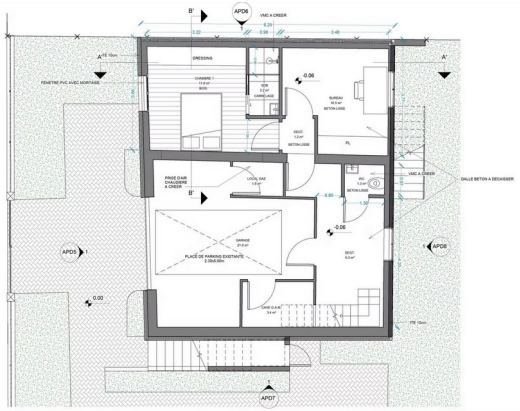
Modélisation en UML

ESIR2 SNR - GLA

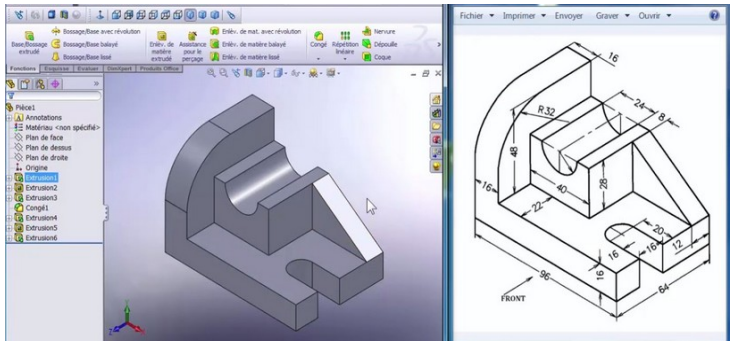
Qu-est ce qu'un modèle ?

- Modèle = abstraction d'un objet ou d'un système réel.
- C'est une représentation abstraite qui facilite l'étude, la conception, la manipulation, la simulation, et la compréhension de l'objet réel.



Plan de masse

représente



Modèle de pièce mécanique

représente

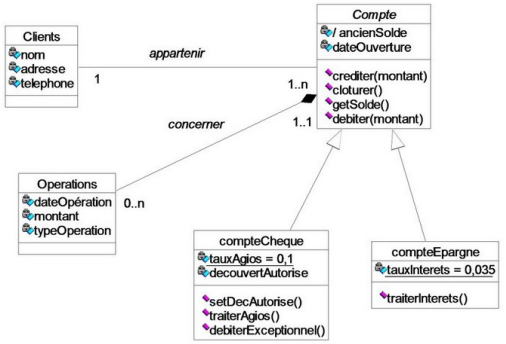


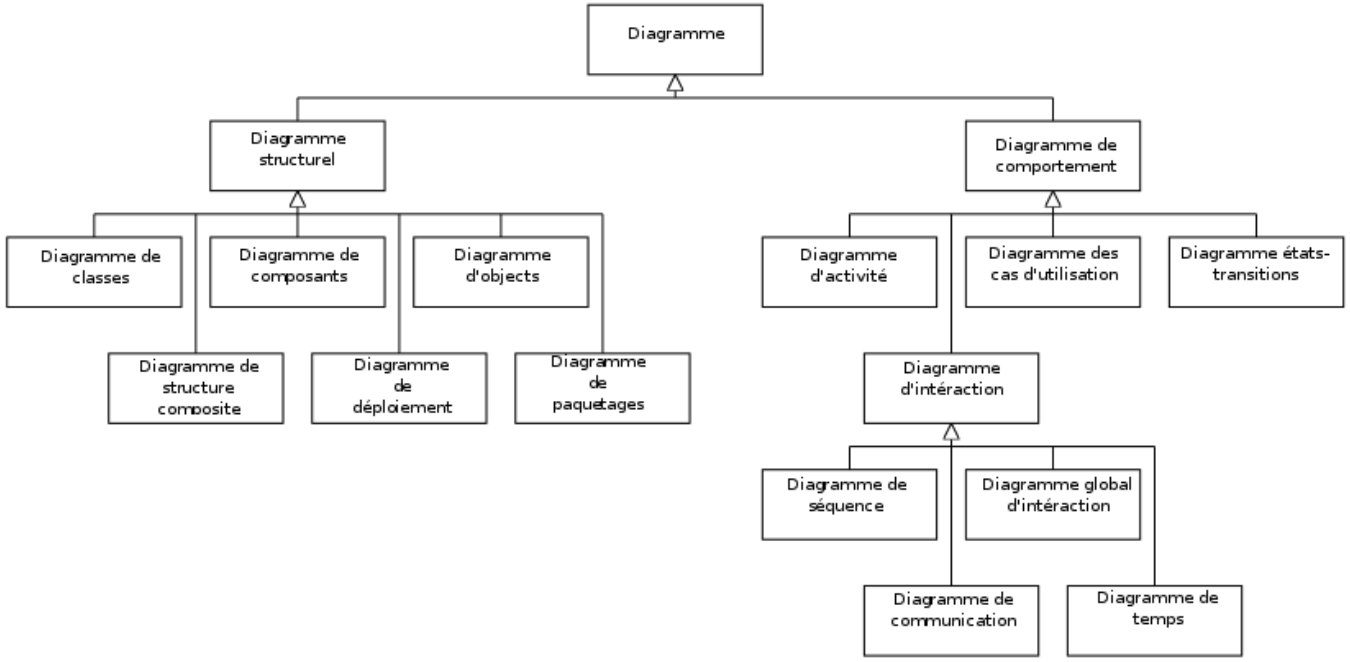
Diagramme de classes

représente

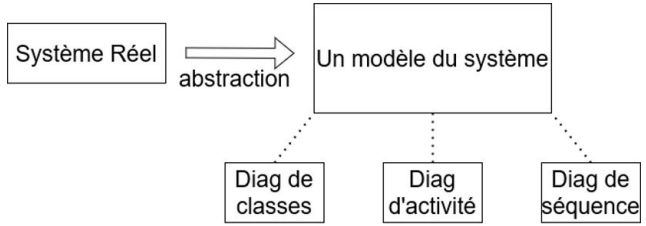
En informatique, le groupe OMG a défini UML !

- UML = Unified Modeling Language
- OMG = Object Management Group
- L'OMG est un groupe de standardisation qui promouvoit l'orienté objet, il a été fondé en 1989
- UML selon l'OMG : est un langage visuel dédié à la spécification, la conception et la documentation d'artefacts d'un système logiciel.
- UML comporte 14 diagrammes qui représentent l'application réelle en plusieurs niveaux de détail et avec différents niveaux d'abstraction. Ils sont regroupés en deux catégories :
  - Diagrammes structurel : permettent de représenter la **structure**
  - Diagrammes comportementaux : utilisé pour représenter le **comportement** interactif d'un système

Les 14 diagramm es UML



Diagrammes UML



Utilisation des différents diagrammes UML pour représenter un modèle

Conception orientée objet

Un objet : un élément défini par un état, un comportement, et une identité

L'état d'un objet : des valeurs qui décrivent l'objet

Le comportement d'un objet : des opérations que l'objet peut effectuer

L'identité d'un objet : un identifiant unique à chaque objet indépendamment de l'état

Les objets sont regroupés dans des **Classes**, et une classe regroupe des **instances** d'objets

Diagramme de classes

- Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.
- Il appartient aux diagrammes structurels puisqu'il décrit les différents concepts du système modélisé sous forme de **classes**, ainsi que les différents échanges entre ces concepts sous forme **d'associations**.

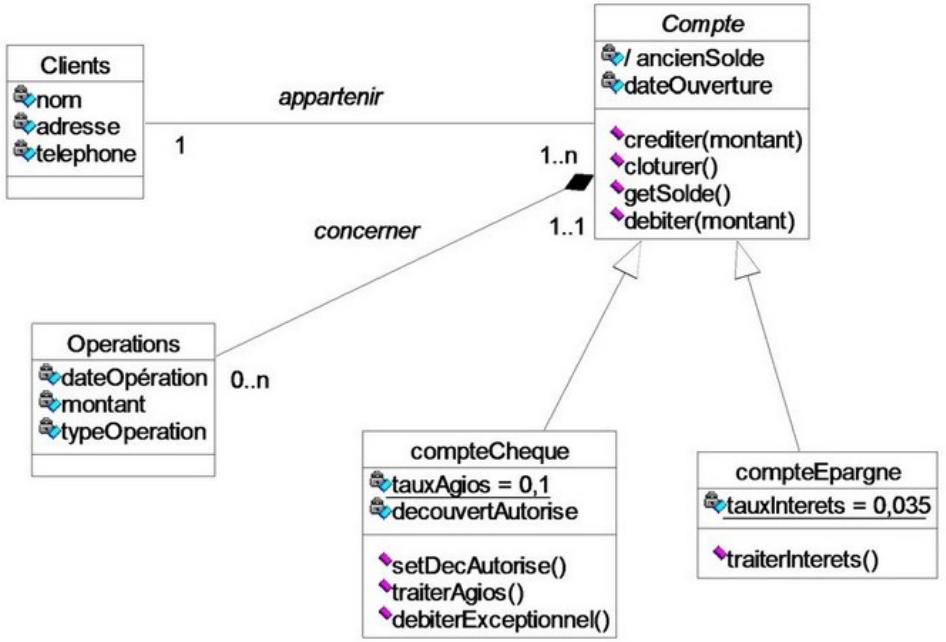
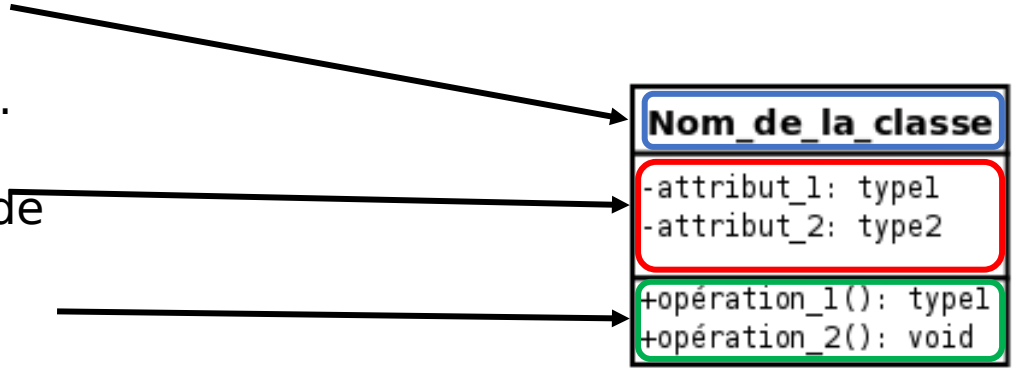


Diagramme de classes représentant une banque

Diagramme de classes - Les classes

- Une classe est graphiquement représentée par un rectange divisé en trois compartiments :
  1. Le premier compartiment est réservé au **nom de la classe**. Par convention, celui ci doit obligatoirement être en majuscule.
  2. Le deuxième compartiment est réservé aux **attributs** des objets de la classe.
  3. Le troisième compartiment est réservé aux **opérations**
- Il est tout à fait possible d'avoir une classe sans attributs, ni opérations, dans ce cas, les compartiments de ces concepts ne sont pas dessinés



Les classes

- Une classe est un regroupement d'objets similaires.
- La classe **Humain** dans cet exemple regroupe tous les objets humains, qu'ils soient hommes ou femmes.
- Adrien, Marie, Kylian, Arthur, Lucie, Clara sont des instances de la classe Humain.

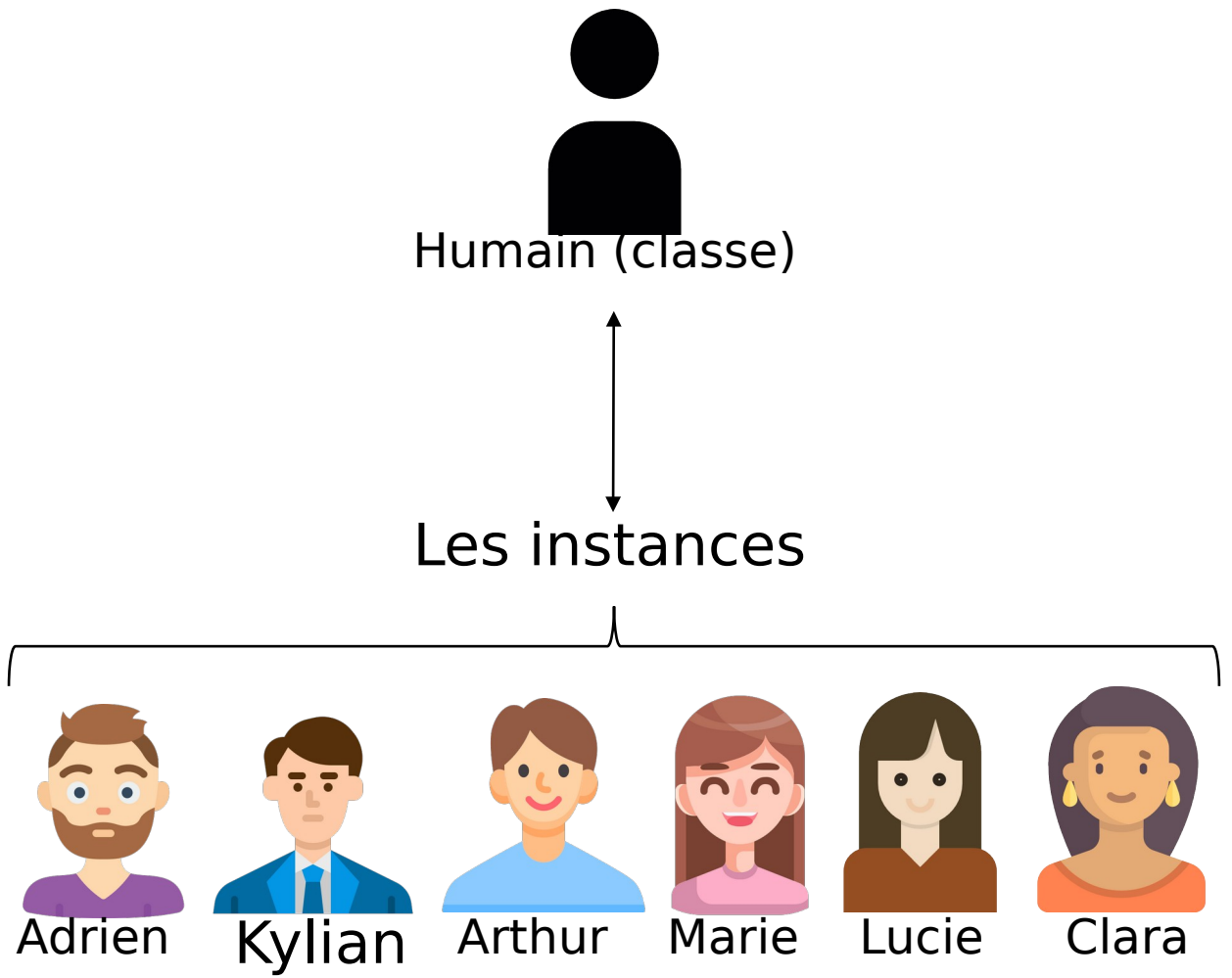
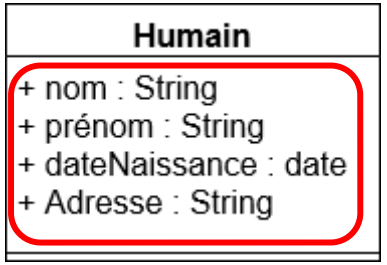


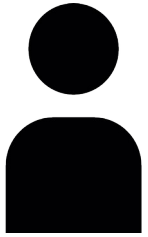
Diagramme de classes – les attributs

- Un attribut désigne une information qui définit la classe. Une classe peut contenir un ou plusieurs attributs. **Un attribut** a un type particulier.
- La classe humain par exemple pourrait contenir les attributs :
  - Nom, prénom, date de naissance, lieu de naissance, sexe, adresse, situation familiale, etc.
- Au niveau des instances, des valeurs sont données à ces attributs.



Les attributs

- Au niveau des instances, des valeurs sont données à ces attributs.
- Par exemple, Adrien et Clara qui sont des instances de la classe Humain ont des valeurs d'attributs différentes.



Humain (classe)

Les instances

Prénom : Adrien

Nom : Martin

Date de naissance : 23/05/1998

Adresse : Rennes

Prénom : Clara

Nom : Robert

Date de naissance : 01/01/1997

Adresse : Rennes

Diagramme de classes – les opérations

- **Une opération** désigne les activités que l'élément abstrait modélisé par la classe pourrait faire. Celle-ci peut prendre des paramètres en entrée, et peut avoir un type de retour.
- Dans notre exemple; la classe Humain contient deux opérations : manger() et marcher(). D'autres opérations peuvent être ajoutées comme dormir().

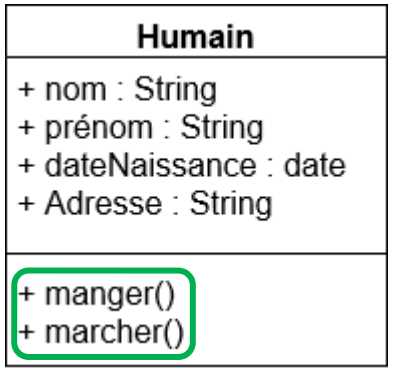


Diagramme de classes – les associations

- **Une association** est une relation entre deux classes (association binaire) ou plus (association n-aire), qui décrit les connexions structurelles entre leurs instances. Une association indique donc qu'il peut y avoir des liens entre des instances des classes associées.
- Une association a une cardinalité qui désigne le nombre d'objets pouvant être liés à travers l'association. Elle est exprimée entre crochets []
- Il est possible d'avoir une association entre une classe et elle-même

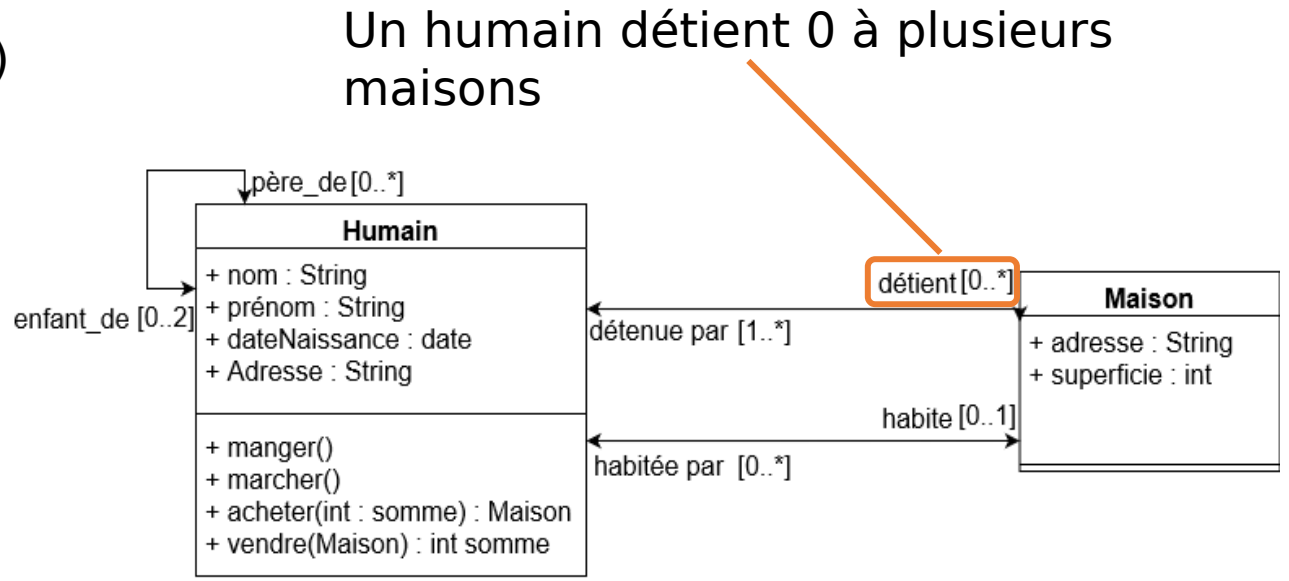
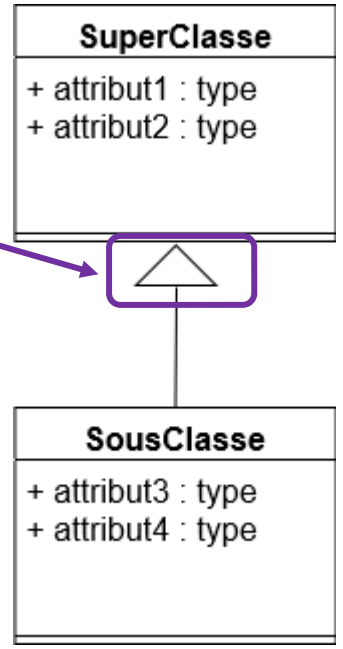


Diagramme de classes - L'héritage et généralisation

- **L'héritage ou la généralisation** désigne un lien hierarchique entre deux classes (une classe mère ou super classe, et une classe fille ou sous classe).
- Ainsi, la classe fille hérite tous les éleménts qui ont été défini dans les classes mères.
- La classe mère devient alors une généralisation de la classe fille, et la classe fille une spécialisation de la classe mère
- Le lien d'héritage est symbolisé par un triangle creux sortant de la super classe vers toutes les sous-classes
- Si une sous classe S1 hérite d'une super classe S2 qui est elle même une sous classe, la sous classe S1 hérite tous les éléments de toutes ses super classes (S2 et ses superclasses)

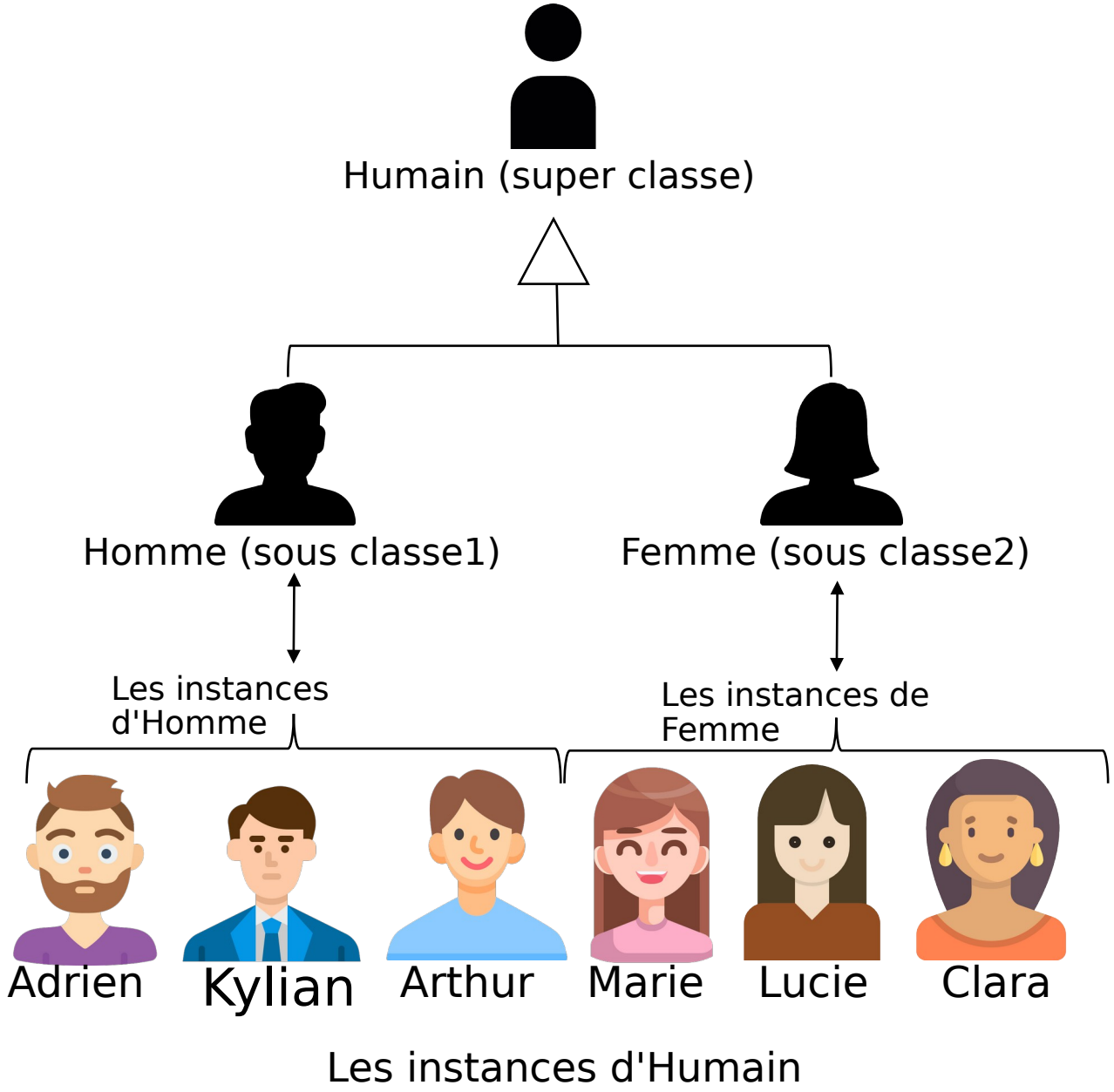


"SuperClasse" est une généralisation de la classe "SousClasse", et la classe "SousClasse" est une spécialisation de la classe "SuperClasse"

SousClasse hérite les attributs "attribut1" et "attribut2"

Les classes

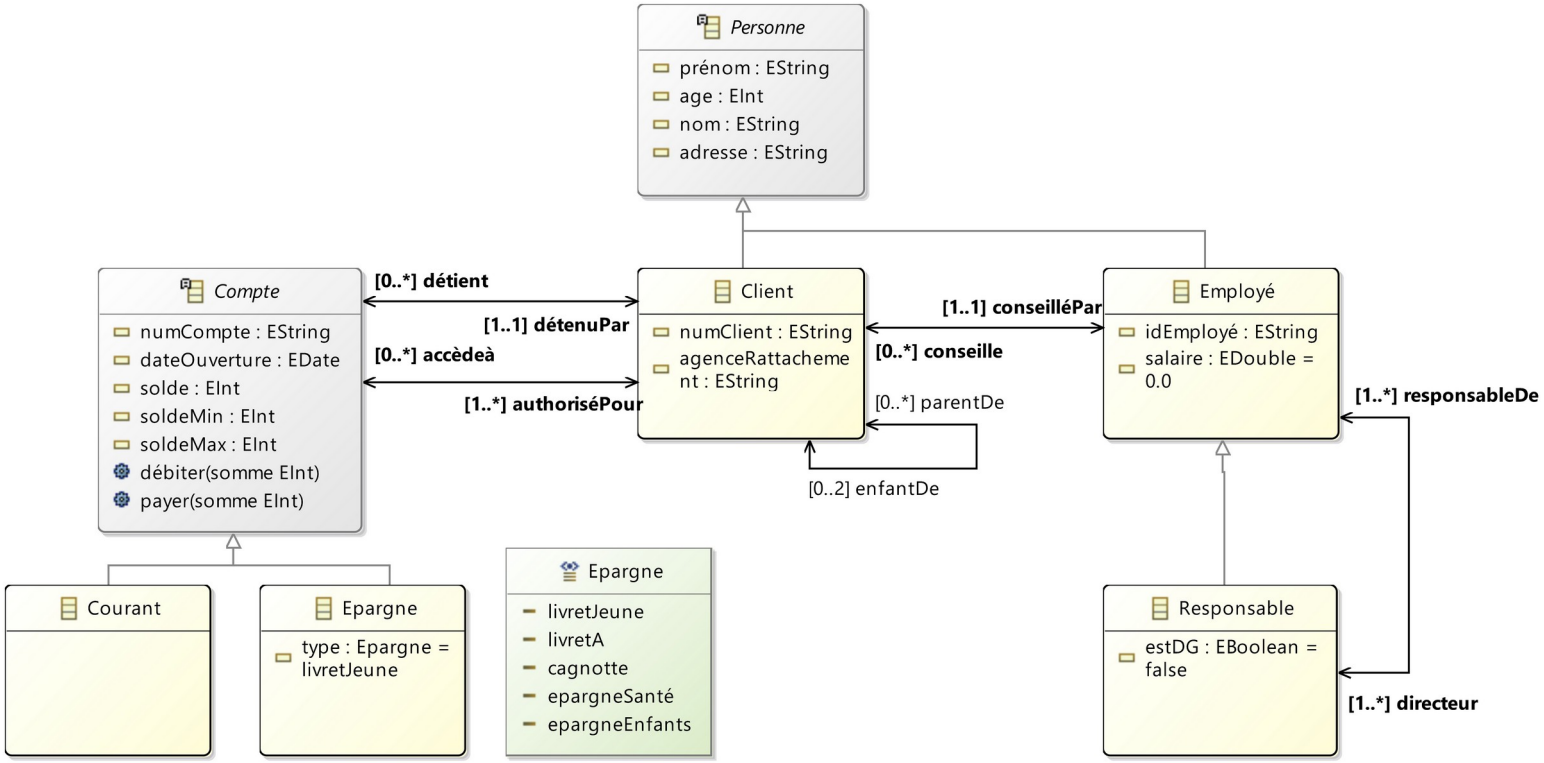
- Il est possible de définir des sous-classes d'une classe en particulier
- Dans notre cas, la classe Humain est la super classe. Les deux classes Homme et Femmes sont des sous classes. Une instance de la classe Homme ou Femme est automatiquement une instance de la classe Humain.



Exemple illustratif – Diagramme de classe d'une banque

- Une banque qui gère les clients et des employés
- Les clients peuvent détenir plusieurs types de comptes (courant, épargne, livret jeune, ...), ils peuvent avoir accès à certains comptes même s'ils leurs appartiennent pas. Ils sont rattachés à une agence à la fois.
- Plusieurs opérations sont nécessaires à la gestion du compte
- Les employés ont des responsables
- Les employés travaillent dans des bureaux (individuels ou collectifs). Seuls les responsables ont droit à des bureaux individuels

Exemple illustratif – Diagramme de classe d'une banque



A vous de jouer ! - Exercice 01

- Pour chaque exemple ci-dessous, indiquez si la relation présentée est une généralisation, une
  agrégation ou une association :
  1. Un pays a une capitale
  2. Une transaction boursière est un achat ou une vente
  3. Les fichiers contiennent des enregistrements
  4. Une personne utilise un langage de programmation dans un projet
  5. Les modems et les claviers sont des périphériques d’entrées/sorties

A vous de jouer ! - Exercice 02

- Pour chaque situation ci-dessous, proposez une modélisation de la réalité.
  1. Une librairie vend des livres, caractérisés par leur auteur et leur nombre de pages ; certains
  livres possèdent également d’autres caractéristiques : une fourchette des âges pour les livres
  pour enfants, et la discipline et le niveau pour les livres scolaires.
  2. On considère une entreprise, et on suppose qu’un chef dirige plusieurs salariés (les
  subordonnés) et que le chef est lui-même un salarié.
  3. On considère une université, et les personnes y travaillant qui peuvent être des étudiants ou des enseignants.

A vous de jouer ! - Exercice 03

- Une classe Véhicule a été caractérisée par les propriétés suivantes : Numéro du véhicule, date de fabrication du véhicule, pavillon du bateau, nombre de réacteurs, superficie des ailes, puissance fiscale, hauteur du mat, nombre de torpilles.
  Quel est le défaut de cette classe ? Proposez une autre représentation à l'aide d'un diagramme de
  classes.

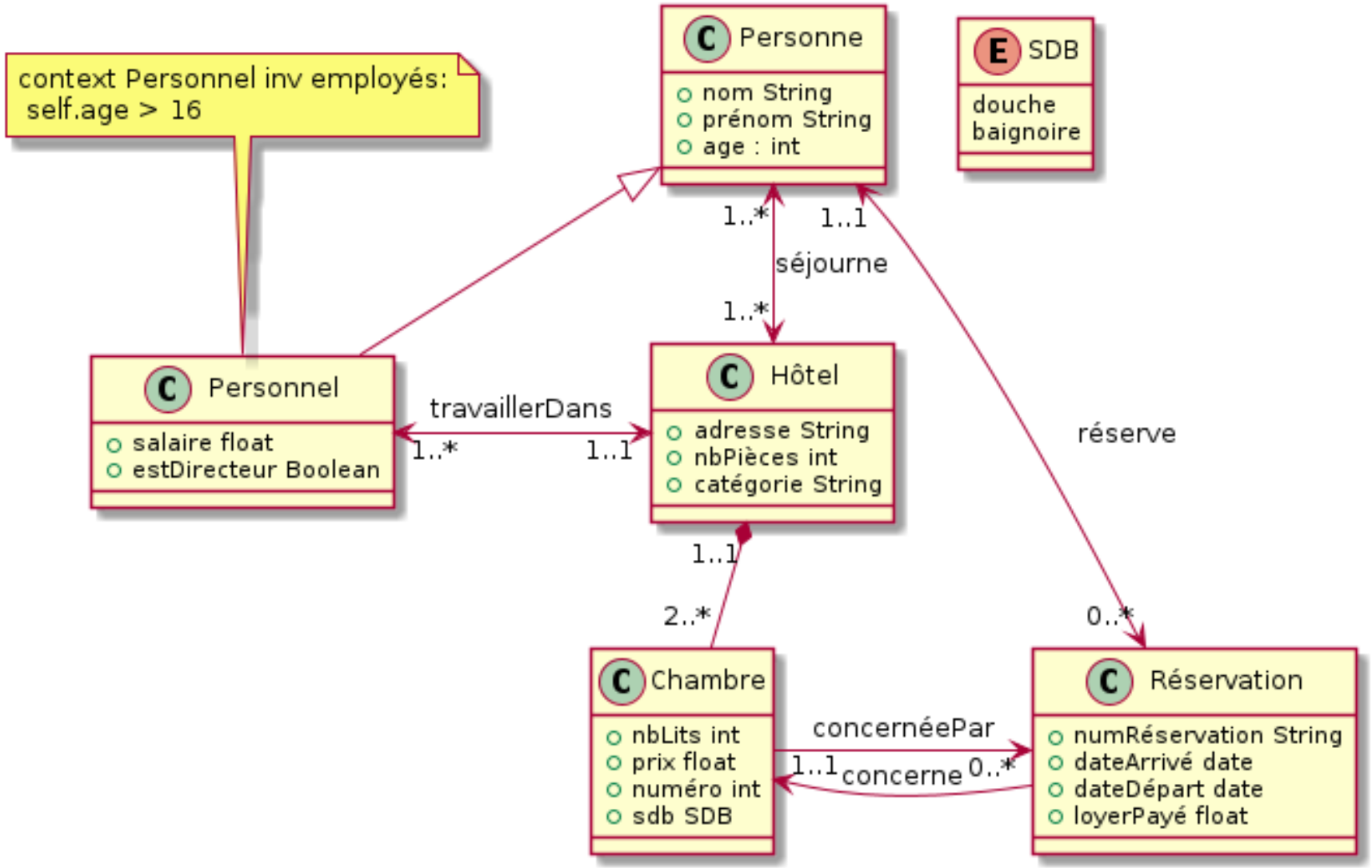
A vous de jouer ! - Exercice 03

- Une classe Véhicule a été caractérisée par les propriétés suivantes : Numéro du véhicule, date de fabrication du véhicule, pavillon du bateau, nombre de réacteurs, superficie des ailes, puissance fiscale, hauteur du mat, nombre de torpilles.
  Quel est le défaut de cette classe ? Proposez une autre représentation à l'aide d'un diagramme de
  classes.

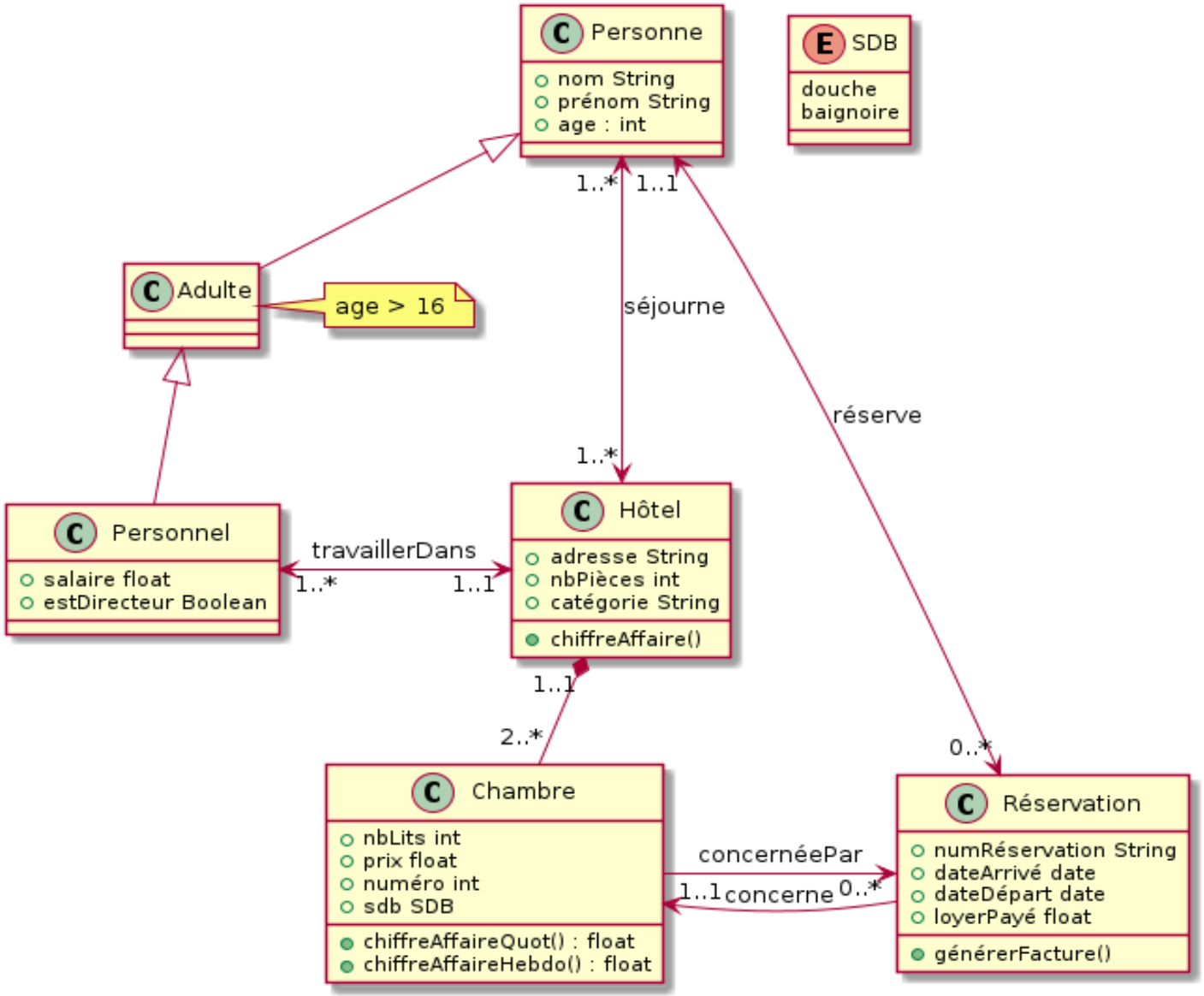
A vous de jouer ! - Exercice 04

- Un hôtel est composé d'au moins deux chambres. Chaque chambre dispose d'une salle d'eau
  : douche ou bien baignoire. Un hôtel héberge des personnes. Il peut employer du personnel et il est impérativement dirigé par un directeur. On ne connaît que le nom et le prénom des employés, des directeurs et des occupants. Certaines personnes sont des enfants et d'autres des adultes (faire travailler des enfants est interdit). Un hôtel a les caractéristiques suivantes : une adresse, un nombre de pièces et une catégorie.
- Une chambre est caractérisée par le nombre de lits qu'elle contient, son prix et son numéro. On veut pouvoir savoir qui occupe quelle chambre à quelle date. Pour chaque jour de l'année, on veut pouvoir calculer le loyer de chaque chambre en fonction de son prix et de son occupation (le loyer est nul si la chambre est inoccupée). La somme de ces loyers permet de calculer le chiffre d'affaires de l'hôtel entre deux dates.

Exercice 04 – correction – Avec contraintes



Exercice 04 – correction – Sans contraintes



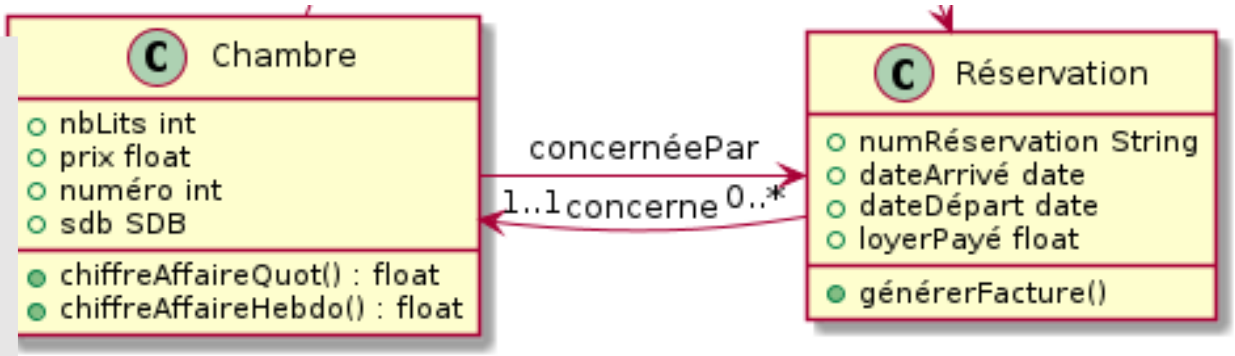
Passage du modèle vers le code

Diagramme de classes vers Java

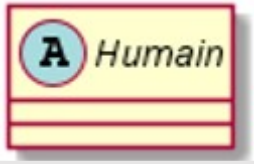
UML	Java
Classe	class
Interface	interface
Héritage	extends
Implémentation interface	implements
Attribut	.
Opération	.
Classe/Opération abstraite	abstract
Redéfinition	.
Surcharge	.
Association, Composition, Aggrégation	.
Enumération	.
Contrainte	.

1 - Création des classes avec le constructeur

```
Public Class
Chambre{

// Le  constructeur
Public Chambre()
{

}
```




```
Public Class
Réservation{
// Le  constructeur
Public Réservation()
{

   }
}
```




```
interface Humain{
// Les interfaces ne doivent pas avoir de
constructeurs
}
```

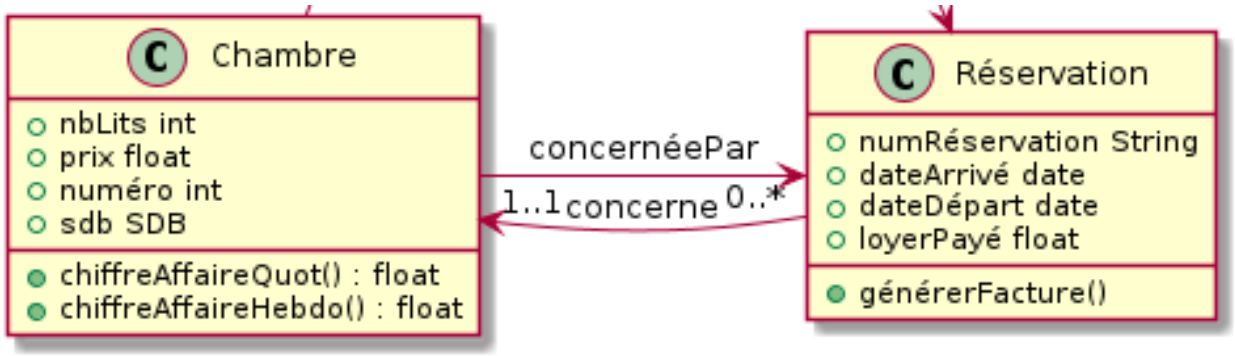
// Une classe implémente une interface comme suit :

```
interface Personne implements Humain{

Personne(){
// Constructeur
}
}
```

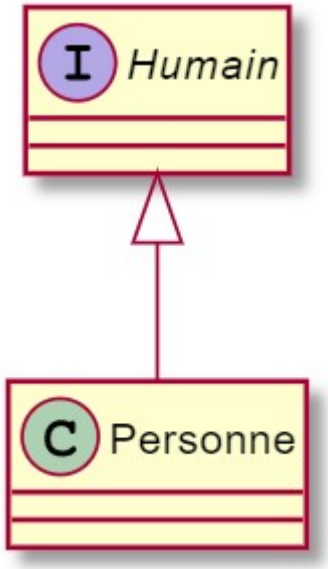
```
Abstract Class Humain{

// Les interfaces ne doivent pas avoir de
constructeurs

}
```

1 - Création des classes avec le constructeur

```
Public Class
Chambre{

// Le  constructeur
Public Chambre()
{

}
```

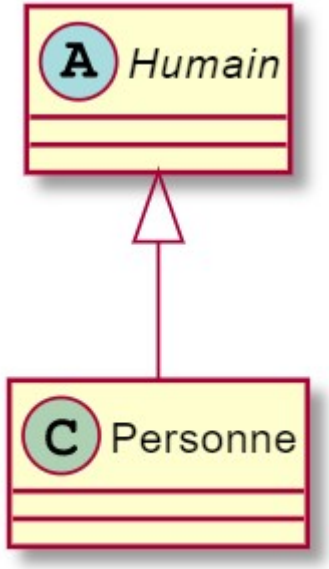



```
Public Class
Réservation{
// Le  constructeur
Public Réservation()
{

    }
}
```




```
interface Humain{
// Les interfaces ne doivent
pas avoir de constructeurs
}
```

```
// Une classe implémente une
interface comme suit :

interface Personne implements
Humain{

Personne(){
// Constructeur
}
}
```

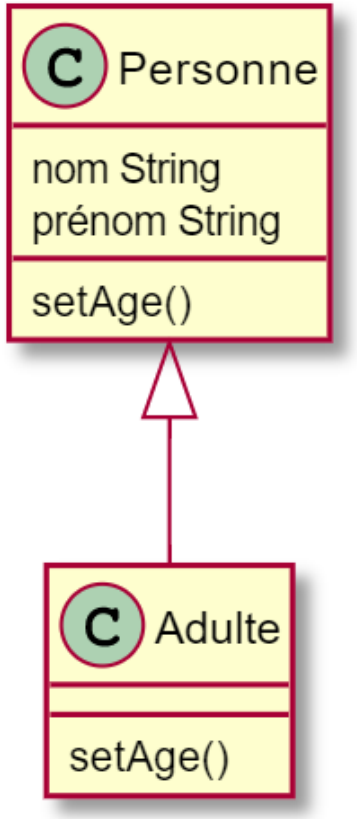



```
Abstract Class
Humain{

// Les interfaces ne
doivent pas avoir de
constructeurs

}
```

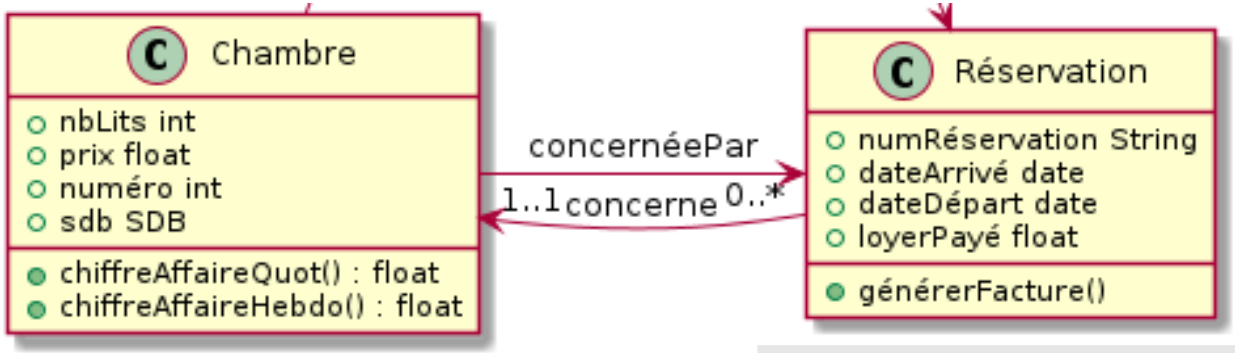
2 - Héritage et redéfinition, et contraintes

```
Public Class Personne{
String nom, prénom;
// Le  constructeur
Public Personne(String nom, String
prénom){
    this.nom = nom;
}
Void setAge(int Age){
    this.age = age;
}
}
```

```
Public Class Adulte extends
Personne{
// Le  constructeur
Public Adulte(){
    Super();
    }
// Méthode redéfinie dans la classe
fille
Void setAge(int age){
 If (age >= 18) {this.age = age}
Else { System.out.println("Age
incorrect");}
}
```



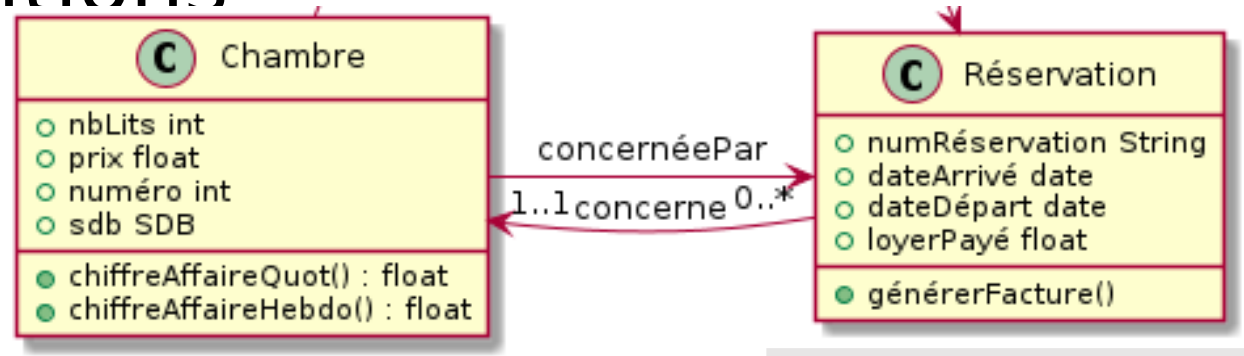
3 – Attributs et Opérations




```
Public Class Chambre{
// Les attributs
Int nbLits,numéro;
Float prix;
// Le  constructeur
Public Chambre(int nbLits, float prix, int numéro){
    this.nbLits = nbLits;
    this.numéro = numéro;
    this.prix = prix; }
// Les opérations
Public float chiffreAffaireQuotidien(){ return (int)
0;}
Public float ChiffreAffaireHebdo() { return (float)
0;}
}
```

```
Public Class Réservation{
// Les attributs
    private String numRéservation;
    Date dateArrivé;
    Date dateDépart;
    float loyerPayé;
// Le  constructeur
Public Réservation(String numRéservation, Date
dateArrivée, Date dateDépart float loyerPayé){
    this.nbLits = nbLits;
    this.numéro = numéro;
    this.prix = prix;
    this.loyerPayé = loyerPayé}

// Les opérations
Public void générerFacture(){ // corps de
l'opération}
}
```

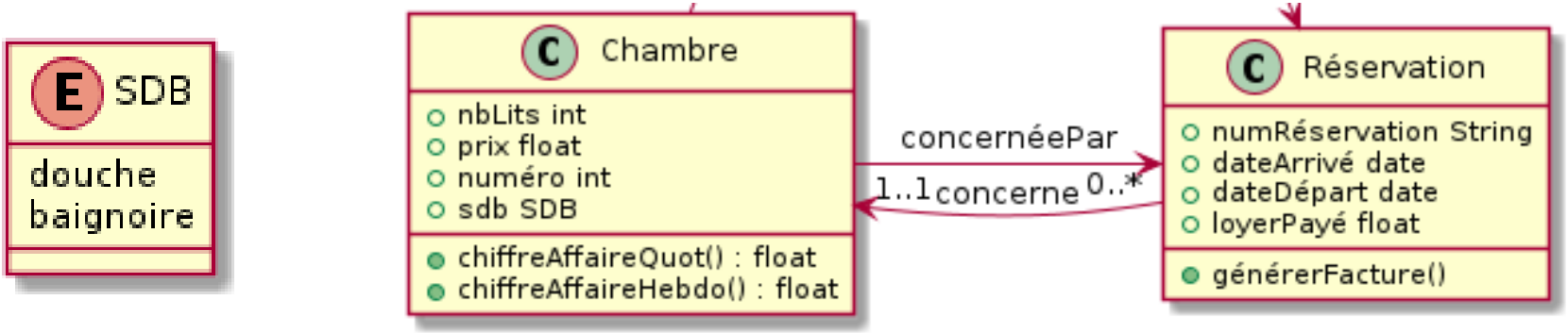
4 – Les associations, aggrégations, et compositions




```
Public Class Chambre{
// Les associations
Set<Réservation> réservations;

// Le  constructeur
Public Chambre( Réservation réservations){
    This.réservations = réservations;}

}
```

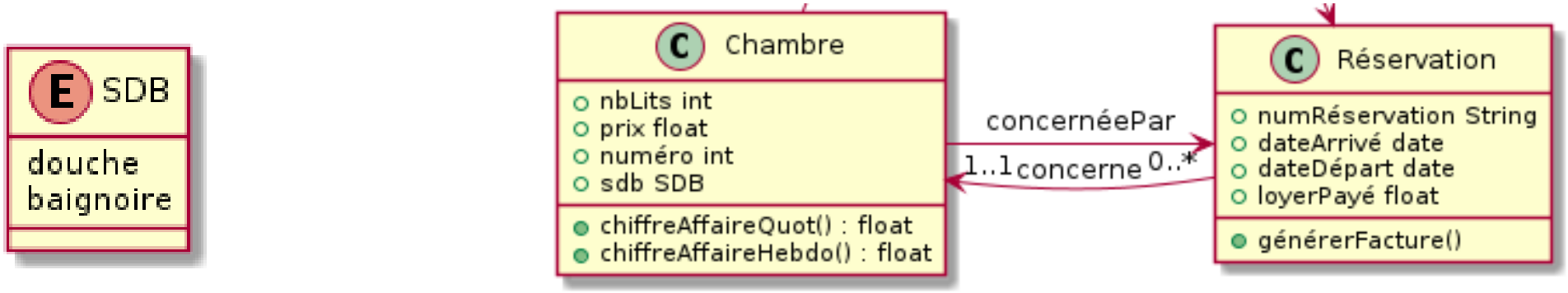
```
Public Class Réservation{
// Les associations
    Chambre chambreReservée;
// Le  constructeur
Public Réservation(Chambre chambreReservée){
    This.chambreRéservée = chambreRéservée;}

}
```

5 - Enumération




```
Public Class Chambre{
// L'énumération
public enum SDB{douche, baignoire};
// Les attributs
Sdb SDB;
Set<Réservation> réservations;
}
```

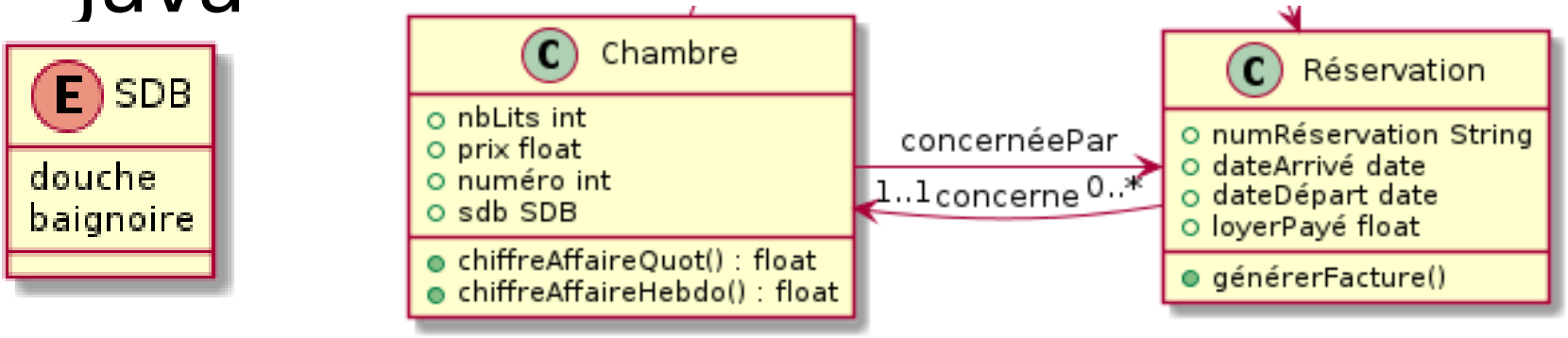
6 - Surcharge




```
Public Class Chambre{

// Les opérations
Public float chiffreAffaireQuotidien(){
// Retourne le chiffre d'affaire de la journée courante
}

Public float chiffreAffaireQuotidien(Date date){
// Retourne le chiffre d'affaire de la journée prise en
entrée
}
}
```

La surcharge consiste à créer une opération identique à celle déjà existante avec des paramètres en entrée qui sont différents

Passage d'un diagramme de classes vers code Java



Public Class Chambre{
public enum SDB{douche, baignoire}; //
L'énumération
// Les attributs
Int nbLits,numéro;
Float prix;
Sdb SDB;
Set<Réservation> réservations;
// Le constructeur
Public Chambre(int nbLits, float prix, int numéro, SDB sdb, Réservation réservations){
this.nbLits = nbLits;
this.numéro = numéro;
this.prix = prix;
this.sdb = sdb;
This.réservations = réservations;} //
// Les opérations
Public float chiffreAffaireQuotidien(){ return (int) 0;}
Public float ChiffreAffaireHebdo(){ return (float)

Public Class Réservation{
// Les attributs private String numRéservation;
Date dateArrivé;
Date dateDépart;
float loyerPayé;
Chambre chambreReservée;
// Le constructeur
Public Réservation(String numRéservation, Date dateArrivée, Date dateDépart float loyerPayé){
this.nbLits = nbLits;
this.numéro = numéro;
this.prix = prix;
this.sdb = sdb;
This.réservations = réservations;} //

// Les opérations
Public void générerFacture(){ // corps de l'opération}
}

Représentation d'une classe en python

```
class MaClasse:
    attribute = "un attribut de classe"
    def __init__(self,attr1, attr2):     // le constructeur de la classe "MaClasse"
        self.attr1 = attr1
        self.attr2 = attr2  //  attr1 et attr2 sont des attributs d'instance
Class MaSousClasse(MaClasse):
    Def __init__(self, attr3):
        Self.attr3 = attr3
        Self.attr3 = Super().attr1
    Def opération(param1, param2):
        Print(param1, " et ", param2)
//création d'instances
nomInstance = MaClasse(attr1, attr2)
```

Représentation d'une classe en python

```
class MaClasse:
    attribute = "un attribut de classe"
    def __init__(self,attr1, attr2):     // le constructeur de la classe "MaClasse"
        self.attr1 = attr1
        self.attr2 = attr2  //  attr1 et attr2 sont des attributs d'instance
Class MaSousClasse(MaClasse):
    Def __init__(self, attr3):
        Self.attr3 = attr3
        Self.attr3 = Super().attr1
    Def opération(param1, param2):
        Print(param1, " et ", param2)
//création d'instances
nomInstance = MaClasse(attr1, attr2)
```

Exercice

https://github.com/Cherfalyes/GLA-2022/

Lien utile :

https://realpython.com/python3-object-oriented-programming/

References

- http://david.roumanet.free.fr/BTS-SIO/SLAM5/UML%20diagramme%20de%20classe.pdf
- https://mrproof.blogspot.com/2012/10/exercice-corrige-uml-diagramme-de.html
- https://www.omg.org/spec/OCL/2.4/PDF
- https://realpython.com/python3-object-oriented-programming/
- http://niedercorn.free.fr/iris/iris1/uml/umltd5.pdf
- https://latavernedutesteur.fr/2017/11/03/les-niveaux-de-test/
- https://www.fil.univ-lille1.fr/~wegrzyno/portail/Info/Doc/HTML/tp_conditionnelle.html
- https://code.tutsplus.com/tutorials/beginning-test-driven