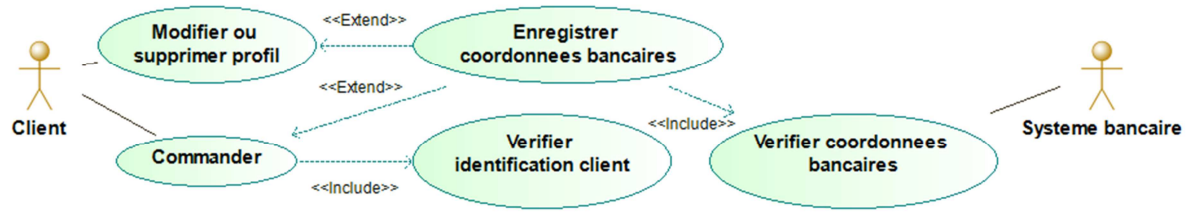
# De la conception UML vers son implémentation JAVA

## Les différents cas inclus ou étendus.

### *Diagramme de cas étudier*




*Cas « vérifier identification client » ou « vérifier coordonnées bancaires ».*

Dans ces deux cas il n'y a pas d'interaction avec un acteur « humain ».

La méthode *verifierIdentificationClient*, point d'entrée du cas « vérifier identification Client », va regarder si l'attribut connecte de l'entité client (correspondant au numéro du client fourni à l'entrée du cas) est bien à **true**. Mais il n'y aura pas d'interaction avec l'acteur HUMAIN client.

La méthode *verifierCoordonneesBancaires*, point d'entrée du cas « vérifier coordonnées bancaires », va correspondre avec un acteur NON HUMAIN pour savoir si les informations concernant une carte bancaire sont valides. Cet acteur non humain est un logiciel extérieur au logiciel en cours de développement chez burgerResto correspondant à un logiciel traitant les informations bancaires entre des applications et les banques.

Dans ces deux cas les contrôleurs des cas principaux vont échanger des messages avec les cas inclus ou étendus :

- le contrôleur du cas «commander » va appeler la méthode *verifierIdentificationClient* du contrôleur du cas « identification Client »,
- le contrôleur du cas «enregistrer coordonnées bancaires » va appeler la méthode *verifierCoordonneesBancaires* du contrôleur du cas « vérifier coordonnées bancaires »,

*Cas « enregistrer coordonnées bancaires ».*

Lorsqu'un profil est créé il n'a pas de carte bancaire qui lui est associé, seulement un nom, un prénom, un login et un mot de passe.

Si on regarde le diagramme de cas, le client peut enregistrer ses coordonnées bancaires :

- lors du cas « Modifier ou supprimer profil »,
- lors du cas « commander ».

Mais le client ne peut pas le faire indépendamment d'un de ces deux cas.

Par contre, contrairement au cas « vérifier identification », le cas « enregistrer coordonnées bancaires » a besoin de l'acteur « client » afin qu'il entre le numéro et la date d'expiration de la carte bancaire.
Un boundary ne pouvant communiquer qu'avec un contrôleur ou un autre boundary, c'est le boundary du cas principal (« commander » ou « modifier ou supprimer profil ») qui appellera le boundary du cas étendu (ou dans un autre exemple inclus) « enregistrer coordonnees bancaires ».

*Les erreurs à ne pas commettre*

Conceptuellement IL NE FAUT PAS :

- Faire des interactions avec un acteur humain depuis un contrôleur,
- Mettre un boundary comme attribut d'un contrôleur.

*A retenir*

**Un cas inclus ou étendu doit échanger avec un acteur humain => le boundary du cas principal appelle la méthode d'entrée du boundary correspondant au cas inclus ou étendu.**

**Un cas inclus ou étendu ne doit pas échanger avec un acteur humain => le contrôleur du cas principal appellera la méthode d'entrée du contrôleur correspondant au cas inclus ou étendu.**

**En conséquence :**
**Les constructeurs des boundary prennent en paramètres d'entrée :**

- **Le contrôleur du cas,**
- **Les boundary des cas inclus et/ou étendu qui ont une interaction avec l'acteur.**

**Les contrôleurs ne possèdent pas de constructeur sauf si le cas principal contient des cas inclus et/ou étendu qui n'ont pas d'interaction avec l'acteur, le contrôleur du cas principal prend en paramètres d'entrée les contrôleurs des cas inclus et/ou étendu.**