Premier programme

Maria-Virginia Aponte, François Barthélémy

NFA031

1 Objectifs du cours

Les objectifs du cours sont d'apprendre les bases de la programmation en utilisant le langage Java comme support. Ces bases sont pratiquement communes avec l'écrasante majorité des langages de programmation et nous éviterons d'utiliser ce qui est trop spécifique au langage Java.

Les notions étudiées seront les suivantes :

— variable, type, expression
— l'instruction conditionnelle (if)
— les boucles (for, while)
— les tableaux
— les méthodes
— l'utilisation des chaînes de caractères
— classes et objets
— la structure d'ArrayList

Le but du cours est de vous donner non seulement des connaissances, mais également une compétence : celle de réaliser concrètement de petits programmes Java utilisant les constructions énumérées ci-dessus et de les exécuter sur un ordinateur.

2 Qu'est-ce qu'un programme ?

Il existe toutes sortes de programmes qui peuvent s'exécuter sur différents dispositifs pour rendre un service. Nous n'allons pas essayer de définir ce qu'est un programme en général, mais présenter le type de programmes que nous allons écrire dans ce cours.

Il s'agit de programmes applicatifs s'exécutant sur un ordinateur et composés d'une suite d'instructions. Le programme utilise des données dont certaines sont inscrites dans le programme, dans des instructions et d'autres viennent de l'extérieur : elles sont tapées au clavier par l'utilisateur. Les programmes auront des résultats qui s'afficheront à l'écran.

3 Programme source et programme cible

Chaque programme a deux formes différentes : une forme qui est le programme tel qu'il est écrit par le programmeur et l'autre est le programme tel qu'il est exécuté par l'ordinateur. Ces deux formes sont différentes : le programme que l'on écrit a une forme textuelle. On peut le lire : il contient des mots-clés et des noms qui ont un sens. Mais l'ordinateur ne sait pas exécuter directement ce texte. Le programme qui s'exécute a une forme binaire : la fameuse suite de 0 et de 1 qui est copiée dans la mémoire de l'ordinateur.

3.1 Traduction : compilation et interprétation

Pour passer d'une forme à l'autre, il faut une traduction qui est exécutée automatiquement par un programme. Il existe deux sortes de traduction : la traduction réalisée une fois pour toutes

par un programme appelé un **compilateur** ou la traduction réalisée à chaque exécution par un logiciel appelé **interpréteur**.

Le compilateur est analogue à la traduction d'un livre alors que l'interpréteur est analogue à la traduction simultanée utilisée pour traduire en direct les propos tenus dans une langue étrangère.

Le langage Java que nous utilisons mixe les deux procédés avec utilisation d'un compilateur puis d'un interpréteur.

Dans cette histoire, le langage des programmes écrits par les programmeurs s'appelle le **langage source** de l'ordinateur. C'est le langage Java qui est unique pour tous les ordinateurs du monde. Le langage des programmes exécutés s'appelle le **langage cible** et il est différent pour chaque type de processeur, d'architecture et de système d'exploitation employé.

Dans le cas de Java, il y a un troisième langage qui est celui des programmes compilés par le compilateur Java. On l'appelle **langage intermédiaire**, langage de la machine virtuelle Java. Il est unique pour tous les ordinateurs mais les machines ne savent pas l'exécuter directement. C'est pour cela qu'il y a besoin d'un interpréteur qui interprète le code intermédiaire : la machine virtuelle Java.

3.2 Étapes pour la création d'un programme

La création de programmes nécessite l'utilisation de différents outils qui peuvent soit être des outils séparés, soit des outils regroupés dans un unique logiciel appelé environnement de développement intégré (IDE en anglais).

Il y a au minimum trois outils : un éditeur de texte, un compilateur et un interpréteur.

L'**éditeur de texte** permet de créer le fichier contenant le programme source. Il s'agit d'une sorte de traitement de texte mais qui ne comporte pas d'élément de mise en page car le but n'est pas d'imprimer un texte. Le format de fichier utilisé est le format texte brut, celui des fichiers .txt en Windows. Dans le cas des programmes sources Java, on n'utilise pas cette extension .txt mais l'extension `.java`. Cette extension doit être utilisée quel que soit le système d'exploitation (Windows, macOS, Linux).

Le **compilateur** prend en entrée le fichier texte contenant le programme source. Si le programme source est correct, le compilateur crée un ou plusieurs fichiers binaires contenant le programme en langage intermédiaire. Ces fichiers ont l'extension `.class`.

L'**interpréteur** prend en entrée un fichier contenant du code intermédiaire et exécute le code correspondant. Pour ce faire, il traduit une ligne de code, puis l'exécute immédiatement avant de passer à la suivante.

Pour ce cours, nous vous conseillons d'utiliser un environnement professionnel : Eclipse. Il est un peu difficile à prendre en main au début, car il a de nombreuses fonctionnalités. Il existe d'autres outils professionnels comme par exemple NetBeans, VS Code ou IntelliJ. Vous pouvez les utiliser si vous le souhaitez, mais nous ne vous fournirons pas d'aide pour eux.

4 Premier programme : "Hello World"

Commençons par le programme le plus simple possible en Java :

Listing 1 – Premier programme simple

```
package nfa031;
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Bonjour␣tout␣le␣monde␣!");
    }
}
```

Ce programme ne fait qu'afficher un message à l'écran. Il constitue un excellent point de départ pour comprendre la structure de base d'un programme Java.

4.1 Structure générale d'un programme Java

Tout programme Java a le squelette suivant :

Listing 2 – Structure générale d'un programme Java

```
package nompackage;
public class NomClasse {
    public static void main(String[] args) {
        // Vos instructions ici
    }
}
```

Éléments importants :

- **Package** `package nfa031;` Cette ligne déclare le **package** (ou paquetage) auquel appartient notre classe. La notion de package est utile pour organiser les gros programmes en parties et sous-parties. Dans ce cours, nous ne ferons que de petits programmes mais nous prendrons l'habitude de faire une déclaration de package même si ce n'est pas forcément nécessaire. Le nom du package (nompackage) est choisi pas le programmeur.
- **Classe principale :** Tout programme Java est composé au minimum d'une classe (mot-clé `class`) dite principale, qui elle-même contient une méthode de nom `main`.
- **Mots réservés :** `package`, `public`, `class`, `static`, `void` sont des mots réservés qui ont un sens particulier pour Java. On ne peut donc pas les utiliser comme nom pour des classes, des variables, etc.
- **Nom du fichier :** Le nom de la classe principale donne son nom au programme tout entier et doit être également celui du fichier contenant le programme, complété de l'extension `.java`.
- **Méthode main :** Obligatoire dans tout programme Java : c'est elle qui "commande" l'exécution. Nous aurons toujours la même ligne d'entête `public static void main(String[] args) {`, dans tous les programmes. Ensuite, les instructions différeront d'un programme à l'autre, de même que le nom du programme (le nom de la classe).
- **Convention :** Par convention, le nom d'une classe commence par une majuscule. Le nom d'un package est écrit en minuscule.

5 Un exemple plus complexe : calcul de l'âge

Étudions maintenant un programme plus intéressant qui calcule l'âge d'une personne :

Listing 3 – Calcul de l'âge

```
package nfa031;
public class CalculAge {
    public static void main(String[] args) {
        int anneeNaissance;
        int anneeActuelle;
        int age;

        System.out.println("En␣quelle␣année␣êtes-vous␣né(e)␣?");
        anneeNaissance = Terminal.lireInt();
        System.out.println("En␣quelle␣année␣sommes-nous␣?");
        anneeActuelle = Terminal.lireInt();
        age = anneeActuelle - anneeNaissance;
        System.out.println("Vous␣avez␣" + age + "␣ans.");
    }
}
```

Ce programme illustre les concepts fondamentaux que nous allons détailler.

Certaines données sont représentées par des noms. C'est le cas de `anneeNaissance`, `anneeActuelle` et `age`. Ces noms désignent des nombres entiers (par exemple 1998, 2025 et 27) qui ne sont pas connus au moment où l'on écrit le programme mais qui seront connus à un certain point de l'exécution du programme et ces nombres peuvent être différents lors de deux exécutions. Ces noms sont ceux de variables.

Chaque ligne du programme est une instruction. Il y a ici trois sortes d'instructions :

— Des déclarations de variables (les trois premières lignes)

— Des invocations de méthodes. Ces instructions, dans notre exemple servent à afficher des informations à l'écran, que ce soit pour signifier que le programme attend l'entrée d'une donnée au clavier ou pour afficher le résultat à la fin. La méthode invoquée ici est `System.out.println`.

— Des affectations qui donnent une valeur à une variable. Ces instructions comportent le nom de la variable, le signe = (égal) et un calcul qui détermine la valeur à donner à la variable.

Nous allons détailler les différents constituants du programme pour être plus précis dans nos explications.

6 Les variables

Les variables sont utilisées pour stocker les données du programme. À chaque variable correspond un emplacement en mémoire, où la donnée est stockée, et un nom qui sert à désigner cette donnée tout au long du programme.

6.1 Déclaration des variables

Une variable doit être déclarée dans le programme avant de pouvoir être utilisée. La syntaxe est :

`type nomVariable;`

Exemples dans notre programme :

```
int anneeNaissance;
int anneeActuelle;
int age;
```

Règles pour les noms de variables :

— Peuvent contenir des lettres, des chiffres, le signe souligné (_) et le dollar ($)

— Ne doivent pas commencer par un chiffre

— Ne peuvent pas être un mot réservé (`int`, `class`, `public`, etc.)

— Respectent la casse (majuscule ou minuscule) : `age` et `Age` sont deux noms différents

6.2 Types de données de base

Java propose plusieurs types de données prédéfinis :

— `int` : nombres entiers (ex : 42, -15)

— `double` : nombres à virgule (ex : 3.14, -2.5)

— `char` : un caractère (ex : 'A', '5')

— `boolean` : valeurs vraies ou fausses (`true`, `false`)

— `String` : chaînes de caractères (ex : "Bonjour")

6.3 Affectation de valeurs

L'instruction d'affectation permet de donner une valeur à une variable :

`nomVariable = expression;`

Exemples :

```
age = 25;
age = anneeActuelle - anneeNaissance;
```

Important : Le symbole `=` ne représente pas une égalité mathématique, mais une affectation. Il faut lire "la variable reçoit la valeur".

6.4 Déclaration avec valeur initiale

Il est possible de faire une déclaration de variable qui contient une affectation. Cela permet de donner une valeur à la variable dès sa création et cette pratique est souvent encouragée. On parle d'initialisation de la variable et la valeur donnée est appelée la valeur initiale de la variable.

Exemples :

```
int anneeActuelle = 2025;
int anneeNaissance = 2025 - 30;
```

7 Les expressions avec opérateurs

Des expressions utilisant des opérateurs peuvent permettre de calculer une valeur qui peut être utilisée pour donner une valeur à une variable ou qui peut être affichée au moyen de l'instruction d'affichage.

Les opérateurs permettent de faire des calculs avec plusieurs valeurs.

7.1 Les expressions arithmétiques

On peut construire des expressions en utilisant les opérateurs arithmétiques :
— `+` : addition
— `-` : soustraction
— `*` : multiplication
— `/` : division
Exemple : `resultat = (a + b) * 2;`

7.2 L'opérateur de concaténation

Le signe + ne désigne pas toujours une opération arithmétique d'addition. Si l'une au moins des valeurs utilisées dans l'expression est une chaîne de caractère, le signe + désigne la concaténation de chaînes de caractères.

Cela est utilisé dans notre programme à la ligne :

```
System.out.println("Vous␣avez␣" + age + "␣ans.");
```

Ici, les signes + permettent de créer une nouvelle chaîne de caractères commençant par les caractères de `"Vous avez "` suivi des chiffres de l'âge contenu dans la variable `age` suivi des caractères de la chaîne `" ans."`. La chaîne issue de ce calcul est ensuite affichée à l'écran.

8 Entrées et sorties

Les entrées au clavier qui permettent de faire entrer des données dans le programme et les sorties, les affichages à l'écran qui permettent de dialoguer avec l'utilisateur et de livrer des résultats, se font au moyen de *méthodes* qui sont de petits programmes utilitaires réalisant une tâche précise. On les appelle des `sous-programmes`.

8.1 Affichage avec System.out

Pour afficher des informations à l'écran :
— `System.out.print(valeur)` : affiche sans passer à la ligne
— `System.out.println(valeur)` : affiche et passe à la ligne
Exemples :

```
System.out.println("Bonjour");
System.out.println(age);
System.out.println("Vous␣avez␣" + age + "␣ans");
```

L'affichage au moyen de la méthode `System.out.print` ou de la méthode `System.out.println` est une instruction. L'appel ou invocation de la méthode est seul sur une ligne et se suffit à lui-même.

8.2 Saisie avec Terminal

La classe Terminal (spécifique à ce cours) permet de lire des données au clavier :
— `Terminal.lireInt()` : lit un nombre entier
— `Terminal.lireDouble()` : lit un nombre à virgule
— `Terminal.lireString()` : lit une chaîne de caractères
— `Terminal.lireBoolean()` : lit une valeur booléenne
— `Terminal.lireChar()` : lit un caractère
Dans le programme de calcul d'âge, des entiers sont lus au clavier et affectés à une variable comme par exemple dans la ligne :

```
anneeNaissance = Terminal.lireInt();
```

Contrairement à un affichage, une saisie au clavier n'est pas une instruction indépendante. Il faut spécifier ce que le programme fait de la valeur entrée au clavier et cela est précisé au moyen d'une instruction qui contient l'appel de la méthode. Dans cet exemple, l'instruction est une affectation et l'appel de la méthode est une expression qui permet de *calculer* l'entier à mettre dans la variable.

Il y a deux sortes de méthodes en Java : les méthodes qui calculent en renvoient un résultat comme `Terminal.lireInt`. L'appel d'une telle méthode est une expression. Les méthodes qui ne renvoient pas de résultat comme `System.out.print`. L'appel d'une telle méthode produit un effet à l'affichage ou dans la mémoire. C'est une instruction seule sur une ligne de programme.

9 Compilation et exécution

La compilation et l'exécution peuvent s'exécuter en ligne de commande dans une fenêtre terminal (fenêtre de commande en Windows) ou au sein d'un IDE comme Eclipse.

9.1 Compilation

Pour compiler un programme Java en ligne de commande : `javac NomFichier.java`
Si la compilation réussit, un fichier `NomFichier.class` est créé. Sinon, des messages d'erreur s'affichent.

Dans Eclipse, la compilation est faite automatiquement lorsqu'on sauve une fichier (commande save).

9.2 Gestion des erreurs de compilation

Reprenons comme exemple le programme `HelloWorld` mais dans lequel on aurait oublié un point-virgule à la fin de l'unique instruction du programme.

Listing 4 – Programme avec erreur

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Bonjour␣tout␣le␣monde␣!")
    }
}
```

Voici une trace de la compilation du programme en ligne de commande :

```
$ javac HelloWorld.java
HelloWorld.java:3: error: ';' expected
        System.out.println("Bonjour tout le monde !")
                                                     ^
1 error
```

Les messages d'erreur indiquent :
— Le fichier contenant l'erreur (HelloWorld.java)
— Le numéro de ligne (3)
— Une explication en anglais (error : ';' expected)
— La position approximative de l'erreur dans le code, donné par le circonflexe qui donne l'endroit de la ligne de code où l'erreur a été détectée

Sous Eclipse, les erreurs sont signalées par une croix rouge en marge de la ligne où il y a l'erreur. Le message est affiché en passant la souris sur cette croix rouge ou en allant consulter l'onglet nommé `problems` dans la fenêtre du bas.

Conseil : Avoir des erreurs de compilation est normal quand on apprend. Il faut les corriger patiemment une par une.

9.3 Exécution

L'exécution n'est possible qu'après avoir corrigé toutes les erreurs de compilation.

Pour exécuter un programme compilé en ligne de commande : `java NomClasse`

Attention : on utilise le nom de la classe, pas le nom du fichier.

Avec Eclipse, on exécute un programme en appuyant sur le bouton comportant un triangle vert.

10 Exécution détaillée du programme CalculAge

Pour bien comprendre comment fonctionne un programme, observons une exécution complète de `CalculAge` avec des données concrètes.

10.1 La notion de séquence d'instructions

Un programme Java s'exécute de façon **séquentielle** : les instructions sont exécutées une par une, dans l'ordre où elles apparaissent dans le code, de haut en bas. Cette propriété fondamentale signifie que :
— L'instruction ligne 3 s'exécute **avant** l'instruction ligne 4
— On ne peut pas utiliser une variable avant de l'avoir déclarée
— L'ordre des instructions est crucial : changer l'ordre change le comportement du programme

Java utilise un **pointeur d'exécution** invisible qui indique quelle instruction va s'exécuter. Ce pointeur avance ligne après ligne, sauf dans certains cas particuliers (boucles, conditions) que nous verrons plus tard.

10.2 Exécution du programme

Supposons qu'un utilisateur lance le programme `CalculAge` et saisisse l'année 1995 comme année de naissance et 2025 comme année actuelle. Voici ce qui se passe dans la mémoire de l'ordinateur, instruction par instruction :

- **Ligne 3 :** `int anneeNaissance;` - Java réserve un espace mémoire pour stocker un nombre entier et l'associe au nom `anneeNaissance`. Cette variable n'a pas encore de valeur définie.
- **Ligne 4 :** `int anneeActuelle;` - Java réserve un second espace mémoire pour un entier nommé `anneeActuelle`, également non initialisé.
- **Ligne 5 :** `int age;` - Java réserve un troisième espace mémoire pour un entier nommé `age`, non initialisé.
- **Ligne 7 :** `System.out.println("En quelle année êtes-vous né(e) ?");` - Le message "En quelle année êtes-vous né(e) ?" s'affiche à l'écran et le curseur passe à la ligne suivante.
- **Ligne 8 :** `anneeNaissance = Terminal.lireInt();` - Le programme s'arrête et attend une saisie clavier. Un curseur clignotant apparaît. L'utilisateur tape "1995" puis appuie sur Entrée. La méthode `Terminal.lireInt()` convertit ce texte en nombre entier 1995 et l'affecte à la variable `anneeNaissance`. La variable contient maintenant la valeur 1995.
- **Ligne 10 :** `System.out.println("En quelle année sommes-nous ?");` - Le message "En quelle année sommes-nous ?" s'affiche à l'écran.
- **Ligne 11 :** `anneeActuelle = Terminal.lireInt();` - Le programme attend une nouvelle saisie. L'utilisateur tape "2025" et appuie sur Entrée. La valeur 2025 est stockée dans `anneeActuelle`.
- **Ligne 13 :** `age = anneeActuelle - anneeNaissance;` - Java évalue l'expression à droite du signe égal : il lit la valeur de `anneeActuelle` (2025), lit la valeur de `anneeNaissance` (1995), effectue la soustraction 2025 - 1995 = 30, puis stocke ce résultat dans la variable `age`.
- **Ligne 15 :** `System.out.println("Vous avez " + age + " ans.");` - Java construit la chaîne de caractères en concatenant "Vous avez " + la valeur de `age` (30) + " ans.", ce qui donne "Vous avez 30 ans.". Ce message s'affiche à l'écran.
- **Fin du programme :** Java atteint la fin de la méthode `main`, le programme se termine. Les variables `anneeNaissance`, `anneeActuelle` et `age` contiennent respectivement 1995, 2025 et 30 au moment de la fin du programme. À la fin du programme, la mémoire utilisée pour l'exécution du programme est effacée et rendue au système d'exploitation qui peut l'utiliser pour autre chose (par exemple pour la donner à l'exécution d'un autre programme). Les données qui étaient en mémoire sont définitivement perdues.

À tout moment de cette exécution, on peut connaître l'état exact de la mémoire : quelles variables existent et quelle valeur elles contiennent. Cette prévisibilité est fondamentale en programmation.

10.3 Points clés à retenir

1. **Séquentialité :** Chaque ligne s'exécute exactement une fois, dans l'ordre
2. **État de la mémoire :** Les variables gardent leur valeur jusqu'à ce qu'on les modifie par une affectation
3. **Interaction utilisateur :** Le programme peut s'arrêter pour attendre une saisie
4. **Calculs immédiats :** Les expressions sont évaluées instantanément lors de l'affectation

Cette exécution séquentielle est la base de tous les programmes. Dans les chapitres suivants, nous verrons comment modifier ce flux avec les conditions et les boucles.

11 Récapitulatif

Dans ce premier chapitre, nous avons vu :

— La structure générale d'un programme Java
— Les concepts de compilation et d'exécution
— Les variables et les types de données
— Les opérations arithmétiques de base
— Les entrées/sorties simples au moyen de méthodes
— La gestion des erreurs de compilation
— La notion de séquence d'instruction

Ces notions constituent les fondations sur lesquelles nous construirons les concepts plus avancés dans les chapitres suivants.

Ce chapitre a été rédigé avec l'assistance d'une intelligence artificielle, Claude 4 Sonnet de la société Anthropic