Introduction au langage JAVA

Programmation JAVA – Cours 1
L2 informatique – 2013/2014

Denis Payet

Plan

- 1) Organisation des cours
- 2) Introduction à la Programmation
- 3) Les variables
- 4) Les types

JAVA - Historique

- Créé en 1994/1995, par la société SUN
- Langage de programmation inspiré du C++
- Initialement conçu pour réaliser des programmes interactifs sur le Web
- Libre (licence GPL) depuis novembre 2006
- Racheté par Oracle en 2010
- Version actuelle (juillet 2011) : Java 7

1) Organisation des cours

Ce qu'il faut savoir pour programmer

- Comment écrire du code (des instructions) :
  - les concepts clés pour coder
  - la syntaxe à utiliser pour écrire du code ?
  - les concepts plus évolués pour produire un code efficace

Cours n°1
Cours n°2
Cours n°3

- Comment structurer le code :
  - dans un fichier source, où taper le code ?

Cours n°4

1ère évaluation

- comment le code source se répartit dans les fichiers ?

Cours n°5

Et pour **bien** programmer en Java ?

- Quels outils utiliser ? — Cours n°6
- Pratiquer & encore pratiquer — Les TP !
- Maîtriser la philosophie du langage — Cours n°7
- Les subtilités de cette philosophie — Cours n°8
- Connaître les librairies existantes — Cours n°9

2ième évaluation

- Suivre l'évolution du langage — Cours n°10

2) Introduction à la programmation

La programmation

- Définition :

Un programme est un code écrit à destination d'un microprocesseur (CPU). Ce code définit le traitement que doit réaliser le CPU.

Le code se compose d'une suite d'instructions que le CPU traite l'une après l'autre.

=> Le code doit être écrit dans un langage compréhensible par le CPU

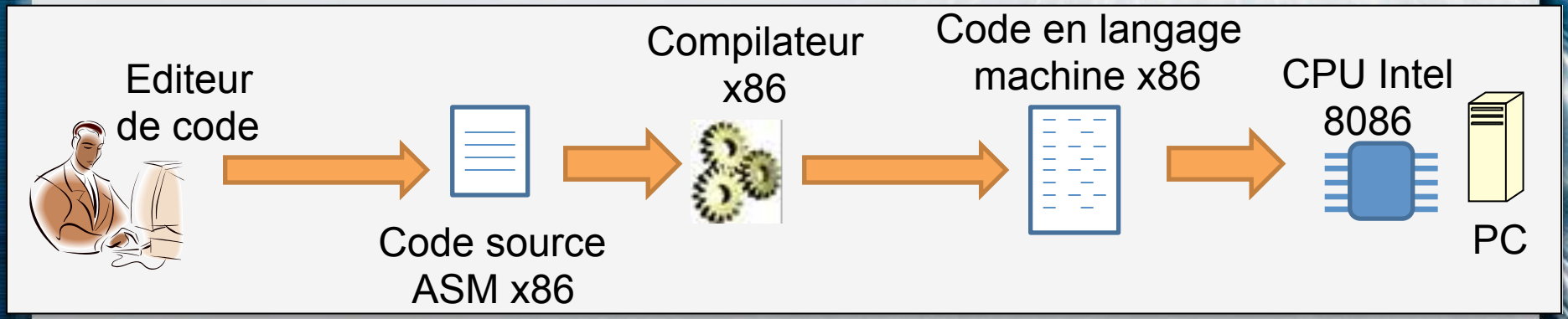
Edition / Compilation / Exécution

Le seul langage que comprend le CPU est le langage machine !

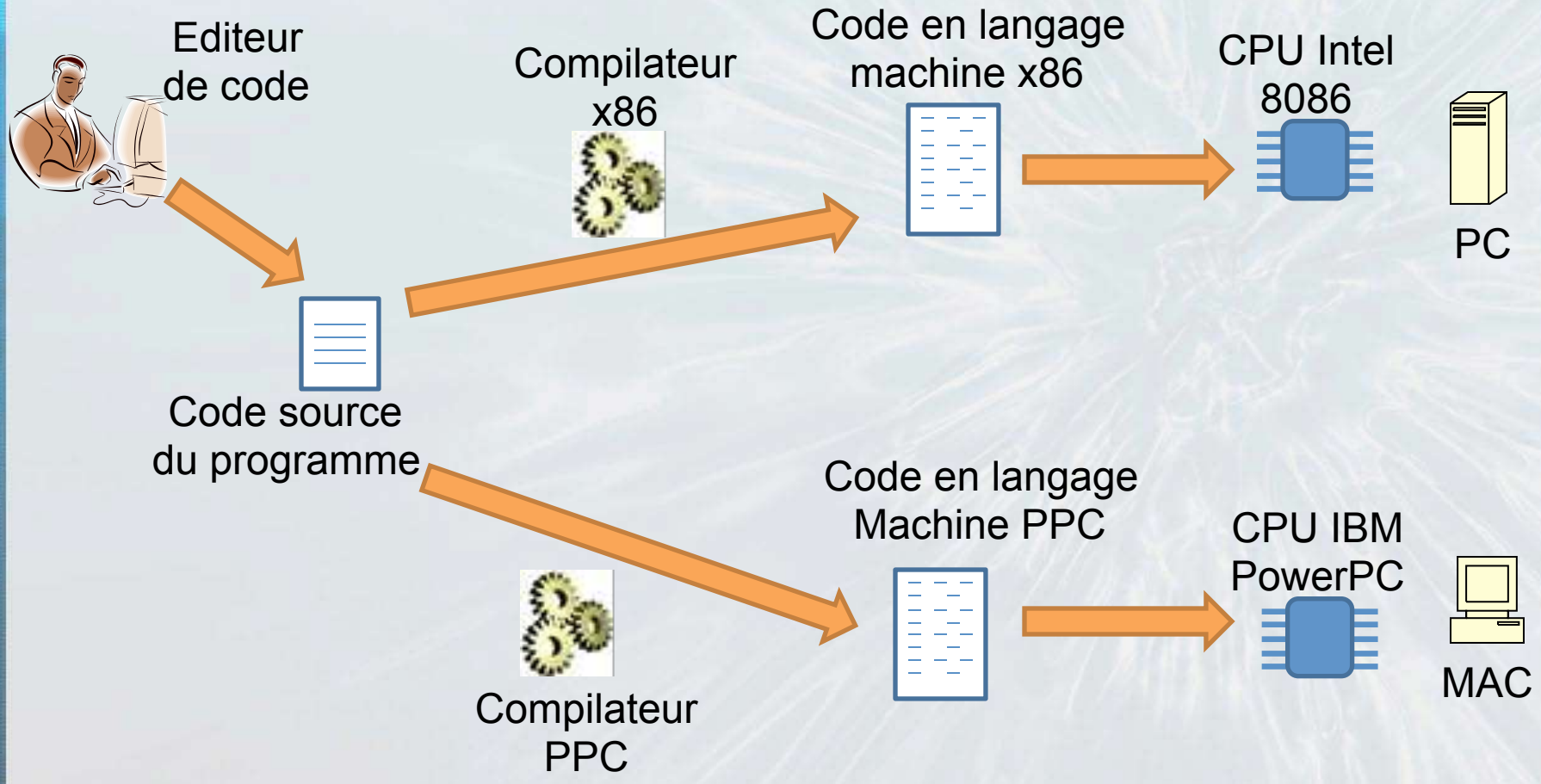
→ Le langage machine correspond à l'écriture binaire de la suite d'instructions à réaliser, directement compréhensibles par le sous-circuit "décodeur d'instruction" du CPU.

→ L'homme ne sait pas écrire en binaire, aussi on utilise un langage d'abstraction pour écrire le code source des programmes.

→ Une fois écrit, il faut utiliser un compilateur pour traduire le code source en langage machine.



CPU différents => codes machines différents



Philosophie du langage JAVA

- Langage Interprété

  => Indépendance vis-à-vis de la plateforme matérielle

  => nécessite la présence d'un logiciel spécial, dit machine virtuelle (JVM) pour fonctionner.

  (fourni dans le JRE)

- Langage 100% Objet

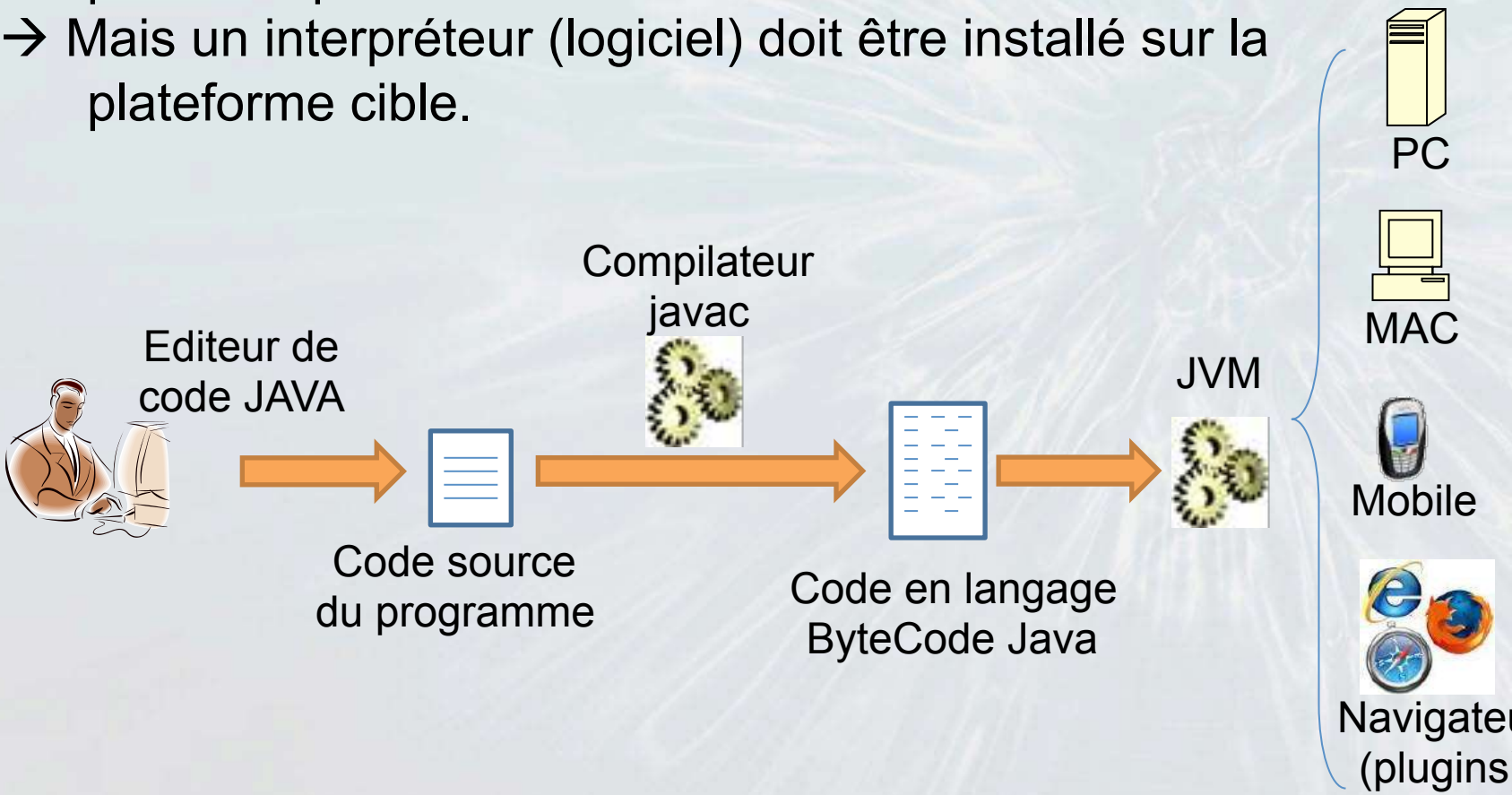
  => Tout code fait partie d'un objet

  (nous verrons plus tard ce qu'est un objet en POO)

Les langages interprétés (Script, Java,...)

→ Le même code source peut être utilisé, **sans le recompiler**, sur plusieurs plateformes matérielles différentes.

→ Mais un interpréteur (logiciel) doit être installé sur la plateforme cible.



L'Environnement Java

- Une plateforme de développement :
  - Une machine virtuelle (JVM)
  - Des librairies de base (API)
- Un ensemble d'outils (le JDK) :
  - Compilateur
  - Debugger
  - Documentation
  - …

Que fait un programme ?

- Un programme réalise un traitement sur un ensemble de données
- Ce traitement consiste à modifier les données, les supprimer ou en créer de nouvelles

3) Les variables

Que fait un programme ? (suite)

- Les données sur lesquels le programme travaille peuvent provenir :
  - d'un support de stockage (disque, BDD,…)
  - d'un autre programme (service réseau, appareil de mesure,…)
  - de l'utilisateur lui-même (via le clavier, la souris,…)
- Les nouvelles données produites par le programme sont :
  - stockées sur un support (disque, BDD,…)
  - transmises à un autre programme (service réseau,…)
  - transmises à l'utilisateur (via l'écran, haut-parleurs,…)

Travailler sur des données ?

- En réalité c'est le CPU qui travaille sur les données, en suivant les directives exprimées par le programme.
- Pour pouvoir travailler, ces données doivent :
  - être accessibles par le CPU
  - pouvoir être désignées dans le code programme

Les données sont placées dans la mémoire de l'ordinateur (chargée, si nécessaire, depuis le disque ou le réseau)

Dans le programme, les données sont désignées par l'**adresse** à laquelle elles sont stockées dans la mémoire.

Rappel : organisation d'une mémoire

- Une mémoire peut être vue comme une armoire dans laquelle chaque tiroir représente une *case* mémoire, c'est-à-dire un espace pouvant contenir une donnée (ou une partie d'une donnée).
- Chaque tiroir est identifié par un numéro appelé une **adresse**.

Adresse	case mémoire
0100	0000 1001
0011	0110 1111
0010	0010 1011
0001	1100 0000
0000	1110 1111

→ Chaque donnée devient alors accessible grâce à son adresse

→ Le nombre de case mémoire associé à une donnée définit la plage des valeurs que peut prendre cette donnée

Des adresses mémoire ?

- La mémoire (vive) est un support volatile, quand il n'y a plus de courant, elle s'efface.
- C'est un support temporaire, elle ne sert qu'à « rapprocher » les données du processeur pour qu'il puisse travailler avec.
- A chaque exécution d'un programme, la mémoire disponible n'est pas la même, aussi les mêmes données ne sont pas stockées à la même adresse à chaque lancement.

Les variables !

- Problème 1 : comment spécifier les adresses des données dans le programme si elles ne sont pas les mêmes d'une exécution à l'autre ?
- Problème 2 : les adresses ne sont que des chiffres, comment le programmeur peut-il reconnaître les données sur lesquelles il exprime un traitement ?

En utilisant des variables !

→ L'adresse d'une donnée est représentée par **le nom** de la variable

→ Une table interne est utilisée pour faire correspondre chaque nom à une adresse

→Le nombre de cases mémoires associées à une variable définit la plage des valeurs qui peuvent être affectées à cette variable. Dans le langage de programmation ces plages de valeurs sont représentées par **des types**

Les variables - définition

Une variable est une entité symbolique qui désigne une zone mémoire qui va servir à stocker une donnée manipulée par le programme.

Dans le code la variable est représentée par son **nom** (une chaine alphanumérique), et possède un **type** qui spécifie la *nature* de la valeur stockée dans la zone mémoire.

Pour nous, cette *nature* peut être : une valeur numérique entière ou décimale plus ou moins grande, un caractère, une chaîne de caractère, un tableau…

→ Mais pour la machine, ce n'est que le nombre de cases mémoire que peut occuper la donnée, et chaque case contiendra une valeur numérique (binaire).

Le nom des variables

- Maximum 247 caractères
- Caractères possibles : les lettres (minuscules & majuscules), les chiffres et le caractère _
- Le nom ne doit pas commencer par un chiffre
- Le nom ne peut pas être un mot réservé du langage

Les mots réservés du langage =
les noms des **types primitifs**
et les **mots-clés** du langage
(nous les verrons dans les différents cours)

Un bon programmeur utilisera des noms de variable explicite pour que son code soit compréhensible !

4) Les types

Type primitif / Type complexe

- Un type primitif est un type qui est prédéfini dans le langage et dont le nombre de case mémoire correspondant est constant.
- Un type complexe est un type qui peut s'exprimer par une combinaison d'autres types. Le nombre de case mémoire qui lui est associé varie en fonction de de la structure de cette combinaison et/ou de la taille de l'information qui lui sera affectée.

  par exemple: les tableaux,
  les chaines de caractère, …

Les types des variables

- Les 8 types primitifs:

Nom	Signification	Taille (octets)	Valeurs possibles
char	Caractère	2	0 à 65535 (caractères Unicode)
byte	Entier très court	1	-128 à 127
short	Entier court	2	- 32 768 à 32 767
int	Entier	4	-2 148 483 648 à 2 147 483 647
long	Entier long	8	-/+ 9,2 milliard de milliard
float	Flottant (nombre réel)	4	$-/+ 3,4*10^{38}$
double	Flottant double	8	$-/+ 1,7*10^{308}$
boolean	booléen	1	true ou false

NB: il n'y a pas de type non signé explicite

Le type **char**

char : ce type est utilisé pour des données qui représente des caractères. Il sert par exemple à faire des traitements sur les textes affichés sur l'écran, les touches du clavier, etc… Exemple de caractère : 'a', 'A', '0', '1' , '+' …

→ Il ne faut pas confondre un nombre et un caractère qui représente ce nombre: **1** n'est pas **'1'**

➔ **On utilise des quotes ' ' pour spécifier un caractère**

Mais : c'est nous (humain) qui donnons un sens aux caractères, pour le CPU ce ne sont que des nombres. Ce sens (nombre <-> caractère) est spécifié par une table de correspondance appelée table ASCII.

La table ASCII

- La table ASCII est une table qui indique quelle valeur numérique correspond à chaque caractère.
- Elle est utilisée par les fonctions d'affichage pour savoir quelle lettre dessiner à chaque valeur d'une case mémoire censée contenir une donnée de type caractère.
- Dans cette table :
  - **'a'** et **'A'** sont des caractères différents
    → ils n'ont pas la même valeur ASCII
  - Le caractère **'1'** n'a pas la valeur 1
  - Les caractères qui se suivent ont des valeurs qui se suivent (les minuscules d'une part et les majuscules d'autre part).
    → la valeur de **'c'** est égale à la valeur de **'a'** + 2
    → la valeur de **'C'** est égale à la valeur de **'A'** + 2
    qui est aussi égale à: **'c' + ('A' - 'a')**

Le type **boolean**

boolean est un type particulier, il est utilisé pour une donnée qui ne peut prendre que 2 valeurs : vrai ou faux.

→ Contrairement à d'autres langages, en java on ne considère pas que des données boolean ont une correspondance numérique (même si en réalité elles en ont inévitablement une).

➔ **0 ne représente pas faux**

Dans ce langage de programmation les valeurs vrai et faux pour ce type s'exprime par les mots clés: **true** et **false**

Compatibilité des types

- Définition :

  Un type A est compatible avec un type B si et seulement si la plage de valeur de A est comprise dans la plage de valeur de B.

Exemple :

- le type **int** est compatible avec le type **double**
- le type **char** est compatible avec le type **int**
- le type **double** n'est pas compatible avec le type **int**
- le type **boolean** n'est compatible avec aucun autre type
- aucun type n'est compatible avec le type **boolean**

Les constantes littérales

- Une constante littérale est une valeur numérique, booléenne, un caractère ou une chaine de caractères directement saisie dans le code source
- Constante numérique :

  51 → entier, type int, exprimé en base 10

  051 → entier, type int, exprimé en base 8 (octal)

  0x51 → entier, type int, exprimé en base 16 (hexadécimal)

  51.0 51. .78 32.41 21.13e7 3.7e-33 → nombre décimal, **de type double !**

  51.0f .78f 3.7e-33f → nombre décimal, **de type float !**
- Constante booléenne : **true** **false**
- Constante caractère :

  'a' 'B' → code ASCII des lettres, type char
- Constante chaine de caractère : **"Un exemple de chaine"**

Fin.