IMPORTANT NOTICE TO STUDENTS

These slides are **NOT** to be used as a replacement for student notes.
These **slides** are sometimes **vague and incomplete on purpose** to spark class discussions

JEE – Design Patterns

Data Access Objects

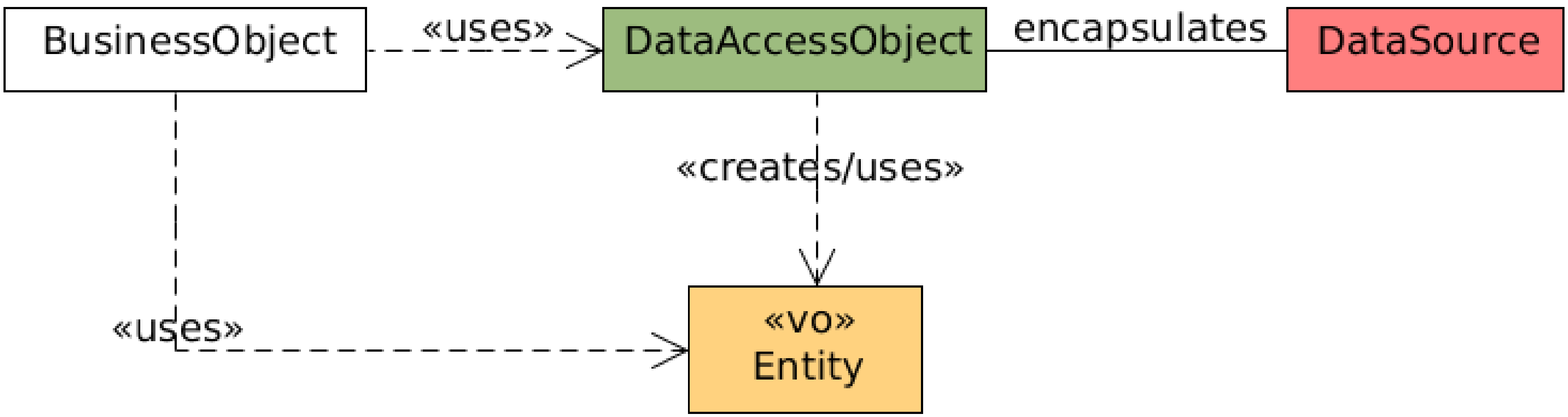
CS 446/646 ECE452
Jun 15th, 2011

WATERLOO
CHERITON SCHOOL OF
COMPUTER SCIENCE

Motivation

Intent

- abstract access to data repository



Motivation

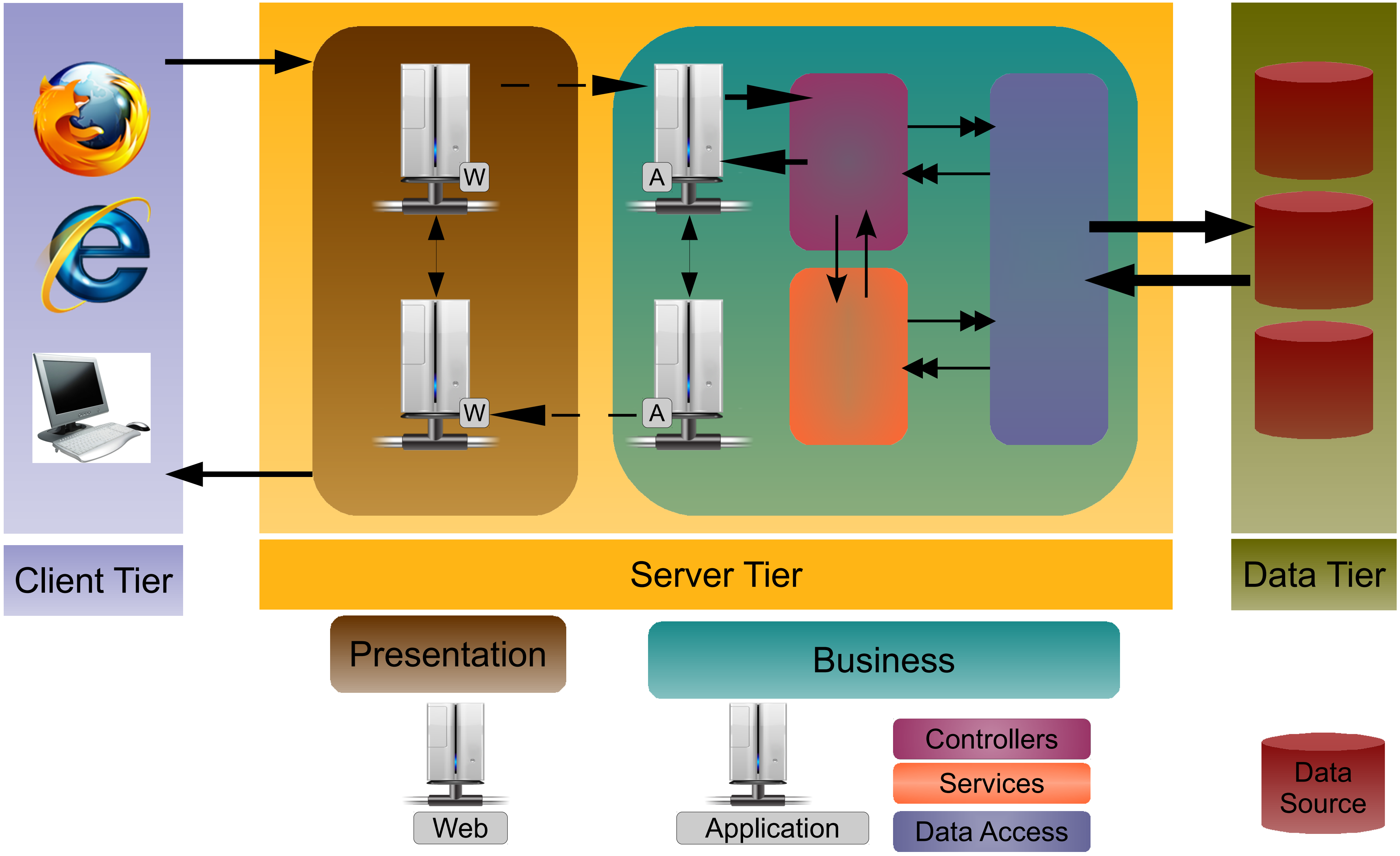
Forces

- enterprise applications will work with **heterogeneous** data-stores
  - persistence storage APIs
  - type of the data-store
    - relational database, flat files, OO db, legacy systems
  - merging access logic into the components make them less portable

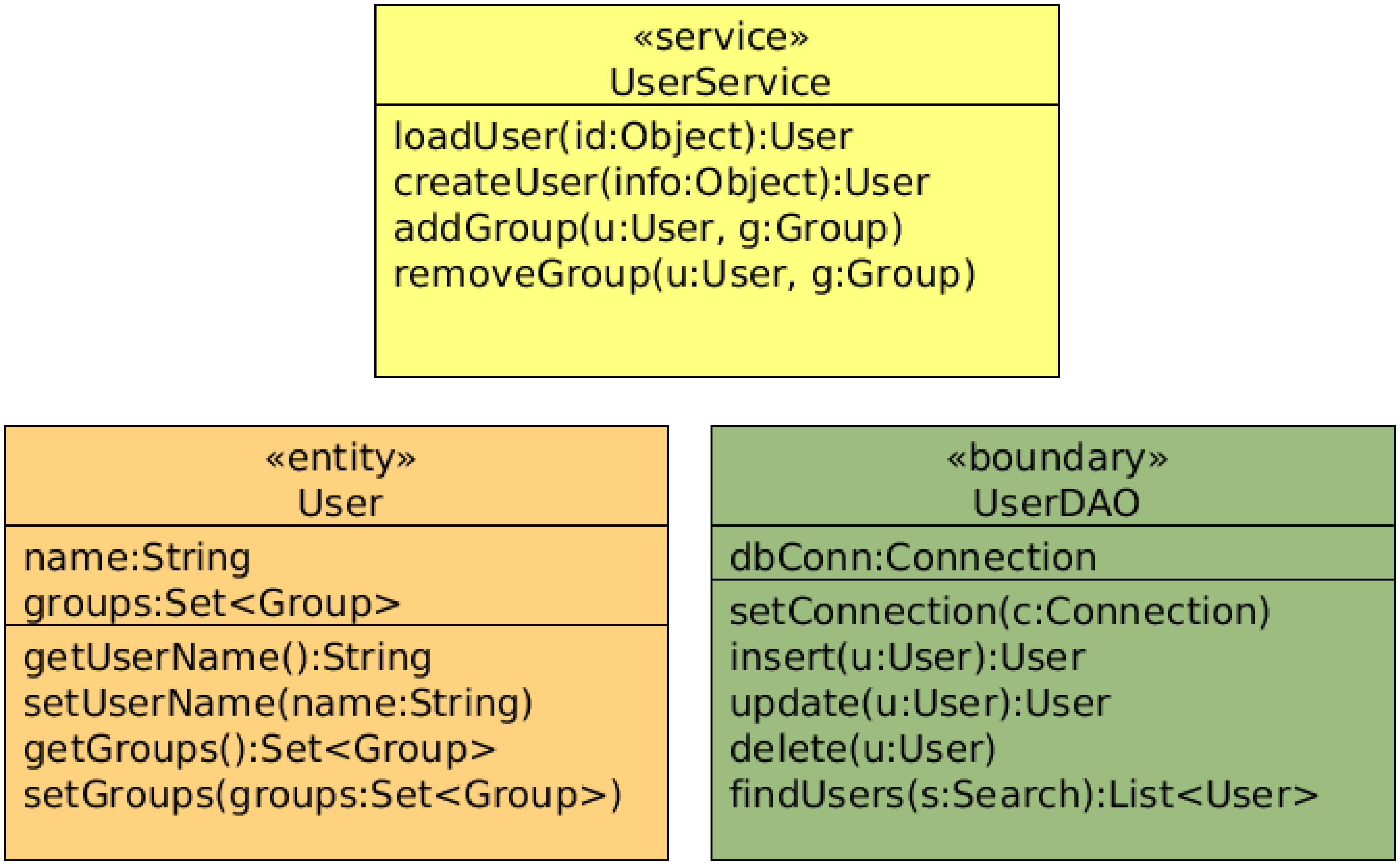
Solution

- Data Access Object (DAO) to **abstract** and **encapsulate** all access to the data source

Data Access Object



Data Access Object



«service»
UserService

loadUser(id:Object):User
createUser(info:Object):User
addGroup(u:User, g:Group)
removeGroup(u:User, g:Group)

«entity»
User

name:String
groups:Set<Group>

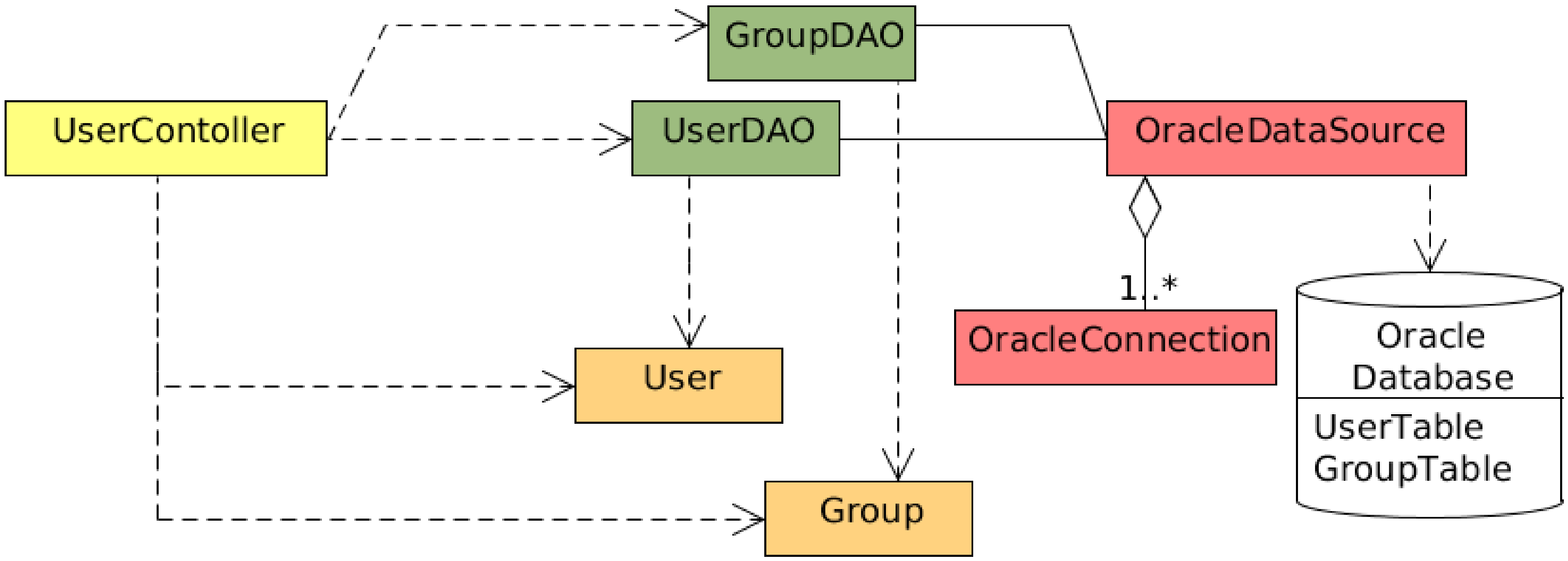
getUserName():String
setUserName(name:String)
getGroups():Set<Group>
setGroups(groups:Set<Group>)

«boundary»
UserDAO

dbConn:Connection

setConnection(c:Connection)
insert(u:User):User
update(u:User):User
delete(u:User)
findUsers(s:Search):List<User>

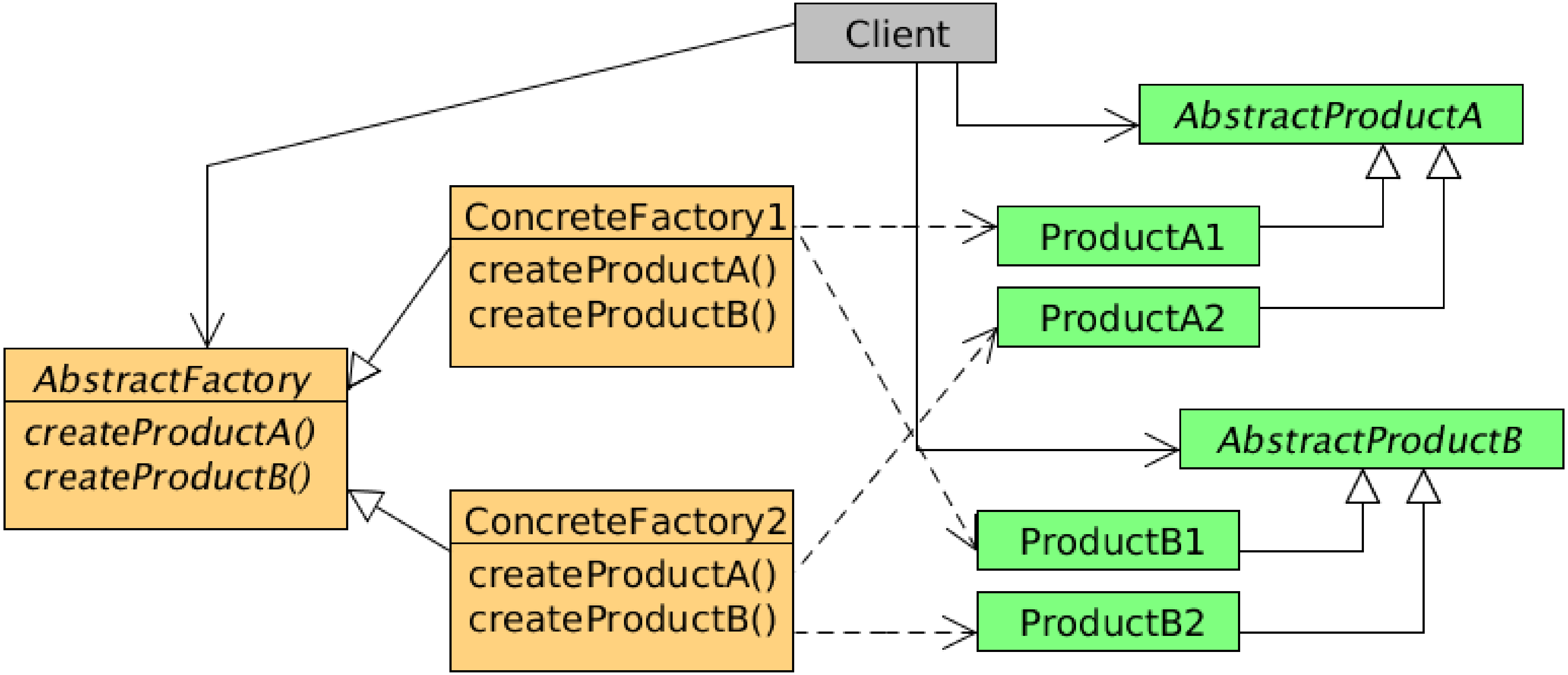
Example



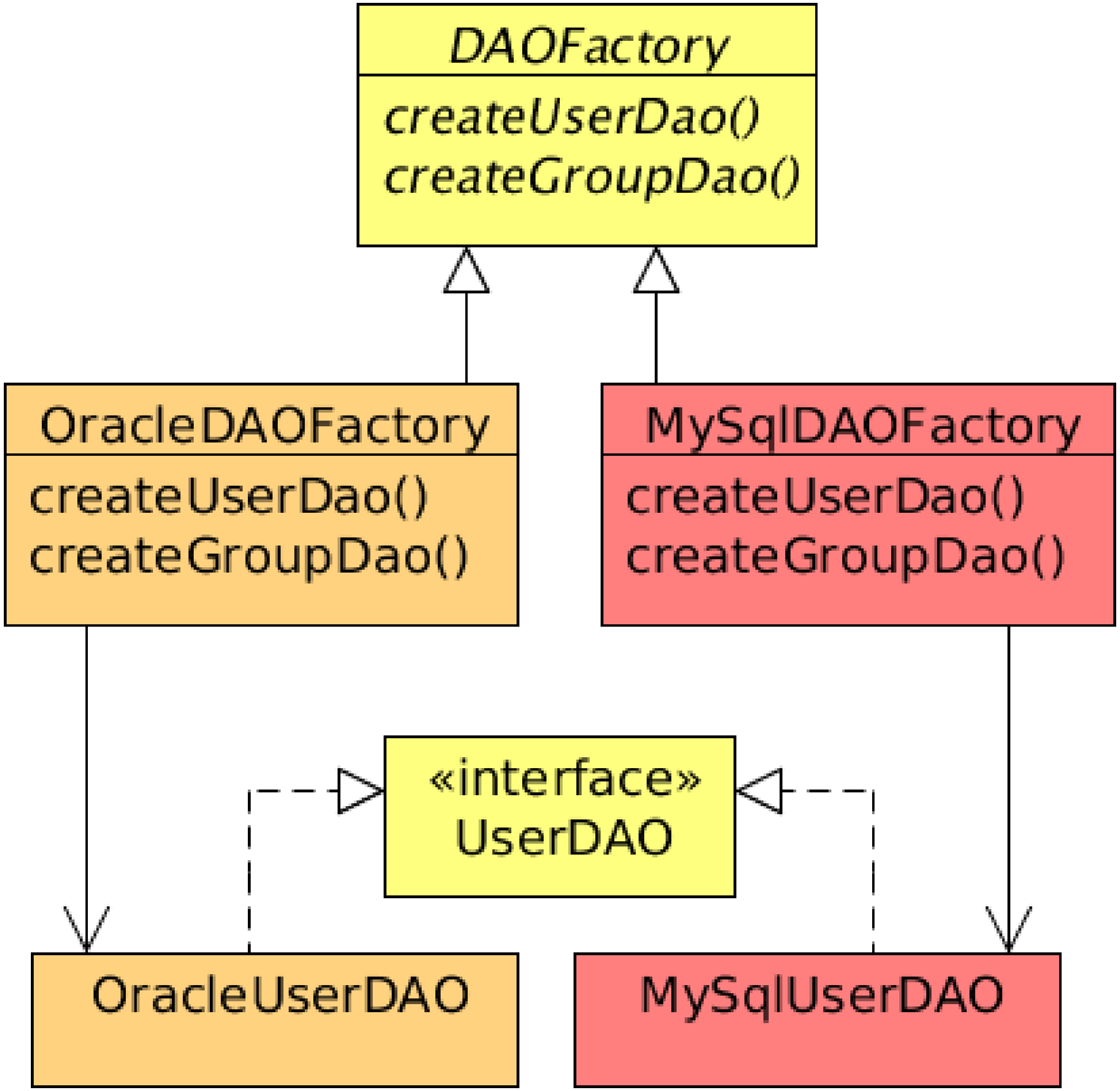
what design pattern should we use for OracleConnection?

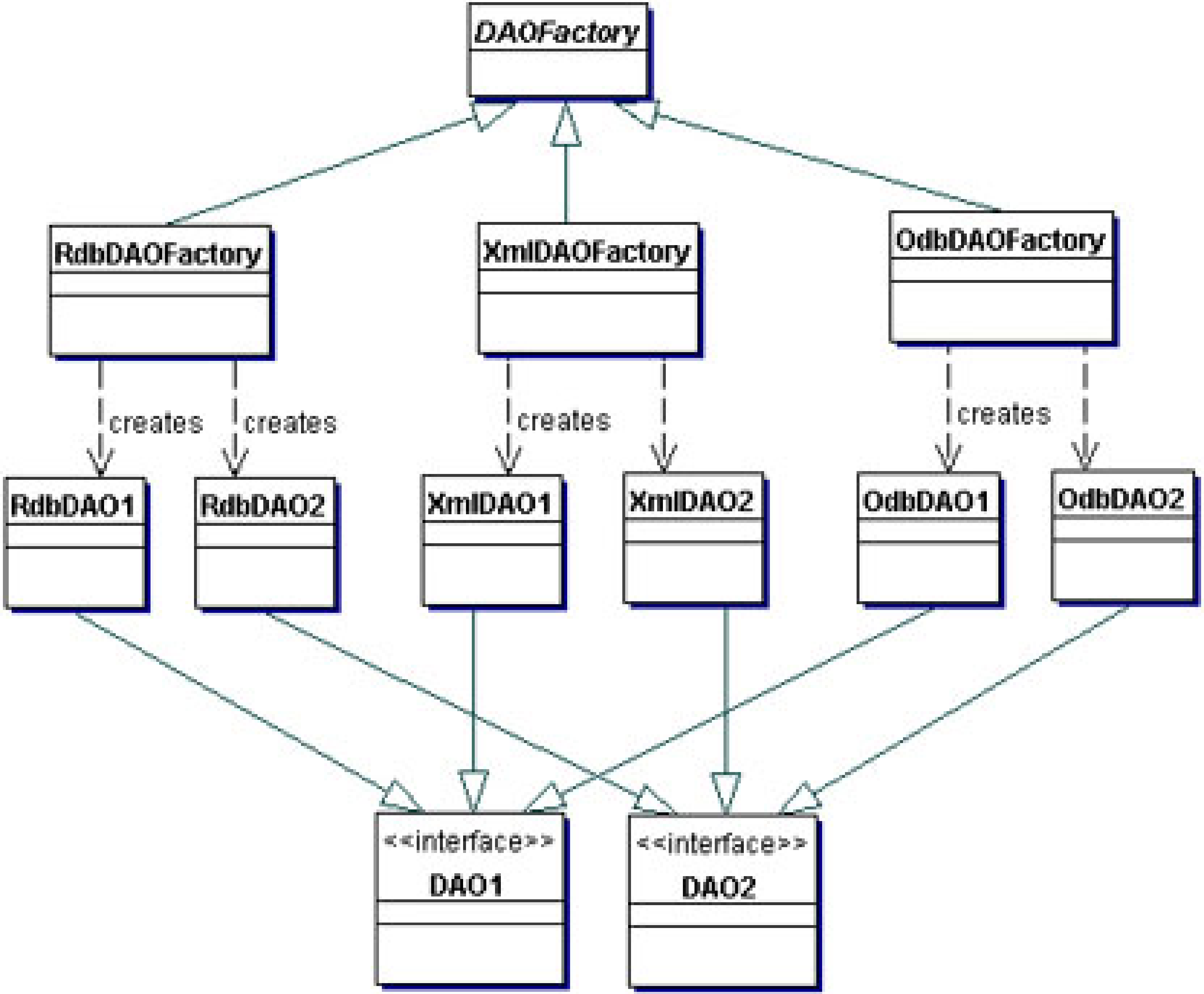
have we achieved database neutrality?

Use Abstract Factory

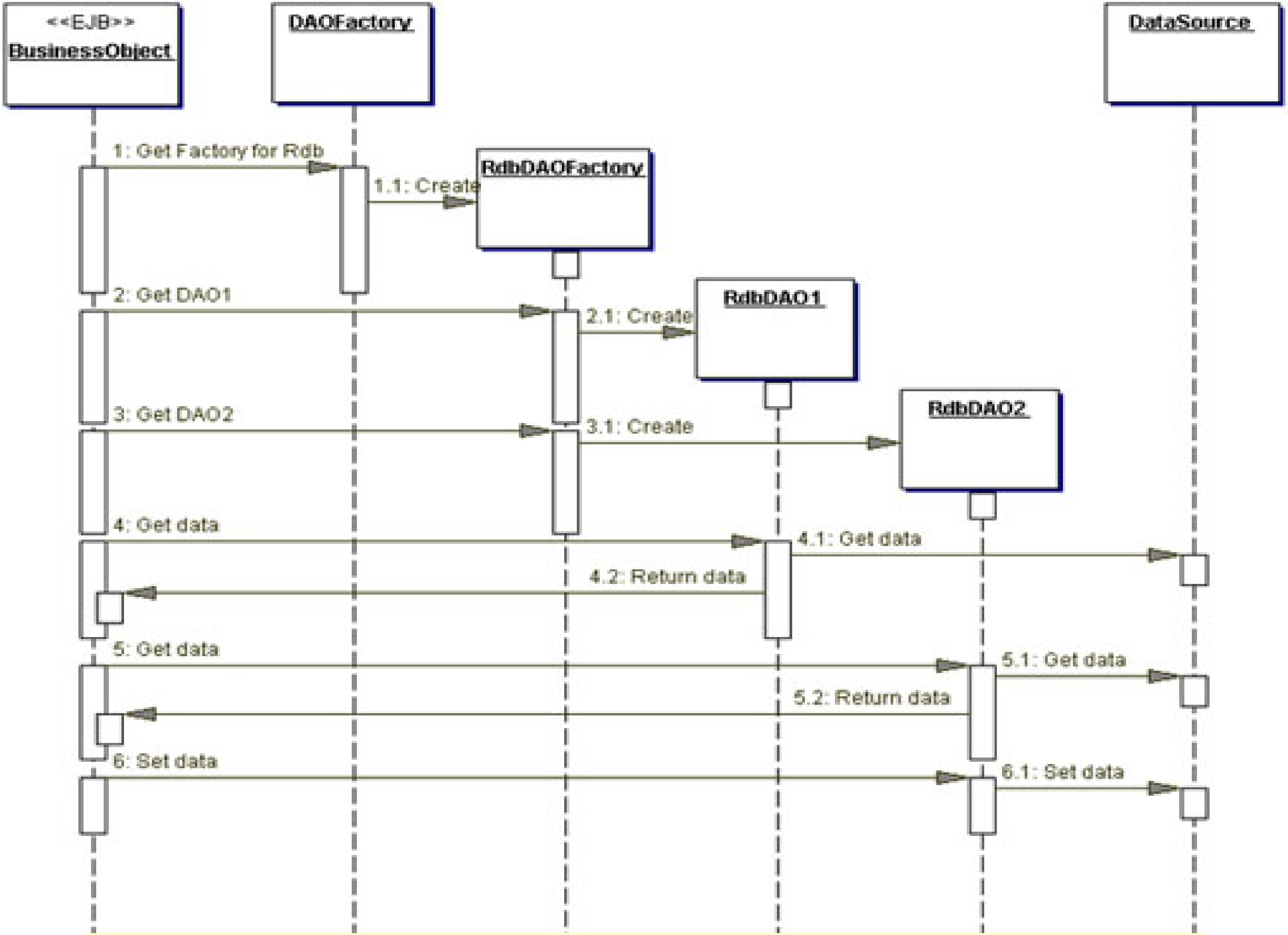


Use Abstract Factory





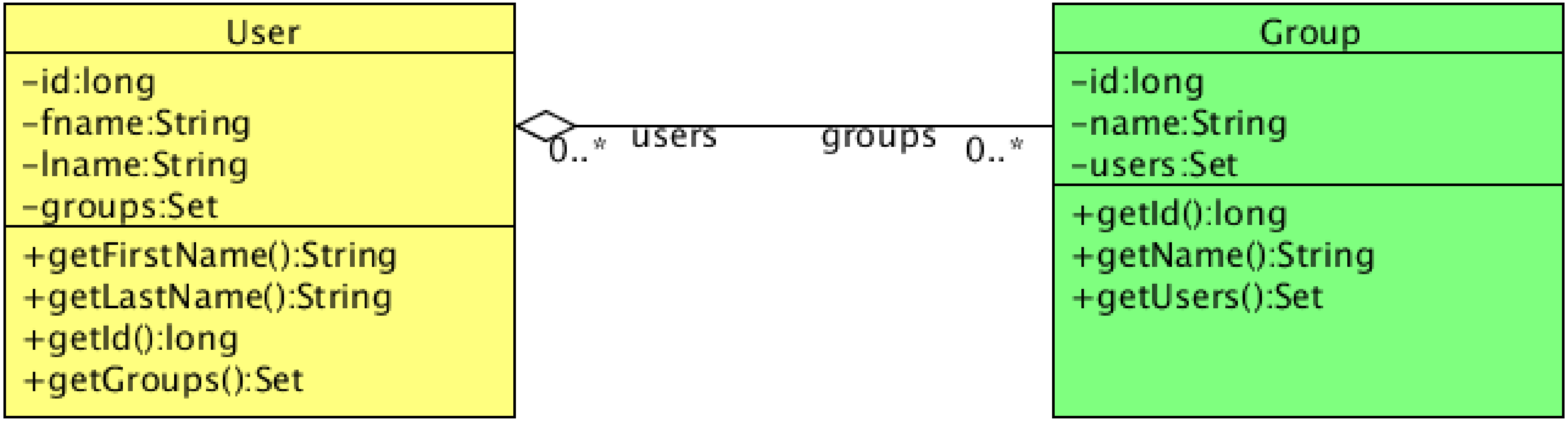
http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

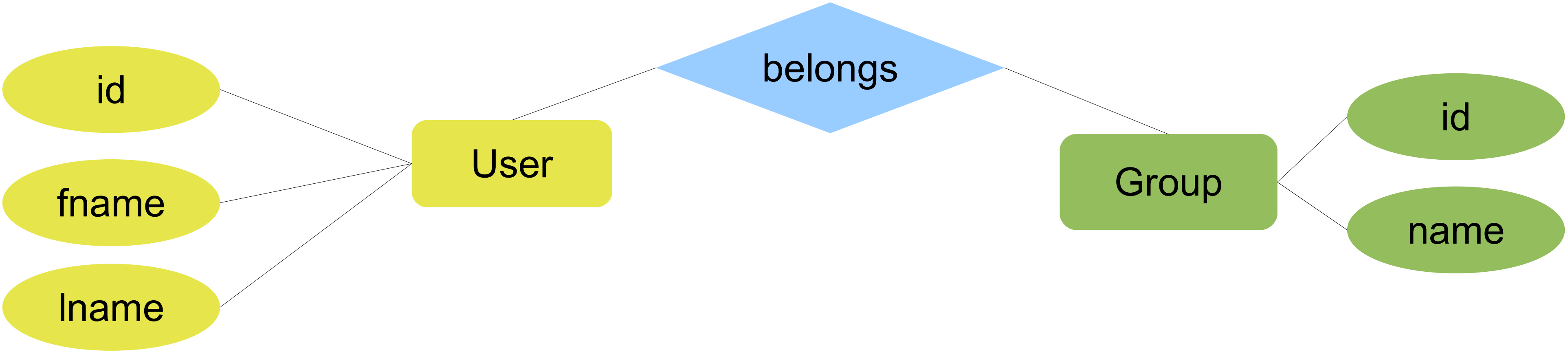


http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

Concerns

OR Mapping





Concerns

OR Mapping

- identifiers
- foreign keys
- **how can we maps objects to database tables?**
  - **example user & groups**
- **how do we map associations**
  - object to object
  - object to **list** of objects
  - object to **map** of objects

Concerns

OR Mapping

- mapping inheritance
  - table per class hierarchy
    - strategy: identify each subtype by a unique discriminator

ID	TYPE	AMOUNT
1	CREDIT	20.00
2	CASH	45.45
3	CHEQUE	2.00

what are some of the limitations?

Concerns

OR Mapping

- mapping inheritance
  - table per sub class
    - one table to represent the common attributes
    - one table per sub-class
    - need to maintain associations
  - independent table per subclass
    - **what do we loose here?**
    - probably most flexible

Concerns

Object Life Cycle

- entity objects are complex (composite)
  - delete a single user object
  - what are we deleting (?,?)
- cascading deletes
  - what qualifies for cascading delete
  - constraints (FK, not null etc...)

Tools

- Hibernate
- Oracle TopLink