkTransactions` that goes through all transactions on a given day and execute `asSubject.notifyObservers` for each occurrence of a `deposit` and `withdraw` of the same `amount`:

```
class Account(owner: ref Customer):
   -"-
   checkTransactions:
      transactionsOfToday.scan
         if (current.what = "deposit") :then
            amount: var Real
            amount := current.whichAmount
            scanTail
               if (current.what = "withdraw") :then
                  if (current.whichAmount = amount) :then
                     asSubject.notifyObservers
```

The method `scanTail` is defined in `scan` and iterates over the rest of the elements in the `Set`.

The `Bank` object (or some object within the `Bank`) calls this method every day for each `Account` object in the list of accounts:

```
...
theAccountsFile: obj OrderedList(#Account)
...
theAccountsFile.scan
   current.checkTransactions
```

The following Sequence Diagram shows the sequence of method calls for a partial run.