Université de Bordeaux

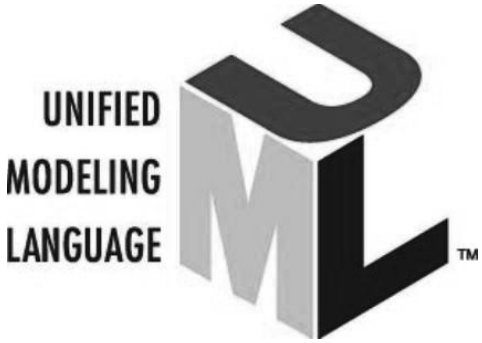
MODELISATION UML

C. Schlick

schlick@u-bordeaux.fr

Modélisation UML

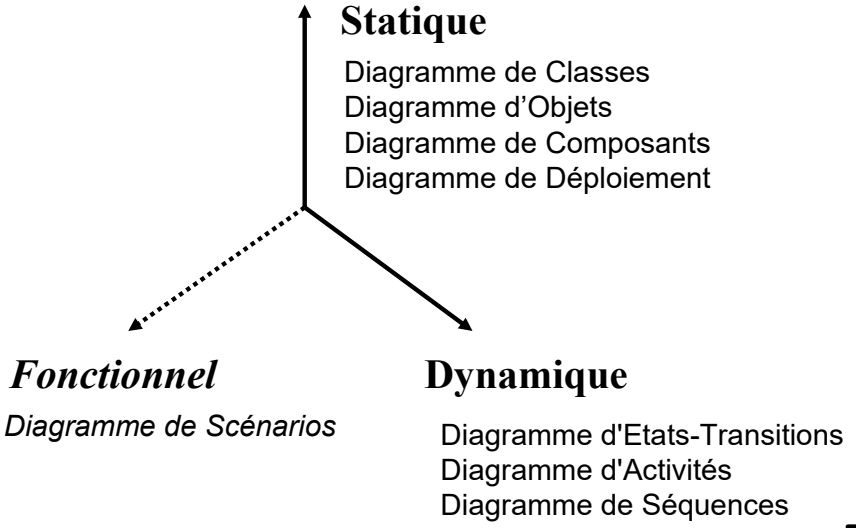
INTRODUCTION



UML ?

- UML est une **notation**, pas une méthode de conception
- UML est un **langage graphique de modélisation objet**
- UML convient à **tous** les langages à paradigme objet
  - Langages compilés : *C++, Objective C, Delphi...*
  - Langages à typage statique : *Java, C#...*
  - Langages à typage dynamique : *Python, Ruby, Perl...*

Axes de Modélisation

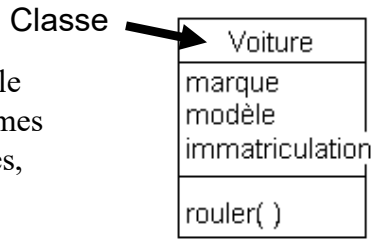


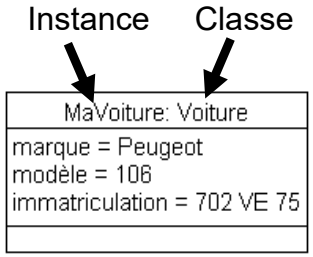
Modélisation UML

AXE STATIQUE

Diagramme de classes

- **Classe**
  - Une description d'un ensemble d'objets qui partagent les mêmes attributs, opérations, méthodes, relations et contraintes
- **Objet**
  - Une entité avec une limite et une identité bien définies qui encapsule un état et un comportement.
  - L'état est défini par des attributs et des relations, et le comportement est défini par des opérations et des méthodes.





Attributs

- **Attribut** = **valeur** stockée dans une classe
- **Syntaxe**

  accès ***nom*** *: type = valeur initiale*
- **Visibilité**

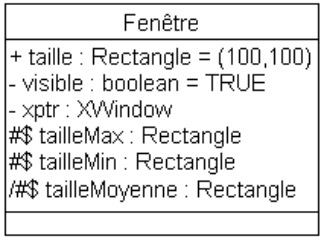
  '+' = public '#' = protégé

  '–' = privé ' ' = package
- **Portée**

  ' ' = portée associée à l'instance (**attribut d'instance**)

  '$' = portée associée à la classe (**attribut de classe**)
- **Dérivation**

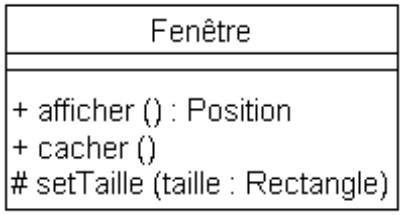
  '/ ' = attribut **dérivé** (i.e. calculé automatiquement) à partir d'un ou plusieurs autres attributs



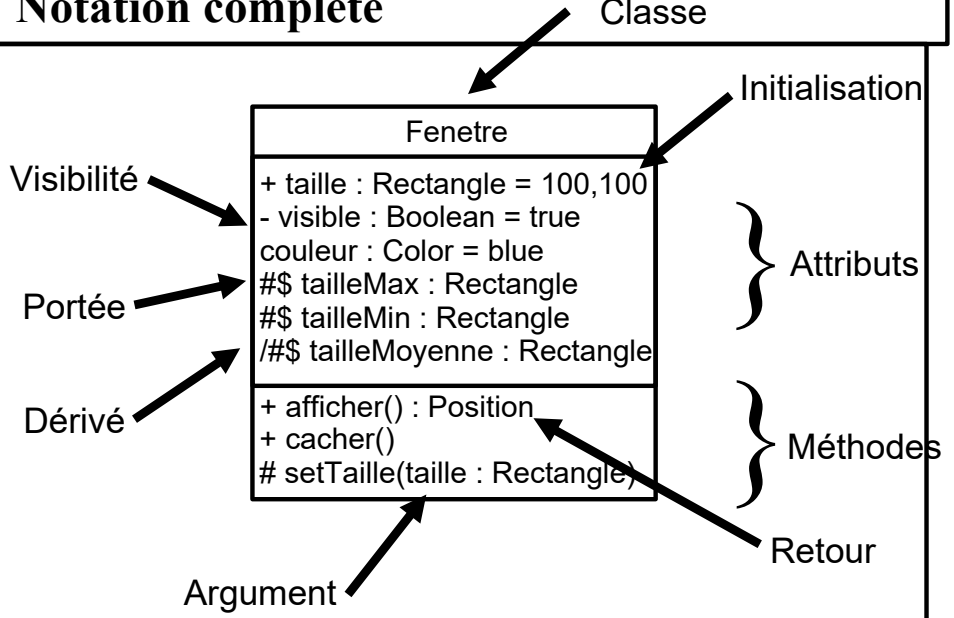
Méthodes

- **Méthode** = **action** associée à une classe
- **Syntaxe**

  accès ***nom*** *(paramètres) : type retour*
- **Visibilité** (même notation que pour les attributs)
- **Portée** (même notation que pour les attributs)



Notation complète



Association

- **Association = Connexion bidirectionnelle** entre classes

  Note : *le cas le plus fréquent est l'**association binaire***
- **Verbe** = caractérise les liens qui existent entre objets
- **Sens =** le sens d'une association est précisé par une **flèche**
- **Rôle** = définit le rôle joué par une classe dans une association
- **Multiplicité** = indique le **nombre d'instances** d'une classe qui peuvent être reliés avec une instance de la classe associée

  1 ou **1..1** : association obligatoire

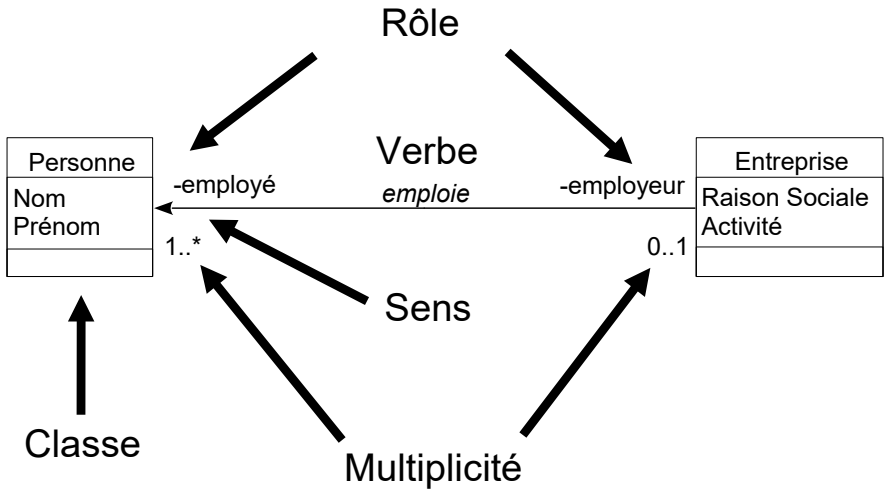
  0..1 : association optionnelle

  0..* ou ***** : nombre quelconque

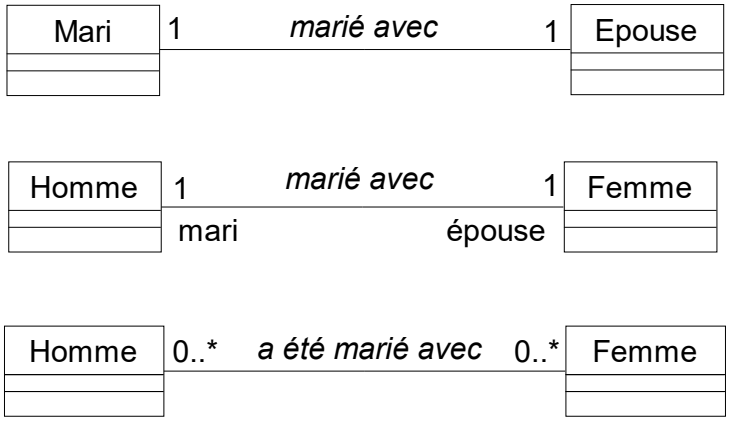
  2..* : au moins 2

  1..5, 10 : entre 1 et 5, ou alors 10

Exemple

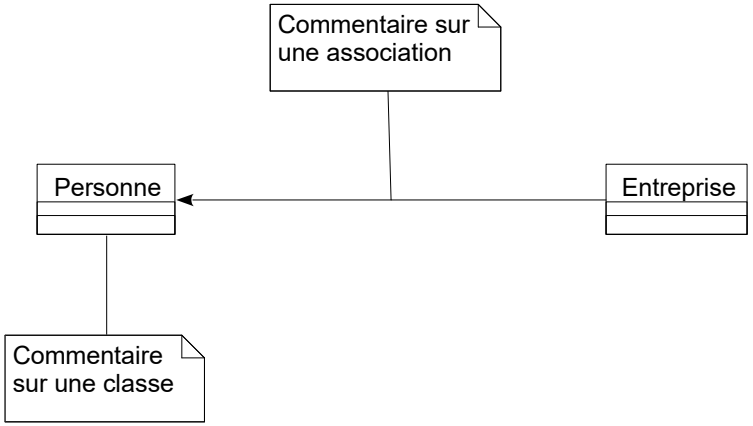


Exemple



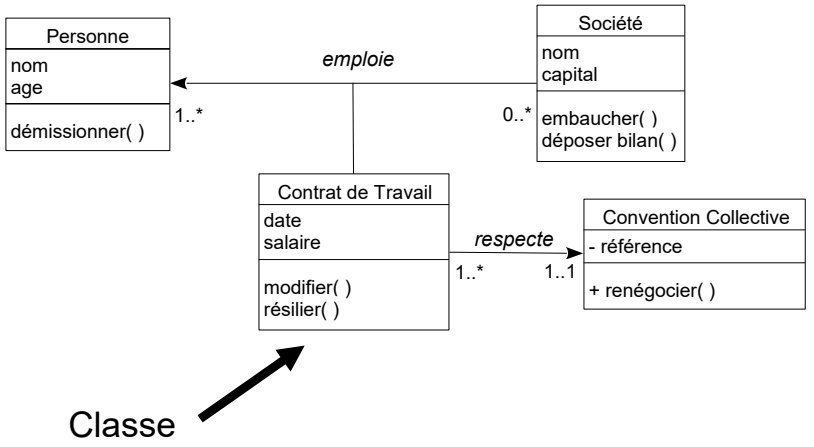
Note

- **Note** = **Commentaire** placé sur un diagramme



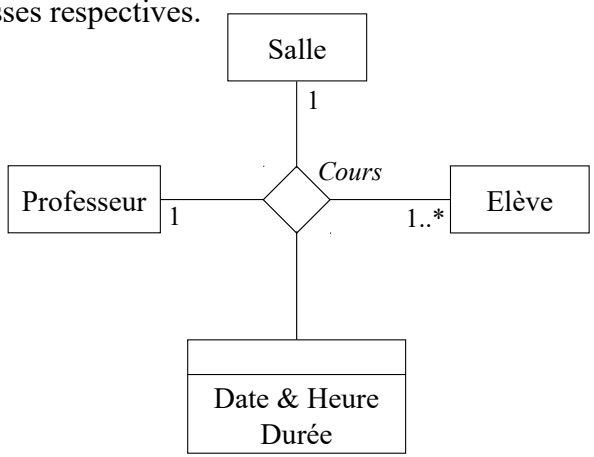
Classe d'Association

- **Classe d'association** = Elément ayant à la fois les propriétés d'une classe et d'une association



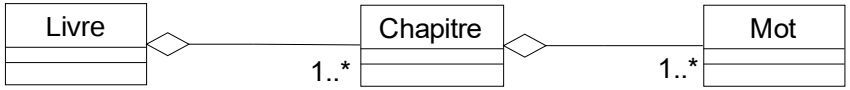
Association n-aire

- **Association n-aire** = Association définie entre ***n*** classes. Chaque instance de l'association est un tuple composé des valeurs des classes respectives.



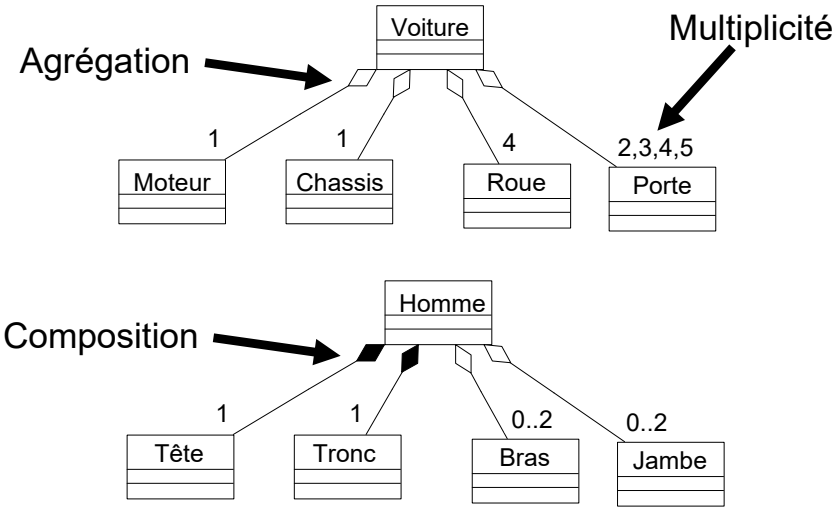
Agrégation et Composition

- **Agrégation** = association particulière correspondant à une **relation d'inclusion** entre deux classes



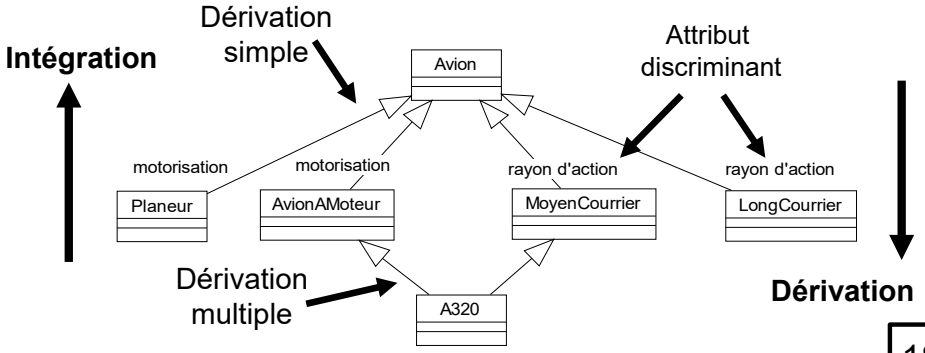
- **Composition** = variante plus forte de l'agrégation où le **cycle de vie** du composant est lié à celui du composite

Exemples



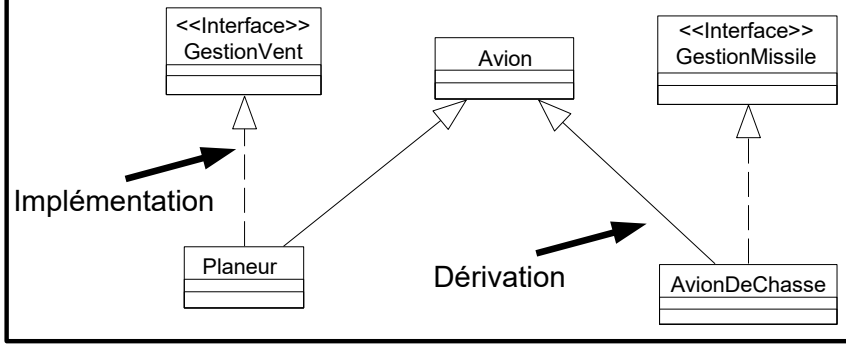
Intégration et Dérivation

- **Intégration** = Création d'une classe plus générale à partir d'un sous-ensemble des attributs et / ou méthodes d'une classe donnée
- **Dérivation** = Création d'une classe plus spécifique à partir d'une classe donnée, en ajoutant de nouveaux attributs et / ou méthodes
- Vocabulaire alternatif : ***Généralisation*** / ***Spécialisation***



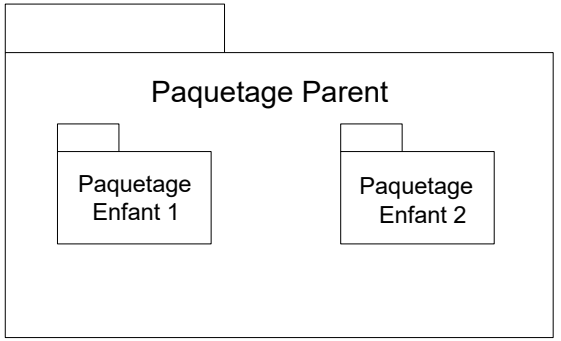
Interface et Implémentation

- **Interface :** Une interface est une classe qui contient uniquement des méthodes mais pas d'attributs. Son rôle est de spécifier un comportement particulier qui pourra être adopté par les classes qui le souhaitent
- **Implémentation :** L'implémentation est une version plus légère de la dérivation puisque la classe concernée ne récupère que des méthodes. Les interfaces permettent notamment de contourner les limitations des langages objets qui n'autorisent pas la dérivation multiple



Structuration en paquetages

- **Paquetage** = **Regroupement** d'un ensemble de classes
- Les paquetages divisent et organisent les classes de la même manière que les dossiers organisent les systèmes de fichiers
- Les paquetages peuvent être **imbriqués** à l'intérieur d'autres paquetages



AXE DYNAMIQUE

Axe dynamique

- L'axe dynamique a pour objectif de définir les séquences **d'événements**, **d'états** et de **réactions** qui doivent survenir dans le système à implémenter
- L'axe dynamique est très fortement lié à l'axe statique, et décrit toutes les caractéristiques du système qui prennent en compte le **temps**, le **séquencement des opérations** et les **interactions** entre objets
- L'axe dynamique inclut deux diagrammes fondamentaux :
  - **Diagramme d'Etats-Transitions**
  - **Diagramme de Séquences**

Diagramme Etats-Transitions *(ou Automate)*

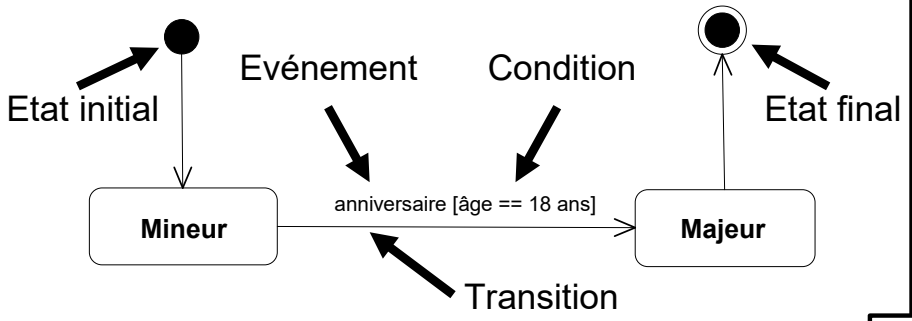
- Il permet de décrire l'**évolution au cours du temps** d'un objet en réponse aux interactions avec d'autres objets
- Il est forcément associé à **une classe**, mais toutes les classes n'en ont pas besoin
- Il se défini comme un graphe orienté, dont les sommets sont appelés **états** et les arcs (orientés) sont appelés **transitions**

Etats

- A chaque instant, un objet se trouve dans un **état** spécifique :
  - **Etat initial** : état d'un objet juste après sa création
    Note : *un objet ne peux avoir qu'un seul état initial*
  - **Etat intermédiaire** : un objet peut passer par plusieurs états intermédiaire au cours de son cycle de vie
  - **Etat final** : état d'un objet juste avant sa destruction
    Note : *un objet peut avoir plusieurs états finaux, tandis que les objets dits éternels (rares) n'ont pas d'état final*
- **Notation :**

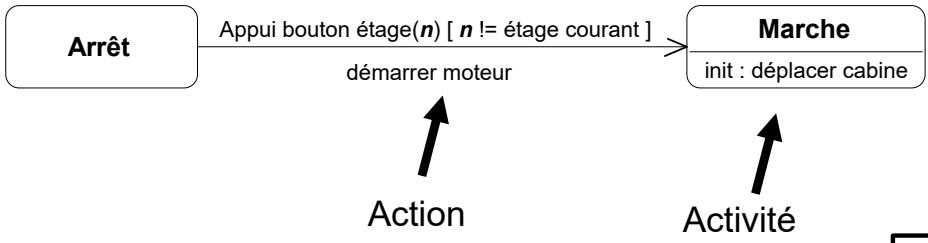
Transition, Condition

- **Transition** : **changement d'état** (entre un état de départ et un état d'arrivée) qui se produit à l'apparition d'un **événement** particulier
- **Condition** : **prédicat** devant être vérifié pour effectuer la transition

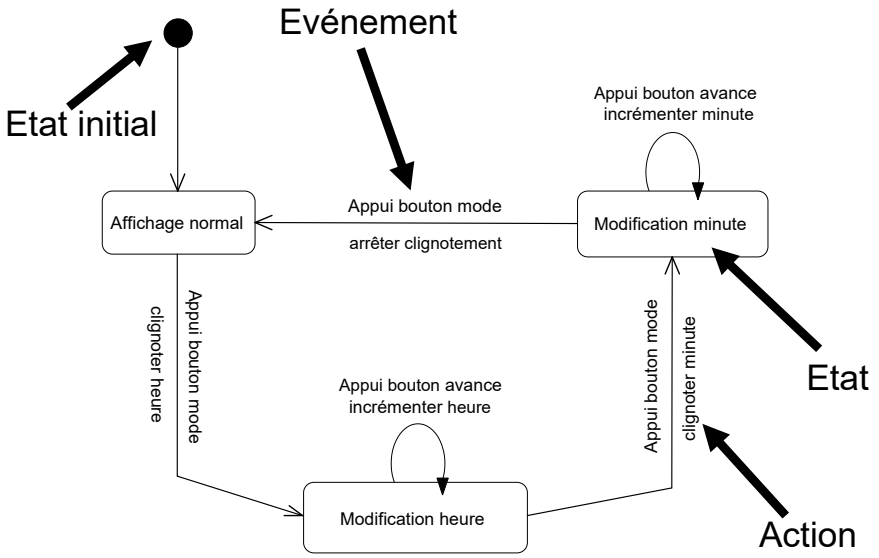


Action, Activité

- **Action** : opération (généralement) **courte et non interruptible**, déclenchée par une transition
- **Activité** : opération (généralement) **longue et interruptible** qui dure tant que l'objet se trouve dans l'état associé à cette activité
  - **init** : action exécutée chaque fois que l'on **arrive** dans l'état
  - **exit** : action exécutée chaque fois que l'on **sort** de l'état

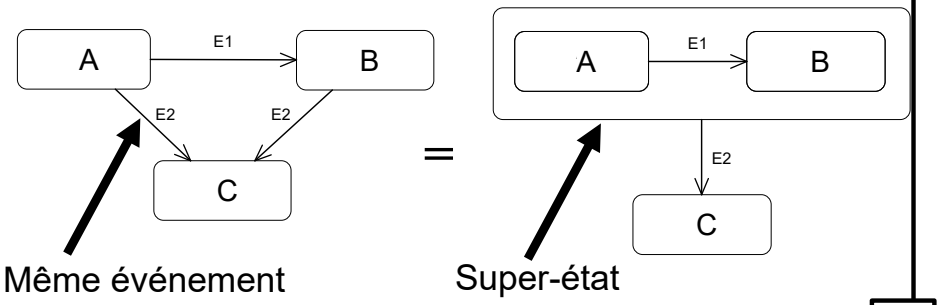


Exemple : *montre à affichage numérique*

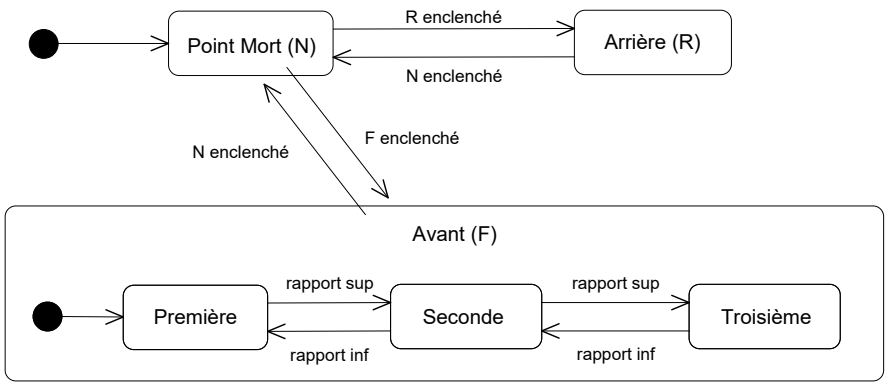


Arborescence d'états

- Dans le cas d'un comportement dynamique complexe, les diagrammes d'états peuvent devenir rapidement illisibles
- Pour éviter ce problème, on peut définir les diagrammes d'états sous forme d'une structure arborescente où plusieurs états sont combinés pour former des **super-états**, afin de permettre une mise en facteur des transitions possibles



Exemple : *boîte de vitesse mécanique*



Historique

- Par défaut, un automate n'a **pas de mémoire**
- La notation **H** offre un **mécanisme pour mémoriser** le dernier sous-état qui l'englobe
- **Exemple** : cycle de lavage d'un lave-vaisselle

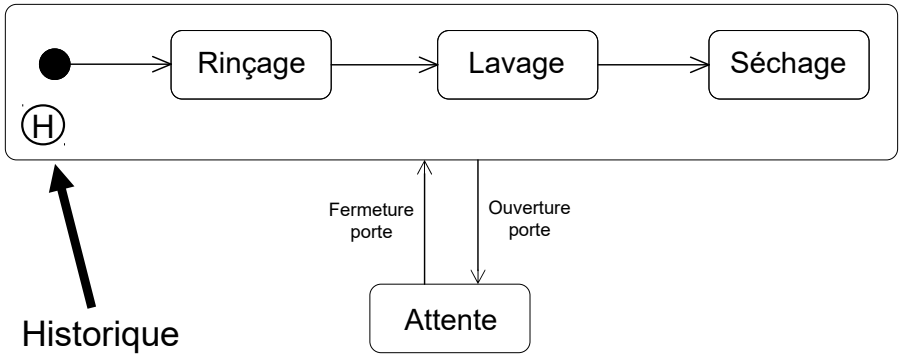
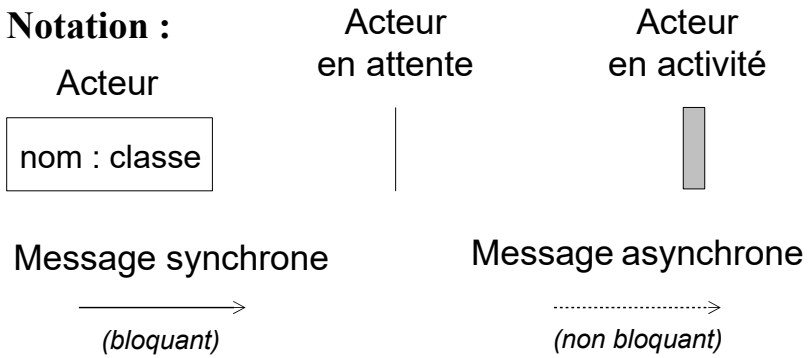


Diagramme de Séquence *(ou **Scénario**)*

- Il permet de définir les **relations dynamiques** entre les objets
- L'axe temporel est placé en vertical (de haut en bas), alors que l'horizontale correspond aux échanges de messages entre objets
- Chaque diagramme de séquence correspond généralement à l'un des **cas d'utilisation** (***use case***) définis dans le cahier des charges
- **Notation :**



Exemple : *appel téléphonique*