


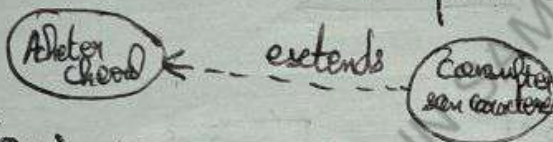
I/ Diagrammes Cas d'utilisation:

①

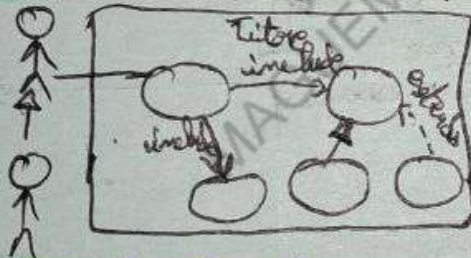
- Cas d'utilisation: 
 Verbe à l'infinitif
 déclenché par un acteur
 ou associé à un ou plusieurs autres
 Cas.

- include: $X \rightarrow Y$: Y est indispensable pour X.
 exemple: pour Commander, il faut s'identifier. $X \xrightarrow{\text{include}} Y$

- extends: $X \rightarrow Y$: Y n'est pas obligatoire pour X (sens de la flèche $Y \xrightarrow{\text{extends}} X$)
 exemple: pour acheter un cheval, on peut consulter son caractère



- Generalisation: se représente par une flèche $\rightarrow \Delta$
 exemple:



* Description Textuelle d'un cas d'utilisation:

- Nom du Cas d'utilisation.
- Boîte de description.
- Acteurs.
- Contexte.
- Données en entrée et pré-conditions.
- Données en sortie et post-conditions.
- Scénario principal pour ce cas d'utilisation: étape à suivre

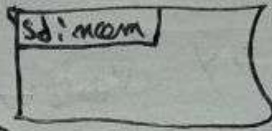
Pour réaliser ce cas.

- variantes, cas d'erreur, déviations des étapes du scénario principal, Scénarios alternatifs, Scénarios d'erreur.

exemple: Cours page 3.

II / Diagrammes de Séquences:

- cadre d'un diagramme de Seq

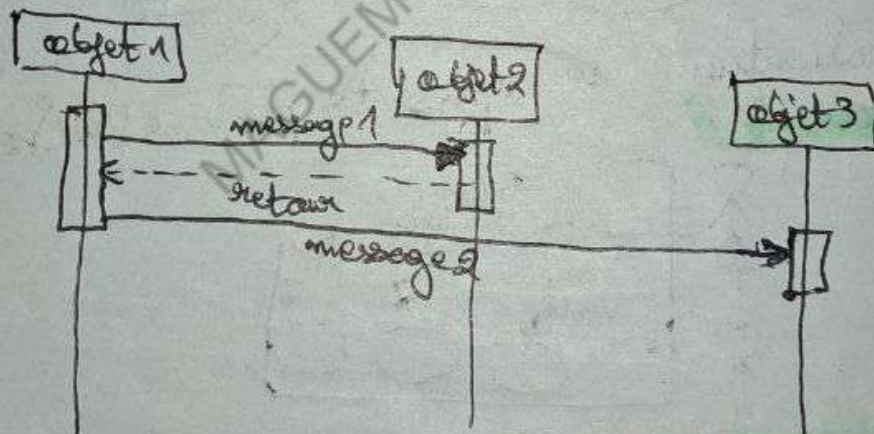


* Message synchrone: il attend la réponse.

son Message de retour: - - - - -

* Message asynchrone: il n'attend pas la réponse.

exemple:



- Les conditions se mettent dans un rectangle Alt



If [Imprimante = occupé] alors (fichier)

Action

[] deplacer(ascenseur, position)

III/ Diagrammes de classes :

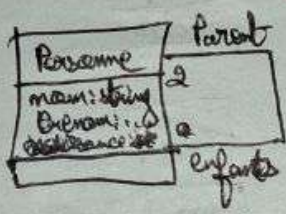


* Association :



Multiplicité (lecture contextuelle du sens)

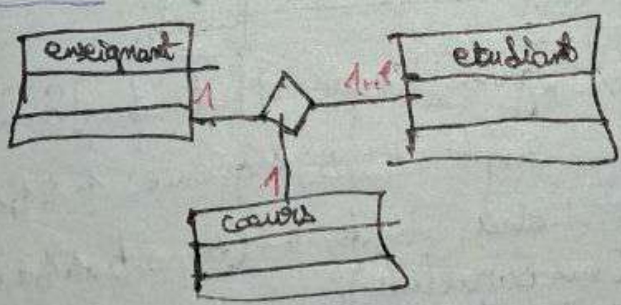
* Association Reflexive :



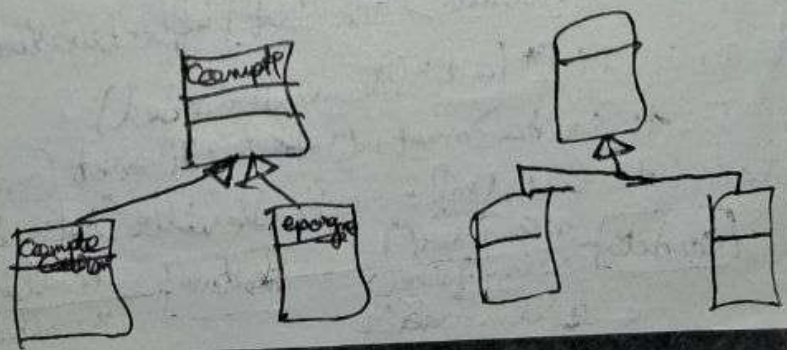
* Classe - association :



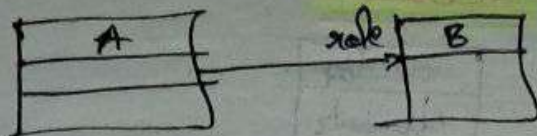
* Association n-à-n :



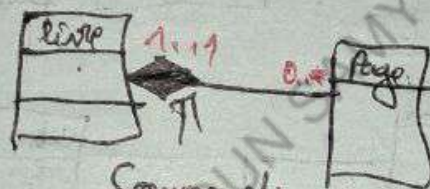
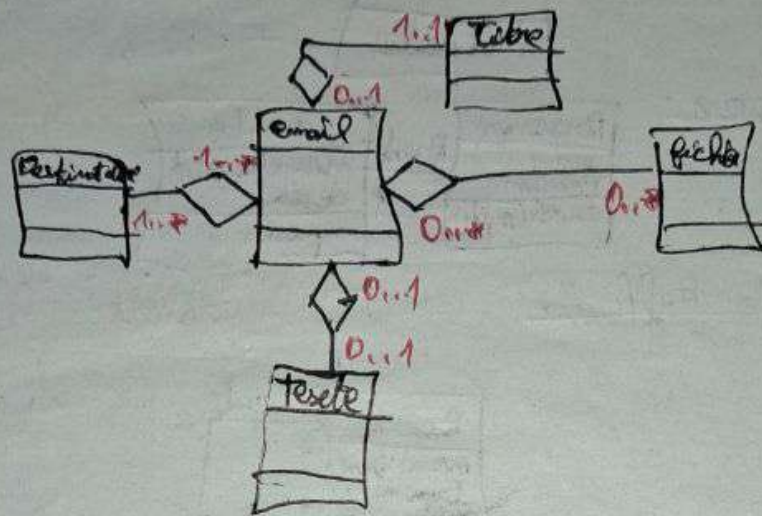
* Héritage :



④



* Aggregation:



Composition (aggregation forte)

IV/ Programme de Collaboration:



message

[Pré-secrétaire] [Garde] [Séquence] [Itération] [Valeurs] [Signature]

Numéro de Message : Simple : 1, 2, 3 ; Séquence : 1, 1, 1, 2, 3, ...

Concurrence : 1, 1.1.a, 1.1.b, 1.2 (les points a, 1.a et 1.1.b peuvent être exécutés en parallèle).

Condition: 1: [solde > montant] effectuer virement()

Iteration: 1: [* i = 1, 3] demenderCode()

Argument: 1: [solde > montant] affecter le montant (montant, destination)
 resultat: 1: listval = recherche ville (date, destination)

Conducteur 1: Demarker (M)
5: Camarguère (C)
4: dem arriere (A)

Matrone 2: allumer (M)
3: allunde (A)

Matrone