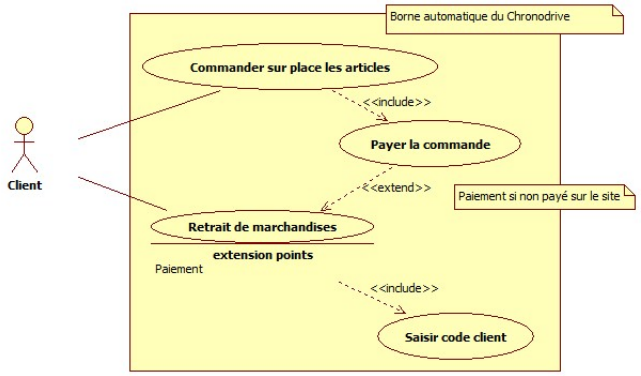
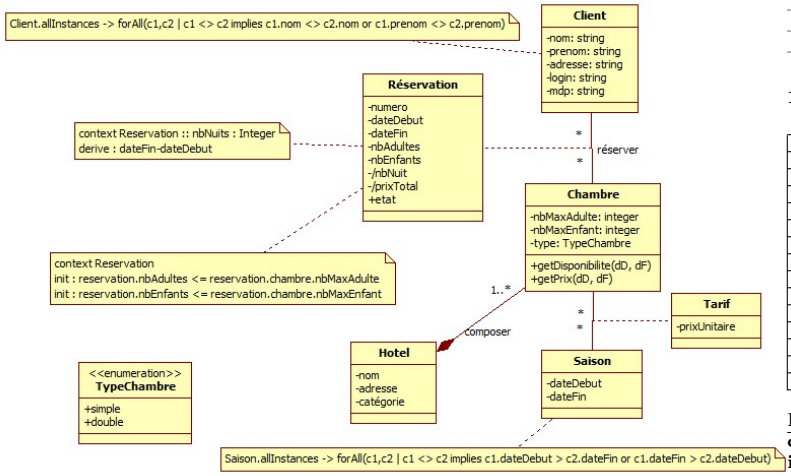
I Definitions

Un attribut de classe est un attribut qui est commun à toutes les instances de la classe. Graphiquement, un attribut de classe est souligné (≡ à static en Java ou C++)
Une opération de classe est une opération qui est appelable sans passer par une instance. Graphiquement, la méthode de classe est soulignée
Une dépendance (dependency) est une relation faible entre classes. Classe A dépend de la classe B si A utilise B dans une de ces opérations
L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet
- Masquer les détails d'implémentation d'un objet (encapsulation)
- Garantir l'intégrité des données contenues dans l'objet

II Diagramme de cas d'utilisation



III Diagramme de classes



IV Contraintes

Contraintes prédéfinies sur les associations :
- {frozen} : fixé lors de la création de l'objet, ne peut pas changer
  - - xMax : Integer{frozen}
- {ordered} : les éléments de la collection sont ordonnés
- {addOnly} : impossible de supprimer un élément
  - # motsClés [*] : String {addOnly}

Qualification (association qualifiée) : restreindre la portée de l'association à quelques attributs ciblés de la classe

Contraintes d'association
- {XOR}
- {subsets <property_name>}
- {redefines <property_name>}
- {union}
- {ordered}
- {bag}
- {sequence} ou {seq}

Contraintes de généralisation
- {complete, disjoint} : pas extensible, pas d'instance commune
- {incomplete, disjoint} : extensible, pas d'instance commune
- {complete, overlapping} : pas extensible, avec des instances communes (instances pouvant appartenir à pls classes filles)
- {incomplete, overlapping} : extensible, avec des instances communes
- Par défaut : {incomplete, disjoint}

V Contraintes OCL

Il existe deux manières pour spécifier le contexte d'une contrainte OCL :

1. En écrivant la contrainte entre accolades dans une note. L'élément pointé par la note est alors le contexte de la contrainte
2. En utilisant le mot-clef context dans un document accompagnant le diagramme

On peut spécifier les pré/post conditions pour les opérations
- pré-conditions doit être vraie avant l'appel de l'opération
- post-conditions doit être vraie après l'appel de l'opération

Dans les post-conditions, on peut utiliser :
- result : indiquer la valeur retournée par l'opération
- @pre : indiquer la valeur d'un attribut avant l'appel de l'opération

Exemple
context Compte : :debiter(montant : Integer)
pre : montant > 0
post : solde = solde@pre - montant

context Compte : :getSolde() : Integer
post : result = solde

Nommer les contraintes :
context class
inv ConstraintName : constraintExpression
Exemple
context Compte
inv soldePositif : self.solde > 0

Commentaire : –

Résultat d'une opération : une expression OCL peut être utilisée pour indiquer le résultat d'une opération de type "query" :
context Compte : :getSolde() : Float
body : solde

Valeurs initiales : une expression OCL peut être utilisée pour indiquer la valeur initiale d'un attribut
context Personne : :marié : Boolean
init : false

Valeurs dérivées : une expression OCL peut être utilisée pour indiquer la valeur dérivée d'un attribut
context Personne : :age : Integer
derive : Date : :current() - dateDeNaissance

Hiérarchie de collections OCL
- Sequence : Elements apparaissent plusieurs fois, ordre spécifique
- Bag : Elements apparaissent plusieurs sans, pas d'ordre spécifique
- Set : Elements apparaissent une seule fois, pas d'ordre spécifique
- OrderSet : Elements apparaissent une seule fois, ordre spécifique

1 Opérations dans les toutes les collections

Operation	Description
size()	The number of elements in the collection
count(object)	The number of occurences of object in the collection
includes(object)	True if the object is an element of the collection
includesAll(collection)	True if all elements of the parameter collection are present
isEmpty()	True if the collection contains no elements
notEmpty()	True if the collection contains one or more elements
iterate(expression)	Expression is evaluated for every element in the collection
sum()	The addition of all elements in the collection
exists(expression)	True if expression is true for at least one element in the collection
forAll(expression)	True if expression is true for all elements
select(expression)	Returns the subset of elements that satisfy the expression
reject(expression)	Returns the subset of elements that do not satisfy the expression
collect(expression)	Collects all of the elements given by expression into a new collection
one(expression)	Returns true if only one element satisfies the expression

Exemple d'opérations
context Banque
inv : **not**(clients → **exists**(age < 18))

context Personne
inv : Personne.allInstances()→**forAll**(p1,p2| p1 <> p2 implies p1.nom <> p2.nom)

context Personne
inv : **if** age < 18 **then** compte→isEmpty()
else compte → notEmpty() **endif**

context Personne
inv : compte→notEmpty() implies banque→notEmpty()

Si une personne possède deux parents, l'un est une femme et l'autre est un homme
Context Person
inv : (self.parent→size()=2) implies ((self.parent→exists(gender=Gender : :male)) AND (self.parent→exists(gender=Gender : :female)))
inv : (self.parent→size()=2) implies (self.parent→one(gender=Gender : :male))

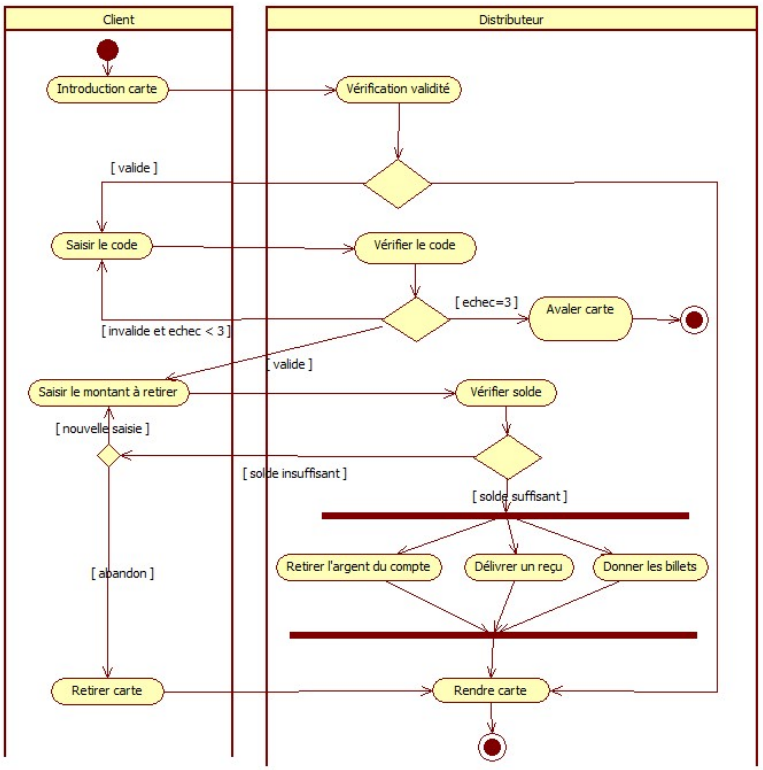
Tous les enfants d'une personne ont bien cette personne comme parent et inversement
Context Person
inv : (self.child→notEmpty()) implies (self.child→forAll(parent→includes(self)))
inv : (self.parent→forAll(child→includes(self)))

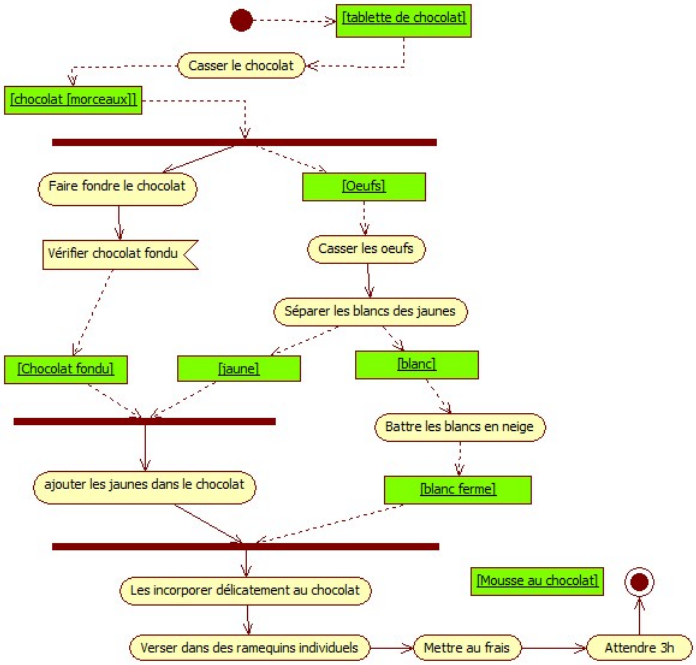
2 Variables

Les variables peuvent être utilisées pour améliorer la compréhension des contraintes complexes
context Personne
inv : let argent = compte.solde→sum() in
age ≥ 18 implies argent > 0

On peut les rendre accessible partout
context Personne
def : argent : Integer = compte.solde->sum()
context Personne
inv : age ≥ 18 implies argent > 0

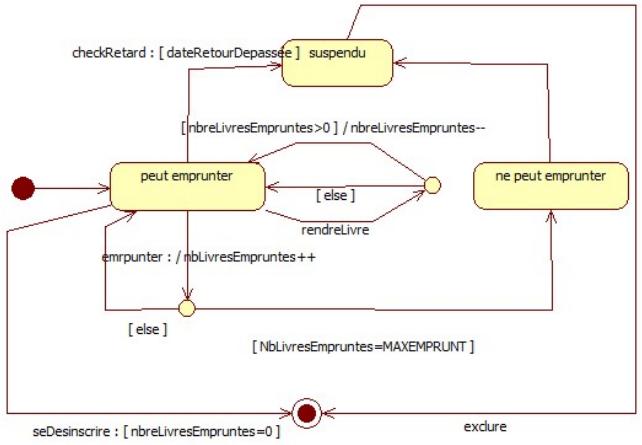
VI Diagramme d'activité

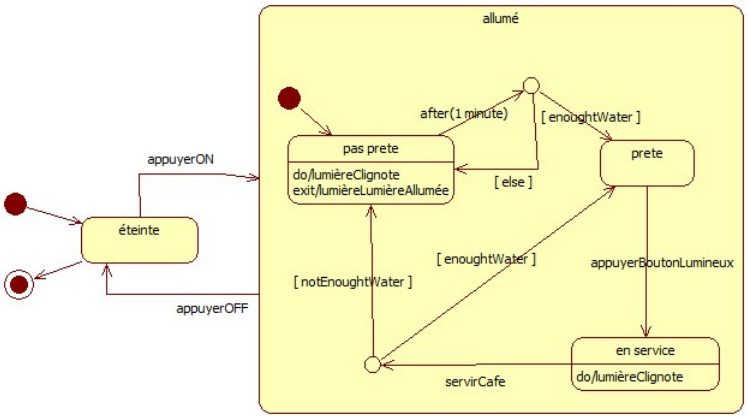




VII Diagramme d'états - transitions

Transition interne : Quatre déclenchements d'action :
- entry/activité : activité est déclenchée à l'entrée dans l'état
- exit/activité : activité est déclenchée à la sortie de l'état
- on événement/activité : activité est déclenchée chaque fois que l'événement se produit
- do/activité : activité liée à l'état quand il y en a une

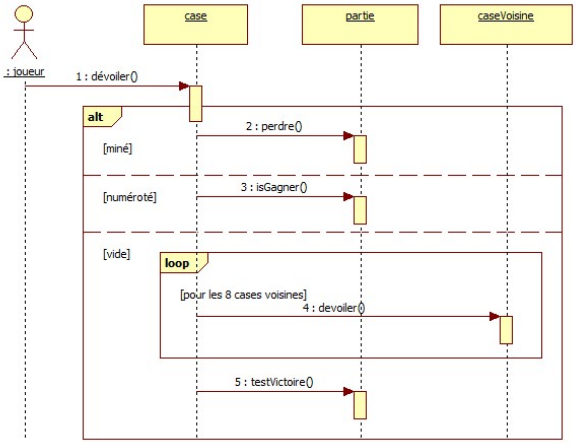


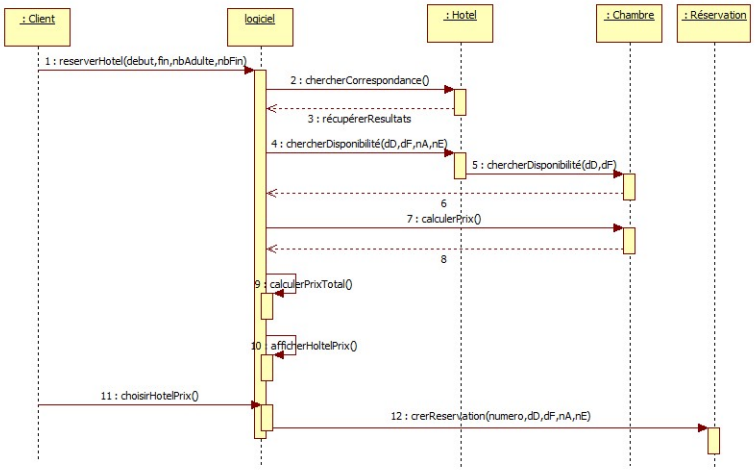


VIII Diagramme de séquence

Les diagrammes de séquence montrent les interactions entre objets selon un point de vue temporel

Opérateurs d'interaction
- Les opérateurs de choix et de boucle : alternative (alt), option (opt), break et loop
- Les opérateurs contrôlant l'envoi en parallèle de messages : parallel (par) et critical region (critical)
- Les opérateurs contrôlant l'envoi de messages : ignore, consider, assertion (assert) et negative (neg)
- Les opérateurs fixant l'ordre d'envoi des messages : weak sequencing (seq) , strict sequencing (strict)





IX Super Etat

Etat contenant des états (notées sous-états) et des transitions ainsi qu'un historique permettant de mémoriser le dernier sous état actif du super-état. Un super-état peut contenir un état de naissance, précisant le premier sous-état actif