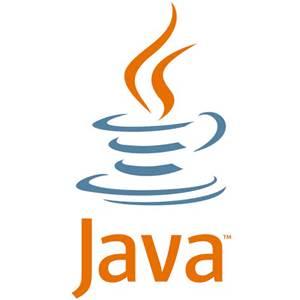
IUT d’Orsay - DUT Informatique - Jean-Claude MARTIN – JEAN-CLAUDE.MARTIN@U-PSUD.FR

**Programmation Orientée Objet (POO)  
COURS 1 : Programmation Structurée en Java et Machine Virtuelle**





# Organisation du cours

* 15 chapitres de à
* DS fin
* Polycopiés cours et feuilles d’exercices sur ...

Modalités d’évaluation des connaissances (à titre indicatif)

* Moyenne = moyenne pondérée DS et CC
* DS : une seule note
* CC : moyenne pondérée de
  + Interrogations (dont surprises) : possibles à chaque séance   
    => Apporter à chaque séance : polys, papier, crayon, gomme
  + Projet
* **Rattrapages seulement sur justificatif d’absence valide et inscription** auprès du secrétariat

Méthode de travail

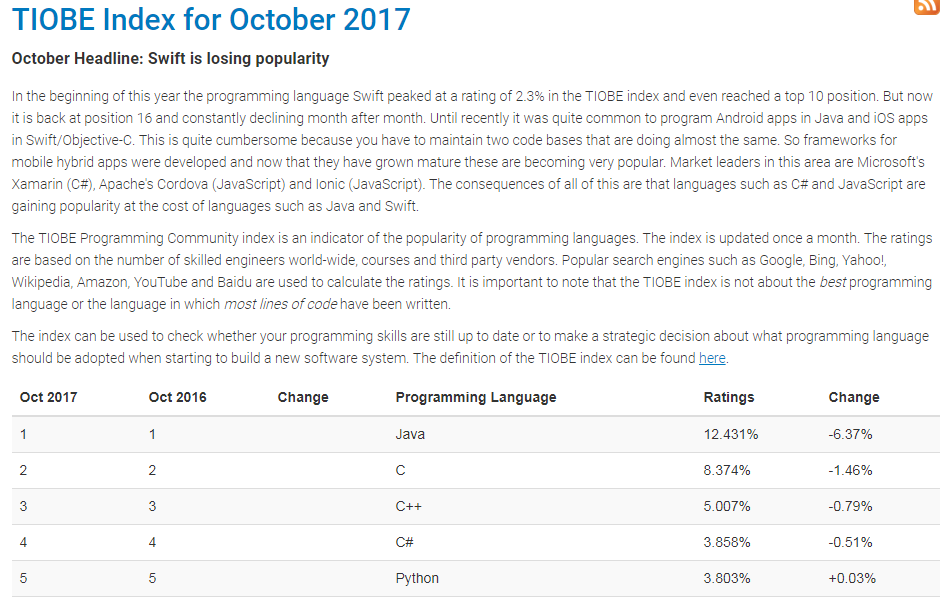
* Prendre des notes sur un cahier ou des feuilles en plus du support de cours qui est très partiel => prévoir feuilles de papier libre

# Objectifs du cours

Vos compétences à la fin de ce cours

* Comprendre les concepts de la programmation objet (POO) et les appliquer au langage Java (et ensuite à d’autres langages) :
  + définition de classes, attributs et méthodes de classe (static), attributs et méthodes d'objets, héritage, classes abstraites, interfaces, polymorphisme, interfaces graphiques avec le modèle MVC
  + sérialisation, threads, Java 2D …

# Pourquoi un enseignement en programmation Java ?



# Pourquoi un enseignement en programmation Java ?

Programmation structurée =  
 ………………………………………………………………………………...  
+

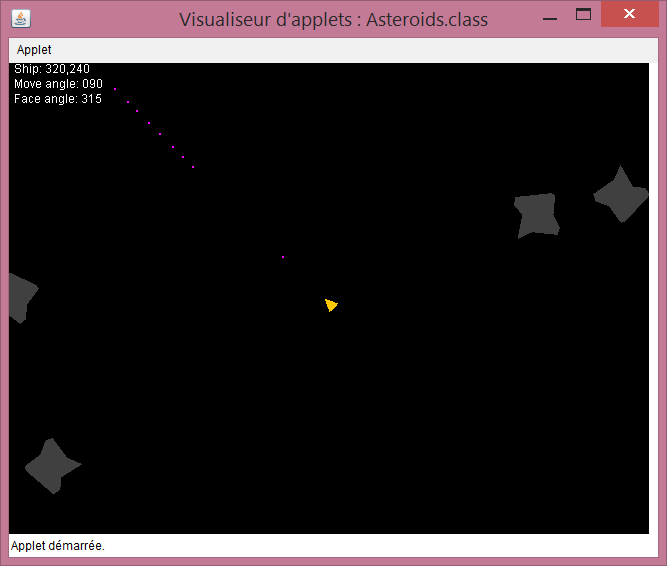
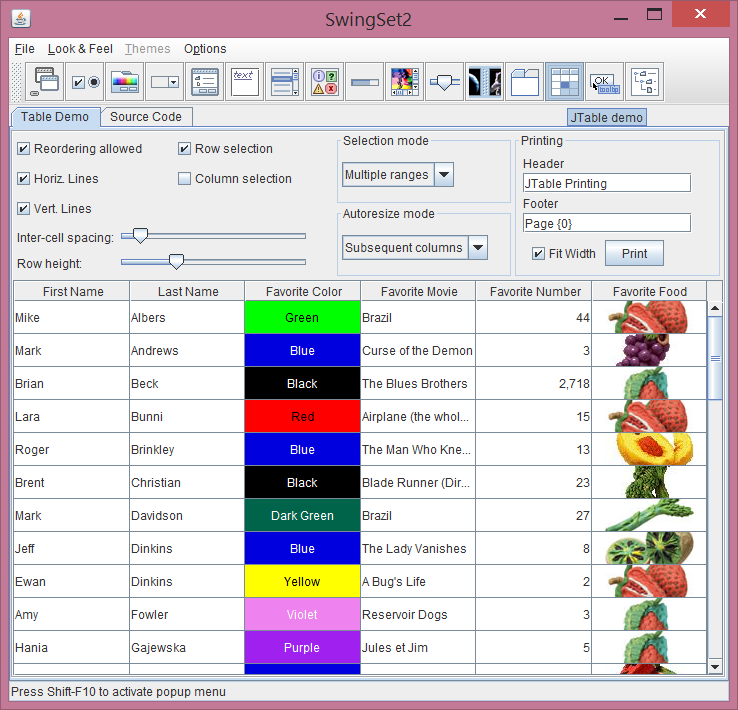
………………………………………………………………………………...

* Java permet la programmation ………………………………………….………... ET ………………………………………………...
* Compilateur strict + messages d’erreur clairs :

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5

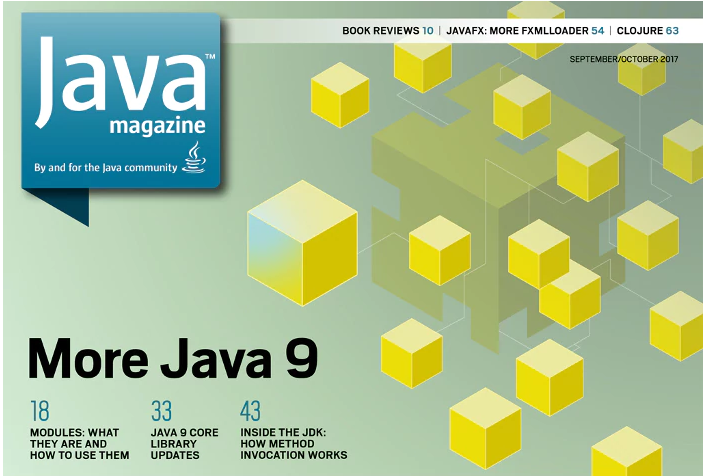
at Exemples01.main(Exemples01.java:8)

* Utilisé dans l’industrie : PSA, compuBase, …
* Interfaces graphiques + réseau + bases de données + web + … => démo de SwingSet [http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html](http://java.sun.com/products/plugin/1.3.1_01a/demos/jfc/SwingSet2/SwingSet2Plugin.html)

[](http://java.sun.com/products/plugin/1.3.1_01a/demos/jfc/SwingSet2/SwingSet2Plugin.html)

* Certains points communs mais aussi de nombreuses différences entre C++ et Java

# Où trouver de l’information sur Java?



Dans mes polys !

Dans les livres que je conseille

Sur le WEB :

Documentation Oracle

Classes du JDK (API)<http://docs.oracle.com/javase/8/docs/api/index.html>

Spécifications de la machine virtuelle Java<http://docs.oracle.com/javase/specs/jvms/se8/html/index.html>

Tutoriaux java textes

Tutoriaux oracle [https://docs.oracle.com/javase/tutorial/](http://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html)

OpenClass Room / Site du Zéro :<https://openclassrooms.com/courses/apprenez-a-programmer-en-java>

RECENT <http://study.com/academy/course/java-programming-tutorial-training.html#information>

Vidéos

Cours de Programmation Orientée Objet de Xavier Blanc<https://www.youtube.com/watch?v=BNEtWb3WceQ>

MOOC<https://www.coursera.org/course/intropoojava>

MOOC EPFL<https://www.youtube.com/watch?v=n0_gC7YalUM&list=PLZbs1ERZ-TGVtIKH119wo-CTcy7Z_2s1X>

Livre en ligne

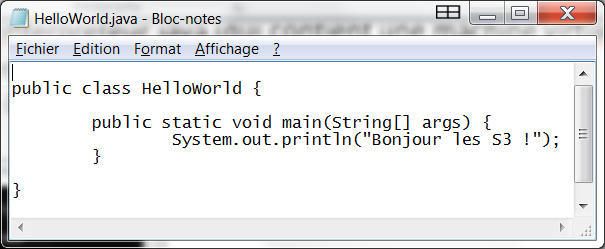
Penser en Java :<http://bruce-eckel.developpez.com/livres/java/traduction/tij2/>

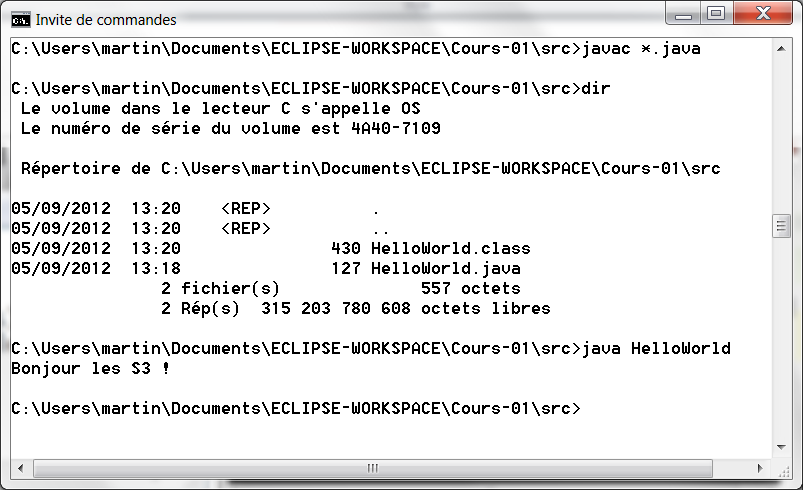
Java magazine<http://www.oracle.com/technetwork/java/javamagazine/index.html>

# Environnements de développement

JDK :...................................................................................................  
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

* Installé à l’IUT : JDK …………... : Environnement de base gratuit
  + plusieurs programmes exécutables depuis la ligne de commande : Javac (compilateur), Java (interpréteur), Javadoc (génération de documentation), …
  + de nombreuses bibliothèques (environ 1000 classes) : interfaces graphiques, réseau, entrées-sorties, …
  + Compilation avec le compilateur javac
    - javac HelloWorld.java => Crée un fichier HelloWorld.class
  + Exécution avec l’interpréteur java (qui contient une machine virtuelle Java)
    - java HelloWorld => Exécute le programme





Versions précédentes du JDK : début 1991 : créé par SUN ; 2006 : Java Standard Edition (JSE) complétée par Java Enterprise Edition (JEE), et Java Micro Edition embarqué

Eclipse .……………………... <http://www.eclipse.org/>

* IDE : Environnement de Devt Integré
* Compilation au fur et à mesure
* Démo Eclipse

# Machine Virtuelle Java (JVM)

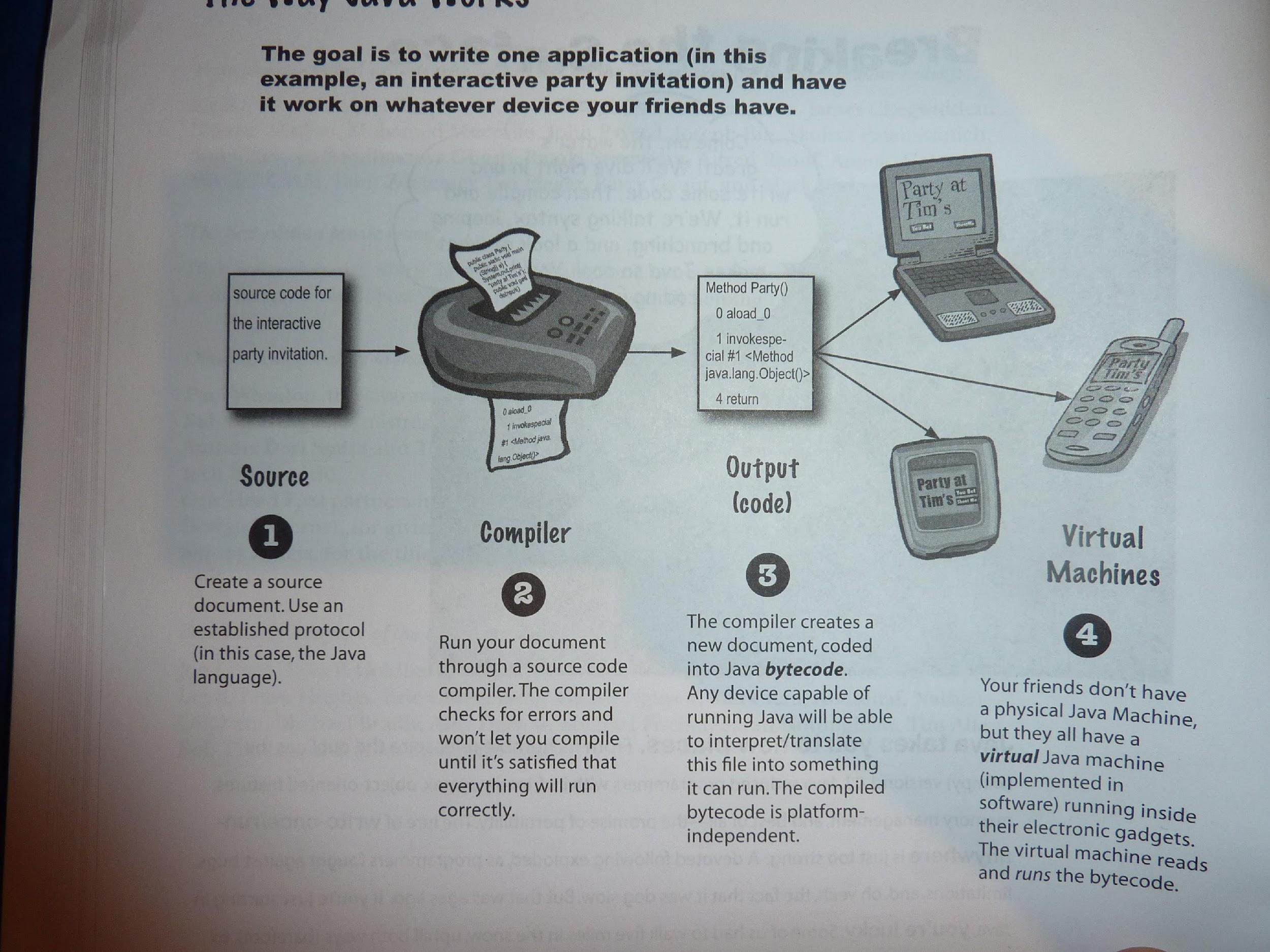
« compile once, run everywhere»

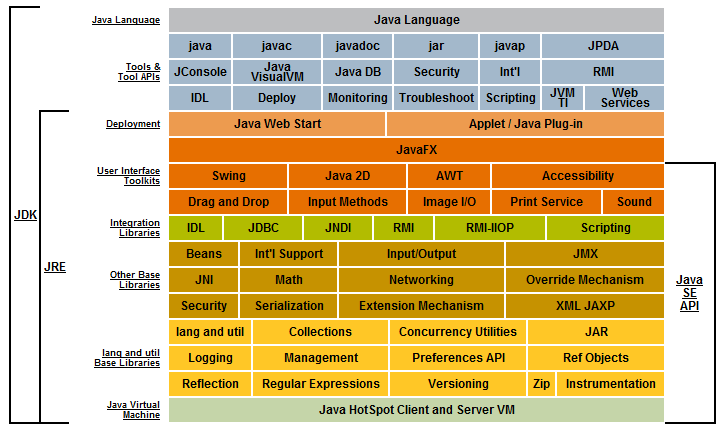
* Définition
  + une machine virtuelle est une **illusion** d'un appareil informatique créée par un logiciel d'émulation
  + le logiciel d'émulation **simule la présence de ressources matérielles** et logicielles telles que la mémoire, le processeur, … permettant d'exécuter des programmes comme si ceux-ci s'exécutaient sur la machine simulée.
  + Les machines virtuelles **sont utilisées pour** exploiter les logiciels d'une machine qui n'existe plus dans le commerce (ordinateur, console de jeu, assistant personnel, ...), pour cacher la machine simulatrice et simuler une machine fictive, telle que la machine virtuelle Java.
* Intérêts : sécurité, portabilité, c’est plus rapide que de recompiler le source
* Inconvénients : exécution moins rapide que du code compilé directement pour la machine cible
* Programmes comportant une machine virtuelle Java :   
  interpréteur Java  
  Navigateurs web (pour pouvoir exécuter des “applets” : des petites applications Java)

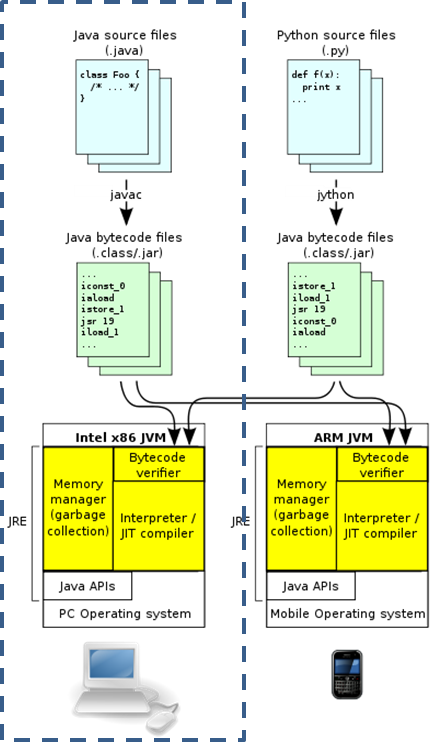
### Vidéo

Introduction to Java Virtual Machine (JVM)<http://youtu.be/G1ubVOl9IBw>









Spécifications de la machine virtuelle Java (**604 pages**) <http://docs.oracle.com/javase/specs/>

# Types et instructions de base

Programme à compléter

**import** java.util.Scanner;

**public** **class** TypesDeBase {

**public** **static** **void** main(String[] args) {

//-----------------------------------------------------

// Déclarations de variables locales à la méthode main () :

types de base qui existent en C

//-----------------------------------------------------

**short** s ; // taille : 2 octets

**int** i ; // taille : 4 octets

**long** l ; // taille : 8 octets

**float** f ; // taille : 4 octets

**double** d ; // taille : 8 octets

**char** c ; // format Unicode sur 2 octets

//-----------------------------------------------------

// Déclarations de types de base qui n’existent pas en C

//-----------------------------------------------------

**byte** by ; // taille : 1 octet

**boolean** b ;

//-----------------------------------------------------

// Affectation de quelques valeurs

//-----------------------------------------------------

i = 0 ;

b = **true** ;

c = 'a' ;

//-----------------------------------------------------

// Affichages

//-----------------------------------------------------

System.*out*.print (i); // AFFICHE LA VALEUR DE I SANS ALLER A LA LIGNE

System.*out*.println (i); // RETOUR A LA LIGNE

System.*out*.println ("i vaut " + i); // AFFICHE i vaut 0

System.*out*.printf ("affichage formaté : %3.2f", 100.678); // .........

System.*out*.println (); // RETOUR A LA LIGNE

//-----------------------------------------------------

// Lecture au clavier

//-----------------------------------------------------

Scanner sc = **new** Scanner(System.*in*);

System.*out*.println("Veuillez saisir un mot : ");

String str = sc.nextLine(); //……………………………………………………………………………   
 // …………………………………………………………………………….....

System.*out*.println("Vous avez saisi : " + str);

System.*out*.println("Veuillez saisir un entier : ");

**int** j = sc.nextInt();

System.*out*.println("Vous avez saisi le nombre : " + j);

//-----------------------------------------------------

// Déclaration de constantes et initialisations

//-----------------------------------------------------

**final** **int** N = 20 ; //CONSTANTE

**final** **int** M ;

System.*out*.println("Veuillez saisir un entier : ");

M = sc.nextInt();

//-----------------------------------------------------

// Conversions implicites

//-----------------------------------------------------

**int** k = 1000 ;

**long** lo ;

lo = k ;

//-----------------------------------------------------

// Conversions explicites (cast)

//-----------------------------------------------------

i = (**int**) 2.5 ;

System.*out*.println ("valeur de (int) 2.5 : " + i);

//-----------------------------------------------------

// Gestion des erreurs en Java

//-----------------------------------------------------

// Diviser par zéro

i = i / 0 ;

// Exception in

Exception in thread "main"

java.lang.ArithmeticException: / by zero

at TypesDeBase.main(TypesDeBase.java:81)

// Saisir un caractère alors qu'on attend un nombre => Essayez !

Schéma de la mémoire

Attention

* Java est sensible à la casse ! par exemple : n est different de N
* Conversions implicites / d’ajustement de type : byte -> short -> int -> long -> float -> double
* Conversions explicites :
  + Le cast (transtypage) est un OPERATEUR
  + role : prévenir le compilateur que l’on est conscient du risque de perte d’information
* Opérateurs logiques paresseux && || : S’ARRETENT DES QU’ILS CONNAISSENT LE RESULTAT
  + & | : courageux : évaluent les deux opérandes
* Opérateur conditionnel : max = a > b ? a : b ;
* Infinity et NaN : … *(cf par exemple Delannoy Programmer en Java p 56)*
* Types de base (…entiers, reels, char, ………………..) vs. type reference (objets, tableaux)

# 

# 

# Structures de données et structures de contrôle

**import** java.util.Arrays;

**public** **class** TableauxEtBoucles {

**public** **static** **void** main(String[] args) {

//-----------------------------------------------------

// Tableau de type de base

//-----------------------------------------------------

**int** t [] ; // DECLARATION d'une REFERENCE de type tableau

// CETTE CASE POURRA ENSUITE CONTENIR L’ADRESSE D’UN TABLEAU

t = **new** **int** [3];// ALLOCATION d'une zone memoire pour n entiers dans le TAS

t[2] = 7; // AFFECTATION d'une valeur ˆ une case du tableau

System.*out*.println(t[2]); // ACCES / LECTURE de la valeur d'une case du tableau et affichage

//-----------------------------------------------------

// Parcours du tableau

//-----------------------------------------------------

// Avec une boucle et un booleen

**boolean** fini = **false** ;

**int** i = 0 ;

System.*out*.println ("\n Parcours du tableau avec un booleen et une boucle while : ");

**while** (! fini) {

System.*out*.println (t[i]);

i++ ;

fini = i == t.length ;

}

// Avec une boucle foreach

System.*out*.println ("\n Parcours du tableau avec une boucle foreach : ");

**…………………………………………………………………………………………**

System.*out*.println (valeur);

// Copie d'un tableau

**int** [] t2 = t ; // ON RECOPIE DANS **…………………………………………………………………………………………**

t2 [0] = 33 ;

// Affichage apres copie : // la 1ere valeur affichee sera ……………...

System.*out*.println ("\n Parcours du tableau t apres modification t2 = t ; t2[0] = 33 : ");

**for** (**int** valeur : t)

System.*out*.println (valeur);

// Initialiser un tableau lors de la declaration

**double** notes [] = {12, 8, 17.5, 2, 19};

// Affichage avant tri

System.*out*.println ("\n Affichage avant tri : ");

**for** (**double** valeur : notes)

System.*out*.println (valeur);

// Trier un tableau

Arrays.*sort*(notes);

// Affichage apres tri

System.*out*.println ("\n Affichage apres tri : ");

**for** (**double** valeur : notes)

System.*out*.println (valeur);

//-----------------------------------------------------

// Exceptions

//-----------------------------------------------------

// Debordement du tableau

System.*out*.println (t[10]);

// Oubli d'allouer le tableau

**int** tab [];

tab = null ;

tab[0] = 10 ;

=> Exception in thread "main" java.lang.NullPointerException

at TableauxEtBoucles.main(TableauxEtBoucles.java:77)

Schéma de la mémoire

# 

Tableaux à deux dimensions

* int [ ][ ] tab = new int [3] [2] ;
* Les lignes peuvent avoir DES COLONNES DE TAILLE DIFFERENTE
* t[i] contient l’adresse d’un tableau à une dimension

# Sous-programmes en Java : les METHODES

(static pour l’instant :-(

**public** **class** Methodes {

// L'ordre de declaration des methodes n'a pas d'importance

//------------------------------------------------------------------

// Declarer un nombre variable d'arguments

//------------------------------------------------------------------

**static** **int** somme (**int** ... valeurs) {

**int** s = 0 ;

**for** (**int** i=0; i < valeurs.length ; i++)

s += valeurs [i];

**return** s ;

}

//------------------------------------------------------------------

// Procedure tentant d'echanger 2 parametres entiers

// Resultat :

//------------------------------------------------------------------

**static** **void** echange (**int** a, **int** b) {

**int** aux = b ;

b = a ;

a = aux ;

System.*out*.println ("durant l'appel apres echange : a = " + a + ", b = " + b);

}

//------------------------------------------------------------------

// Procedure tentant de modifier la case d'un tableau passe en parametre

// Resultat :

//------------------------------------------------------------------

**static** **void** modifierTab (**int** t[]) {

t[0] = 9 ;

}

//------------------------------------------------------------------

// Programme principal

//------------------------------------------------------------------

**public** **static** **void** main(String[] args) {

//------------------------------------------------------------------

// Nombre variable de parametres

//------------------------------------------------------------------

System.*out*.println (*somme* (1, 2, 3));

System.*out*.println (*somme* (2, 8));

System.*out*.println (*somme* ());

//------------------------------------------------------------------

// Appel de la procedure tentant d'echanger 2 parametres entiers

//------------------------------------------------------------------

**int** i = 1 ;

**int** j = 2 ;

System.*out*.println ("avant appel : i = " + i + ", j = " + j);

*echange* (i, j);

System.*out*.println ("apres appel : i = " + i + ", j = " + j);

//------------------------------------------------------------------

// Appel de la procedure tentant de modifier la case d'un tableau passe en parametre

//------------------------------------------------------------------

**int** t[] = {1, 2, 3};

System.*out*.println ("\n Affichage avant appel : ");

**for** (**int** valeur : t)

System.*out*.println (valeur);

*modifierTab* (t);

System.*out*.println ("\n Affichage apres appel : ");

**for** (**int** valeur : t)

System.*out*.println (valeur);

Schéma de la mémoire

Résumé

* Procédure = méthode ne retournant pas de résultat (type void)
* Fonction = méthode retournant un résultat
* Le mot-clé **static** devant une méthode indique que CE N’EST PAS DE L’OBJET => PAS BIEN
* Les paramètres sont passés par **VALEUR / COPIE**
* Lors du passage d’un tableau en paramètre, on passe SON ADRESSE

# Autres notions

* Portée des variables
* Commentaires et mise en page !!
* Examiner le fichier .class : Javap -c Nombres
* Fichiers archives .jar : Archive permettant de regrouper plusieurs fichiers .class

Quizz

* <http://www.funtrivia.com/trivia-quiz/SciTech/Java-47826.html>
* <http://www.funtrivia.com/playquiz/quiz2042701763c98.html>