Chapitre 1

Introduction : notions de base

Le cours NFA032 est consacré à l'apprentissage des concepts de la programmation objet tels qu'ils sont mis en oeuvre en Java. Cet apprentissage suppose connus un certain nombre de notions de base étudiées dans le cours NFA031. Le présent chapitre est consacré à un bref rappel de ces notions.

1.1 Structure d'un programme Java

La structure minimale d'un programme Java consiste en une classe comportant une méthode `main`. Cette méthode contient une suite d'instructions.

1.2 Variables, types de base et types objets

Une variable associe un nom à un espace mémoire destiné à recevoir une valeur d'un type donné. Cette valeur peut varier au fil de l'exécution du programme et elle peut être inconnue au moment où l'on écrit le programme. On la désigne par le nom de la variable.

Une variable est déclarée avec son type suivi de son nom et éventuellement sa valeur initiale.

```
int nombre; // type int, nom nombre
int ligne = 5; // type int, nom ligne, valeur initiale 5
```

Les types utilisables pour déclarer une variables sont les types de base, les types tableaux (cf. ci-dessous) et les types objets.

Les types de base :

- int : nombres entiers
- double : nombres à virgule (approximation finie des nombres réels)
- char : caractères
- boolean : les valeurs de vérité true et false
- d'autres types entiers et nombre à virgule (non utilisés dans ce cours) : byte, long, float.

Les types objets comportent notamment les chaînes de caractère (type String).

On donne une nouvelle valeur à une variable au moyen d'une instruction d'affectation. La nouvelle valeur remplace ce qui était dans la variable et qui est alors perdu.

```
nombre = 17;
nombre = nombre + 1;
```

Dans ce cours, nous utilisons une classe spécifique pour réaliser les entrées-sorties écran/clavier, avec des méthodes pour les types int, double, char, boolean et String. Pour lire une valeur au clavier, on utilise les méthodes `Terminal.lireInt`, `Terminal.lireDouble`, etc. Pour écrire une valeur à l'écran, on utilise les méthodes `Terminal.ecrireInt` et `Terminal.ecrireIntln`. La première écrit simplement la valeur et la seconde écrit la valeur puis passe à la ligne. La valeur est donnée en paramètre, entre parenthèses. Par exemple : `Terminal.ecrireInt(147)`.

Exemples de déclarations, d'affectations et d'entrées-sorties pour les différents types :

```
public class Decls{
    public static void main(String[] vargs){
        // declarations
        int entier;
        double avirgule;
        char carac;
        boolean b;
        String ch;
        // affectation de valeurs
        entier = 17;
        avirgule = 14.369;
        carac = 'u';
        b = true;
        ch = "Bonjour";
        // saisie au clavier
        entier = Terminal.lireInt();
        avirgule = Terminal.lireDouble();
        carac = Terminal.lireChar();
        b = Terminal.lireBoolean();
        ch = Terminal.lireString();
        // affectation d'expressions avec opérateurs
        entier = 4*entier + 12;
        avirgule = avirgule / 12.36;
        b = b || true; // ou logique
        ch = "salut " + "les copains"; // concatenation de chaines
        // affichage à l'écran
        Terminal.ecrireIntln(entier);
        Terminal.ecrireDoubleln(avirgule -1.1);
        Terminal.ecrireCharln(carac);
        Terminal.ecrireBooleanln(b);
        Terminal.ecrireStringln(ch + " !");
    }
}
```

1.3 If, boucles et booléens

L'instruction conditionnelle (if) et les boucles (for, while) permettent de faire varier l'exécution du programme en fonction de la valeur d'une condition. Une condition consiste en un calcul (expression) donnant un résultat booléen (true ou false).

Un exemple d'expression booléenne consiste en une combinaison de valeurs booléennes au moyen de connecteurs logiques :

- `et`, noté `&&` en java

– `ou`, noté `||` en java
– `non`, noté `!` en java

Un autre exemple de calcul booléen consiste en une comparaison entre valeurs d'autres types :

– test d'égalité, noté == en java
– test de différence, noté != en java
– comparaisons d'ordre : <, <, <= et >=

Voici quelques expressions booléennes. Ces expressions peuvent apparaître en tant que condition dans un if ou une boucle, mais elle peut également être affectée à une variable de type `boolean` comme c'est le cas ici.

```
public class Bools{
    public static void main(String[] args){
        boolean b1 = true;
        boolean b2;
        Terminal.ecrireString("Entrez la valeur de b2 (true ou false): ");
        b2 = Terminal.lireBoolean();
        int x = 3;
        b1 = b1 && (b1 || b2);
        b1 = x<10;
        b1 = (x == 10); // test si X est egal a 10
        b2 = (x != 10) && b1;
        b1 = ('a' < 'Z') || (1.2*38 == 57.6);
        Terminal.ecrireBooleanln(b1);
        Terminal.ecrireBooleanln(b2);
    }
}
```

L'instruction if a la structure suivante :

```
if (condition){
   instruction-1;
   instruction-1;
   ...
   instruction-n;
}else{
   instruction-n+1;
   ...
   instruction-m;
}
```

Si la condition est vraie (le résultat du calcul vaut `true`), alors les instructions `instruction-1` à `instruction-n` sont exécutées, si elle est fausse, ce sont les instructions `instruction-n+1` à `instruction-m` qui sont exécutées.

La structure d'une boucle `while` est la suivante :

```
while(condition){
   instruction-1;
   instruction-1;
   ...
   instruction-n;
}
```

Si la condition est vraie, alors les instructions `instruction-1` à `instruction-n` sont exécutées puis la condition est recalculée, si elle est encore vraie, alors les instructions `instruction-1` à `instruction-n` sont exécutées une seconde fois, etc. Si la condition est fausse, la boucle est terminée, le programme passe à l'instruction suivante. La condition peut être fausse la première fois qu'on la calcule : dans ce cas, les instructions ne sont pas exécutées du tout. Si elle est fausse la deuxième fois qu'on la calcule, alors les instructions ont été exécutées une fois. Si la condition est fausse la n^{ime} fois qu'on la calcule, alors les instructions ont été exécutées $n-1$ fois.

Il se peut que la condition ne devienne jamais fausse et alors le programme exécute les instructions sans jamais s'arrêter. Cela peut être une erreur du programme ou le comportement souhaité par le programmeur.

Voici un exemple de boucle while.

```
public class ExNote{
    public static void main(String[] args){
        int note;
        Terminal.ecrireString("Entrez␣votre␣note:␣");
        note = Terminal.lireInt();
        while (note<0 || note>20){
            Terminal.ecrireStringln("Vous␣vous␣êtes␣trompé");
            Terminal.ecrireStringln("La␣note␣doit␣être␣comprise␣entre␣0␣et␣20");
            Terminal.ecrireStringln("Veuillez␣recommencer");
            Terminal.ecrireString("Entrez␣votre␣note:␣");
            note = Terminal.lireInt();
        }
        Terminal.ecrireStringln("Votre␣note␣est␣" + note);
    }
}
```

La deuxième boucle est la boucle `for` que l'on utilise lorsque l'on connaît le nombre de tours de boucle à effectuer au moment où l'on commence à exécuter cette boucle.

La boucle for a la structure suivante :

```
for(initialisation;condition;incrémentation){
   instruction-1;
   ...
   instruction-n;
}
```

L'initialisation est une instruction exécutée avant de débuter la boucle. La condition est calculée en début de boucle. Si elle vaut `true`, alors les instructions `instruction-1` à `instruction-n` sont exécutées, puis l'instruction `incrémentation` est exécutée, puis la condition recalculée, etc. Si elle vaut `false`, la boucle est terminée.

Cette boucle peut se traduire dans le programme suivant qui comporte une boucle `while` :

```
initialisation;
while(condition){
   instruction-1;
   ...
   instruction-n;
   incrémentation;
}
```

Voici un exemple de boucle for qui affiche 10 lignes à l'écran.

```
public class ExFor{
    public static void main(String[] args){
        for (int i=1; i<10; i=i+1){
            Terminal.ecrireStringln(i+ "␣kilomètre(s)␣à␣pied,␣ça␣use,␣ça␣use");
            Terminal.ecrireStringln("ça␣use␣les␣souliers");
        }
    }
}
```

Les instructions de la boucle sont exécutées 9 fois, la condition est calculée 10 fois. Voici ce qui s'affiche à l'écran lors de l'exécution :

```
1 kilomètre(s) à pied, ça use, ça use
ça use les souliers
2 kilomètre(s) à pied, ça use, ça use
ça use les souliers
3 kilomètre(s) à pied, ça use, ça use
ça use les souliers
4 kilomètre(s) à pied, ça use, ça use
ça use les souliers
5 kilomètre(s) à pied, ça use, ça use
ça use les souliers
6 kilomètre(s) à pied, ça use, ça use
ça use les souliers
7 kilomètre(s) à pied, ça use, ça use
ça use les souliers
8 kilomètre(s) à pied, ça use, ça use
ça use les souliers
9 kilomètre(s) à pied, ça use, ça use
ça use les souliers
```

Il existe une troisième boucle en java, la boucle `do...while`. Elle a la structure suivante :

```
do{
   instruction-1;
   ...
   instruction-n;
}while(condition);
```

C'est une variante de la boucle `while` où la condition est calculée à la fin de la boucle, après avoir exécuté les instructions 1 à n. Cela signifie que ces instructions sont toujours calculée au moins une fois.

1.4 Tableaux

Un tableau est une structure de données qui contient plusieurs valeurs du même type. Les différentes valeurs sont numérotées à partir du numéro 0. S'il y a 10 valeurs, elles sont numérotées de 0 à 9.

0	1	2	3	4	5	6	7	8	9
127	0	2	55	-30	0	125	8	1	45

Le type d'un tableau est noté avec le type des élément suivi de crochets : `int[]` pour un tableau contenant des entiers, `String[]` pour un tableau contenant des chaînes de caractères.

Avant d'utiliser un tableau, il faut le créer avec l'instruction `new`, en précisant la taille du tableau, c'est à dire le nombre de valeurs qu'il contient. Ce nombre ne changera jamais au cours du programme. `new int[12]` crée un tableau de 12 cases numérotées de 0 à 11, chaque case contenant un entier. A la création du tableau, chaque case contient 0.

Chaque case du tableau est utilisable comme une variable de type `int` à laquelle on peut affecter une valeur et dont on peut utiliser la valeur dans un calcul.

```
int[] tab;
tab = new int[12];
tab[0]=145;
tab[1]=37;
Terminal.ecrireIntln(tab[1]*5+1);
```

Les tableaux sont souvent parcourus au moyen d'une boucle `for`.

Voici par exemple un programme qui met dans les cases d'un tableaux les premières puissances de 2 :

```
int[] tab;
tab = new int[8];
tab[0]=1;
for (int i=1;i<8;i++){
    tab[i]=tab[i-1]*2;
}
// affichage du tableau
for (int i=0; i<8; i++){
    Terminal.ecrireIntln(tab[i]);
}
```

On peut obtenir la taille d'un tableau `tab` au moyen de l'expression `tab.length`.

Un tableau à deux dimensions peut être vu comme un tableau ayant des lignes et des colonnes. Chaque case est caractérisée par un numéro de ligne et un numéro de colonne.

Par exemple, pour représenter les heures de présence d'une personne à son poste de travail sur une semaine, on peut utiliser un tableau de booléens à 7 colonnes (les 7 jours) et 24 lignes (les 24 heures de la journée). Le booléen sera true si la personne est là à l'heure et au jour correspondant à la case et false si elle n'est pas là.

```
boolean[][] pres = new boolean[7][24];
// les trois premiers jours, la personne est la de 4H00 à 12H00
for (int jour=0; jour<3; jour++){
  for (int heure=4; heure<12; heure++){
    pres[jour][heure]=true;
  }
}
// enseuite, deux jours de repos et deux jours du soir
for (int jour=5; jour<7; jour++){
  for (int heure=16; heure<24; heure++){
    pres[jour][heure]=true;
```

```
    }
  }
  // affichage
  for (int heure=0;heure<24; heure++){
    for (int jour=0; jour<7; jour++){
      if (pres[jour][heure]){
        Terminal.ecrireString("XXX");
      }else{
        Terminal.ecrireString("␣␣␣");
      }
    }
    Terminal.ecrireStringln("");
  }
```

1.5 Sous-programmes

Un sous-programme est un morceau d'un programme distingué soit parce qu'il est utilisé à de multiple reprises dans le programme, soit parce qu'il réalise une tâche spécifique qu'il est plus clair de considérer comme une unité spéarée.

Un sous-programme a des données en entrée qui viennent via des paramètres. Il peut renvoyer une valeur à la fin de son exécution.

En Java, un sous-programme est une méthode. On l'écrit avec un entête et un corps. L'entête comprend dans l'ordre : le type de la valeur renvoyée (ou `void` si aucune valeur n'est renvoyée), le nom, la liste des paramètres entre parenthèses. Le corps est un bloc : un suite d'instruction entre accolades. Une instruction spécifique sert à renvoyer le résultat de la méthode, l'instruction `return`.

```
double moyenne(double[] tab){
  double somme = 0.0;
  for (int i=0; i<tab.length; i++){
    somme = somme + tab[i];
  }
  return somme/tab.length;
}
```

1.6 Compilation et exécution d'un programme Java

Un programme java se tape dans un fichier texte ayant l'extension .java. Pour créer ce fichier, il faut un éditeur de texte (par exemple jedit) ou un environnement de programmation intégré qui comprend un éditeur (par exemple Eclipse). Ce programme a une forme compréhensible pour un être humain, mais il n'est pas directement exécutable sur machine, il faut le traduire dans un autre format.

C'est le compilateur qui assure cette traduction. Dans un terminal, on utilise la commande `javac`. Dans un environnement intégrer, on actionne un bouton ou un choix de menu déroulant. La compilation peut échouer s'il y a des erreurs. Il faut alors corriger le progamme dans l'éditeur et recommencer la compilation.

S'il n'y a pas d'erreur, le compilateur crée un certain nombre de fichiers avec l'extension .class qui contient le code machine. En java, ce code machine n'est pas directement exécutable par le processeur, il doit être exécuté par un interpréteur de code Java. C'est ce qu'on appelle le runtime java. Dans un

terminal, c'est la commande `java`. Dans un environnement intégré, c'est un bouton `run` ou un choix de menu déroulant.

Pour les exemples de ce cours, on utilise une classe appelée `Terminal` qui réalise les entrées-sorties écran-clavier. Cette classe est à télécharger sur le site web du cours (dans la séance 000 sous pleiad pour la formation à distance). Le but de cette classe est notamment de faciliter les lectures au clavier qui demandent sinon des incantations plus ou moins obscures. Pour compiler les exemples du cours (que vous pouvez copier-coller dans les fichiers pdf), vous devez copier le fichier `Terminal.java` dans le même dossier que le programme.