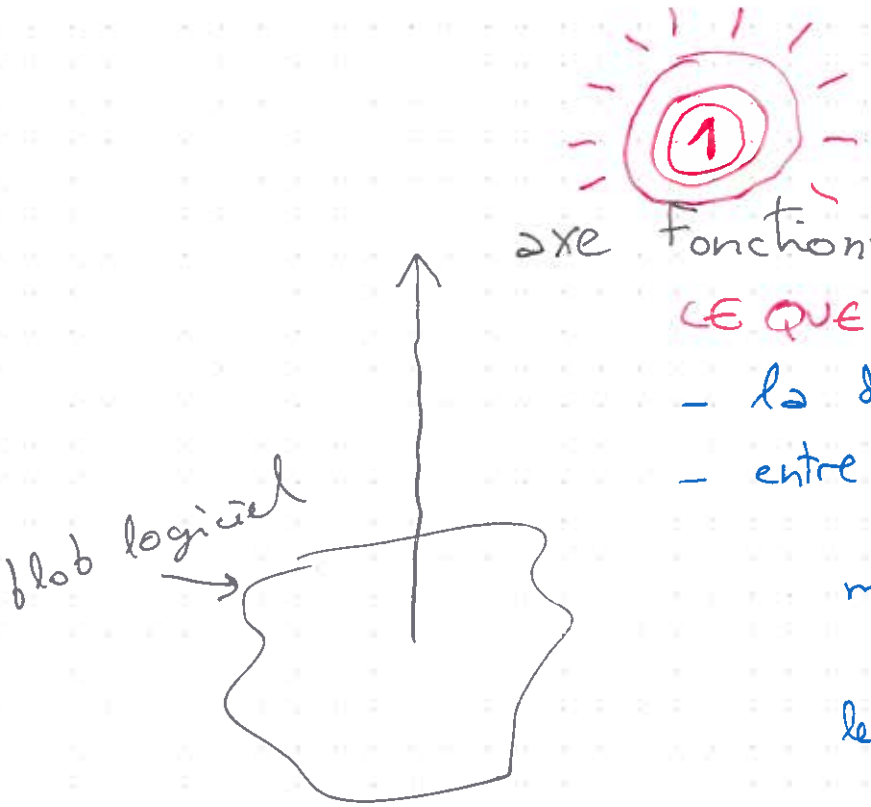
Cours de modélisation objet 2

MOD2 — CM#2

T. Gérard

2024



CE QUE FAIT LE LOGICIEL

- la description la plus importante
- entre MOA et MOE

↓

maîtrise d'ouvrage
=
le logiciel
=
le propre du logiciel

maîtrise d'œuvre
=
le logiciel
=
les gens qui mettent en œuvre / qui code

diagrammes de cas d'utilisation

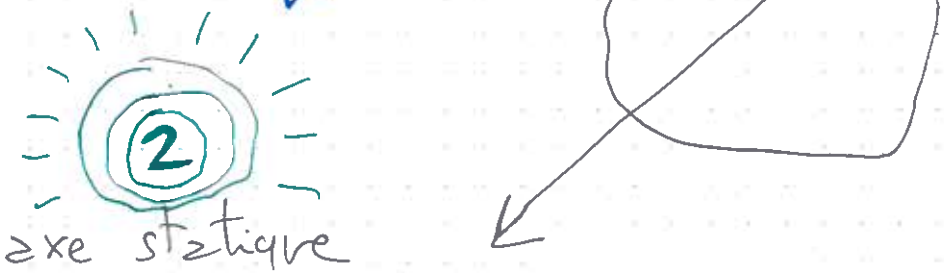
+ pour chaque cas :
  - sa description textuel
  - son scénario (dynamique)

on traduit les Fonctionnalités en description du "code"

1

axe Fonctionnel

CE QUE FAIT LE LOGICIEL



axe statique

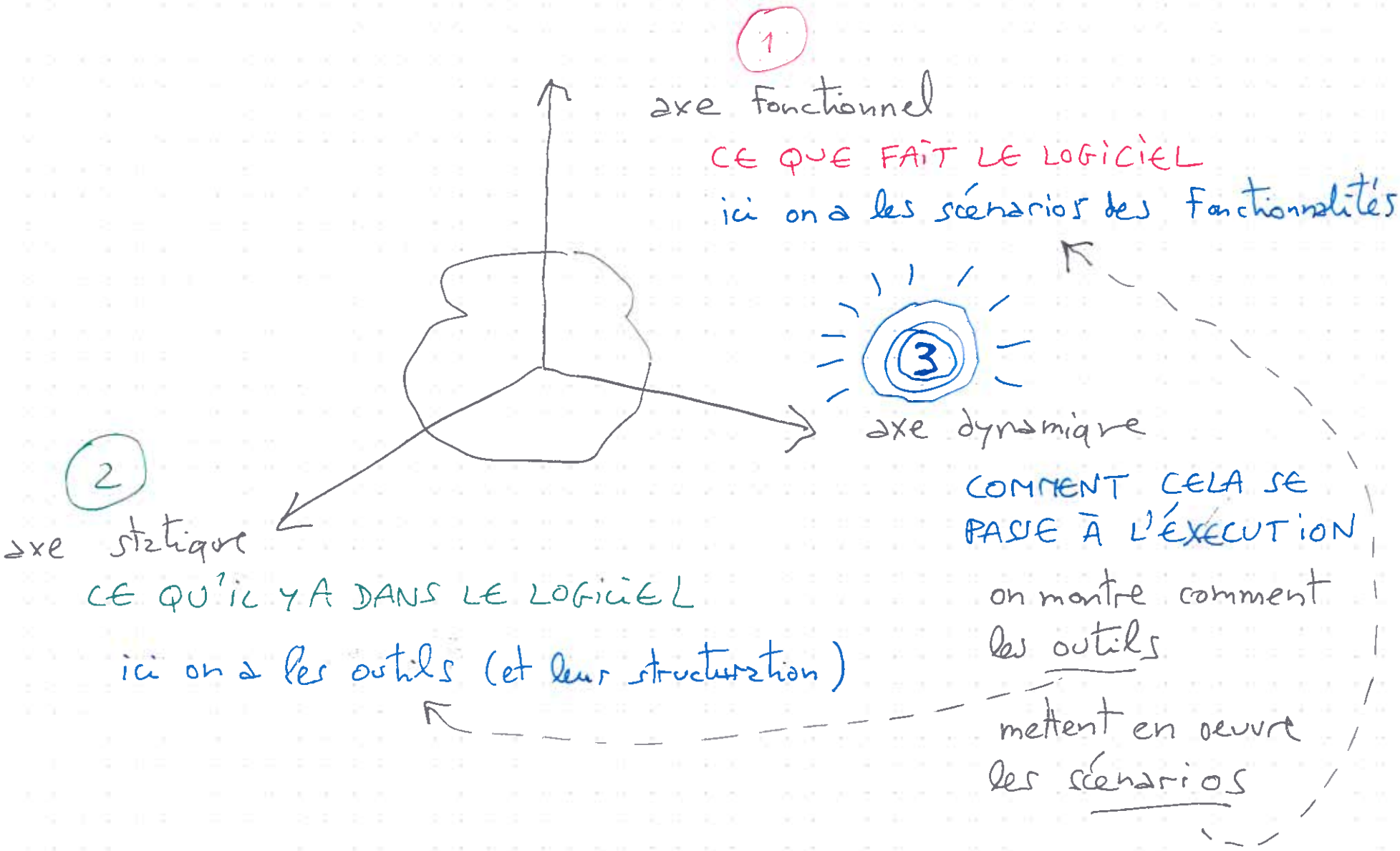
CE QU'IL Y A DANS LE LOGICIEL

- des paquetages
- des composants
- des classes

} et leurs relations

=> diagrammes de classes (pour le partiel)

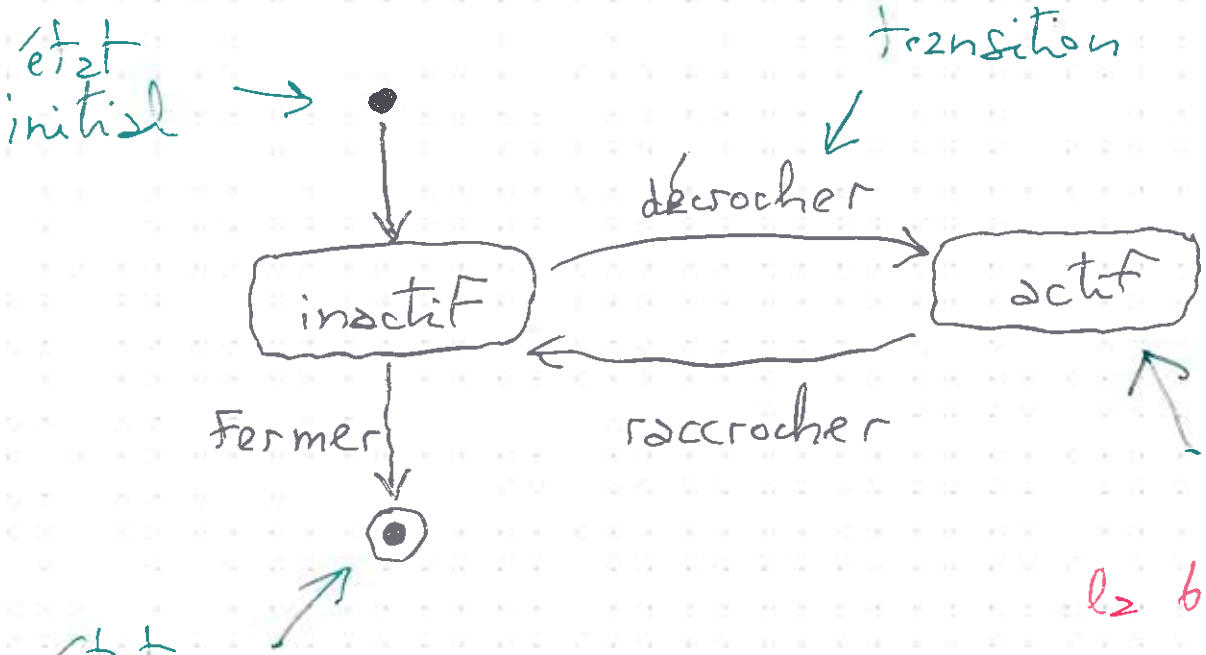
la modélisation statique doit pouvoir supporter ("implémenter") les Fonctionnalités attendues


1
axe Fonctionnel
CE QUE FAIT LE LOGICIEL
ici on a les scénarios des Fonctionnalités
3
axe dynamique
COMMENT CELA SE PASSE À L'EXÉCUTION
on montre comment les outils
mettent en oeuvre les scénarios
2
axe statique
CE QU'IL Y A DANS LE LOGICIEL
ici on a les outils (et leur structuration)

Modélisation dynamique :

- diagramme états-transitions (pas dans le partiel)
- diagramme de séquence (demandé au partiel)

DIAGRAMMES ÉTATS-TRANSITIONS



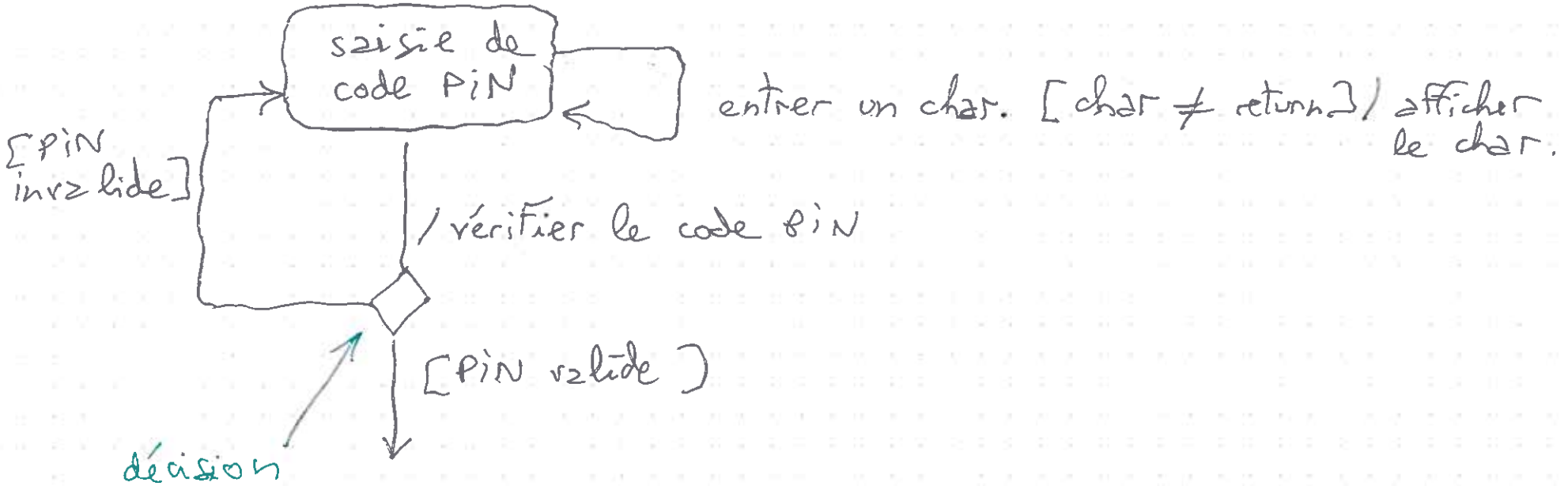
Sur une transition :

événement [garde] / activité

ce qui déclenche la transition

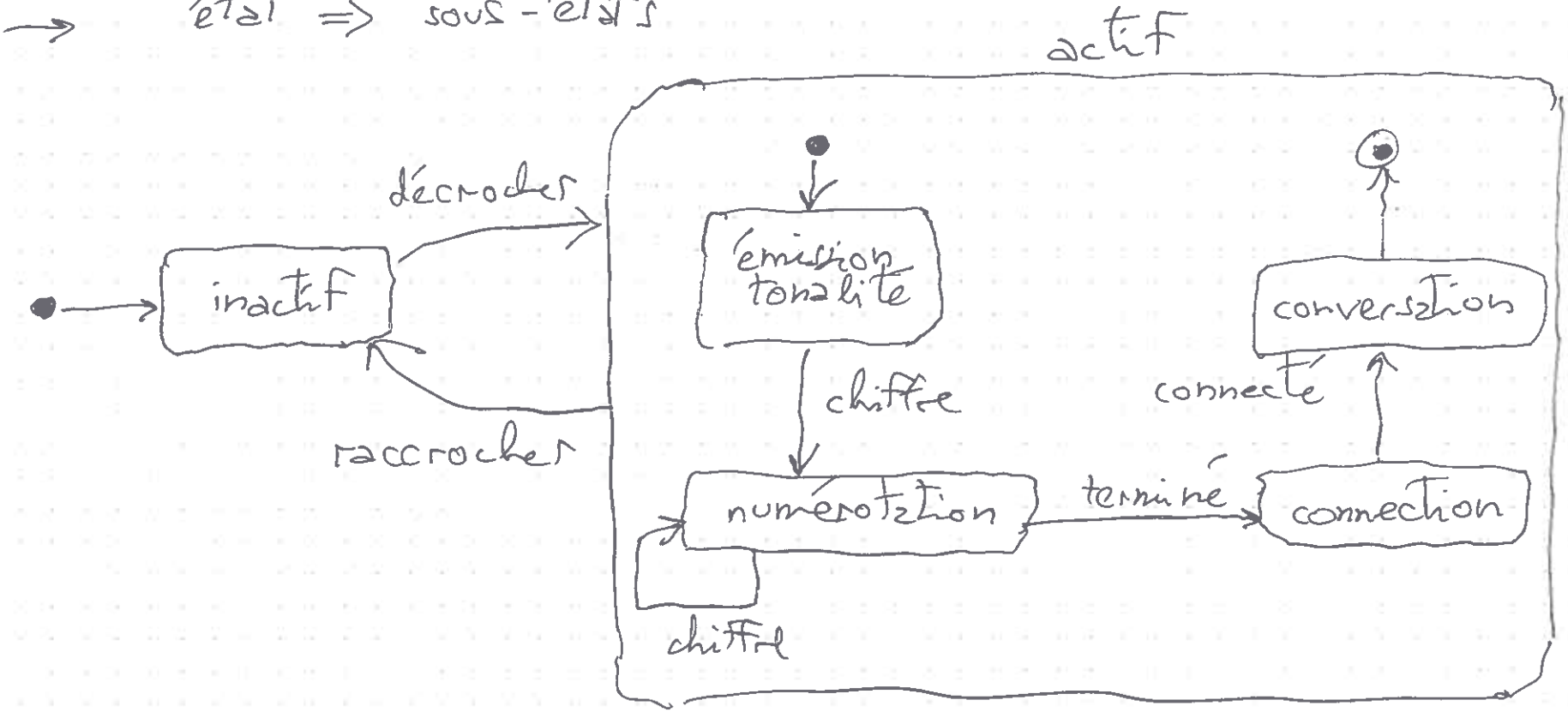
la transition n'est effectuée que si la garde est vraie

ce qui se produit lors de la transition

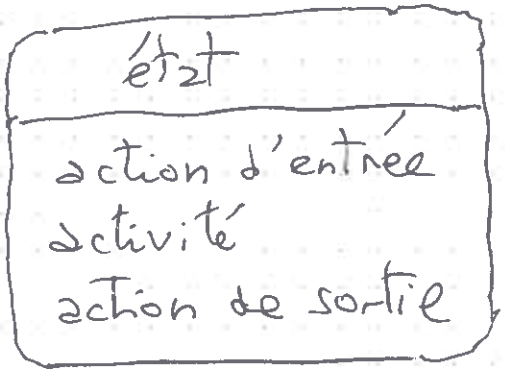


On a aussi une modélisation "hiérarchique"

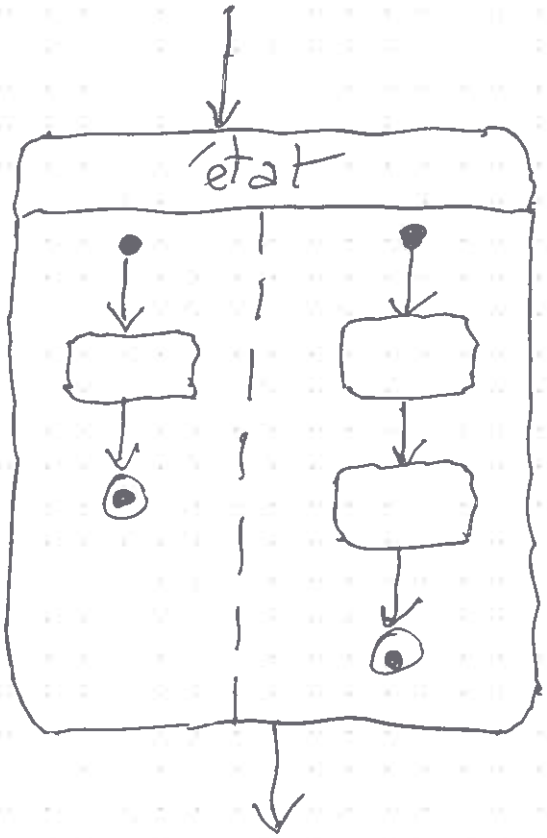
→ état ⇒ sous-états



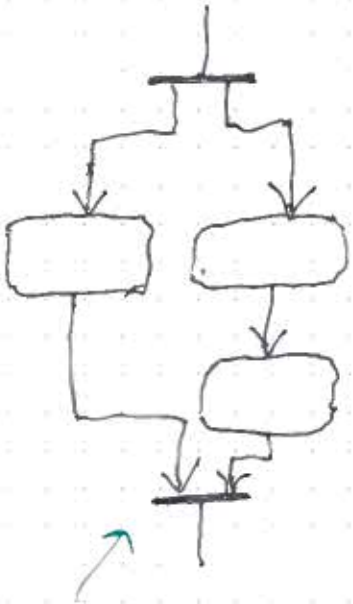
avec aussi



et on peut modéliser / dessiner de la concurrence :



ou



rendez-vous

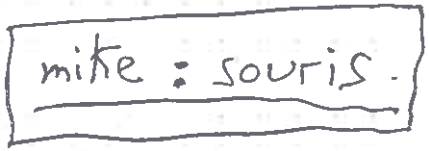
DIAGRAMMES DE SÉQUENCE

le temps qui passe

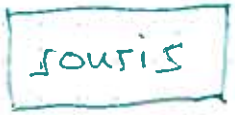
se lit implicitement

de haut en bas

les objets sont sur l'axe horizontal



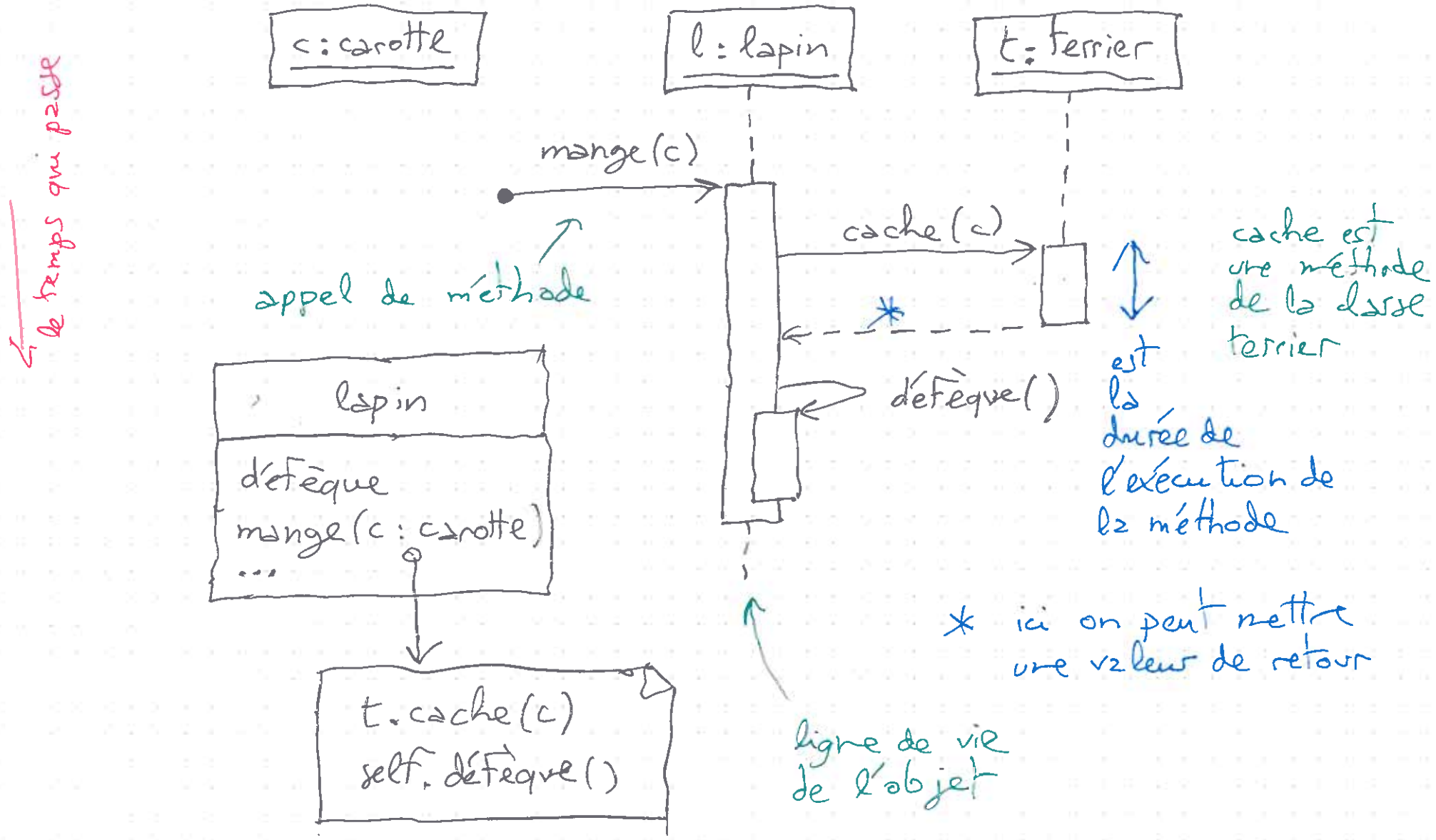
mike est une instance de la classe



on peut avoir des objets anonymes

pas de nom d'instance ici

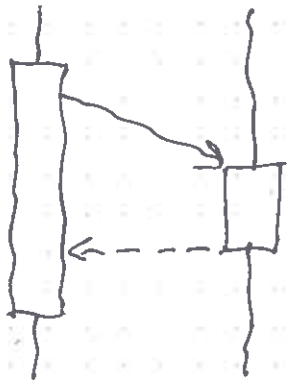
Exemple simple :



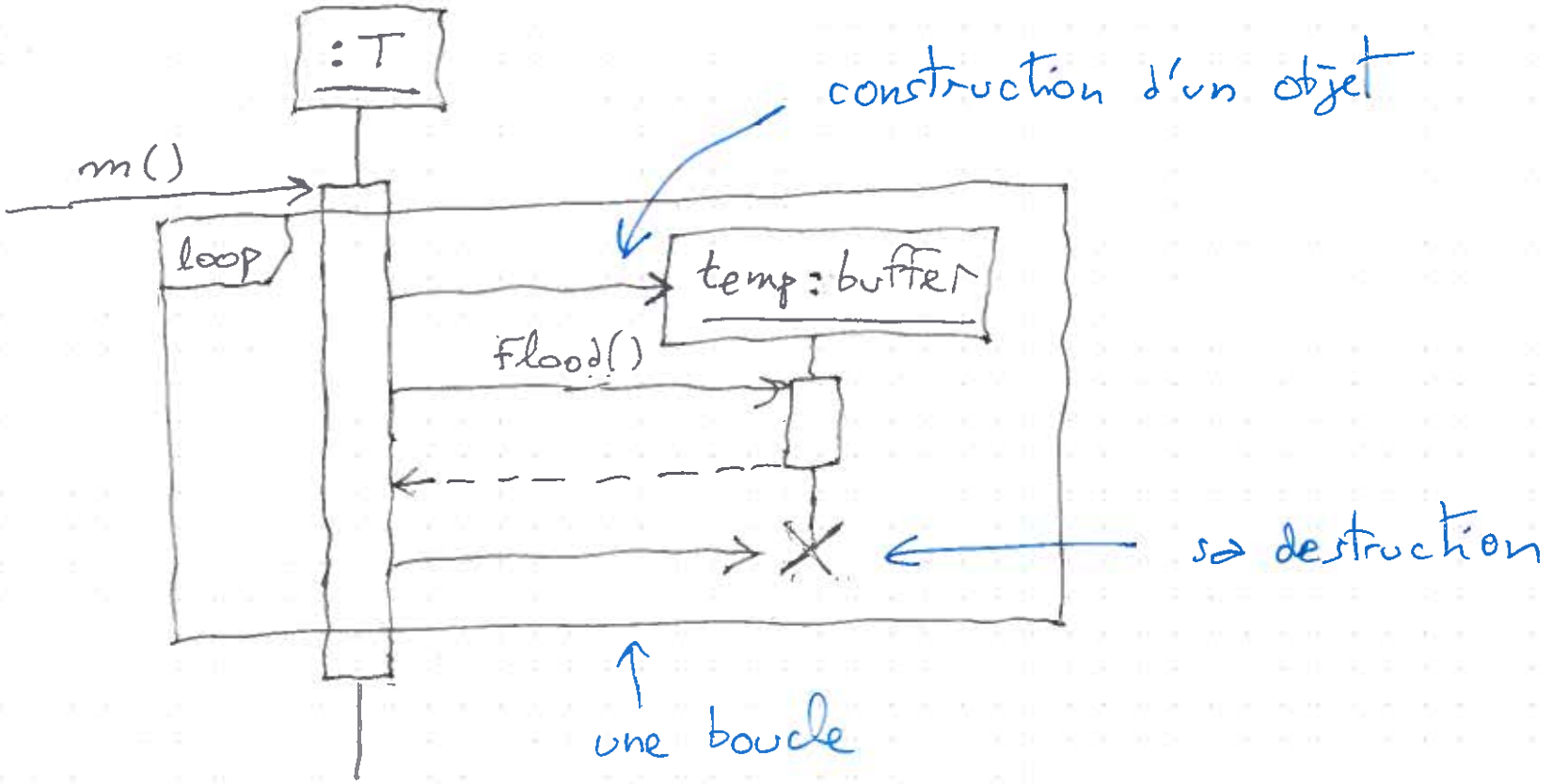
épaisse
cette barre montre que l'objet est actif

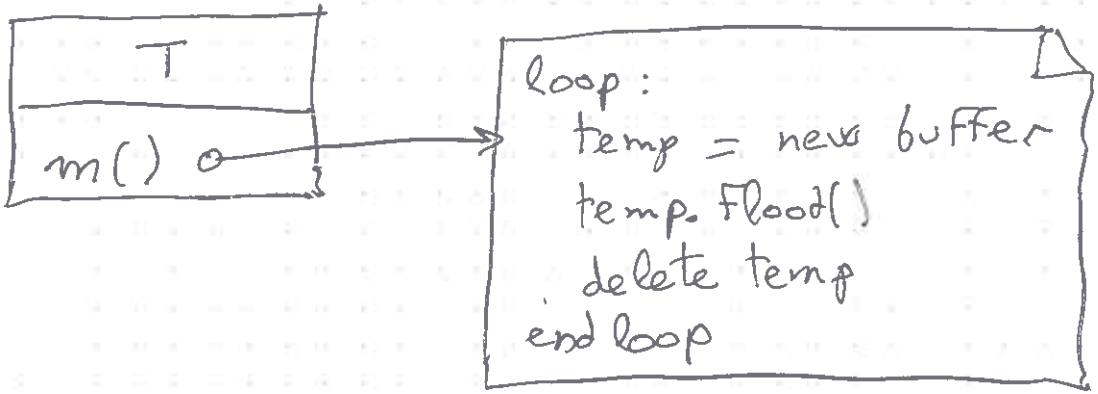
// remarque : un objet peut être
- en base de données et pas chargé en mémoire
- sérialisé et idem
- dans une application tierce qui n'est pas en mémoire (car swappée)

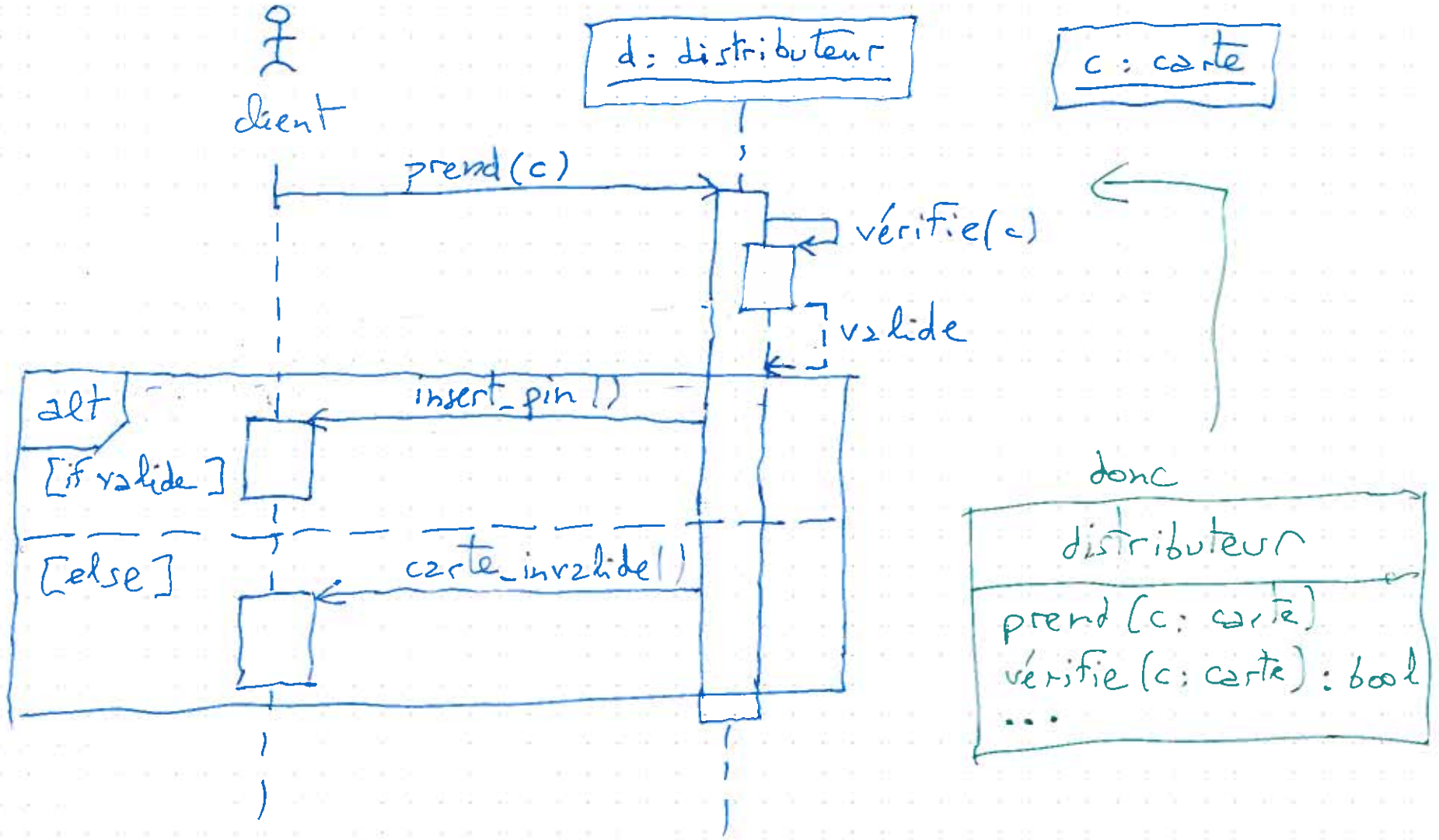
on peut indiquer le temps de passage d'un message



↕ temps entre l'appel et le début de son traitement


:T
m()
loop
temp : buffer
Flood()
construction d'un objet
sa destruction
une boucle


T
m()
loop :
temp = new buffer
temp. Flood()
delete temp
end loop

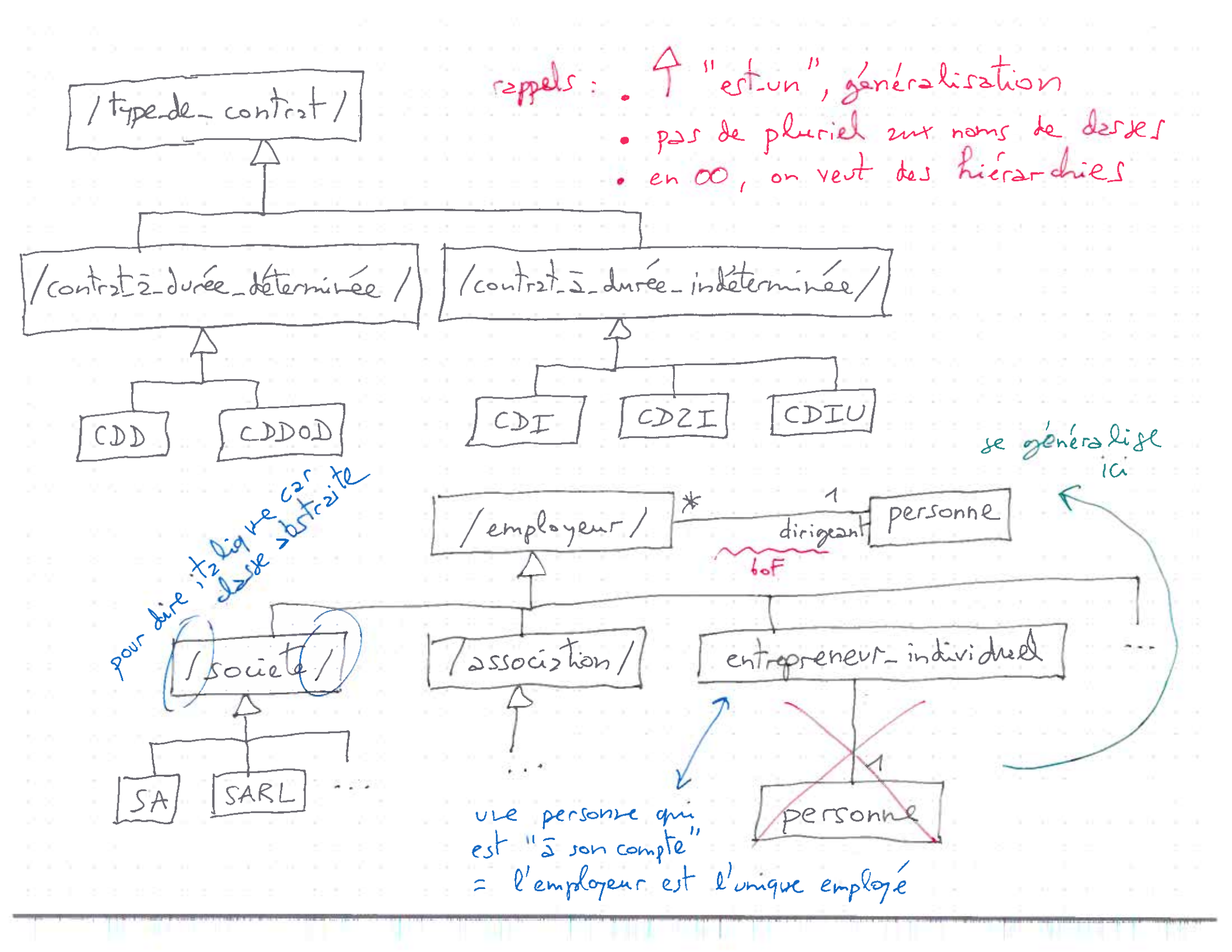

client
d : distributeur
c : carte
prend(c)
vérifie(c)
valide
alt
insert_pin()
[if valide]
[else]
carte_invalide()
donc
distributeur
prend(c: carte)
vérifie(c: carte) : bool
...

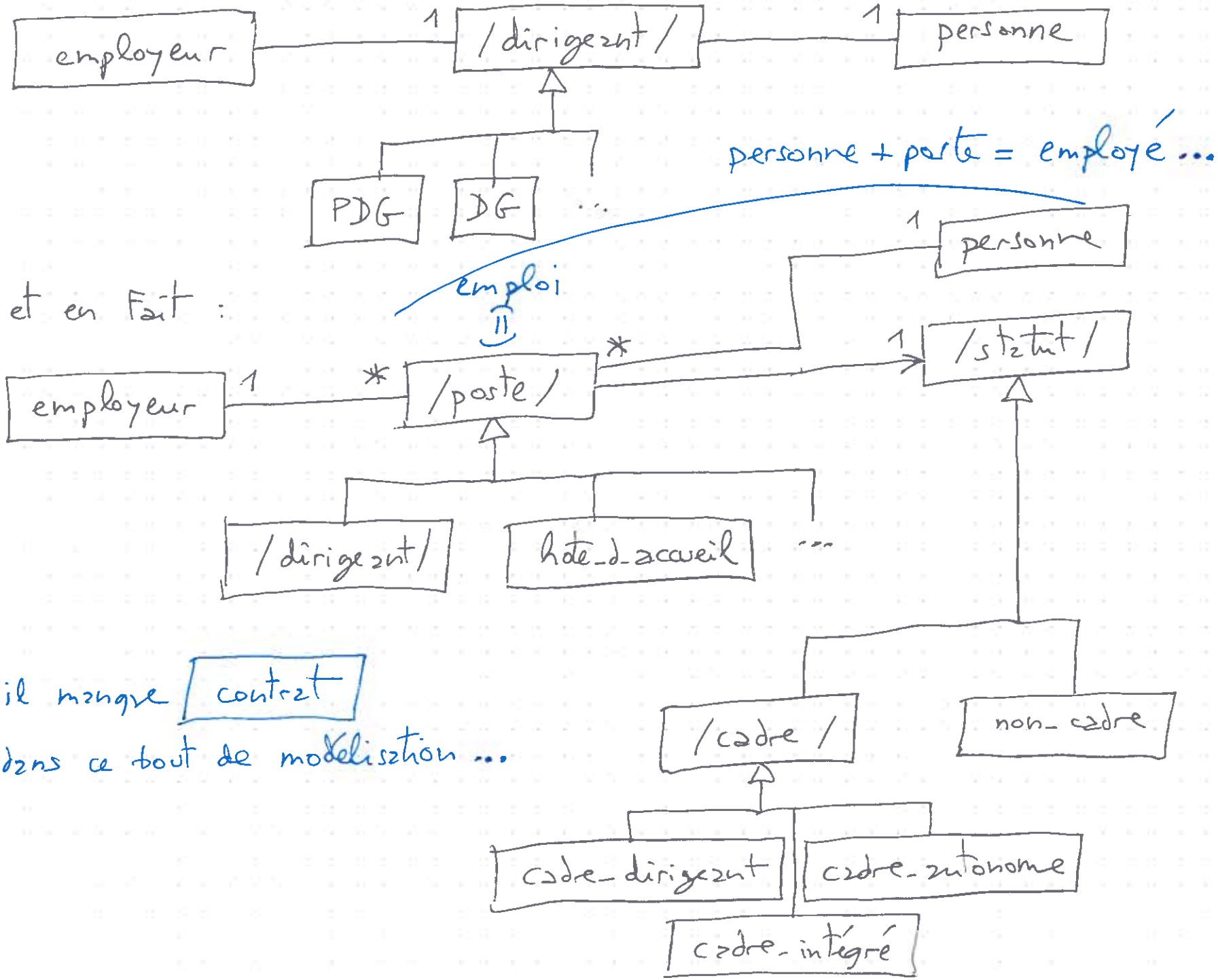
La pause STATIQUE (complément de MOD1...)

On a des sociétés qui emploient des gens.
Il y a différents types de contrats et des métiers différents.
Une association aussi peut être un employeur.
On peut être plusieurs fois employé (cumul de jobs)

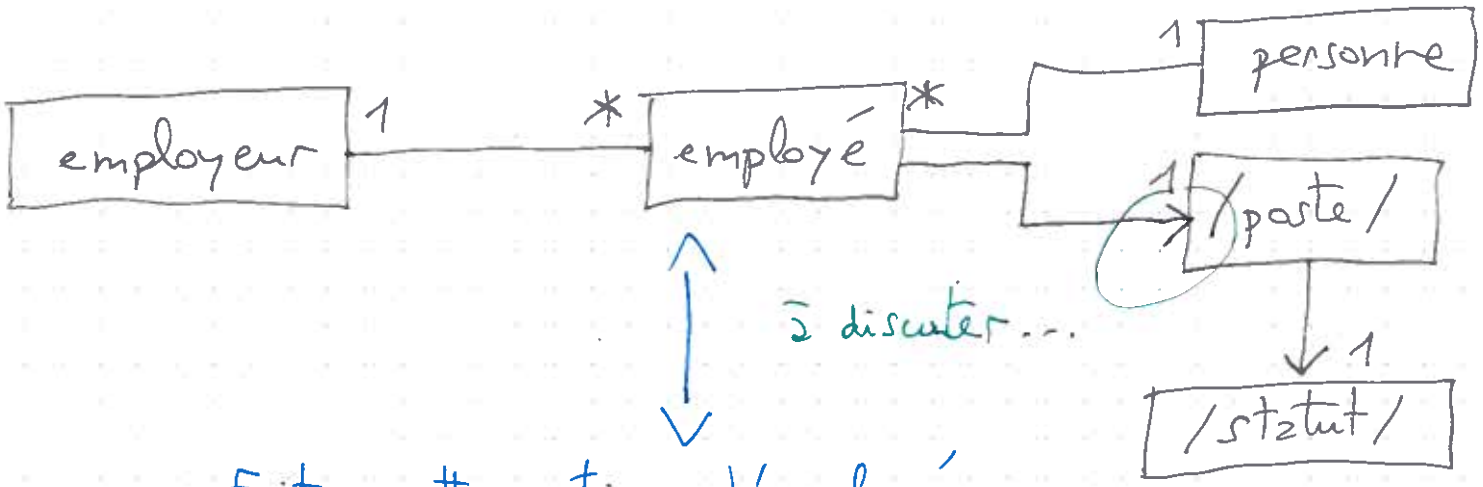
brainstorm

?


rappels :
• ↑ "est_un", généralisation
• pas de pluriel aux noms de classes
• en OO, on veut des hiérarchies
/type_de_contrat/
/contrat_à_durée_déterminée/
/contrat_à_durée_indéterminée/
CDD
CDDOD
CDI
CDZI
CDIU
se généralise ici
pour dire "ligne car classe abstraite"
/employeur/
*
1
dirigeant
personne
bof
/société/
/association/
entrepreneur_individuel
...
SA
SARL
...
...
1
personne
une personne qui est "à son compte"
= l'employeur est l'unique employé

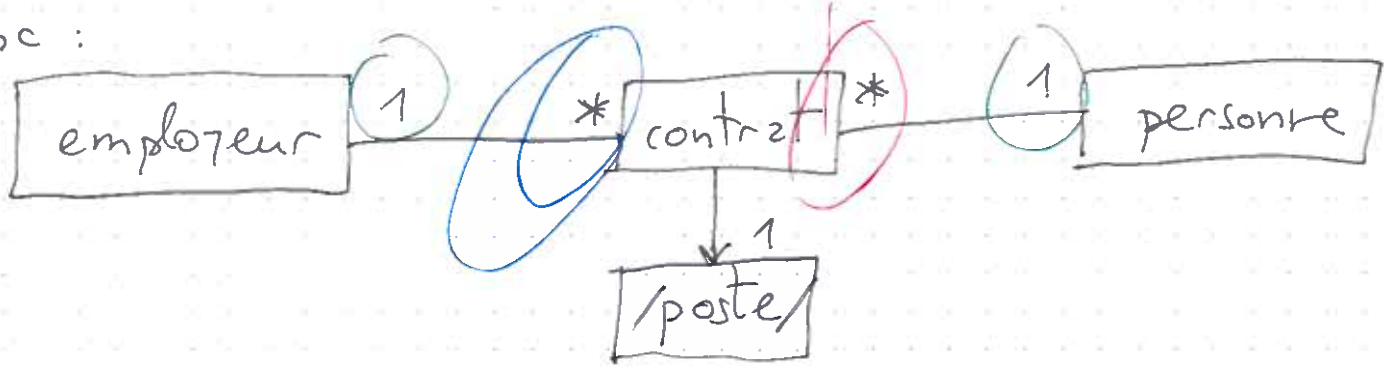

employeur
1
/dirigeant/
1
personne
PDG
DG
...
personne + poste = employé ...
et en Fait :
emploi
1
personne
employeur
1
*
/poste/
*
1
/statut/
/dirigeant/
hôte_d_accueil
...
il manque contrat
dans ce tout de modélisation ...
/cadre/
non_cadre
cadre_dirigeant
cadre_autonome
cadre_intégré

déjà :



en fait, cette notion d'employé correspond à un contrat

on a donc :



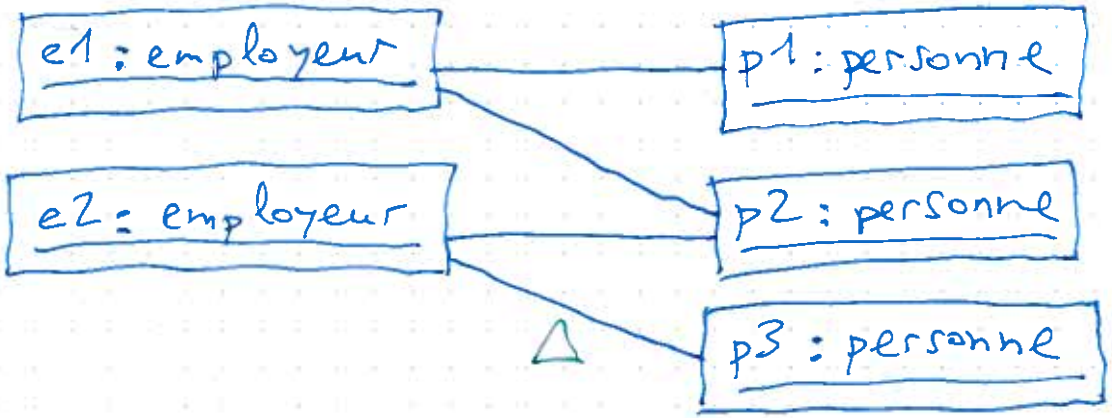
⇒
- un contrat est entre 1 employeur et 1 personne (contrat a un attribut pour chacun)
- un employeur a contractualisé plusieurs fois (une instance d'employeur a un conteneur de contrats en fait)
- une personne peut avoir plusieurs contrats

oublier la notion de contrat et

En fait, traduire qu'un employeur peut avoir plusieurs employés et qu'un employé peut travailler pour plusieurs employeurs s'écrirait :

↑ classe

exemple :



← objet / instance

e1 emploie p1 et p2 ; p2 est aussi employé par e2

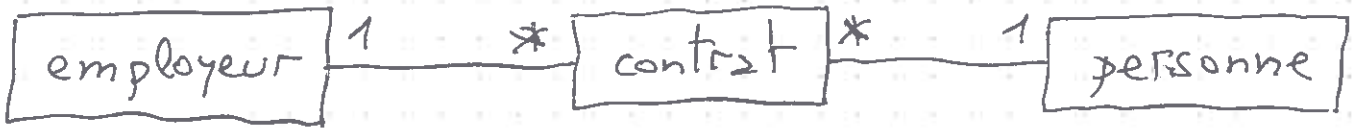
à chaque instance de relation "1 employeur — 1 personne" (comme △) / relation entre 2 instances (ex : e2 et p3)

on doit avoir 1 instance de [contrat]

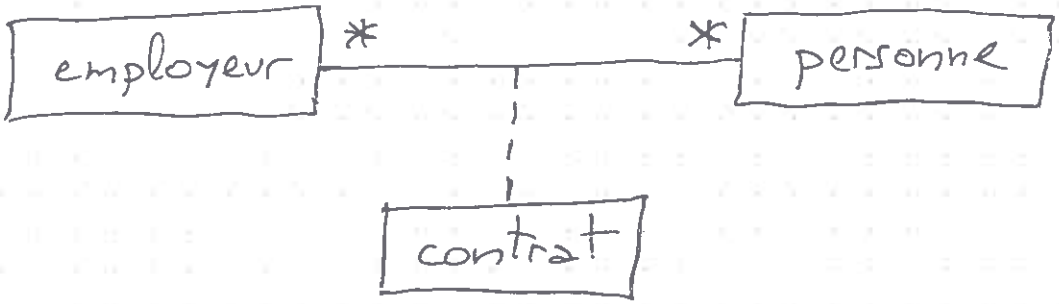
↑

ce type de classe s'appelle "classe d'association"

la modélisation

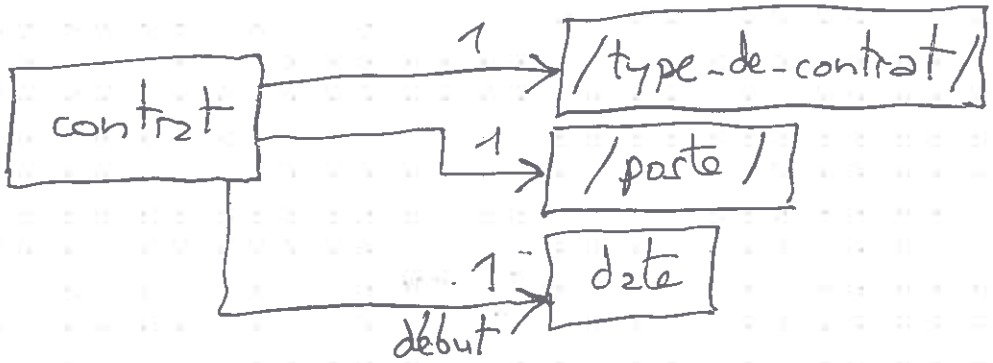


s'écrit plus explicitement en UML :

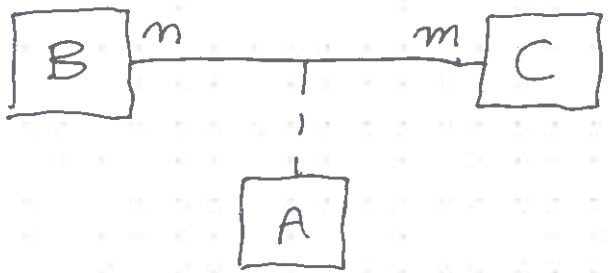


la classe d'association contrat permet de stocker des informations propres à chaque association 1 employeur / 1 personne employée

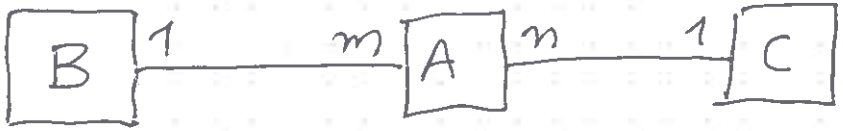
on a



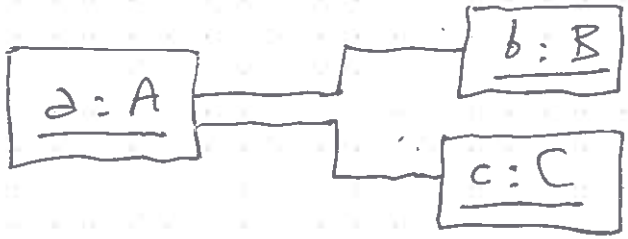
le diagramme



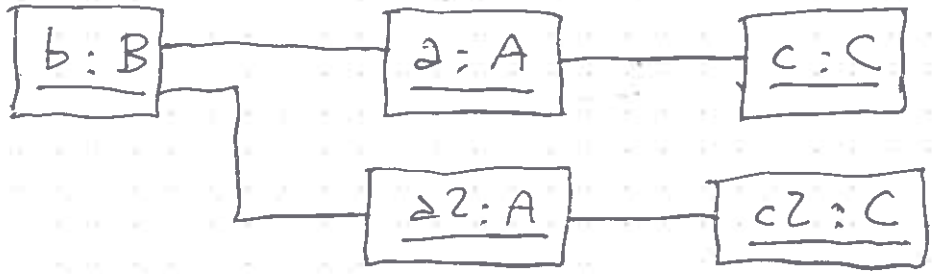
se traduit en :



OK pour la lecture des multiplicités ?

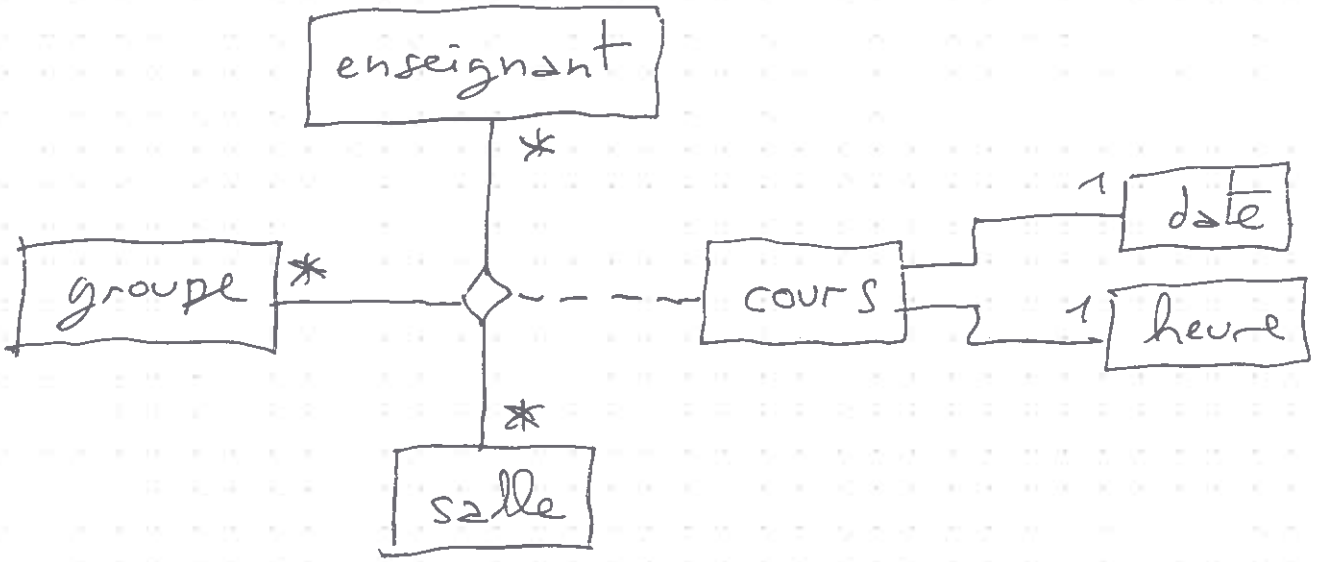


et

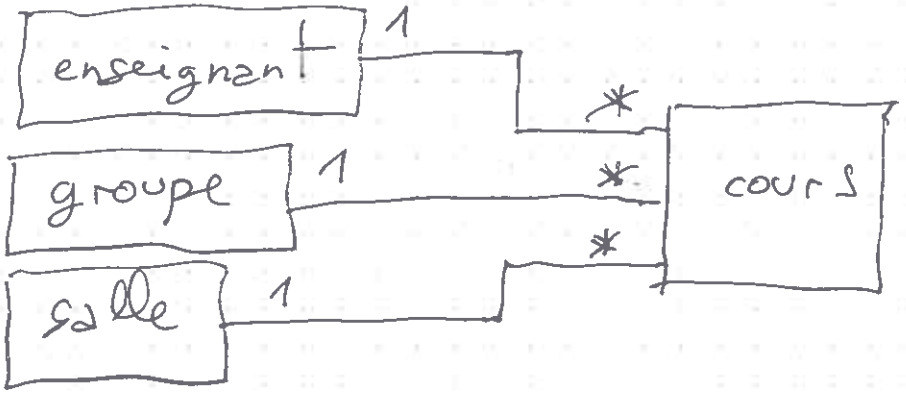


ici $m = 2$

on peut avoir des classes d'association n-aires :



qui se traduit par



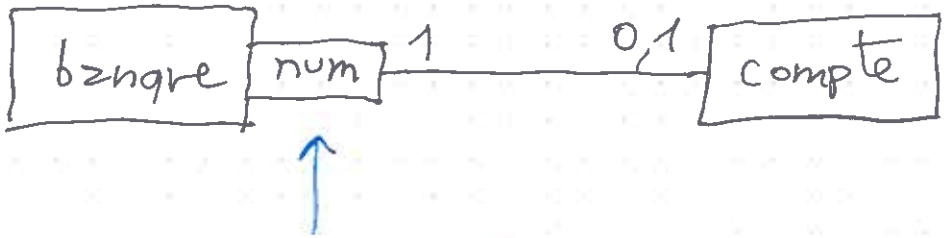
Autre exemple de notation UML particulière

une banque détient/gère plusieurs comptes

tout compte est rattaché à une banque en particulier

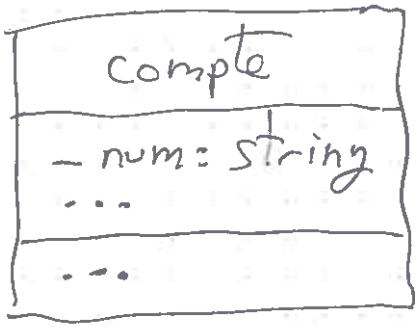
mais

avec un numéro de compte, l'instance de compte est connue ou pas
(car faux numéro ou numéro de compte mais d'une autre banque)



ici num est une clef (ou attribut qualificatif)

ou a en fait



num est un attribut de compte