Persistent data

Version 070418

Lessons from today's lecture

- We reap the benefits of earlier good design!
  - The persistency of the data only affects the data process
- We find that modern software has both administrative and more real-time aspects
  - and there is a good reason to keep them separated
- We introduce more third party software
  - and more magic "boiler plate" code
- We perform moderate re-engineering and clean-up
  - due to requirements of the new third party software
  - due to the fact that some data structures are not needed any more

How to make data persistent

- ICU8 has data in transient memory
  - terminating the program also removes the data
- Alternative 1: Sequential file
  - Store when program terminates onto sequential files
  - Read the sequential files when starting the program
- Alternative 2: Data base
  - Every data transaction is made in the data base
  - Oracle is the obvious choice at Ifi
- Alternative 3: Hibernate
  - Hibernate is a powerful, high performance object/relational persistence and query service.
  - Hibernate may use different kinds of data management below
  - http://www.hibernate.org/

Comparing the alternatives

	Sequential file	Database	Hibernate
Pro	•Simple to implement	•SQL and Oracle are well known •Robust	•More abstract interface •Open source
Con	•Not robust for errors during run •Slow for large data	•Depends on database server	•Even more to learn (puh!)

We choose alternative 2 – The Oracle SQL base

- Everybody having taken INF1050 has access to the Oracle base at Ifi
  - If you have not taken this course, contact Drift for password
- There is a simple (but slightly cryptic) interface to the Oracle base to be used from Java
- Strategy:
  - Choose a solution extremely similar to the transient one
    - Two tables: *gsmuser* and *hotspot*
  - Create the tables external to the program
    - through *gqlplus*
  - Populate the hotspot table externally

gqlplus – from Linux

- >gqlplus *brukernavn*@ifiora
- >Enter password: *mypassword*
- SQL>start sql-setup-070410.sql
  - DROP TABLE gsmuser;
  - DROP TABLE hotspot;

  - CREATE TABLE gsmuser
  - (nickname VARCHAR(30) PRIMARY KEY
  - ,staticid VARCHAR(8) UNIQUE
  -);

  - CREATE TABLE hotspot
  - (hotname VARCHAR(30) PRIMARY KEY
  - ,decx FLOAT
  - ,decy FLOAT
  -);

  - INSERT INTO hotspot(hotname,decx,decy) VALUES ('Ifi',10.717409, 59.943460);
  - INSERT INTO hotspot(hotname,decx,decy) VALUES ('Kolbotn', 10.798018, 59.807351);
  - INSERT INTO hotspot(hotname,decx,decy) VALUES ('Oslo-S', 10.762593, 59.908570);

textual file with SQL

killing any old versions of the tables

creating the tables

creating the hotspots

more gqlplus

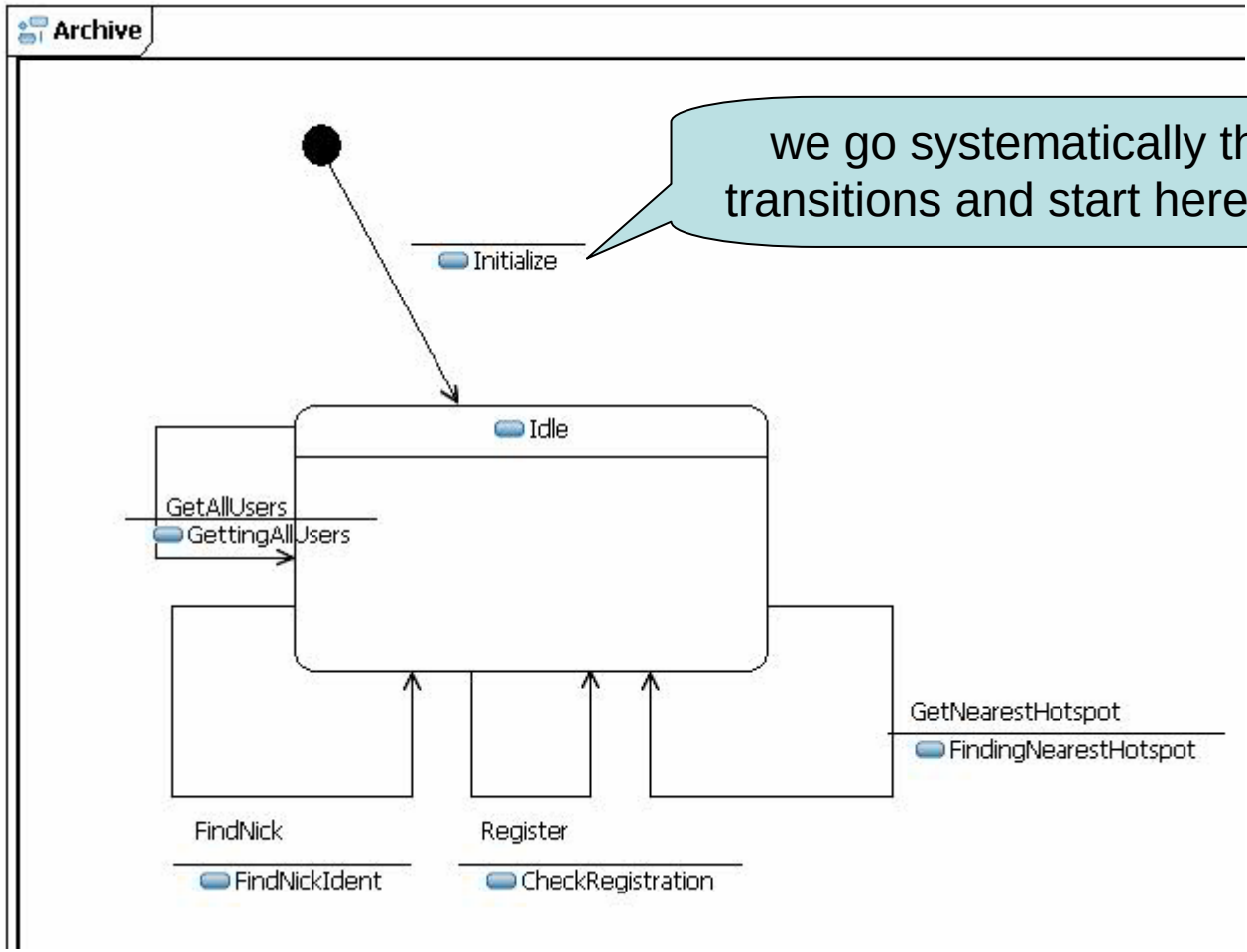
- SQL>SELECT table_name FROM user_tables;
  - gives you the names of all your tables (on your user)
    - TABLE_NAME

```
------------------------------
GSMUSER
HOTSPOT
```

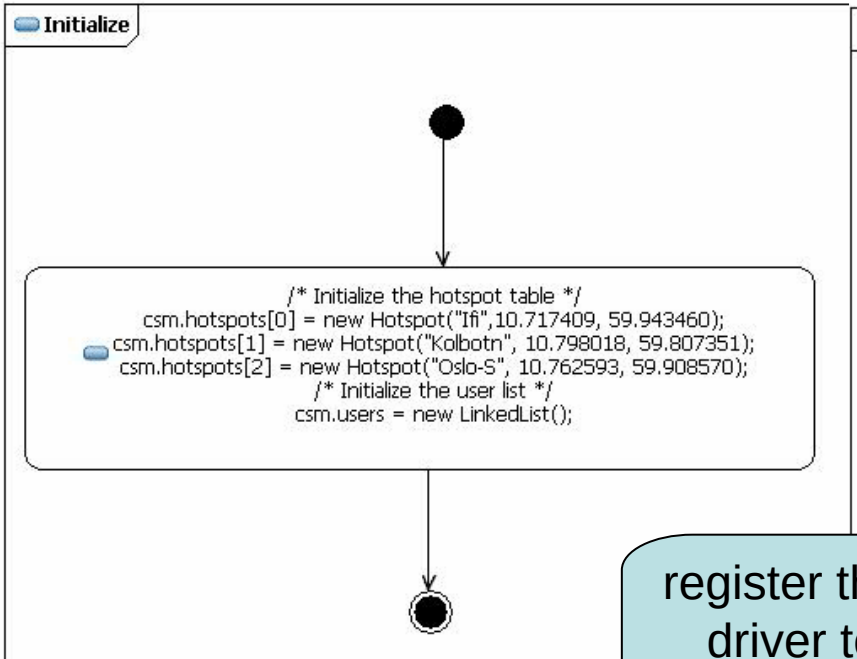
- SQL>DESCRIBE gsmuser
  - this is an SQL*plus command and should *not* end in a semicolon!
  - It gives you all the attributes of the mentioned table
    - Name

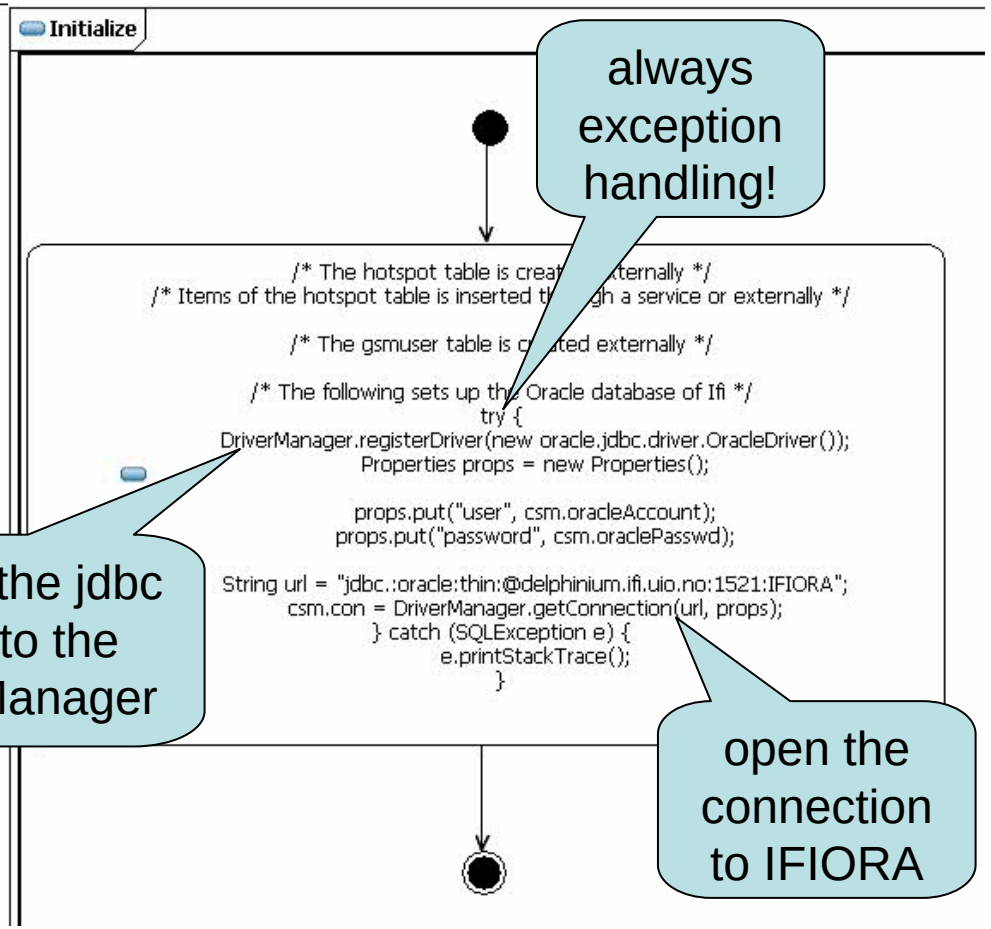
```
Name                                      Null?    Type
----------------------------------------- -------- --------------
NICKNAME                                  NOT NULL VARCHAR2(30)
STATICID                                           VARCHAR2(8)
```

Modifying the Archive process



Initialize





always exception handling!

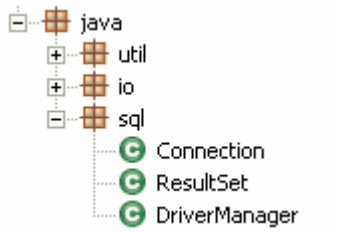
register the jdbc driver to the Driver Manager

open the connection to IFIORA

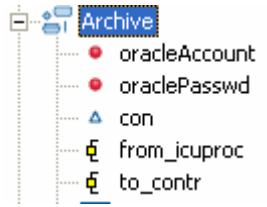
INF 2120

Opening TODOs with the model

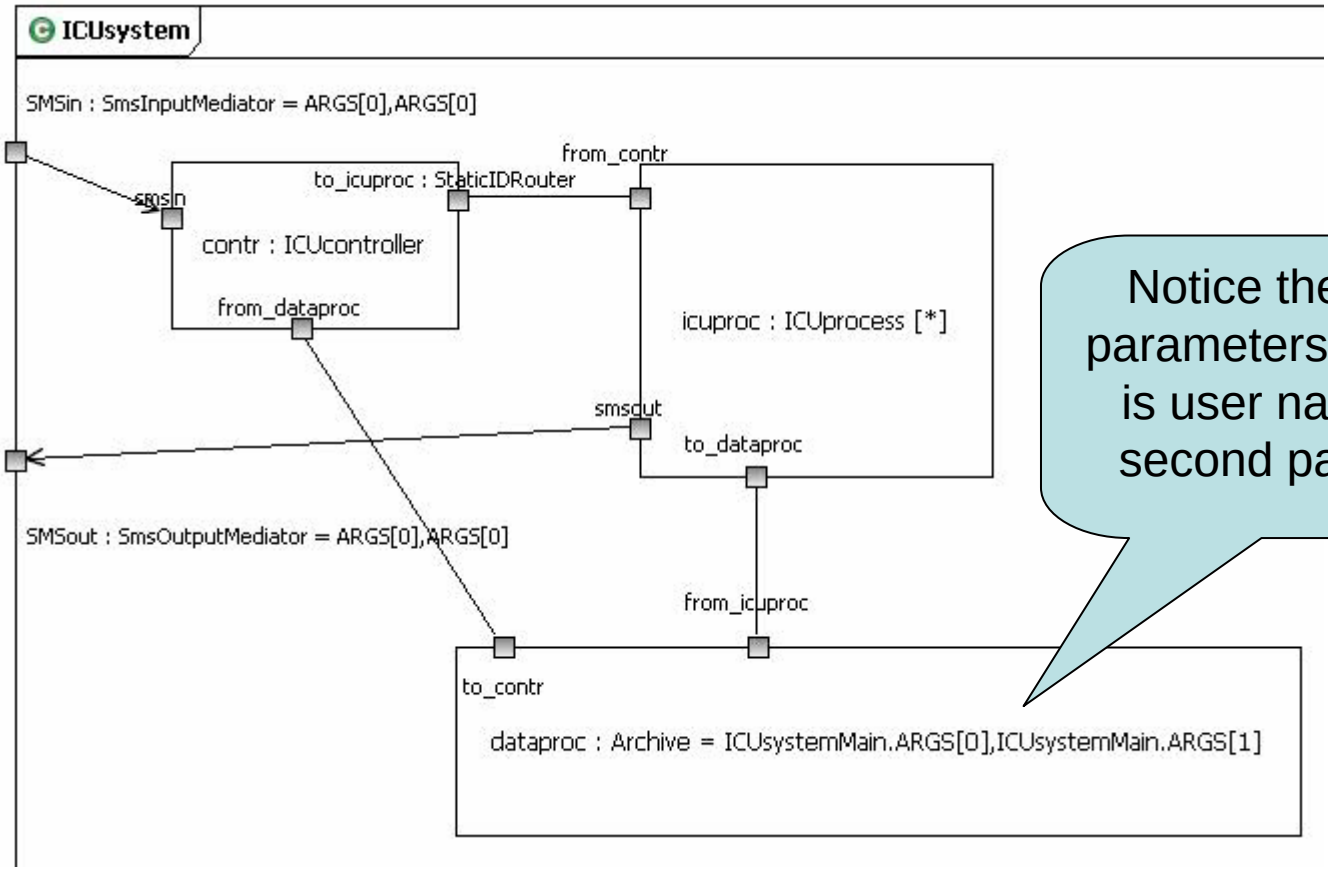
- Add *jdbc.jar* to the java-project of your system
  - this is necessary for the Java execution and compilation
- Add a java.sql package to the model
  - with Classes:
    - Connection
    - ResultSet
    - DriverManager



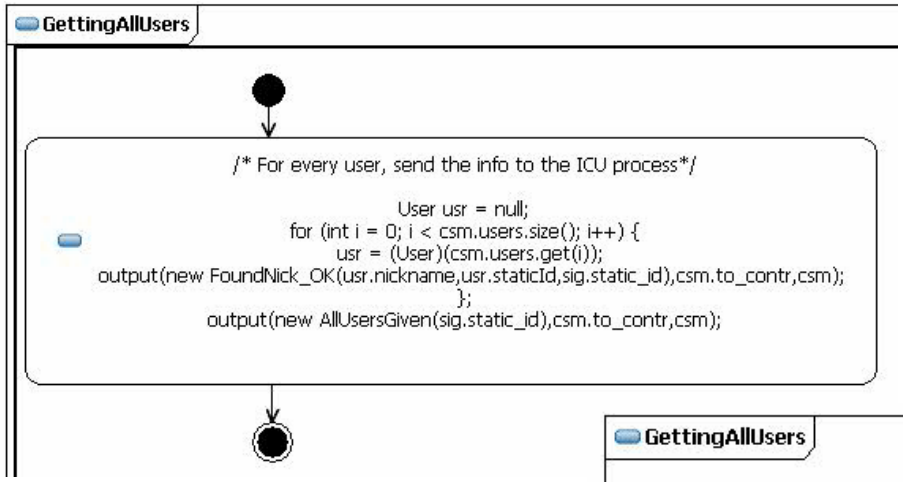
  - You may copy this package from the ICU9 model to your own.
- Add two parameters to the Archive process
  - oracleAccount (a string with your user name)
  - oraclePasswd (a string with corresponding password)
- Add an attribute referring to the Connection

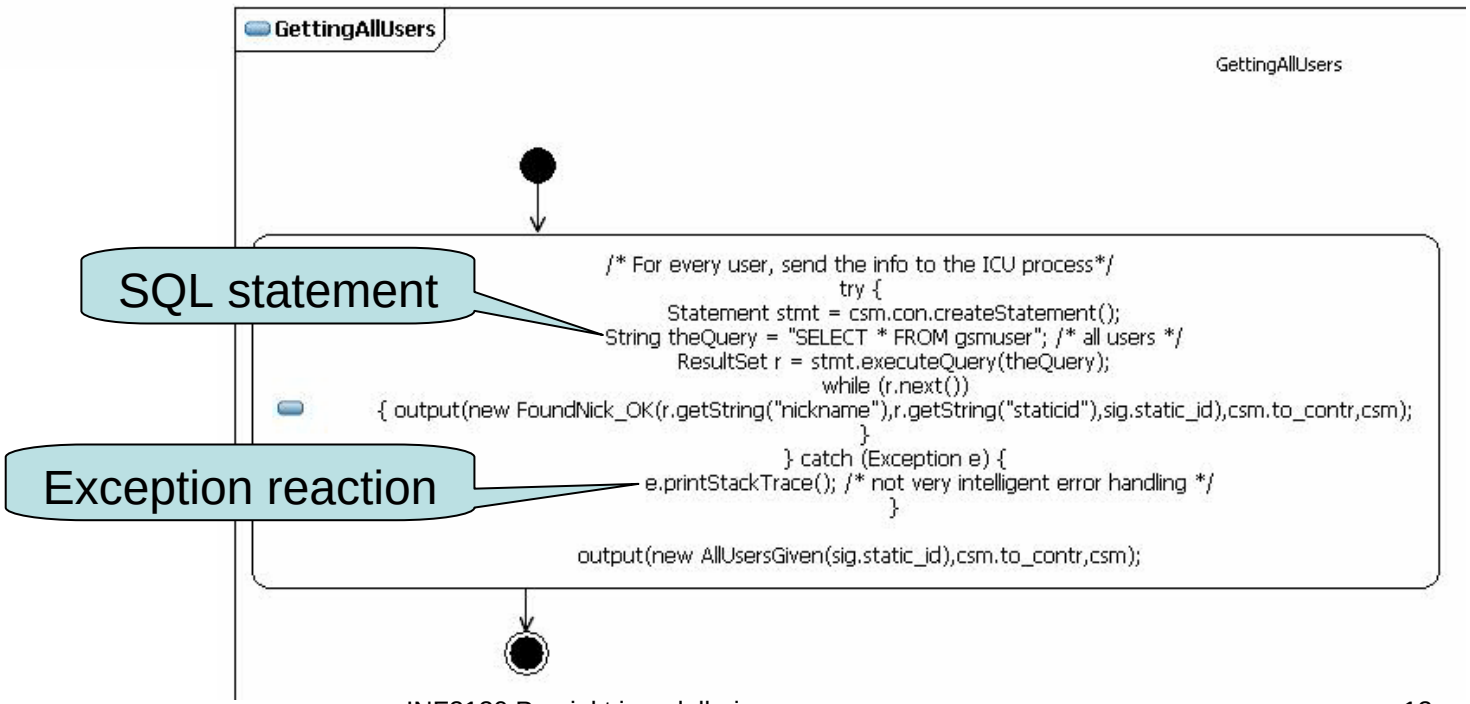


Revised ICUsystem composite structure



GettingAllUsers





Details of the jdbc interface

```
try {
        Statement stmt = csm.con.createStatement();
        String theQuery = "SELECT * FROM gsmuser";
        ResultSet r = stmt.executeQuery(theQuery);
        while (r.next())
        { output(new FoundNick_OK(r.getString("nickname"),
          r.getString("staticid"),sig.static_id),
          csm.to_contr,csm);
        }
} catch (Exception e) {
  e.printStackTrace(); /* not very intelligent error handling */
}

output(new AllUsersGiven(sig.static_id),csm.to_contr,csm);
```

create statement from connection

select all users

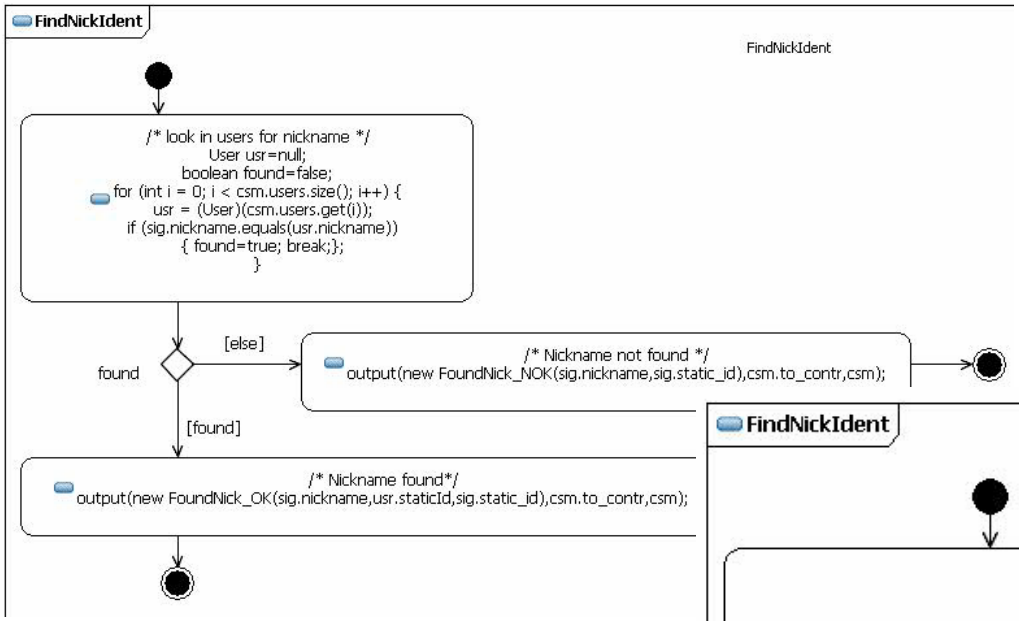
execute the query

iterate through the rows of the result

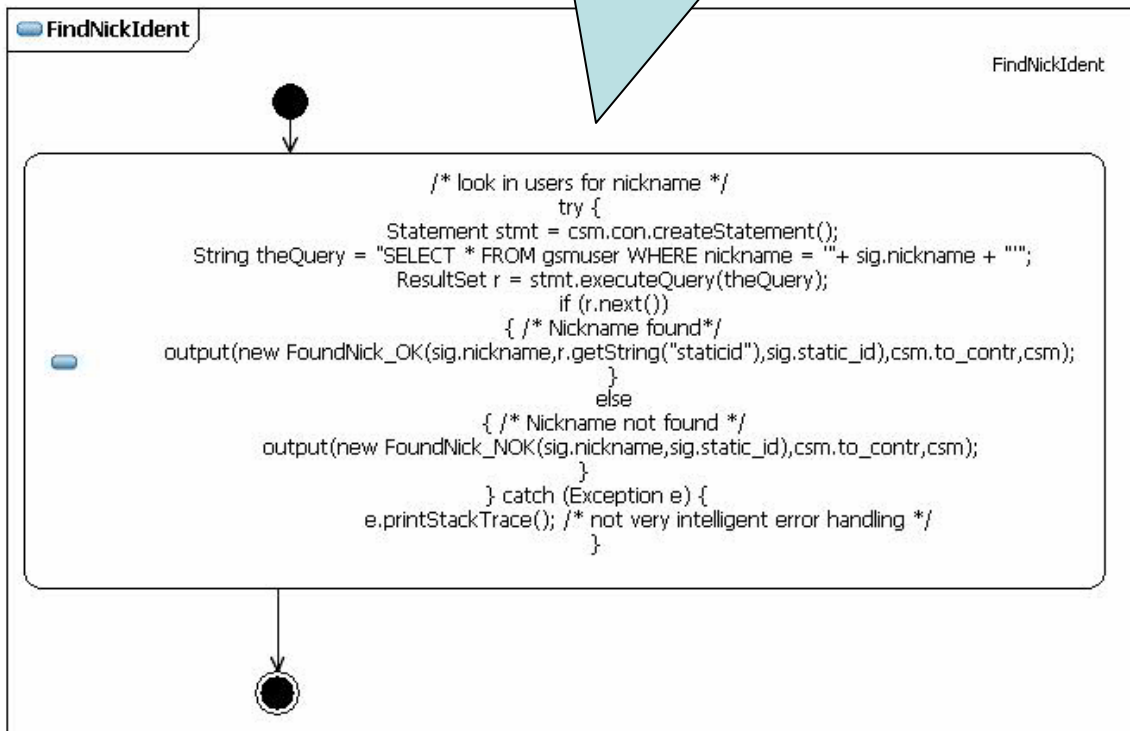
get the ”nickname” value of the current row

very rudimentary exception handling indeed!

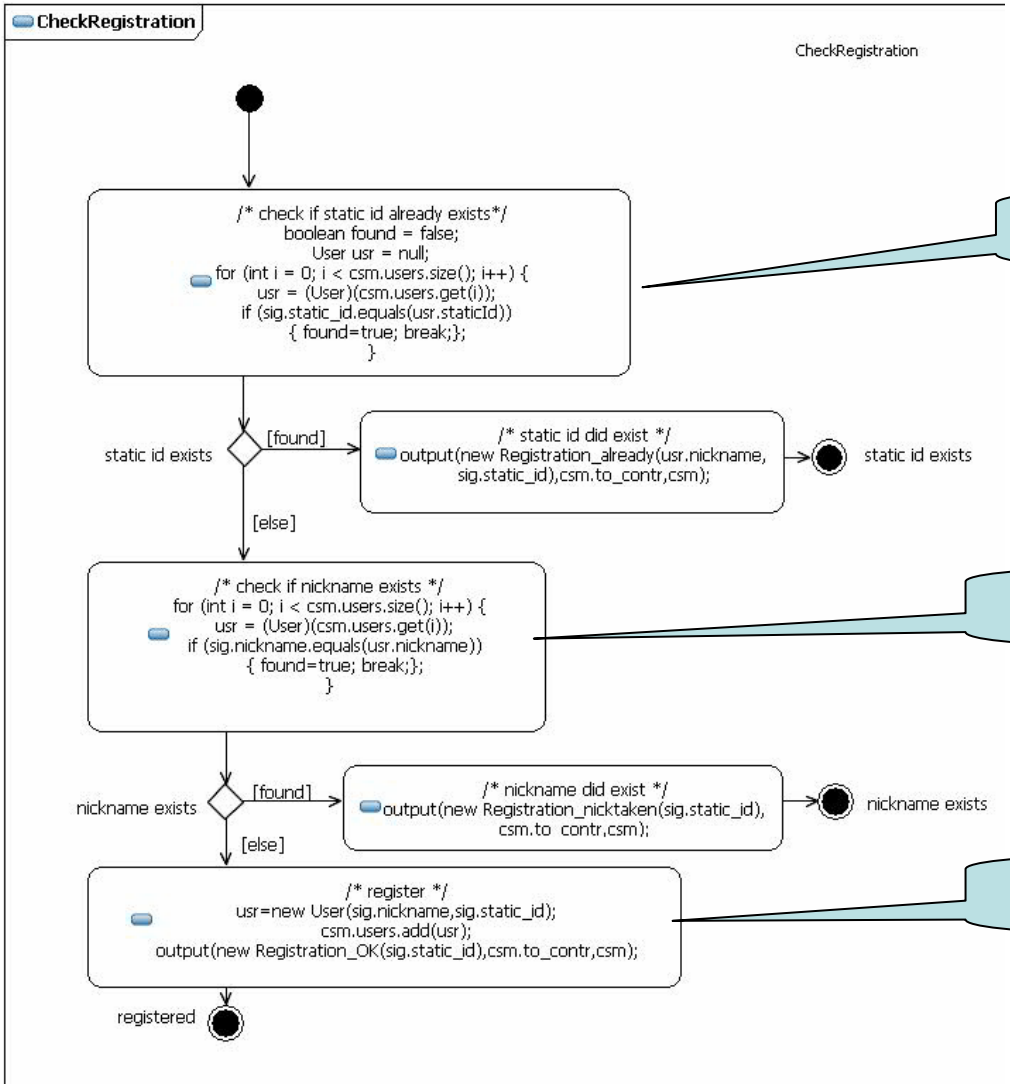
FindNickIdent



The variants are included inside the try-clause



Check Registration (ICU8 – the transient data)



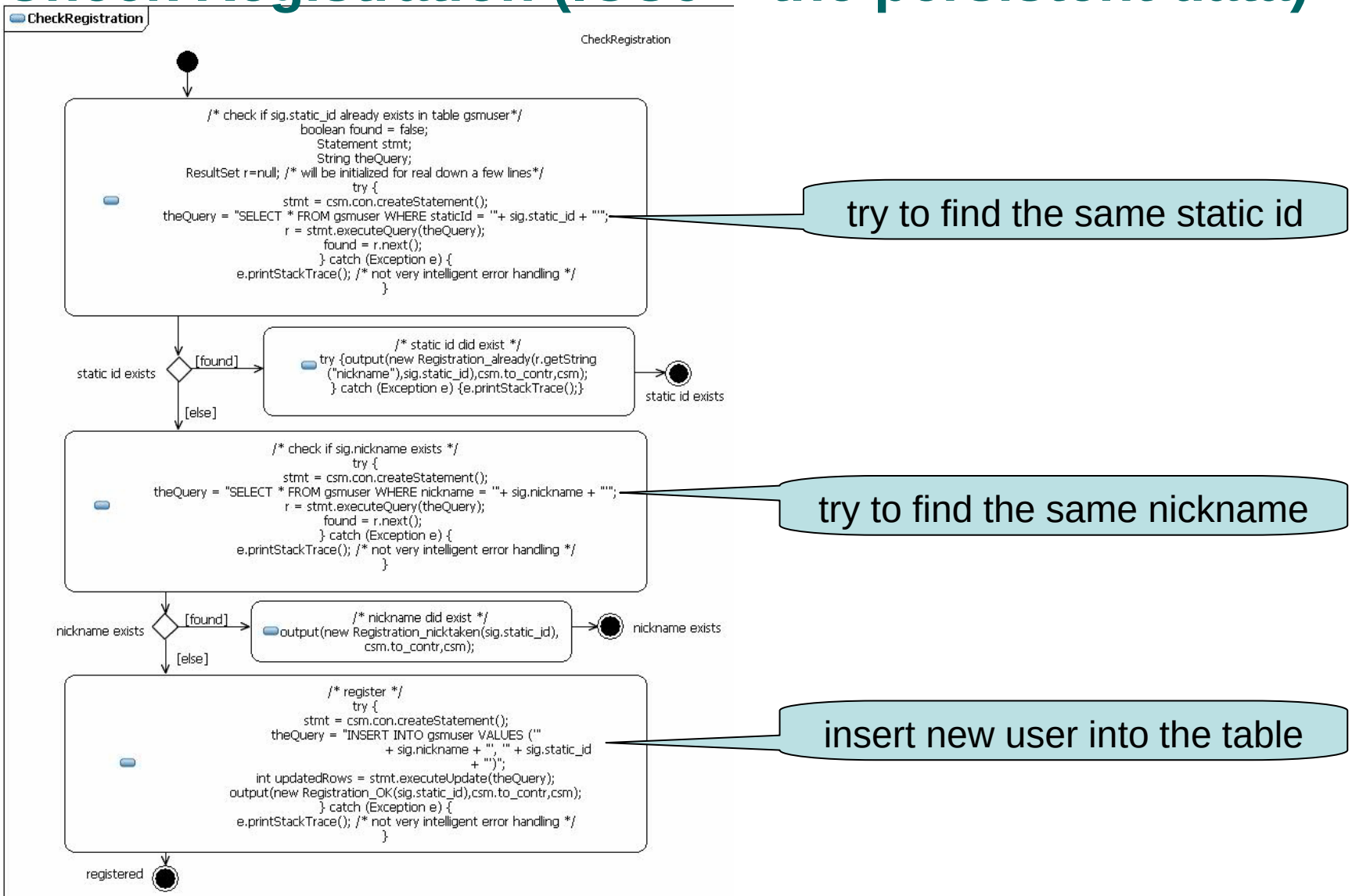
try to find the same static id

try to find the same nickname

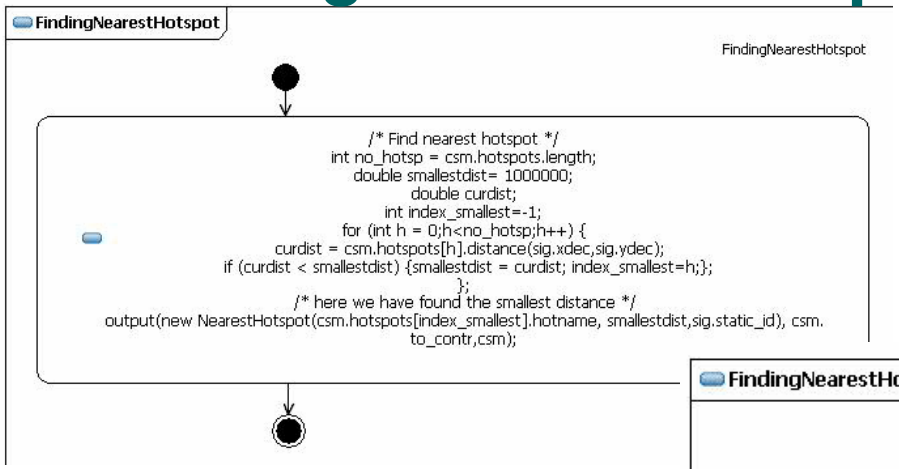
insert new user into the table

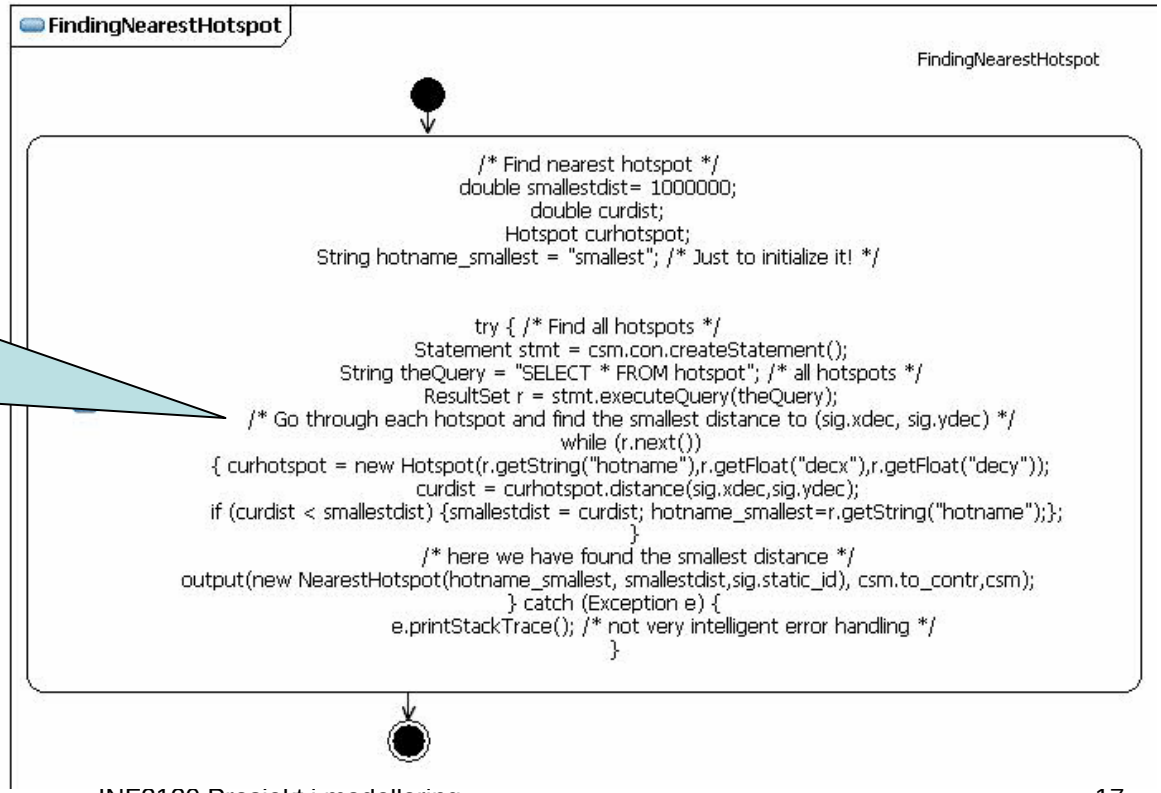
INF 2120

Check Registration (ICU9 – the persistent data)



Finding Nearest Hotspot





Here we have Archive perform the comparison and the calculation of distance for every row

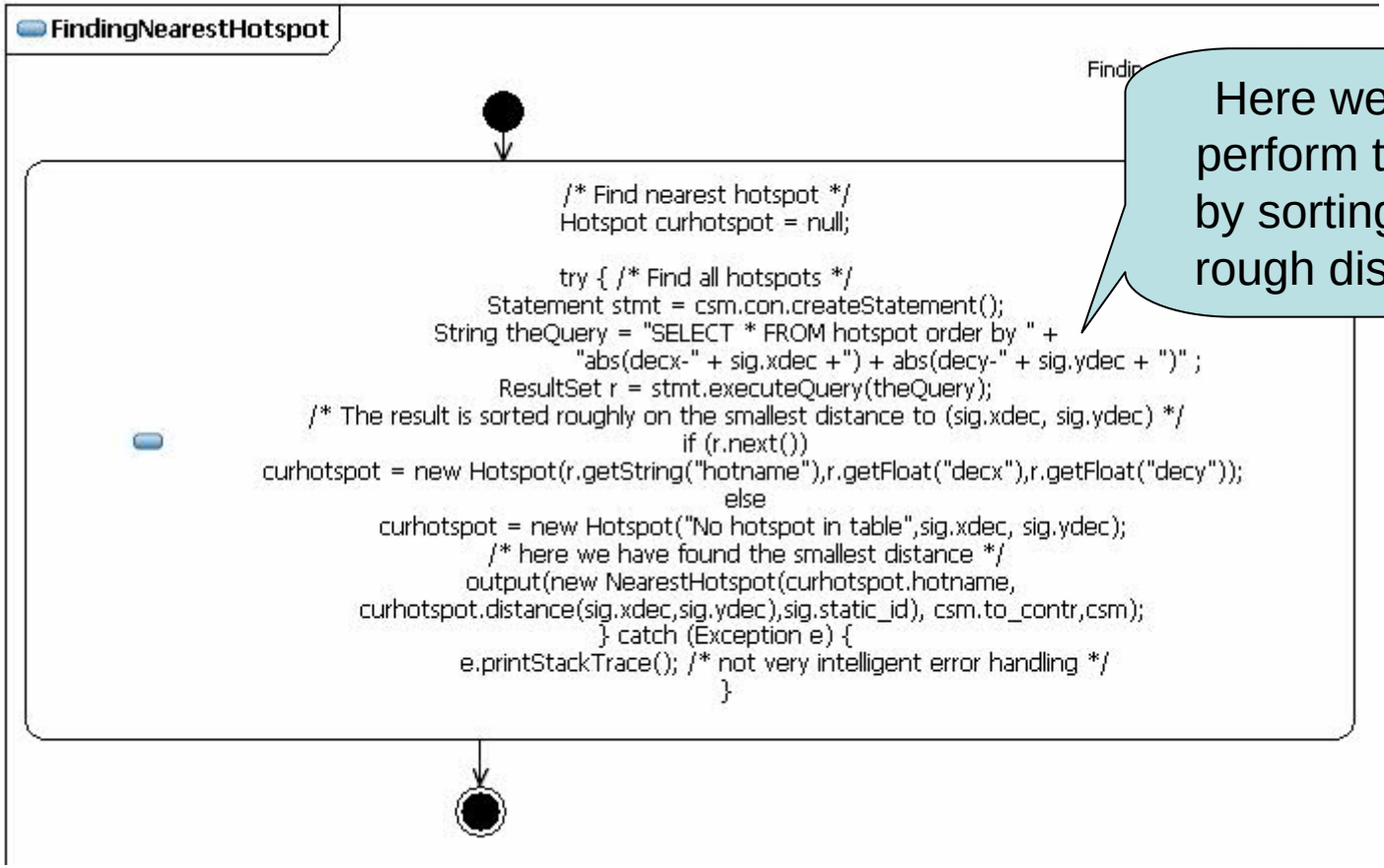
Final cleaning up

- There are some ICU8 concepts that are no more needed
  - transient tables *users* and *hotspots*
  - class User
    - used to define the objects of the *users* table
- class Hotspot is kept
  - since we use its *distance* operation
    - but if we really wanted we could of course move that

Reflections on the Oracle data base approach

- The data have become persistent
  - this has made the data more robust against failures at runtime
- The Archive still defines a critical region
  - since there is still only one data process in our system, only one data transaction will take place at one time
- We have introduced even more distribution
  - behind the scenes and somewhat hidden in the model
  - In principle the data process (Archive) is a gateway to another world
- Could we ”outsource” more work to this ”other world”?
  - In our data base implementation of FindingNearestHotspot, the Archive did all the comparison. Could we outsource that?

Finding Nearest Hotspot by the Oracle machine



Here we have IFIORA perform the comparison by sorting the rows on a rough distance estimate

In more detail ...

```
Hotspot curhotspot = null;
try { /* Find all hotspots */
Statement stmt = csm.con.createStatement();
String theQuery = "SELECT * FROM hotspot order by " +
                  "abs(decx-" + sig.xdec +") + abs(decy-" + sig.ydec + ")" ;
ResultSet r = stmt.executeQuery(theQuery);
/* The result is sorted roughly on the smallest distance to (sig.xdec, sig.ydec) */
if (r.next()) /* here reading the first record only */
 curhotspot = new Hotspot(r.getString("hotname"),r.getFloat("decx"),r.getFloat("decy"));
else
 curhotspot = new Hotspot("No hotspot in table",sig.xdec, sig.ydec);
/* here we have found the smallest distance */
output(new NearestHotspot(curhotspot.hotname,
        curhotspot.distance(sig.xdec,sig.ydec),sig.static_id), csm.to_contr,csm);

} catch (Exception e) {
  e.printStackTrace(); /* not very intelligent error handling */
}
```

Summary of persistence

- We changed only the Archive
  - and its instantiation in the enclosing composite structure
  - This is a proof that our separation of concern worked well
- We discussed the distribution og work between
  - our ICU-program
  - and the Oracle data base server
- The transition structures were quite unchanged
  - but due to java exception handling it was easier to have less UML
- The cleaning up was trivial
- Note: the data base here was very, very simple indeed!
  - the general principles should apply even for more realistic cases!