Langages de programmation
Notes de cours

Maria-Iuliana Dascalu, PhD

mariaiuliana.dascalu@gmail.com

www.mariaiulianadascalu.com

Université Polytechnique de Bucharest
Département d'Ingénierie en Langues étrangères

Évaluation

- La note finale = E1+E2+E3+E4+E5
  - E1 - examen écrit (en session) : 40%
  - E2 - test écrit partiel (11-12 nov): 20%
  - E3 - activité pendant les travaux pratiques (en class + devoirs): 20% + 20%
  - E4 - tests inopinés pendant le cours: 5%
  - E5 – bonus (pour participation exemplaire dans les cours): 5%
  - La note finale > 10!!!
- Pour passer:
  - exigences minimales pour entrer dans l'examen:
    - 10 présences en TP (Les TP’s ne peuvent être refaits!!!)
    - E2 >=4.5 (Le test peut être répété dans la dernière semaine de cours!!!)
    - E3>=4.5
  - la note finale >=4,5 et E1 >=4.5

References

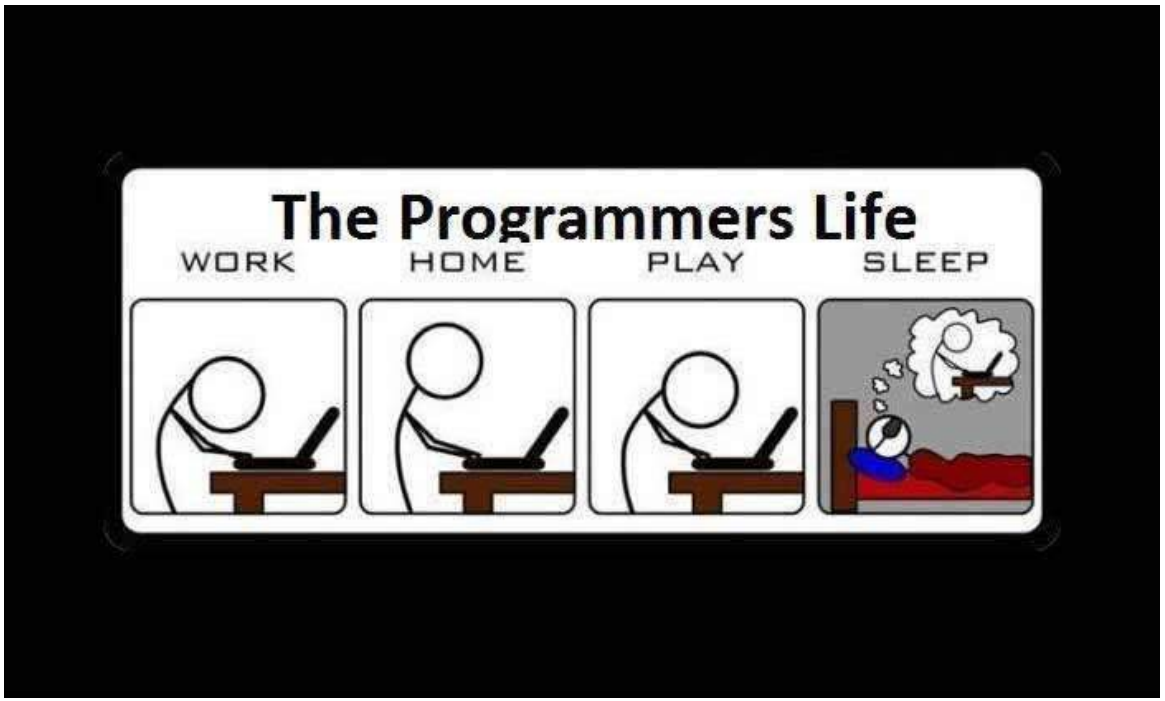
- Dascalu, M.I., “Langages de programmation - notes de cours”: **https://mariaiulianadascalu.com -> Teaching - > Langages de programmation/ Moodle**
- http://docs.oracle.com/javase/tutorial/
- http://penserenjava.free.fr/
- http://bruce-eckel.developpez.com/livres/java/traduction/tij2/
- Horstmann, C., G. Cornell, “Core Java 2”
- Mughal, K., Rasmussen, R. “Programmer's Guide to Java SCJP Certification”
- Dragoi, G., “Langages de programmation - notes de cours” (2010-2014)
- Serbanati, L.D., et Bogdan, C.M.,”Programarea orientata spre obiecte cu exemplificari in limbajul Java”, vol. 1, Polipress, pag. 249, ISBN: 978-606-515-109-3 (2010) (en roumain)

Si vous avez des questions…

- Cours: mariaiuliana.dascalu@gmail.com
  - le sujet de e-mail doit être: [LP] [numéro de groupe] [nom de student]; example: LP 1210F Popescu Ion
  - bureau: Decanat (mardi, 11-12, jeudi, 12-14)
  - **https://mariaiulianadascalu.com -> Teaching - > Langages de programmation/ Moodle**

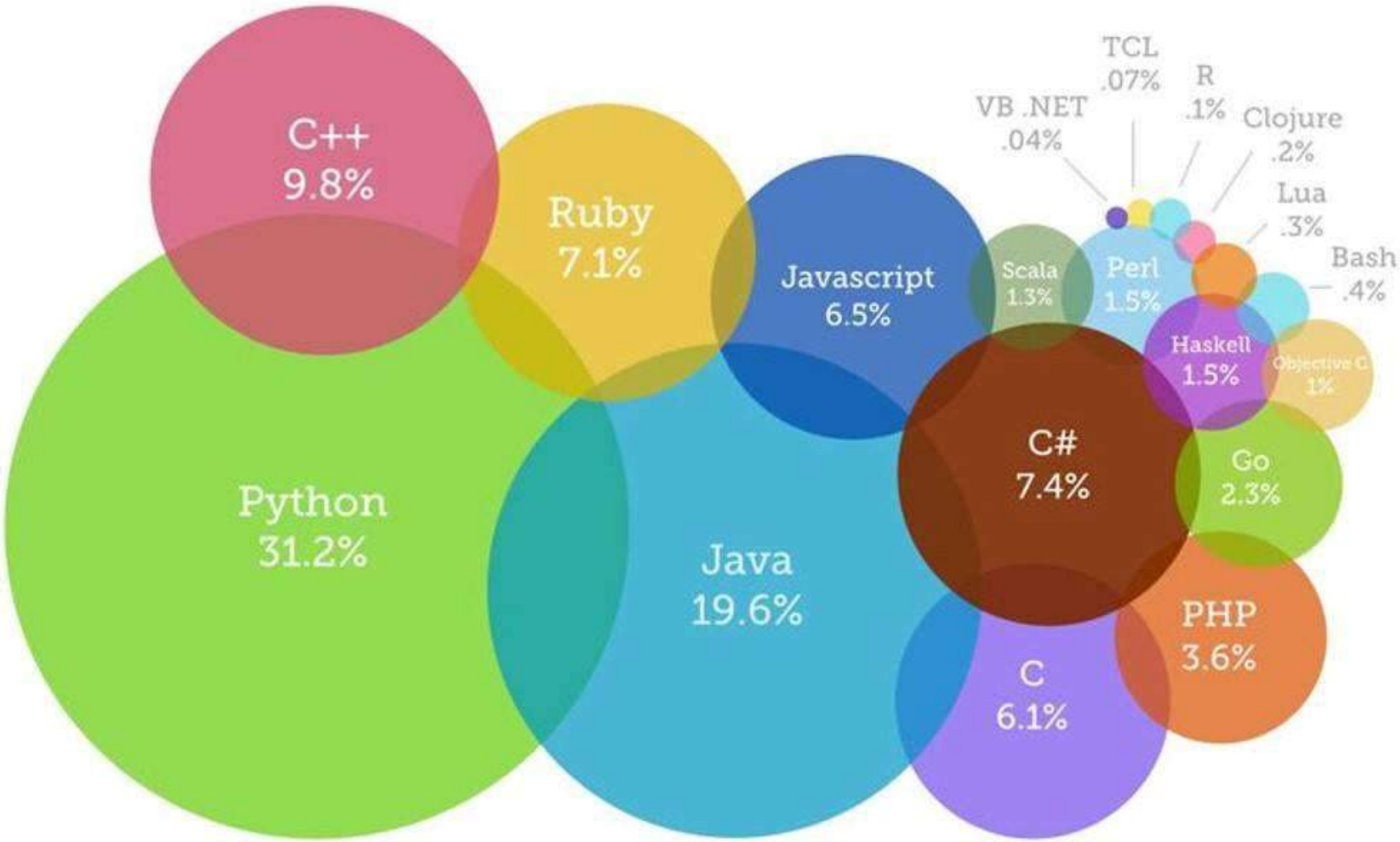
- TP:
  - Iulia Stanica (pour les groupes de CTI de lundi): iulia.stanica@gmail.com/ Moodle
  - Cosmin Radu (pour CTI de mercredi): raduioancosmin@yahoo.com/ Moodle
  - Alexandru Mitrea (pour CTI de jeudi et les groupes de ETTI de jeudi): danalexmitrea@gmail.com/ https://sites.google.com/site/javafils/

Objectifs



Objectifs

- être initieés à l'informatique et en tout premier lieu à la programmation (les concepts et principes fondamentaux)
- apprendre Java, choisi comme le langage «support» :
  - notions de base, types de données, variables
  - expressions, structures de contrôle, méthodes, récursions
  - concepts de base de la programmation orientée objet
  - relations entre les classes: association, héritage
  - polymorphisme
  - les classes abstraites
  - I/O en Java
- vous devriez être capable, à l'issue de ce cours, de vous familiariser assez rapidement avec un autre langage de programmation que Java!


C++
9.8%
Ruby
7.1%
Javascript
6.5%
TCL
.07%
R
.1%
VB .NET
.04%
Clojure
.2%
Lua
.3%
Bash
.4%
Scala
1.3%
Perl
1.5%
Haskell
1.5%
Objective C
1%
C#
7.4%
Go
2.3%
Python
31.2%
Java
19.6%
PHP
3.6%
C
6.1%

Objectifs pour aujourd'hui

- Introduction aux langages de programmation
- Introduction à Java
  - Développement et l'exécution d'une application de programme Java
  - Types et variables
  - Types de données en Java
  - Eléments de langage

Introduction aux langages de programmation

- **L'informatique** est une science de l'**abstraction** = il s'agit de créer le bon **modèle** pour un problème et d'imaginer les bonnes techniques automatisables et appropriées pour le résoudre.
- Un **modèle** est représenté par **un programme** et manipulé dans un ordinateur.
- Les programmes sont **des listes d'instructions** pour le processeur d'ordinateur, définissant des opérations à réaliser sur **des données**.
- Un programme indique à un ordinateur, dans les moindres détails, la séquence des étapes qui sont nécessaires pour accomplir une tâche. L'ordinateur n'a pas l'intelligence des séquences d'instructions-il exécute simplement qui ont été préparées à l'avance.
- L'ordre des instructions est le flot d'exécution ou **flot de contrôle.**
- Pour écrire des programmes, on se sert d'une notation, appelée **langage de programmation**:
  - langues naturelles (anglais, français) vs. langues artificielles (construits)
    - langages formels -> langages de programmation (syntaxe et sémantique)

Syntaxe et sémantique des langages de programmation

- **La syntaxe** est un ensemble de principes et de règles pour construire des programmes corrects dans un langage de programmation; ces règles sont la «grammaire» et les programmes sont les «phrases» du langage de programmation.
- **La sémantique** donne sens pour les constructions syntaxiques.
- La syntaxe est basée sur les composants de phrases: un programme est composé de pièces où les pièces peuvent être agrégées à partir d'autres pièces et ainsi de suite. Ces pièces sont des éléments syntaxiques de la langue: déclaration, expression, variable, l'opérateur et ainsi de suite.
- John **B**ackus et Peter **N**aur introduits pour la première fois en 1960 une notation **f**ormelle pour décrire la syntaxe d'une langue donnée: **BNF**.

Les méta-symboles de la BNF sont:

- **::=** signifiant “est défini comme”
- | significant “ou”
- **< >** utilisé pour entourer les noms des catégories

Les équerres(**< >**) distinguent les noms des règles de syntaxe (**symboles non-terminaux**) de **symboles terminaux** qui sont écrites exactement comme ils doivent être représentés.

```
<conditional instruction> ::= if (<condition>)
                                  <instruction>
                              | if (<condition>)
                                  <instruction>
                              else
                                  <instruction>
```

```
<conditional instruction> ::=  if (<condition>)
                                  <instruction>
                               [ else
                                  <instruction> ]
```

éléments optionnels sont enfermés dans les méta symboles [et]

```
<identifier> ::= <letter> { <letter> | <digit> }
<identifier> ::= <letter> <letter>*
<identifier> ::= <letter>+
```

* *est zéro ou plusieurs fois*
\+ *est 1 ou plusieurs fois*

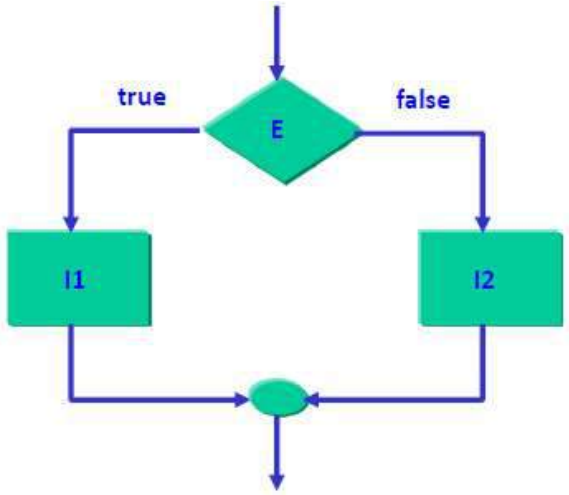
```
<statement_sequence> ::= <statement> { ";" <statement> }
```

Définition de l'instruction conditionnelle

Syntaxe:

```
<conditional instruction> ::= if (<condition>)
                                <instruction>
                              else
                                <instruction>
```

Sémantique :



Système de type

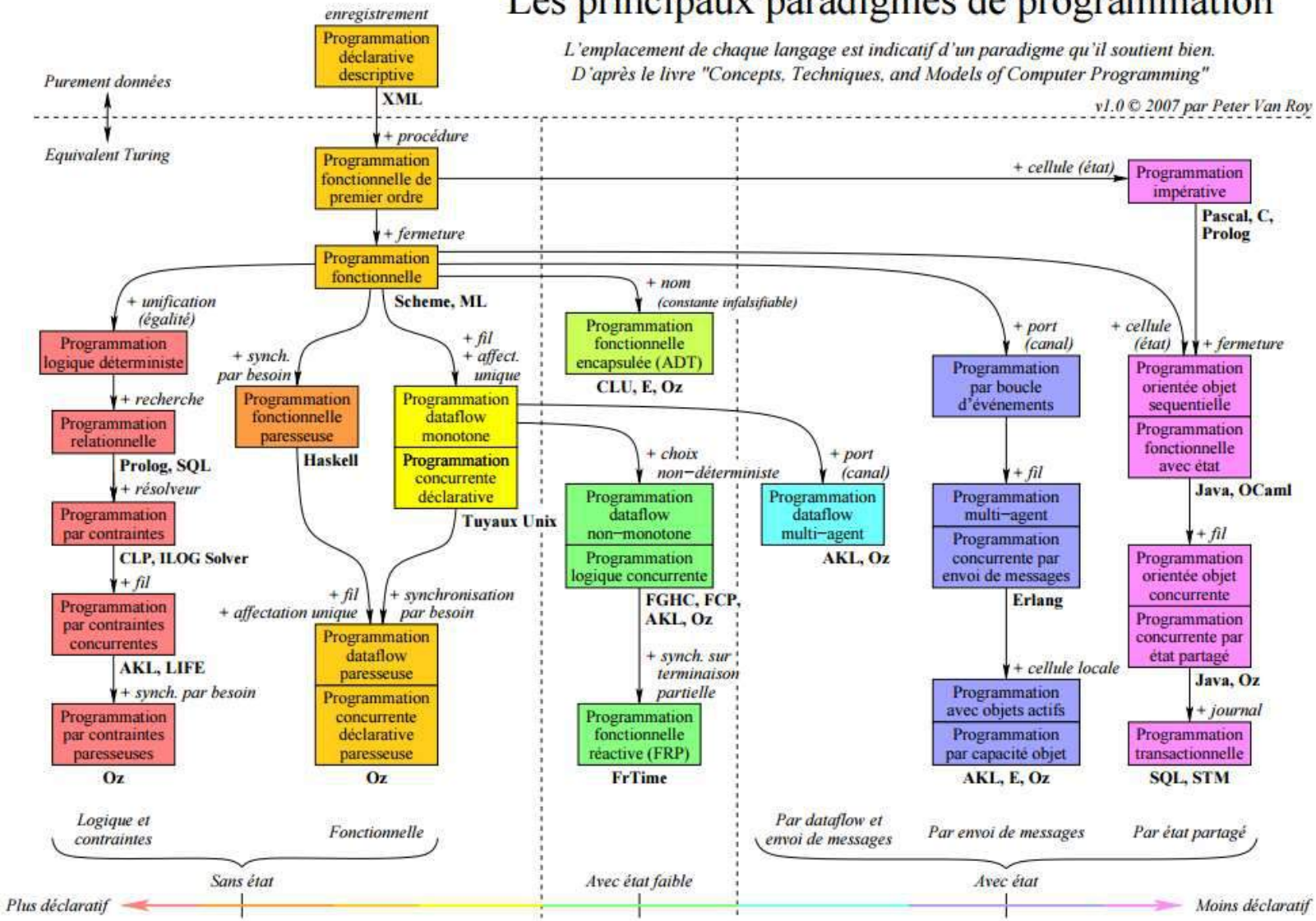
- comment les valeurs et les expressions sont classées en types, comment ils sont manipulés et comment ils interagissent
- typé vs. non typé: vérifier si l'opération est applicable au type de données
- le typage statique vs. dynamique: déterminer le type à la compilation ou à la l'exécution
- typage faible et forte: le traitement de la valeur d'un type en tant que valeur d'un autre type

Paradigmes de programmation

- Un paradigme de programmation est un style fondamental de la programmation informatique, qui sert comme un moyen de la construction de la structure et les éléments de programmes d'ordinateur.
- Chaque langage de programmation réalise un ou plusieurs paradigmes de programmation.
- Exemples de paradigmes de programmation:
  - impérative- flux de contrôle est une séquence explicite pour les commandes
  - fonctionnelle - calcul procède par appels (imbriqués) de fonctions qui évitent tout état global
  - logique (à base de règles) - programmeur spécifie un ensemble de faits et de règles, et un moteur déduit les réponses aux questions
  - orientée objet - calcul est effectué en envoyant des messages à des objets; objets ont état et comportement; objets ont leur état et le comportement basé sur l'appartenance à une classe
  - ……

Les principaux paradigmes de programmation

L'emplacement de chaque langage est indicatif d'un paradigme qu'il soutient bien.
D'après le livre "Concepts, Techniques, and Models of Computer Programming"

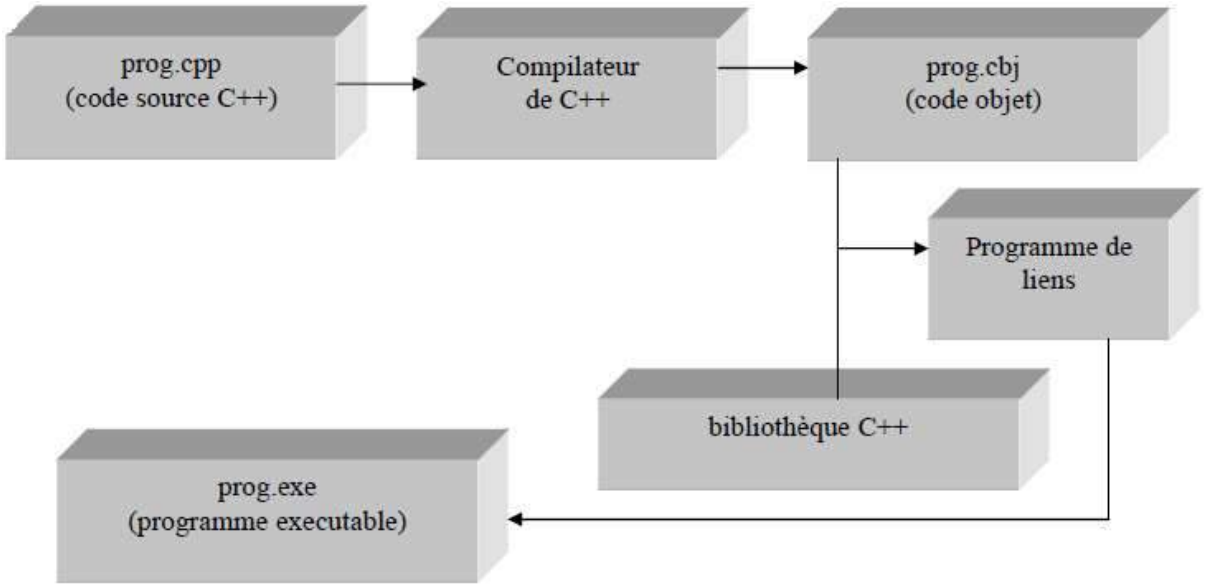


Niveau d'abstraction: langages de bas niveau vs. langages de haut niveau

- Langages de bas niveau:
  - commandes ou fonctions sont proches des instructions de processeur
  - exemples:
    - **code machine/code objet/ langage machine** est le seul langage qui peut être traité par un ordinateur directement, sans une transformation précédente (un flux de données binaire brut)
    - **langage d'assemblage** utilise des symboles à la place de chiffres binaires pour décrire les champs d'instructions (une instruction par instruction machine)
- Langages de haut niveau:
  - sont des langages de programmation avec une forte abstraction des détails de l'ordinateur (un programmeur crée un programme en utilisant des opérations puissantes qui seront ensuite converties en de nombreuses opérations de la machine)
  - un programme source (ou fichier source) est un fichier texte qui contient des instructions écrites dans un langage de haut niveau; il ne peut pas être exécuté par un processeur sans quelques étapes intermédiaires; habituellement, un programme source est **traduit** en un programme de langage machine/code objet par un programme d'application appelé **traducteur**.

Compilateur vs. interpréteur

- Le traducteur peut être compilateur ou interpréteur:
  - lorsque le code écrit dans un langage est interprété, sa syntaxe est lu et exécuté directement, sans phase de compilation (e.g. Javascript)
  - lorsque le code écrit dans un langage est compilé, sa syntaxe est transformé en une forme exécutable avant d'exécuter (e.g. C, C++, Java).



Si on utilise un environnement intégré de développement ces étapes seront exécuter automatiquement en appuyant une touche fonctionnelle ou les barres d'outils.

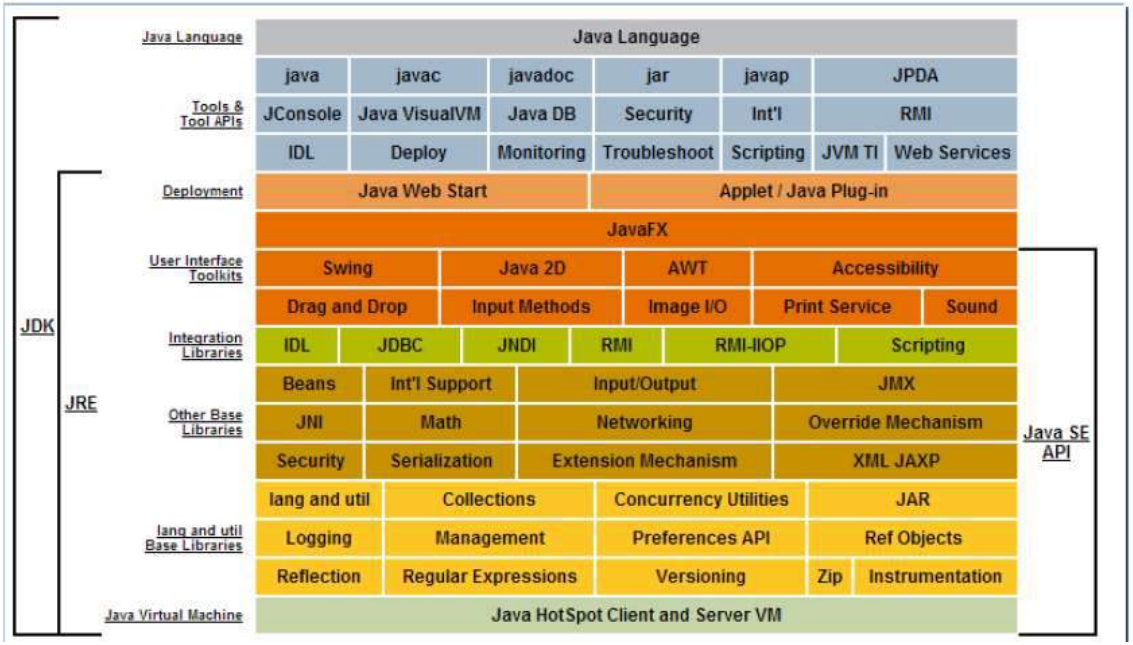
Introduction à Java

Une technologie développée par SUN Microsystems™ lancée en 1995

- Un langage de programmation
- Une plateforme , environnement logiciel dans lequel les programmes java s'exécutent.

- Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C++.
- Java est un langage de programmation compilé, mais plutôt que d' être compiler directement au code machine exécutable, il est compilé en une forme binaire intermédiaire appelée Java Virtual Machine (JVM) byte code. Le byte code est ensuite compilé et / ou interprété pour exécuter le programme.
- Java est portable : il est indépendant de toute plate-forme (nous avons besoin seulement de JVM).
- Java est orienté objet (mais autres paradims de programmation sont également pris en charge, par exemple, impérative) .
- Java assure la gestion de la mémoire: l'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au *garbage collector* qui restitue les zones de mémoire laissées libres suite à la destruction des objets.
- Java est fortement typé: toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser un cast ou une méthode statique fournie en standard pour la réaliser.
- Java est simple (il n'y a pas des pointeurs, pour éviter les incidents en manipulant directement la mémoire, comme en C/C++). Mais Java contient un vaste ensemble de paquets de bibliothèques qui sont nécessaires pour écrire des programmes utiles.

- JRE: Java Runtime Environment (si vous voulez juste pour exécuter des programmes Java, il suffit)
- JDK: Java Development Kit (si vous voulez développer et compiler les programmes Java, vous en avez besoin)
- Il y a different éditions de Java: Java SE, Java EE, Java ME…



Standard Edition JSE

Fourni les compilateurs, outils, runtimes, et APIs pour écrire, déployer, et exécuter des applets et applications dans la langage de programmation Java

Entreprise Edition JEE

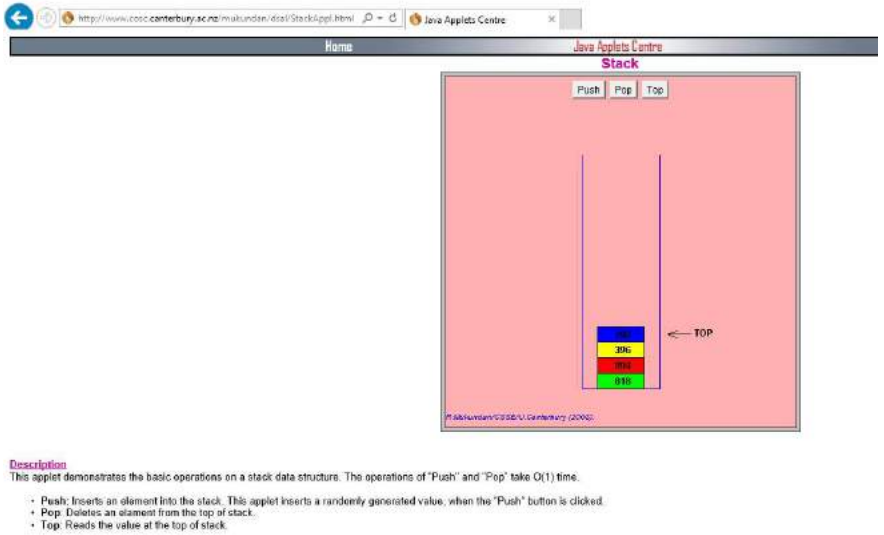
Destinée au développement d'applications « d'entreprise » (*«business applications»*) robustes et interopérables. Simplifier le développement et le déploiement d'applications distribuées et articulées autours du web.

Mobile Edition JME

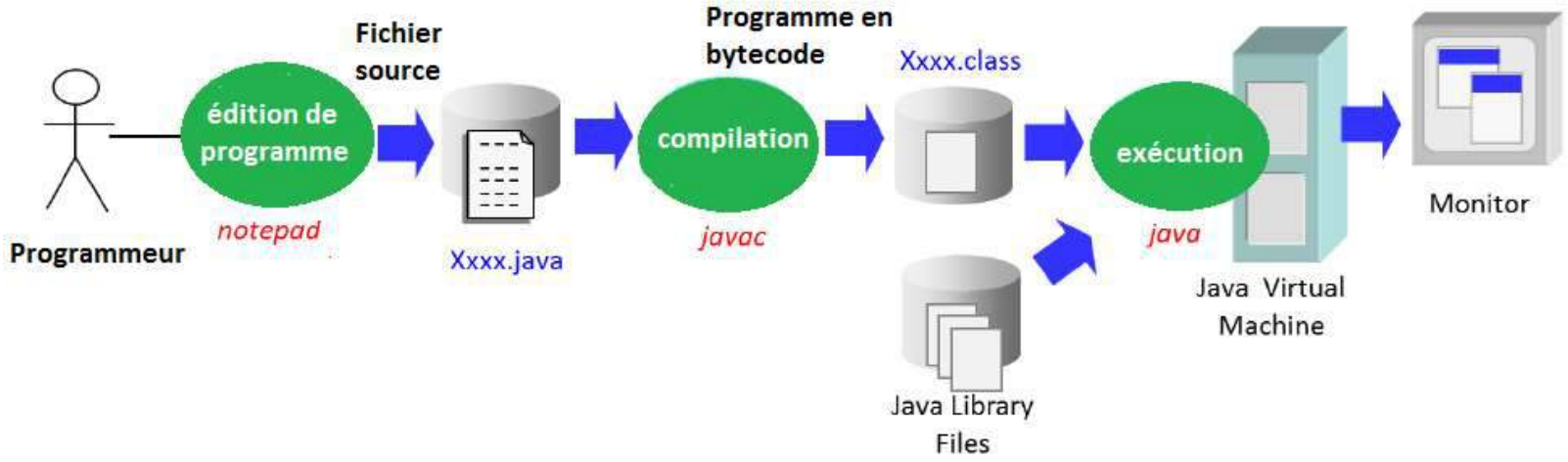
Environnement d'exécution optimisé pour les dispositifs « légers » :
- Carte à puce (smart cards)
- Téléphones mobiles
- Assistants personnels (PDA)

Java SE

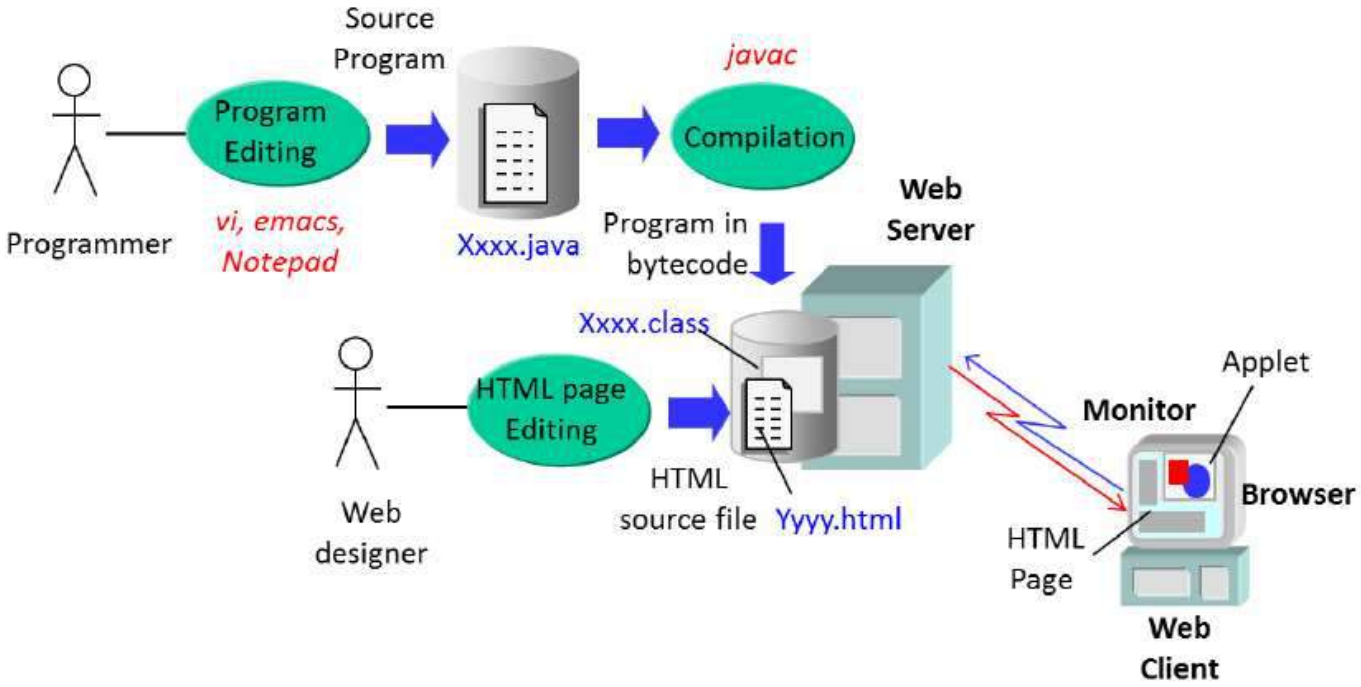
- Il existe 2 types de programmes:
  - Une application autonome (stand alone program) est une application qui s'exécute sous le contrôle direct du système d'exploitation.
  - Une applet est une application qui est chargée par un navigateur et qui est exécutée sous le contrôle d'un plugin de ce dernier (il n'y a pas la les applets n'ont pas de méthode main()).



Développement et l'exécution du programme d'application Java



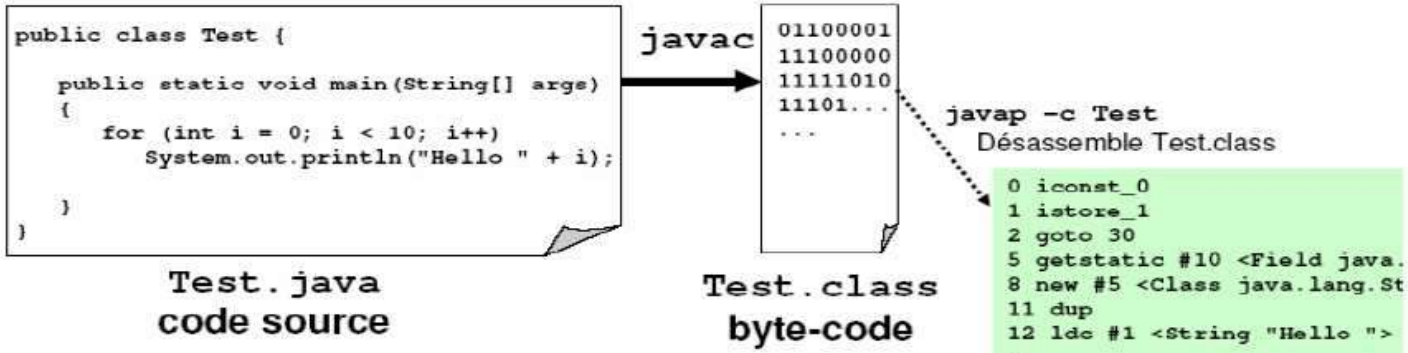
Développement et l'exécution d'un applet Java



Le langage Java

Un langage compilé / interprété

- Compilation d'un programme JAVA : génération de byte-code

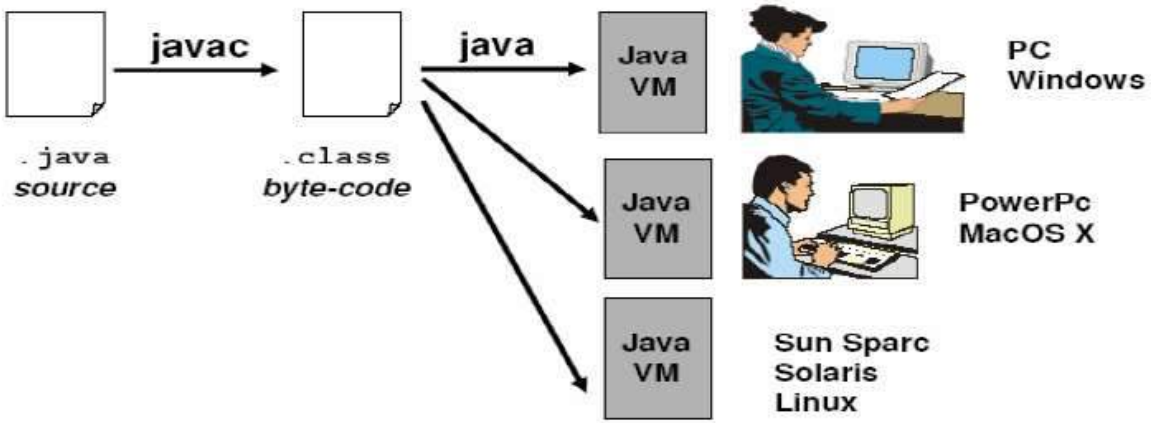


- Le byte-code est :
  - proche d'un langage machine
  - indépendant de la plateforme d'exécution (matériel + OS)

La machine virtuelle Java

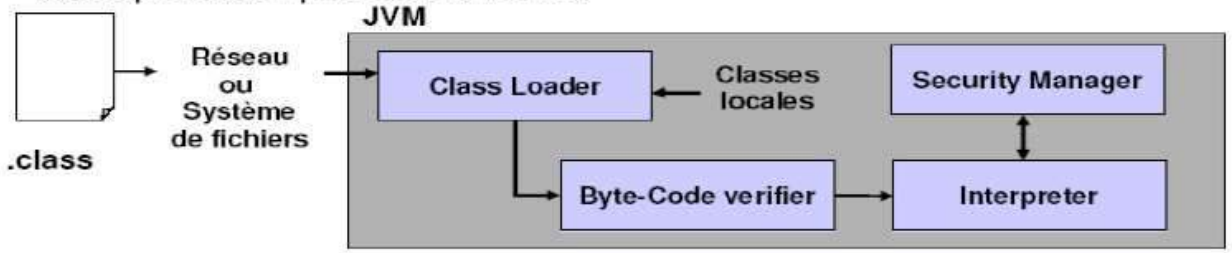
Exécution d'un programme Java compilé

- **byte-code assure la portabilité des programmes Java**
  - langage d'une Machine Virtuelle
  - à l'exécution un interpréteur simule cette machine virtuelle



La machine virtuelle java
Principes de fonctionnement

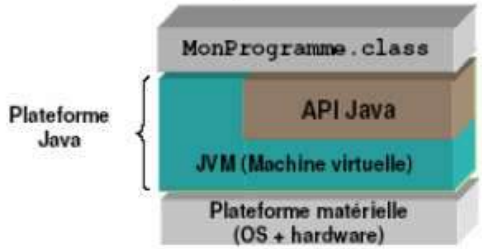
- **Chargement**
  - chargement sélectif et dynamique des classes
  - vérification statique du code (tentatives de modification de la machine virtuelle, ...)
- **Protection lors de l'exécution**
  - Le "security manager" possède un droit de veto (accès "sauvages" au système de fichiers, ...)
  - Cette protection peut-être modulée



La plateforme Java

- **Plateforme**
  - Environnement matériel et/ou logiciel dans lequel un programme s'exécute.
    - La plus part des plateformes sont la combinaison d'un OS et du matériel sous-jacent (*MS Windows + Intel, Linux + Intel, Solaris + Sparc, Mac Os X + Power PC*)

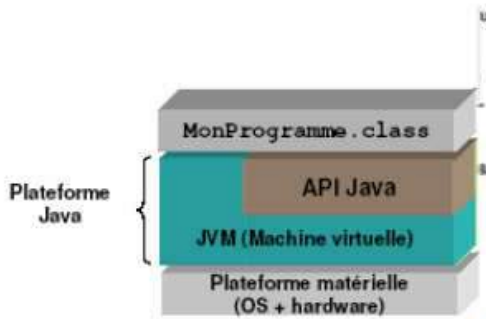
- **La plateforme Java est entièrement logicielle et s'exécute au dessus des plateformes matérielles**

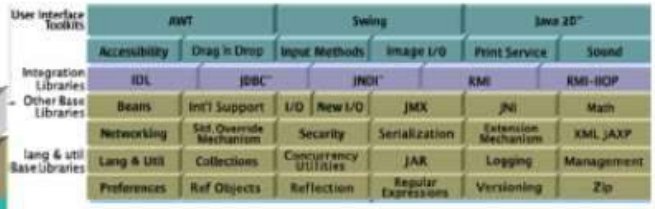


API (Application Programming Interface) Java :
Bibliothèques Java standards sur lesquelles le programmeur peut s'appuyer pour écrire son code

La plateforme Java
API Java

- **API Java**
  - (très) vaste collection de composants logiciels (classes et interfaces)
  - organisée en bibliothèques (***packages***)
  - offre de nombreux services de manière standard (indépendamment de la plateforme matérielle)

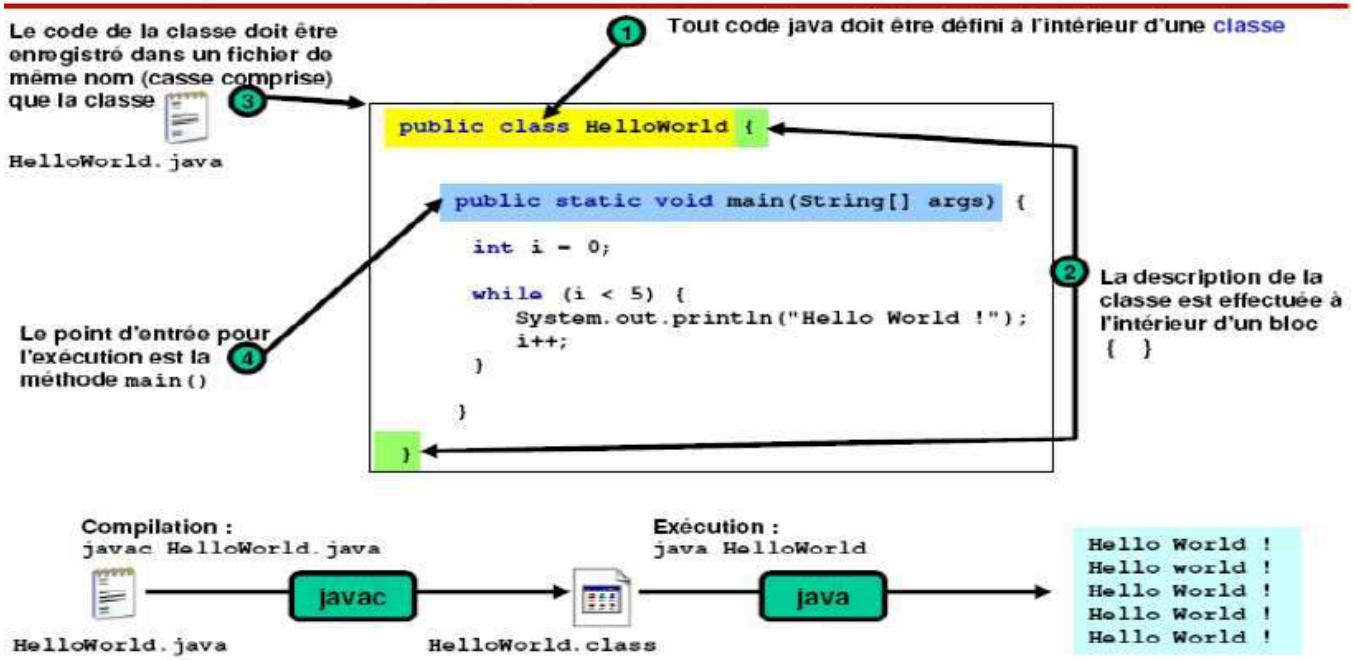




Programmer en Java nécessite une bonne connaissance de l'API.
Attention à la courbe d'apprentissage (learning curve) qui est peut être longue

Le Langage Java

Mon premier programme Java (pas très objet…)



```java
public class JavaApplication1 {

    public static void main(String[] args) {
        System.out.println(args[0]+" "+args[1]);
    }

}
```

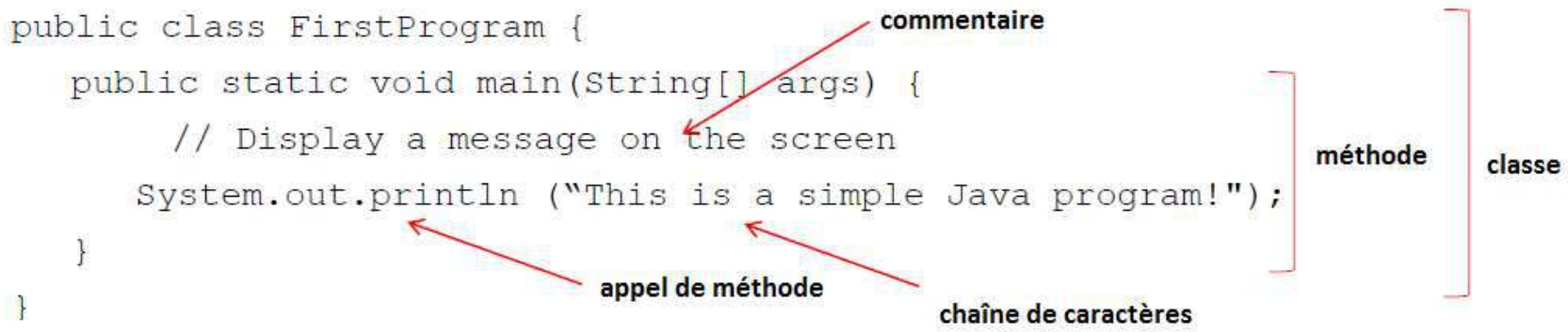
```
C:\LP>dir
 Volume in drive C has no label.
 Volume Serial Number is 3AF7-1511

 Directory of C:\LP

13.10.2015  05:38    <DIR>          .
13.10.2015  05:38    <DIR>          ..
13.10.2015  05:36               147 JavaApplication1.java
               1 File(s)            147 bytes
               2 Dir(s)  622.705.594.368 bytes free

C:\LP>"C:\Program Files (x86)\Java\jdk1.8.0_60\bin\javac" JavaApplication1.java

C:\LP>dir
 Volume in drive C has no label.
 Volume Serial Number is 3AF7-1511

 Directory of C:\LP

13.10.2015  05:38    <DIR>          .
13.10.2015  05:38    <DIR>          ..
13.10.2015  05:38               597 JavaApplication1.class
13.10.2015  05:36               147 JavaApplication1.java
               2 File(s)            744 bytes
               2 Dir(s)  622.705.590.272 bytes free

C:\LP>"C:\Program Files (x86)\Java\jdk1.8.0_60\bin\java" JavaApplication1
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at JavaApplication1.main(JavaApplication1.java:4)

C:\LP>"C:\Program Files (x86)\Java\jdk1.8.0_60\bin\java" JavaApplication1 cucu bau
cucu bau

C:\LP>_
```

Eléments de base du langage

- Nous traitons Java d'abord comme un langage de programmation classique. Nous aborderons les objets ultérieurement.
- Dans un programme on trouve deux choses: des données et les instructions qui les manipulent. On s'efforce généralement de séparer les données des instructions.

- Java utilise les types de données suivants:
  - les nombres entiers;
  - les nombres réels;
  - les caractères et chaînes de caractères;
  - les booléens;
  - les objets.

Un programme simple



Types et variables

- En Java, toute valeur a un type. “This is a simple Java program!” a le type String, l'objet System.out a le type PrintStream et le numéro 13 a le type int.
- Pour rappeler une valeur, vous devez maintenir dans une variable.
- Une variable est un emplacement de stockage dans la mémoire de l'ordinateur qui a un type, un nom et un contenu.
- Vous utilisez des variables pour stocker les valeurs que vous souhaitez utiliser à une date ultérieure.
- Identifiants pour les variables, les méthodes et les classes sont composées de lettres, chiffres et caractères de soulignement.
- Par convention, les noms de variables doivent commencer par une lettre minuscule.
- Les noms de classe doivent commencer par une lettre majuscule.

Types de données en Java

- Les types de données de référence (Leurs valeurs sont des pointeurs vers des objets complexes et les tableaux. Ils sont gérés par la JVM):
  - arary (tableau)
  - class
  - Interface
  - null
- Les primitives

```
int     4 bytes   -2.147.483.648 → 2.147.438.647
short   2 bytes   -32.768 → 32.767
long    8 bytes   -9.223.372.036.854.775.808L→-9.223.372.036.854.775.807L

float   4 bytes   ±3,40282347E+38F
double  8 bytes   ± 1,79769313486231570E+308

char    2 bytes   -32.768 → 32.767
```

Octets

Les primitives

- **Initialisation des primitives:**
- Les primitives doivent être déclarées et initialisées avant d'être utilisées. Une primitive non initialisée produira une erreur à la compilation :*« Variable may not have been initialized».*
- Remarquons que les primitives, lorsqu'elles sont employées comme membre de classe, possède des valeurs par défaut (tableau suivant). Il n'est donc pas nécessaire de les initialiser !

primitive	*valeur par défaut*
boolean	*false*
char	*'\u0000' (null)*
byte	*(byte)0*
short	*(short)0*
int	*0*
long	*0L*
float	*0.0f*
double	*0.0d*

primitive	*syntaxe*
char	*'x'*
int	*5 (décimal), 05 (octal), 0x5 (hexadécimal)*
long	*5L, 05L, 0x5L*
float	*5.5f ou 5f ou 5.5E5f*
double	*5.5 ou 5.5d ou 5.5E5d ou 5.5E5*
boolean	*false ou true*

Déclaration des variables

- Rôle des déclarations:
  - Un programme manipule des données caractérisées par un nom et un type. Ces données sont stockées en mémoire. Au moment de la traduction du programme, le compilateur affecte à chaque donnée un emplacement en mémoire caractérisé par une adresse et une taille. Il le fait en s'aidant des déclarations faites par le programmeur.
- **Une variable:**
  - est identifiée par un nom et se rapporte à un type de données;
    int jour, mois, annee;
  - possède un nom, un type et une valeur;
    int i=3 , j=4 ;
  - sa valeur peut être modifiée pendant le programme.
- Les variables peuvent être initialisées lors de leur déclaration (Une instruction peut tenir sur plusieurs lignes).

Exemple :

```
char
code
=
'D';
```

Exemple :

```
int nombre; // déclaration
nombre = 100; //initialisation
OU int nombre = 100; //déclaration et initialisation
```

Eléments de langage

- Identificateurs : Composition des noms des variables. Restrictions et conventions d'affectation des noms.
- Littéraux : Composition des noms des constantes. Affectation de leurs valeurs.
- Mots clés : Mots prédéfinis du langage
- Instructions
- Blocs de code : Comment les instructions sont-elles regroupées ensemble ?
- Commentaires : Ajout de commentaires et de remarques au programme
- Expressions : Qu'est-ce qu'une expression et comment en écrire une ?
- Opérateurs : Opérateurs utilisés dans le langage. Leur utilisation dans les expressions.

Eléments de langage

Identifier::= *letter* (*letter*+*digit*)*
letter::=('A'-'Z')+('a'-'z')+'_'+'$'
digit::='0'-'9'

Identificateurs

Literals

- integers: 57 -456 0x5A9F 0745 6894576123L
- reals: 3.402F -45.89
- boolean: true false
- characters: 'f' 'T' '9' '\u0A48' \110 '\\' '\n' '\f' '\t' '\'' '\"'
- strings: "This is a string" "" "0123456789\n" "+" " "

Littéraux (Un *littéral*, ou constante, représente une valeur qui ne change jamais.)

Operators: + - * / % & | ^ ~ && || ! < > <= >=
<< >> >>> = ? ++ -- == += -= *= /= %= &= |=
^= != <<= >>= >>>= . [ ] ( )

Opérateurs

Separators: { } ; , :

Séparateurs

Comments: /*.....*/ //..... /**.....*/

Commentaires