Aide-mémoire

Java

Vincent **Granet**
Jean-Pierre **Regourd**

Aide-mémoire
Java

5e édition

DUNOD

Photo de couverture : © by-studio — Fotolia

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-079038-8

Table des matières

Avant-propos

Cette cinquième édition de l'*Aide-mémoire de Java* intègre les récents enrichissements apportés au langage Java depuis la version 9 jusqu'à la version 11, en particulier l'inférence de type sur les variables locales et la notion de *module*. Ce nouveau concept réorganise entièrement l'API Java.

Une section sur la gestion des processus systèmes a également été ajoutée à l'ouvrage. La classe `Process` qui les décrit était présente dès les premières versions du langage, mais la version 9 du langage l'a considérablement enrichie.

Depuis ses origines en 1991, le langage à objets Java connaît un engouement qui ne cesse de croître et une riche communauté de concepteurs et d'utilisateurs s'est développée et consolidée autour de ce langage. Son modèle objet, simple mais néanmoins puissant, en fait aujourd'hui un des rares langages généralistes à être aussi bien enseignés dans les universités qu'utilisés dans le monde industriel.

Malgré sa forme réduite, ce livre est une présentation pédagogique et didactique qui couvre les aspects fondamentaux et appliqués du langage. Si l'ensemble des mécanismes de Java est détaillé, une présentation exhaustive des milliers de classes (plus de 4 000) de l'API (*Application Programming Interface*) est impossible. Notre choix, forcément partiel et partial, a été guidé par l'intérêt pratique des classes retenues.

Cet ouvrage a aussi été conçu pour offrir au lecteur une présentation condensée du langage Java. Il s'adresse tout aussi bien aux lecteurs ayant déjà une pratique de la programmation, et désireux d'aborder un nouveau type de langage, qu'aux programmeurs Java ressentant le besoin d'un ouvrage synthétique.

La première partie du livre, jusqu'au chapitre 7, expose les éléments et les mécanismes propres au langage. La seconde s'attache à présenter l'environnement de programmation du langage.

Les deux premiers chapitres sont consacrés aux constructions et aux objets de base de Java. Au travers d'un exemple *fil rouge*, les deux suivants décrivent en détail le modèle objet de Java.

La généricité, mécanisme de paramétrage des classes abondamment utilisé dans l'API, est l'objet du chapitre 5. Partie intégrante du langage, les exceptions et les processus légers (*threads*) sont abordés dans le chapitre 6. Enfin, le chapitre 7 présente les fonctions anonymes, qui a été la grande nouveauté de Java 8.

La seconde partie du livre présente l'API et les outils de développement. Le chapitre 8 donne une présentation détaillée des mécanismes d'entrée-sortie. L'accent a été mis sur la manipulation des fichiers. Le chapitre 9 décrit une sélection de classes de l'API. Tout d'abord, nous y traitons les structures de données, collections, tables et les objets *stream*, qui permettent, à l'aide de fonctions anonymes, le traitement efficace de grands flots de données. Puis, nous présentons l'accès et la gestion des processus systèmes, les classes pour la communication réseau, les environnements graphiques AWT et Swing. Le chapitre 10 présente une API particulièrement importante, JDBC. Nous y décrivons les éléments essentiels indispensables au maniement des bases de données en nous appuyant sur un exemple *suffisamment complet* pour initier de réels développements dans ce vaste domaine d'application. Le dernier chapitre est consacré aux outils de développement de la distribution officielle Java et à la notion de module.

L'ouvrage s'achève par un catalogue de sites web consacrés à Java, complété par une bibliographie thématique.

Le site de cet ouvrage propose les codes sources de certains exemples du livre. Il est accessible à l'adresse :

`users.polytech.unice.fr/~vg/index-aidememoire.html`

Enfin, nous souhaitons remercier toute l'équipe Dunod, en particulier Brice Martin pour sa relecture très minutieuse de cette cinquième édition et Jean-Luc Blanc pour son aide précieuse et ses conseils toujours avisés.

Sophia Antipolis, mai 2019.

1 Java, les bases

Comme l'indique son titre, ce premier chapitre présente les notions de base du langage Java. Après avoir donné, à l'aide d'un exemple, une première description de son modèle objet, nous présenterons les éléments fondamentaux du langage : les types élémentaires, les expressions et les énoncés.

1.1 Une première application

Généralement, les ouvrages d'apprentissage des langages de programmation commencent par la présentation d'une application simple qui donne au lecteur une première impression sur le langage. Nous ne dérogerons pas à cette règle en présentant une première application qui met en évidence plusieurs notions fondamentales du langage à objets Java. Celles-ci sont brièvement décrites dans cette section mais seront, bien sûr, plus amplement détaillées dans les chapitres suivants.

Cette première application, donnée dans la figure 1.1, affiche la date du jour en toutes lettres, comme par exemple *jeudi 4 juillet 2019*. Elle nous permettra de mettre en évidence les notions de commentaire, de classe, de méthode et de variable.

Les commentaires

Les commentaires jouent un rôle essentiel dans la compréhension des fonctionnalités d'une application, mais aussi, et surtout, dans la vérification de sa validité.

```
/**
 * Cette application écrit la date du jour
 * en toutes lettres (e.g. jeudi 4 juillet 2019)
 */
import java.util.Date;
import java.text.SimpleDateFormat;

class DateDuJour {
    public static void main(String[] args) {
        // créer la date du jour
        Date dateDuJour;
        dateDuJour = new Date();
        // créer son modèle de présentation
        SimpleDateFormat fd;
        fd = new SimpleDateFormat("EEEE dd MMMM yyyy");
        // afficher la date du jour selon ce modèle
        System.out.println(fd.format(dateDuJour));
    }
}
```

Figure 1.1 Application DateDuJour

Le langage propose trois formes de commentaires différenciées par leurs parenthéseurs :

1. `/**` *suite quelconque de caractères* `*/`
2. `/*` *suite quelconque de caractères* `*/`
3. `//` *suite quelconque de caractères jusqu'à la fin de la ligne*

La première sert à la documentation des classes et d'autres éléments des applications, et peut être exploitée par le générateur `javadoc`[1]. La documentation produite au format HTML (*HyperText Markup Language*) offre une présentation régulière et lisible à l'aide de n'importe quel navigateur. Les commentaires de documentation peuvent apparaître sur plusieurs lignes et contenir des directives spéciales de mise en pages destinées au générateur `javadoc`.

1. Cf. chapitre 11.

La deuxième et la troisième forme de commentaire sont généralement utilisées pour la description de l'implémentation du code (antécédent, conséquent, invariant, etc.). Notez que, dans la troisième forme, le commentaire s'achève à la fin de la ligne, alors que dans la deuxième il peut apparaître sur plusieurs lignes, comme pour les commentaires de documentation.

Les classes

Le langage à objets [2] Java est un langage de classes. Cela veut dire que la conception d'une application écrite dans ce langage suit un modèle de structuration autour d'objet décrits par des classes qui en précisent les propriétés, variables et méthodes.

Les variables désignent des données qui sont des valeurs de tout type. Les méthodes sont des actions, c'est-à-dire des procédures ou des fonctions. Par défaut, les variables et les méthodes sont liées aux objets, mais si leur déclaration le précise, à l'aide du mot-clé **`static`**, elles sont alors liées à la classe [3].

D'un point de vue statique, celui du compilateur, une application Java est une collection de classes qui décrit les objets qui seront manipulés à l'exécution. Notre première application ne comporte qu'une seule classe, nommée `DateDuJour`, écrite dans un fichier qui porte son nom et suffixé par `.java`, c'est-à-dire `DateDuJour.java`. Par convention [4], la lettre initiale de chacun des mots qui forment le nom de la classe est en majuscule.

Toutefois, notre application utilise quatre autres classes, `String`, `System`, `Date` et `SimpleDateFormat`, définies par ailleurs et mises à la disposition du programmeur par l'environnement de programmation Java. Ces classes, avec plusieurs centaines d'autres, forment ce que l'on appelle l'API (*Application Programming Interface*) Java [5].

2. Avec sa version 8, Java a introduit le paradigme fonctionnel (cf. le chapitre 7), toutefois la langage reste principalement un langage à objets.

3. Cf. chapitre 3.

4. Pour les conventions d'écriture de code Java, voir l'URL :
`https://www.oracle.com/technetwork/java/codeconventions-135099.html`

5. Cf. chapitre 9.

Les classes unies par une même sémantique sont regroupées dans des *paquetages* [6], eux-même répartis, depuis la version 9 du langage dans différents *modules* [7]. L'API de Java est formée de nombreux paquetages comme par exemple `java.lang`, `java.io` ou encore `java.util` [8]. Ces trois paquetages appartiennent au module `java.base`.

Sauf pour `java.lang`, le nom d'une classe doit être qualifié par le nom du paquetage auquel elle appartient. Toutefois, la directive d'importation, **`import`**, permet un accès non qualifié aux classes des paquetages. Ainsi dans notre application, la directive d'importation de la classe `Date` appartenant au paquetage `java.util` permet l'utilisation du nom `Date` dans la classe `DateDuJour` sans avoir à le préfixer par `java.util`. Il en va de même pour la classe `SimpleDateFormat` du paquetage `java.text`.

La méthode main

L'exécution d'une application Java commence implicitement par celle d'une méthode statique `main` dont l'en-tête est *toujours* le même. En particulier, le paramètre formel `args` est un tableau de chaînes de caractères qui donne accès aux paramètres du programme lors de son exécution. Même s'il reste inutilisé par la suite (comme dans notre application), ce paramètre *doit* apparaître dans l'en-tête.

Le corps d'une méthode peut contenir des déclarations de variables locales et des instructions. Java est un langage *typé*, ce qui veut dire que les déclarations des noms utilisés dans l'application doivent spécifier les types des objets qu'ils désigneront [9]. En Java, les classes sont assimilées à des types.

La méthode `main` de notre exemple déclare deux variables locales. La première, `dateDuJour`, permet de désigner des objets de la classe `Date`.

6. Cf. chapitre 3.

7. Cf. le chapitre 11 page 297.

8. Le paquetage `java.lang` contient les classes de base du langage, `java.io` des classes pour les entrées-sorties, et `java.util` des classes utilitaires nécessaires à bien des applications.

9. Toutefois, la version 10 du langage permet de déclarer des variables sans indiquer leur type. Il est déterminé par *inférence de type*.

De tels objets représentent un temps écoulé en millisecondes depuis une date de référence [10]. La seconde variable, `fd`, permet, quant à elle, de désigner des objets de la classe `SimpleDateFormat`, qui définit des modèles de présentation des dates.

Pour créer, on dit *instancier*, les objets que pourront désigner ces deux variables, on utilise l'opérateur **`new`**, qui applique un constructeur. Un constructeur est une sorte de méthode qui porte *toujours* le nom de la classe dans laquelle il est défini, et dont le rôle est d'initialiser les variables de l'objet au moment de sa création.

Le constructeur `Date()` initialise un objet `Date` à la date du jour, c'est-à-dire le nombre de millisecondes entre la date de référence et le moment de la création de l'objet. L'opérateur d'affectation, le signe =, associe à la variable `dateDuJour` une référence à l'objet créé. La valeur de la variable `dateDuJour` *n'est pas* l'objet créé mais une sorte de pointeur sur cet objet, appelé référence. La constante symbolique **`null`** est une référence particulière pour signifier l'absence de référence à un objet particulier [11].

De même, la seconde affectation associe à la variable `fd` une référence à un objet `SimpleDateFormat`. Le constructeur utilisé possède comme paramètre une chaîne de caractères (définie par la classe `String`) dont la valeur décrit un modèle de présentation des dates selon la forme : jour (EEEE), numéro du jour (dd), mois (MMMM) et année (yyyy).

Pour accéder aux variables ou aux méthodes d'un objet, on utilise une *notation pointée* qui suit la syntaxe : nom de l'objet, suivi d'un point, suivi du nom de la variable ou de la méthode avec ses paramètres effectifs.

La classe `SimpleDateFormat` possède une méthode `format`. Elle renvoie une chaîne de caractères qui est la représentation d'un objet `Date` passé en paramètre suivant le modèle de présentation des dates de l'objet courant. Appliquée à l'objet désigné par `fd`, cette méthode renvoie la date du jour selon son modèle (`fd.format(dateDuJour)`).

Enfin, l'écriture sur la sortie standard de la chaîne de caractères qui représente la date du jour est faite par l'application de la méthode `println` sur l'objet statique `out` de la classe `System`.

10. 1er janvier 1970, GMT 00:00:00.

11. C'est l'équivalent des constantes `NULL` et `nil` des langages C et Lisp.