Chapitre 1 : Introduction à l'algorithmique

Juan Carlos QUEZADA

INSA Strasbourg

Planning du cours

Planning

1. Introduction
2. Séquences et tests
3. Itérations
4. Les tableaux et les matrices
5. Les sous-programmes
6. Les algorithmes de tri et de recherche dans les tableaux
7. Applications : Géométrie algorithmique

Évaluation

- Rendus du TP5, TP6 et TP7 (une semaine après)

Définitions

Informatique

Sciences et techniques de traitement automatique de l'information

Ordinateur

Calculateur électronique capable d'effectuer une succession de tâches définies par l'utilisateur

Langage de programmation

Code permettant de décrire des algorithmes et leurs données sous une forme permettant leur exécution par une machine, ET leur compréhension par des lecteurs humains.

Programme

Codage d'un algorithme dans un langage de programmation

Définition d'algorithme

Définition

Suite finie, séquentielle de règles que l'on applique à un nombre fini de données, permettant de résoudre des classes de problèmes semblables

- Un algorithme, c'est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné.
- Pour fonctionner, un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter.

Exemple d'algorithme : Changer une roue

1. Sortir cric, manivelle, et roue de secours
2. Monter la voiture sur le cric
3. Démonter la roue
4. Monter la roue de secours
5. Enlever le cric
6. Ranger cric, manivelle, et roue de secours

Exemples d'algorithmes

Monter la voiture sur le cric

si la roue crevée est à gauche alors
    Installer le cric à gauche
sinon
    Installer le cric à droite
fin si
Faire monter le cric

Démonter une roue

Prendre la manivelle
tant que il reste au moins un boulon faire
    Dévisser un boulon
    Retirer le boulon
    Ranger le boulon
fin tant que
Retirer la roue

Composants d'un langage algorithmique

Langage

- lexique
- syntaxe
- sémantique

Dénotation

- nommer/désigner les objets

Objets élémentaire

- constantes
- opérations élémentaires

Ordonnancement

- séquentielle
- conditionnelle

Langage

Lexique : mots clés

- un ensemble de mots
- en général réservés à cet usage
- évite l'ambiguïté

Syntaxe

- Elle est définie à partir d'une grammaire formelle
- On utilisera dans ce cours une notation proche de celle de la forme de Backus-Naur (BNF) pour la description de règles syntaxiques pour notre pseudo-langage
- Pseudo-langage : série de conventions qui ressemble à un langage de programmation authentique

Présentation générale de Java

Petit historique du langage

- On peut faire remonter la naissance de Java à 1991. A cette époque, des ingénieurs de chez SUN ont cherché à concevoir un langage applicable à de petits appareils électriques (on parle de code embarqué).
  Pour ce faire, ils se sont fondés sur une syntaxe très proche de celle de C++, en reprenant le concept de machine virtuelle.
  L'idée consistait à traduire, d'abord un programme source, non pas directement en langage machine, mais dans un pseudo langage universel, disposant des fonctionnalités communes à toutes les machines.
  Ce code intermédiaire, dont on dit qu'il est formé de byte codes, se trouve ainsi compact et portable sur n'importe quel ordinateur ; il suffit simplement que ce dernier dispose d'un programme approprié (on parle alors de machine virtuelle) permettant de l'interpréter dans le langage de la machine concernée.

Présentation générale de Java

Petit historique du langage

- En fait, ce projet de code embarqué n'a pas abouti en tant que tel. Mais ces concepts ont été repris en 1995 dans la réalisation du logiciel *HotJava*, un navigateur Web écrit par SUN en Java et capable d'exécuter des *applets* écrits précisément en *byte* codes.
- Les autres navigateurs Web ont suivi, ce qui a contribué à l'essor du langage qui a beaucoup évolué depuis cette date, sous forme de versions successives.

Présentation générale de Java

A ne pas confondre !

- Langage Java : (Sun) pour écrire des programmes Java (Source xxx.java) compilés (pseudocode xxx.class) et exécutables dans une machine virtuelle quelconque.
- Applet Java : programme Java destiné à être appelé dans une page HTML. Exécution sur les machines clientes.
- Servlet Java : applet destinée à être exécuté côté serveur.
- JavaBeans : composants logiciels Java réutilisables.
- JavaScript : origine (Netscape) et objectif différents du langage Java : langage de script (non Orienté Objet) intégré dans les pages HTML et exécutables au sein du navigateur.

Le texte d'un premier programme JAVA

En général, un programme JAVA a cet aspect :

```
public class Main
{
    public static void main(String[] args)
    {
        System.out.println("Hello World ! ");
    }
}
```

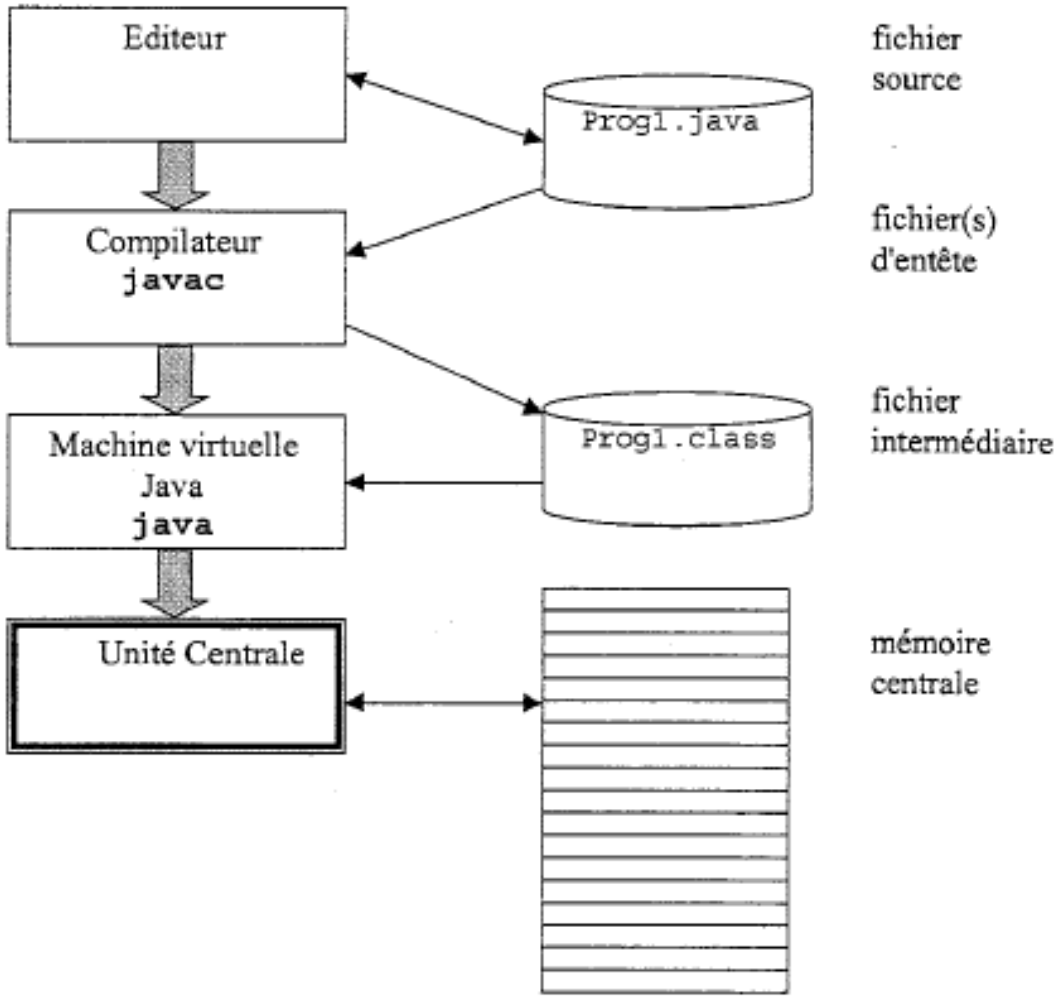
Java est un langage de Programmation Orienté Objets (P.O.O.). Par nature, un programme est composé d'une classe ou de la réunion de plusieurs classes. Il sera impossible de créer un programme n'utilisant aucune classe.

Exécution d'un programme

- Les programmes JAVA sont saisis dans un fichier texte appelé le **fichier source**. Le nom de ce fichier est le nom de la classe suffixé par **.java**
- Pour que ce code source puisse être compréhensible par un ordinateur, il faut utiliser un logiciel appelé compilateur, qui le traduira dans un langage intermédiaire, stocké dans un fichier suffixé **.class**
- Dans le cas de Java, le langage intermédiaire correspond au langage d'une machine virtuelle, appelée la machine virtuelle Java. Les ordinateurs ne comprennent directement qu'un seul langage (appelé langage machine) qui, en général, est différent du langage intermédiaire de java.
- La machine virtuelle Java est donc interprétée par un autre logiciel qui est capable d'exécuter ce langage intermédiaire. Ce logiciel donne toujours le contrôle au sous-programme main.

Nous allons mettre en œuvre tout ce processus (édition, compilation, exécution) avec l'aide d'un EDI (IDE en anglais) libre, appelé *Geany*.

Exécution d'un programme



Programme et instructions

Instructions

Le traitement que le programme doit exécuter est décrit dans le sous-programme *main* par une séquence d'instructions.
Les instructions se terminent généralement par un point virgule (;)

Commentaires

- jusqu'à la fin de la ligne :

```
// Ceci est un commentaire
```

- commentaire sur plusieurs lignes :

```
/*
...
...
...
*/
```

Types

Booléens : *boolean*

true, *false*

Caractères : *char*

Les caractères sont codés sur 16 bits. Les constantes du type caractère sont exprimées entre deux apostrophes : 'a'
Quelques caractères spéciaux :

- `\n` : la nouvelle ligne
- `\t` la tabulation
- `\` la « simple quote »
- `\\` la barre oblique inverse (« backslash »)

Types

Chaîne : *String*

- Elles appartiennent au type non primitif *String*
- Les constantes du type String apparaissent entre guillemets : "Hello World !"
- La concaténation utilise l'opérateur + : "Hello " + "World !"

Types

Entiers : *byte, short, int, long*

- Codés en binaire en complément à deux, les *byte* sur 1 octet, les *short* sur 2 octets, les *int* sur 4 octets et les *long* sur 8 octets.
- Nous pouvons écrire les entiers en notation décimale, en notation octale (préfixée par 0) ou hexadécimale (préfixée par 0x).
- Nous utiliserons le type *int* lors de la traduction du mot clé *entier* de l'algorithmique.

Réels : *float, double*

- Codés en virgule flottante selon le format IEEE 754-1985, les *float* sur 4 octets et les *double* sur 8 octets.
- Les constantes réelles sont des chiffres décimaux comprenant un point décimal, suivies d'un exposant optionnel.
- Nous utiliserons le type *double* lors de la traduction du mot clé *réel* de l'algorithmique

Types numériques

Type Numérique	Plage
Byte (octet)	0 à 255
Entier simple	-32 768 à 32 767
Entier long	-2 147 483 648 à 2 147 483 647
Réel simple	$-3{,}40\times10^{38}$ à $-1{,}40\times10^{45}$ pour les valeurs négatives $1{,}40\times10^{-45}$ à $3{,}40\times10^{38}$ pour les valeurs positives
Réel double	$1{,}79\times10^{308}$ à $-4{,}94\times10^{-324}$ pour les valeurs négatives $4{,}94\times10^{-324}$ à $1{,}79\times10^{308}$ pour les valeurs positives

Opérateurs

Opérateurs arithmétiques

- $+$
- $-$
- $*$
- / (entière si elle s'applique sur deux entiers, réelle sinon)
- % (modulo : reste de la division entière)

Opérateurs d'affectation

- var = exp ;
- Par exemple $b = i * 4$;

Opérateurs

Opérateurs de comparaison

- $<$
- $<=$
- $>$
- $>=$
- $==$ (égalité)
- $!=$ (différent)

Pour le type booléen, seuls l'égalité et l'inégalité peuvent être utilisés.

Opérateurs

Opérateurs logiques

- && (ET logique)
- || (OU logique)
- ! (négation)

Exemples :

```
x >= 3 && x <= 7
x < 2 || x > 6
!(x > 3 && x <= 7)
```

L'évaluation d'une expression logique se fait de gauche à droite. Lors de l'évaluation d'une expression logique, l'opérande à droite n'est évalué que si la connaissance de sa valeur est indispensable pour décider si l'expression est vraie ou fausse.

- `expr1 && expr2` : *expr2* n'est évalué que si *expr1* est vrai
- `expr1 || expr2` : *expr2* n'est évalué que si *expr1* est faux

Table d'opérateurs

Opérateur	Priorité	Associativité
$-$ unaire	1	
$*$ / div mod	2	Gauche
$+$ $-$	3	Gauche
$==$, $!=$, $<$, $>$, $<=$, $>=$	4	Impossible
non unaire	5	
et	6	Gauche
ou	7	Gauche

Opérateurs

Opérateurs d'incrémentation et de décrémentation

- ++ (incrémentation)
- −− (décrémentation)

Exemple : i=i+1 ; peut être remplacée par i++ ; ou ++i ;
La valeur de cette expression diffère en fonction de la position de l'opérateur.
Exemple : si la valeur de i est 5

- L'expression n = ++i − 5, donnera à *i* la valeur 6 et à *n* la valeur 1
- En revanche, l'expression n = i++ - 5, affectera à *i* la valeur 6 et à *n* la valeur 0 !

JAVA dispose aussi de ce qu'on appelle des *Opérateurs d'affectation élargie*.
Nous pouvons remplacer

- i = i+k par i += k, ou encore,
- a = a*b par a *= b.

Ces opérateurs permettent de condenser l'écriture de certaines instructions.
Attention : i++ est la même chose que i+=1.

Conversion de types

Définition

Java est un langage fortement typé. Le compilateur vérifie la cohérence des types des opérandes dans les expressions. Ces types doivent être identiques ou compatibles. Tout type numérique est compatible avec un type numérique de capacité de codage supérieure.

```
byte < short < int < long < float < double
```

Lorsque les opérandes sont de types différents, nous pouvons réaliser une opération de conversion explicite (cast).

```
var = (type) exp
```

Exemple : Nous disposons de deux réels (r1 et r2), et nous devons obtenir leur « division entière » ... Pour ce faire, nous pouvons forcer la conversion de ces deux réels en entiers. Si z est un entier, nous faisons :

```
z = (int) r1 / (int) r2
```

Conversions implicites

entier → flottant

si un opérateur nécessite un flottant, et qu'un entier est fourni, cet entier est automatiquement converti en flottant.

tout type → chaine

si un opérateur nécessite une chaîne, et qu'une valeur d'un autre type est fournie, cette valeur est automatiquement converti en chaine (représentation textuelle de la valeur).

exemples

- 1 + false : toujours pas de sens
- 2 * 4.2 → 2.0 * 4.2 → 8.4
- "pi ≈" + 3.14 → "pi ≈" + "3.14" → "pi ≈ 3.14"

Les variables

Noms de variables

- Suite de caractères alphanumériques (lettres ou chiffres). Le premier caractère doit être une lettre. On fait la distinction entre majuscules et minuscules.
- Par convention, un nom de variable commence avec une lettre **minuscule**, et est composé par de lettres ou de chiffres.
- D'habitude, on utilise les majuscules pour mettre en évidence la première lettre d'un mot dans une suite de mots, par exemple, *codeMatiere*.
- Il est conseillé de ne pas utiliser des lettres accentuées dans un nom de variable ou de constante.
- Il y a une cinquantaine de mots clés, propres au langage, qui ne peuvent pas être utilisés comme noms de variables.

Déclaration des variables

Syntaxe

type id ;

Exemples

```
int i;
double quantite;
boolean estPremier;
```

Il est possible de donner une valeur initiale à une variable lors de la déclaration :

Exemples

```
int i = 0;
boolean estPremier = false;
```

Déclaration de constantes

Syntaxe

- On les déclare comme des variables initialisées, en précédant la déclaration par le mot clé *final*.
- Par convention, les noms de constantes sont entièrement en majuscules, avec le caractère _ pour séparer les mots.

Exemple

```
final double TVA = 19.6 ;
```

Le mot clé *final* sert à distinguer une variable qui peut être modifiée d'une constante qui ne changera pas de valeur.

Affichage des résultats

Syntaxe

Pour afficher nos résultats à écran, nous avons plusieurs possibilités. Le plus facile est d'utiliser :

- Pour afficher une chaîne de caractères ou la valeur d'une variable ou d'une constante : *System.out.print() ;*
- Pour afficher une chaîne de caractères ou la valeur d'une variable ou d'une constante et passer à la ligne suivante : *System.out.println() ;*

Exemple (si on suppose que nous avons une variable affectée prix)

```
System.out.println("Le prix avec la TVA est de "+ prix);
```

Lecture des données

La classe *Lire.java*

- Pour simplifier les opérations de lecture, nous utiliserons la classe Lire.java, que nous devons toujours copier dans le répertoire de compilation (dans le même répertoire que la classe que nous sommes en train de développer).
- La forme d'une instruction de lecture de donnée est :

```
nom_variable_à_lire=Lire.première_lettre_type_variable();
```

Soit si *a* est une variable de type *int* (déclarée préalablement), que l'on veut lire au clavier, on écrira : `a=Lire.i();`
Ainsi, les opérations de lecture sont définies pour chaque type de base.

La classe *Lire.java*

- Pour les chaînes de caractères :

```
String a ;
a = Lire.S() ;
```

- Pour les byte :

```
byte a;
a = Lire.b() ;
```

- Pour les short :

```
short a
a = Lire.s() ;
```

- Pour les int :

```
int a
a = Lire.i() ;
```

- Pour les long :

```
long a
a = Lire.l() ;
```

La classe *Lire.java*

- Pour les double :

```
double a
a = Lire.d() ;
```

- Pour les float :

```
float a
a = Lire.f() ;
```

- Pour les char :

```
char a
a = Lire.c() ;
```

Conventions d'écriture des programmes

Pour les noms de programmes

- Le nom d'un programme comportant un seul mot (français ou anglais) est écrit entièrement en minuscules, excepté la première lettre. Par exemple on écrit `Bonjour` plutôt que `bonjour`.
- Le nom d'un programme comportant plusieurs mots est écrit de la façon suivante : la première lettre de chaque mot est en majuscule, toutes les autres lettres étant en minuscules. Les différents mots sont collés les uns aux autres. Par exemple on écrira `CalculDeLaMoyenne` plutôt que `calcul_de_la_moyenne` ou toute autre solution.

Pour la présentation des programmes

- Un programme ne comporte qu'une seule instruction par ligne.
- Toutes les lignes comprises entre une accolade ouvrante et l'accolade fermante qui lui correspond sont indentées d'un cran vers la droite.
- Une accolade fermante est seule sur sa ligne et est alignée avec le

Exercices

Voir sur Moodle le document "Premier programme java.pdf"