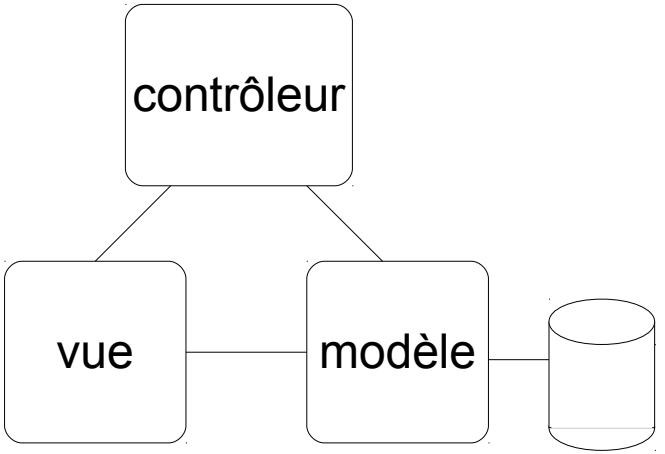
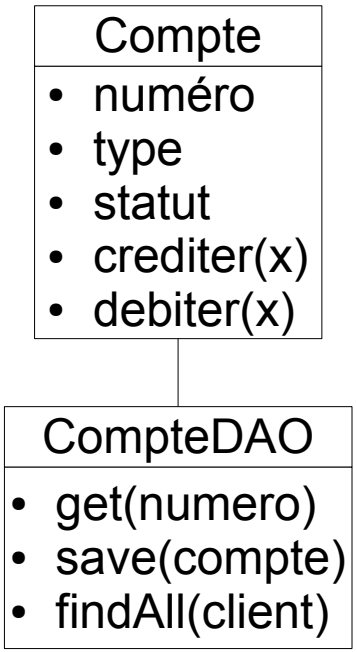
Persistance de données

- Dans l'approche MVC, les données de l'application sont décrites par le *modèle*.
- Le modèle assure également la persistance de ces données : au minimum, un stockage durable et cohérent est attendu
- Dans ce cours, nous détaillerons le mode de persistance le plus répandu, fondé sur l'usage de bases de données relationnelles (ce qui ajoute l'atomicité et l'isolation des transactions).



Persistance de données artisanale

- Certaines applications web implémentent leurs propres mécanismes de persistance dans une base de données.
- L'architecture générale consiste à doter chaque classe (type) du modèle d'une classe « compagnon » qui fournit les opérations de base (lecture, recherche, modification,...) dans la BD.



- Les désavantages sont nombreux :
  - nécessité d'écrire du SQL à la main
  - nécessité de prévoir toutes les méthodes utiles dans les classes compagnon
  - nécessité de gérer les exceptions SQL à la main
  - nombre de classes doublé

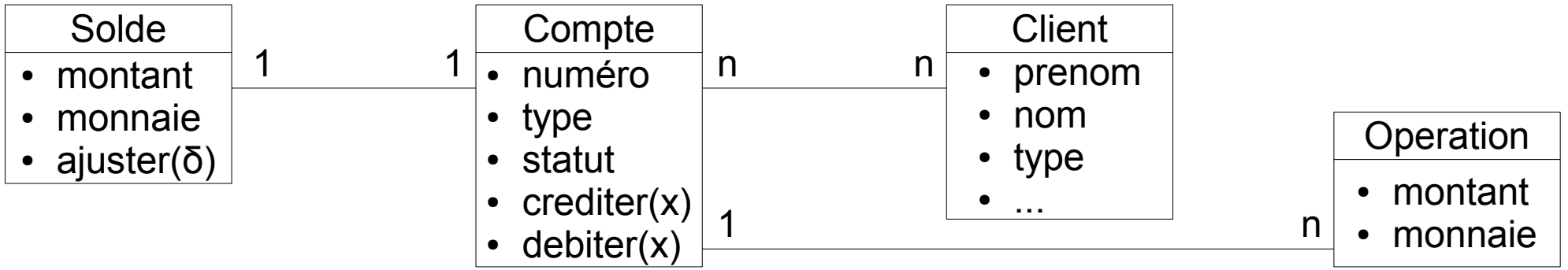
ORM

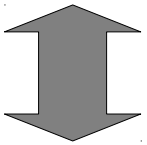
- Comme les données sont modélisées par des types (classes) et des relations entre elles (entités – relations), un mécanisme générique a été défini pour fournir une bijection entre le modèle et sa forme persistée dans une BD relationnelle.
- Ce mécanisme générique est « ORM » : object-relationship mapping.
- L'ORM se décline selon le langage qui décrit le modèle de données : le mécanisme précis n'est pas le même en Python ou en Java, mais les grandes lignes restent bien les mêmes.

ORM - Principes

- Une classe d'objets est représentée par une table dans la base de données.
- Dans cette table, les colonnes contiennent les différentes propriétés définies pour la classe.
- Un objet est représenté par une ligne de la table.
- Les relations entre les objets sont représentées
  - par des relations entre les lignes de tables (p.ex. des clés étrangères)
  - ou par des tables supplémentaires

ORM - Exemple





Solde

id	montant	monnaie	compte_id
1	100.00	EUR	10
2	200.00	EUR	11

Client

id	prenom	nom	type
21	Jean	Dupont	particulier
22	Pierre	Dupont	particulier

Compte

id	numéro	type	statut	solde_id
10	AB356	PEL	ouvert	1
11	ZX874	CHQ	ouvert	2

Compte_Client

compte_id	client_id
10	21
11	21
11	22

Operation

id	montant	monnaie	compte_id
31	+100.00	EUR	10
32	-15.00	EUR	11

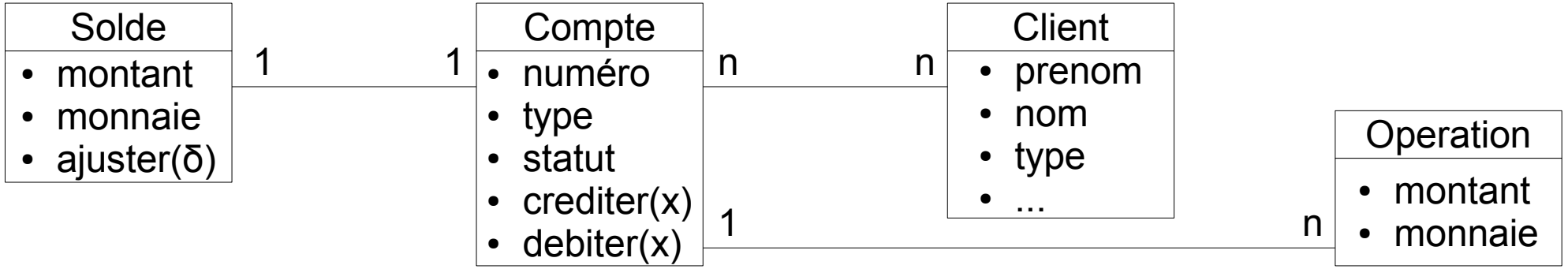
JPA

- En Java, le mécanisme ORM de référence est spécifié par le paquetage JPA (Java Persistence API) : `javax.persistence`
- Ce paquetage contient notamment des annotations qui permettent de *définir* dans le modèle les paramétrages nécessaires pour mettre en place une persistance en base de données relationnelle.
- La mise en place pratique de la persistance sera ensuite assurée par un composant tiers, p.ex. Hibernate et MySQL.

JPA - Bases

- Les classes à persister sont annotées par @Entity.
- On peut spécifier des paramètres supplémentaires de la table associée avec @Table.
- Les champs publiques de la classe sont associés aux colonnes de la table ; on peut gérer des paramétrages avec @Column.
  Pour exclure un champ de la persistance, on l'annote avec @Transient.
- Les relations entre objets sont annotées avec @OneToOne, @ManyToOne, @OneToMany, @ManyToMany pour indiquer l'arité des relations.
- Le champ contenant la clé primaire est marqué par @Id.

JPA - Exemple




```
@Entity
public class Solde {
  public double montant;
  public String monnaie;
  @OneToOne
  public Compte compte;
  ...
}
```

```
@Entity
public class Client {
  public String prenom;
  public String nom;
  public String type;
  @ManyToMany
  public Set<Compte> comptes;
  ...
}
```