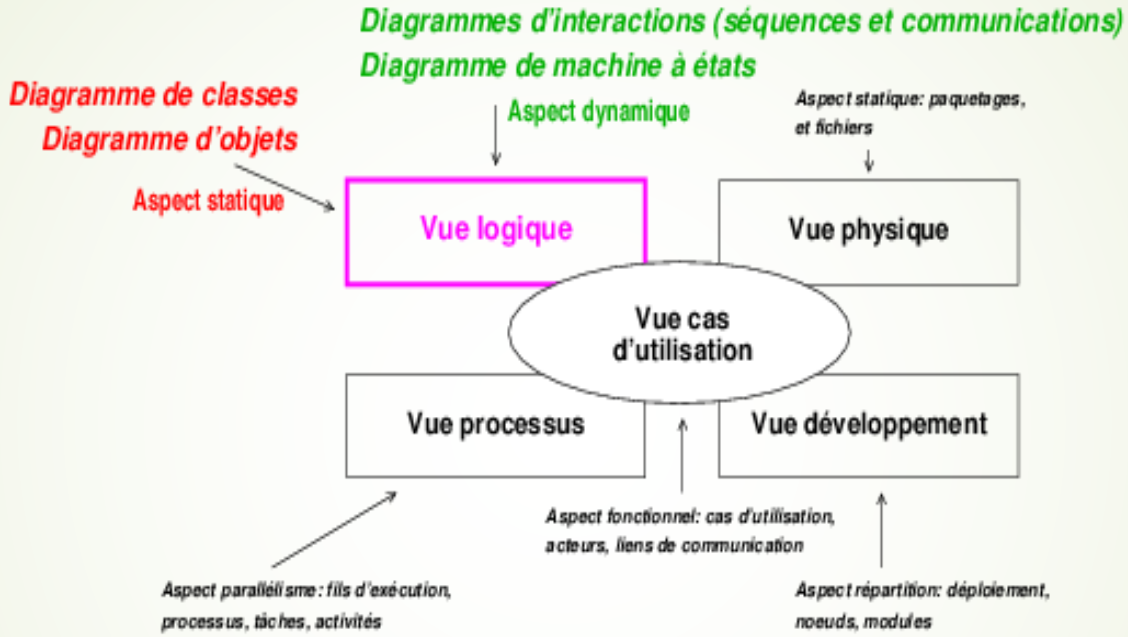
Chapitre IV : Analyse et conception, aspects dynamique de la vue logique

OBJECTIFS DU CHAPITRE:

- ✓ Modéliser l'ordre des interactions entre objets à travers un diagramme de séquence;
- ✓ Modéliser la répartition des interactions sur les liens entre objets à travers un diagramme de communication
- ✓ Modéliser le comportement des objets selon leurs états à l'aide d'un diagramme de machine à états
- ✓ Représenter la collaboration entre objets pour la réalisation d'un cas d'utilisation

Diagrammes communs à l'analyse et à la conception



- Vue logique
  - Aspects statiques = structure du problème et de la solution
    - Diagrammes de classes et d'objets
  - Aspects dynamiques = comportement des éléments de la structure
    - Diagrammes de séquence, de communications et de machine à états

PLAN

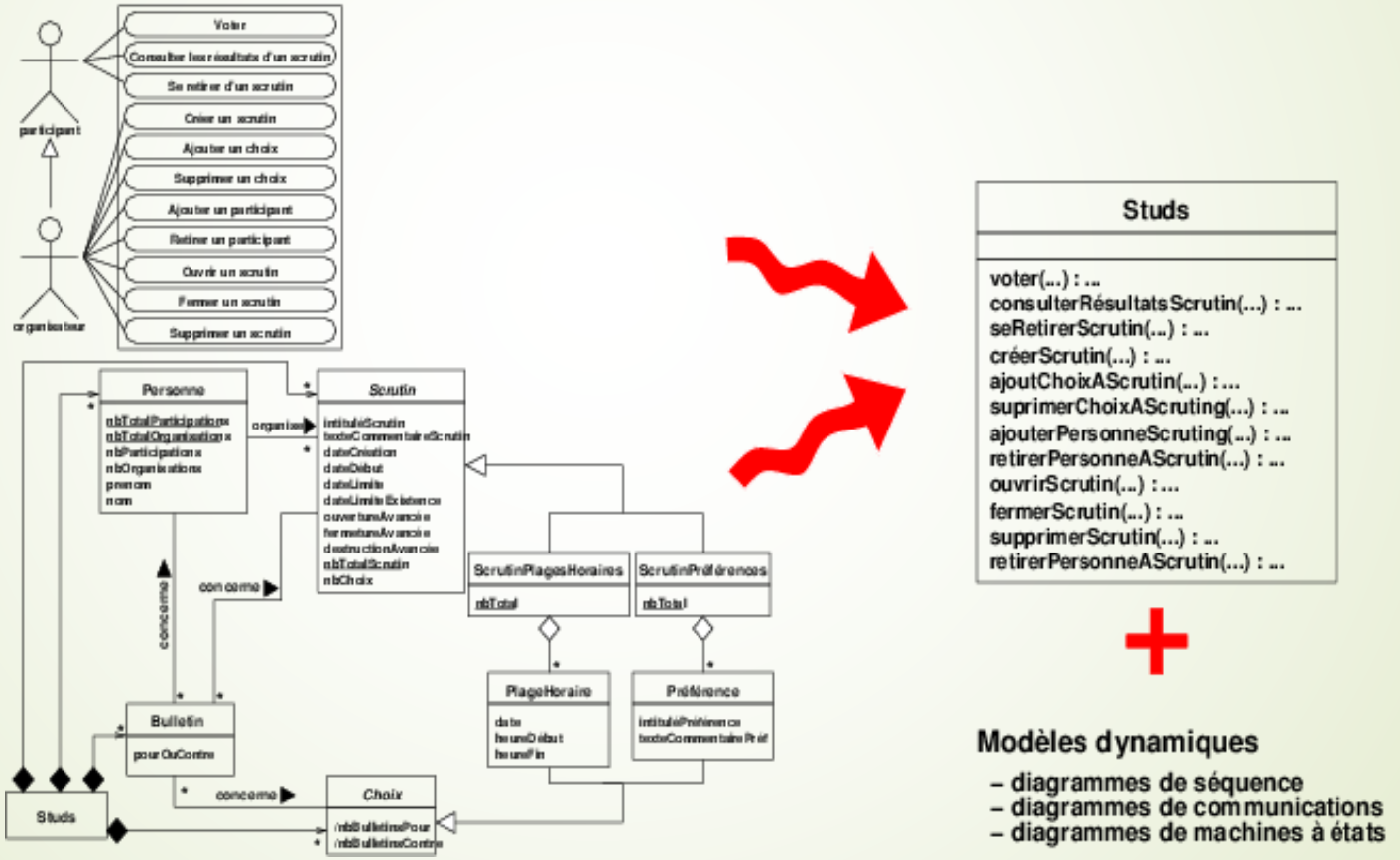
- Modélisation des aspects dynamiques;
- Diagramme de séquence;
- Diagramme de communication;
- Diagramme de machine à états.

Modélisation des aspects dynamiques

- **Points de départ** : Modèles de la vue cas d'utilisation + diagrammes de classes
- **Objectifs** :
- Identifier les différents événements venant du monde externe et montrer l'enchaînement dans le système que provoquent ces événements;
- Un événement est « quelque chose » qui se produit à un moment donné dans le temps et qui n'a pas de durée;
- Le but du modèle dynamique est de trouver les relations temporelles et événementielles entre les objets, de montrer les interactions entre les objets, et de définir les états des objets qui déterminent la réaction face à un événement.
- spécifier les algorithmes des cas d'utilisation en parcourant le graphe de classes et des objets ?

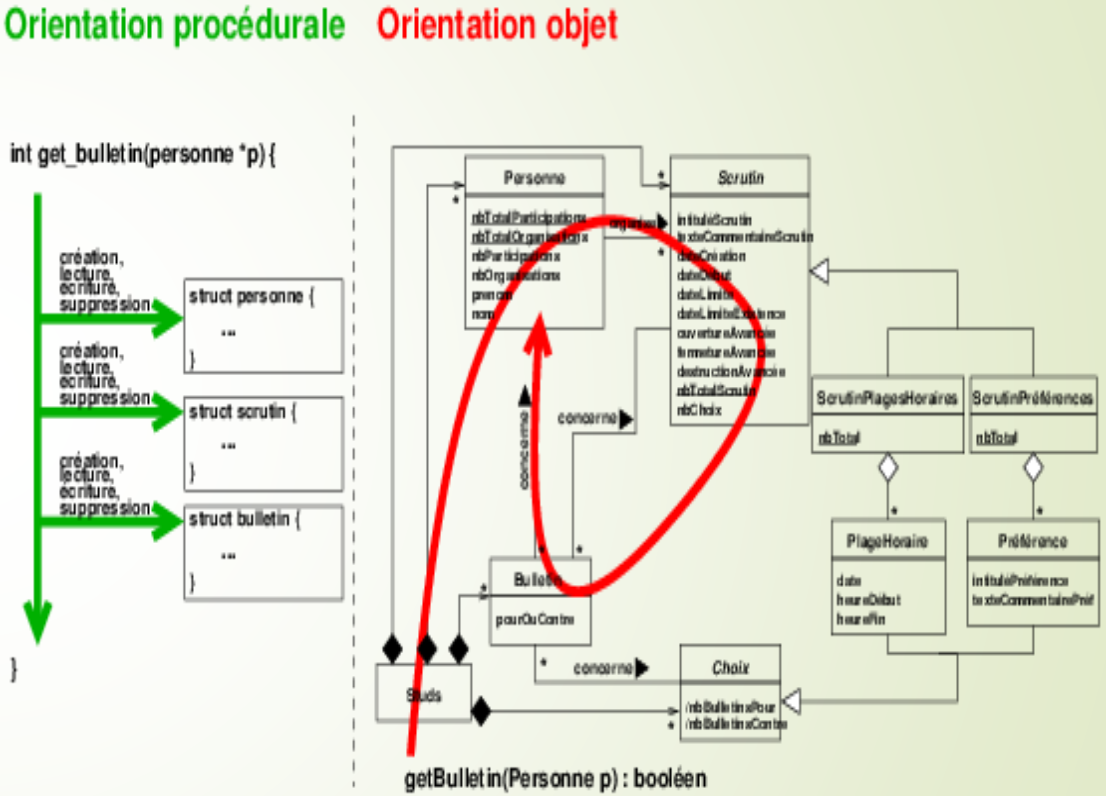
Modélisation des aspects dynamiques

- Points de départ : Modèles de la vue cas d'utilisation + diagrammes de classes
- Objectif : spécifier les algorithmes des cas d'utilisation en parcourant le graphe de classes et des objets



Algorithme : orientations procédurale et objet

- Dans l'orientation procédurale, l'écriture d'un algorithme consiste en l'écriture d'une séquence d'instructions dans un bloc (une procédure) avec des accès aux structures de données
- Dans l'orientation objet, l'écriture d'un algorithme consiste en l'écriture de plusieurs opérations réparties dans plusieurs classes, ces opérations s'appelant les unes les autres en respectant les associations entre classes.



Modèle dynamique de l'analyse et de la conception

- Les différents diagrammes du modèle dynamique de l'analyse et de la conception sont :
  - **Diagramme de séquence** : ordre des interactions entre objets
  - **Diagramme de communications** : répartition des interactions sur les liens entre objets
  - **Diagramme de machine à états** : comportement des objets selon leurs états
- Les diagrammes UML du modèle dynamique de l'analyse et de la conception sont complémentaires. À ces trois types de diagrammes, UML ajoute deux autres types de diagrammes:
  - le **diagramme de temps** qui spécifie précisément l'instant d'occurrence des événements dans le temps
  - **le diagramme de vues globales des interactions** qui permet de rassembler dans le même diagramme des comportements différents sans leurs détails pour dessiner une vue globale des interactions.

Diagramme de séquence

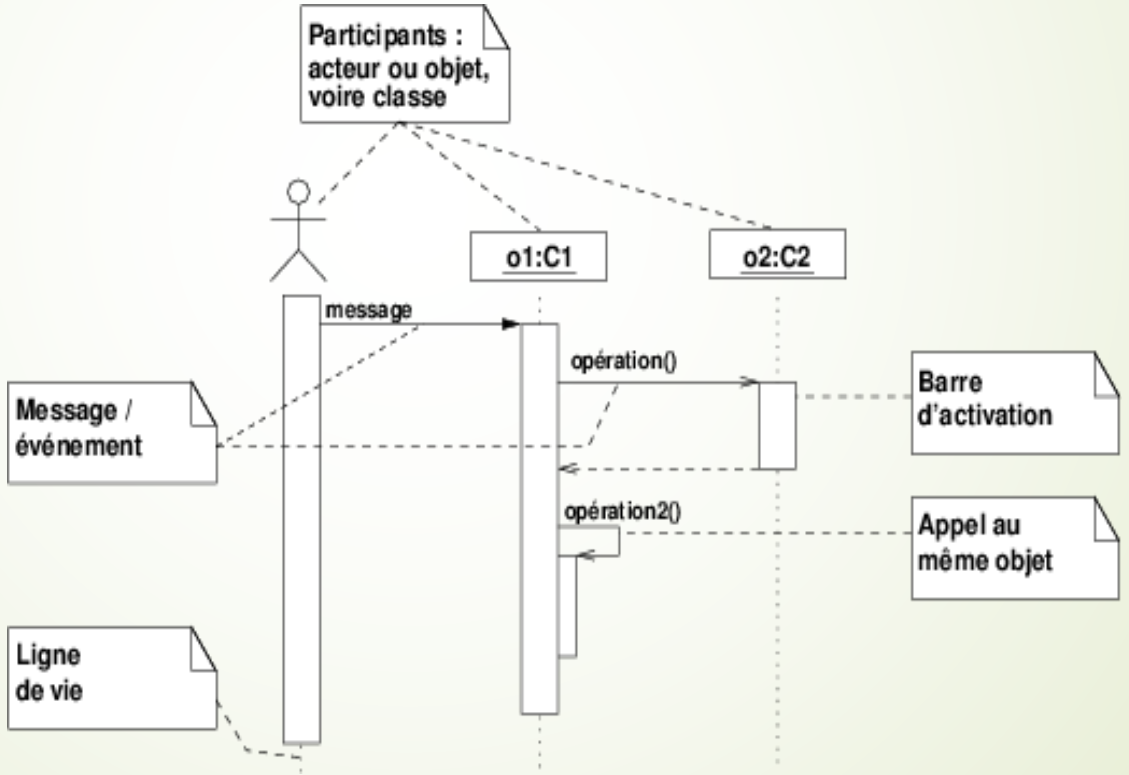
- Modéliser l'ordre des interactions
- Participants, Ligne de vie, Temps et messages
- Exemple de diagramme de séquence « Ouvrir un scrutin »
- Syntaxe et types de messages
- Création et suppression d'objets
- Fragments de séquence « ref » et « opt »
- Fragments de séquence « loop »
- QCM

Modéliser l'ordre des interactions

- Le diagramme de séquence montre l'ordre des échanges de messages et le passage du temps.
- C'est un diagramme dit temporel. Les principaux concepts sont les **objets participants à la séquence**, le **temps**, les **messages**, la **création** et la **suppression** de participants.
- Un diagramme de séquence spécifie le comportement d'un cas d'utilisation ou d'une partie de celui-ci.
- Comme ces diagrammes deviennent vite imposants en taille, la notion de fragment permet de les construire de façon modulaire.

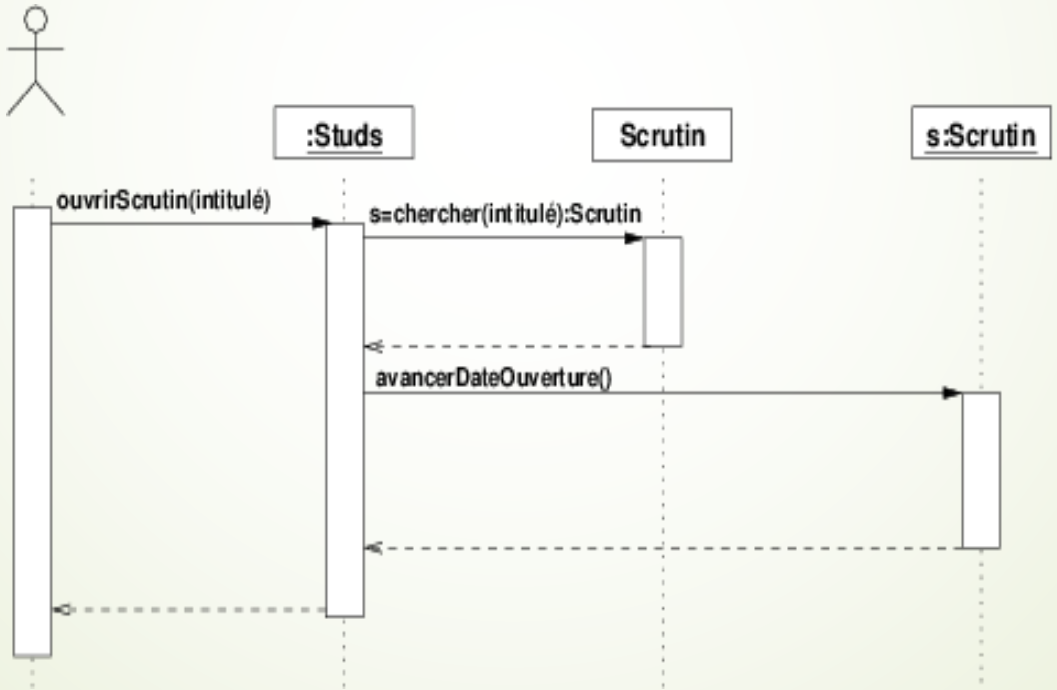
Participants, Ligne de vie, Temps et messages

- Chaque participant possède une ligne de vie représentée par une ligne verticale en pointillée.
- Une flèche reçue par un participant modélise la réception d'un message et se traduit par l'exécution d'une opération.
- La durée de vie de l'opération est symbolisée par un rectangle appelé dans la notation UML une barre d'activation(elles sont facultatives).



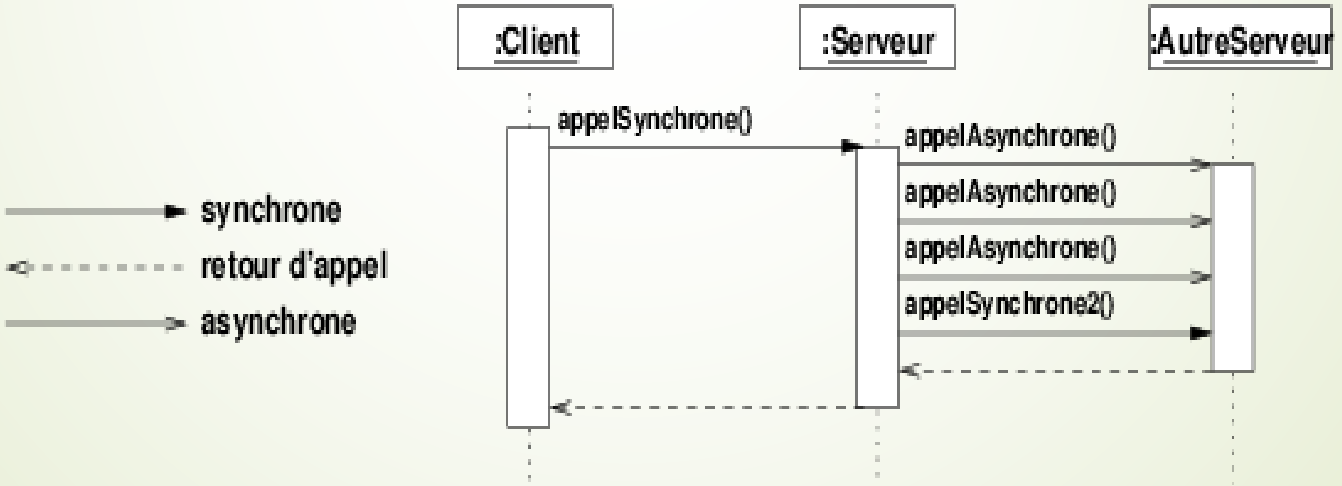
Exemple de diagramme de séquence « Ouvrir un scrutin »

- Ce diagramme de séquence très simple modélise les interactions du cas d'utilisation « Ouvrir un scrutin » (ie avancer la date d'ouverture du scrutin).
- Le diagramme montre des interactions entre un acteur, une classe et deux objets.
- L'acteur est celui qui initie la séquence. La classe est appelée pour une opération de classe.
- L'objet trouvé lors de l'appel à l'opération chercher est utilisé pour un appel d'opération d'instance : avancerDateOuverture



Syntaxe et types de messages

- **Syntaxe complète**
  - *attribut = nomMessage(arguments) : type_retour*
    - *attribut* peut être un attribut de l'objet appelant ou une variable locale
    - *nom_message* est une opération de l'objet appelé
- **Synchrone :** l'expéditeur est bloqué pendant le traitement
  - Le retour d'appel est optionnel (implicite)et spécifie la fin du traitement
- **Asynchrone :** l'expéditeur continue son exécution pendant le traitement du message

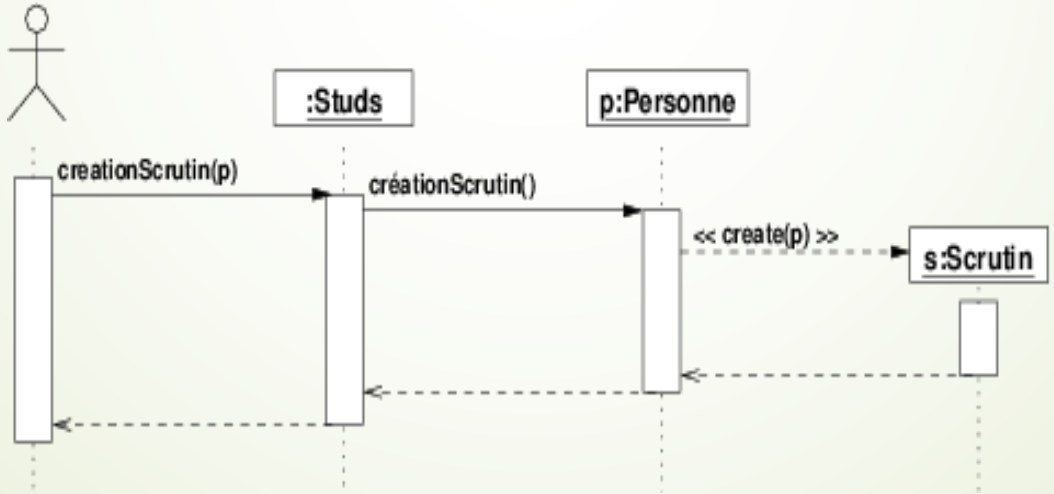


Syntaxe et types de messages(suite)

- Un *événement* est une interaction pendant laquelle « quelque chose » arrive. Les événements sont les constituants de base des messages, aussi appelés « signaux » en automatique.
- Un *message* est constitué d'un événement d'émission chez l'appelant et d'un événement de réception chez l'appelé, et possède une signature : « `attribut = nom_message(arguments) : type_retour` ».
- La notation UML des diagrammes de séquence n'oblige pas à renseigner tous les éléments des prototypes des messages. Les premiers diagrammes de séquences de l'analyse indiquent par exemple uniquement les noms des opérations. Ensuite, les mêmes diagrammes sont raffinés pour y ajouter les arguments et les types de retour, puis les attributs des classes appelantes ou variables locales recevant les valeurs de retour.
- Un message asynchrone ne possède pas de valeur de retour.

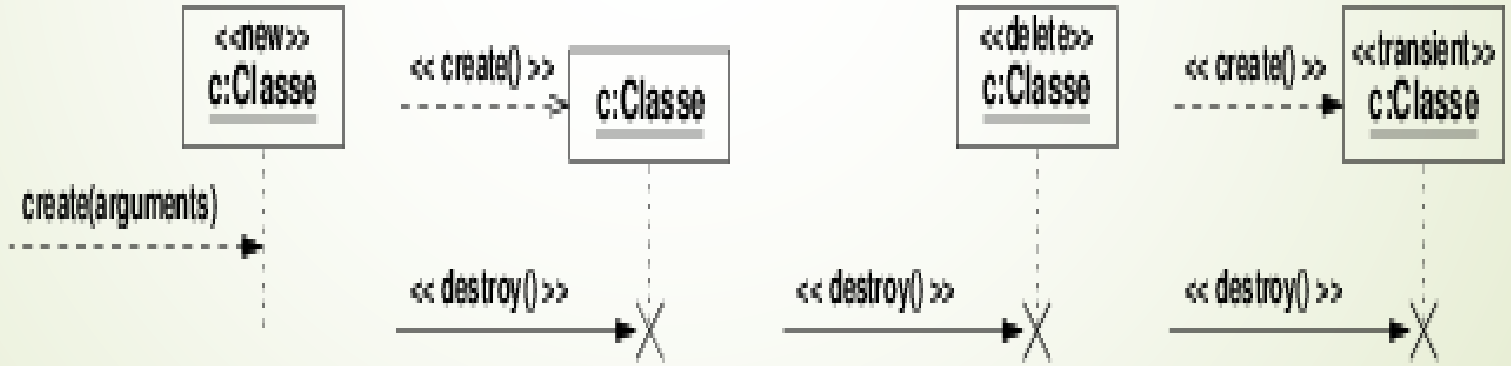
Création et suppression d'objets

- Certains objets vivent pendant tout le diagramme, d'autres sont créés et/ou meurent pendant la séquence.
- Pour montrer qu'un participant est créé lors de la séquence, on peut soit placer l'objet en haut du diagramme en y ajoutant le stéréotype «*new*» et utiliser un message synchrone appelant l'opération `create(arguments)`,
- soit placer l'objet plus bas dans le diagramme au niveau du message synchrone de création et utiliser le stéréotype «`create(arguments)`» pour nommer ce message synchrone.



Création et suppression d'objets

- Par analogie avec la création d'un objet dans un diagramme de séquence, la destruction d'un objet pendant une séquence est modélisée en plaçant l'objet en haut du diagramme en y ajoutant le stéréotype «*delete*» et utiliser un message synchrone appelant l'opération *destroy()* .
- Enfin, un objet est dit transitoire (en anglais, *transient*) lorsqu'il est créé puis détruit durant la même séquence. Le stéréotype de l'objet est alors « «*transient*» ».

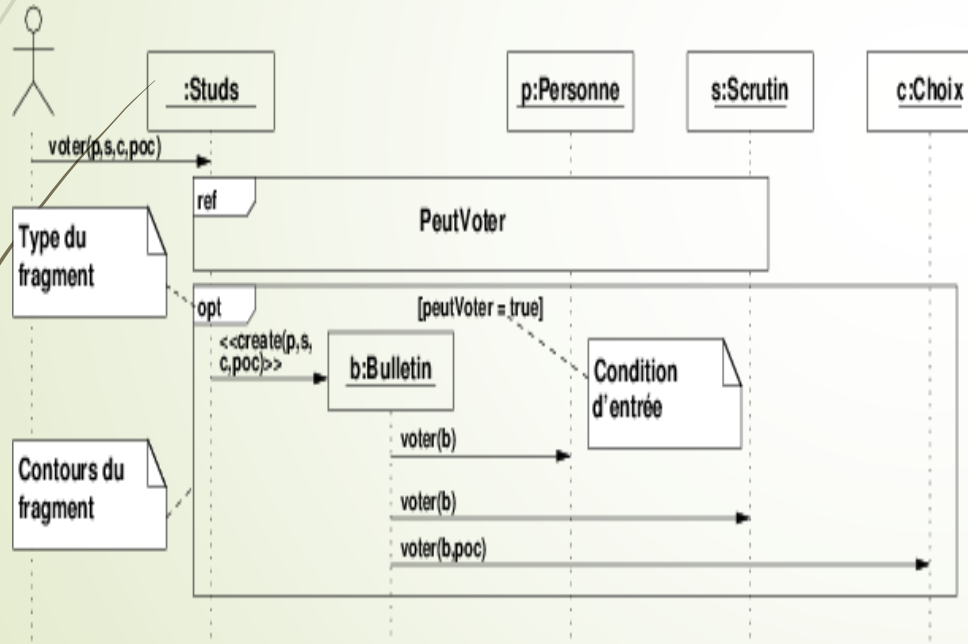


Fragments de séquence « ref » , « opt » et « loop »

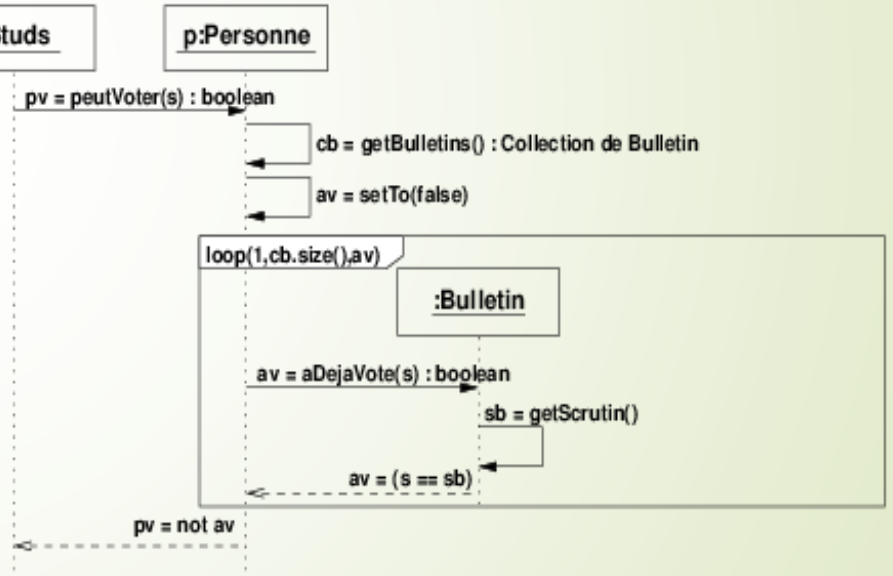
- UML propose une notion de bloc appelé « fragment de séquence » permettant d'inclure dans un rectangle des sous-parties de diagrammes de séquence.
- **Le fragment de séquence *ref*** permet d'inclure une sous-séquence du diagramme de séquence, la sous-séquence étant décrite dans un autre diagramme de séquence.
- **Le fragment de séquence *opt*** présente une sous-séquence exécutée si une condition de garde est vraie. Les termes de la condition sont souvent des valeurs de retour des messages précédant dans le temps le fragment de séquence optionnel.
- **Le fragment de séquence *loop*** permet d'itérer un traitement un nombre maximum de fois jusqu'à une condition qui peut faire sortir de la boucle avant que le nombre de fois maximum ne soit atteint.

Fragments de séquence « ref » et « opt »

- ***ref*** : sous-séquence détaillée dans un autre diagramme de séquence
- ***opt*** : sous-séquence optionnelle exécutée si condition de garde est vraie
- ***loop*** : boucle un nombre maximum de fois tant que la condition est vraie
  - loop(valeur initiale, maximum, condition)



poc=pour ou contre



pv=peut voter; av=a voté

QCM

1- Quelles entités peuvent être dessinées dans un diagramme de séquence ?

a- fragment
b- instance
c- message
d- condition
e- opération

2- Un objet se distingue-t-il d'une classe parce qu'il est souligné ?
3- Pendant un message synchrone, l'expéditeur est-il bloqué en attente d'une réponse ?
4- En réponse à un message synchrone, l'appelé renvoie-t-il un seul retour d'appel ?
5- Un fragment de séquence est-il une partie optionnelle d'une séquence ?

Diagramme de communications

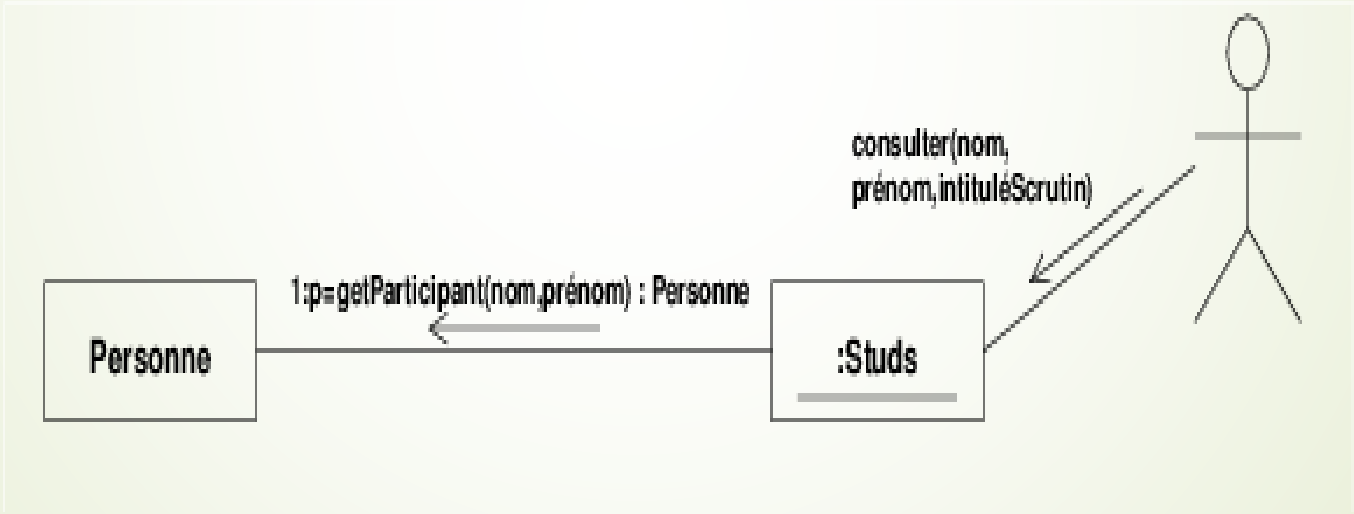
- Modéliser les liens d'interactions
- Participants, liens d'interaction et messages
- Messages conditionnés, messages en séquence
- Messages emboités
- Itérations de messages
- Messages concurrents
- Choix entre séquences et communications
- QCM

Modéliser les liens d'interactions

- Le diagramme de communications et le diagramme de séquence sont des diagrammes équivalents (on peut construire l'un à partir de l'autre).
- Le diagramme de séquence possède une approche plus temporelle et le diagramme de communication une approche plus spatiale.
- Le diagramme de communications montre les différents messages qui se propagent d'un objet à l'autre : il attire l'attention de l'analyste et du concepteur sur le fait qu'un message transite sur une association, qui doit donc exister entre la classe de l'objet appelant et la classe de l'objet appelé.
- un objet doit avoir une méthode appropriée pour traiter chaque événement qu'il reçoit.

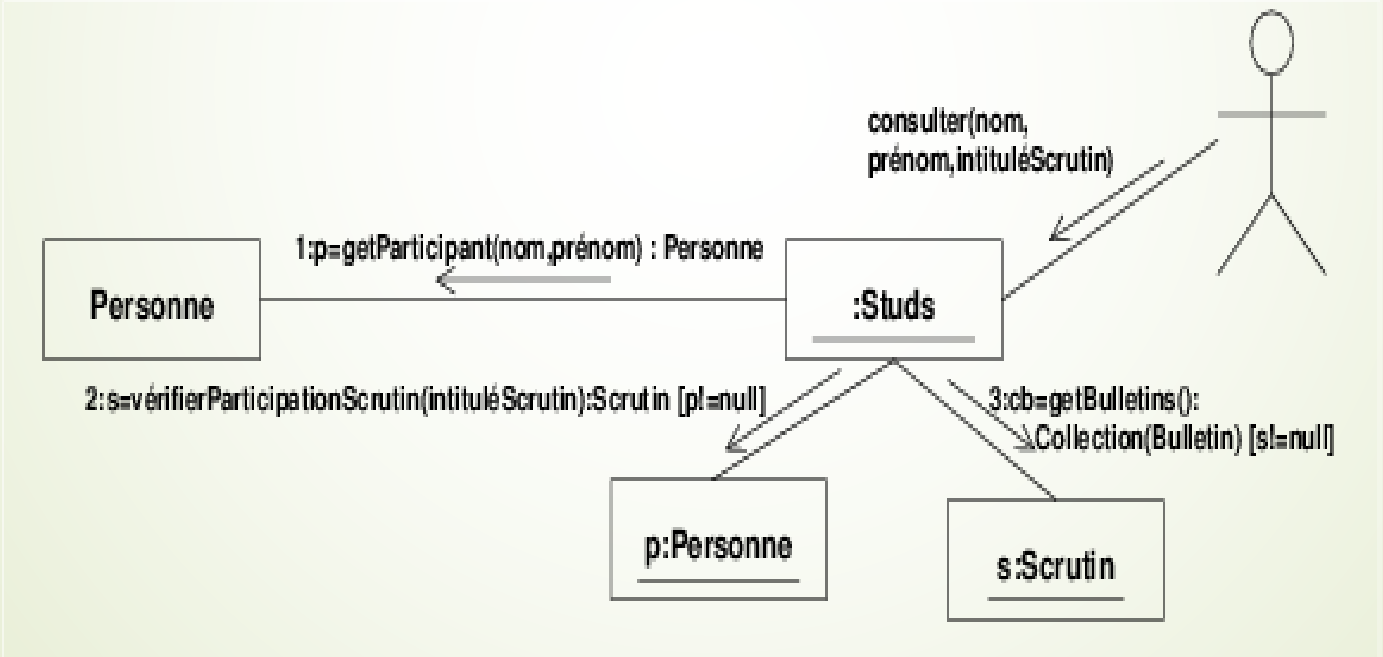
Participants, liens d'interaction et messages

- Exemple du cas d'utilisation « consulter les résultats d'un scrutin »
- Ce diagramme contient, en plus de l'objet anonyme de type Studs, la classe Personne car un appel à la méthode de classe `getParticipant()` de la classe Personne est utilisé pour réaliser ce cas d'utilisation.



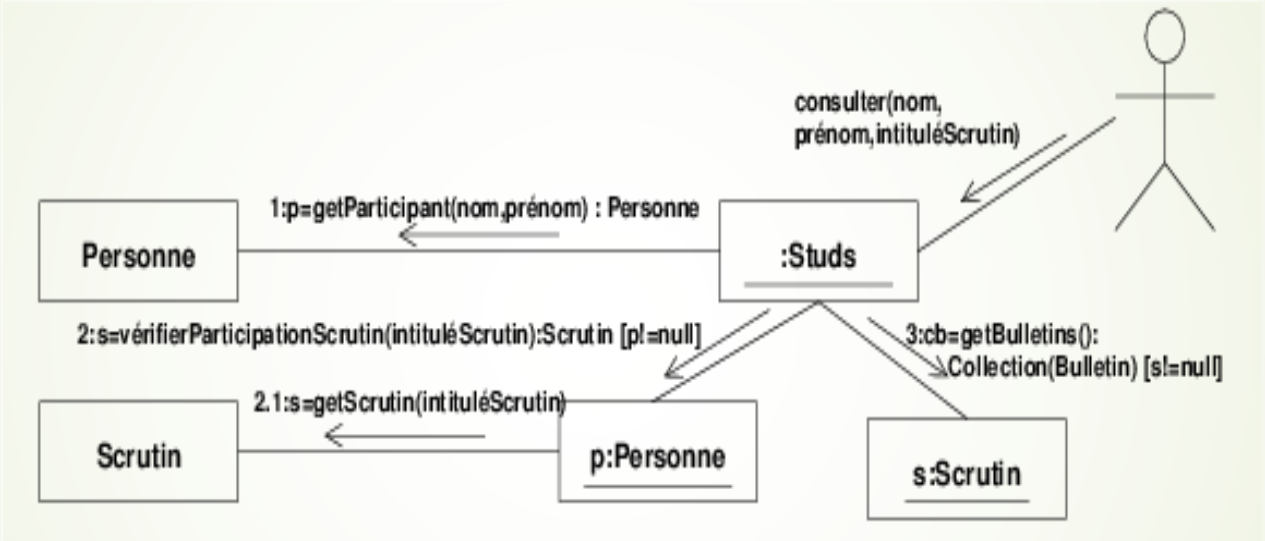
Messages conditionnés, messages en séquence

- L'exécution de l'envoi d'un message peut être conditionnée, par exemple par l'obtention d'une valeur de retour pertinente (ici, valeur de p non nulle) à un message précédent dans la séquence (ici, le message 1).
- Cette façon de faire correspond à la notion de fragment optionnel du diagramme de séquence.



Messages emboités

- Il y a emboîtement des messages pour marquer la relation de cause à effet (un objet ayant reçu le message 2 déclenche en réaction le message 2.1).



- **Lecture du diagramme :** *message 1* en réaction au message venant de l'acteur ; puis *message 2*, qui provoque le *message 2.1* ; et lorsque les traitements des *messages 2.1* et *2* sont terminés, envoi du *message 3* puis fin de traitement dans l'objet anonyme de classe Studs

Itérations de messages

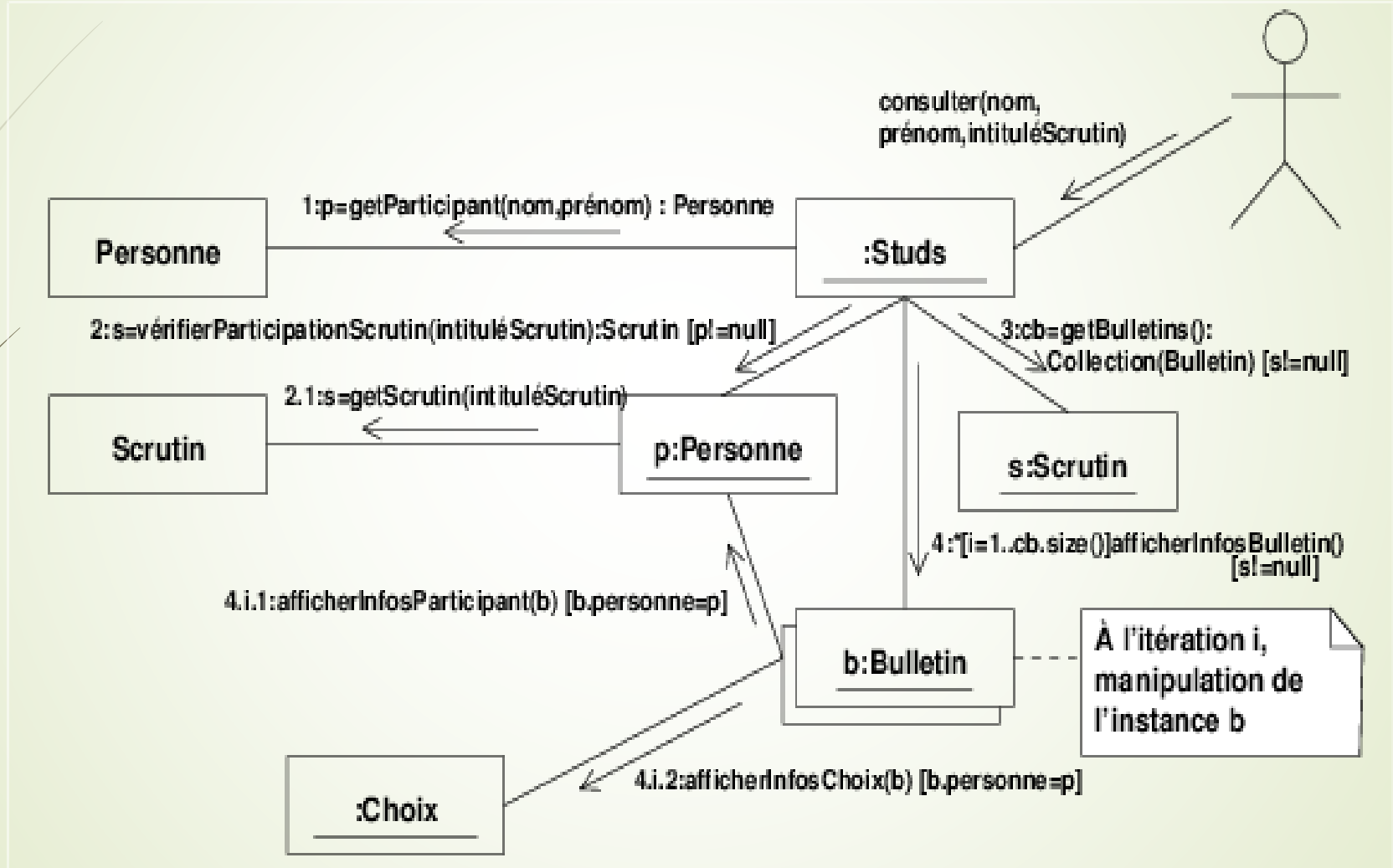
- La modélisation dans un diagramme de communications de ce qui correspond au fragment *loop* du diagramme de séquence utilise les lettres et l'étoile.

- Dans l'exemple, le message « *4 :*[i=1..cb.size()]afficherInfosBulletin()* » exprime le parcours de la collection de bulletins *cb*, de l'indice 1 à l'indice correspondant à la taille (ici écrit symboliquement *cb.size()*), et l'appel de l'opération *afficherInfosBulletin* sur chaque instance de la collection.

- à l'itération *i*, l'instance manipulée s'appelle **b**. L'étoile au début de l'expression du message 4 indique l'itération.

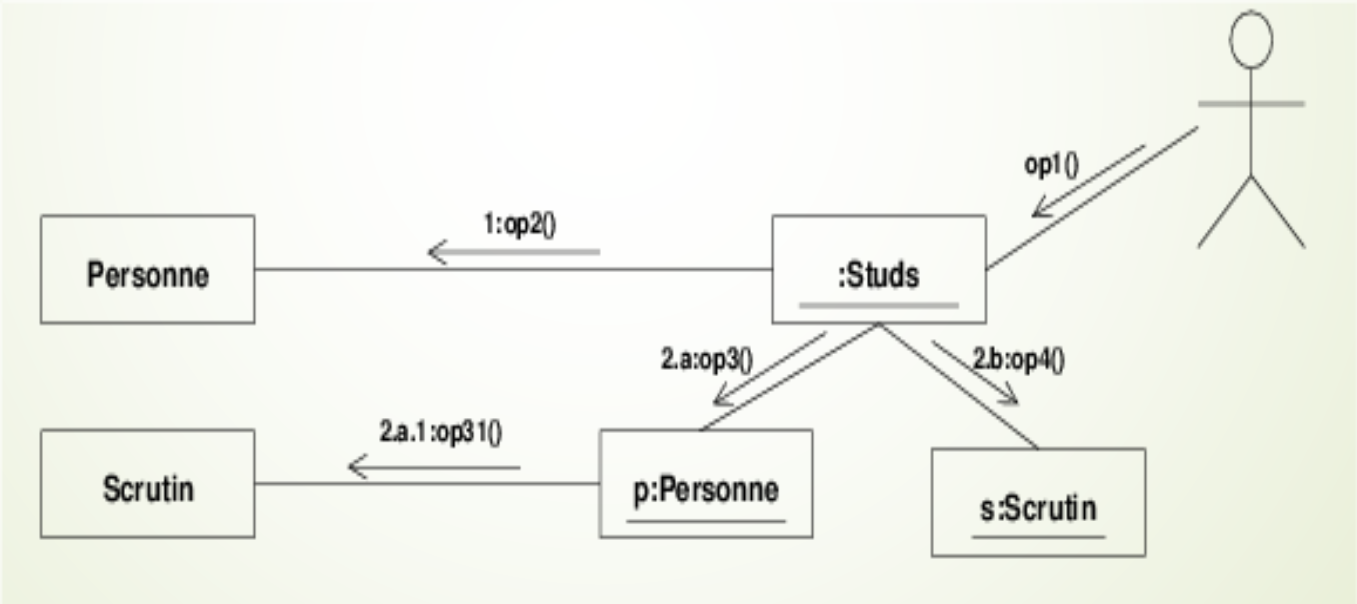
- La réaction à l'appel de l'opération *afficherInfosBulletin* sur chaque instance de la collection provoque les messages emboîtés *4.i.1* et *4.i.2*.

Itérations de messages



Messages concurrents

- Il est possible d'exécuter des actions concurrentes suite à l'envoi en parallèle de plusieurs messages.
- Les lettres sont alors utilisées pour la numérotation : par exemple, le message 2.a est concurrent au message 2.b. La réaction au message 2.a provoque dans l'exemple l'envoi du message 2.a.1.



Choix entre séquences et communications

- **Les plus du diagramme de séquence :**
  - Montrer l'ordre des interactions : le premier objectif du diagramme
  - Montrer les messages asynchrones : différents types de messages
  - Montrer les parties optionnelles, les itérations : concept de fragments
- **Les plus du diagramme de communications :**
  - Montrer les liens entre participants : le premier objectif du diagramme
  - Montrer les participants : l'un des objectifs du diagramme
  - Un peu plus facile à construire et à adapter : numérotation plutôt que séquencement
- **Les deux sont équivalents pour :**
  - Montrer la signature des messages / opérations
  - Montrer le parallélisme
    - Diagramme de séquence : concept de fragment de séquence
    - Diagramme de communications : numérotation des messages

QCM

1. Quelles entités peuvent être dessinées dans un diagramme de communications ?
   a. fragment
   b. message
   c. condition
   d. ligne de vie
   e. lien d'interaction
2. Est-ce que plusieurs messages peuvent transiter sur un même lien d'interaction ?
3. Un diagramme de séquence et un diagramme de communications peuvent-ils contenir des classes ?

Diagramme de machine à état

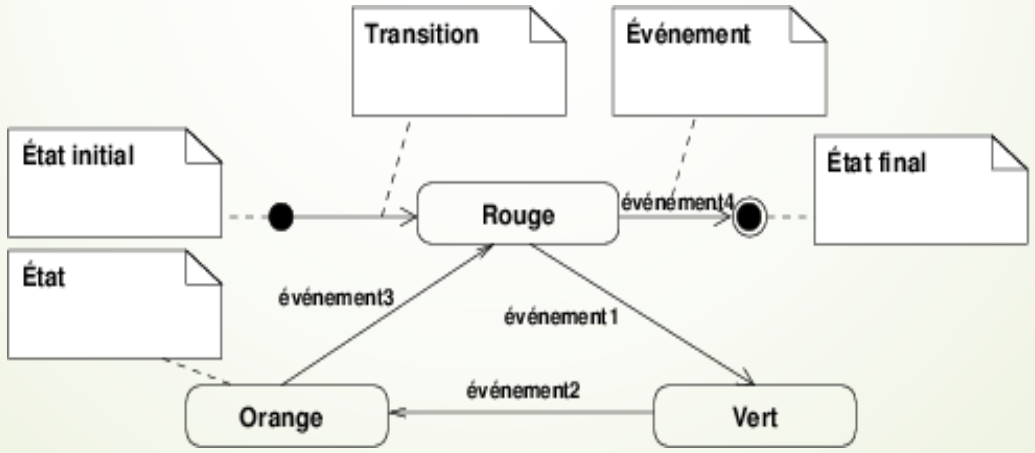
- Modéliser l'état des objets d'une classe
- Types d'état, événements et transitions
- Evénements action et condition d'une transition
- Exemple de machine à état de la classe Scrutin
- Actions liées à un état
- QCM

Modéliser l'état des objets d'une classe

- Certaines classes sont plus complexes et influent de manière plus importante sur le comportement général du système;
- Il est important de décrire les événements provoquant les changements d'états des objets de ces classes
- Les changements d'états sont décrits dans des machines à états;
- C'est d'ailleurs le seul diagramme UML montrant explicitement le cycle de vie complet d'un objet d'une classe.

Types d'état, événements et transitions

- Un objet peut se retrouver dans une situation satisfaisant certaines conditions, ou réalisant certaines actions en réaction à des événements. Cette situation est appelée **un état**. Un état possède un **nom** et se caractérise par une certaine stabilité et une certaine durée.
- Les deux extrémités d'une machine à états sont déterminées par les pseudo-états initial et final.
- Un objet passe dans un état donné par l'événement qui modifie ses attributs, et quitte cet état par un autre événement qui les modifie à nouveau

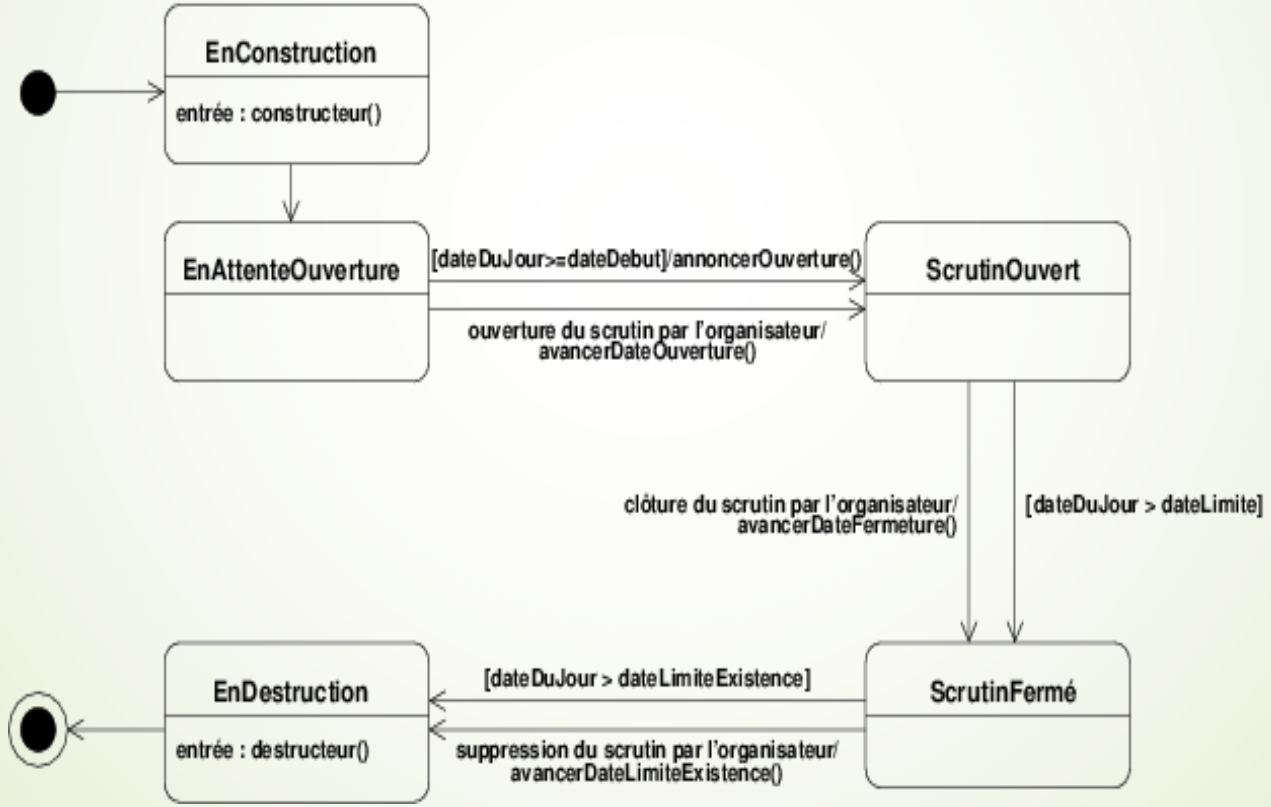


Evénements action et condition d'une transition

- Syntaxe complète
  - `événement[condition]/action`
- Si un événement est indiqué sur une transition et qu'il n'y a pas de condition, alors la transition est franchie dès que l'événement est généré par le système.
- Si une condition est spécifiée, alors la transition est franchie uniquement si la condition est valide.
- Si plusieurs transitions sont franchissables, alors la transition effectivement franchie est choisie aléatoirement.
- Les transitions sans événement sont franchies dès que l'action à l'intérieur de l'état est terminée ; ce type de transition est appelé « **transition implicite** ».

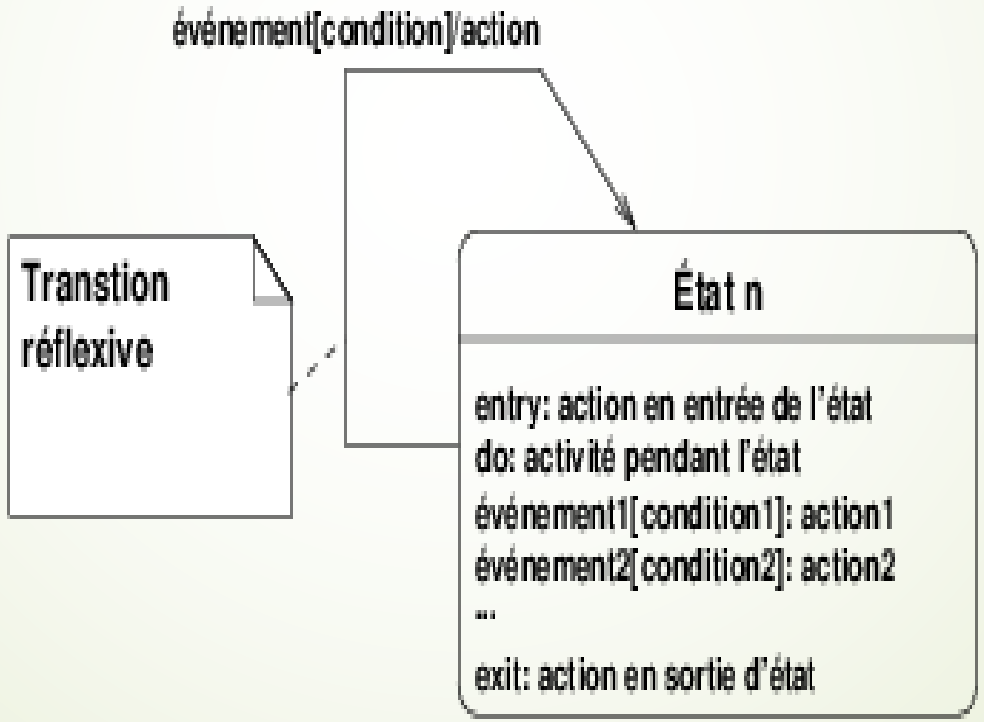
Exemple de machine à état de la classe Scrutin

- Pour l'événement particulier de création de l'objet, nous introduisons l'état *EnConstruction* car nous considérons que les actions liées à l'initialisation de l'objet sont complexes.



Actions liées à un état

- Actions exécutées à l'entrée et à la sortie d'un état
- Action exécutée pendant toute la durée de présence dans l'état
- Action interne déclenchée par un événement



QCM

1. Quelles entités peuvent être dessinées dans un diagramme de machine à états ?
    a. fragment
    b. message
    c. condition
    d. objet
    e. état initial
    f. transition
2. Une transition réflexive peut-elle provoquer la séquence d'actions « exit », « action de la transition », « entry », et « do » ?