```
sig Bank {
 number: one Int,
 name: one String,
 branch: set Branch,
 transactions: set Transaction
}

sig Branch {
 name: one String,
 number: one Int,
 accounts: set Account
}

abstract sig Account {
 acNumber: one Int,
 balance: one Int,
 owner: one AccountHolder
}

sig IndividualAccount extends Account { }
sig BusinessAccount extends Account { }

sig AccountHolder {
 id: one Int,
 name: one String
}

sig Transaction {
 id: one Int,
 status: one String,
 from: one Account,
 to: one Account,
 amount: one Int,
 date: one Date
}
sig Date {}


fact {
 no disj b1, b2: Bank |  b1.number = b2.number
 no disj b1, b2: Branch | b1.name = b2.name
 // No two account numbers and the transaction numbers are the same
 all disj a1, a2: Account | a1.acNumber != a2.acNumber
 all disj t1, t2: Transaction | t1.id != t2.id

 // An account number can not be duplicated on multiple branches
 all disj b1, b2: Branch | no (b1.accounts.acNumber & b2.accounts.acNumber)

 // For a single successful transaction of an individual and business account, the account holder transfer maximam amount of 50,000 and 150,000 respectively.
 all t: Transaction | t.status ="passed" => (t.from in IndividualAccount => t.amount < 50000) and (t.from in BusinessAccount => t.amount < 1500000)

 // An account holder has to transfer a minimum amount of 5000 from individual account to business account and 3000 for vice versa.
 all t: Transaction | t.status = "passed" &&
   (t.from + t.to not in IndividualAccount or t.from + t.to not in BusinessAccount)
       => (t.from in IndividualAccount => t.amount >=5000) and
        (t.from in BusinessAccount => t.amount >=3000)

 // After a successful transaction, the  amount must be reflected to the respective accounts.
 all t: Transaction | t.status ="passed" => (t.from.balance = t.from.balance - t.amount) and (t.to.balance = t.to.balance + t.amount)
}

assert a1{
 //The amount of a transaction must be positive
    // Not Found
 all t: Transaction | t.amount >0
}
check a1 for 5

assert a2{
 // No two account holders have the same account number
     // Found
 no disj u1, u2: AccountHolder | no( owner.u1 & owner.u2)
}
check a2 for 5

assert a3{
 // An account holder can not have more than 2 accounts of any particular bank
     // Found
 all b: Bank | all disj a1, a2: b.branch.accounts| a1.owner != a2.owner
}
check a3 for 5

assert a4{
 // All failed transactions transfer an amount which is greater than 5000 from individual to business account
     // Not Found
 all t: Transaction | t.status="failed" && t.from=IndividualAccount && t.to=BusinessAccount => t.amount>5000
}
check a4 for 5

assert a5{
 //For every successful transaction, the balance of the sender must be sufficient and there must have a valid sender and receiver account.
     // Not Found
 all t: Transaction | t.status ="passed" => (t.from.balance >= t.amount) and (t.from in Account and t.to in Account)
}
check a5 for 5
```