Master 2 LITL
Programmation pour le TAL (SLT0905V)
Langages de programmation, Java, Objets et UML : tour d'horizon

Franck Sajous/CLLE-ERSS

Séance 1

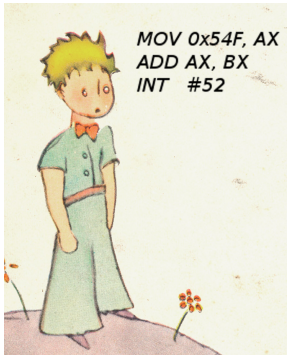
CLLE ERSS UMR 5263
COGNITION, LANGUES, LANGAGE, ERGONOMIE

http://fsajous.free.fr/

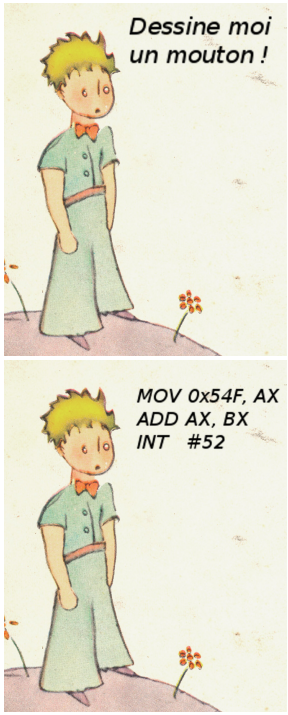
Sommaire

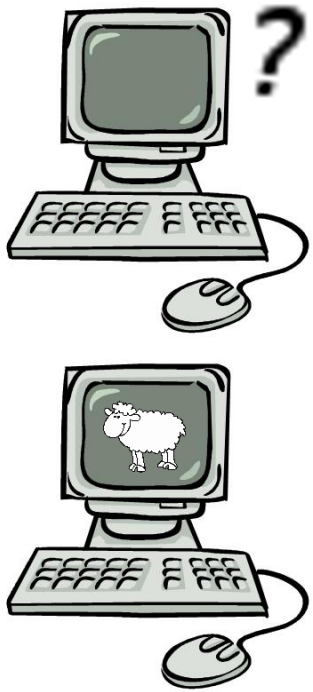
Communication homme-machine. . .

Communication homme-machine. . .



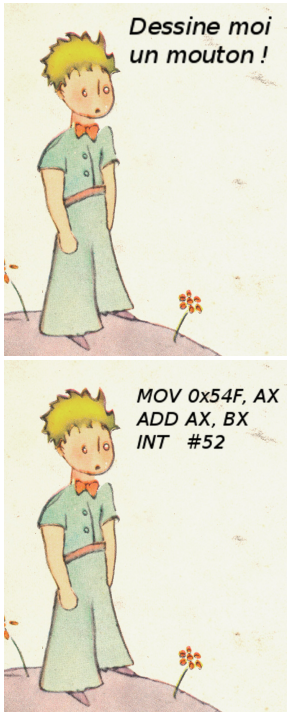
Communication homme-machine...

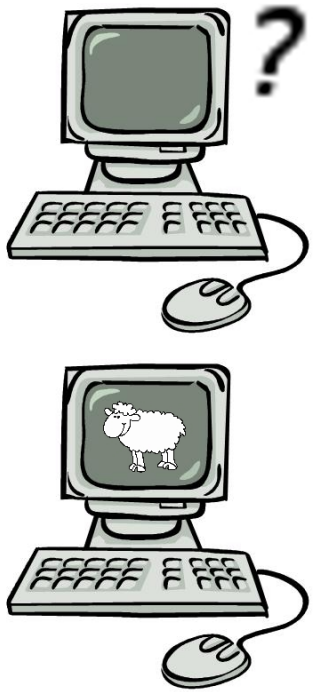




- *"Dessine-moi un mouton"* ≈ instructions dans un langage de programmation de haut niveau
- `MOV, ADD, INT...` ≈ instructions élémentaires d'un langage intermédiaire plus "parlant" pour la machine

Communication homme-machine...





- *"Dessine-moi un mouton"* ≈ instructions dans un langage de programmation de haut niveau
- `MOV, ADD, INT...` ≈ instructions élémentaires d'un langage intermédiaire plus "parlant" pour la machine
- exemple pas tout à fait réaliste... Il faut bien plus que 3 lignes d'assembleur pour produire ce résultat !
  sont aussi en jeu ici : OS, HD, driver de la carte vidéo, etc. Instructions intermédiaires : *ouvre le fichier* `mouton.jpg` *dans* `C:\\images`, *copie le contenu dans la mémoire vidéo*, etc. → différents degrés (couches) d'abstractions

Langage de programmation

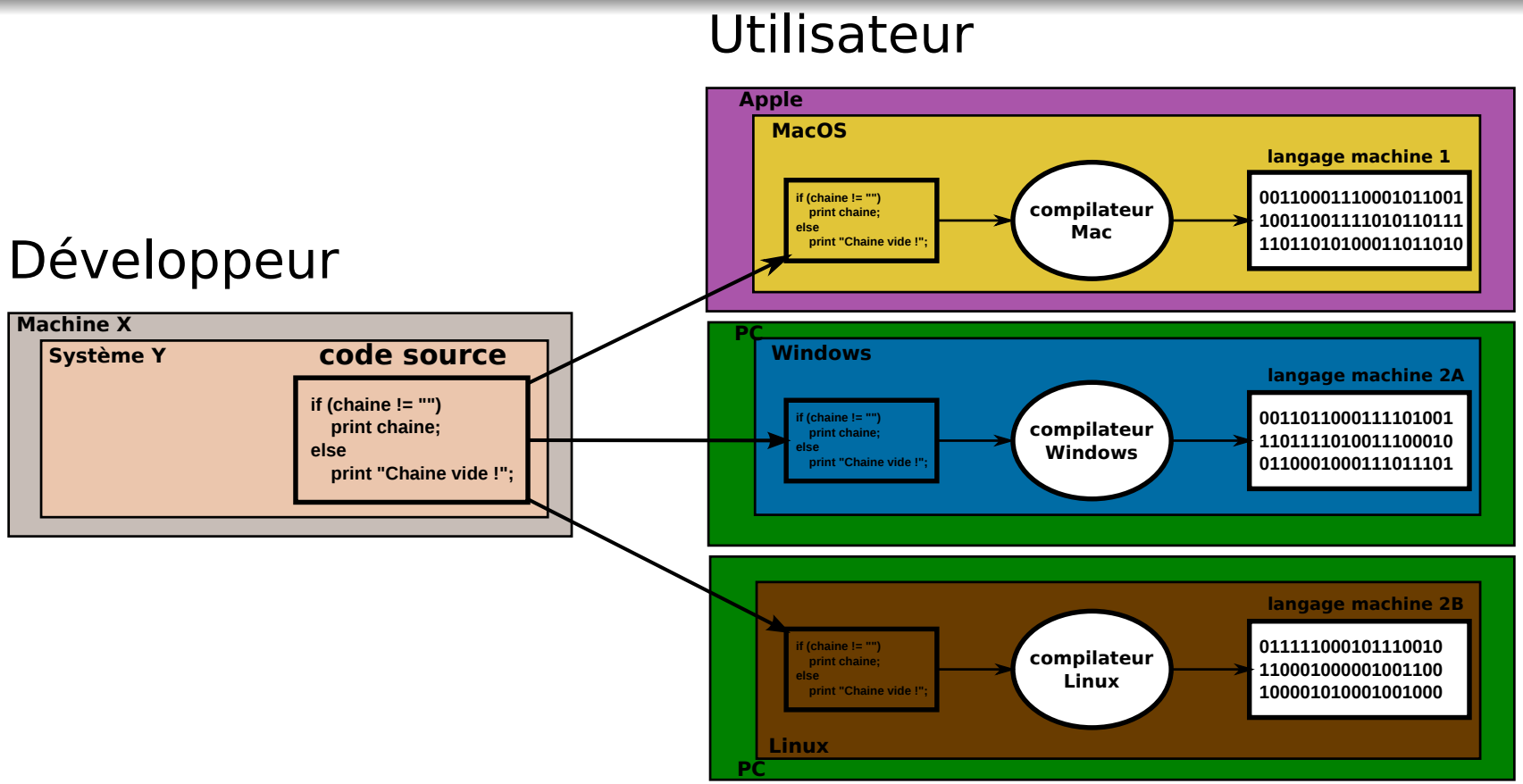
Éléments de définition

- Programme : ensemble d'instructions à exécuter pour réaliser une tâche
- Langage de programmation : ensemble des instructions possibles et façon de les organiser (assembler, ordonner) de manière à être interprétable par une machine
- Programme : désigne le code source (les instructions saisies par le développeur *"dessine-moi un mouton"*), stockées dans un fichier (et, *par ext.* le programme exécutable, le programme en cours d'exécution...)

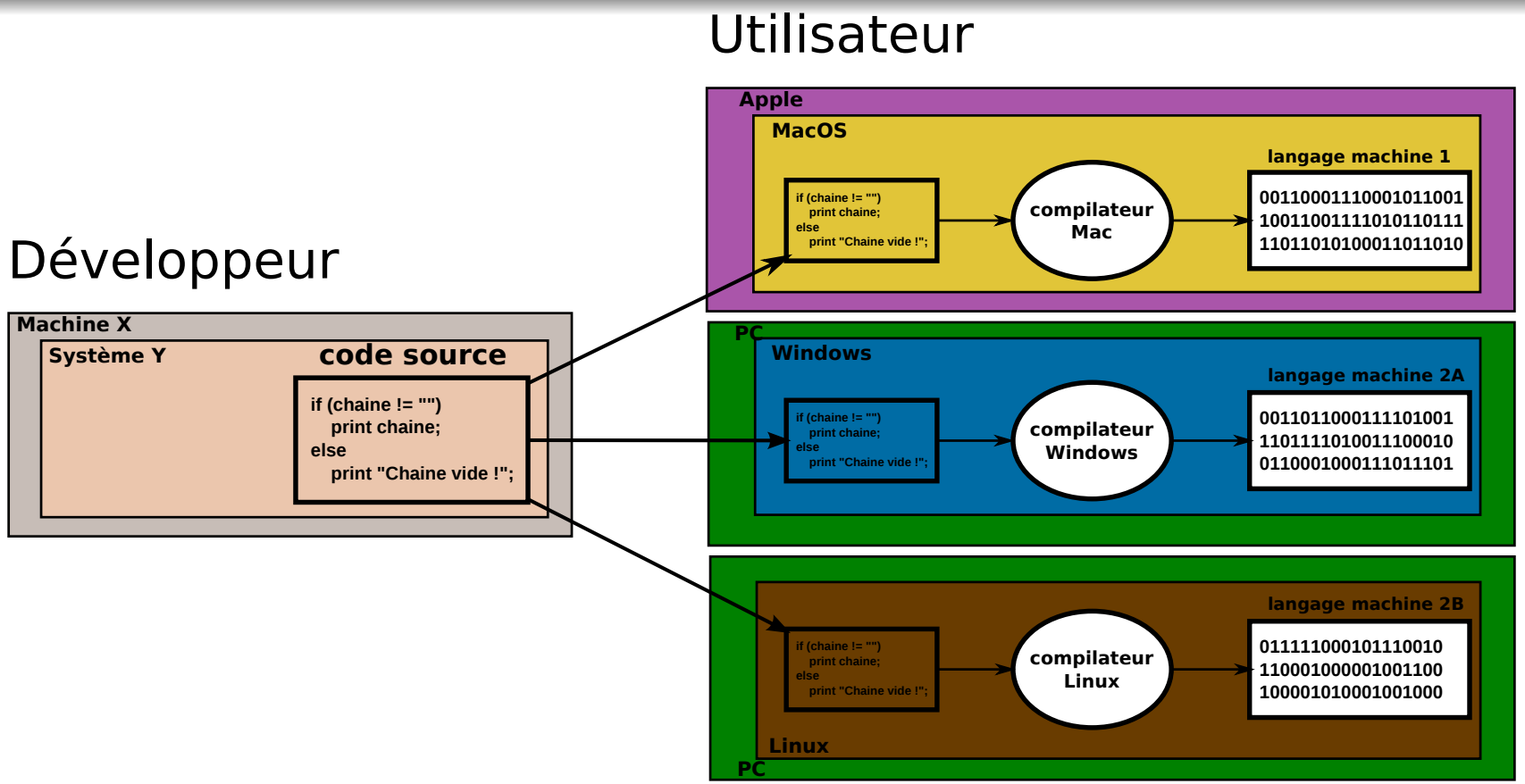
Programme source/programme exécutable

- Programme exécutable : ensemble d'instructions élémentaires qu'une machine peut exécuter directement (*langage machine*)
- Langage de programmation : (souvent) indépendant de la machine (langage de haut niveau)
- Langage machine : proche du matériel, toujours spécifique à la machine physique (langage de bas niveau)
- Entre le code source et le langage machine, une étape : la compilation

Compilation



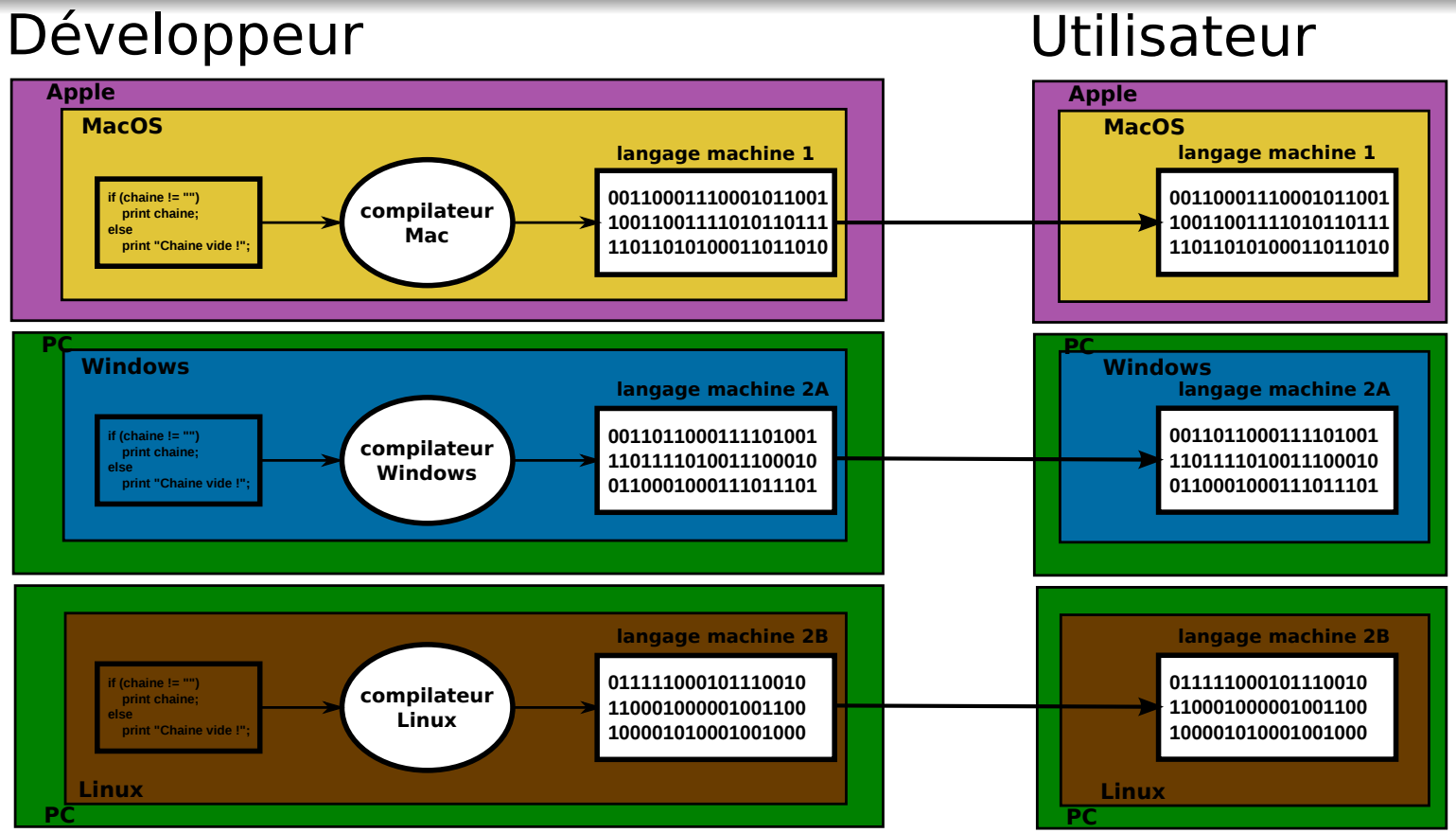
Compilation



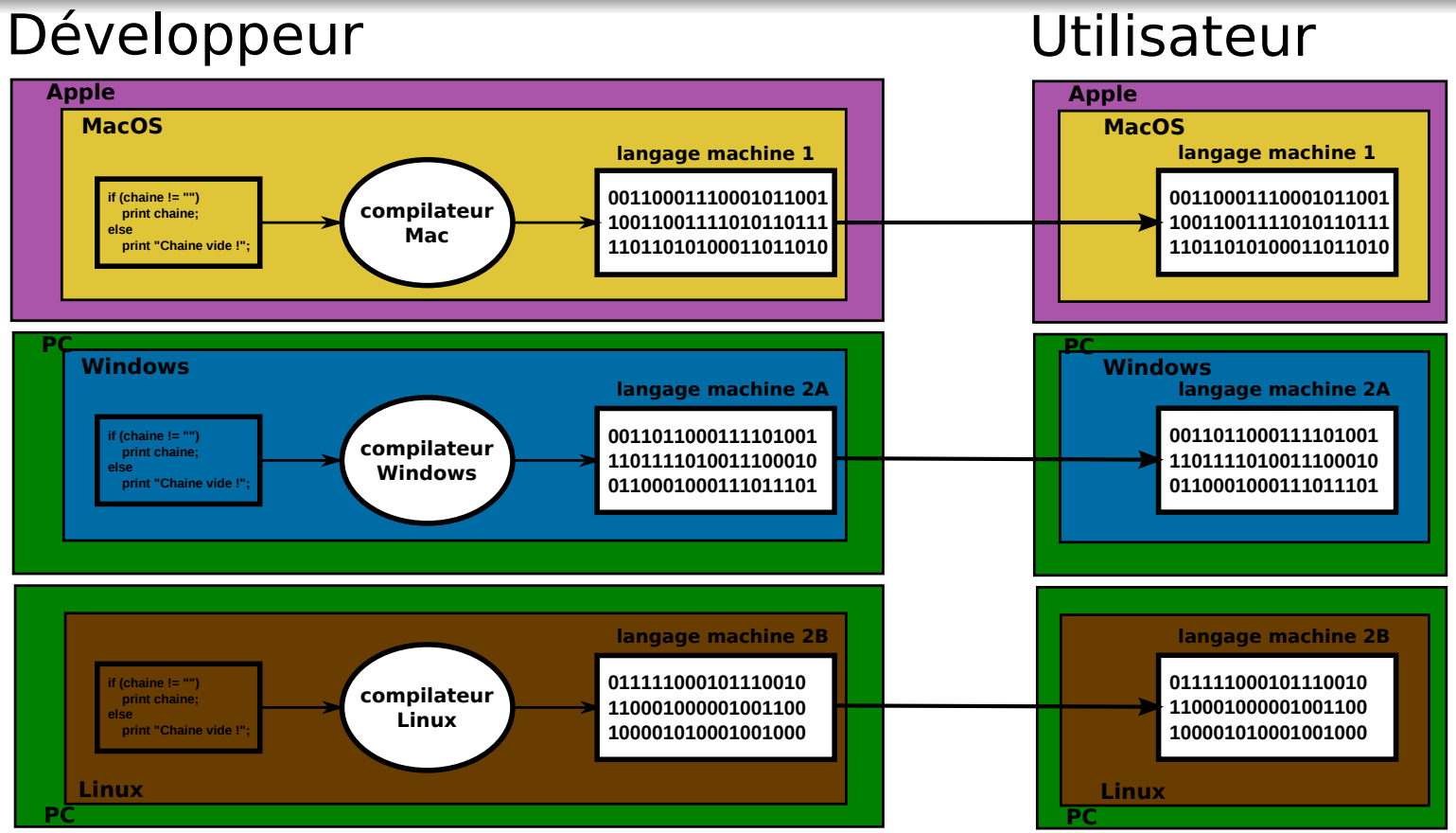
Logiciel libre (Open source)

- Distribution du code source (et éventuellement d'exécutable(s))
- L'utilisateur peut modifier le programme et le recompiler

Compilation (2)



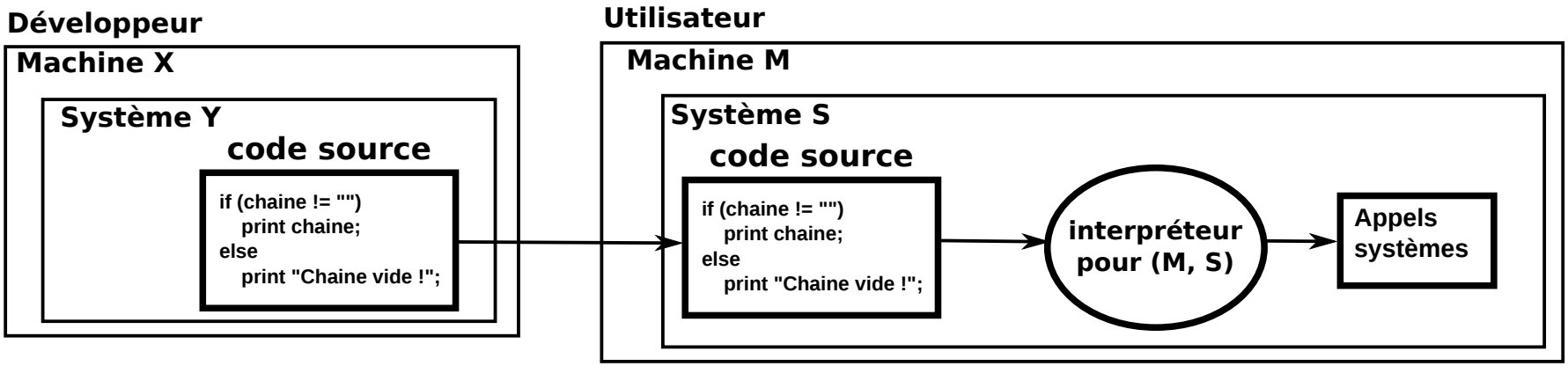
Compilation (2)



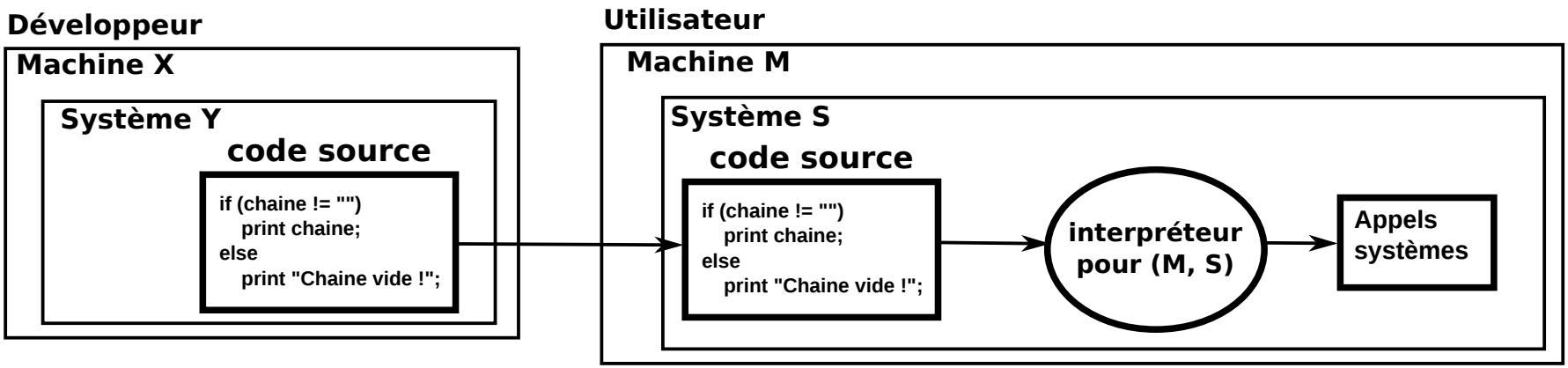
Logiciel propriétaire

- Distribution du programme sans le code source
- Nécessité de compilation sur chaque couple (machine, OS) pour produire autant d'exécutables que souhaité

La cas des langages interprétés (Perl, Python, etc.)



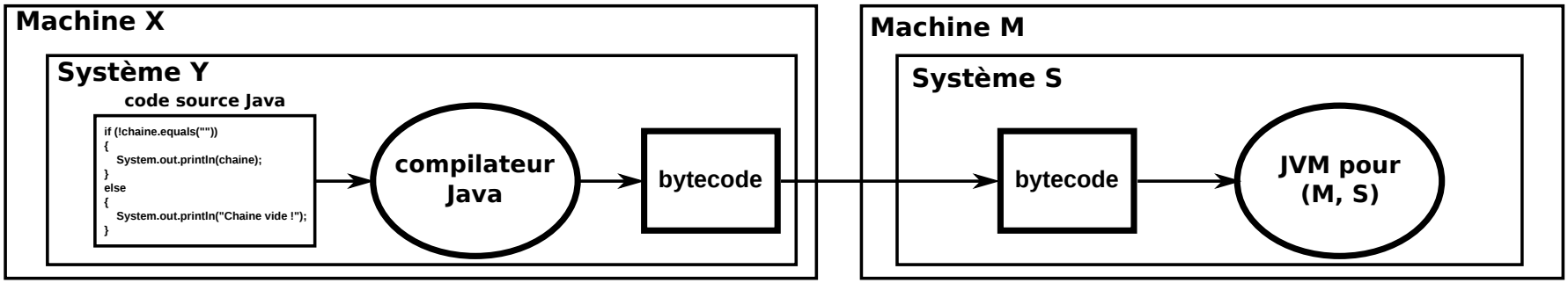
La cas des langages interprétés (Perl, Python, etc.)



Langages interprétés

- Ni le développeur ni l'utilisateur ne doivent effectuer de compilation
- L'interpréteur, spécifique à une machine et un OS, est installé une fois pour toutes (soit en même temps que l'OS, e.g. Perl sous Linux, soit par l'utilisateur)
- Exécution plus lente

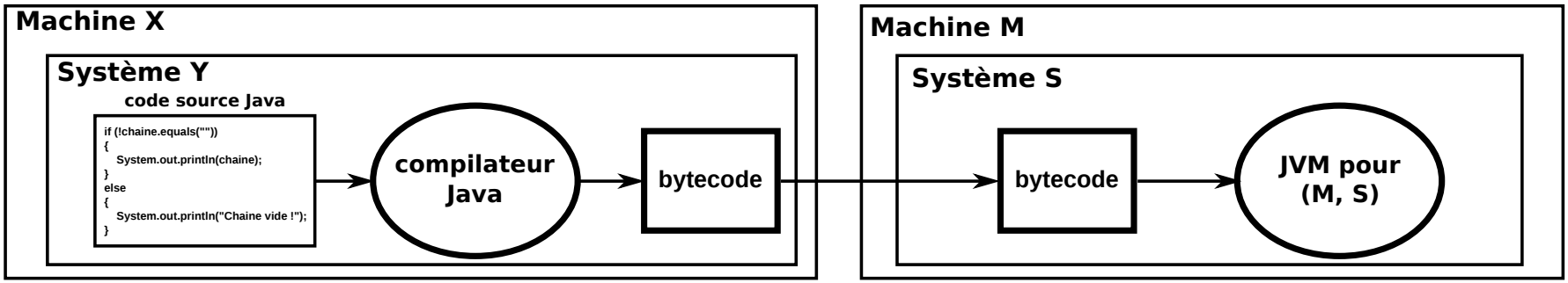
Java, bytecode et JVM



Un cas hybride : à la fois compilé et interprété

- Java compilé en bytecode ≈ langage machine pour... une machine virtuelle
- Java Virtual Machine (JVM) : interprète le bytecode

Java, bytecode et JVM



Un cas hybride : à la fois compilé et interprété

- Java compilé en bytecode ≈ langage machine pour... une machine virtuelle
- Java Virtual Machine (JVM) : interprète le bytecode
- Portable, le même bytecode peut être distribué pour plusieurs plateformes *"write once, run anywhere"*
- L'utilisateur qui récupère le bytecode n'a pas besoin de le compiler
- Le bytecode, langage intermédiaire plus « concret », nécessite moins d'actions pour être interprété par la JVM qu'un langage interprété classique ($\rightarrow$ plus lent qu'un langage compilé, plus rapide qu'un langage interprété classique)

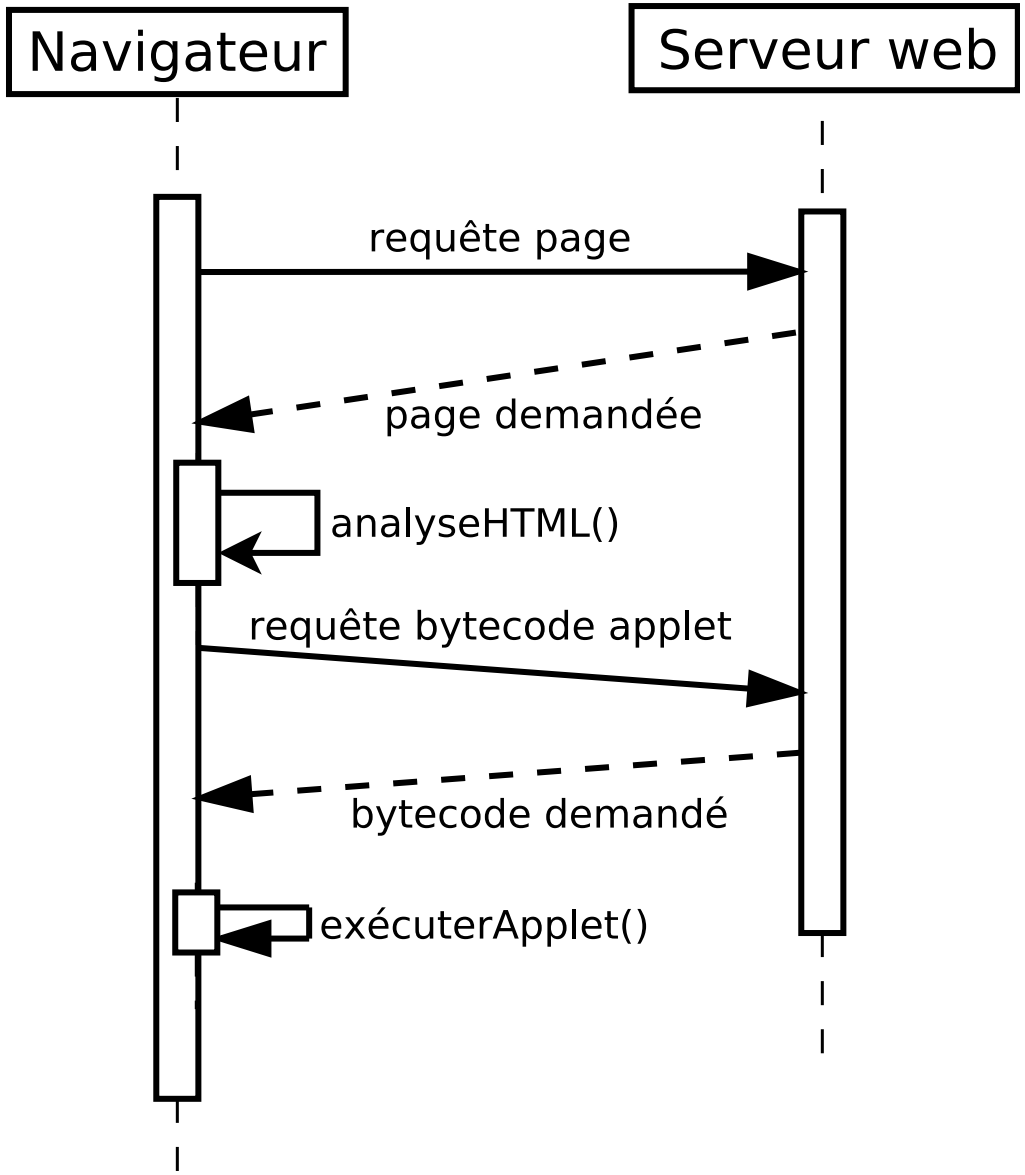
Environnements d'exécution

Différentes applications, différents environnements

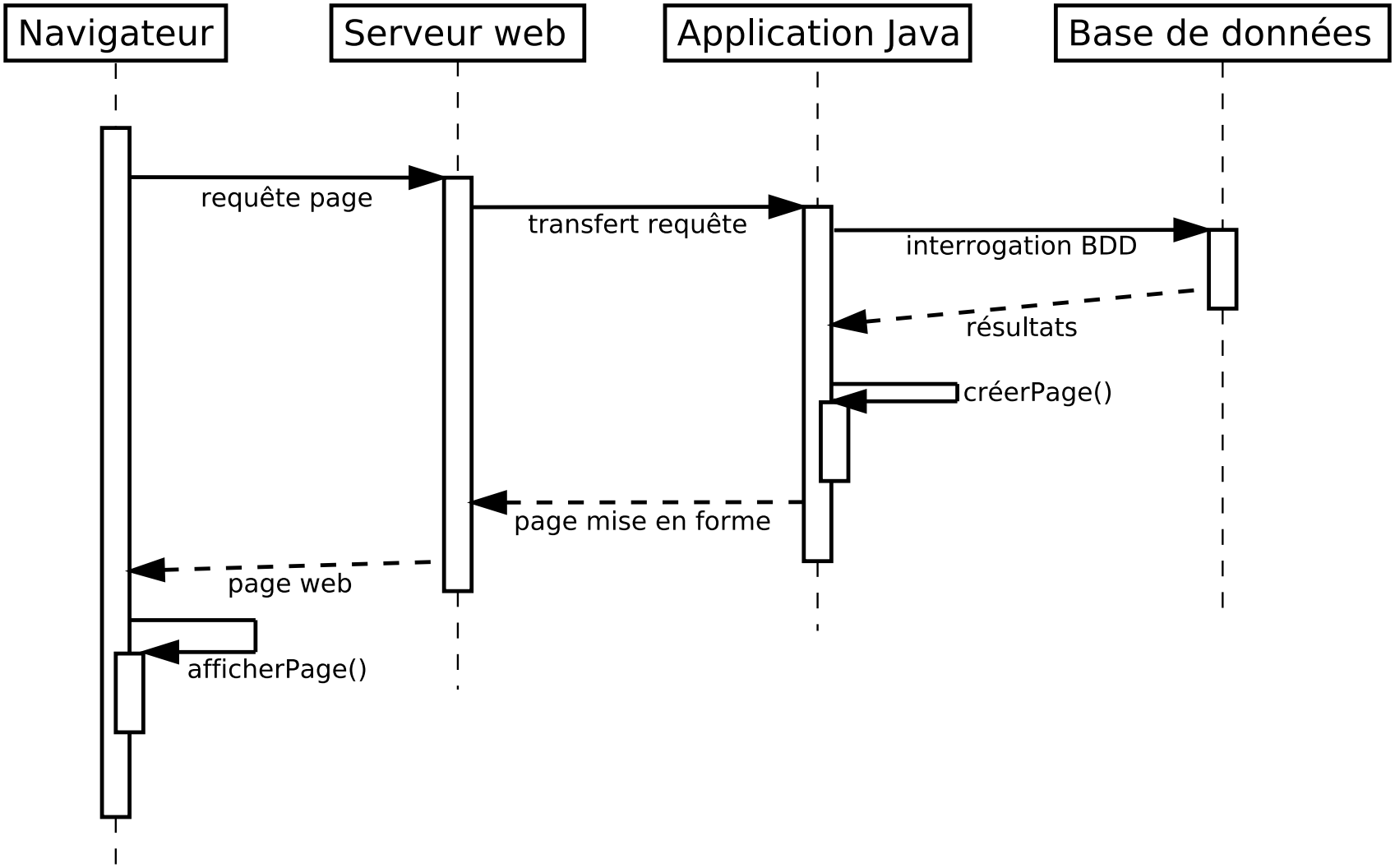
- Les débuts : Java créé au tout début des 90s par Sun Mircosystems, initialement pour des systèmes embarquées. Présenté officiellement en 1995 comme langage de programmation
- Applications *standalone* : on récupère le bytecode et on l'exécute « en local » (e.g. : Eclipse, Oxygen, Talismane)
- Applets : programme java qui s'exécute dans la fenêtre d'un navigateur.

  `http://mainline.brynmawr.edu/Courses/cs110/spring2002/Applets/Examples.html`
- Programmation côté serveur
  - un navigateur demande une page à un serveur web
  - le serveur web interagit avec une application Java
  - cette application construit la page « à la demande », éventuellement en interagissant avec une base de données
  - le serveur web « sert » la page construite au navigateur
  - e.g. commerce électronique, Archipel (BU), Leximédia 2007, bases interactives « voisins de », GlàffOLI :
    - `http://redac.univ-tlse2.fr/LexiMedia2007/`
    - `http://redac.univ-tlse2.fr/voisinsdelemonde/`
    - `http://redac.univ-tlse2.fr/glaffoli/`
- Smartphones : Android

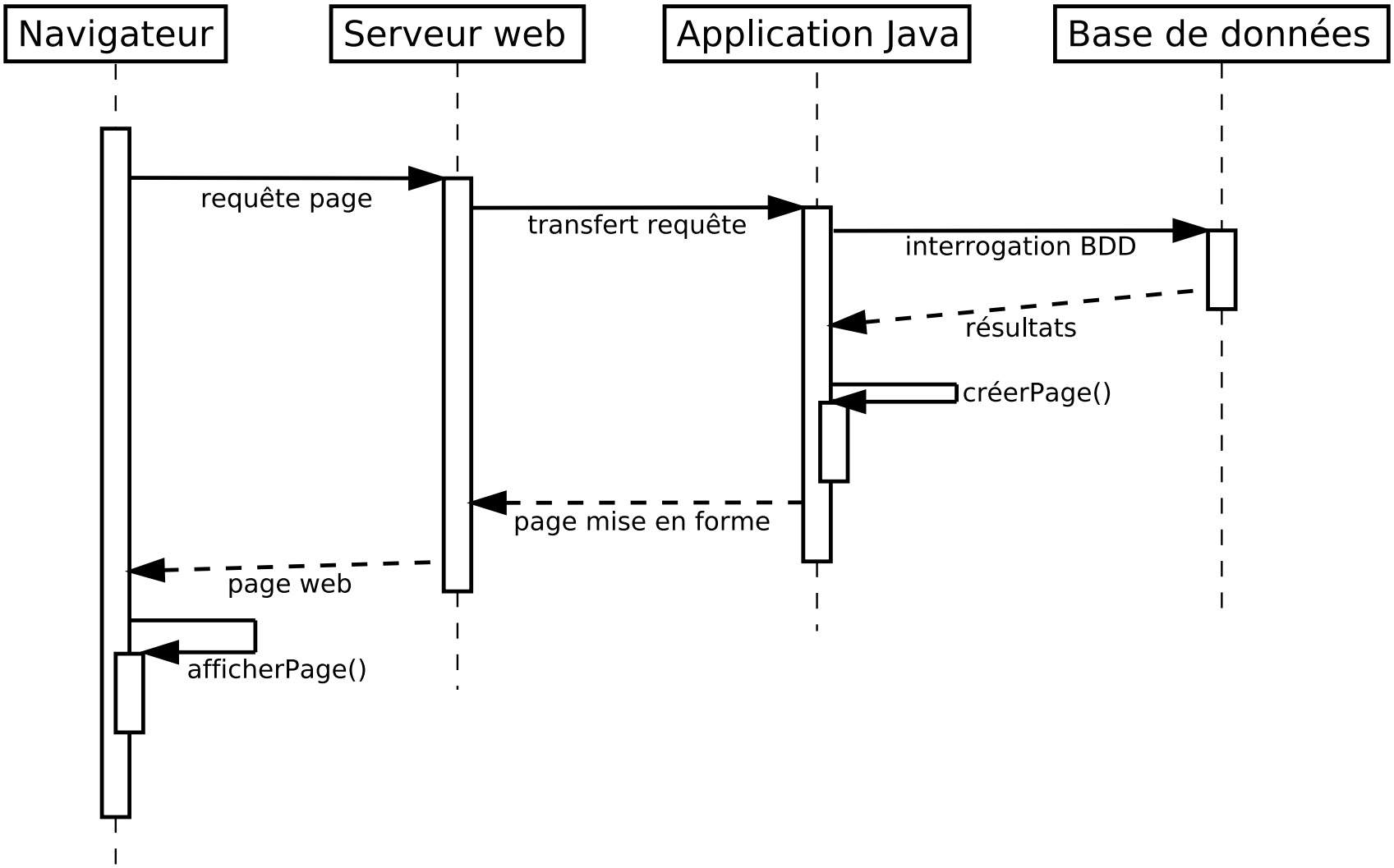
Applets


Navigateur
Serveur web
requête page
page demandée
analyseHTML()
requête bytecode applet
bytecode demandé
exécuterApplet()

Java côté serveur

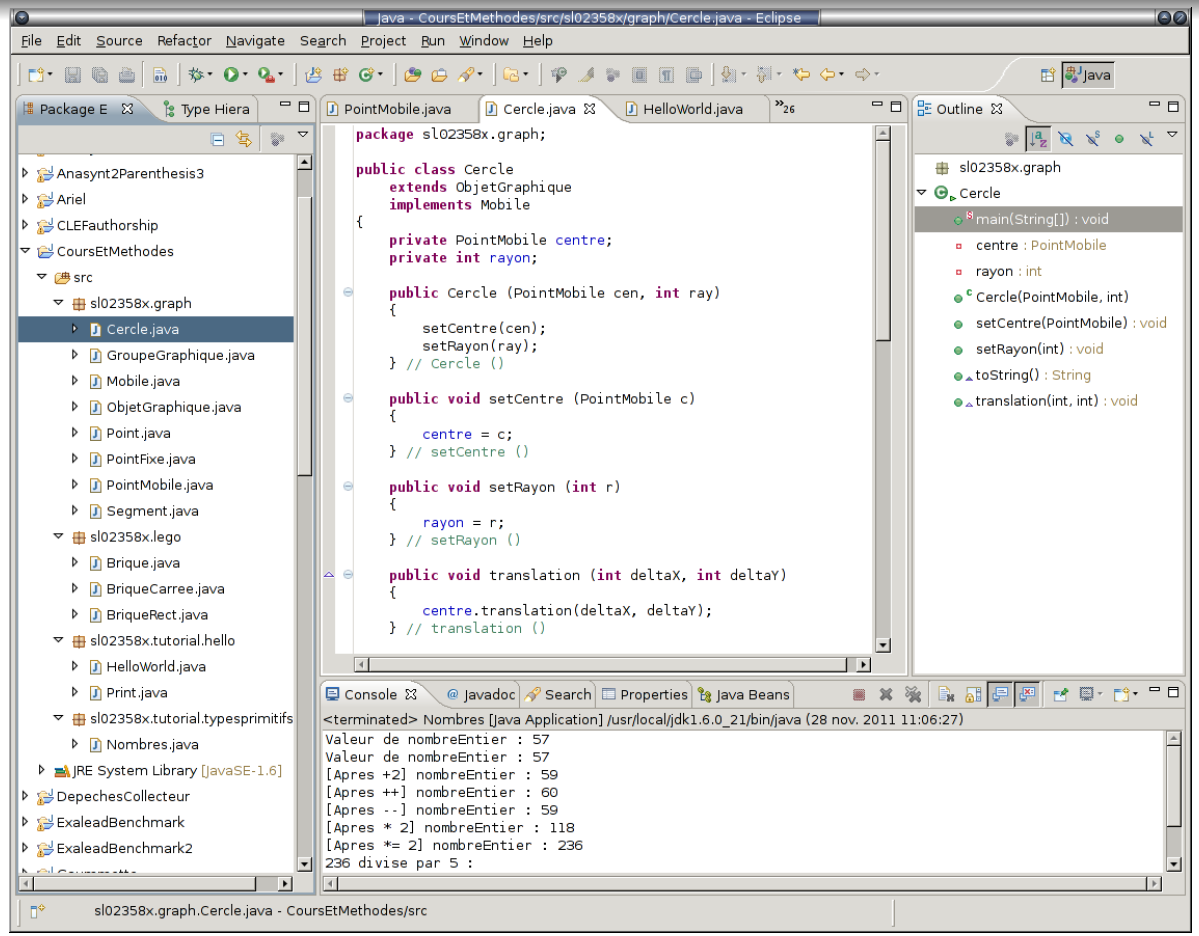


Java côté serveur



Ceci est un diagramme de séquence UML

Dans le cadre de l'U.E.



Eclipse

- Eclipse = éditeur + compilateur + JVM
- Développeur = utilisateur = vous

Types de langages

Langages impératifs procéduraux

e.g. C, FORTRAN (compilés), Perl (interprété) : séquences d'instructions + procédures, qui opèrent sur des données et disent à la machine ce qu'elle doit faire (et comment)

Langages logiques, déclaratifs

e.g. Prolog : prédicats et systèmes d'inférence

Langages fonctionnels

e.g. Caml, Lisp : déclaration de fonctions mathématiques (récursives), Lambda-calcul

Langages objets

e.g. Java, C++, ADA, Eiffel, SmallTalk : prog. structurée, similitude avec les langages procéduraux, effet légo, encapsulation des données, héritage, polymorphisme, etc.

Types de langages

Langages impératifs procéduraux

e.g. C, FORTRAN (compilés), Perl (interprété) : séquences d'instructions + procédures, qui opèrent sur des données et disent à la machine ce qu'elle doit faire (et comment)

Langages logiques, déclaratifs

e.g. Prolog : prédicats et systèmes d'inférence

Langages fonctionnels

e.g. Caml, Lisp : déclaration de fonctions mathématiques (récursives), Lambda-calcul

Langages objets

e.g. Java, C++, ADA, Eiffel, SmallTalk : prog. structurée, similitude avec les langages procéduraux, effet légo, encapsulation des données, héritage, polymorphisme, etc.

Quel est le meilleur langage ?

Types de langages

Langages impératifs procéduraux

e.g. C, FORTRAN (compilés), Perl (interprété) : séquences d'instructions + procédures, qui opèrent sur des données et disent à la machine ce qu'elle doit faire (et comment)

Langages logiques, déclaratifs

e.g. Prolog : prédicats et systèmes d'inférence

Langages fonctionnels

e.g. Caml, Lisp : déclaration de fonctions mathématiques (récursives), Lambda-calcul

Langages objets

e.g. Java, C++, ADA, Eiffel, SmallTalk : prog. structurée, similitude avec les langages procéduraux, effet légo, encapsulation des données, héritage, polymorphisme, etc.

Quel est le meilleur langage ?

Mauvaise question !

POO : principes

Propriétés

- maintient les principes de la programmation structurée
- introduit la notion d'objets, de classes, d'encapsulation, d'héritage...
- facilite la réutilisabilité du code, le développement en équipe

Adaptation du paradigme *programme = instructions + données*

- prog. structurée : structures de données et procédures ($\approx$ fonctions)
- POO : modélisation d'objets du monde réel ou de concepts abstraits
  Objet = membres/attributs ($\approx$ données) + méthodes ($\approx$ fonctions)
- Principes fondamentaux :
  - Encapsulation : un objet manipule ses propres données uniquement.
    En POO pure, un objet X n'a accès aux données d'un objet Y qu'à travers les méthodes de Y qui constituent son "interface".
  - Abstraction des données (implémentation masquée) : on ne connaît pas le détail de l'implémentation de la méthode d'un objet que l'on utilise

  $\rightarrow$ exemples à venir