112   **MATIÈRE** Advanced Database

13) a) I was in charge of creating the user ⑤
interface for the ATM.
I used Swing and Intellij (my IDE) to
perform this task.
I created every window and positionned the
buttons, text areas and table.
I also created the interactions between each
window (i.e: clicking a button opens a new
window, etc).
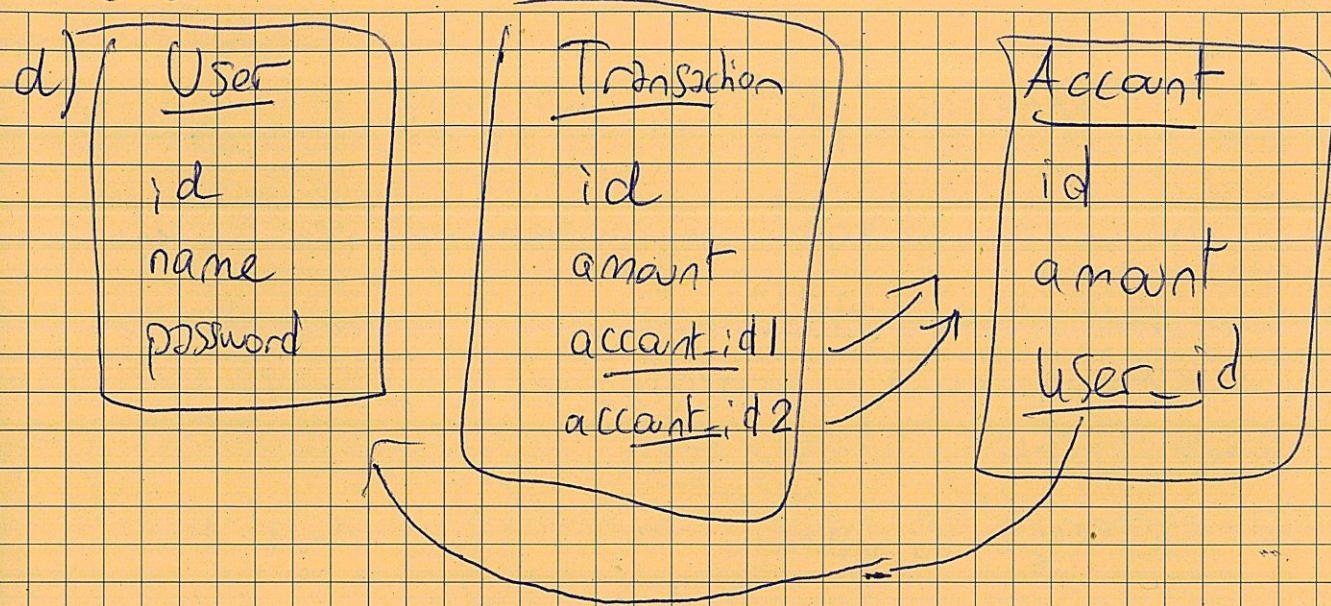
As a group, we created the database schema
and I was part of it.

b) I found it hard to open a window
from a remote class and also to make
sure that closing a window would not
close the others except for the first
one -
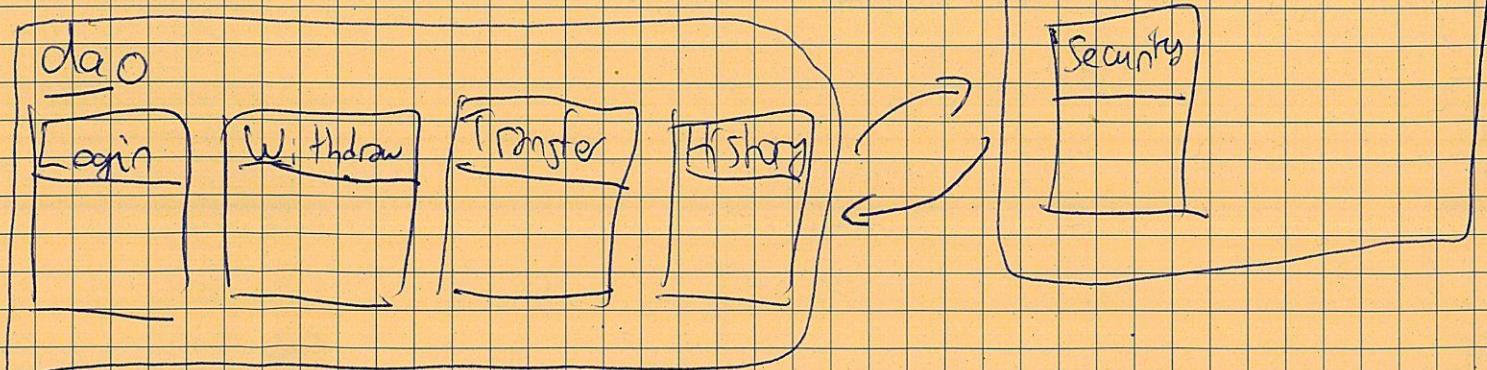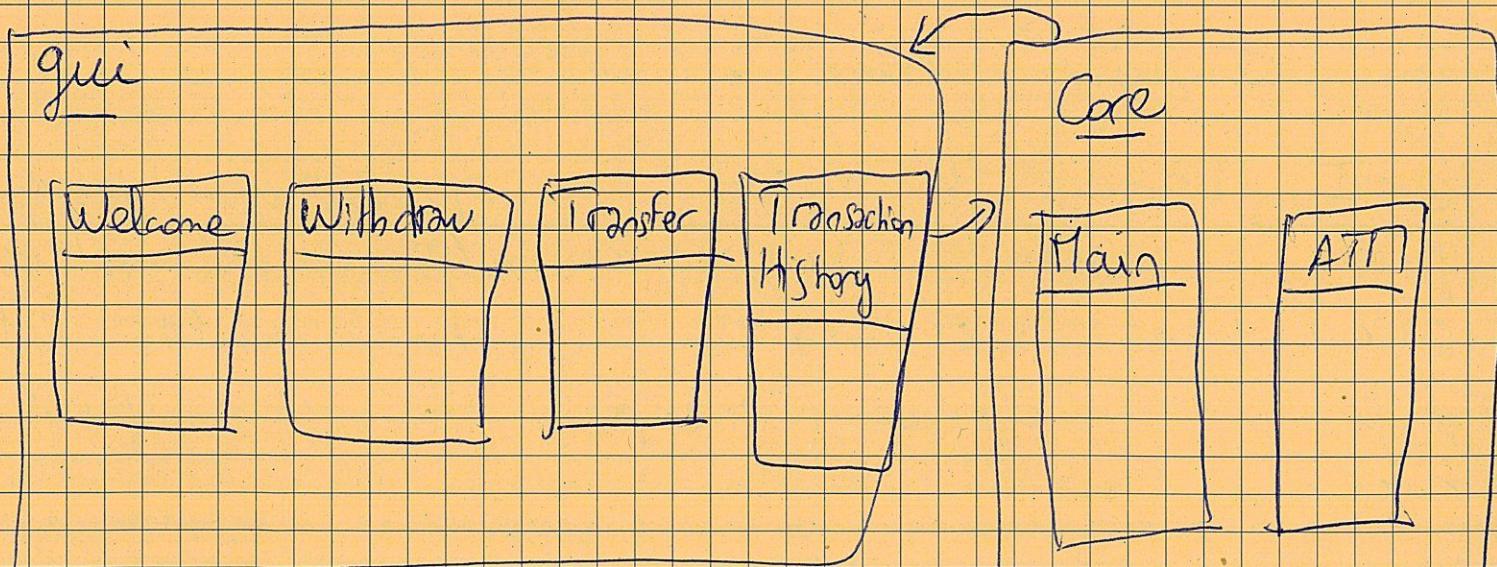
c) Our software can
- log in into an account
- withdraw money
- put/transfer money
- show transaction history

Database:

d)

User
id
name
password

Transaction
id
amount
accont_id1
accont_id2

Accont
id
amount
user_id

Classes: each big square is a package
little square is a class:

gui

| Welcome | Withdraw | Transfer | Transaction History |

Core

| Main | | ATM |

dao

| Login | Withdraw | Transfer | History |

Security

The GUI package is responsible of showing Swing windows, DAO is for Oracle queries and Core calls, the GUI filling it from data from DAO package.

1) This was not done in TP and the slide mention it without explanation. ⓪

2) The 3-tiered architecture consists of one or several clients connecting to an application server, which can be in java, or c++, or a web service! - whatever, which will then call the database itself (which can contain some PL/SQL depending on where the procedural code is).

3) Response time and throughput are important data to consider in order to tune ⓪,⑤ performances.
Response time is the time a query will take to complete before the results are returned.
Throughput is the speed at which results are returned ("débit").

4) Sharding is the action of representing documents
(= SQL rows) as shards. ①

Map-reduce is the action of regrouping all the
data we need (map) that be very diverse
in maps (associative arrays like key = value)
and reducing it in order to return less
results with more order.

5) Select ename, sal, lag(sal), avg(sal) over (order by
sal 3 preceding) from emp; ①

6) Select ename, job, sal Over (partition by deptno
order by sal desc limit 3) from emp; ①.₅

7) select ename, job, sal from emp e1 where
e1. sal > (select avg (sal) from emp e2 where
e1. deptno = e2. deptno); ①

8) ~~select sum(sal), deptno from emp over(sal, deptno)~~

Select sal, deptno from emp group by cube(sal, deptno);
①

9. select sal, deptno, job group by cube(sal, ~~job~~ job)
order by deptno asc;

(1)

10. a) select sal, deptno, job group by rollup(sal,
job) order by deptno asc;

(1)

b) select sal, deptno, job group by cube(deptno, job)
order by ~~job~~ deptno asc;

left ~~outer~~ outer
11) select name, deptno from emp join dept
on ~~dept~~ . deptno = ~~emp~~ . deptno
    emp   dept

(0,5)

12) select name, deptno from emp right outer join
dept on ~~emp~~ . deptno = ~~dept~~ . deptno
         emp   dept

(0,5)

13) is above

14)

15) 2) A multiple sequence alignment is used to compare DNA from different specis while a sequence alignment is used to compute a score to find the best ① alignments sequence

b) We can use Smith & Waterman

16) use est

db. emp. insert ({ empname : " John ", sal = 500,
                           dept : " ohio " })

db. emp. insert ({ empname : " Bob ", job : " Analyst ",
                           dept = 10 })

We do not use joins in Mongo DB to
improe performance

17) db. emp. find ({ " sal ": { $gte : 400 }})