



De Programmeur à Développeur

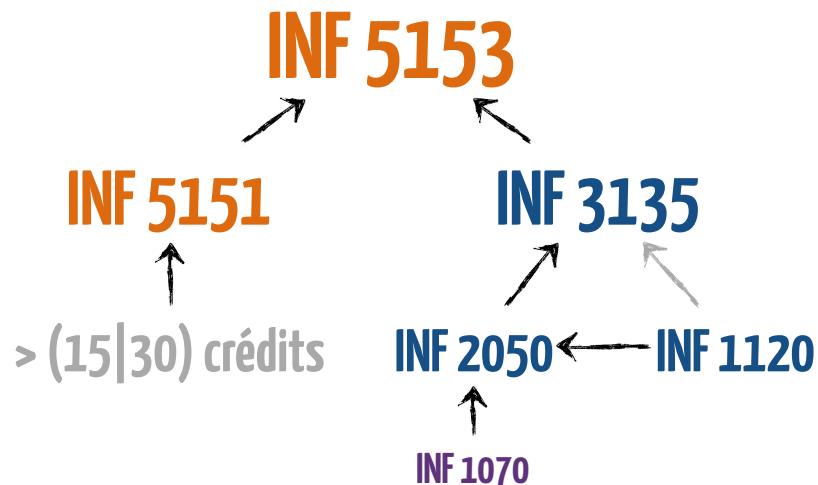
UQÀM | Département d'informatique

Crédit Images: Pixabay & Pexels

Sébastien Mosser
INF 5153 - Cours #1 - A19



Vous êtes ici



Bienvenue dans ce cours de Génie Logiciel !



Michalis Famelis @MFamelis · 12 oct.

Most obvious proof that Devs are better than Wizards: Devs are often asked to perform magic. Wizards are never asked to write software.



Sébastien Mosser

"geek, snowboarder & composition-driven guy"

- 19-...: Professeur, UQAM 
- 12-18: Maître de Conférences, UCA 
- 11-12: Chargé de Recherche, SINTEF 
- 10-11: Postdoc, Inria Lille Nord-Europe
- 07-10: PhD, Composition Logicielle 

Leçon introductory

On va survoler le contenu du cours, et tenter de vous donner une vision globale de ce qui va se passer pendant les 14 prochaines semaines.

Logistique du cours



- | | | |
|----------|------------------------------------|----------|
| 1 | Logistique du cours | 2 |
| 3 | Génie Logiciel ? | 4 |
| 5 | Principes de Génie Logiciel | 6 |
| | Langage de modélisation | |
| | Harry Potter (Kata) | |
| | TP #1: Jeu de Poker | |

Les diapos ne sont pas un polycopié ...

Prenez des notes!

Posez des questions!

Discutez, râlez, échangez, ...

Les diapos sont uniquement un support

Pas de laptop pendant les cours ...



Contact : N'hésitez pas. Vraiment ...



Sébastien Mosser
@petitroll

Parfois on regarde les choses telles qu'elles sont en se demandant pourquoi.
Parfois, on les regarde telles qu'elles pourraient être en se disant pourquoi pas.

✉ Montréal, Québec
✉ mosser.github.io

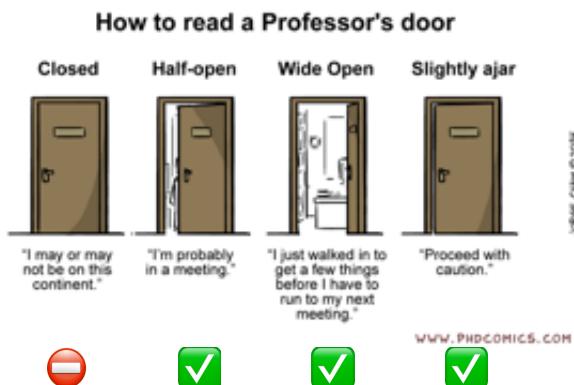
 @petitroll

Courriel : ne pas utiliser

(~150 courriels/jour, souvent beaucoup de latence)



Disponibilités sans rendez-vous



Jeudi, 08h00 - 09h30, PK-4820

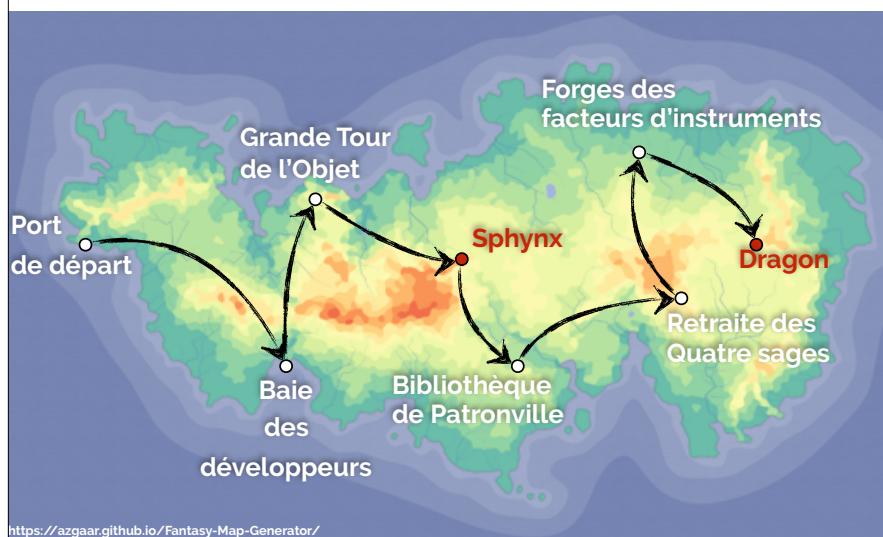
“We are going on an adventure”

Attention, INF5153 (et la conception en général) est un cours difficile, qui demande un travail continu et régulier durant la session.



Qui demande un travail
continu
et régulier
durant la session.

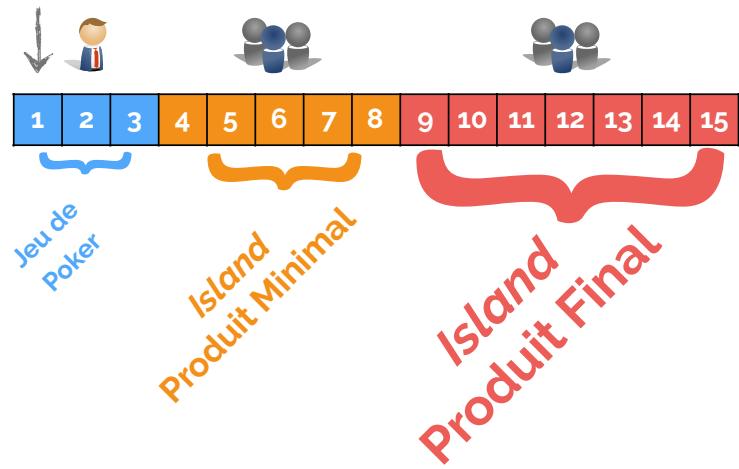
La traversée du Mordor d'INF-5153 ...



<https://azgaard.github.io/Fantasy-Map-Generator/>

#Semaine	Atelier (Mardi, 2h)	Cours (Jeudi, 3h)
#36		Pourquoi concevoir ? De Programmeur à Développeur.
#37	Poker	Encapsulation, Types et Interfaces
#38	Poker	Composition versus Héritage
#39	Island (MVP)	Principes de conception (GRASP, SOLID)
#40	Island (MVP)	Live-coding: Dice Forge
#41	Island (MVP)	Patrons de conception (padawan)
#42	Révisions	Examen intra
#43	Présentations	Présentations
#44	Island (Complet)	Retours sur l'intra & les projets
#45	Island (Complet)	Patrons de conception (chevalier)
#46	Island (Complet)	Patrons de conception (maître)
#47	Island (Complet)	Live-coding: UQAM Unit Test Framework
#48	Island (Complet)	Mesure et visualisation du logiciel (Code as a crime scene)
#49	Présentations	Présentations
#50	Révisions	Examen Final

Organisation des projets de session



Projet #1 : Jeu de Poker

Analyser un code légataire

Ré-usiner un projet

Identifier une conception pertinente

Justifier ses choix de conception



Projet #2 : Exploration d'îles

Développement non-trivial

Capture de drapeau

Exploitation de ressources

Spécification ouverte

Exécution hebdomadaire

Gestion de l'incertitude

Ce projet nécessite un travail en équipe régulier durant la session

Travail en équipe (4 étudiants)

Now it's our time to go out
(My best friend)
And set the world's people free
And we can do it together, you and me
But mostly me
You and me, but mostly me
Are gonna change the world forever
'Cause I can do most everything

- The Book of Mormon (the musical)

"Je travaille mieux tout seul"

"Les autres me ralentissent"

"Je suis meilleur que les autres"

"Je connais pas les autres"



Concevoir ⇒ Discuter

Le travail sur *Island* est ambitieux

Pour collaborer efficacement, une bonne conception est nécessaire.

Bref, cette contrainte n'est pas négociable

Entente d'Évaluation

Date(s)	Travail à rendre	Objectif	Poids
05.09 → 24.09	Projet 1	Conception guidée (individuel)	10%
17.10	Examen intra	Principes fondamentaux	20%
25.09 → 20.10	Projet 2 - PMV	Concevoir et développer un produit minimal & viable	10%
21.10 → 15.12	Projet 2 - Final	Corriger une conception, intégrer des évolutions	20%
12.12	(Examen final)	Principes fondamentaux & avancés	40%

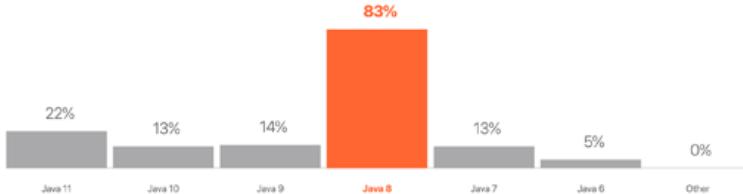
Pour les examens, une feuille **manuscrite** de notes est autorisée, ainsi que la "cheatsheet" de syntaxe UML du cours.

Génie
Logiciel ?



Outils utilisés en support au cours

Which versions of Java do you regularly use?



Which IDE / editor do you use the most for Java development?



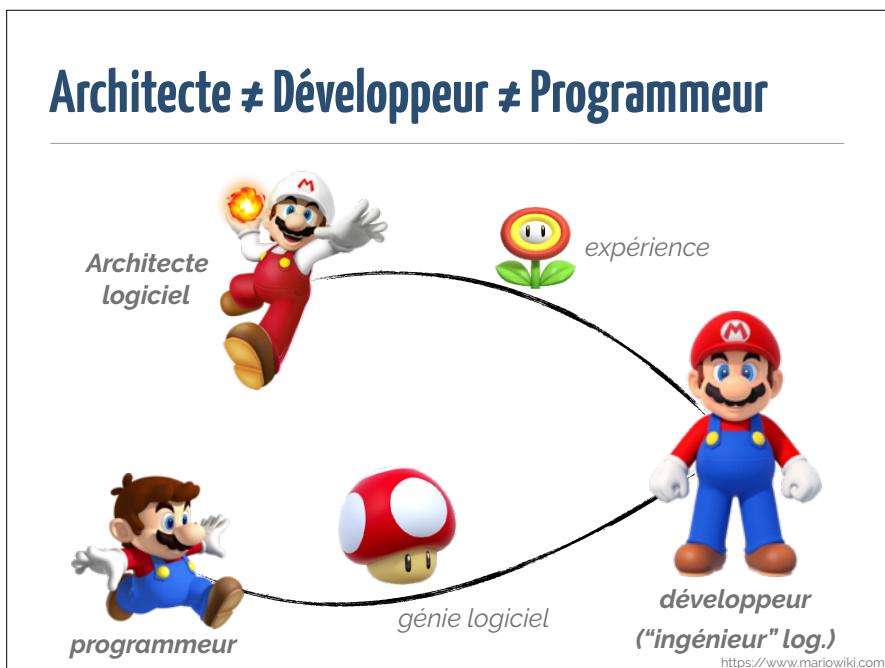
<https://www.jetbrains.com/lp/devcosystem-2019/java/>

Définition du “Génie Logiciel”



Le génie logiciel est une **science de génie industriel** qui étudie les **méthodes de travail** et les **bonnes pratiques** des ingénieurs qui **développent** des logiciels.

“On time, on specs, on budget.”



Définition du “Génie Logiciel”



Le génie logiciel s'intéresse en particulier aux **procédures systématiques** qui permettent d'arriver à ce que des **logiciels de grande taille** correspondent aux **attentes du client**, soient **fiables**, aient un **coût d'entretien réduit** et de **bonnes performances** tout en respectant les **délais** et les **coûts de construction**.

How big is big?



Exemple “simple” : un navigateur internet comme Firefox

How big is big?



How big is big?



Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
C++	5,603,549	1,290,339	18.7%	1,126,900	8,020,788	28.0%
JavaScript	4,881,396	1,584,928	24.5%	1,050,582	7,516,906	26.2%
HTML	3,015,483	143,466	4.5%	335,143	3,494,092	12.2%
C	2,569,343	685,175	21.1%	421,963	3,676,481	12.8%
Rust	1,456,928	268,409	15.6%	149,412	1,874,749	6.5%
Python	970,207	287,116	22.8%	254,540	1,511,863	5.3%
XML	646,542	8,826	1.3%	28,295	683,663	2.4%
Java	350,237	128,549	26.8%	71,775	550,561	1.9%
Assembly	240,815	26,811	10.0%	33,293	300,919	1.1%
CSS	237,465	14,729	5.8%	33,456	285,650	1.0%
Autonconf	115,066	1,592	1.4%	15,927	132,585	0.5%
shell script	90,151	17,783	16.5%	13,838	121,772	0.4%
Postscript	72,532	311	0.4%	0	72,843	0.3%
Objective-C	59,600	9,813	14.1%	13,096	82,509	0.3%
Make	48,043	13,955	22.5%	12,699	74,697	0.3%
OpenGL Shading	32,878	34,825	51.4%	10,433	78,136	0.3%
AMPL	29,585	9,398	24.1%	2,969	41,952	0.1%
Perl	16,771	3,301	16.4%	3,577	23,649	0.1%

<https://www.openhub.net/p/firefox>

Objectif : Se protéger



Travailler la conception permet de faire converger la logique d'affaire, les spécifications, le code écrit, et d'avoir des garanties sur l'architecture du système

Maitriser l'existant pour prédire le coût d'une évolution

Et l'analyse alors ?

Le client exprime ses besoins, exigences et spécifications dans un cahier des charges formalisé qui permet au développeur de livrer un produit conforme en tout point à ce qui est attendu par celui-ci.



Dans la "vraie vie", on est souvent rendu à devoir travailler avec un ensemble de récits utilisateurs mal fichus, et à devoir faire avec.

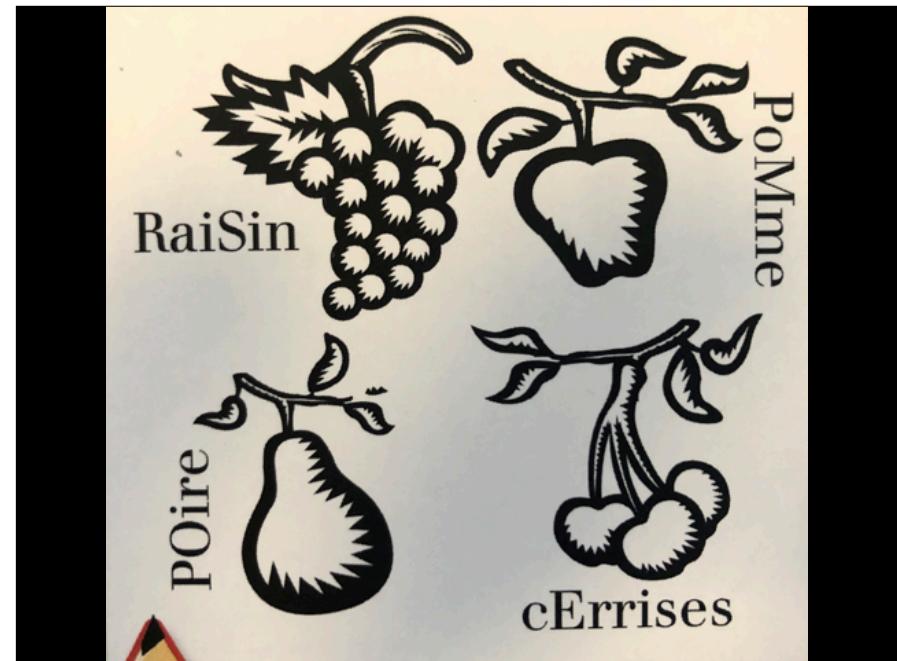
Et c'est le développeur que l'on blâmera en cas de bogue.

Comment faire ? Time to play !



- Un volontaire (au bureau)
- Tous les autres :
 - Prenez une feuille vierge

Le volontaire dispose de trois (3) minutes pour vous décrire le produit que vous devez dessiner. On comparera par la suite vos réalisations et le produit qu'il fallait obtenir.

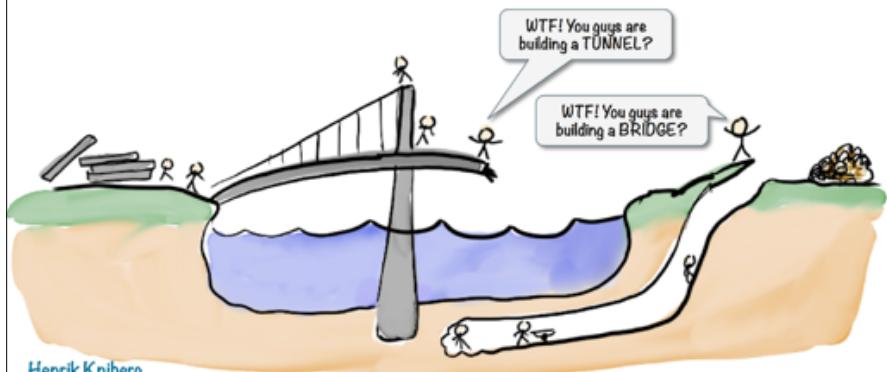


Critères d'acceptations

1. Il y a au moins sept (7) grains de raisin sur la grappe
2. La pomme est située à droite du raisin et au-dessus des cerises
3. On voit exactement quatre (4) feuilles sur la branche supportant la pomme
4. Il y a exactement trois cerises sur la branche
5. Dans le mot "RaiSin", seuls le "R" et le "S" sont en majuscule
6. On voit exactement une feuille sur la grappe de raisin
7. Le mot "POire" est écrit verticalement à gauche de la poire
8. Seules les deux (2) premières lettres du mot "POire" sont en majuscules
9. Le mot "PoMme" est écrit verticalement à droite de la pomme
10. Le mot "cErrises" est mal orthographié

Bref,
Posez des
questions !

Misalignment



La conception est aussi utile
à l'équipe de développement

Les modèles à la rescousse

- L'adage dit "**The code is the truth**"
 - Mais il est difficile de comprendre 20M de lignes de code !
 - 1 seconde par ligne = 231.5 jours de lecture (24/7)
- Les modèles de **conception** vont aider à **maîtriser le produit**
 - Vue "d'en haut" du système
 - Ou au contraire, zoom détaillé sur un points critique
- Utilisation d'une **lingua franca** compréhensible par tous
 - Par exemple le langage UML
 - On doit en respecter l'orthographe et la grammaire

Un exemple ?

```
#include <stdio.h>
#include <string.h>
char *d =
"@n'+#/*{w+/w#cdnr+,{}r/*de}+,*{*,/w{#+,/w#q#+n+,#{l+,/n{n+,/+#n+,/#\n
;#q#n+,/+k#;*+,'r :d'3,{w+K w'K:+}#';dq#'1 \
q#+d'K#/+k#;q#r}eKK#{w'r}eKK{n1}'#/;#q#n{})#}w){}{n1}'#+#n';d}rw' i;# \
{n1!/{n#'; r#'{w'r nc{n1}'#/,{1,+K {rw' iK;{{n1}}/w#q#n'wk nw' \
iwk{K{n1}/w%'1##w# i;:{n1}'/*{q#l'd;r'{nlwb!/*de}'c \
;:{n1'-}rw'/*,##'*#nc,',#nw'/+kd'+e+;#'rdq#w! nr' / ') }+}{rl#'{n' ')# \
}'#)(!!/";
char *s = "!ek;dc i@bK'(q)-[w]*n+r3#1,{:nuwloca-0;m .vpbks,fxntdCeghiry";
int exec(int t, int _, char *a)
{
    if (t < 0) { while (t++ < 0) { a = 1 + index(a, '/'); }; return exec(0, _, a); }
    if (t == 0) { while (*a != '/') { putchar(index(s, *a++)[31]); }; return 0; }
    if (t == 2) { exec(0, 0, d); exec(1, _, 0, d); exec(-13, 0, d); }
    if (t < _) { exec(t+1, _, a); }
    if (t == -27+t, 0, d); if (t == 2 && _ < 13) { return exec(2, _, ""); }
    return 0;
}
int main() { return exec(2, 2, ""); }
```

extrait du livre de Martin Robillard

```
$ gcc -o mystery mystery.c
$ ./mystery
```

```
int exec(int t, int _, char * a) {
    if (t < 0) {
        while (t++ < 0) {
            a = 1 + index(a, '/');
        }
        return exec(0, _, a);
    }
    if (t == 0) {
        while (*a != '/') {
            putchar(index(s, *a++)[31]);
        }
        return 0;
    }
    if (t == 2) {
        exec(0, 0, d);
        exec(1 - _, 0, d);
        exec(-13, 0, d);
    }
    if (t < _) {
        exec(t + 1, _, a);
    }
    exec(-27 + t, 0, d);
    if (t == 2 && _ < 13) {
        return exec(2, _ + 1, "");
    }
    return 0;
}

int main() {
    return exec(2, 2, "");
}
```

```
$ gcc -o mystery mystery.c
$ ./mystery
```

Respect des conventions de codage du langage C

C'est "opti"

Et en plus ça s'exécute comme il faut ... où est le problème ?

Twelve Days of Christmas

On the first day of Christmas my true love gave to me
a partridge in a pear tree.

On the second day of Christmas my true love gave to me
two turtle doves
and a partridge in a pear tree.

On the third day of Christmas my true love gave to me
three french hens, two turtle doves
and a partridge in a pear tree.

...



```
public class Mystery {
```

```
private static String[] gifts = {
    "A partridge in a pear tree.", "Two turtle doves and",
    "Three french hens", "Four calling birds",
    "Five golden rings", "Six geese a-laying",
    "Seven swans a-swimming", "Eight maids a-milking",
    "Nine ladies dancing", "Ten lords a-leaping",
    "Eleven pipers piping", "Twelve drummers drumming",
    "And a partridge in a pear tree.", "Two turtle doves"
};

private static String[] days = {
    "first", "second", "third", "fourth", "fifth", "sixth", "seventh",
    "eighth", "ninth", "tenth", "eleventh", "Twelfth"
};

public static void main(String[] args) {
    for (int i = 0; i < days.length; i++) {
        System.out.printf("%nOn the %s day of Christmas%n", days[i]);
        System.out.println("My true love gave to me:");
        for (int j = i; j >= 0; j--) {
            System.out.println(gifts[i == 11 && j < 2 ? j + 12 : j]);
        }
    }
}
```

Que pensez vous de cette version là ?

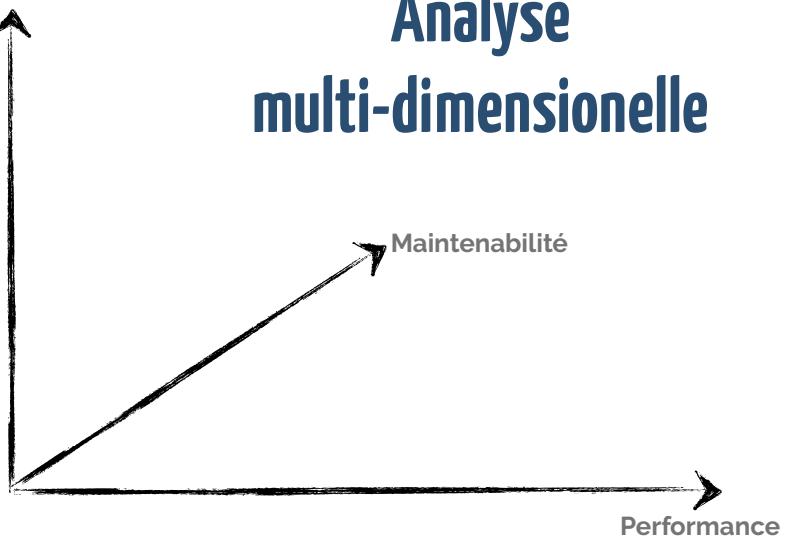
Et de celle-ci ?

```
public static void main(String[] args) {  
    List<Event> events = new ArrayList<>();  
    Event first = new Event(new Gift("a partridge in a pear tree"));  
    events.add(first);  
  
    Event second = first.buildNext(new Gift("two turtle doves"));  
    events.add(second);  
  
    // ...  
    events.forEach(System.out::println);  
}
```

4 version ≠ produisant le même résultat !

Sécurité

Analyse multi-dimensionnelle



VOUS LE VOULEZ COMMENT
VOTRE PROJET ?
(VOUS POUVEZ FAIRE JUSQU'À DEUX CHOIX)

Rapide
Laid
Pas cher
ca sera plus
Cher
ca sera plus
Lent
De bonne
qualité

"Gratuit" n'est pas une option.

De la recherche en Génie logiciel ?



ICSE 2019 MONTREAL

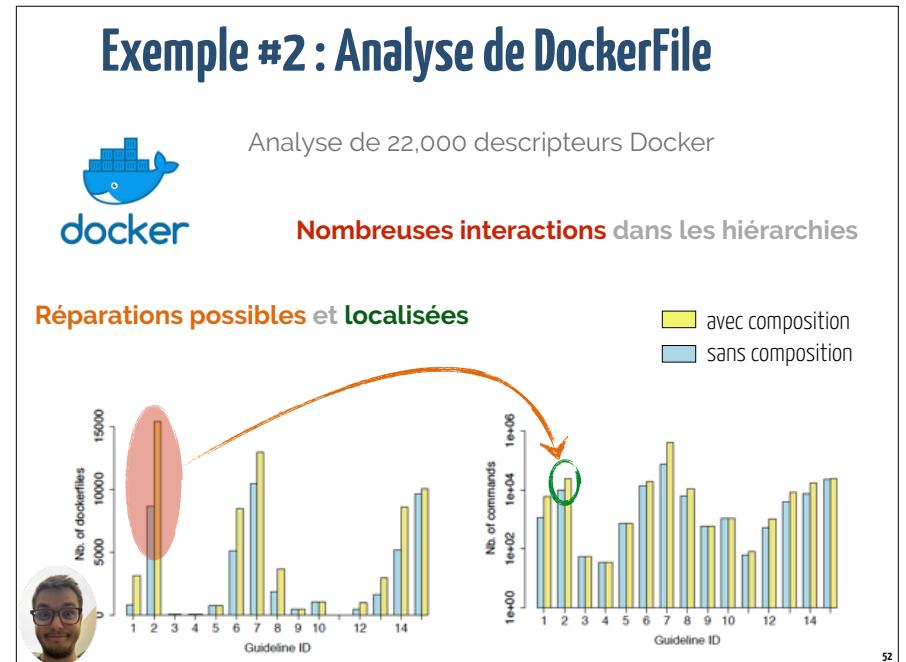
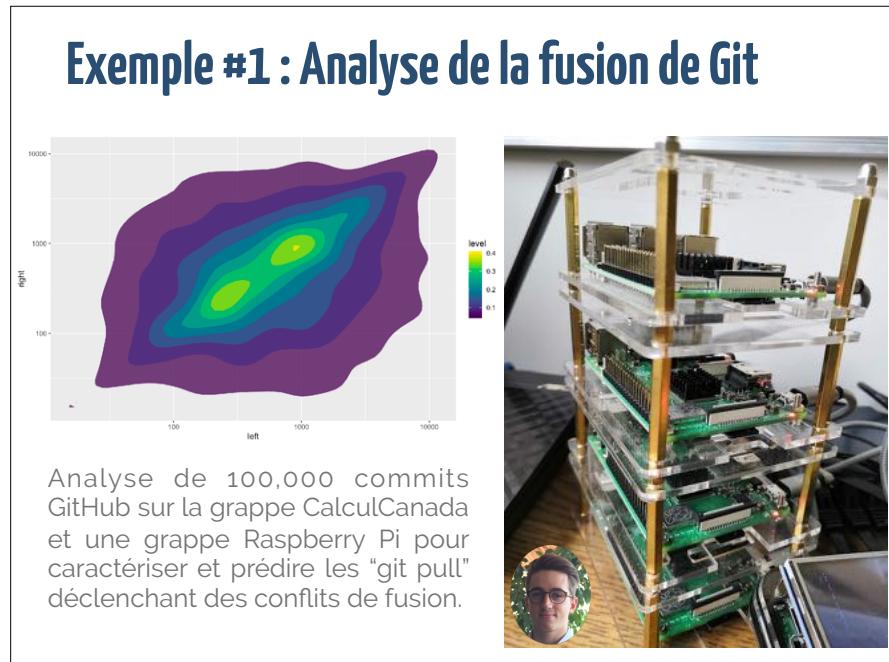
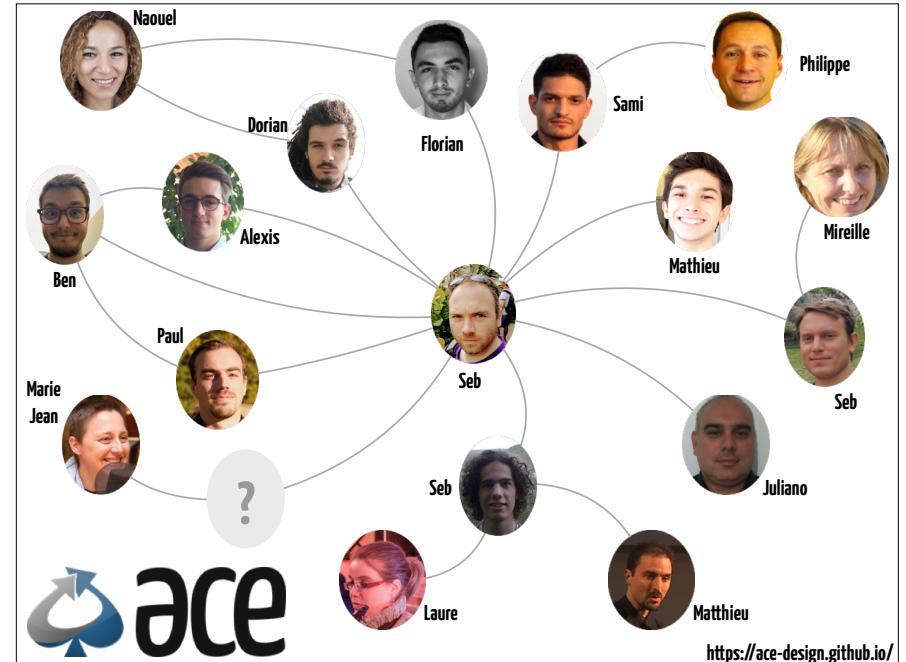
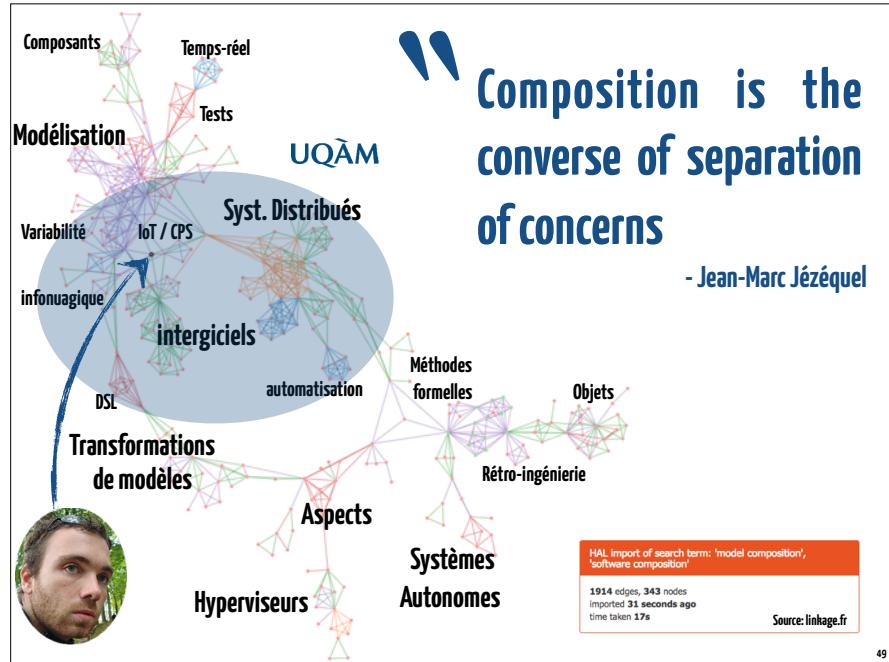
© 2017 Artwork designed by Loogart.com



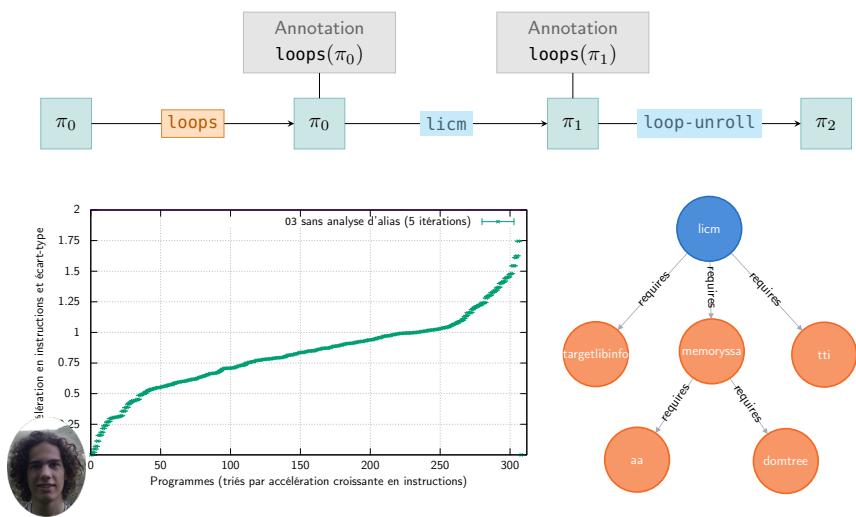
Software Engineering at Montréal

<http://bit.ly/se-mtl>

Prochain le 04.10, 2:30PM, McGill



Exemple #3 : Cartographie de compilateur



Publicité outrancière 😊



Principes de
Génie Logiciel

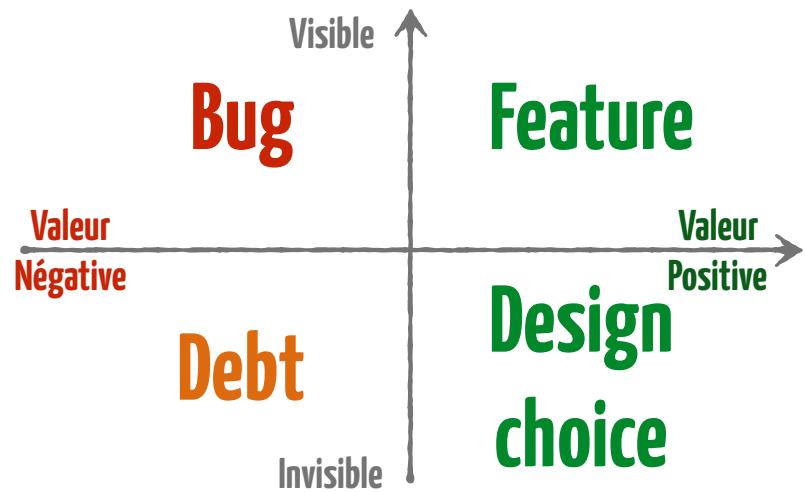


Objectif : Livrer de la valeur

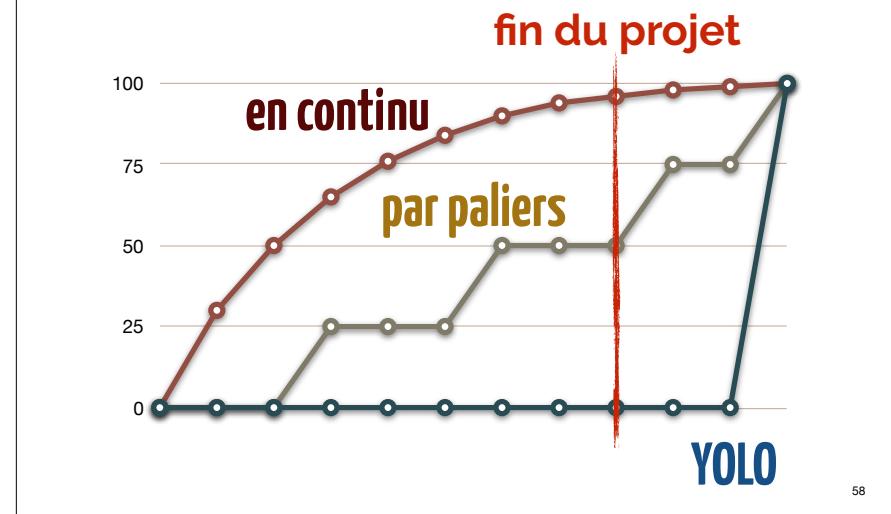
Insiste sur le "**quoi**", pas sur le "**comment**"

Qu'est-ce que je **saurais faire demain** que
je ne **savais pas déjà faire aujourd'hui** ?

Un peu de vocabulaire



Valeur cumulée livrée

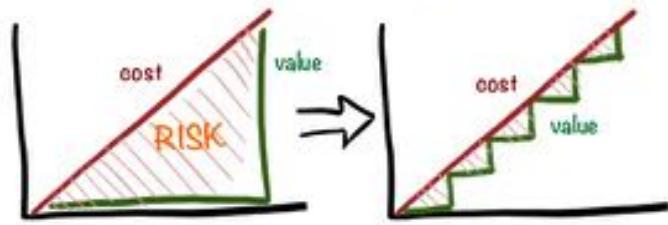


58

Agile = Iterative + Incremental

Don't try to get it all right
from the beginning

Don't build it all at once



Henrik Kniberg

Le risque de livraison est un "simple" calcul d'intégral

Maximize Value, not Output



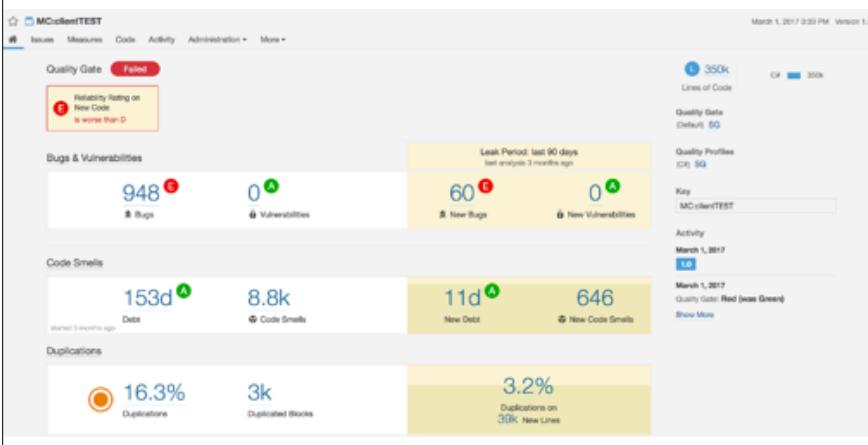
Il est normal de s'endetter au début d'un projet



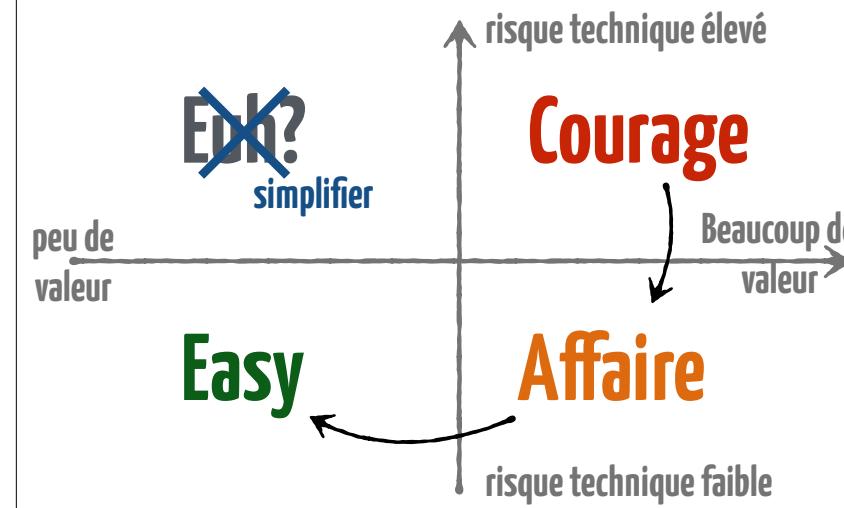
10 cents plutôt qu'un million de \$



Analyse statique de la dette



Par quoi commencer ?



La conception pour estimer

- On peut **exploiter les modèles** de conception pour :
 - Estimer le **risque d'intégration** d'une fonctionnalité
 - **Échanger** avec le client sur **l'impact** de ses demandes
- Cette analyse de risque permet de mieux **anticiper** :
 - Une livraison dans les délais (**on time**);
 - Une livraison dans le respect de la spécification (**on specs**);
 - Une livraison à coûts maîtrisés (**on budget**)

Concevoir aide à Comprendre

Heliocentrism



Geocentrism



S.O.L.I.D : l'essentiel !

- **Single responsibility principle (SRP)** : une classe n'a qu'une seule responsabilité (ou préoccupation).
- **Open/closed principle (OCP)** : une classe doit être ouverte à l'extension (par héritage, par exemple) mais fermé à la modification (attributs privés, par exemple).
- **Liskov substitution principle (LSP)** : les objets d'un programme doivent pouvoir être remplacés par des instances de leurs sous-types sans «casser» le programme.
- **Interface segregation principle (ISP)** : il vaut mieux plusieurs interfaces spécifiques qu'une unique interface générique.
- **Dependency inversion principle (DIP)** : il faut dépendre des abstractions, pas des réalisations concrètes.

Simon Urli

Tests Unitaires



Scott W. Ambler
@scottwambler

[Suivre](#) ▾

A good programmer looks both ways before crossing a one-way street.

[Traduire le Tweet](#)

02:28 - 21 juin 2019

6 Retweets 10 J'aime



0 6 10

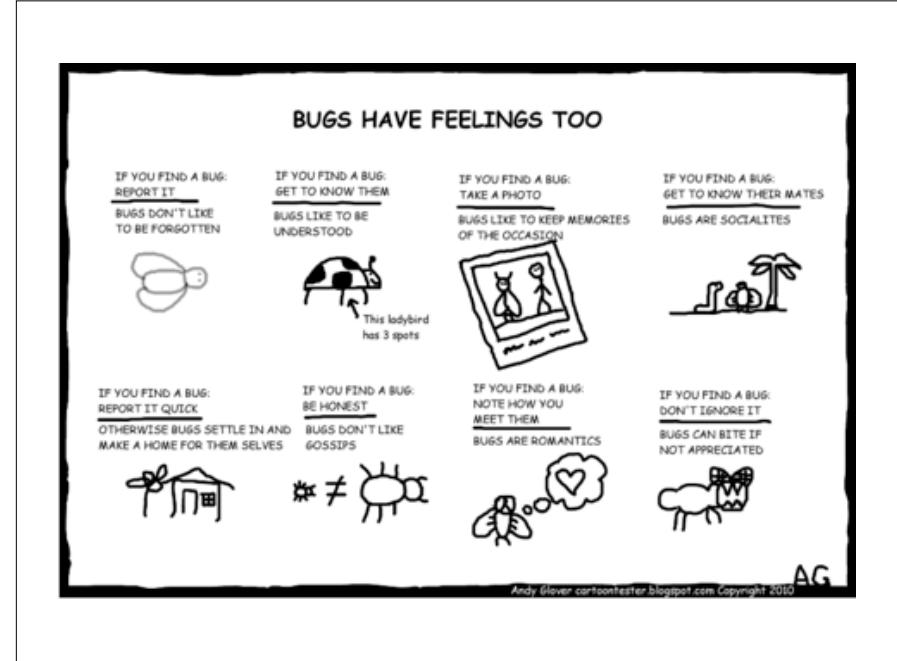
“Du code qui compile pas, ça a jamais tué personne.
Alors que du code qui compile par contre ...”



All code is guilty
until being
proven innocent

Junit 4.12

UQÀM Unit
Test
Framework



Langage de
modélisation
(UML)



Avertissement

INF-5153 N'EST PAS UN COURS D'UML

Même si on utilise intensivement UML

UML n'est qu'un Langage
(c'est même écrit dans le titre)

D'où vient UML ?

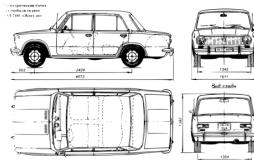
- Dans les années 90 :
 - Pléthore de notations et de méthodes (OMT, Booch, ...)
- Personne n'y comprend plus rien !
 - Besoin d'un langage standard
- Création de l'entreprise Rationale
 - aka "La communauté de l'objet"
- Alliance des auteurs des méthodes existantes

Unified Modeling Language



Pourquoi Modéliser ?

Spécifier la structure et le comportement d'un système



Visualiser un système



Aider à la construction d'un système



Documenter les décisions



P. Collet

5

UML, avec un U comme “Unifié”

depuis 1997

Mélange de plusieurs notations

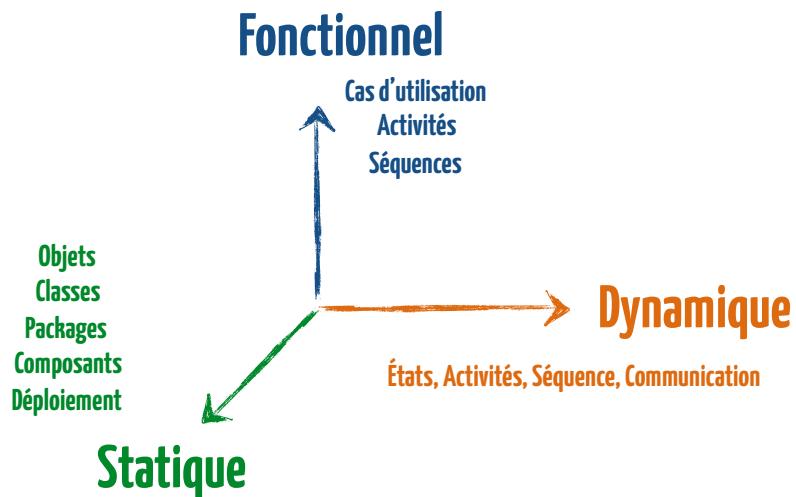


Standardisation (OMG)

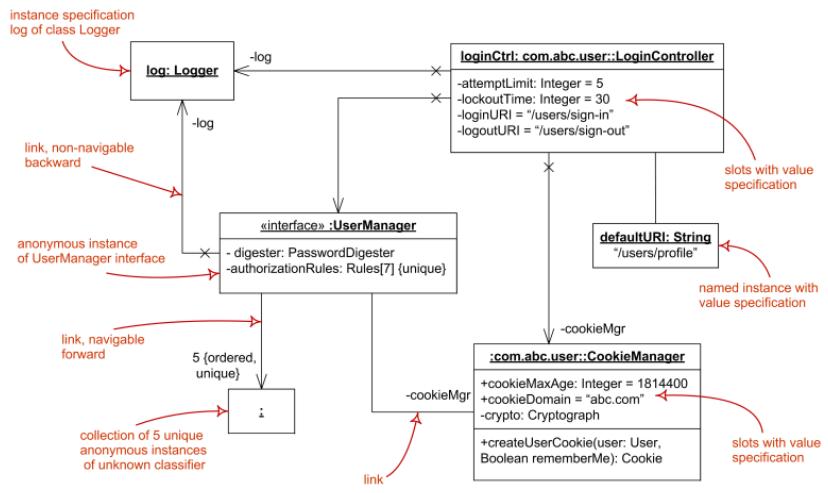
Au pays des objets où s'étendent les ombres
Un Langage pour les gouverner tous
Un Langage pour les trouver
Un Langage pour les amener tous,
Et dans les ténèbres les lier
Au pays des objets où s'étendent les ombres.

Dernière version : UML 2.5.1
(12.17, 800 pages)

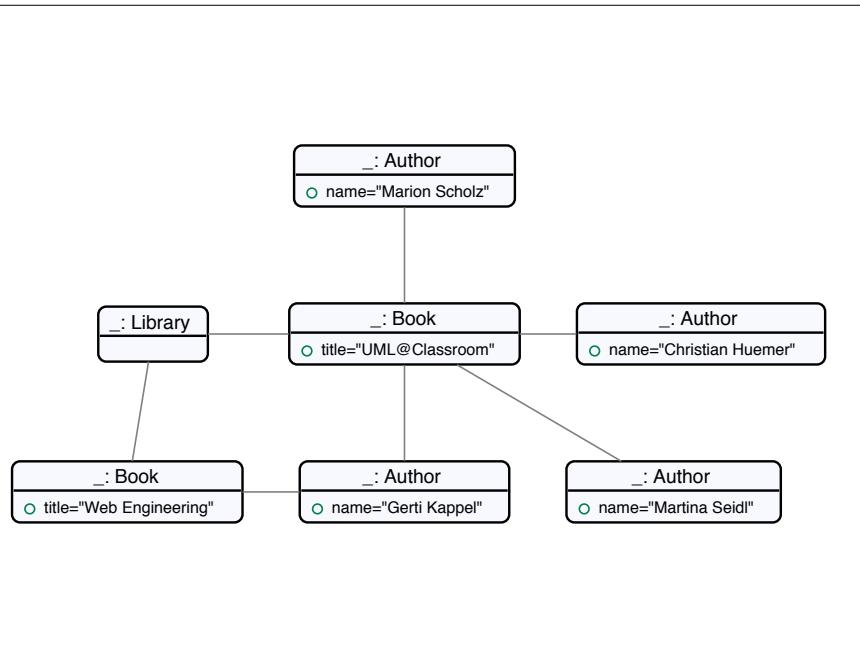
Principaux diagrammes UML



Objets



uml-diagrams.org

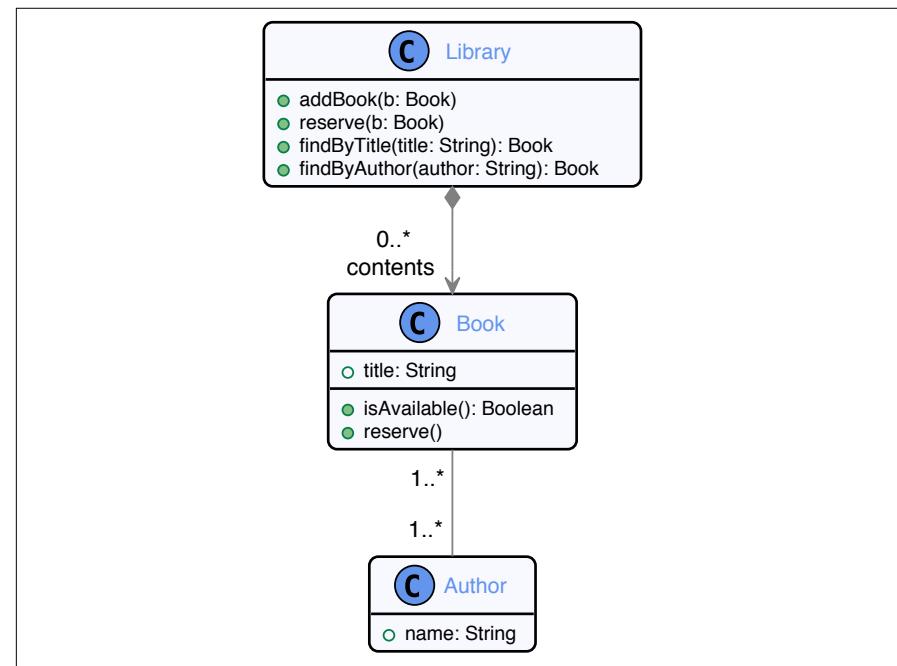
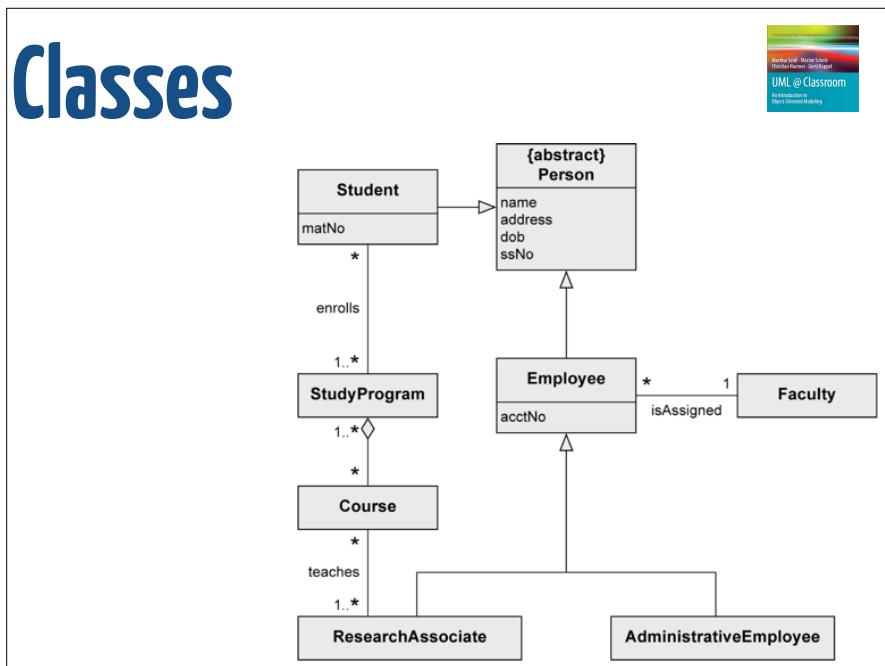
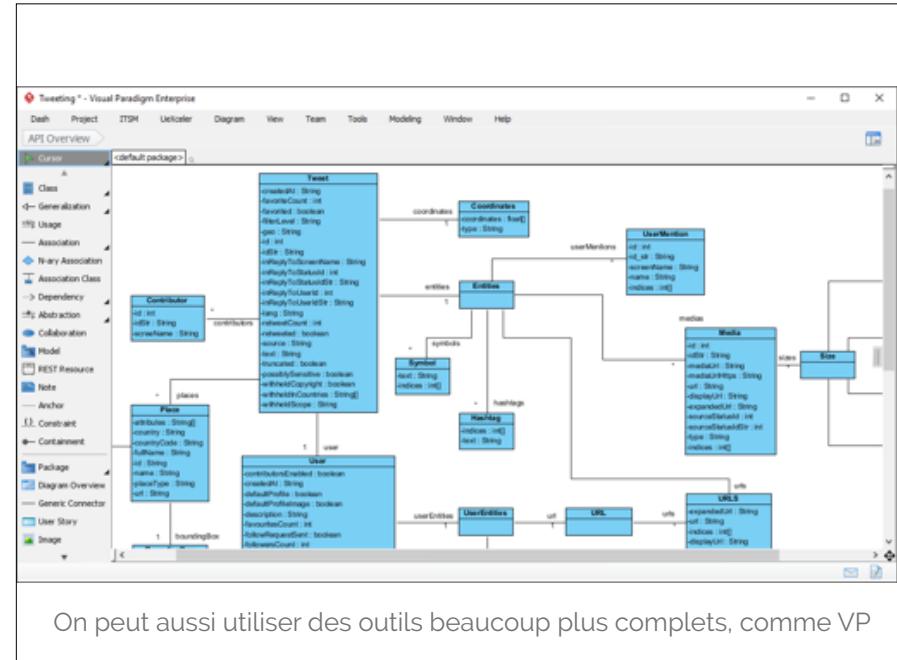
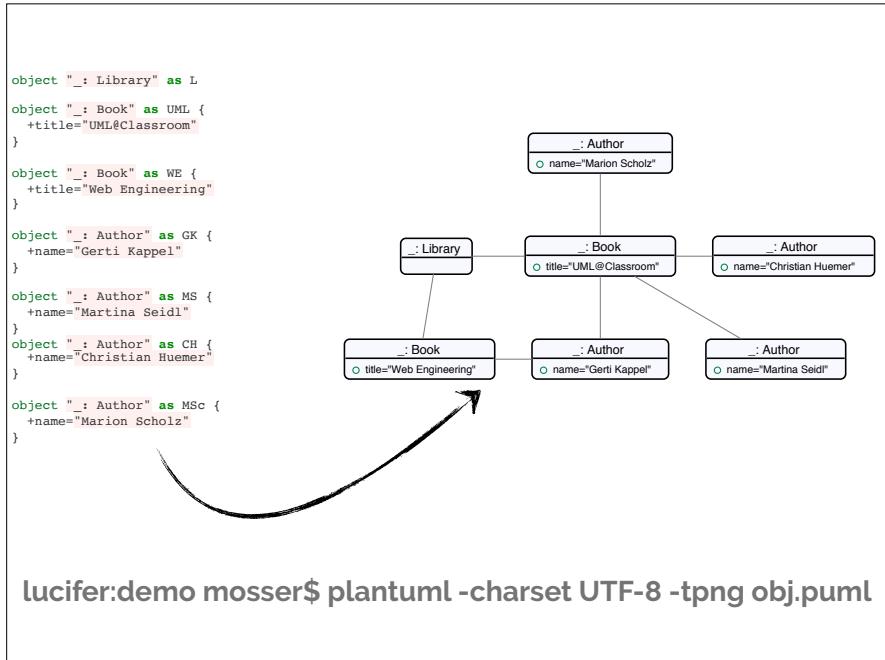


```

object "_: Library" as L
object "_: Book" as UML {
  +title="UML@Classroom"
}
object "_: Book" as WE {
  +title="Web Engineering"
}
object "_: Author" as GK {
  +name="Gerti Kappel"
}
object "_: Author" as MS {
  +name="Martina Seidl"
}
object "_: Author" as CH {
  +name="Christian Huemer"
}
object "_: Author" as MSC {
  +name="Marion Scholz"
}
  
```

`L` - UML
`L` -- WE
`UML` -- GK
`WE` - GK
`UML` -- MS
`UML` - CH
`MSc` -- UML

PlantUML, langage textuel de description de modèles UML



```

class Library {
    + addBook(b: Book)
    + reserve(b: Book)
    + findByTitle(title: String): Book
    + findByAuthor(author: String): Book
}

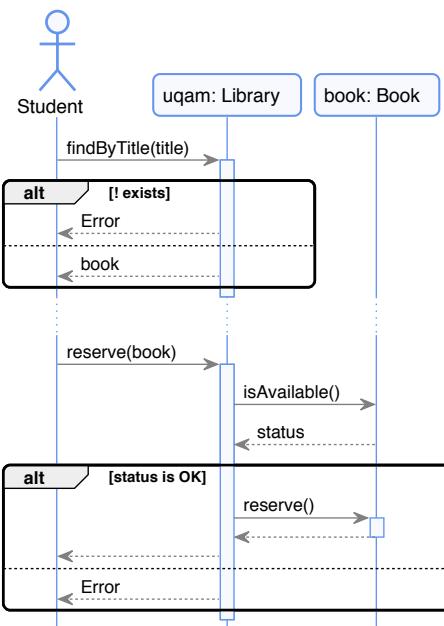
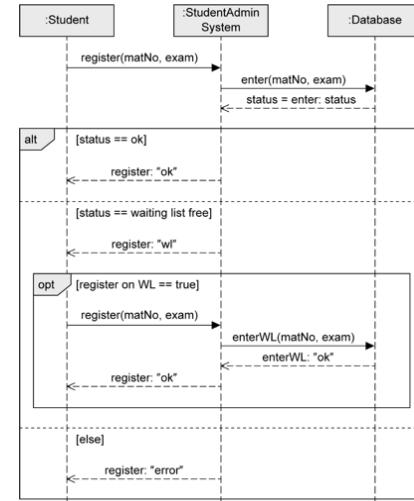
class Book {
    + title: String
    + isAvailable(): Boolean
    + reserve()
}

class Author {
    + name: String
}

Book "1..*" -- "1..*" Author
Library *--> "0..*\ncontents" Book

```

Séquence



Quels sont les défauts ?

```

actor "Student" as S
participant "uqam: Library" as L
participant "book: Book" as B

```

```

S -> L: reserve(book)
activate L
L -> B: isAvailable()
B --> L: status
alt status is OK
    L -> B: reserve()
    activate B
    B --> L
    deactivate B
    L --> S
else
    L -> S: Error
end
deactivate L

```

Harry Potter

(Code Kata)



Exemple de calcul de prix



= ?

$$5 \times 8 \times 0,75 = 30$$



$$3 \times 8 \times 0,90 = 21,6$$

} 51,60

Spécifications

- On considère les 5 premiers livres de la saga uniquement.
- Chaque livre coûte 8\$
- Politiques de rabais en fonction du panier du consommateur :
 - 2 livres différents : 5%
 - 3 livres différents: 10%
 - 4 livres différents : 20%
 - La série complète : 25%
- Le rabais ne s'applique que sur les livres concernées par la politique

<http://www.codingdojo.org/cgi-bin/index.pl?action=browse&id=KataPotter&revision=41>

Piège !



= 51,60

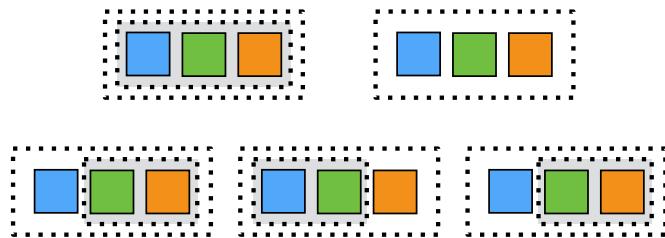
$$4 \times 8 \times 0,8 = 25,6$$



} = 2 \times 25,6 = 51,20

Situation classiquement combinatoire

B1 B2 B3



Ordre de grandeur : Nombre de Bell

Bell(7) = ??

877

```
public class PotterDiscountCalculator {  
    private double[] discountRates;  
    private int[] discounts;  
  
    private void init() {  
        discountRates = new double[]{ 1, 0.95, 0.90, 0.80, 0.75 };  
        discounts = new int[5];  
    }  
  
    public double calculatePrice(List<Integer> order) {  
        init(); // resetting the calculator before computing a price  
        if (order == null || order.isEmpty())  
            return 0.0;  
        int[] booksInOrder = calculateBooksInOrder(order);  
        calculateDiscounts(booksInOrder);  
        optimizeDiscounts();  
        double price = 0.0;  
        for (int i = 0; i < discounts.length; i++)  
            price += (8 * (i + 1) * discounts[i]) * discountRates[i];  
        return price;  
    }  
  
    protected void optimizeDiscounts() {  
        while (discounts[2] > 0 && discounts[4] > 0) {  
            discounts[2]--;  
            discounts[4]--;  
            discounts[3] += 2;  
        }  
    }  
  
    protected void calculateDiscounts(int[] booksInOrder) {  
        if (booksInOrder == null)  
            return;  
        int differentFromZero = 0;  
        for (int i = 0; i < booksInOrder.length; i++) {  
            if (booksInOrder[i] > 0) {  
                differentFromZero++;  
                booksInOrder[i] = 0;  
            }  
        }  
        if (differentFromZero > 0) {  
            discounts[differentFromZero - 1] += 1;  
            calculateDiscounts(booksInOrder);  
        }  
    }  
  
    private int[] calculateBooksInOrder(List<Integer> order) {  
        int[] result = new int[5];  
        for (int book : order)  
            result[book - 1]++;  
        return result;  
    }  
}
```

Yes
we
can!

```
private double[] discountRates;  
private int[] discounts;  
  
private void init() {  
    discountRates = new double[]{ 1, 0.95, 0.90, 0.80, 0.75 };  
    discounts = new int[5];  
}  
  
public double calculatePrice(List<Integer> order) {  
    init(); // resetting the calculator before computing a price  
    if (order == null || order.isEmpty())  
        return 0.0;  
    int[] booksInOrder = calculateBooksInOrder(order);  
    calculateDiscounts(booksInOrder);  
    optimizeDiscounts();  
    double price = 0.0;  
    for (int i = 0; i < discounts.length; i++)  
        price += (8 * (i + 1) * discounts[i]) * discountRates[i];  
    return price;  
}
```

<http://bfindeiss.blogspot.fr/2013/07/the-katapotter-in-java-solved-in-31.html>

```

protected void calculateDiscounts(int[] booksInOrder) {
    if (booksInOrder == null)
        return;
    int differentFromZero = 0;
    for (int i = 0; i < booksInOrder.length; i++) {
        if (booksInOrder[i] > 0)
            differentFromZero++;
        booksInOrder[i]--;
    }
    if (differentFromZero > 0) {
        discounts[differentFromZero - 1] += 1;
        calculateDiscounts(booksInOrder);
    }
}

```

<http://bfindeiss.blogspot.fr/2013/07/the-katapotter-in-java-solved-in-31.html>

```

@Test
public void edgeCase() {
    assertEquals( 51.20,
        calculator.calculatePrice(build(1, 1, 2, 2, 3, 3, 4, 5)), 0.01);
    assertEquals(141.20, calculator.calculatePrice(
        build( 1, 1, 1, 1, 1,
                2, 2, 2, 2, 2,
                3, 3, 3, 3,
                4, 4, 4, 4, 4,
                5, 5, 5, 5      )), 0.01);
}

@Test
public void simpleDiscounts() {
    assertEquals(15.20, calculator.calculatePrice(build(1, 2)), 0.01);
    assertEquals(21.60, calculator.calculatePrice(build(1, 3, 5)), 0.01);
    assertEquals(25.60, calculator.calculatePrice(build(1, 2, 3, 5)), 0.01);
    assertEquals(30.00, calculator.calculatePrice(build(1, 2, 3, 4, 5)), 0.01);
}

```

```

private int[] calculateBooksInOrder(List<Integer> order) {
    int[] result = new int[5];
    for (int book : order)
        result[book - 1]++;
    return result;
}

protected void optimizeDiscounts() {
    while (discounts[2] > 0 && discounts[4] > 0) {
        discounts[2]--;
        discounts[4]--;
        discounts[3] += 2;
    }
}

```

<http://bfindeiss.blogspot.fr/2013/07/the-katapotter-in-java-solved-in-31.html>

```

public class PotterDiscountCalculator {
    private double[] discountRates;
    private int[] discounts;

    private void init() {
        discountRates = new double[]{ 1, 0.95, 0.90, 0.80, 0.75 };
        discounts = new int[5];
    }

    public double calculatePrice(List<Integer> order) {
        init(); // reinitializing the calculator before computing a price
        if (order == null || order.isEmpty())
            return 0.0;
        int[] booksInOrder = calculateBooksInOrder(order);
        calculateDiscounts(booksInOrder);
        optimizedDiscounts();
        double price = 0.0;
        for (int i = 0; i < discounts.length; i++)
            price += (8 * (i + 1) * discounts[i]) * discountRates[i];
        return price;
    }

    protected void optimizeDiscounts() {
        while (discounts[2] > 0 && discounts[4] > 0) {
            discounts[2]--;
            discounts[4]--;
            discounts[3] += 2;
        }
    }

    protected void calculateDiscounts(int[] booksInOrder) {
        if (booksInOrder == null)
            return;
        int differentFromZero = 0;
        for (int i = 0; i < booksInOrder.length; i++) {
            if (booksInOrder[i] > 0)
                differentFromZero++;
            booksInOrder[i]--;
        }
        if (differentFromZero > 0)
            discounts[differentFromZero - 1] += 1;
        calculateBooksInOrder(booksInOrder);
    }

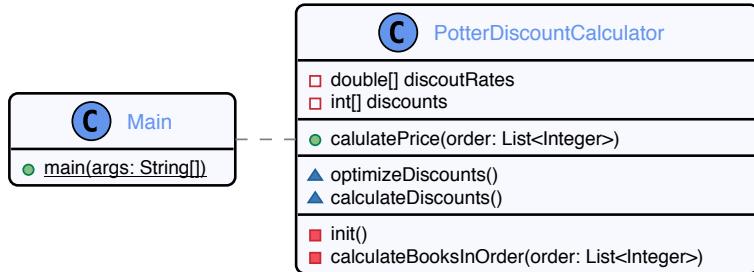
    private int[] calculateBooksInOrder(List<Integer> order) {
        int[] result = new int[5];
        for (int book : order)
            result[book - 1]++;
        return result;
    }
}

```

Sérieusement
we
can !

Yes

Version Modèle (Structurel)



Problèmes



“



Bon, j'ai réfléchi un peu à ce problème et algorithmiquement, **il y a trois options** pour le résoudre : une **naïve** (ton exemple de code java) qui est exponentielle en le nombre de bouquin à acheter, une **dynamique** qui est linéaire en le nombre de bouquins à acheter (quadratique si le nombre des offres de prix est infini) et enfin une qui est **logarithmique** (amorti) en le nombre de bouquins (mais seulement avec un nombre d'offres de prix finis) et **qui sera en pratique en temps constant** (à moins d'avoir un nombre de bouquin à acheter qui ne tient pas sur un int32, mais il y a peu de chances quand même).

- Christophe Papazian



```
from itertools import product
Bundles = list(product((0,1),repeat=5))[1:]
tags = [0,800,2*8*95,3*8*90,4*8*80,5*8*75]

def bundles(t) :
    for k in Bundles :
        if all(t[i]>=k[i] for i in range(5)) :
            yield tags[sum(k)], tuple(t[i]-k[i] for i in range(5))

def best_price(t,cache={(0,0,0,0,0):0}) :
    t=tuple(sorted(t))
    try : return cache[t]
    except KeyError :
        cache[t]=res=min(a[0]+best_price(a[1]) for a in bundles(t))
        return res

def price(l) : return best_price(tuple(l.count(i) for i in range(5)))/100
```

“Mais **je trouve ça plutôt bof**, ça passe les tests, et c'est polynomial en le nombre de bouquins en entrée, mais c'est un polynome de degré 5, donc les performances sont mauvaises en pratique. **Faut que je réfléchisse pour la solution logarithmique...** mais pas ce soir” - Christophe

Guide de survie en Complexité

```
if(christophe.says("Je dois réfléchir"))
    return "Fly you fools!"
```

=> Le problème n'est pas trivial

Comment encapsuler la complexité du calcul de manière à rendre le code source compréhensible et maintenable ?

Version améliorée



```
tags = [0,800,2*8*95,3*8*90,4*8*80,5*8*75]

def best_price(t) :
    t=tuple(sorted(t))
    m=min(t[0],t[2]-t[1])
    return tags[5]*(t[0]-m)+tags[4]*(2*m+t[1]-t[0])+
           tags[3]*(t[2]-t[1]-m)+tags[2]*(t[3]-t[2])+ 
           tags[1]*(t[4]-t[3])

def price(l) :
    return best_price(tuple(l.count(i) for i in range(5)))/100
```

"ok, j'ai une version logarithmique qui passe les tests ;P. Version python avec test, c'est linéaire par rapport au log du nombre de bouquins achetés." - Christophe

Règle de 3 de la conception logicielle

Lisibilité
Lisibilité
Lisibilité

Principe de SOLIDité en conception objet

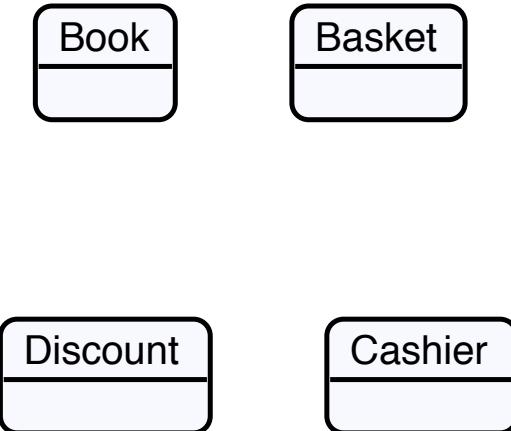
- S: Single Responsibility
- O: Open / closed principle
- L: Liskov-compliant (substitution)
- I: Interface Segregation
- D: Dependency inversion

Ici

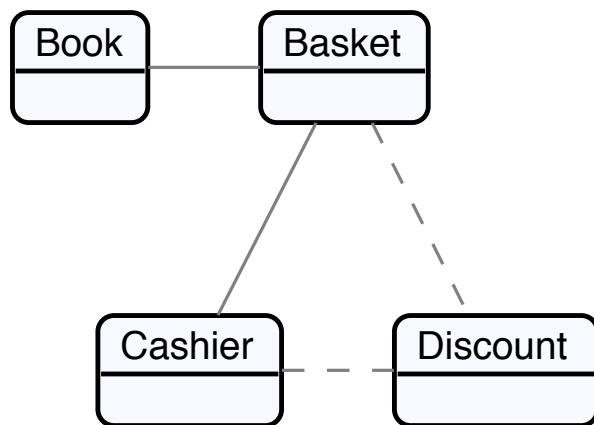
Après

Encore
Après

Modularisation



Modularisation



Calcul du Prix pour $b \in \text{Basket}$

b is empty => **0.0**

Let **discounts** = { $d \in \text{Discount} \mid d \text{ can be applied to } b$ }

discounts is empty => **$|\text{books}|_b * \text{PRICE}$**

Let **candidates** = { $p \in \mathbb{R} \mid d \in \text{discounts}, p = \text{compute}(b, d)$ }

=> **$\min(\text{candidates})$**

NAIVE

Calcul du prix avec un rabais d pour un panier b

Let **local price** = **d** applied to **b**

Let **remaining** = clone of **b** without the books used in **d**

=> **local price** + **compute(remaining)**

NAIVE

Implémentation Interne

```
public class Basket {  
    private Map<Book, Integer> contents = new HashMap<>(Book.class);  
  
    public Basket() {  
        for(Book b: Book.values())  
            contents.put(b, 0);  
    }  
  
    public Basket(Book... chosen) {  
        this();  
        for(Book book: chosen)  
            contents.put(book, contents.get(book) + 1);  
    }  
}
```

on sen fiche!

Description Comportementale

C Basket

- this()
- this(books: Book[])
- duplicate(): Basket
- howMany(b: Book): Int
- howManyDifferent(): Int
- howManyBooks(): Int
- isEmpty(): Bool
- remove(books: Book[])

Structure ?

“OSEF”

“

One of the **great leaps in OO** is
to be able to answer the question

“how does this work?”

with “**I don’t care**”.

- Alan Knight

Exemple : Le cas des **Rabais**



```
public class Discount {
    [REDACTED]
    public Discount(int nbBooks, double percentage) { [REDACTED] }
    public boolean canBeApplied(Basket b) { [REDACTED] }
    public double apply(Basket b) { [REDACTED] }
    public Basket removePayedBooks(Basket b) {
        [REDACTED]
    }
    private Map<Book, Integer> contents(Basket b) {
        [REDACTED]
    }
}
```

```
public class Discount {
    [REDACTED]
    public Discount(int nbBooks, double percentage) { [REDACTED] }
    public boolean canBeApplied(Basket b) { [REDACTED] }
    public double apply(Basket b) { [REDACTED] }
    public Basket removePayedBooks(Basket b) {
        [REDACTED]
    }
    private Map<Book, Integer> contents(Basket b) {
        [REDACTED]
    }
}
```

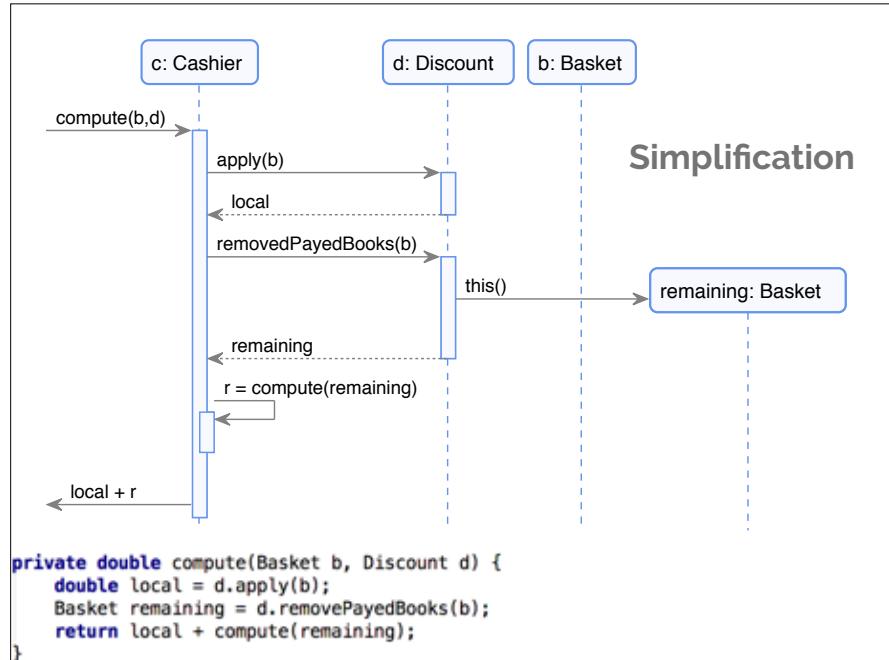
On s'en fiche !

Let **local price = d** applied to **b**

Let **remaining** = clone of **b** without the books used in **d**

=> **local price + compute(remaining)**

```
private double compute(Basket b, Discount d) {
    double local = d.apply(b);
    Basket remaining = d.removePayedBooks(b);
    return local + compute(remaining);
}
```



b is empty => **0.0**
Let **discounts** = { $d \in \text{Discount} \mid d \text{ can be applied to } b$ }
discounts is empty => $|\text{books}|_b * \text{PRICE}$
Let **candidates** = { $p \in \mathbb{R} \mid d \in \text{discounts}, p = \text{compute}(b, d)$ }
=> **min(candidates)**

```

private double compute(Basket b) {
    if(b.isEmpty()) { return 0.0; }

    List<Discount> availables = findAvailableDiscount(b);
    if(availables.isEmpty()) {
        return PRICE * b.howManyBooks();
    } else {
        return availables.stream().map(d -> compute(b, d)).min(Double::compare).get();
    }
}
    
```

```

private double compute(Basket b) {
    if(b.isEmpty()) { return 0.0; }

    List<Discount> availables = findAvailableDiscount(b);
    if(availables.isEmpty()) {
        return PRICE * b.howManyBooks();
    } else {
        return availables.stream().map(d -> compute(b, d)).min(Double::compare).get();
    }
}
    
```

b is empty => **0.0**
Let **discounts** = { $d \in \text{Discount} \mid d \text{ can be applied to } b$ }
discounts is empty => $|\text{books}|_b * \text{PRICE}$
Let **candidates** = { $p \in \mathbb{R} \mid d \in \text{discounts}, p = \text{compute}(b, d)$ }
=> **min(candidates)**

Let **local price** = **d** applied to **b**
Let **remaining** = clone of **b** without the books used in **d**
=> **local price + compute(remaining)**

```

private double compute(Basket b, Discount d) {
    double local = d.apply(b);
    Basket remaining = d.removePayedBooks(b);
    return local + compute(remaining);
}
    
```

Projet #1

Le jeu de Poker



Projet "Jeu de Poker"

- Auteur : Sébastien Mosser
- Version : 2019.09
- Durée : 16 heures sur 2,5 semaines (2 ateliers de 2h + travail personnel de 12h)

Lisez bien le sujet jusqu'au bout avant de commencer à travailler sur le projet.

Objectifs du projet

- Prendre en main les outils utilisés en support du cours;
- Analysier un code légataire en vue de le faire évoluer;
- Transformer ce code en véritable application orientée objet;
- Faire évoluer un code objet en respectant dans la mesure du possible : 1. les principes S, O et L de SOLID ; 2. la Loi de Demeter (principe de connaissance minimale); 3. les principes objet de faible couplage / forte cohésion.
- Décrire une application orientée objet avec le langage UML (diagrammes d'objets, de séquences et de classes).

Les ateliers sont des temps où l'équipe enseignante (prof + démo) est présente pour vous donner de la rétro-action sur votre projet. Il est vital de profiter de ce temps de travail prévu dans votre emploi du temps pour améliorer vos projets avant la remise finale.

Exemple de code légataire

```

private static String findComb(String[] cs) {
    // Detect a flush
    char col = cs[0].charAt(1);
    boolean f = true;
    for(String c: cs) {
        if (c.charAt(1) != col)
            f = false;
    }
    if(f)
        return "F " + maxVal(cs);

    // Detect a pair
    char vMax = 'X';
    for(int i = 0; i < cs.length; i++) {
        for(int j = i+1; j < cs.length; j++) {
            if (cs[i].charAt(0) == cs[j].charAt(0)
                && (vMax == 'X' || getTable().get(cs[i].charAt(0)) > getTable().get(vMax))) {
                vMax = cs[i].charAt(0);
            }
        }
    }
    if(vMax != 'X')
        return "P " + vMax;

    // Nothing interesting => Highest card
    return "H " + maxVal(cs);
}

```

Travail à réaliser (24.09.2019, 23h50)

- Analyser le code légataire fourni
 - Comment le faire évoluer ?
 - Quels sont ses défauts ?
- Proposer un **ré-usinage orienté objet**
 - Quels sont les concepts "métiers" ?
 - Mise en oeuvre d'évolution fonctionnelle
- Analyse du code objet conçu
 - De quelles propriétés dispose votre conception ?

Grille d'(auto-) évaluation

Élement	Critère d'évaluation	Note (/100)
Questions	(#1) Évolution du code légataire	/5
	(#2) Analyse des défauts du code légataire	/10
	(#3) Justification des choix de conception	/15
	(#4) Évolution du code objet	/5
Modèles	Justesse & Pertinence de la conception	/15
	Cohérence inter-modèles	/5
Code	Respect des principes de conception	/15
	Qualité du code Java et du dépôt Git	/10
	Cohérence du code avec les modèles	/10
	Qualité des tests	/10

