Architecture Logicielle Quelques rappels sur la POO et UML(1/53)

F. Nicart

UMU

Architecture Logicielle



Quelques rappels sur la POO et UML

Florent Nicart

Université de Rouen

2016-2017

Architecture Logicielle Quelques rappels sur la POO et UML(2/53)

F. Nicart

UMI

UM

Annotations

Jiagrammes d Jaccoc

Diagrammes d'obje

Diagrammoo a obje

Les membres

Classes abstrait

Eventions

Généricité

Les interface Héritage

Simple

Agrégation

Composition

Dépendanc

Diagrammes de séquence

Acteurs Évènement

Exemples
Objets actif

Contenu

Ce chapitre a pour but de faire un bref rappel sur

- Les concepts de la programmation orientée objet,
- la terminologie employée et
- les notations UML associées.

Architecture Logicielle Quelques rappels sur la POO et UML(3/53)

F. Nicart

UMI

UML

Annotations Diagrammes de

Diagrammes d'obje

Concepts de POO

Les membres

Classes abstrait

Généricité Les interfaces

Les association

Simple Agrégation

Polymorphis

Diagrammes

Acteurs Évènements Exemples

UML : Unified Modeling Language



- UML est un langage de description graphique unifié 1.
- Ce n'est pas une méthode d'analyse.
- Spécifications élaborées par l'Object Management Group, consortium à but non lucratif.
- UML facilite le processus de conception des logiciels. Il contribue également à leur documentation et à faciliter les échanges entre développeurs.
- Convergence de OMT(Rumbaugh), OOD(Booch) et OOSE(Jacobson).

Architecture Logicielle Quelques rappels sur la POO et UML(4/53)

F. Nicart

UIVI

UML

Diagramme

Diagrammes d'obie

Diagrammes d'obje

Los mombros

Classes abstraits

Exceptions

Généricité

Les interface

Les association

Agrégation

Composition

Dépendanc

Diagrammes de séquences

Acteurs Évènements Exemples

Exemples Objets actifs

UML : Unified Modeling Language

- Diagrammes de structures :
 - diagrammes de classes,
 - diagrammes d'objets,
 - diagrammes de composants,
 - diagrammes de déploiement.
- Diagrammes de comportement :
 - diagrammes de cas d'utilisation,
 - diagrammes de séquences
 - diagrammes d'activité, de collaboration, etc. ...
- Diagrammes de gestion de modèles :
 - diagrammes de paquetage,
 - diagrammes de sous-systèmes,
 - diagrammes de modèles.

Architecture Logicielle Quelques rappels sur la POO et UML(5/53)

F. Nicart

UML

UML

Annotations

Diagrammes de

Diagrammes d'obje

Concepts of

Lee membre

Classes abstrait

Exceptions

Les interfaces

Héritage

Les associatio

Simple

Agrégation

Compositio

Dépendance

Dépendance

de séquence

Acteurs

Evenement.

Objets actifs

Démarche de modélisation



Cas d'utilisation



Diagramme de Séquence "système"



Diagramme de classes d'analyse



séquence de conception



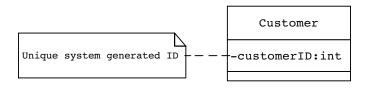
Diagramme de classes de conception

Architecture Logicielle Quelques rappels sur la POO et UML(6/53)

F. Nicart

Annotations

Annotations



- Comme tout langage, UML autorise les commentaires (annotations).
- Une note est représentée par une boite avec le coin supérieur droit plié reliée à l'élément annoté par une ligne en pointillés.
- Commun à tous les types de diagrammes.

Un commentaire doit faire apparaître une information importante qui ne peut être déduite du schéma lui-même. Architecture Logicielle Quelques rappels sur la POO et UML(7/53)

F. Nicart

UMI

Annotation

Diagrammes de

Diagrammes d'obje

Diagrammes a obje

Les membres

Classes abstraite

Exceptions

Les interfaces

Les association

Agrégation

Polymorphism

Discussion

de séquence

Évènements Evemples

Exemples
Objets actifs

Diagrammes de classes et d'objets

Nous allons introduire dès maintenant deux des formalismes graphiques d'UML :

- les diagrammes de classes et d'objets,
- le rôle de chacun, utilisation liées,
- présentation sommaire (du diagramme de classes),
- les notations détaillées seront introduites au fur et à mesure pour illustrer les rappels sur les concepts de POO.

Architecture Logicielle Quelques rappels sur la POO et UML(8/53)

F. Nicart

UMI

LIMI

Annotations Diagrammes de

classes

Diagrammes d'obje

Concepts de

Les membre

Classes shetr

Exceptions

Généricité Les interfaces

Héritage

Les association

Agrégation

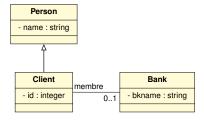
Polymorphis

Dépendanc

Diagrammes de séquences

Acteurs Évènements Exemples Obiets actifs

Diagrammes de classes



- Un diagramme de classes représente la structure statique d'un logiciel,
- il exprime l'ensemble des configurations possibles des instances des entités impliquées.
- Entités : classes, classes abstraites, interfaces.
- Relations: association, composition, agrégation, implémentation, héritage.

Architecture Logicielle Quelques rappels sur la POO et UML(9/53)

F. Nicart

Diagrammes de classes

Représentation d'une classe

Syntaxe:

ClassName

Atttributes

Operations

Une classe est représentée par une boite à trois compartiments contenant:

le nom de la classe.

2 ses attributs (ou variables d'instance),

3 ses méthodes (ou opérations).

Exemple:

Customer

-name: String -userID : Integer

-password: String

+getName(): String

+setName(newName : String)

+getUserId(): Integer

+setUserId(newId: Integer)

Architecture Logicielle Quelques rappels sur la POO et UML(10/53)

F. Nicart

UMI

Diagrammes de

classes

Visibilité des membres

 La visibilité des membres peut être altérée par les modificateurs public, protégé, privé.

Symbol	Scope
+	Public
#	Protected
-	Private

Public Protected Friendly / Package (No Speci-	Access Access	in Other Packages Access No Access No Access	n the same Package Access Access Access	n Other Packages Access Access No Access
<i>fier</i>) Private	No Access	No Access	No Access	No Access

Architecture Logicielle Quelques rappels sur la POO et UML(11/53)

F. Nicart

UMI

UMI

Annotations Diagrammes

Diagrammes d'objets

Concents de

Les membres

Classes abstraite

Exceptions Généricité

Les interfaces Héritage

Les associations

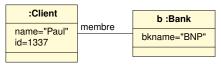
Agrégation Composition

Polymorph

Discussion

de séquence Acteurs Évènements Exemples

Diagrammes d'objets



Example of a bank-client association.

- Un diagramme d'objets (ou d'instances) montre une configuration particulière d'objets (d'instances) et leur liens.
- Correspond à une instance d'un diagramme de classes :
- Entités : objets (instances de classes).
- Relations : liens (correspondant à des associations).
- Il sert à illustrer d'un cas particulier et doit porter un nom explicite.
- Il ne montre que le niveau de détails nécessaires à la compréhension du scénario.

Architecture Logicielle Quelques rappels sur la POO et UML(12/53)

F. Nicart

UM

UM

Diagramme

Diagrammes d'objets

Concepts de

Les membres

Classes abstra

Exceptions

Les interface

Les association

Agrégation

Composition

Dépendanc

Diagrammes de séquences

Acteurs Évènements

Exemples Objets actif:

Représentation d'un objet

Syntaxe:

instanceName :ClassName

attribute=InstanceValue

Notation similaire à la classe :

- le nom est un couple : nom de l'instance (facultatif), nom de la classe(peut être omis),
- les attributs sont donnés lorsqu'ils sont valués,
- dernier compartiment vide.

Architecture Logicielle Quelques rappels sur la POO et UML(13/53)

F. Nicart

UIVII

Annotations

lagrammes d lasses

Diagrammes d'objets

Concepts de

Les membres

Classes abstraite
Exceptions

Les interfaces

Les associations

Agrégation

Polymorph

Diagrammes de séquence

Acteurs Évènements

Exemples
Objets actifs

Diagramme d'objets

Instance de diagramme de classes

Un diagramme d'objets permet de décrire un exemple possible dans un ensemble dénoté par un diagramme de classe. Exemples :

Diagramme de classes :

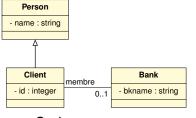
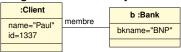


Diagramme de d'objets :



Example of a bank-client association.

Code :

```
// Example of a bank-client association :
Bank b = new Bank("BNP");
return new Client("Paul", 1337, b);
```

Architecture Logicielle Quelques rappels sur la POO et UML(14/53)

F. Nicart

UML

UML

Annotations

Diagrammes

Diagrammes

Concepts de

POO

Les membres

Classes abstra

Exceptions

Généricité

Les interface

Heritage

Les assoc

Simple

Compositio

Polymorphisn

Diagrammes

40.009

Évènements

Evenement:

Objets actifs

Rappels de quelques concepts fondamentaux de la POO

Architecture Logicielle Quelques rappels sur la POO et UML(15/53)

F. Nicart

UML

Annotations
Diagrammes de

Diagrammes d'objets

Les membres

Classes abstra

Généricité Les interface

Héritage

Simple

Agregation Composition

Polymorpr

Diagrammes

de séquence: Acteurs

Évènements Exemples

Classes comme espaces de noms

Exemple en C++:

Définition:

Utilisation:

```
lib.tools.Utils u=new lib.tools.Utils();
// Ou :
using namespace lib.tools;
Utils u=new Utils();
```

En Java, ce rôle est joué par les paquetages et les classes elles-mêmes. Les membres de classes peuvent être :

- des variables, des constantes, des énumérations
- d'instances ou de classe (membres statiques),
- des méthodes, des méthodes statiques

mais aussi des définitions imbriquées

- de classes et
- d'interfaces

Architecture Logicielle Quelques rappels sur la POO et UML(16/53)

F. Nicart

UMI

UML

Annotations

classes

Diagrammes d'obj

Les membres

Olassas abatas

Eveentions

Généricité

Les interfac

Héritage

Simple

Agrégation Composition

Polymorpi

Diagrammes de séquences Acteurs

Acteurs Évènements Exemples

Membres de classe (statiques)

En UML, les membres statiques sont soulignés. En java :

- 1 Les variables de classe (attributs statiques) peuvent être initialisés au chargement du paquetage.
- 2 Une méthode statique est appelée sur la classe et non sur une instance. Elle ne peut accéder qu'aux variables de classe.

```
FileLogger
```

-out : BufferedOutputStream

+getInstance() : FileLogger

```
Date dt=Test.getTestDate();
```

Architecture Logicielle Quelques rappels sur la POO et UML(17/53)

F. Nicart

UMI

UMI

Diagrammo

classes

Diagrammes d'obj

Concepts

Les membres

Classes abstra

Exceptions

Les interfac

Héritage

Ci---I-

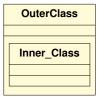
Agrégation

Polymorphis

Diagrammes de séquence

Acteurs
Évènements
Exemples

Définitions imbriquées



UML permet la définition d'une classe à l'intérieur d'une autre mais ce concept n'est pas généralisé :

- Naturel en Java et C#
- seulement struct et enum en C++.

```
-ID: long

+createMemento(): Memento
+setMemento(memento: Memento)
+getLastProcessedID(): long
+process(): boolean

Memento

-lastProcessedID: long
+getID(): long
```

x=**new** DataConverter.Memento();

Imbrication uniquement de l'espace de nom!

Architecture Logicielle Quelques rappels sur la POO et UML(18/53)

F. Nicart

UMI

UMI

Diagrammes

Diagrammes d'obje

Canaanta da

Les membres

Classes abstraites

Classes abs

Exceptions

Généricité

Les interfac

Les association

Simple

Agrégation

Composition

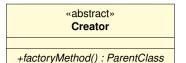
Dépendan

Diagrammes de séquences

Acteurs Évènements Exemples

Classes et méthodes abstraites

- Une méthode sans définition (sans corps) est dite abstraite (mot clé abstract en Java). Sa définition sera fournie dans une classe dérivée.
- Une classe qui possède au moins une méthode abstraite est dite abstraite. (Peut être forcé avec le mot clé abstract). Un telle classe ne peut être instanciée.



En UML, les classes et méthodes abstraites s'écrivent en italique.

La notation « abstract » existe aussi pour les classes.

Architecture Logicielle Quelques rappels sur la POO et UML(19/53)

F. Nicart

UM

LIMI

Annotations

Diagrammes

Diagrammes d'obje

Diagrammes d'obje

Les membres

Classes abstraite Exceptions

Généricité

Les associations

Simple

Agrégation Composition

Dépendan

Diagrammes de séquence

Acteurs Évènements Exemples

Les exceptions

 En Java, throws indique que la fonction définie est susceptible de déclencher une exception.

 L'équivalent en UML est une ligne en pointillés annotée avec throws entre la méthode et la boite de l'exception.

Architecture Logicielle Quelques rappels sur la POO et UML(20/53)

F. Nicart

UML

UMI

Annotations

Diagrammes

Diagrammes

.

Concepts de

P00

Classes abstraite

Exceptions

Généricité

Les interfa

Héritage

Simple

Agrégation

Polymorphisn

Diagrammes

de séquenc

Évènements

Exemples
Objets actifs

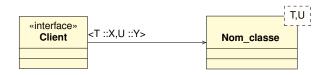
La généricité

 Les types génériques permettent d'employer une classe en typant, à l'utilisation, certains de ses types internes.

```
public class ArrayList <E> extends AbstractList <E> implements List <E> {
    ^^lboolean    ^^ladd(E e) { ... }
    ...
}

// Utilisation :
List <String > sl = new ArrayList < String > ();
sl . add("coucou");
```

 En UML, les paramètres génériques sont indiqués en haut à droite. Ex :



Architecture Logicielle Quelques rappels sur la POO et UML(21/53)

F. Nicart

UMI

UMI

Annotation

Diagrammes

Diagrammes d'obje

Diagrammes d'obje

POO Les membres

Classes abstraite

Généricité

Héritage

Les association

Agrégation Composition

Polymorphisme

Diagrammes de séquences

Acteurs Évènements Exemples Obiets actifs

Signatures et surcharge

Définition (Signature de fonction)

La signature d'une fonction est définie par le nom de la fonction, le nombre et le type de ses paramètres.

Le type de retour et le nom des paramètres sont exclus :

```
// Definissent la meme fonction :
List getList(int i);
List getList(int rank);
Iterator getList(int rank);
```

Surcharge et polymorphisme ad hoc :

```
// Definissent differente fonctions :
List merge(List a, List b);
Vector merge(Vector a, Vector b);
```

Architecture Logicielle Quelques rappels sur la POO et UML(22/53)

F. Nicart

UML

UMI

Diamoration

classes

Diagrammes d'obj

Concepts de

Les membres

Classes abstra

Exceptions

Les interfaces

Les association

Simple

Agrégation Composition

Dépendanc

Diagrammes de séquence:

Acteurs Évènements

Exemples Obiets actif

Les interfaces

 Une interface permet de spécifier un vocabulaire : un ensemble de méthodes.

- Une interface peut être vue comme une classe abstraite sans attribut (cf C++).
- En UML, une interface se représente comme une classe dont le nom est préfixé du mot interface entre chevrons.

Architecture Logicielle Quelques rappels sur la POO et UML(23/53)

F. Nicart

UMI

UM

Annotations Diagramme:

Cidooco

Diagrammes d'obje

Concents de

Les membres

Classes shatrait

Ciasses abstra

Généricité

Les interfaces

Héritage

Ci---I-

Agrégation

Polymorphis

Dépendance

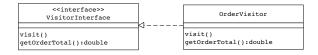
Diagrammes

Acteurs

Évènements

Objets actifs

Réalisation d'interface



- On dit qu'une classe implémente ou réalise une interface lorsqu'elle fournit une implémentation pour (toutes) les méthodes de cette dernière.
- Une classe peut implémenter plusieurs interfaces.
- Une notation abrégée permet de ne pas redéfinir les interfaces connues ou déjà définie par ailleurs :

OrderVisitor
visit() getOrderTotal():double

Architecture Logicielle Quelques rappels sur la POO et UML(24/53)

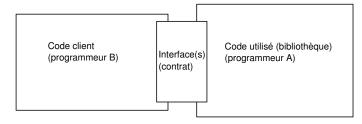
F. Nicart

Les interfaces

Les interfaces

Programmation par contrats

- Les interfaces sont la base de la programmation par contrat.
- Une interface définit un contrat d'utilisation entre un composant et son client (le code qui l'utilise)
- par extension entre les développeurs!



Architecture Logicielle Quelques rappels sur la POO et UML(25/53)

F. Nicart

UMI

UM

Annotations Diagrammes de classes

Diagrammes d'objet

POO '

Les membres

Cidsses abstrar

Exceptions

Les interfaces

Héritage

Les associatio

Agrégation

Composition

Dépendanc

de séquence

Acteurs Évènements

Exemples

Les interfaces

Opérations et sémantique

- Un contrat implique une sémantique que l'on ne retrouve pas forcément dans la définition des méthodes.
- À ce niveau, on doit parler d'opérations plutôt que de méthodes.

Opération *Brancher*:





brancher





Architecture Logicielle Quelques rappels sur la POO et UML(26/53)

F. Nicart

UML

UML

Annotation

Diagramm

Diagrammes

Concepts de

POO

Les membre

Cidoses abstr

Exceptions

Les interfaces

Héritage

Les association

Agrégation

Composition

D.

Dependance

de séquence

Acteurs

Evenements

Obiets actif

Les interfaces

Opérations et sémantique

Mais en y regardant de plus près :









Architecture Logicielle Quelques rappels sur la POO et UML(27/53)

F. Nicart

UMI

UM

Annotation

Diagramme

Disgrammes d'objet

Diagrammes d'objet

Lee membree

Les membre

Classes abstrait

Généricité

Les interiac

1 -- ----

Cimple

Agrégation

Composition

Dénenda

Diagrammes

de séqu

Évènements

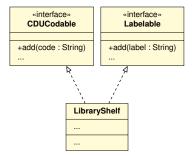
Exemples

Objets acti

Les interfaces

Opérations et sémantique

Exemple en POO : conflit entre CDU² et étiquettes d'étagères.



Quel sens donner à LibraryShelf.add(String)?

2. Codification Décimale Universelle.

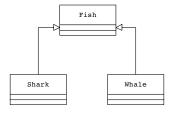
Architecture Logicielle Quelques rappels sur la POO et UML(28/53)

F. Nicart

Héritage

L'héritage

- L'héritage permet de spécialiser le concept représenté par une classe.
- On parle de super-classe (ici Fish) et de sous-classe (ici Shark et Whale).



Ou encore de classe mère ou classe de base et On encore de classe fille ou classe dérivée.

On dit aussi que :

- Shark et Whale sous-classent Fish.
- Shark et Whale spécialisent Fish.
- Fish généralise Shark et Whale.

Ce processus de hiérarchisation s'appelle *classification*.

Architecture Logicielle Quelques rappels sur la POO et UML(29/53)

F. Nicart

UM

UM

Annotations Diagrammes

Diagrammes d'obj

Diagrammes d'obj

Les membres

Classes abstraite

Exceptions

Les interfaces

Héritage

Les associa

Agrégation

Polymorphis

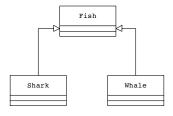
Dependance

Diagrammes de séquence

Acteurs Évènements Exemples

L'héritage

- L'héritage permet de spécialiser le concept représenté par une classe.
- On parle de super-classe (ici Fish) et de sous-classe (ici Shark et Whale).



Ou encore de *classe mère* ou *classe de base* et On encore de *classe fille* ou *classe dérivée*.

On dit aussi que :

- Shark et Whale sous-classent Fish.
- Shark et Whale spécialisent Fish.
- Fish généralise Shark et Whale.

Ce processus de hiérarchisation s'appelle classification.



Réaction du public attendue!

Architecture Logicielle Quelques rappels sur la POO et UML(30/53)

F. Nicart

ПМП

UML

Annotation

Diagramme

Diagrammes d'

Diagrammes a obj

Concepts d

Les membre

Classes abstra

Cánáriaitá

L -- i-t--f---

Héritage

Les associati

Les associati

Agrégation

Composition

Dépendan

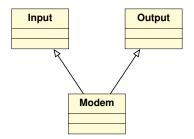
Diagrammes

Acteurs

Évènements

Exemples Objets setif

L'héritage Simple ou multiple



- Certains langages supportent l'héritage (classification) multiple: C++, Eiffel, Python, ...
- Java ne supporte que l'héritage simple :

```
class Shark extends Fish {
...
}
```

Architecture Logicielle Quelques rappels sur la POO et UML(31/53)

F. Nicart

UMI

UMI

Annotations Diagrammes

classes

Diagrammes d'obje

POO

Les membres

Ciasses abstrai

Généricité

Les interface

Héritage

Les associati

Agrégation

Composition

Dépendan

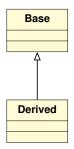
de séquence

Évènements

Objets actifs

Héritage

- La classe dérivée hérite de l'intégralité du contenu de la classe de base.
- L'interface « publique » de la classe de base est incluse dans celle de la classe dérivée.
- Changement de visibilité uniquement possible : protégé
 → public (par redéfinition)



- Ressemble à un copier coller du code de la classe de base...
- En Java : assemblage dynamique à l'exécution
- danger si la classe de base change voir ocp plus tard

Architecture Logicielle Quelques rappels sur la POO et UML(32/53)

F. Nicart

UML

UMI

Annotations

Diagramme

Diagrammes d'obj

, i

POO

Les membres

Classes abstr

Cánáriaitá

Les interface

Héritage

Les associat

Simple

Compositio

Polymorphis

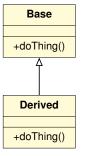
Diagrammes

Acteurs

Évènements

Ohiets actif

Protections contre l'héritage



- Surcharger une méthode dans une classe dérivée s'appelle une redéfinition.
- En Java, on peut empêcher la redéfinition d'une méthode en la déclarant final.
- De la même manière, on peut interdire la dérivation d'une classe entière la déclarant final.

Architecture Logicielle Quelques rappels sur la POO et UML(33/53)

F. Nicart

UML

Annotations
Diagrammes

classes Diagrammes d'obie

Concents de

Les membres

Classes abstraite

Exceptions Généricité

Généricité Les interfaces Héritage

Les associations

Agrégation Composition

Polymorphism Dépendance

de séquence

Acteurs Évènement

Évènements Exemples

Les associations

- Les associations correspondent à la possibilité données aux instances de se faire mutuellement référence.
- C'est ce qui constitue l'architecture d'une application
 OO et que retranscrit le diagramme de classe.
- L'association simple symbolise ces références.
- l'agrégation et la composition sont des raffinements sémantiques de l'association simple (notion de propriété, de composant et de tout)...

Architecture Logicielle Quelques rappels sur la POO et UML(34/53)

F. Nicart

UML

UMI

Annotation

Diagrammes o

Diagrammes d'obie

Diagrammes a objet

Les membres

Classes abstrait

Exceptions

Généricité Les interfaces

Héritage

Les associations

Simple Agrégation

Composition Polymorphis

Dépendance

Diagrammes de séquences

Acteurs Évènements

Exemples Objets actifs

L'association simple

 Il y a association entre une classe A et une classe B si A contient les informations permettant d'atteindre B :

LogAbstraction	bridge	LoggerBridge
	1	+logList()

On parle de navigation de A vers B :

```
LogAbstraction la = ...
LoggerBridge lb=la.getLogBridge (...);
```

 Les relations sont sujettes à multiplicité (ex : collections, références nulles, ...) et peuvent être bidirectionnelles.

Notation	Description
1	No More than One
01	Zero or One
*	Many
0*	Zero or Many
1*	One or Many

Architecture Logicielle Quelques rappels sur la POO et UML(35/53)

F. Nicart

UML

LIM

Annotations

Diagrammes o

Diagrammes d'obje

Diagrammoo a obje

Les membres

Classes abstrait

Exceptions Généricité

Les interfaces

Les associations

Simple Agrégation

Polymorphis

Diagrammes

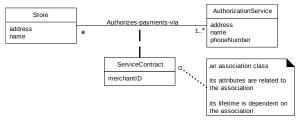
de séquence

Évènements

Exemples Objets actif:

Classes d'association

 Les associations peuvent être représentées par des classes d'association :



- Une ligne en pointillés relie la ligne de la relation à la classe qui la représente.
- La classe de relation permet de munir la relation d'attributs et de méthodes.

Architecture Logicielle Quelques rappels sur la POO et UML(36/53)

F. Nicart

UML

UML

Annotati

Diagrammes

classes

Diagrammes d'obj

Concepts de POO

Les membres

Classes abstrai

Généricité

Les interfaces

Les association

Agrégation

Composition

Dépendanc

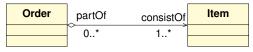
Diagrammes de séquence

Acteurs Évènements

Exemples
Objets actifs

L'agrégation

- Une classe A agrège une classe B lorsque A possède une (ou plusieurs) référence(s) de type B.
- Le losange identifie la classe qui représente l'agrégat (le « tout ») :



- Un composé peut appartenir plusieurs agrégats.
- Ce type de relation comprend aussi des cardinalités.
- Implémentation identique à l'association simple :

```
class Order {
private:
    List<Lineltem> items;
public:
    Order() {
    items=new List<Lineltem>();
    items.add(new Lineltem("Free_Gift", 0));
}
};
```

Architecture Logicielle Quelques rappels sur la POO et UML(37/53)

F. Nicart

UMI

UM

Annotatio

Diagramme

Disgrammes d'obie

Diagrammoo a obje

Concepts de

Les membres

Classes abstra

Exceptions

Lee interface

Les interiaci

Les association

Simple

Composition

Polymorph

Diagrammes

de séquenc

Évènements

Exemples Objets actif

La composition



- La composition est une agrégation avec une sémantique plus forte :
 - le composé ne partage pas ses composants,
 - les cycles de vie des composés et composants sont liés (le composant n'existe pas sans son composé).
- Le losange est noircit,
- la cardinalité du coté composé est toujours au maximum de 1.

Architecture Logicielle Quelques rappels sur la POO et UML(38/53)

F. Nicart

UML

UML

Annotations

classes

Diagrammes

Concepts de

Les membres

Classes abstra

Exceptions

Généricité

Les interface

Héritage

Les association

Agrégation

Composition

Dénendance

Diagrammes de séquence:

Acteurs

Évènements

Objets actifs

La composition

 La composition peut être réalisée naturellement en C++ en incluant les composants « physiquement »dans le composé :

```
1 class A {
private:
    B b;
public:
    A():b() {}
};
```

Architecture Logicielle Quelques rappels sur la POO et UML(39/53)

F. Nicart

UM

UMI

Annotations

classes

Diagrammes d'obje

Concepts de

Les membres

Classes abstraite Exceptions

Généricité Les interfaces

Héritage

Simple

Composition

Dépendance

Diagrammes de séquences

Acteurs Évènements Exemples Obiets actifs

Implémentation de la composition

Quelque soit le langage

La composition est un concept qui nécessite des précautions pour être implémentée :

- s'assurer que B est privée,
- instancier B depuis A (ou copier un référence),
- prendre soin de ne jamais retourner B, une copie si nécessaire.

```
public class A {
    private B b;

public A() { b=new B(); }
    public setB(B p) { b=p.clone(); }

getB() { ^^Ireturn b.clone(); }
}
```

 En résumé, s'assurer que A possède l'unique référence pointant vers B. (Action du ramasse-miette) Architecture Logicielle Quelques rappels sur la POO et UML(40/53)

F. Nicart

UML

UML

Annotations Diagrammes o

Diagrammes d'obje

Diagrammes d'obje

Les membres

Classes abstraite

Exceptions

Généricité Les interfaces

Héritage

Simple

Agrégation

Polymorphisme

Diagrammes de séquences Acteurs

Acteurs Évènements Exemples Obiets actifs

Le Polymorphisme

- Le polymorphisme (par sous-typage³) repose sur les associations.
- Il consiste à distinguer au niveau des références leur type statique de leur type dynamique
- i.e. le type de l'objet pointé n'est pas nécessairement le même que celui utilisé au moment de la déclaration de la référence :

```
Animal ani;
ani=new Penguin();
```

• le sous-typage s'étend aux interfaces. (Polymorphisme au delà de la taxonomie)

^{3.} par opposition au polymorphisme ad hoc vu précédemment.

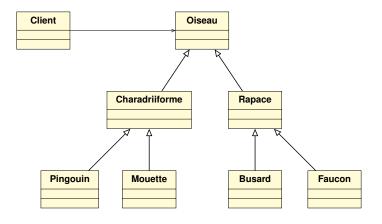
Architecture Logicielle Quelques rappels sur la POO et UML(41/53)

F. Nicart

Héritage

Polymorphisme

Polymorphisme et interfaces



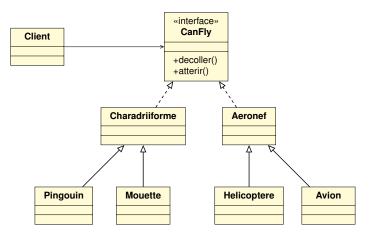
 Si le type de la référence est une classe, alors le type de l'instance appartient à l'arborescence d'héritage de cette classe.

Architecture Logicielle Quelques rappels sur la POO et UML(42/53)

F. Nicart

Polymorphisme

Polymorphisme et interfaces



 Si le type de la référence est une interface, alors le type de l'instance appartient la forêt des arborescences d'héritage des classes implémentant cette interface.

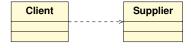
Architecture Logicielle Quelques rappels sur la POO et UML(43/53)

F. Nicart

Dépendance

Les dépendances

 Une dépendance indique qu'une modification de la cible aura un impact sur la source.



Une classe C dépendra d'une classe (ou une interface) S Iorsque

- C envoie un message à S,
- C contient S dans ses données.
- C utilise S comme paramètre d'une méthode.

Architecture Logicielle Quelques rappels sur la POO et UML(44/53)

F. Nicart

UML

UML

Annotation

Diagramme

Diagrammes

Concents de

P00

Les membres

Classes abstr

Généricité

Les interface:

1 -- -----

Simple

Composition

Dépendance

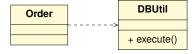
Diagramme

Acteurs

Évènemen

Objets actif

Les dépendances



- Sur cet exemple, une modification de execute() (et donc de l'interface de DBUtil) aura un impact sur le fonctionne de la classe Order qui l'utilise.
- Utile pour indiquer une dépendance en l'absence d'association/héritage/implémentation.

Architecture Logicielle Quelques rappels sur la POO et UML(45/53)

F. Nicart

UMI

Diagrammes de

Héritage

Diagrammes de séquences

Les diagrammes de séquences

Architecture Logicielle Quelques rappels sur la POO et UML(46/53)

F. Nicart

UMI

UML

Annotations Diagrammes

Diagrammes d'obie

Diagrammes d'obje

Les membres

Classes abstraite

Exceptions Généricité

Les interfaces

Les association

Agrégation

Composition

Diagrammo

de séquence

Évènement

Exemples Objets setif

Diagrammes de séquences

- Les diagrammes de séquences complètent les diagrammes de classes en exhibant l'ordre des flux d'évènements entre les différentes acteurs.
- Il permet de modéliser les comportements typiques du système.
- La chronologie se lit de haut en bas.
- Chaque acteur est représenté par une boite étiquetée par : [<nom_du_rôle>] : [<Nom_du_type>],
- sa ligne de vie est représentée en pointillé :



Architecture Logicielle Quelques rappels sur la POO et UML(47/53)

F. Nicart

UML

UM

Annotatio

claceae

Diagrammes d'ob

Concepts de

Les membres

Clacene abeti

Eveentions

Généricité

. . . .

LUG II NOTICE

1 -- -----

01 1

Agrégation

Composition

Diameter

Diagrammes

Évènements

Evenements

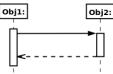
Ohiets actife

Diagrammes de séquences Évènements

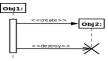
Messages asynchrones :



Messages synchrones :



Messages création/desctruction :



Architecture Logicielle Quelques rappels sur la POO et UML(48/53)

F. Nicart

UML

UMU

Annotatio

Diagrammes de

Diagrammes d'ol

Diagrammes d ob

Concepts of

Les membre

Classes abstra

Exception

Généricité

Les interfac

Héritage

LUS dSSUCIAL

Omple

O-----iti-

Compositio

Dépendan

Diamenana

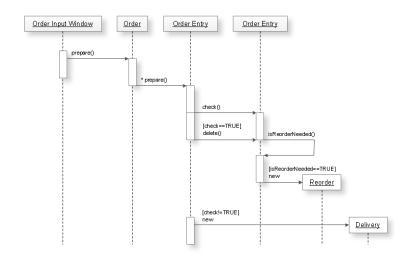
de séquenc

Acteurs

Exemples

Objets actif

Diagrammes de séquences Exemples



Architecture Logicielle Quelques rappels sur la POO et UML(49/53)

F. Nicart

UMI

Diagrammes de

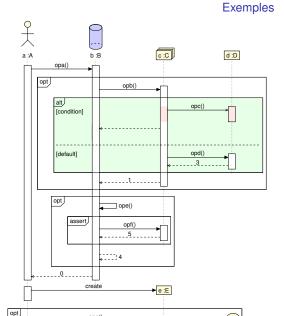
Les interfaces

Héritage

Dépendance

Exemples

Diagrammes de séquences



Architecture Logicielle Quelques rappels sur la POO et UML(50/53)

F. Nicart

UML

UML

Annotatio

Diagramme

Diagrammes d'ob

POO

Les membre

Classes abstra

Généricité

Généricité

Les interfac

Heritage

Les association

Agrégation

Composition

Dolumorphi

de séquence

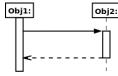
Acteurs

C-----

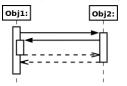
Objets actifs

Diagrammes de séquences Objets actifs

Objets actifs et passifs :



• Exécutions simultanées (ré-entrance) :

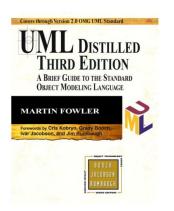


Architecture Logicielle Quelques rappels sur la POO et UML(51/53)

F. Nicart

Objets actifs

Pour aller plus loin...



UML Distilled : A Brief Guide to the Standard Object Modeling Language,

Martin Fowler, Addison Wesley.

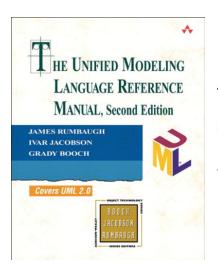
ISBN-13:978-0321193681.

Architecture Logicielle Quelques rappels sur la POO et UML(52/53)

F. Nicart

Objets actifs

Pour aller plus loin...



The Unified Modeling Language Reference Manual, James Rumbaugh, Ivar Jacobson, Grady Booch, Addison-Wesley Professional. ISBN-13:978-0321718952.

Architecture Logicielle Quelques rappels sur la POO et UML(53/53)

F. Nicart

JML

UML

Annotation

claceae

Diagrammes d'obje

Les membres

Classes abstrai

Exceptions

Généricité

Les interfac

Héritage

Les association

Agrégation

Composition

Dánandana

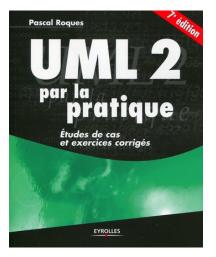
Diagrammes

Acteurs

Évènements

Objets actifs

Pour aller plus loin...



UML2 par la pratique, Pascal Roques, Eyrolles. ISBN-13: 978-2212114805.