INS MANUEL VÁZQUEZ MONTALBÁN

M03. Programació

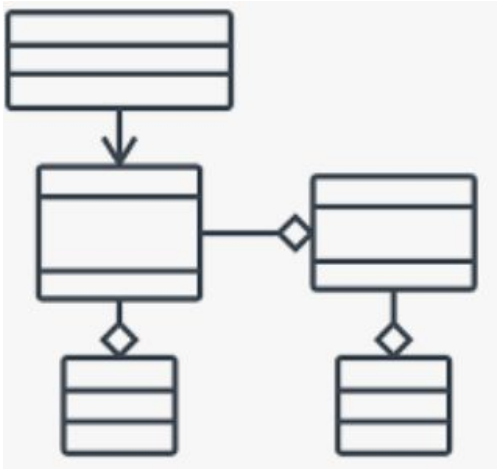
UF6. OOP. Introducció a la persistència en BD

1. UML

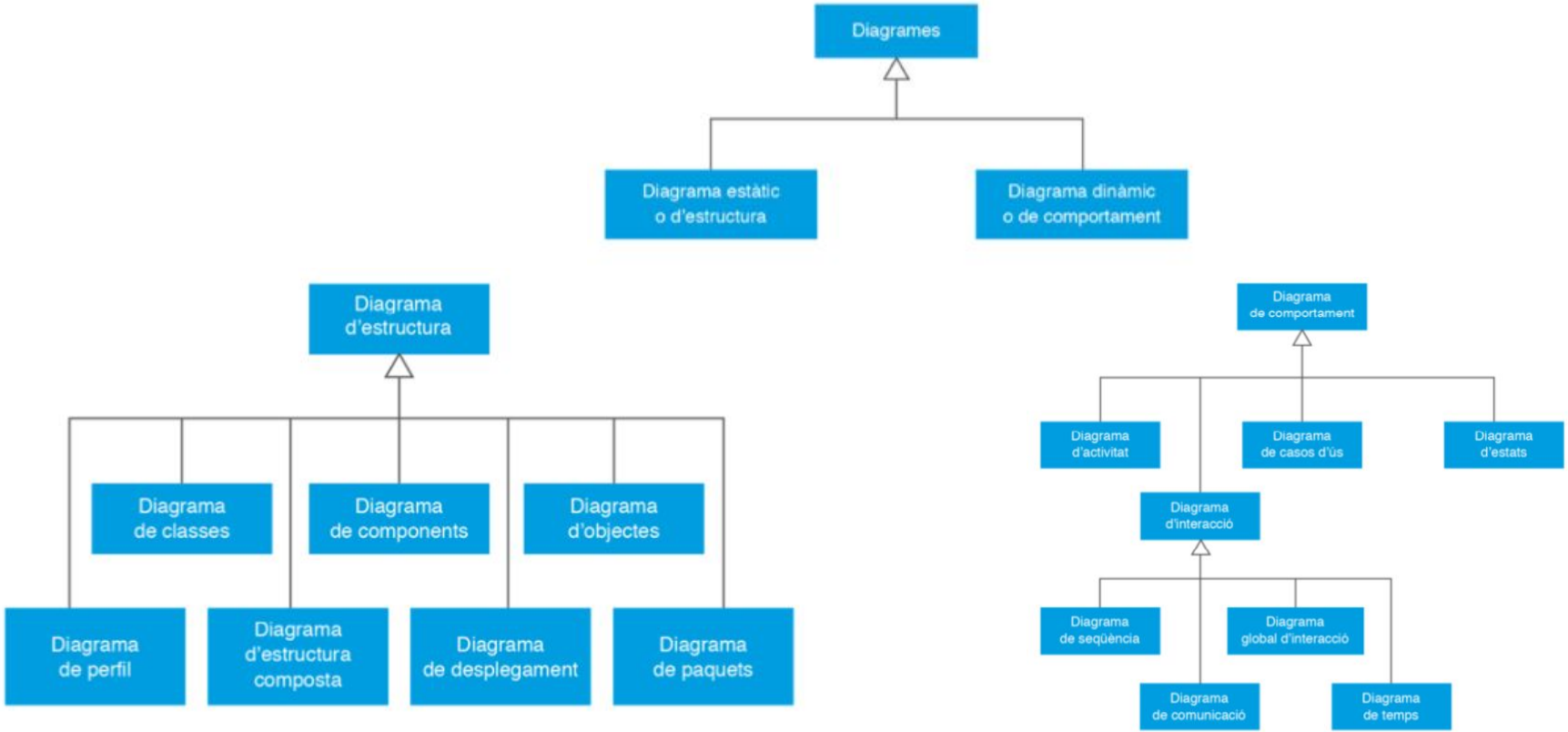
UML

De l'anglès Unified Modeling Language. Són un conjunt de notacions gràfiques que serveixen per especificar, dissenyar, elaborar i documentar models de sistemes i aplicacions informàtiques.

No es tracta d'un llenguatge de programació, sinó d'unes convencions de notacions per a representar la informació.



UML. Principals tipus de diagrames



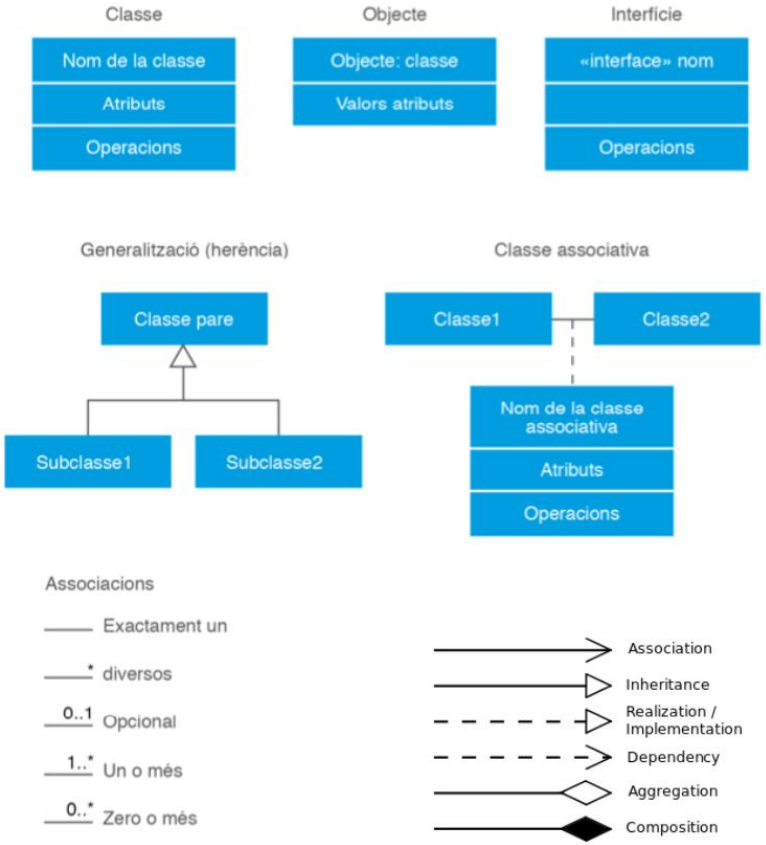
UML. Diagrama de classes

Un diagrama de classes representa les classes que seran utilitzades dins el sistema i les relacions entre elles.

Conceptes principals:

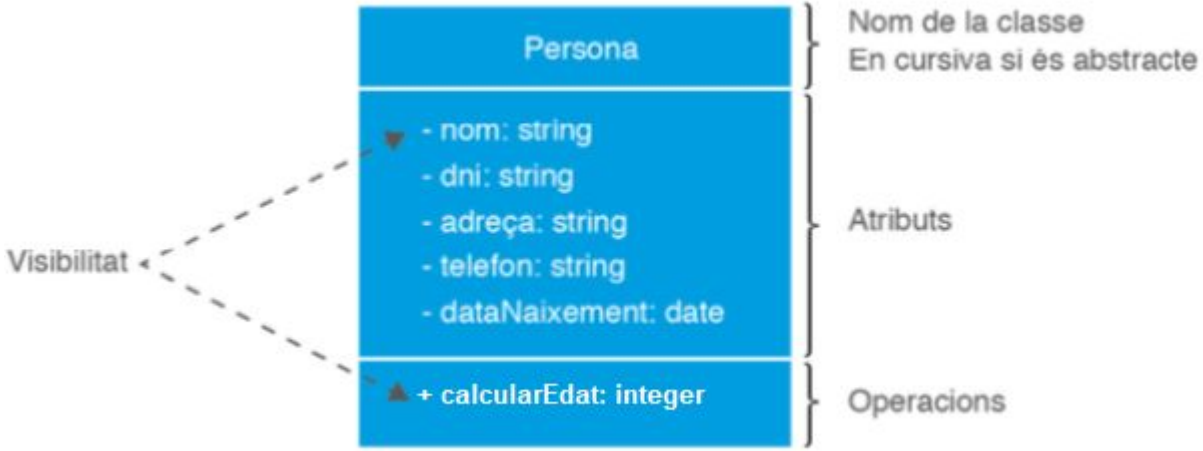
- Classe, atribut i mètode
- Visibilitat
- Objecte (Instanciació)
- Relacions (Herència, composició i agregació)
- Classe associativa
- Interfícies

UML. Diagrama de classes



UML. Diagrama de classes. Classe

Una classe descriu un conjunt d'**objectes** que comparteixen els mateixos **atributs** (característiques) i **mètodes** (comportaments) que representen aquests darres les accions que poden realitzar aquests objectes o que es poden realitzar sobre ells.



UML. Diagrama de classes. Visibilitat

La visibilitat d'un atribut o operació definirà l'àmbit des del qual podran ser utilitzats aquests elements. Aquesta característica es coneix com encapsulació.

Possibles visibilitats:

- **Public**: Per a tots els elements.
- **Private**: Només dins de l'objecte.
- **Protected**: Només dins de l'objecte i especialitzacions.
- **Default/Package private**: Només als elements del paquet.

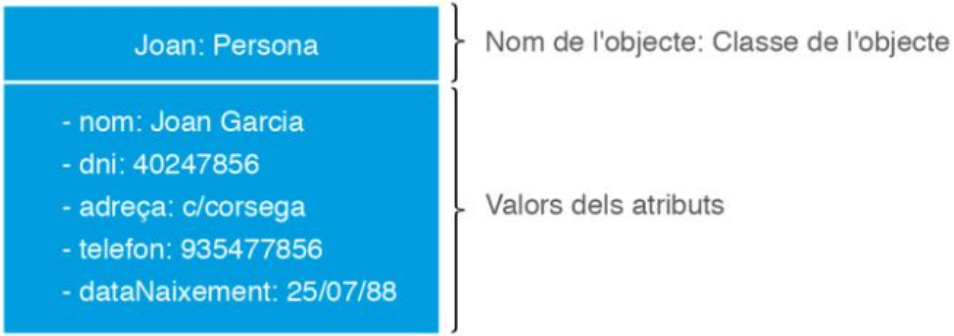
UML. Diagrama de classes. Visibilitat

La notació que s'utilitza per a determinar l'àmbit d'un atribut o mètode és la següent:

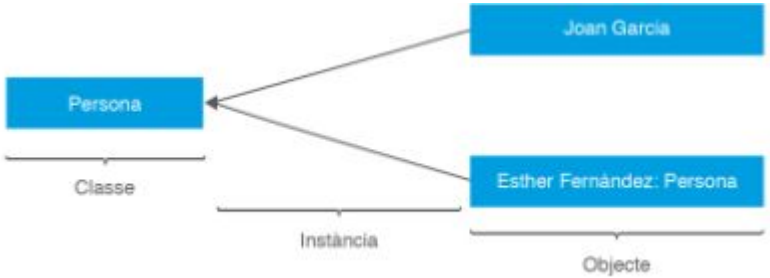
Símbol	Àmbit	
+	Public	L'element és accessible per tots els altres elements
-	Private	Només accessible en el mateix objecte
#	Protected	Només accessible en el mateix objecte i especialitzacions
~	Default	Només accessible en el paquet on està l'objecte

UML. Diagrama de classes. Objectes

Un objecte és la instanciació d'una classe:

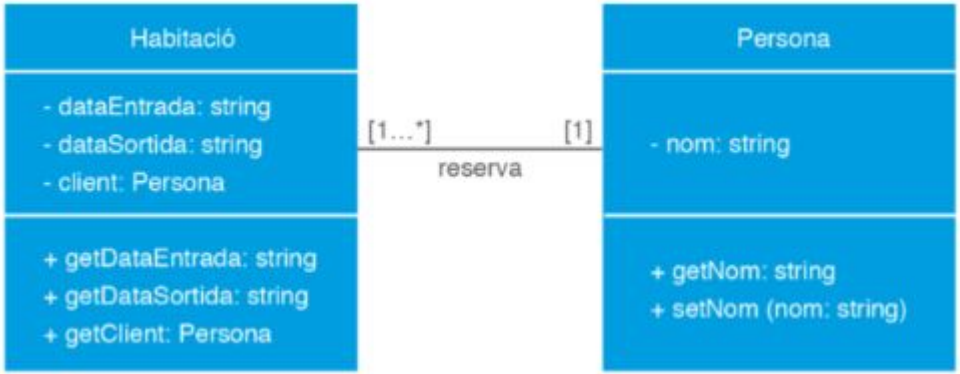


El procés d'instanciar es generar instàncies d'una classe com a objecte:



UML. Diagrama de classes. Relacions

Les relacions són elements imprescindibles en un diagrama de classes. Per relació s'entén que un **objecte *obj1* demani a un altre *obj2***, mitjançant un **missatge**, que **executi una operació** de les definides en la classe de l'obj2.



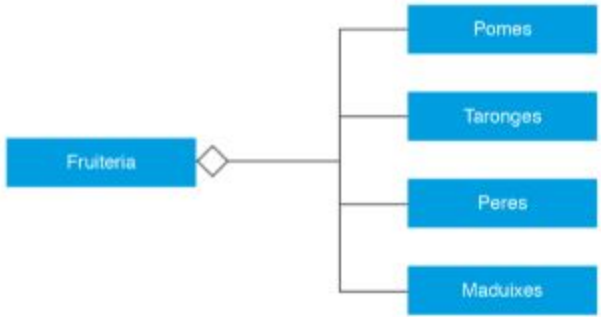
UML. Diagrama de classes. Relacions. Associació

Una associació defineix les connexions entre dos o més objectes, la qual cosa permet associar objectes que instancien classes que col·laboren entre si.

Es representen amb una **línia contínua** i defineixen la **multiplicitat**, representada amb el format [n...m].

UML. Diagrama de classes. Relacions. Agregació

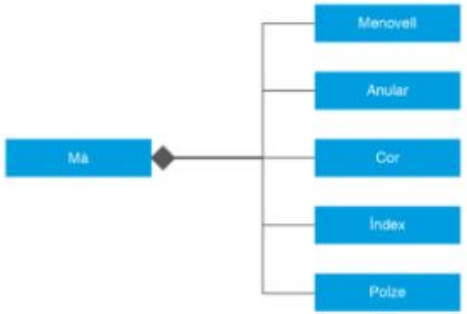
Es tracta d'una relació que es coneix com "tot - part". La idea és que un **objecte pot estar format per altres objectes**.



Es representa mitjançant una **línia contínua** que finalitza en un dels extrems per un **rombe buit**, sense omplir. El rombe buit s'ubicarà a la part de l'objecte base.

UML. Diagrama de classes. Relacions. Composició

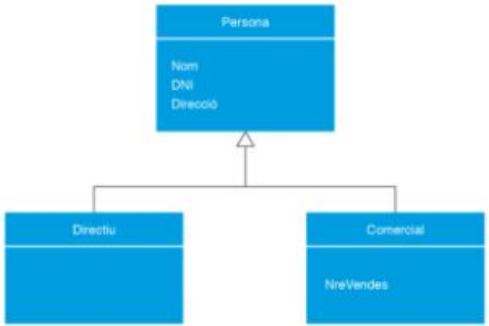
A diferència de l'agregació, si deixa d'existir l'objecte base, deixaran d'existir també els objectes inclosos.



Es representa mitjançant una **línia contínua** que finalitza en un dels extrems per un **rombe ple**.

UML. Diagrama de classes. Relacions. Herència

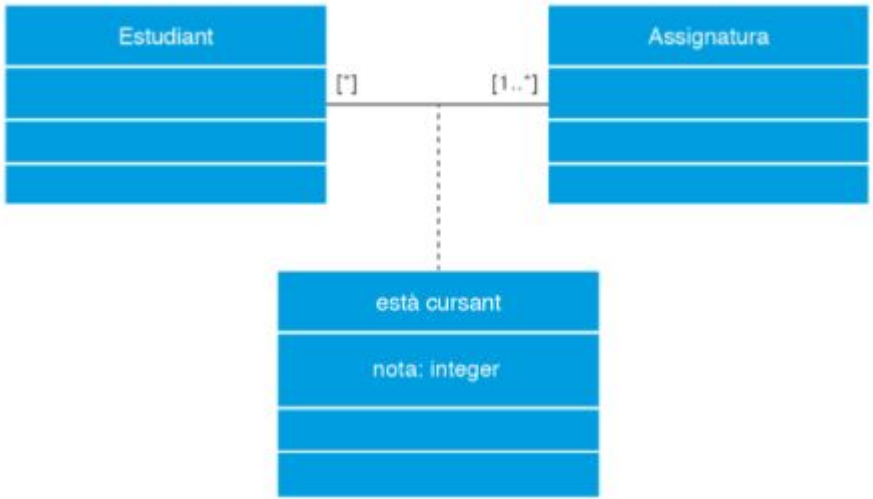
Una classe és anomenada classe mare (o **superclasse**). L'altra (o les altres) són les anomenades classes filles o **subclasses**, que **hereten els atributs i els mètodes** de la classe mare.



Es representa mitjançant una **fletxa** que surt de la classe filla i que acaba a la classe mare.

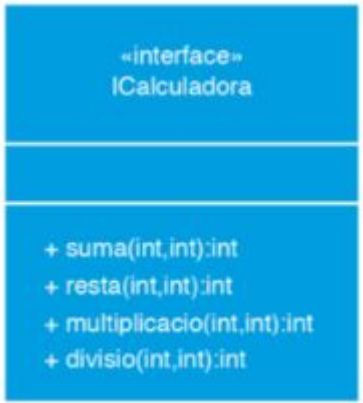
UML. Diagrama de classes. Classes associatives

Quan una associació té propietats o mètodes propis es representa com una classe unida a la línia de l'associació per mitjà d'una línia discontínua. Tant la línia com el rectangle de classe representen el mateix element conceptual: l'**associació**.



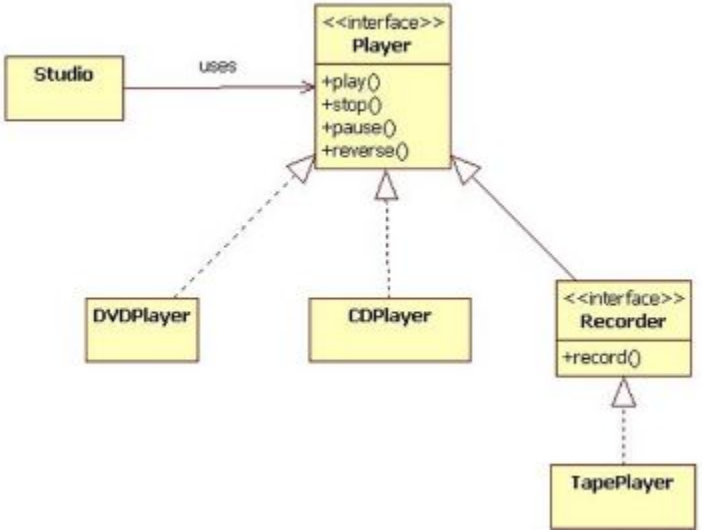
UML. Diagrama de classes. Interfaces

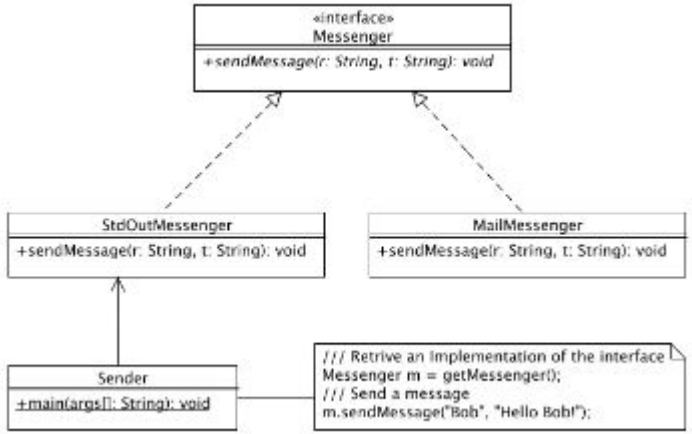
Una **interfície** conté la **declaració de les operacions sense la seva implementació**, que hauran de ser implementades per una classe o component.

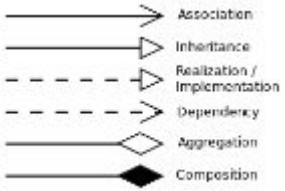


Es representa mitjançant una línia discontínua: 

UML. Diagrama de classes. Interfaces

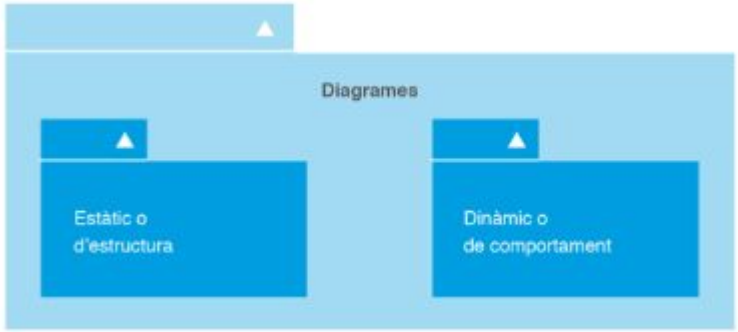


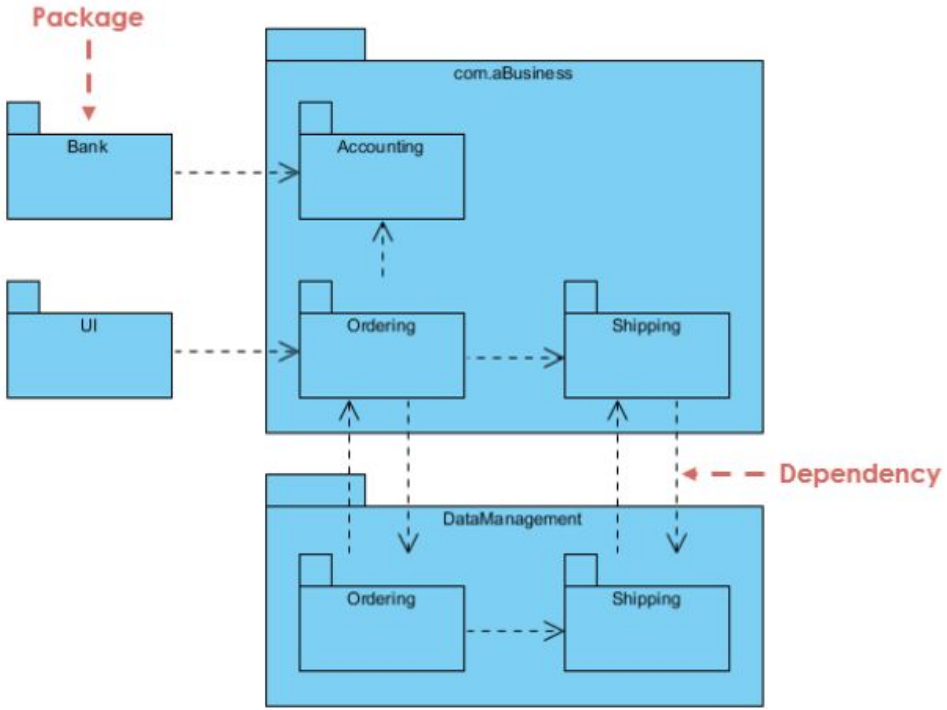




UML. Diagrama de classes. Packages

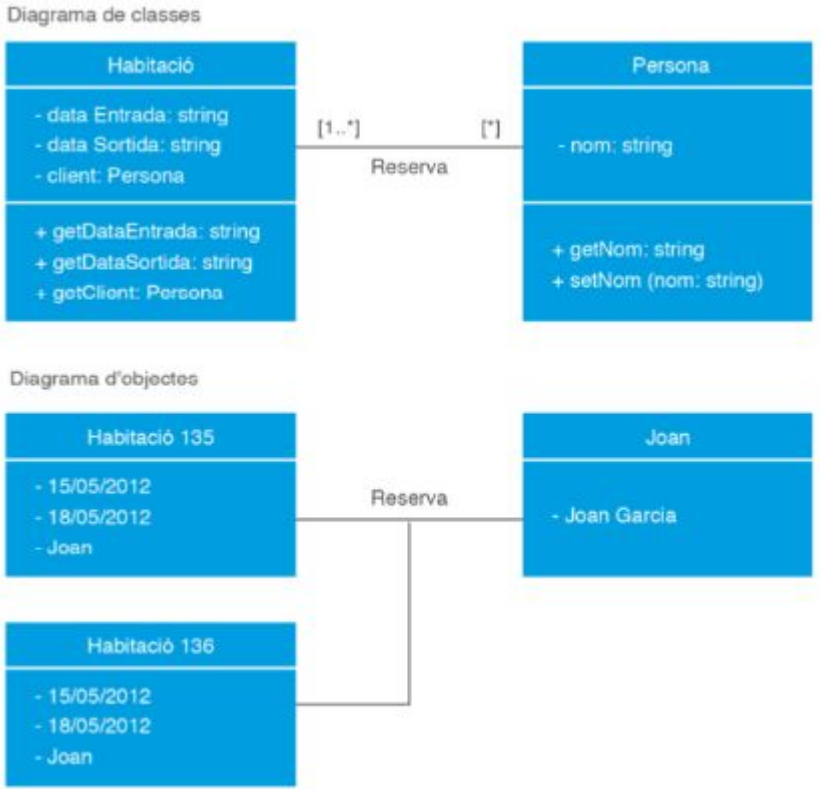
Els **packages** permeten organitzar els elements del model en grups relacionats semànticament; **un package no té un significat per si mateix**.





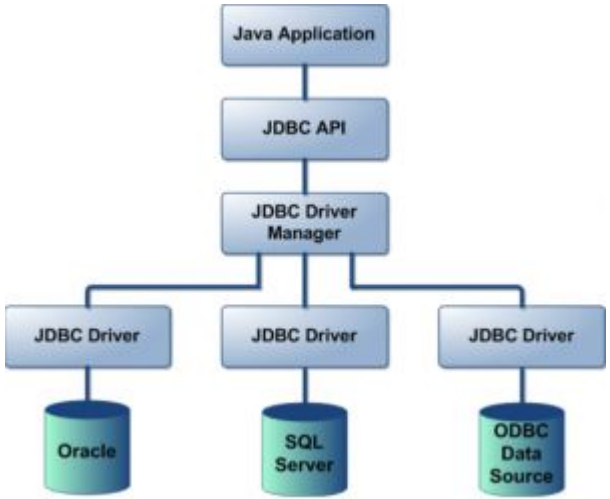
UML. Diagrama de classes. Diagrama d'objectes

El diagrama d'objectes només pot contenir **instàncies** i **relacions entre objectes**: enllaços i dependències que tinguin lloc entre instàncies.



2. Persistència en BBDD

Java Database Connectivity (JDBC)



La llibreria Java Database Connectivity (JDBC) és una API que proporciona accés a bases de dades des de Java. Consta de dos paquets: java.sql i *javax.sql*.

Per tal de poder treballar amb un sistema gestor de bases de dades determinat, necessitem un controlador (Driver) que faci d'intermediari entre la tecnologia JDBC i la base de dades.

Per tant, només hem de descarregar el controlador o driver concret per a aquesta plataforma. A l'hora d'utilitzar qualsevol SGDB que usi SQL les sentències són sempre les mateixes, només canvia el driver de connexió.

Java Database Connectivity (JDBC)

DriverManager

- Representa el gestor de drivers (component on es registren els drivers que volem utilitzar en la nostra aplicació).
- És el component encarregat d'obrir connexions a les bases de dades

Connection

- Representa una connexió contra una base de dades específica

Statement

- Representa una sentència SQL sense informació dinàmica que volem executar contra una BD

PreparedStatement

- Representa una sentència SQL amb informació dinàmica i que ha estat compilada en el SGBD abans de la seva execució

Java Database Connectivity (JDBC)

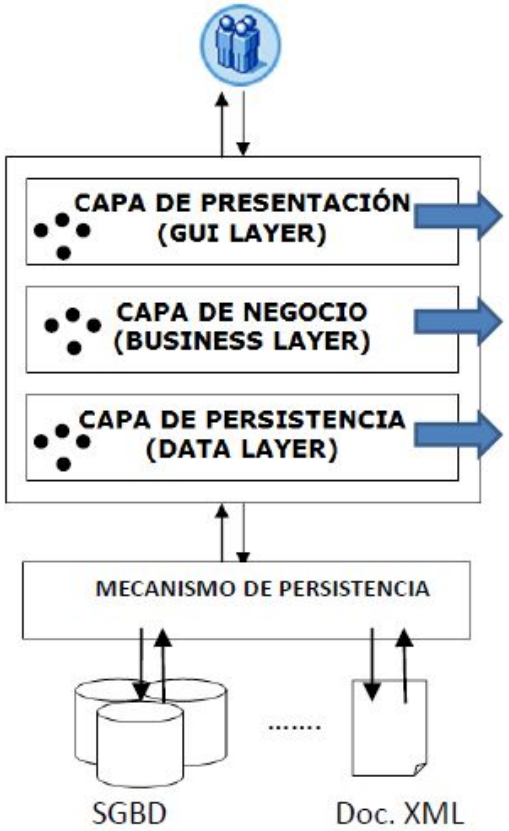
CallableStatement

- Representa una sentència d'invocació a un procediment emmagatzemat resident en una BD

ResultSet

- Representa a un iterador/cursor sobre el conjunt de resultats associat a l'execució d'una sentència SQL de consulta

Java Database Connectivity (JDBC)



Classes que representen les **vistes** d'una aplicació. Implementen la interfície gràfica a mostrar a l'usuari, així com els mecanismes per a interactuar amb ell

Classes que defineixen als objectes que representen el **domini** d'una aplicació: és a dir, els objectes que representen els conceptes/elements que volem representar en una aplicació.

Classes que implementen la **persistència** dels objectes de la nostra capa de negoci: com s'emmagatzemaran aquests objectes i com es recuperaran posteriorment.

The ORM MAPPING PROBLEM

Cal aconseguir que els objectes de la capa de negoci es facin persistents ("hibernin") en algun mitjà de persistència (SGBD, documents XML, ..) i poder-los recuperar més davant aquests objectes de la seva hibernació.

Per tant, caldrà "mapejar" les nostres entitats de negoci (objectes) a una estructura de dades relacional (Object-Relational Mapping), per tal de poder executar les sentències SQL corresponents.

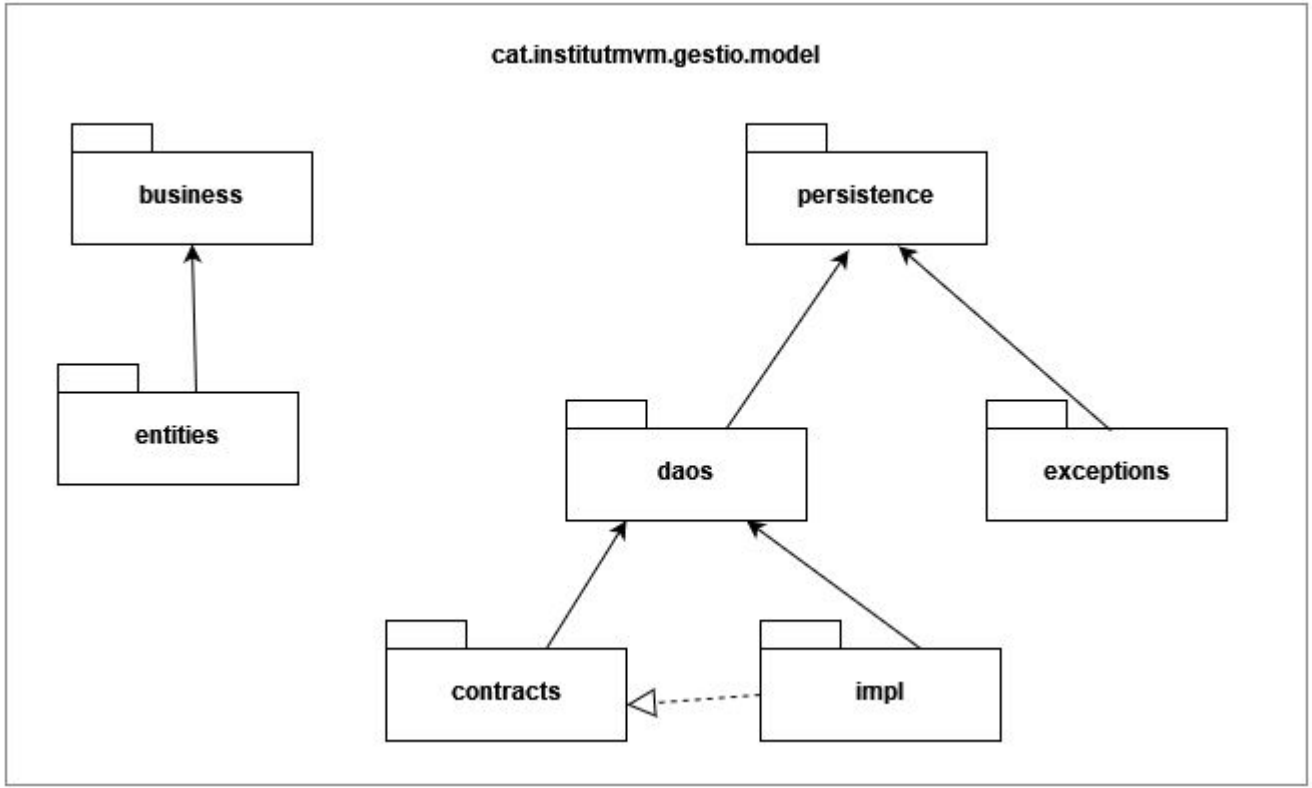
El següent pas serà implementar la capa de persistència (mitjançant DAO o delegant en un framework, com ara Hibernate).

Disseny de la capa de persistència (DAO)

"**DAO** proporciona una interfície única d'accés al repositori de persistència i independent de qual sigui aquest, permetent-ho canviar sense modificar la resta de l'aplicació".

1. Oferir una interface que defineixi les operacions de persistència o API de persistència per cada tipus de DAO que sigui independent del repositori
2. Crear un tipus d'excepció pròpia i independent del mecanisme de persistència que encapsuli totes les situacions errònies que es puguin produir en els DAOs
3. Oferir una implementació específica d'aquest API de persistència segons el repositori a utilitzar
4. Implementar una factoria o fàbrica de DAOs (patró de disseny factory)

Disseny de la capa de persistència (DAO). Diagrama UML



Disseny de la capa de persistència (DAO)

Passos a seguir

1. Implementar la classe que modela l'objecte
2. Implementar l'entitat en el SGBD que modela l'estat de l'objecte
3. Implementar l'interface que defineix totes les operacions de persistència necessàries
4. Implementar una classe de gestió d'excepcions específica per a DAOs
5. Implementar una classe que apliqui el patró de disseny factory