Module A213
Bases de la programmation orientée objet

Rémi Watrigant
remi.watrigant@inria.fr

(basé sur le cours de Christelle Caillouet)

Organisation du module

- Répartition horaire :
  - 10h CM
  - 30h TD/TP
- Evaluation…? :
  - un ou plusieurs TP noté(s)
  - évaluation à mi-parcours
  - DS final

Objectif du module (PPN 2013)

➔ **Développer un programme dans un langage de POO à partir d'une conception détaillée.**

- En lien avec le module A214 Bases de la COO.

Contenu du PPN

- Concepts fondamentaux de la programmation orientée objet (encapsulation, composition, polymorphisme, héritage, cycle de vie des objets)
- Lecture d'une conception orientée objet détaillée
- Mise en œuvre de tests unitaires
- Utilisation de briques logicielles, d'interfaces de programmation (API : Application Programming Interface), de bibliothèques
- Sensibilisation aux bonnes pratiques de la programmation (versions, documentation du code)

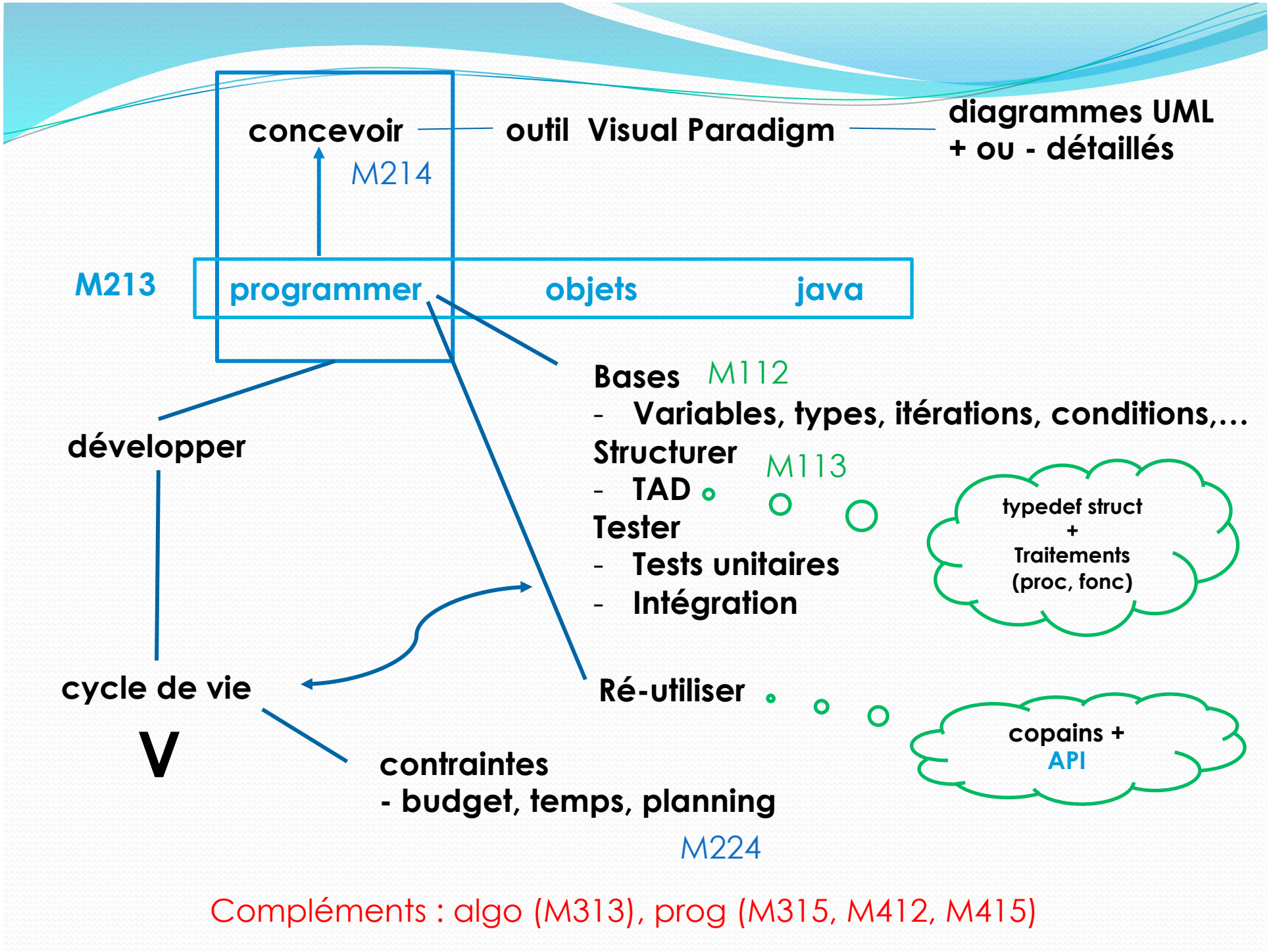
Choix pour le module

- Choix d'une méthodologie d'analyse et de conception (Unified Modeling Language - **UML**)
- Choix d'un environnement de conception (**Visual Paradigm**)
- Choix d'un langage de programmation orienté objets (**Java**)
- Choix d'un environnement de développement IDE (**Eclipse**)
- Choix d'un framework de tests unitaires (**JUnit**)

Autres langages orienté objet

- C++ : très utilisé
- C# : langage de Microsoft (appartient à .NET)
- Objective C : langage utilisé par Apple
- PHP : langage très utilisé sur le Web
- Python
- Ruby
- Eiffel
- Ada
- Smalltalk
- …

- La syntaxe change mais les concepts sont les mêmes!



Compléments : algo (M313), prog (M315, M412, M415)

Pré-requis

- Modules M112 & M113 :
  - Structures algorithmiques fondamentales : choix, répétitions
  - Notion de sous-programmes (nommage des variables, assertions, documentation, etc.)
  - Notion de types et de données
  - Apprendre à réutiliser les fonctions, procédures ou méthodes existantes du langage
  - Gestion des erreurs
  - Programmation modulaire
  - Concepts et mise en œuvre des TAD

Module connexe M214 COO

- Modélisation objet pour l'analyse et conception détaillée en UML
- Production de tests unitaires, problématique de la non régression
- Gestion des versions dans le développement
- Documentation du code
- Sensibilisation aux bonnes pratiques de la conception et du développement

Module connexe M224 Gestion de projet

- La démarche projet
- Les acteurs : le maître d'ouvrage, le maître d'œuvre, les sous-traitants, le comité de pilotage
- L'équipe projet : répartition des rôles
- Le cahier des charges : analyse et compréhension des besoins du client
- La définition des tâches, planification et enchaînement, attribution des ressources
- Les outils d'ordonnancement : graphe Pert, diagramme de Gantt
- La documentation

Modules complémentaires en DUT2

- **M313 : Algorithmique avancée (java)**
  Savoir mettre en œuvre des structures de données avancées (y compris récursives) et les algorithmes qui les manipulent

- **M315 : Conception et programmation avancées**
  Produire une conception détaillée en appliquant des modèles de conception, la mettre en œuvre en utilisant des bonnes pratiques de programmation orientée objet

- **M412 : Programmation répartie (java)**
  Savoir programmer une application répartie (multi processus – multi threads – distribuée sur un réseau)

Bibliographie

- Site de référence :

http://www.oracle.com/fr/java/index.html

- Documentation en ligne (développement) :

http://docs.oracle.com/javase/8/docs/api/

- Livres :
  - La programmation orientée objet, H. Bersini, Eyrolles
  - Programmer en java, 9ème édition, C. Delannoy, Eyrolles
  - En ligne : Penser en java, B. Eckel http://bruce-eckel.developpez.com/livres/java/traduction/tij2/

La programmation orientée objet

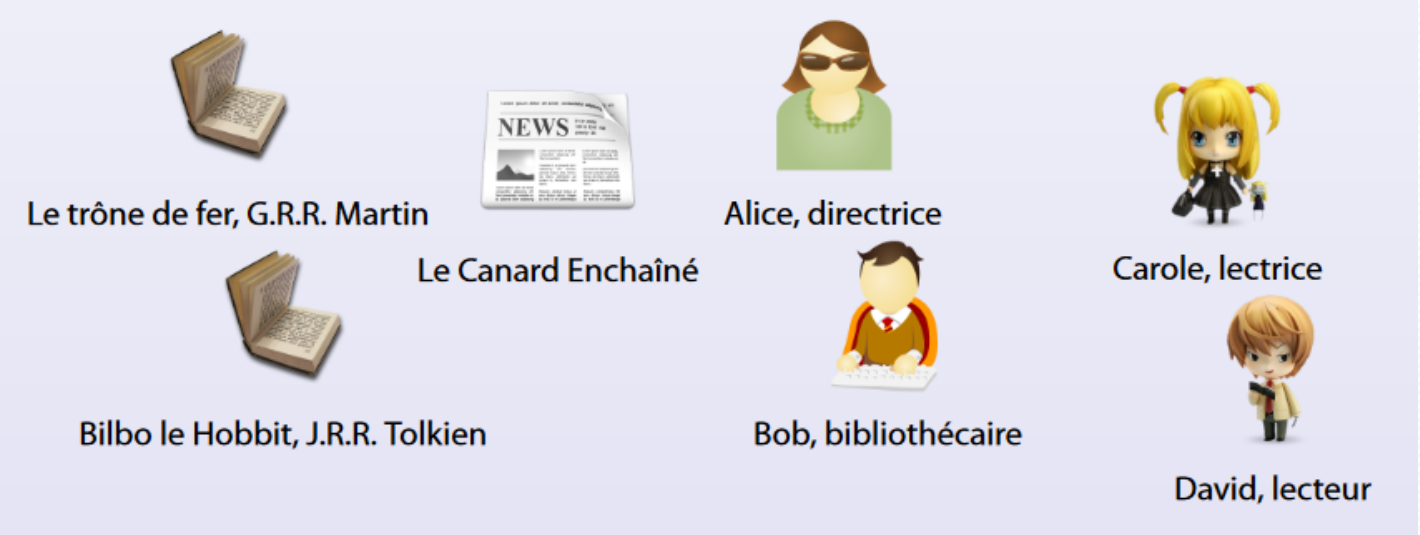

MATLAB
Java
Python
PHP
Lisp
VisualBasic
Perl
JavaScript
C++
COBOL
Lua
Groovy
Ruby
SQL
HTML
Pascal
C#
depositphotos

Différents paradigmes de programmation

	IMPERATIF	FONCTIONNEL	OBJET	DESCRIPTIF
DEMARCHE	Procédurale Série d'instructions, sauts conditionnels	Flots de données Diagramme de structure	Objets, classes, composition, réseau de messages	Besoin, expressif, léger
CONCEPTS	Itération, structures de contrôles Exécution d'instructions qui modifient l'état de la mémoire	Evaluation d'expressions qui ne dépendent que de la valeur des arguments, et non de l'état de la mémoire	Objet, classes, méthodes, encapsulation, héritage, relations de composition, d'utilisation, …	Description des buts à atteindre à l'aide d'une syntaxe légère
LANGAGES	Fortran, C, Pascal	Lisp, Scheme, Caml	SmallTalk, C++, Java, Python	HTML, XML, LaTeX

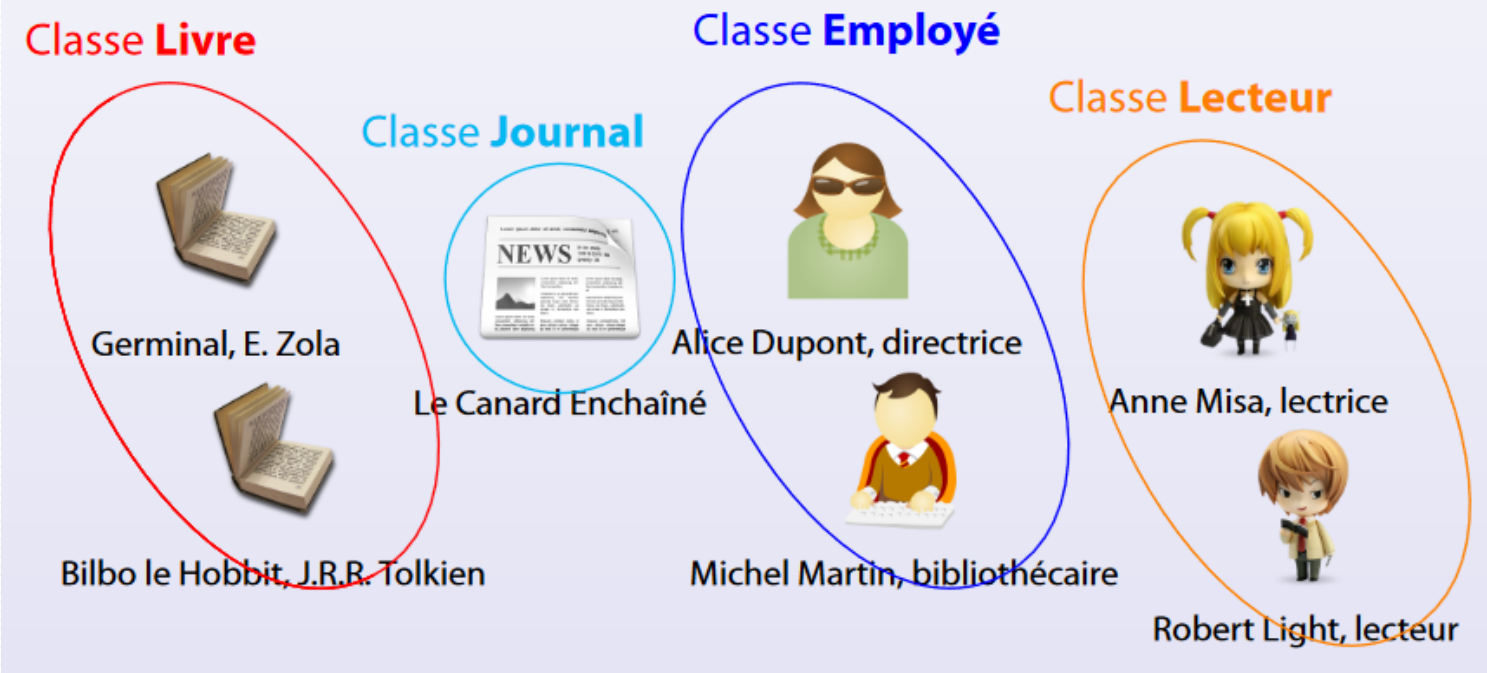
L'approche orientée objet

- Approche procédurale : « *Que doit faire mon programme ?* »
- Approche objet : « *De quoi doit être composé mon programme ?* »
  - Conséquence d'un choix de modélisation fait pendant la conception.



L'orienté objet

- Méthodologie centrée sur les données (**objets**)
- Chaque objet est un composant autonome
- Trio <objet, attributs, valeurs >



Le style objet

- Les objets représentent des données modélisées par des **classes** qui définissent des types
  - Un peu comme `typedef struct` en C

- Les classes définissent les actions que les objets peuvent prendre en charge et la manière dont les actions affectent leur état.
  - Ces traitements sont des **méthodes**

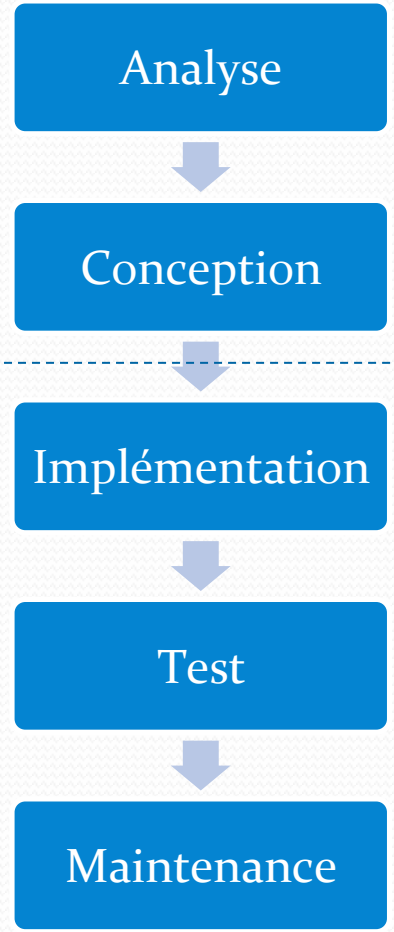
- Les données d'un objet sont appelés ses **attributs**

La programmation orientée objet

- Les objectifs :
  - Faciliter le développement, la maintenance, et l'évolution des applications;
  - Permettre le travail en équipe;
  - Augmenter la qualité des logiciels (moins de bugs).

- Solutions proposées :
  - Découpler (séparer) les parties des projets;
  - Limiter (et localiser) les modifications lors des évolutions;
  - Réutiliser facilement du code.

Modèle pour le développement logiciel



- **Unified modeling language (UML)**
  - Standard pour l'analyse et la conception orientée objet
  - Première version standard en 1997
  - Actuellement version 2.5

- **Java**
  - Pas un standard mais très largement utilisé
  - Première version standard en 1995
  - Dernière version stable : JDK 1.8 (Java 8)

Concepts abordés

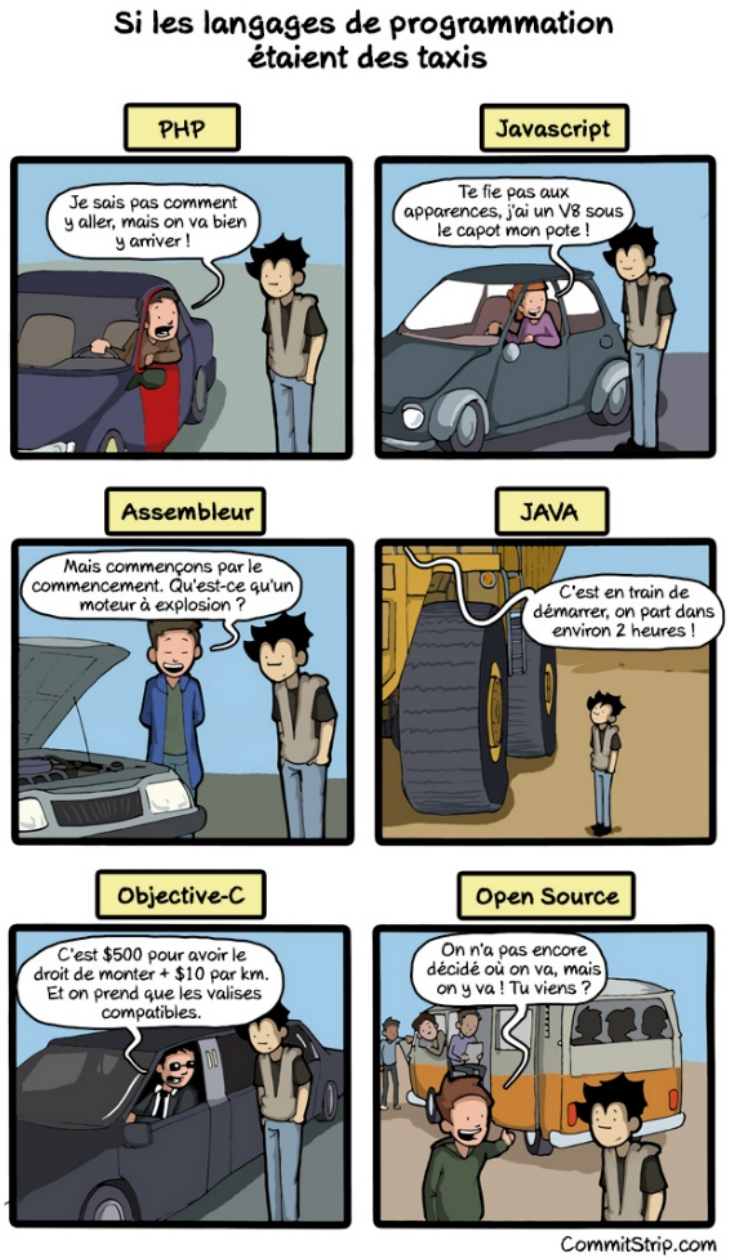
- Niveau conception/programmation
  - Instanciation d'objets à partir de classes
  - Encapsulation
  - Composition
  - Héritage
  - Polymorphisme
  - Généricité
  - Persistance
  - Tests unitaires

Le langage Java

Historique

- Créé en 1995 par *Sun Microsystems*
- *Oracle* rachète *Sun* en 2009 et détient désormais Java

- Essor du langage grâce à Internet (navigateurs web et applet):
  - Java JDK 1.01 et 1.02 en 1996, 1.1 en 1998
  - Java 2 (Playground) J2SE 1.2 en 1999 , (Kestrel) J2SE 1.3 en 2000, (Merlin) J2SE 1.4 en 2004
  - Java 5 (Tiger) J2SE 5.0 en 2004
  - Java 6 (Mustang) JSE 6.0 en 2006
  - Java 7 (Dolphin) JSE 7.0 en 2011
  - Java 8 JSE 8.0 en 2014



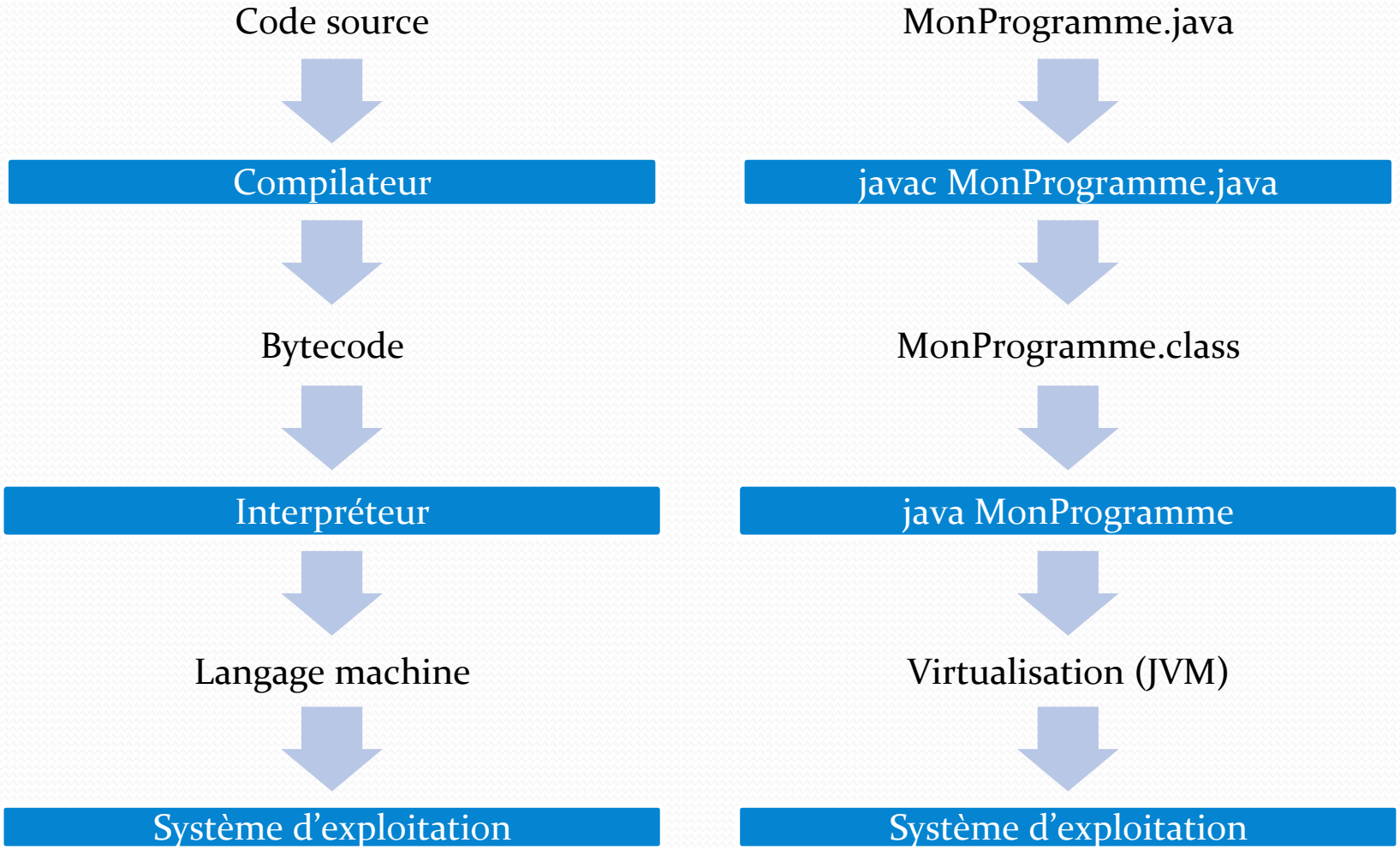
- Chaque langage a des avantages et des inconvénients.

- Java est :
  - Modulaire : *on peut écrire des portions de code « génériques »*
  - Rigoureux : *erreurs détectées plutôt à la compilation qu'à l'exécution*
  - Portable : *le programme compilé peut s'exécuter sur plusieurs plateformes*

Le langage Java

- En quelques mots :
  - Orienté Objet
  - Simple, Robuste, Dynamique et Sécurisé
  - Indépendant de la Plateforme (VM)
  - Semi Compilé/Semi Interprété
  - Fortement typé
  - Bibliothèque Importante (JDK API)

Langage compilé et interprété



Le bytecode

- Le **langage source Java** est défini par la JLS (*Java Language Specification*) éditée par Sun-Oracle
  - Syntaxe + sémantique

- Le code source d'une classe contenue dans un fichier est compilé avec la commande **javac**
  - cela produit un code intermédiaire, appelé **bytecode**

- Le bytecode d'une classe est destiné à être chargé par une **machine virtuelle** qui doit l'exécuter avec la commande **java**, par interprétation.
  - L'argument est le nom d'une classe (sans extension .class)

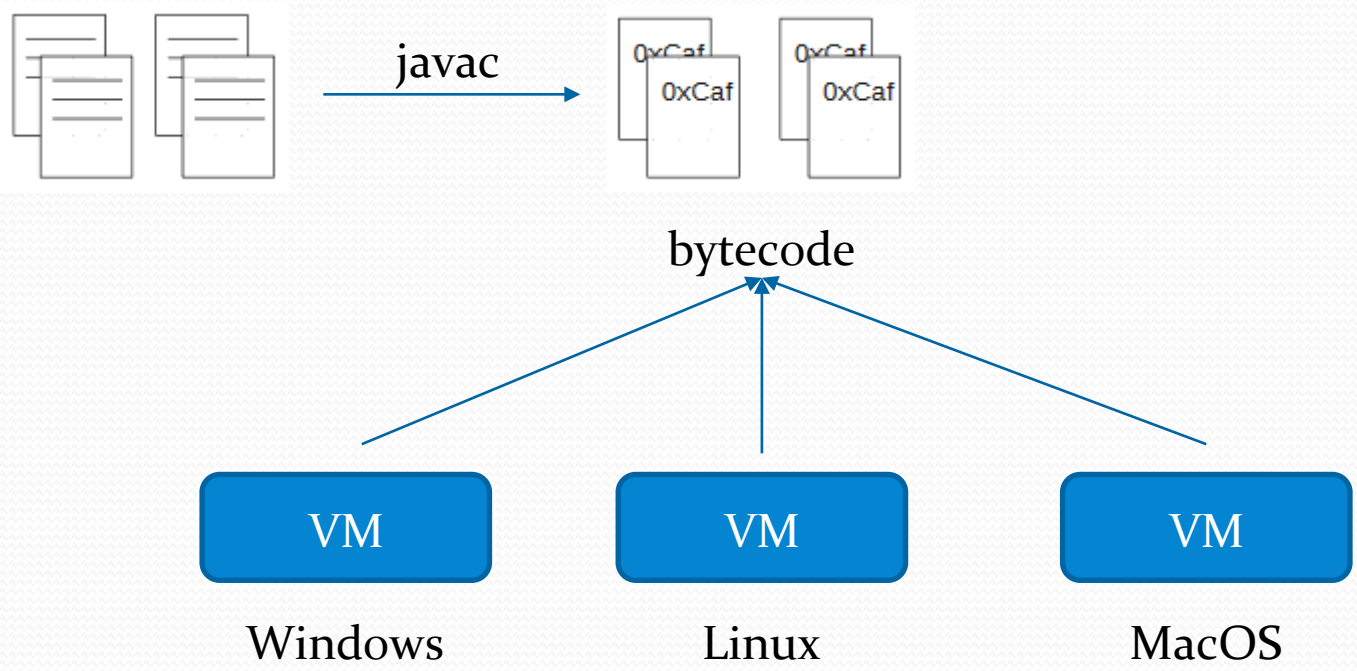
La machine virtuelle (JVM)

- **Rôle :** Abstraire le comportement d'une machine
- **But :** Rendre indépendant de la plateforme

- La JVM :
  - Garantit le même environnement d'exécution sur les différentes plateformes d'accueil (Windows, Linux, MacOS)
  - Optimise (comme un OS) l'exécution des applications en fonction de la machine

➔ Une JVM traduit le bytecode dans le langage machine de la plateforme d'accueil.

Portabilité entre différents environnements



Java : un langage et une plateforme

- Dans la technologie Java, on a besoin :
  - Du **langage de programmation** et du compilateur
  - De la **JVM** et des **APIs** (*Application Programming Interfaces*) regroupées dans une plateforme :
    - Java SE (*Java Platform, Standard Edition*) : Java SE 8 pour applications classiques, desktop
    - Java EE (*Java Platform, Enterprise Edition*) : Java EE 8 pour développer et déployer des applications serveur, Web services etc.
    - Java ME (*Java Platform, Micro Edition*) : J2ME pour les applications embarquées, PDA, smartphones, etc.
- Si l'on veut juste exécuter, il suffit du **JRE** (*Java Runtime Execution*) par opposition au **JDK** (*Java Development Kit*)

Diagramme conceptuel Java

http://docs.oracle.com/javase/8/docs