INFO701C1_MTIAY
SCRUM et Bonnes pratiques JAVA

Présentation générale des interventions

Jérémie DERUETTE

Légendes et conventions

Objectif global de la journée

Jour

Quoi

Pourquoi

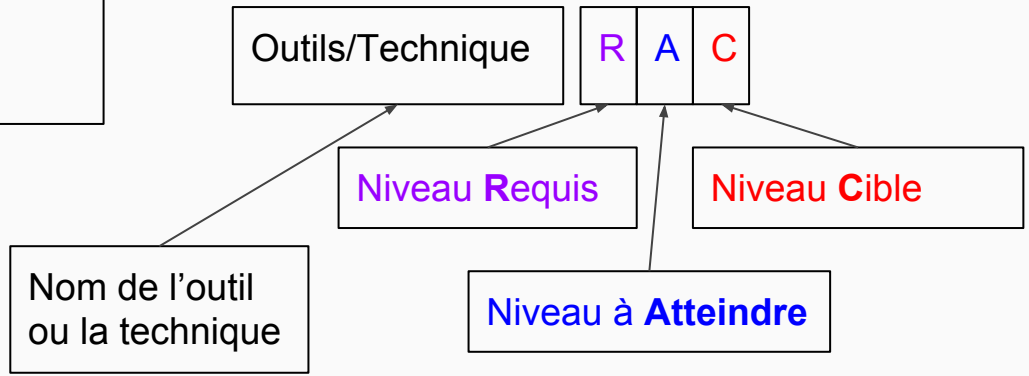
Comment

Info, outil, technique

Quel Niveau ?

0 : inconnu
1 : peu utilisé
2 : déjà utilisé et je connais les bases
3 : utilisé régulièrement, je connais les cas avancés
4 : je maîtrise les cas avancés
5 : je connais parfaitement

Design Pattern MVC {- 0 | 1 | 3 -}



Scrum et gestion de projet

Semaine 1

Gestion de projet {- 0 | 1 | 1 -}

Scrum {- 0 | 1 | 3 -}

XP {- 0 | 1 | 1 -}

Git et GitHub {- 0 | 1 | 1 -}

Open Source {- 0 | 1 | 1 -}

Savoir ce qu'est un projet

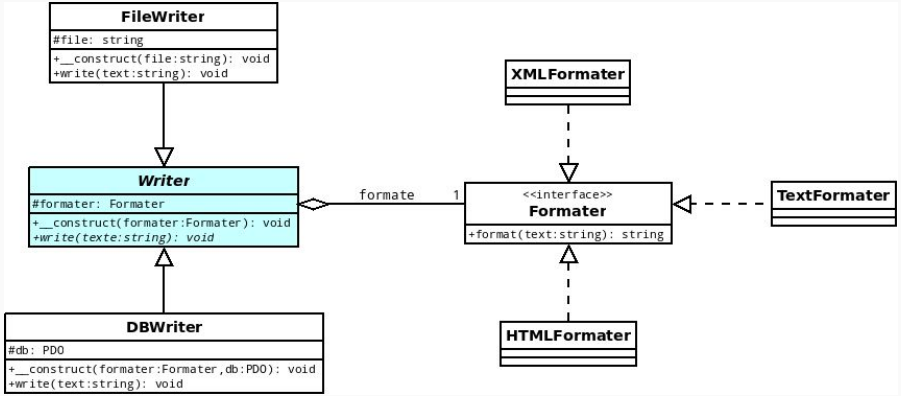
Savoir gérer un projet en Agile

Mettre en oeuvre pour le projet

1. Cours Agile/Scrum
2. Planning Release du projet
3. Récupérer exemple sur GitHub
   a. https://github.com/jderuette/iae_annecy_etape1
4. Ajouter prénom

Rappels POO

- Les classes définissent la structure et les traitements possibles
- Les objets sont des instances d'une classe
- Une interface définit un contrat entre deux classes
- Chaque classe a en général UNE responsabilité



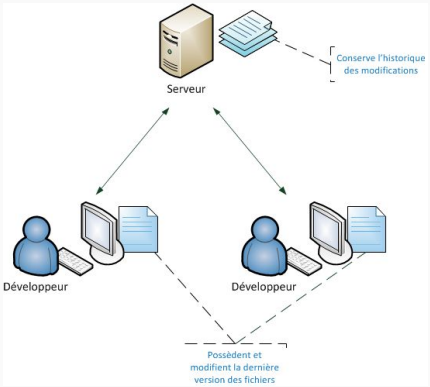
Les classes permettent de définir le "domaine" d'une application : ce que sait gérer l'application.

Exercice : qu'est-ce qu'un chien ?

Git {- 0 | 1 | 1 -}

Git est un système de gestion de versions.
Il permet à plusieurs personnes de travailler en même temps sur le même code source sans risquer des pertes de données.

- Il détecte lorsqu'un fichier a été modifié par deux personnes en même temps et évite que le "dernier" à modifier écrase les lignes de code du précédent
- Il permet de stocker tout le code dans un endroit centralisé



Clone : faire une copie en local d'un projet
Fetch : récupérer les modifications sur le repository
Merge : intégrer les modifications des autres dans notre code
Pull : (eclipse) : Fetch + merge
Commit : grouper plusieurs modifications pour les envoyer plus tard
Push : envoyer nos commits sur le repository

GitHub {- 0 | 1 | 1 -}

GitHub s'appuie sur Git et ajoute des fonctionnalitées "sociales".

- Il fournit un système de stockage du code gratuit si le code est OpenSource
- Il facilite la visualisation des données dans Git
- Il permet d'avoir une "copie personnelle" d'un projet Git (Fork)
- Il permet de proposer d'intégrer les modifications qu'on a apporté au "propriétaire" d'un projet (Pull Request)

Fork : récupérer un projet dans notre GitHub
Pull Request : demander à intégrer une de nos modifications dans le repository principal (celui à partir duquel on a effectué le fork).

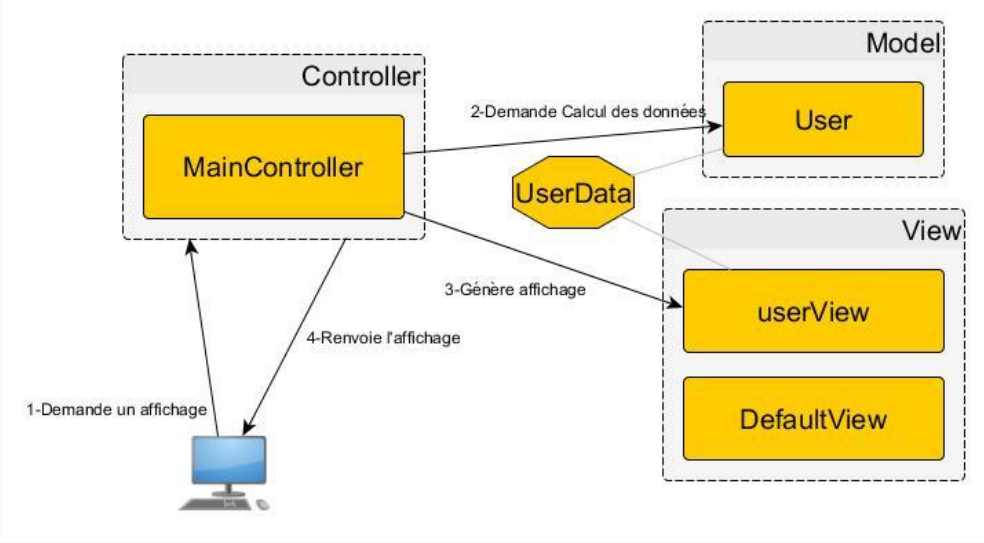
Exercice : Faire un fork du projet iae_anency_etape1, puis faire un clone avec Eclipse

MVC (Model View Controller)

MVC est un design Pattern qui permet de faciliter les interactions utilisateurs avec une application.

- Le Model contient tous les traitements métiers
- La View permet de représenter les données
- Le Controller reçoit les données de l'utilisateur et décide de l’action à effectuer

Dans une application il y a en général plusieurs Controllers/Models/Views.



Build qualité et découverte Design Patern

Semaine 2

Scrum {- 1 | 2 | 3 -}

Eclipse {- 0 | 1 | 3 -}

Maven {- 0 | 1 | 2 -}

PMD/checkstyle {- 0 | 1 | 2 -}

MVC {- 0 | 1 | 4 -}

Java {- 0 | 2 | 4 -}

Utiliser le code des autres

Partager son code

Auditer son code (ou celui des autres)

1. Ajouter une action
2. Lancer un Build avec rapports
3. Etudes des rapports (PMD/CheckStyle)
4. Planning pocker US 1,2,3,4
5. Réalisation US
6. Démo/rétro

Installer et configurer Eclipse

1. Installer Eclipse Neon
2. Installer le plugin PMD
   a. Aller dans help->install new Software
   b. Ajouter un site avec l'URL https://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/ (nommez le "PMD for Eclipse Update Site")
   c. Saisir "PMD" dans la recherche
   d. Vérifier que "PMD for Eclipse Update Site" est sélectionné dans la liste des repository "Work with"
   e. Cocher la case "PMD for elcipse 4"
   f. Suivre les instructions d'installation
3. Installer le plugin Checkstyle
   a. Aller dans help->Eclipse marketPlace
   b. Filtrer sur "checkstyle"
   c. Cliquer sur "install" de "Checkstyle plugin 6.19.1"

Mettre en oeuvre Dev JAVA et IOC

Semaine 3

Refactoring {- 0 | 2 | 3 -}

MVC {- 1 | 2 | 4 -}

IoC {- 0 | 1 | 2 -}

JavaDoc {- 0 | 1 | 1 -}

Stratégie de test {- 0 | 2 | 2 -}

JUnit {- 0 | 2 | 2 -}

Maîtriser les concepts Objets

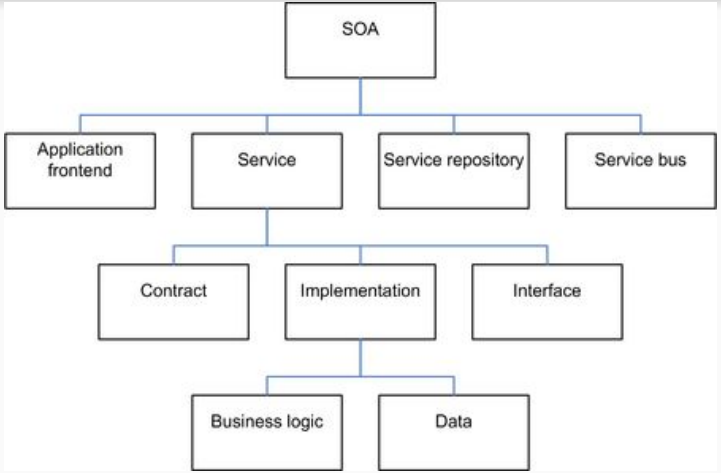
Mettre en oeuvre des designs patterns

Savoir quoi tester et comment

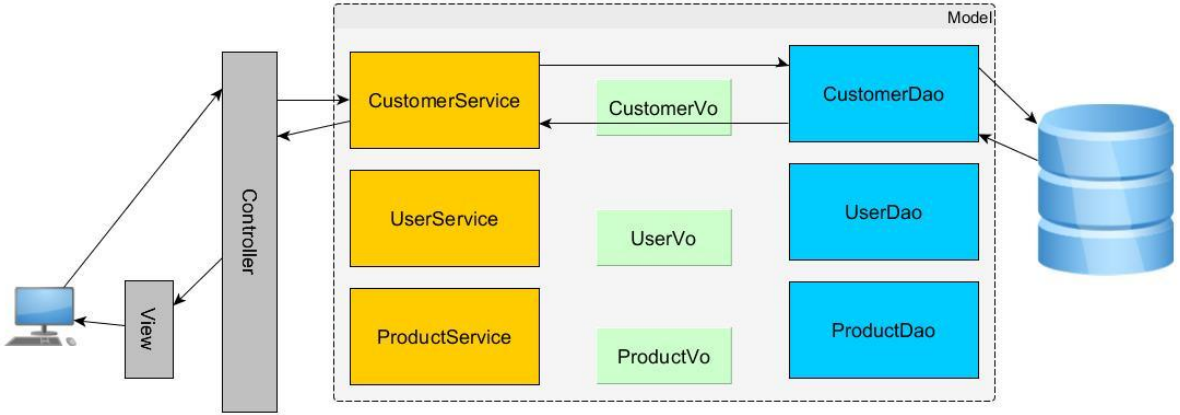
1. Planning pocker US 5,6,7,8
2. Réalisation des User Story
3. Démo/rétro

Service Oriented Architecture et Data Driven Architecture

SOA est un style de conception qui met en avant les services métiers. Les developement se feront en fonction des traitements à effectuer.
La gestion des données arrivent au second plan.



En DDA, ce sont les données qui sont l'élément central qui vont diriger les développements. Les services découlent de ces données et sont en général très génériques.



Mettre en oeuvre Dev JAVA

Semaine 4

MVC {- 2 | 3 | 4 -}

IoC {- 1 | 2 | 2 -}

Java {- 2 | 4 | 4 -}

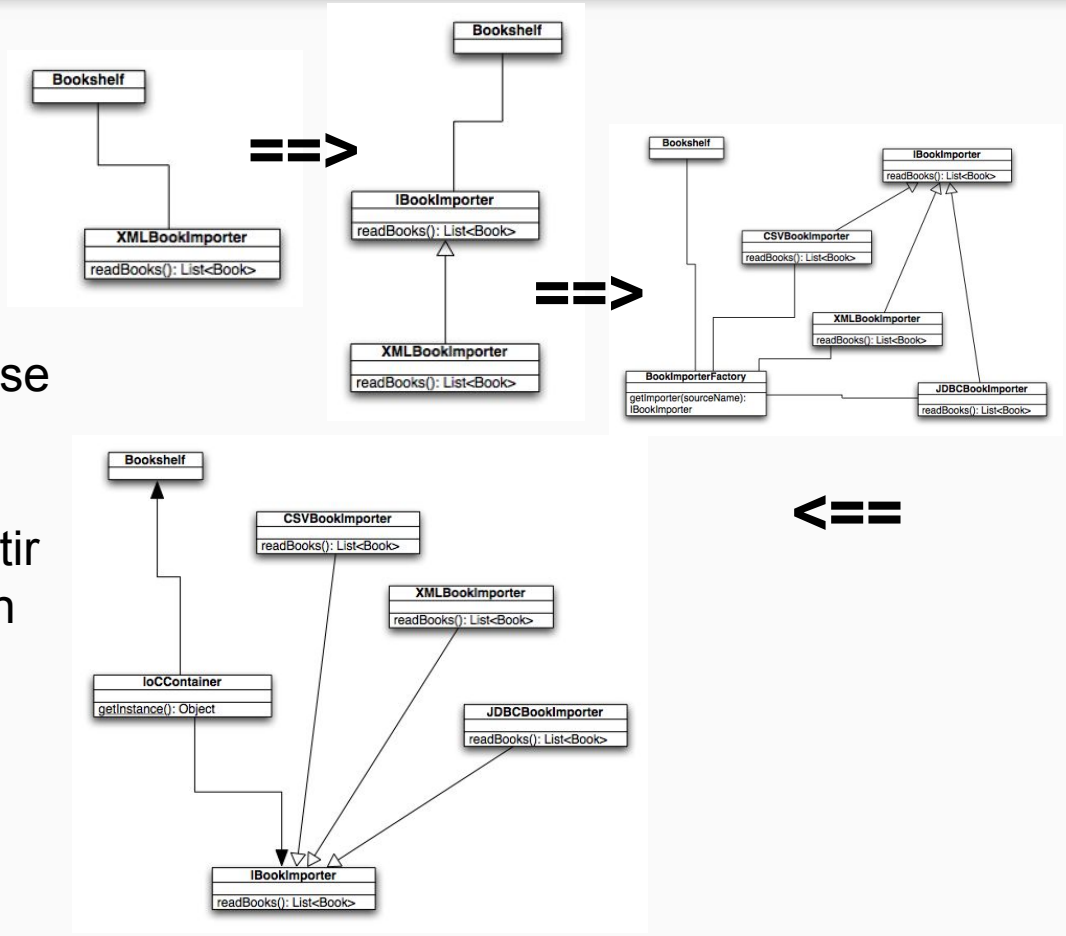
Maîtrise du développement

1. Planning US 9,10
2. Validation des acquis (Dev Objet)

IoC (Inversion of Control) Inversion de dépendance

http://gfx.developpez.com/tutoriel/java/ioc/

Une Classe est susceptible de dépendre d'une autre Classe pour fonctionner.
Pour éviter de coupler ces deux Classe on peut utiliser une Interface.
L'IoC va permettre de rendre indépendant des Objects créés à partir de ces Classe la dépendance vers un objet de l'autre classe.

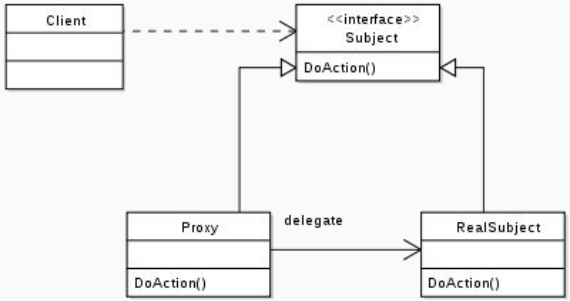


Design Pattern

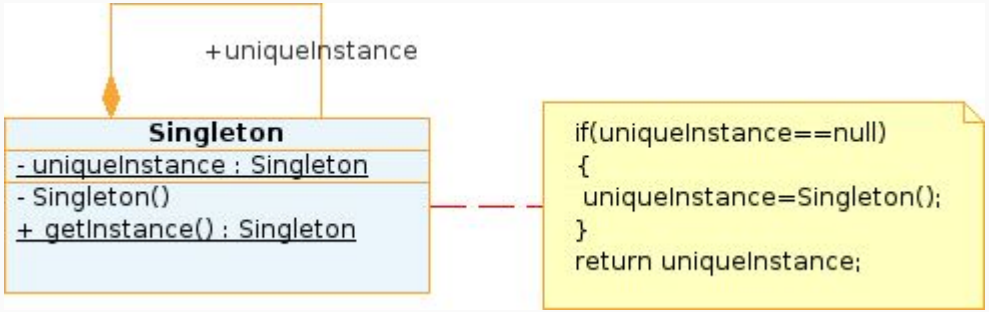
https://fr.wikipedia.org/wiki/Patron_de_conception

Un patron, de conception est un solution réutilisable à un problème donné.
Pour pouvoir utiliser efficacement un Design Patern il faut d'abord avoir identifié notre problème. Une alternative est de suivre ces “règles” comme étant des bonne pratiques.

A lire : Tête la première : Design Patterns



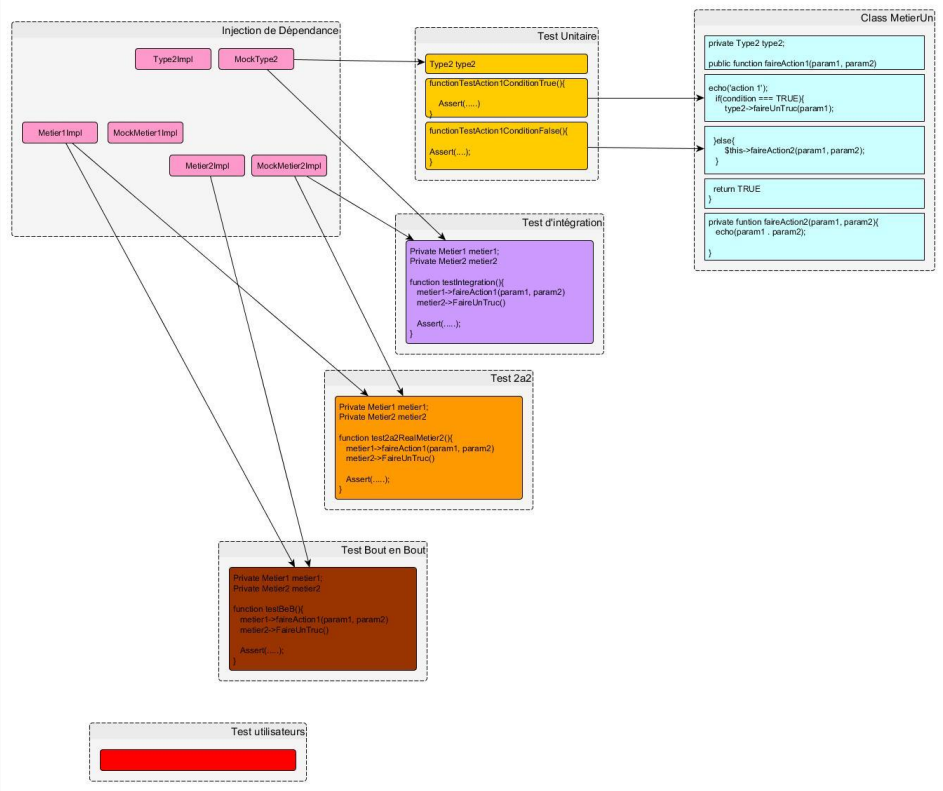
https://fr.wikipedia.org/wiki/Proxy_(patron_de_conception)



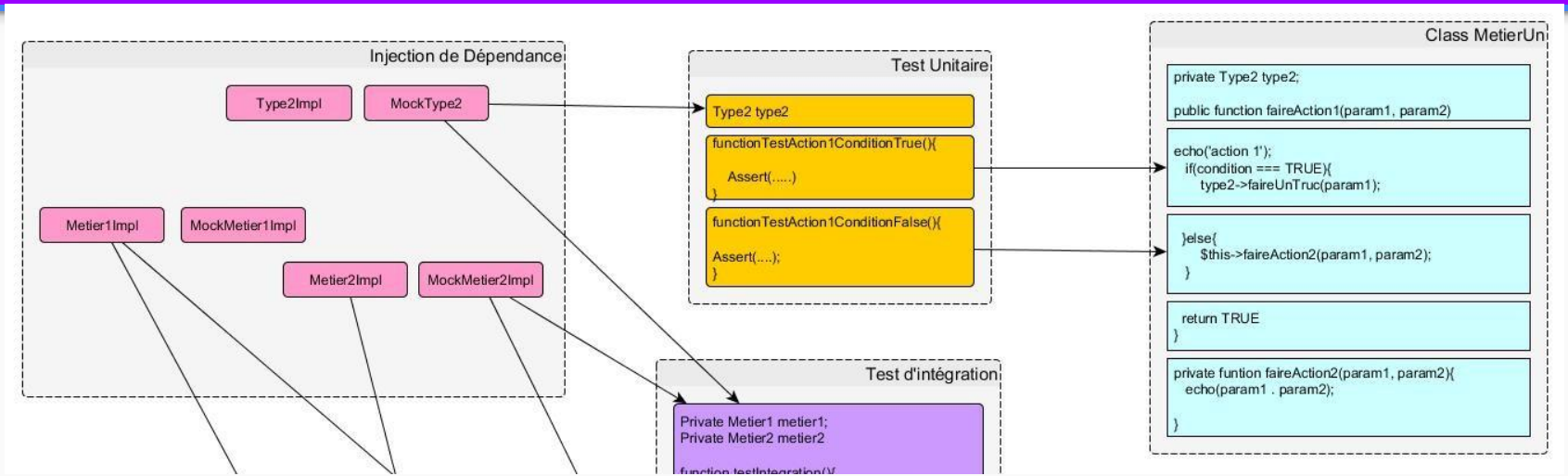
http://design-patterns.fr/singleton

Les différents type de tests

- Les Tests Unitaires (TUs) testent chaque ligne de code.
- Les Tests d'intégrations tests plusieurs classes/modules entre eux
- Les test 2a2 testent deux Système d'Informations entre eux, deux par deux.
- Les tests Bout en Bout testent toutes la chaine des différents SIs ensembles.
- Les tests utilisateurs se font en générale sur un environnement Bout en Bout, mais sont effectuées par des "vrais" utilisateurs



JUnit, couverture de test et bouchons (mocks)



- JUnit est un outils aidant à effectuer les les Tests Unitaires en Java.
- Il y a des version pour d'autre langages (phpUnit, …) regrouper sous le nom xUnit

Les tests unitaires ayant pour but de vérifier des lignes de codes, il est assez facile de quel est la "couverture du code".
Cobertura est un outils (et un plugin maven) qui permet de calculer et visualiser la couverture du code sur du code en JAVA.

Livraison d’un projet

Semaine 5

Maven {- 1 | 2 | 2 -}

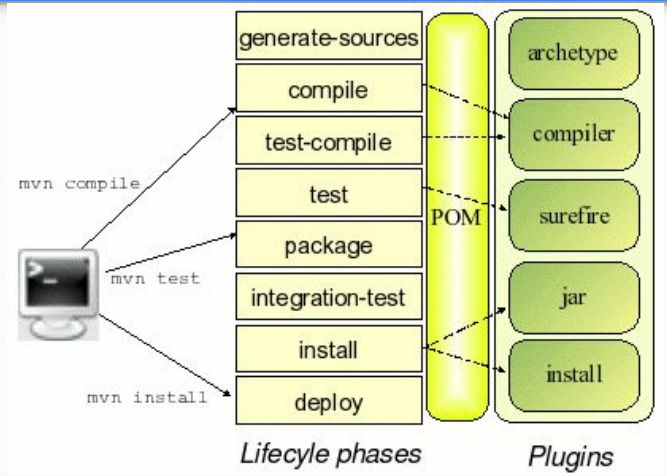
Savoir comment livrer une application java (jar)

1. Finalisation projet
2. packaging et livraison du projet

Maven

Maven est un outils de construction d'application pour des projet JAVA.

- gère les dépendances
- gère le cycle de construction
- gère le stockage et l'archivage de dépendances
- propose un repository central
- permet l'ajout de repository
- il s'intègre facilement dans les principaux IDE (notamment éclipse)
- Il se base sur un seul fichier de configuration pom.xml par projet



Goal : objectif à atteindre qui va déclencher des actions
Dépendance : librairie externe
Zero Conf : principe permettant d'utiliser un "module" avec une configuration par défaut viable.
Repository : endroit contenant des dépendances (maven central, repository local)

Dependency Injection (Injection de dépendances)

L'injection de dépendance (DI) permet de mettre en oeuvre de l'IoC assez facilement.
Un élément "externe" sera chargé d'injecter les "bons" Object dans les autre Objects.
Les dépendances sont donc gérées par cette élément externe et plus par chaque Object

```
App appli = new Application();
appli.setCustomerService(New
CustomerServiceImpl(New
CustomerDaoSerializeImpl()));
```

Injection du service dans l'application via un setter.
Injection de la Dao via le constructeur

```
class CustomerServiceImpl {
  CustomerDao dao;
  //Constructeur
  public void addCustomer(String name){
    //controles
    this.dao.save(name);
  }
}
```