INFO0101

INTRODUCTION À L'ALGORITHMIQUE ET À LA PROGRAMMATION

COURS 2

LE LANGAGE JAVA

Pierre Delisle, Cyril Rabat, Christophe Jaillet et Jessica Jonquet
Département de Mathématiques, Mécanique et Informatique
Septembre 2020

Plan de la séance

- Structure d'un programme Java
- Variables, types
- Mots-clés
- Opérateurs
- Structures de sélection
- Boucles

Historique de Java

- Créé en 1991 par SUN
- Conçu originellement pour code embarqué et applets pour navigateurs Web
- Versions

1996	1998	1999	2000	2002	2004	2006	2011	2014	2017	2018	2018	2019
1.0	1.1	1.2 Devient Java 2	1.3	1.4	J2 SE 5.0	Java SE 6.0	Java SE 7	Java SE 8	Java SE 9	Java SE 10	Java SE 11	Java SE 12-13

- Aujourd'hui
  - Java SE 11 : version LTS (Long Term Support)
  - Java SE 14 (JDK 14) : dernière version

Java : langage de Programmation Orientée Objet (POO)

- Objets, classes, encapsulation
- Héritage
- Permet la programmation structurée
- 2 catégories de programmes
  - Interface console
    - Écran + clavier
  - Interface graphique (GUI)
    - Souris, boutons, fenêtres, boîtes de dialogue, etc.
- Portabilité → JVM – Machine virtuelle

Étapes de programmation

- Écrire le texte du programme dans un fichier (source) à l'aide d'un éditeur de texte
  - NomDuProgramme.java
- Compiler le fichier source
  - javac NomDuProgramme.java
  - Génère le fichier NomDuProgramme.class
- Exécuter le programme
  - java NomDuProgramme (pas d'extension)

- Édition de code : Notepad++
  - Logiciel libre fonctionnant sous Windows
  - http://notepad-plus-plus.org/

- Compilation du code : Compilateur
  - Programme qui transforme un code source (fichier texte) en code exécutable (objet)
- Il existe des dizaines de compilateurs pour divers langages
  - C, C++, Fortran, Pascal, Basic, Cobol, …
- Particularité de Java
  - Il ne compile pas du code exécutable directement, mais du « bytecode » interprété par la machine virtuelle Java (JVM)

PREMIER PROGRAMME JAVA

Structure d'un programme

```
class PremierProg {
        public static void main (String [ ] args) {
                System.out.println("Mon premier programme Java (et non le dernier !)");
        }
}
```

- Mots clés (que l'on verra en détails plus tard)
  - public, class, static, void, main, String
  - Pour l'instant, on les place sans se poser de questions !
- Accolades ({ … })
  - début et fin d'un bloc de code
- PremierProg
  - Nom du programme
- public static void main (String [] args)
  - Procédure principale
- System.out.println
  - routine Java d'affichage à l'écran

Entrées et sorties

Affichage à l'écran

- System.out.print("Bonjour !");
  - Affiche ce qui est entre les guillemets et demeure sur la même ligne, à la position qui suit le dernier caractère affiché
- System.out.println("Bonjour !");
  - Affiche ce qui est entre les guillemets et se place en début de ligne suivante

Entrées au clavier

- Importation de la classe Scanner
  - import java.util.Scanner ;
- Déclaration d'une variable de type Scanner
  - Scanner clavier = new Scanner(System.in)
- Lecture d'une entrée d'un format donné
  - clavier.nextLine() ; /*Chaîne de caractères*/
  - clavier.nextInt() ; /*Entier*/
  - clavier.nextDouble() ; /*Réel*/
- ...
- Son utilisation sera vue en TP

Règles d'écriture d'un programme

- Identificateurs
- Mots-clés
- Séparateurs
- Format libre
- Commentaires

Identificateurs et mots-clés

Identificateur

- Suite de caractères pour désigner les différentes entités d'un programme (variables, fonctions, classes, etc.)
  - nom, Code_Programme, Valeur2, …
- Se compose de lettres, de chiffres et du caractère « _ »
- Doit commencer par une lettre
- Distingue les majuscules et les minuscules
  - Nom et nom sont 2 identificateurs différents !

Mot-clé

- Mot réservé par le langage qui ne peut être utilisé comme identificateur
  - int
  - while
  - if
  - break
  - float
  - public
  - static
  - Etc…

Mise en forme d'un programme

- Séparateur
  - 2 identificateurs/mots-clés successifs doivent être séparés par un espace blanc ou par une fin de ligne
    - int x;
    - String nom;
    - public static …
    - Etc…
- Format libre
  - Un fichier source peut être mis en page de plusieurs façons
  - Le code

```
int x = 2;
int y = 3;
x = x + y;
```

  - Peut être écrit

```
int x = 2; int y = 3; x = x + y;
```

  - Mais attention à la lisibilité !
- Commentaire
  - Permet d'écrire du texte qui n'est pas traité par le compilateur (explications, détails, etc.)
  - Zone de texte en commentaire
    - Texte entre les caractères « /* » et « */ »

```
/*  Voici un commentaire */
```

- Commentaire de fin de ligne
  - Texte qui suit les caractères « // »

```
int x; int y;  //  Voici un commentaire de fin de ligne
```

Programmer en Java à la maison sous Windows

- Télécharger le JDK (Java SE Development Kit)
  - https://www.oracle.com/java/technologies/javase-downloads.html#javasejdk
  - Dans la section Java SE 14, cliquer sur « JDK Download »
  - Aller à la section « Java SE Development Kit 14.0.2 », sous-section « Product / File Description »
  - Cliquer sur le lien « jdk-14.0.2_... » qui correspond à votre système (probablement Windows)
  - Suivre les instructions
- Installer le JDK
  - Exécuter le fichier téléchargé (ex : jdk-14.0.2_windows-x64_bin.exe)
  - Suivre les instructions
- Vous trouverez la documentation d'installation à l'adresse donnée ci-haut
- Programmer en Java à la maison sous Linux
  - Si vous êtes déjà un utilisateur de Linux, vous devriez être capable de vous organiser !

TYPES DE DONNÉES JAVA

Entier

- Un bit est réservé au signe
  - 0 → positif
  - 1 → négatif
- Les autres bits servent à représenter
  - La valeur absolue du nombre pour les positifs
  - Le complément à 2 du nombre pour les négatifs

Type	Taille (octets)	Valeur minimale	Valeur Maximale
byte	1	-128 (Byte.MIN_VALUE)	127 (Byte.MAX_VALUE)
short	2	-32768 (Short.MIN_VALUE)	32767 (Short.MAX_VALUE)
int	4	-2147483648 (Integer.MIN_VALUE)	2147483647 (Integer.MAX_VALUE)
long	8	-9,22E+018 (Long.MIN_VALUE)	9,22E+018 (Long.MAX_VALUE)

```
int n; //variable de type entier
```

Flottant

- Permet de représenter (de manière approchée) une partie des nombres réels
- Représentation : s M * 2E
  - s : signe (+1 ou -1)
  - M : Mantisse
  - E : Exposant
- 2 éléments à considérer
  - Précision
  - Domaine couvert
- Dans un programme, les constantes flottantes peuvent s'écrire de 2 façons
  - Décimale → 6567.4552
  - Exponentielle → 6.5674552e3

Type	Taille (octets)	Valeur minimale	Valeur Maximale
float	4	1,40239846E-45 (Float.MIN_VALUE)	3,40282347E38 (Float.MAX_VALUE)
double	8	4,9406…..E-324 (Double.MIN_VALUE)	1,7976…..E308 (Double.MAX_VALUE)

```
double n; //variable de type flottant
```

Caractère

- Représenté en Unicode plutôt qu'en ASCII
- 2 octets plutôt qu'un
  - 65536 combinaisons plutôt que 128
- Mot clé *char*

```
char c1; //variable de type caractère
```

- Constantes de type caractère
  - Entre apostrophes
  - 'a' 'E' '+'

Booléen

- Représente une valeur logique de type vrai/faux
  - Notées *true* et *false*
- Mot clé *boolean*

```
boolean ordonne;          //variable de type booléen
ordonne = n < p;          //affectation
```

Initialisation de variables

- Une variable doit être déclarée et initialisée avant d'être utilisée

```
int n;          //déclaration
n = 15;         //initialisation
```

- ou

```
int n = 15;     //déclaration et initialisation
                //dans la même instruction
```

- Peut être effectué n'importe où dans le programme

- Constante
  - On peut déclarer une variable en obligeant qu'elle ne soit pas modifiée ultérieurement
  - Mot clé *final*

```
final int n = 20;   //n est déclaré constant
n = 12;             //erreur
```

- L'initialisation n'a pas à être faite en même temps que la déclaration, mais doit être faite une seule fois dans le programme

LES OPÉRATEURS EN JAVA

Opérateurs arithmétiques

- Binaires

Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo	%

- Unaires

Opposé	-
Identité	+

- Conversions implicites : ajustement de type

```
int n, p;
float x, y;
y = n * x + p;
```

- Conversion de n en float pour la multiplication, résultat de type float
- Conversion de p en float pour l'addition, résultat de type float
- Ne doit pas dénaturer la valeur initiale
  - int → long → float → double

Opérateurs relationnels

- A priori définis pour des opérandes de même type, mais aussi soumis aux conversions implicites

Opérateur	Signification
<	Inférieur à
<=	Inférieur ou égal à
>	Supérieur à
>=	Supérieur ou égal à
==	Égal à
!=	Différent de

Opérateurs logiques

Opérateur	Signification
!	Négation
&	Et
^	Ou exclusif
|	Ou inclusif
&&	Et (avec court-circuit)
||	Ou inclusif (avec court-circuit)

- Court-circuit : second opérande évalué seulement si nécessaire
  - a < b && c < d
  - Si « a < b » est faux, inutile d'évaluer « c < d »

Opérateur d'affectation

- Opérateur =
- Le premier opérande doit être une référence à un emplacement dont on peut modifier la valeur (variable, objet, élément de tableau)
  - i = 5;
  - x = i + 8;
- Soumis aux règles de conversion implicites
  - byte → short → int → long → float → double
  - char → int → long → float → double

Opérateurs d'incrémentation et de décrémentation

- Opérateur ++ → incrémente une variable
  - i++ → équivalent à « i = i + 1 »
- Opérateur -- → décrémente une variable
  - i-- → équivalent à « i = i - 1 »
- À gauche : préincrémentation → ++i
- À droite : postincrémentation → i++
- Effet sur l'expression, non sur la variable
  - si i = 5

```
n = ++i - 5        //i = 6, n = 1
n = i++ - 5        //i = 6, n = 0
```

Opérateurs d'affectation élargie

- Une expression comme « i = i + k »
- Peut être remplacé par « i += k »
- a = a * b → a *= b
- De manière générale
  - Variable = Variable opérateur expression
  - Peut être remplacé par
  - Variable opérateur= expression
- Valable pour opérateur arithmétique seulement
  - « a <= b » n'est pas une affectation !

Opérateurs de cast (transtypage)

- Cast : forcer la conversion d'une expression dans un type de son choix
  - (type) expression
- Exemple : si n et p sont de type int
  - (double) (n/p)
  - Convertit l'expression entière en double
- Attention aux parenthèses! Si n = 10 et p = 3
  - (double) (n / p) → l'expression vaut 3.0
  - (double) n / p → l'expression vaut 3.3333…

PREMIÈRES STRUCTURES DE SÉLECTION EN JAVA

Préambule : blocs et instructions

- Instruction
  - Simple : terminé par un ;
    - i = 12 + b;
  - Structurée : choix, boucles → if…else…
  - Une instruction structurée peut contenir une ou plusieurs instructions simples/structurées(blocs)
- Bloc
  - Suite d'instructions placées entre accolades ({…})
  - Peut être vide
    - {}
  - Peut contenir 1 instruction
    - {n = 4;}
  - Peut contenir plusieurs instructions
    - {n = 4; i = 2;}

if … else (Si…alors…Sinon)

```
if (condition)
    Instruction 1
[  else
    Instruction 2]
```

Note : les crochets signifient que ce qui est à l'intérieur est facultatif. Ils ne font pas partie de la syntaxe

- Instruction1 et Instruction2
  - Instructions simples
  - Instructions structurées
  - Bloc

```
if (i < 10) {
    System.out.println("Plus petit que 10");
}
else {
    i -= i;
    System.out.println("Toujours plus petit que 10");
}
```

switch (Cas…Parmi)

```
switch (expression) {
    case Constante1 :
        Instruction1
        break;
    case Constante2 :
        Instruction2
        break;
    ...
    case Constante_n :
        Instruction_n
        break;
    default :
        InstructionDefaut
}
```

```
switch (n) {
    case 0 :
        System.out.println("nul");
        break;
    case 1 :
        System.out.println("un");
        break;
    default :
        System.out.println("trop grand");
}
```

STRUCTURES ITÉRATIVES EN JAVA

for (Pour)

- Prototype

```
for (initialisation; condition; incrémentation)
    Instruction
```

- Exemple

```
for (i = 1; i <  5; i++) {
    System.out.print ("Bonjour");
    System.out.println (i + " fois");
}
```

while (TantQue)

- Prototype

```
while (condition)
    Instruction
```

- Exemple

```
n = 15;
while (n < 100) {
    n += 3;
    System.out.println ("Toujours plus petit que 100");
}
```

do…while (Faire…TantQue)

- Prototype

```
do
    Instruction
while (condition);
```

- Exemple

```
n = 15;
do {
    n += 3;
    System.out.println ("Toujours plus petit que 100");
} while (n < 100) ;
```

PROCHAIN COURS :

FONCTIONS ET PROCÉDURES