Diseño de Aplicaciones Orientadas a Objetos

Ing. Fernando Soriano
Cátedras: Sistemas de Información II y
Análisis y Diseño Orientado a Objetos
Universidad FASTA
fernandos@ufasta.edu.ar
-
www.FernandoSoriano.com.ar

Temario

- Diseño de paquetes
- OR Mapping
- www.FernandoSoriano.com.ar

Diseño de paquetes

- Objetivo: diseñar paquetes físicos robustos.

- A continuación se verán algunas guías para el diseño de grano fino de los paquetes.

Guía: Paquete vertical y horizontal cohesivo

- Es el principio básico de dividir en módulos funcionalmente cohesivos
- El acoplamiento interno de un paquete se puede medir como la relación entre el *Número de Relaciones Internas* y *El Número de Clases*. Este índice se denomina **Cohesión Relacional.**
- Donde las *Relaciones Internas* son las relaciones de atributos, parámetros, herencia e implementación de interfaces

Guía: Paquete vertical y horizontal cohesivo

- Cuanto más alto sea el índice, mejor o más cohesivo será un paquete.
- Un valor de CR bajo dice:
  - El paquete contiene elementos no relacionados pero no es relevante.
  - El paquete contiene elementos no relacionados y está mal diseñado.
  - Contemplar que puede haber agrupaciones de subconjuntos con alto CR, pero en realidad el global no lo es

Guía: Paquete de una familia de Interfaces

- Consiste en agrupar interfaces relacionadas dentro de un paquete

Guía: Paquete de clases inestables

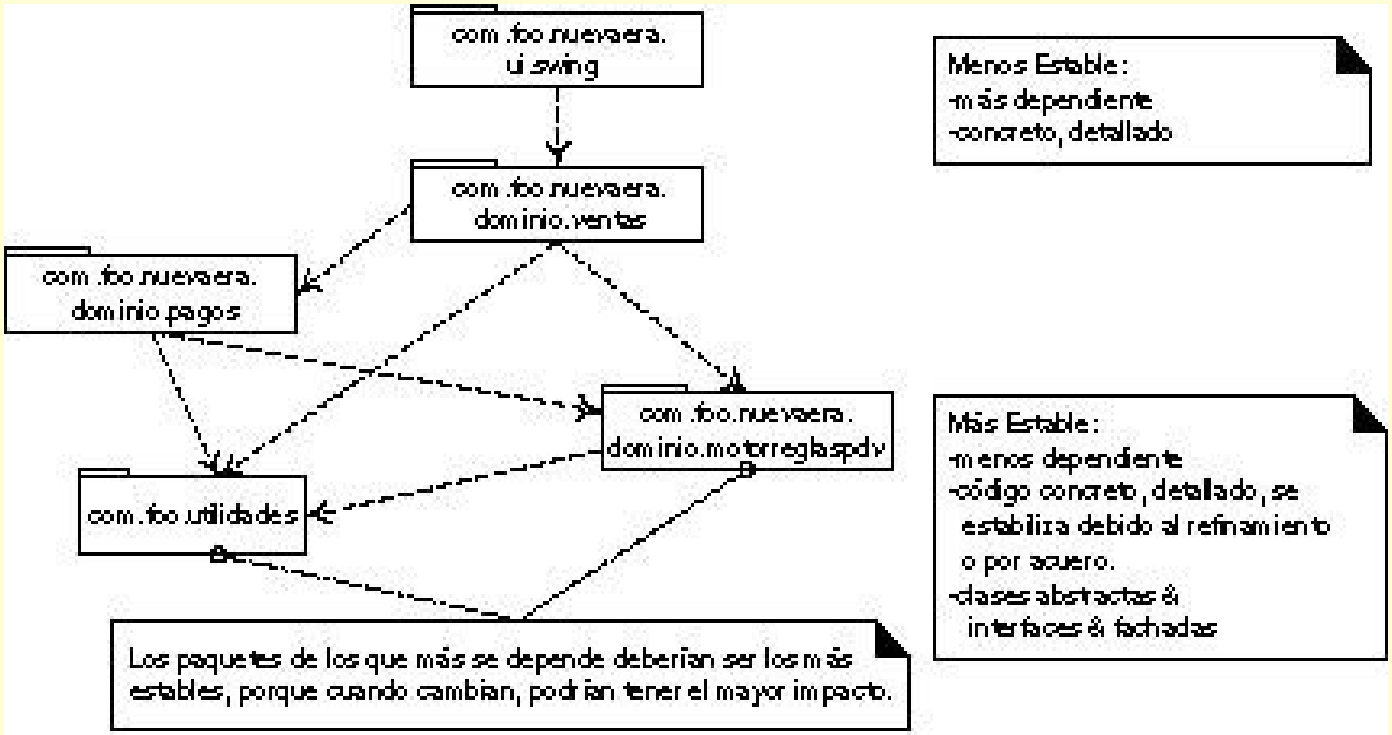
- Basados en el concepto de que los paquetes son la unidad para el equipo de desarrollo, es conveniente colocar las clases instables de un paquete en otro paquete

Guía: Paquetes más responsables y estables

- Los paquetes más responsables (de los que más se depende) debieran ser los mas estables.
- Consideraciones para lograrlo
  - Contiene en su mayor parte interfaces y clases abstractas.
  - No depende de otros paquetes o depende de otros muy estables.
  - Contiene código muy estable ya que ha superado todas las etapas de test.
  - Es obligatorio planificar los cambios

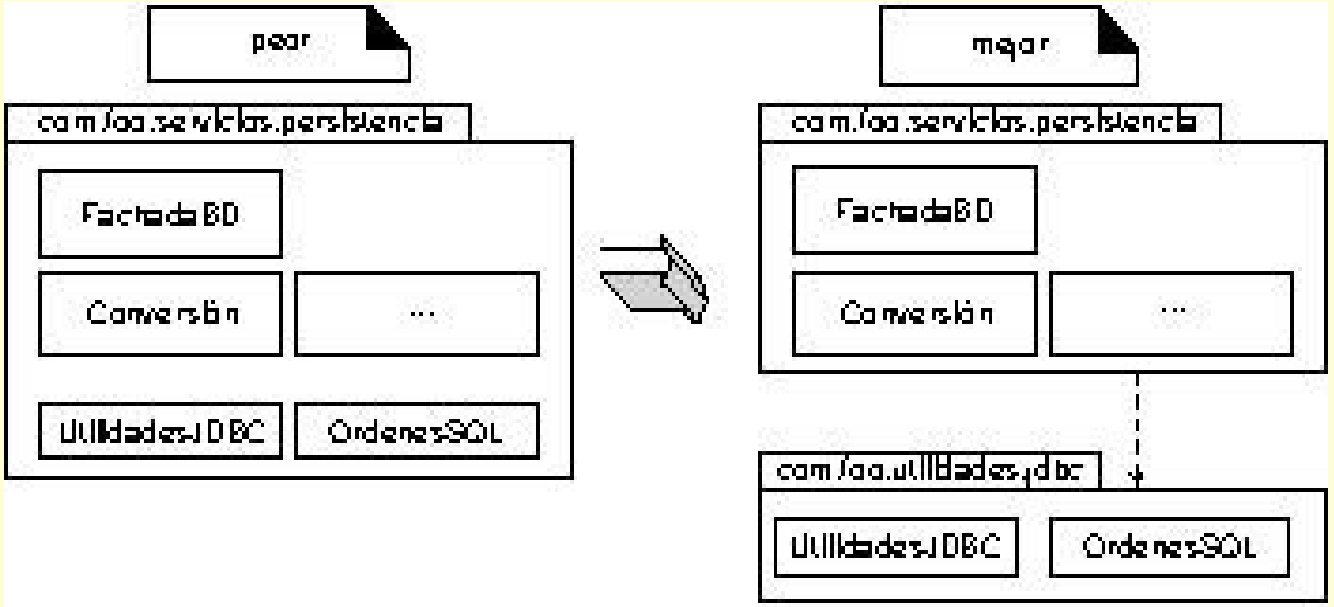
Guía: Paquetes más responsables y estables

- Ejemplo



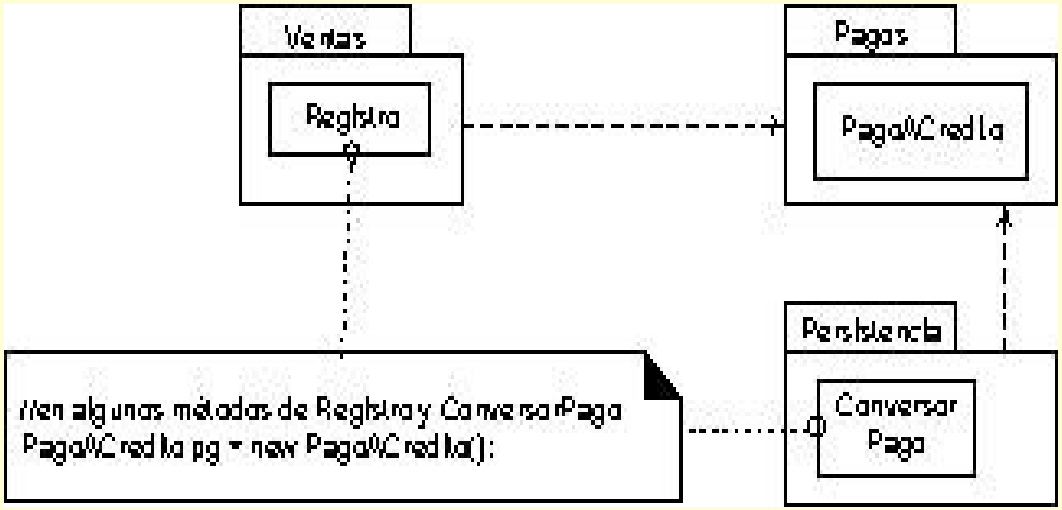
Guía: Separar los tipos independientes

- Ejemplo

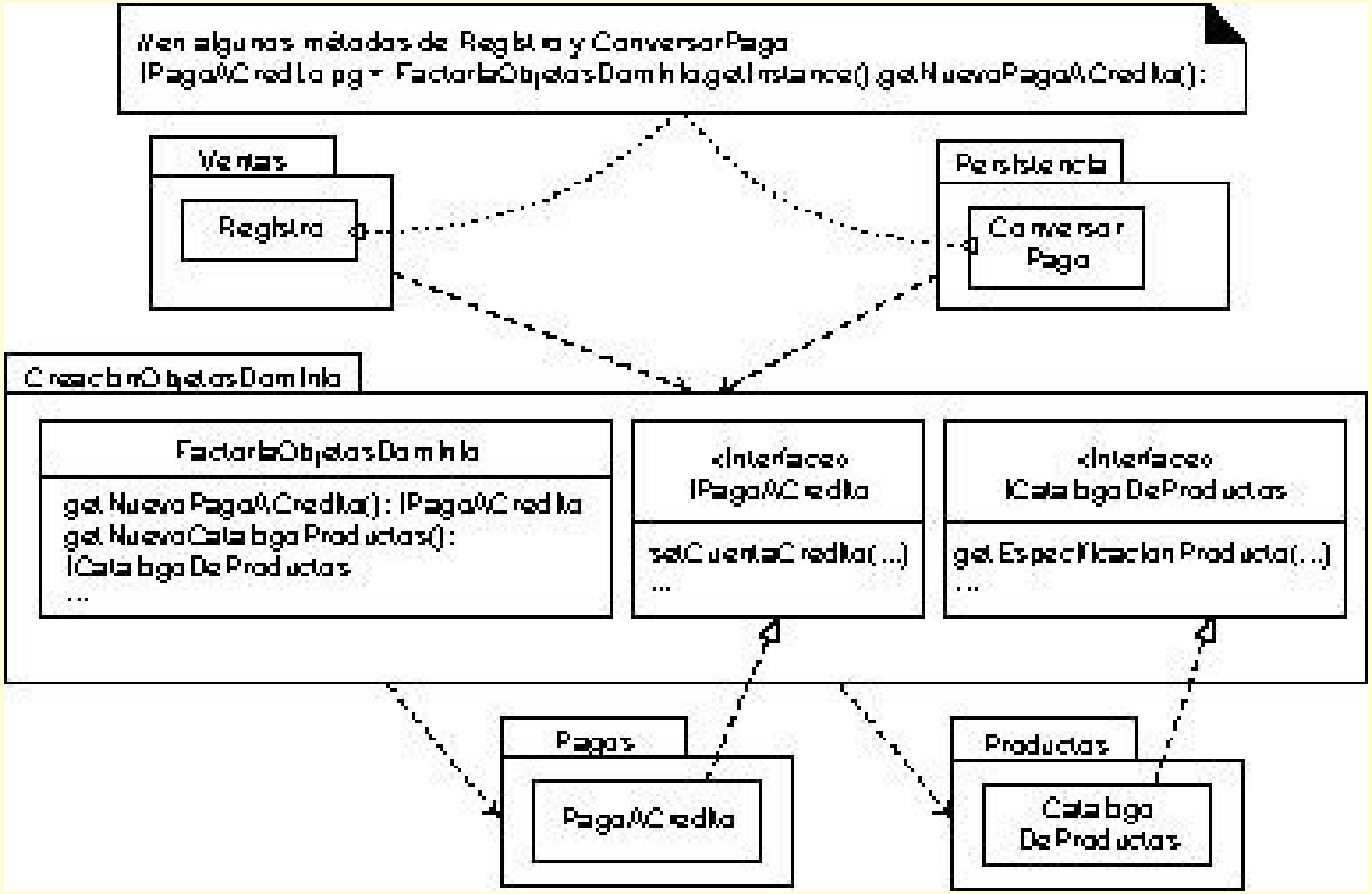


Guía: Usar factorías

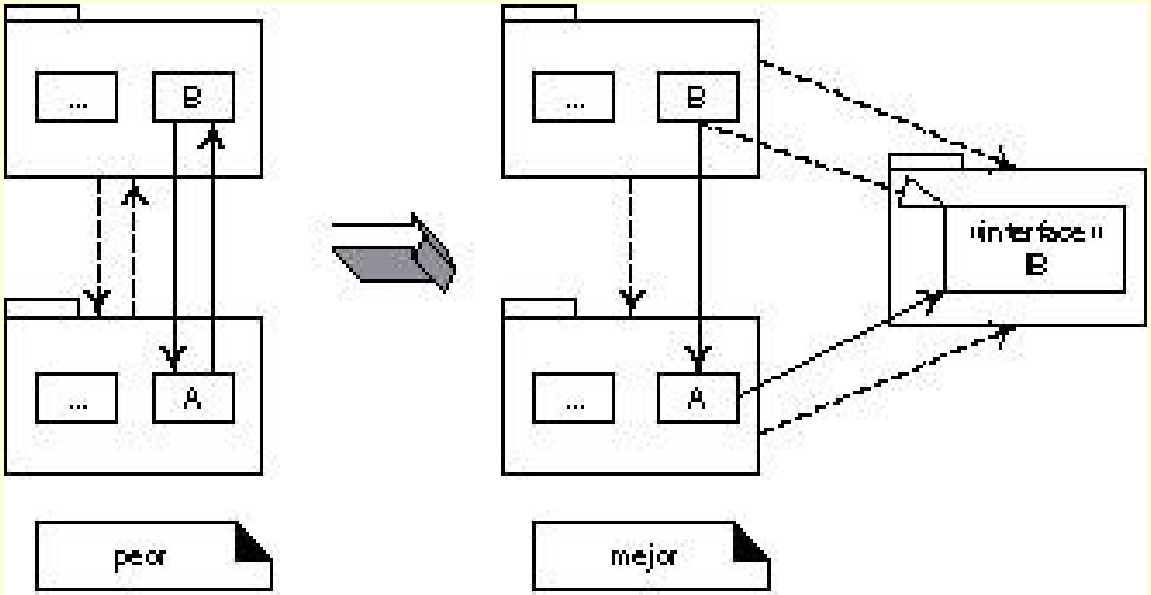
- Problema de acoplamiento



Guía: Usar factorías

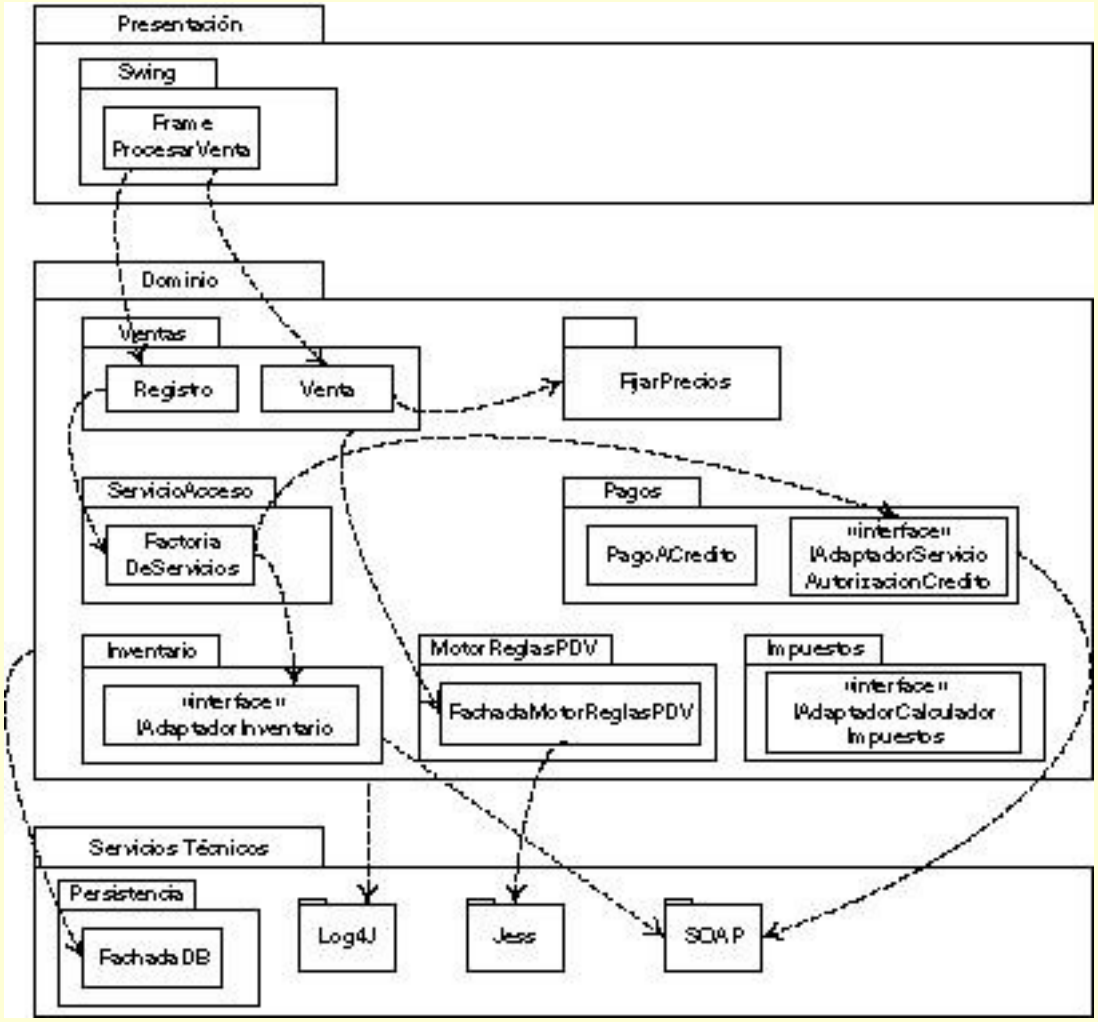


Guía: Paquetes sin ciclos



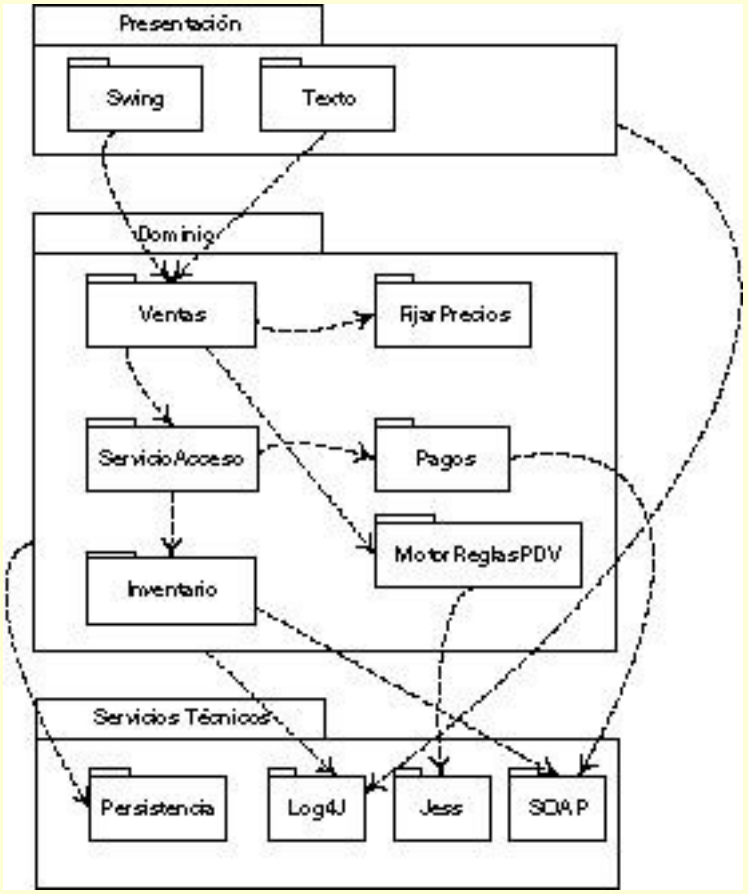
UML y paquetes

- Con detalles

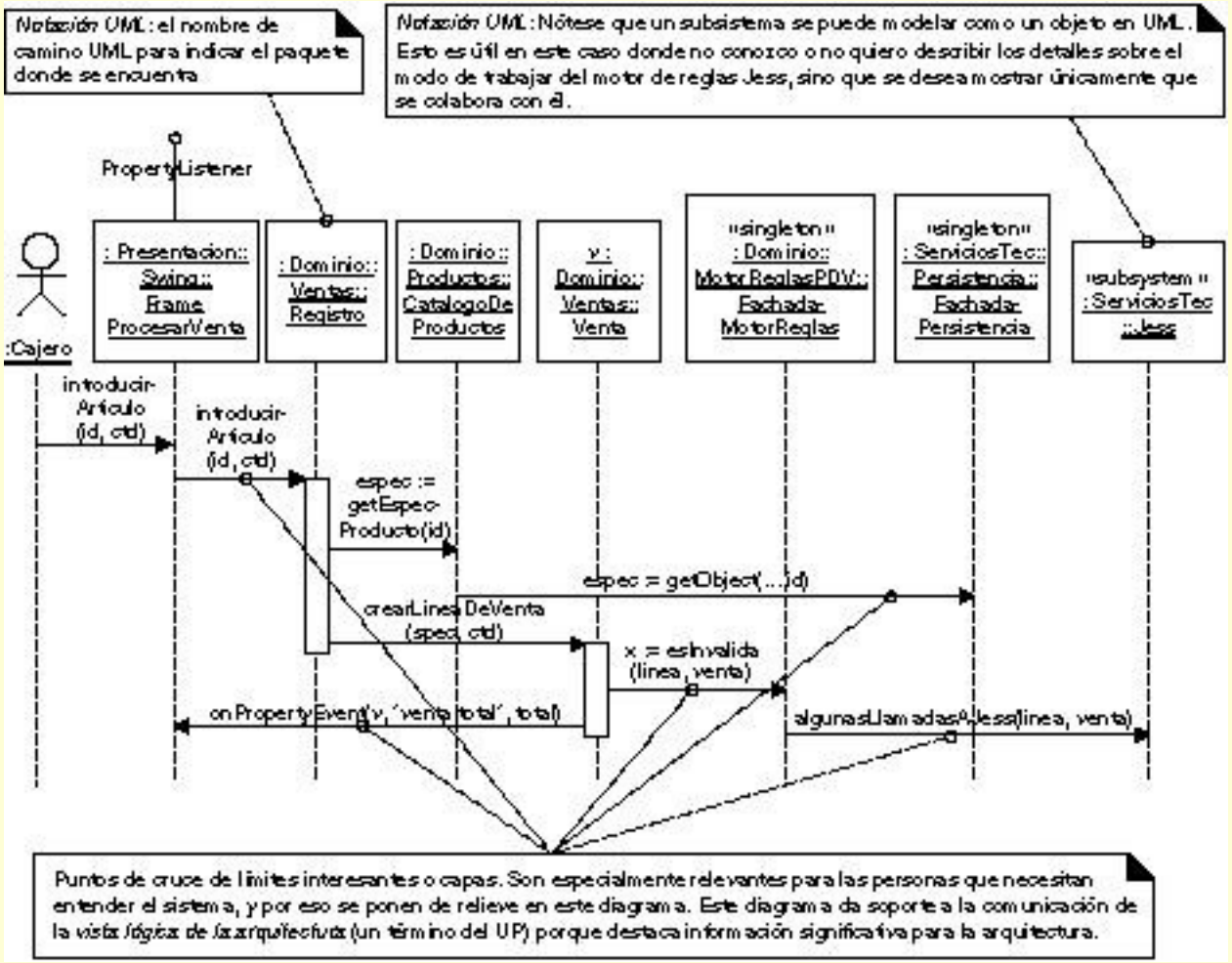


UML y paquetes

- Sin detalles



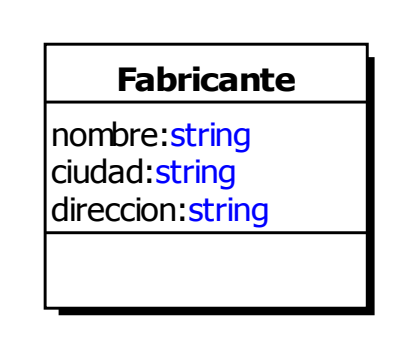
UML y paquetes



OR Mapping

- Problemática
  - Colecciones, herencia...
- Clases - Tablas

UML

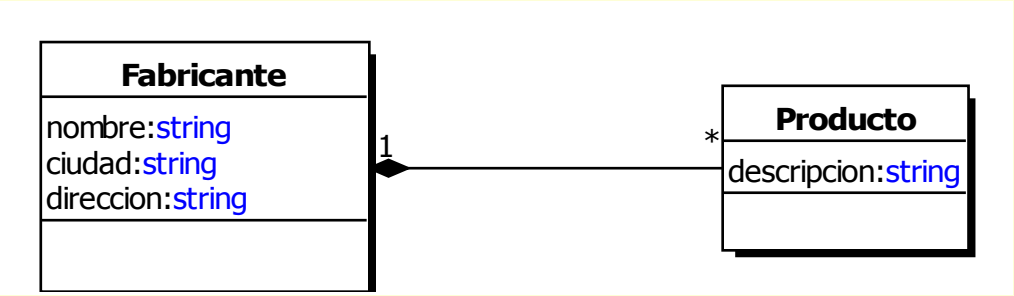


SQL

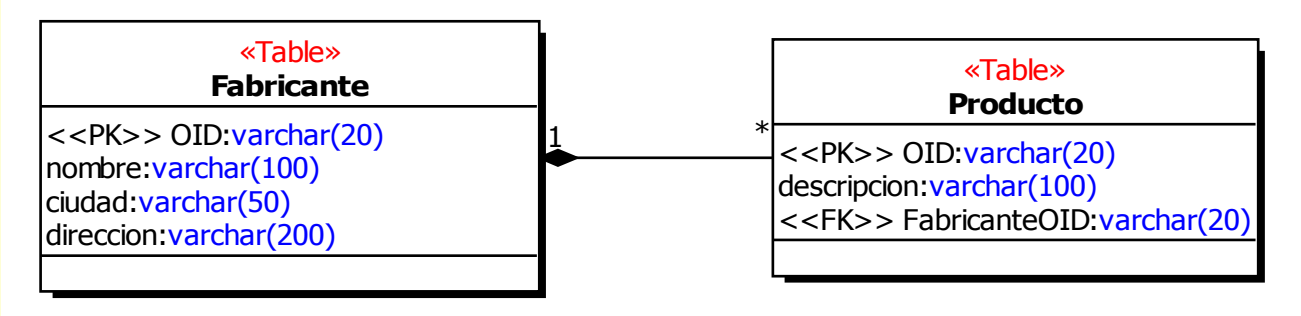
```
create table Fabricante(
        OID varchar(20),
        nombre varchar(100),
        ciudad varchar(50),
        direccion varchar(200)
)
```

OR Mapping

- Clases – Tablas

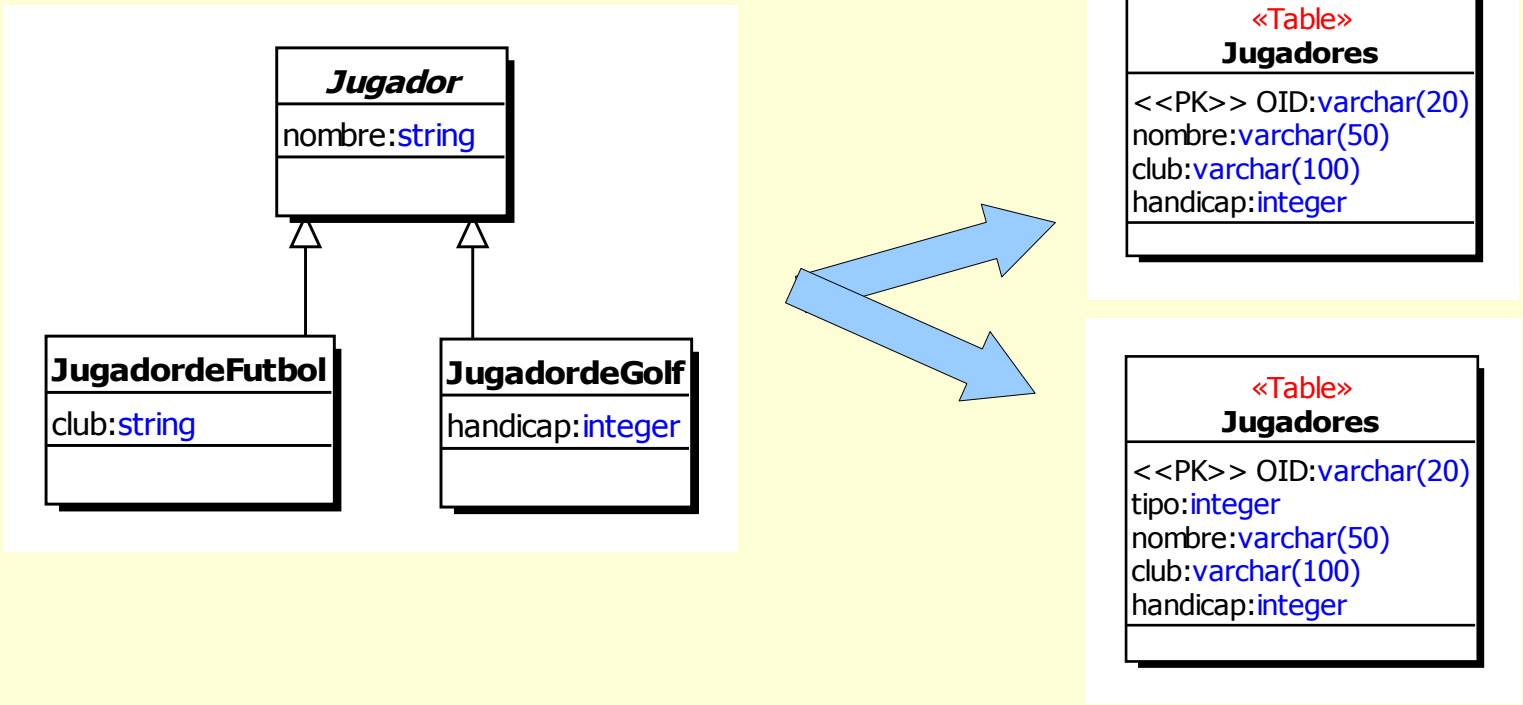


- Notación UML para modelo de datos



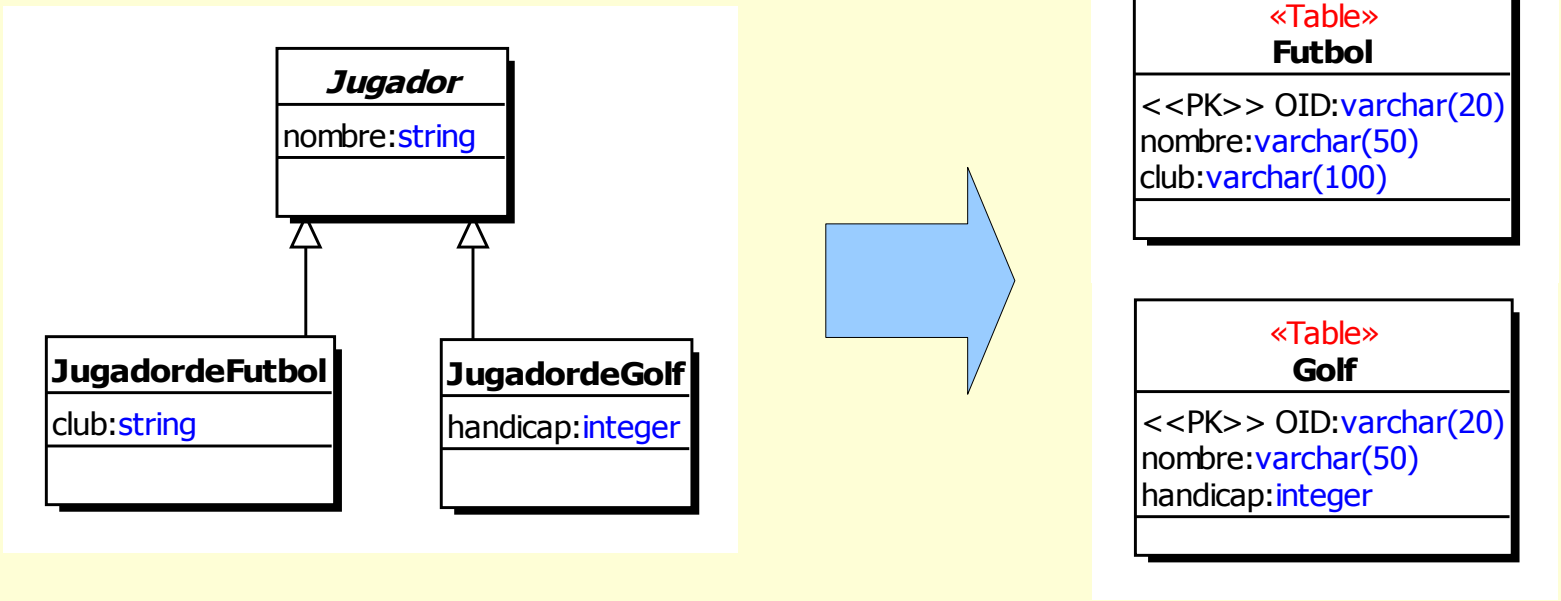
OR Mapping

- Clases (herencia) – Tablas
  - Solución unica tabla



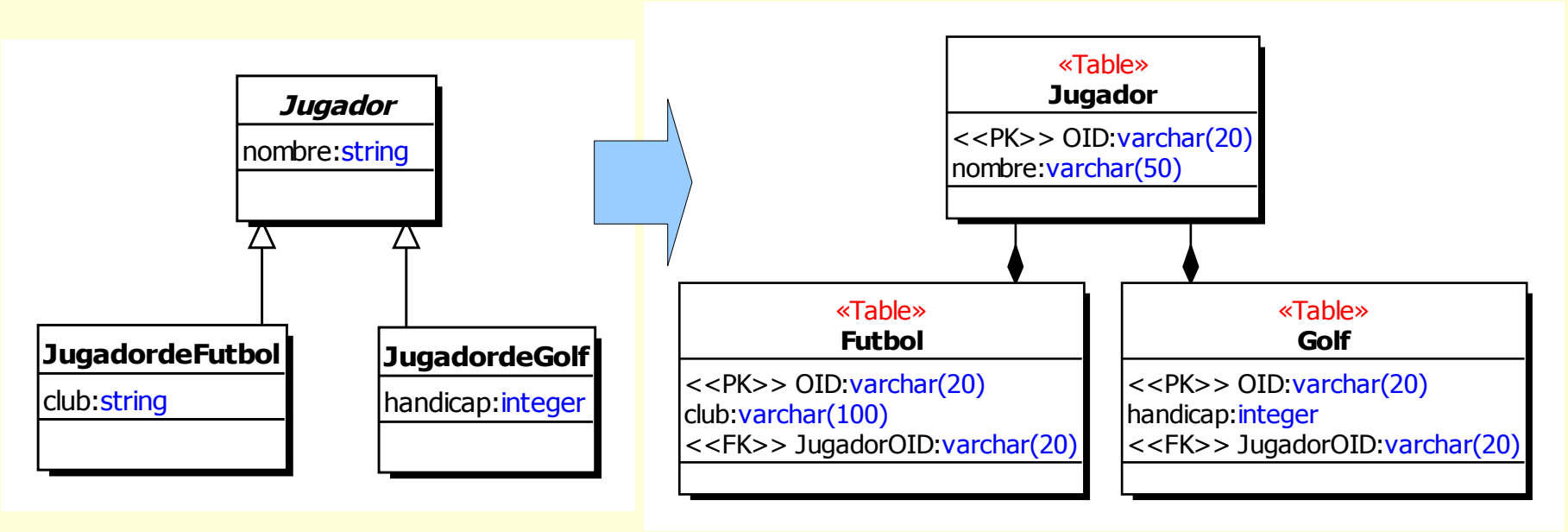
OR Mapping

- Clases (herencia) – Tablas
  - Solución 1 tabla por clase concreta



OR Mapping

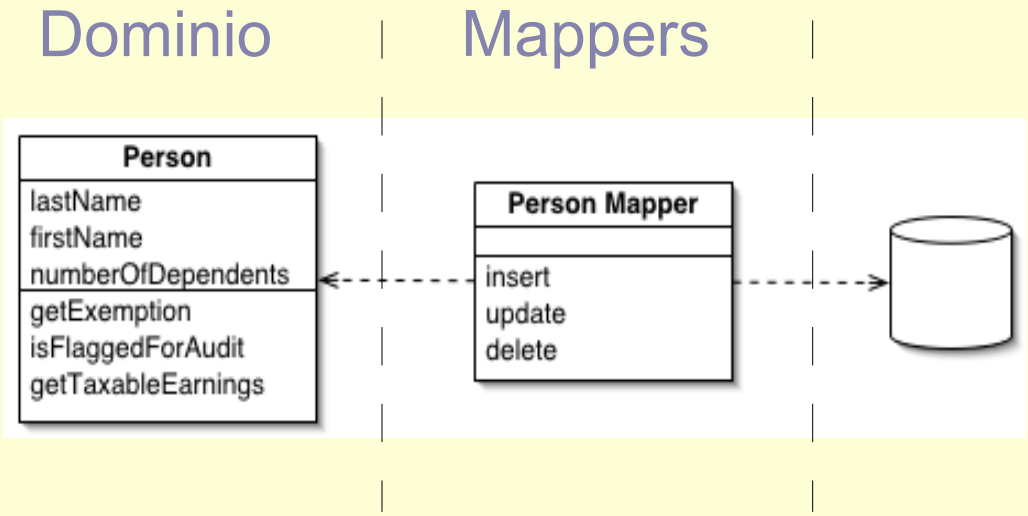
- Clases (herencia) – Tablas
  - Solución 1 tabla por clase



OR Mapping

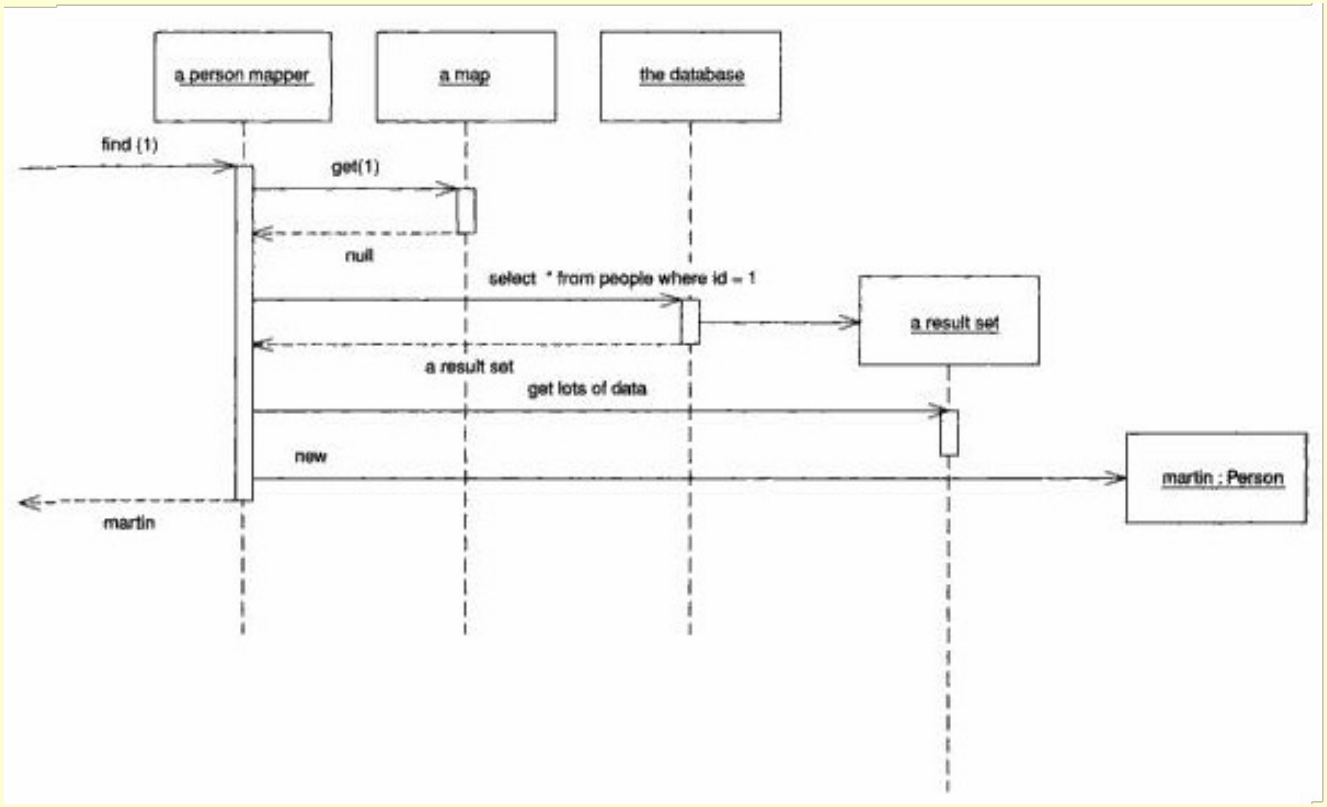
- Solución usando patrón Data Mapper (Fowler PEAA)
  - Permite transferir datos entre la App y el RDBMS (en ambas direcciones)
  - Permite aíslar una de otra.
  - Evita que los objetos de dominio persistentes conozcan la DB, necesiten SQL y conozcan el modelo de datos.
  - Inversamente, el Mapper no conoce la capa de dominio

OR Mapping


Dominio
Mappers
Person
lastName
firstName
numberOfDependents
getExemption
isFlaggedForAudit
getTaxableEarnings
Person Mapper
insert
update
delete

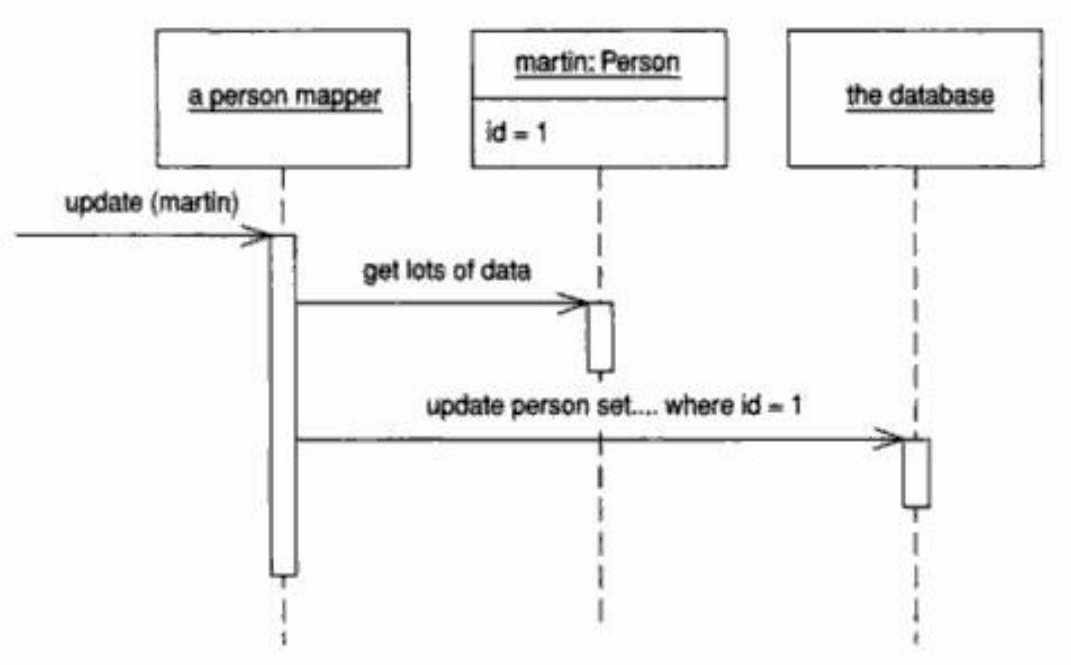
OR Mapping

- Instanciando un objeto



OR Mapping

- Actualizando la DB



www.FernandoSoriano.com.ar

- Ambler, S. The Object Primer. Second Edition. Cambridge University Press. 2001.
- Martin, J. Y Odell, J. Analisis y diseño orientado a objetos. Prentice Hall. 1992.
- Larman, C. UML y Patrones. 2da Edicion. Prentice Hall. 2002.
- Eckel, Bruce. Thinking in Java. Prentice Hall. 1998.
- OMG. UML Specification v1.3. 1999.
- Fowler, M. Patterns of Enterprise Application Architecture. Addison-Wesley. 2003.
- Gamma, Helm, Johnson y Vlissides. Design Patterns. Addison-Wesley. 1995.
- Booch, G. Rumbaugh, J. y Jacobson, I. The Unified Modeling Language User Guide. 1999.