Un best-seller qui a déjà conquis plus de 50 000 étudiants !

Anne Tasso

Le livre de JAVA premier langage

Avec 109 exercices corrigés

11e édition

Corrigé du projet et des exercices
Code source des exemples du livre

EYROLLES

Le livre de JAVA premier langage

11e édition

Apprendre Java en douceur

Vous avez décidé de vous initier à la programmation et souhaitez opter pour un langage largement utilisé dans le monde professionnel ? Java se révèle un choix idéal comme vous le constaterez dans ce livre conçu pour les vrais débutants en programmation.

Vous apprendrez d'abord, à travers des exemples simples en Java, à maîtriser les notions communes à tous les langages : variables, types de données, boucles et instructions conditionnelles, etc. Vous franchirez un nouveau pas en découvrant par la pratique les concepts de la programmation orientée objet (classes, objets, héritage), puis le fonctionnement des librairies graphiques AWT et Swing (fenêtres, gestion de la souris, tracé de graphiques). Cet ouvrage vous expliquera aussi comment réaliser des applications Java dotées d'interfaces graphiques conviviales grâce au logiciel libre NetBeans. Enfin, vous vous initierez au développement d'applications avec l'interface Android Studio.

Chaque chapitre est accompagné de deux types de travaux pratiques : des exercices, dont le corrigé est fourni sur l'extension web du livre, et un projet développé au fil de l'ouvrage, qui vous montrera comment combiner toutes les techniques de programmation étudiées pour construire une véritable application Java.

À qui s'adresse ce livre ?

- Aux étudiants de 1er cycle universitaire (IUT, Deug...) ou d'écoles d'ingénieurs
- Aux vrais débutants en programmation : passionnés d'informatique et programmeurs autodidactes, concepteurs de sites Web souhaitant aller au-delà de HTML et JavaScript, etc.
- Aux enseignants et formateurs recherchant une méthode pédagogique et un support de cours pour enseigner Java à des débutants

Sur le site www.annetasso.fr/java

- Consultez les corrigés du projet et des exercices
- Téléchargez le code source de tous les exemples du livre
- Dialoguez avec l'auteur

Maître de conférences à l'université Paris-Est Marne-la-Vallée, **Anne Tasso** enseigne le langage Java en formation initiale et continue, au sein du département MMI (Métiers du Multimédia et de l'Internet) de l'IUT de Marne-la-Vallée. Son public universitaire est essentiellement constitué de débutants en programmation, ce qui lui a permis d'élaborer une méthode pédagogique structurée et imagée. Son objectif est d'expliquer, avec des mots simples, les techniques de programmation jusqu'à un niveau avancé.

Sommaire

www.editions-eyrolles.com

Le livre de JAVA premier langage

Avec 109 exercices corrigés

CHEZ LE MÊME ÉDITEUR

Autres ouvrages sur Java

C. Delannoy. – **Programmer en Java (9e édition)**.
N°14007, 2014, 940 pages.

C. Delannoy. – **Exercices en Java (4e édition)**.
N°14009, 2014, 360 pages.

C. Delannoy. – **Programmer en Java (6e édition)**. *Java 5 et 6*.
N°13443, 2012, 788 pages (format semi-poche).

J.-B. Boichat. – **Apprendre Java et C++ en parallèle (4e édition)**.
N°12403, 2008, 600 pages + CD-Rom.

A. Patricio. – **Java Persistence et Hibernate.**
N°12259, 2008, 364 pages.

E. Puybaret. **– Bien programmer en Java 7.**
N°12794, 2012, 428 pages.

R. Fleury. **– Les Cahiers du programmeur Java/XML.**
N°11316, 2004, 218 pages.

P. Haggar. – **Mieux programmer en Java**. *68 astuces pour optimiser son code*.
N°9171, 2000, 256 pages.

J.-P. Retaillé. – **Refactoring des applications Java/J2EE**.
N°11577, 2005, 390 pages.

Autres ouvrages

C. Delannoy. – **Programmer en Fortran**. *Fortran 90 et ses évolutions - Fortran 95, 2003 et 2008*.
N°14020, 2015, 454 pages.

C. Delannoy. – **Le guide complet du langage C**.
N°14012, 2014, 844 pages.

C. Delannoy. – **S'initier à la programmation et à l'orienté objet**. *Avec des exemples en C, C++, C#, Python, Java et PHP.*
N°14067, 2014, 382 pages.

G. Dowek *et al*. – **Informatique et sciences du numérique**. *Manuel de spécialité ISN en terminale*.
N°13676, 2013, 354 pages.

G. Dowek *et al*. – **Informatique pour tous en classes préparatoires aux grandes écoles**. *Manuel d'algorithmique et programmation structurée avec Python*.
N°13700, 2013, 408 pages.

Anne Tasso

Le livre de JAVA premier langage

Avec 109 exercices corrigés

11e édition

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Avec cette onzième édition, je tiens à remercier tous mes nombreux lecteurs pour leurs félicitations qui me vont droit au cœur et leurs remarques toujours constructives.

Je remercie également tous mes étudiants, qui par leurs interrogations, leurs retours et leur curiosité m'ont permis d'écrire ce livre avec le souhait d'apporter des explications claires et précises.

Et enfin, un merci tout particulier à Antoine Derouin qui m'offre le temps de faire naître chaque livre avec beaucoup d'esprit, d'attention et de patience.

Table des matières

Partie II
Initiation à la programmation orientée objet

Avant-propos

Organisation de l'ouvrage

Ce livre est tout particulièrement destiné aux débutants qui souhaitent aborder l'apprentissage de la programmation en utilisant le langage Java comme premier langage.

Les concepts fondamentaux de la programmation y sont présentés de façon évolutive, grâce à un découpage de l'ouvrage en trois parties, chacune couvrant un aspect différent des outils et techniques de programmation.

Le chapitre introductif, « Naissance d'un programme », constitue le préalable nécessaire à la bonne compréhension des parties suivantes. Il introduit aux mécanismes de construction d'un algorithme, compte tenu du fonctionnement interne de l'ordinateur, et explique les notions de langage informatique, de compilation et d'exécution à travers un exemple de programme écrit en Java.

La première partie concerne l'étude des « Outils et techniques de base » :

- Le chapitre 1, « Stocker une information », aborde la notion de variables et de types. Il présente comment stocker une donnée en mémoire, calculer des expressions mathématiques ou échanger deux valeurs, et montre comment le type d'une variable peut influencer le résultat d'un calcul.
- Le chapitre 2, « Communiquer une information », explique comment transmettre des valeurs à l'ordinateur par l'intermédiaire du clavier et montre comment l'ordinateur fournit des résultats en affichant des messages à l'écran.
- Le chapitre 3, « Faire des choix », examine comment tester des valeurs et prendre des décisions en fonction du résultat. Il traite de la comparaison de valeurs ainsi que de l'arborescence de choix. Avec en exemple, la nouvelle structure de test `switch` de la version 7 de Java.
- Le chapitre 4, « Faire des répétitions », est consacré à l'étude des outils de répétition et d'itération. Il aborde les notions d'incrémentation et d'accumulation de valeurs (compter et faire la somme d'une collection de valeurs).

La deuxième partie, « Initiation à la programmation orientée objet », introduit les concepts fondamentaux indispensables à la programmation objet.

- Le chapitre 5, « De l'algorithme paramétré à l'écriture de fonctions », montre l'intérêt de l'emploi de fonctions dans la programmation. Il examine les différentes étapes de leur création.
- Le chapitre 6, « Fonctions, notions avancées », décrit très précisément comment manipuler les fonctions et leurs paramètres. Il définit les termes de variable locale et de classe, et explique le passage de paramètres par valeur.

- Le chapitre 7, « Les classes et les objets », explique à partir de l'étude de la classe `String`, ce que sont les classes et les objets dans le langage Java. Il montre ensuite comment définir de nouvelles classes et construire des objets propres à l'application développée. Avec en exemple, une nouvelle façon de comparer des chaînes de caractères grâce à la nouvelle structure de test `switch` de la version 7 de Java.
- Le chapitre 8, « Les principes du concept d'objet », développe plus particulièrement comment les objets se communiquent l'information, en expliquant notamment le principe du passage de paramètres par référence. Il décrit ensuite les principes fondateurs de la notion d'objet, c'est-à-dire l'encapsulation des données (protection et contrôle des données, constructeur de classe) ainsi que l'héritage entre classes et la notion d'interfaces.

La troisième partie, « Outils et techniques orientés objet », donne tous les détails sur l'organisation, le traitement et l'exploitation intelligente des objets.

- Le chapitre 9, « Collectionner un nombre fixe d'objets », concerne l'organisation des données sous la forme d'un tableau de taille fixe.
- Le chapitre 10, « Collectionner un nombre indéterminé d'objets », présente les différents outils qui permettent d'organiser dynamiquement en mémoire les ensembles de données de même nature, notamment les nouvelles fonctionnalités de Java 8, à savoir les expressions lambda et les streams. Il est également consacré aux différentes techniques d'archivage et à la façon d'accéder aux informations stockées sous forme de fichiers.
- Le chapitre 11, « Dessiner des objets », couvre une grande partie des outils graphiques proposés par le langage Java (bibliothèques AWT et Swing). Il analyse le concept événement-action.
- Le chapitre 12, « Créer une interface graphique », expose dans un premier temps le fonctionnement de base de l'environnement de programmation NetBeans. Puis, à travers trois exemples très pratiques, il montre comment concevoir des applications munies d'interfaces graphiques conviviales.
- Le chapitre 13, « Développer une application Android », décrit comment créer votre toute première application Android, tout en expliquant la structure de base nécessaire au déploiement de cette application. Il présente ensuite le développement d'une application plus élaborée ainsi que sa mise à disposition sur un serveur dédié.

Ce livre contient également en annexe :

- un guide d'installation détaillé des outils nécessaires au développement des applications Java (Java, NetBeans), sous Linux, Mac OS X et sous Windows 2000, NT, XP et Vista ;
- toutes les explications nécessaires pour construire votre environnement de développement d'applications Java ou Android, que ce soit en mode commande ou en utilisant la plateforme ou Android Studio pour les applications sur smartphone ou tablette ;
- un index, qui vous aidera à retrouver une information sur le thème que vous recherchez (les mots-clés du langage, les exemples, les principes de fonctionnement, les classes et leurs méthodes, etc.).

Chaque chapitre se termine sur une série d'exercices offrant au lecteur la possibilité de mettre en pratique les notions qui viennent d'être étudiées. Un projet est également proposé au fil des chapitres afin de développer une application de gestion d'un compte bancaire. La mise en œuvre de cette application constitue un fil rouge qui permettra au lecteur de combiner toutes les techniques de programmation étudiées au fur et à mesure de l'ouvrage, afin de construire une véritable application Java.

Extension Web Les codes sources des exemples, des exercices et du projet sont téléchargeables depuis l'extension Web *www.annetasso.fr/Java* en cliquant sur le lien Sources.

Introduction

Naissance d'un programme

Aujourd'hui, l'informatique en général et l'ordinateur en particulier sont d'un usage courant. Grâce à Internet, l'informatique donne accès à une information mondiale. Elle donne aussi la possibilité de traiter cette information pour analyser, gérer, prévoir ou concevoir des événements dans des domaines aussi divers que la météo, la médecine, l'économie, la bureautique, etc.

Cette communication et ces traitements ne sont possibles qu'au travers de l'outil informatique. Cependant, toutes ces facultés résultent davantage de l'application d'un programme résidant sur l'ordinateur que de l'ordinateur lui-même. En fait, le programme est à l'ordinateur ce que l'esprit est à l'être humain.

Créer une application, c'est apporter de l'esprit à l'ordinateur. Pour que cet esprit donne sa pleine mesure, il est certes nécessaire de bien connaître le langage des ordinateurs, mais, surtout, il est indispensable de savoir programmer. La programmation est l'art d'analyser un problème afin d'en extraire la marche à suivre, l'algorithme susceptible de résoudre ce problème.

C'est pourquoi ce chapitre commence par aborder la notion d'algorithme. À partir d'un exemple tiré de la vie courante, nous déterminons les étapes essentielles à l'élaboration d'un programme (voir section « Construire un algorithme »). À la section suivante, « Qu'est-ce qu'un ordinateur ? », nous examinons le rôle et le fonctionnement de l'ordinateur dans le passage de l'algorithme au programme. Nous étudions ensuite, à travers un exemple simple, comment écrire un programme en Java et l'exécuter (voir section « Un premier programme en Java, ou comment parler à un ordinateur »). Enfin, nous décrivons, à la section « Le projet : Gestion d'un compte bancaire », le cahier des charges de l'application projet que le lecteur assidu peut réaliser en suivant les exercices décrits à la fin de chaque chapitre.

Construire un algorithme

Un ordinateur muni de l'application adéquate traite une information. Il sait calculer, compter, trier ou rechercher l'information, dans la mesure où un programmeur lui a donné les ordres à exécuter et la marche à suivre pour arriver au résultat.

Cette marche à suivre s'appelle un algorithme.

Déterminer l'algorithme, c'est trouver un cheminement de tâches à fournir à l'ordinateur pour qu'il les exécute. Voyons comment s'y prendre pour construire cette marche à suivre.

Ne faire qu'une seule chose à la fois

Avant de réaliser une application concrète, telle que celle proposée en projet dans cet ouvrage, nécessairement complexe par la diversité des tâches qu'elle doit réaliser, simplifions-nous la tâche en ne cherchant à résoudre qu'un problème à la fois.

Considérons que créer une application, c'est décomposer cette dernière en plusieurs sous-applications qui, à leur tour, se décomposent en micro-applications, jusqu'à descendre au niveau le plus élémentaire. Cette démarche est appelée analyse descendante. Elle est le principe de base de toute construction algorithmique.

Pour bien comprendre cette démarche, penchons-nous sur un problème réel et simple à résoudre : comment faire un café chaud non sucré ?

Exemple : l'algorithme du café chaud

Construire un algorithme, c'est avant tout analyser l'énoncé du problème afin de définir l'ensemble des objets à manipuler pour obtenir un résultat.

Définition des objets manipulés

Analysons l'énoncé suivant :

```
Comment faire un café chaud non sucré ?
```

Chaque mot a son importance, et « non sucré » est aussi important que « café » ou « chaud ». Le terme « non sucré » implique qu'il n'est pas nécessaire de prendre du sucre ni une petite cuillère.

Notons que tous les ingrédients et ustensiles nécessaires ne sont pas cités dans l'énoncé. En particulier, nous ne savons pas si nous disposons d'une cafetière électrique ou non. Pour résoudre notre problème, nous devons prendre certaines décisions, et ces dernières vont avoir une influence sur l'allure générale de notre algorithme.

Supposons que, pour réaliser notre café, nous soyons en possession des ustensiles et ingrédients suivants :

```
café moulu
filtre
eau
pichet
cafetière électrique
tasse
électricité
table
```

En fixant la liste des ingrédients et des ustensiles, nous définissons un environnement, une base de travail. Nous sommes ainsi en mesure d'établir une liste de toutes les actions à mener pour résoudre le problème et de construire la marche à suivre permettant d'obtenir un café.

Liste des opérations

```
Verser l'eau dans la cafetière, le café dans la tasse, le café dans
le filtre.
Remplir le pichet d'eau.
Prendre du café moulu, une tasse, de l'eau, une cafetière électrique,
un filtre, le pichet de la cafetière.
Brancher, allumer ou éteindre la cafetière électrique.
Attendre que le café remplisse le pichet.
Poser la tasse, la cafetière sur la table, le filtre dans la
cafetière, le pichet dans la cafetière.
```

Cette énumération est une description de toutes les actions nécessaires à la réalisation d'un café chaud.

Chaque action est un fragment du problème donné et ne peut plus être découpée. Chaque action est élémentaire par rapport à l'environnement que nous nous sommes donné.

En définissant l'ensemble des actions possibles, nous créons un langage minimal qui nous permet de réaliser le café. Ce langage est composé de verbes (`Prendre`, `Poser`, `Verser`, `Faire`, `Attendre`, etc.) et d'objets (`Café moulu`, `Eau`, `Filtre`, `Tasse`, etc.).

La taille du langage, c'est-à-dire le nombre de mots qu'il renferme, est déterminée par l'environnement. Pour cet exemple, nous avons, en précisant les hypothèses, volontairement choisi un environnement restreint. Nous aurions pu décrire des tâches comme « prendre un contrat EDF » ou « planter une graine de café », mais elles ne sont pas utiles à notre objectif pédagogique.

Question Quelle serait la liste des opérations supplémentaires si l'on décidait de faire un café sucré ?

Réponse Les opérations seraient :

```
Prendre du sucre, une petite cuillère.
Poser le sucre dans la tasse, la cuillère dans la tasse.
```

Remarque Telle que nous l'avons décrite, la liste des opérations ne nous permet pas encore de faire un café chaud. En suivant cette liste, tout y est, mais dans le désordre. Pour réaliser ce fameux café, nous devons ordonner cette liste.

Ordonner la liste des opérations

```
1.  Prendre une cafetière électrique.
2.  Poser la cafetière sur la table.
3.  Prendre un filtre.
4.  Poser le filtre dans la cafetière.
5.  Prendre du café moulu.
6. Verser le café moulu dans le filtre.
```

```
 7. Prendre le pichet de la cafetière.
 8. Remplir le pichet d'eau.
 9. Verser l'eau dans la cafetière.
10. Poser le pichet dans la cafetière.
11. Brancher la cafetière.
12. Allumer la cafetière.
13. Attendre que le café remplisse le pichet.
14. Prendre une tasse.
15. Poser la tasse sur la table.
16. Éteindre la cafetière.
17. Prendre le pichet de la cafetière.
18. Verser le café dans la tasse.
```

L'exécution de l'ensemble ordonné de ces tâches nous permet maintenant d'obtenir du café chaud non sucré.

Remarque

L'ordre d'exécution de cette marche à suivre est important. En effet, si l'utilisateur réalise l'opération 12 (`Allumer la cafetière`) avant l'opération 9 (`Verser l'eau dans la cafetière`), le résultat est sensiblement différent. La marche à suivre ainsi désordonnée risque de détériorer la cafetière électrique.

Cet exemple tiré de la vie courante montre que, pour résoudre un problème, il est essentiel de définir les objets utilisés puis de trouver la suite logique de tous les ordres nécessaires à la résolution dudit problème.

Question

Où placer les opérations supplémentaires, dans la liste ordonnée, pour faire un café sucré ?

Réponse

Les opérations se placent à la fin de la liste précédente de la façon suivante :

```
19. Prendre du sucre.
20. Poser le sucre dans la tasse.
21. Prendre une petite cuillère.
22. Poser la cuillère dans la tasse.
```

Vers une méthode

La tâche consistant à décrire comment résoudre un problème n'est pas simple. Elle dépend en partie du niveau de difficulté du problème et réclame un savoir-faire : la façon de procéder pour découper un problème en actions élémentaires.

Pour aborder dans les meilleures conditions possibles la tâche difficile d'élaboration d'un algorithme, nous devons tout d'abord :

- Déterminer les objets utiles à la résolution du problème.

- Construire et ordonner la liste de toutes les actions nécessaires à cette résolution.

Pour cela, il est nécessaire :

- d'analyser en détail la tâche à résoudre ;
- de fractionner le problème en actions distinctes et élémentaires.

Ce fractionnement est réalisé en tenant compte du choix des hypothèses de travail. Ces hypothèses imposent un ensemble de contraintes, qui permettent de savoir si l'action décrite est élémentaire et ne peut plus être découpée.

Cela fait, nous avons construit un algorithme.

Passer de l'algorithme au programme

Pour construire un algorithme, nous avons défini des hypothèses de travail, c'est-à-dire supposé une base de connaissances minimales nécessaires à la résolution du problème. Ainsi, le fait de prendre l'hypothèse d'avoir du café moulu nous autorise à ne pas décrire l'ensemble des tâches précédant l'acquisition du café moulu. C'est donc la connaissance de l'environnement de travail qui détermine en grande partie la construction de l'algorithme.

Pour passer de l'algorithme au programme, le choix de l'environnement de travail n'est plus de notre ressort. Jusqu'à présent, nous avons supposé que l'exécutant était humain. Maintenant, notre exécutant est l'ordinateur. Pour écrire un programme, nous devons savoir ce dont est capable un ordinateur et connaître son fonctionnement de façon à établir les connaissances et capacités de cet exécutant.

Qu'est-ce qu'un ordinateur ?

Notre intention n'est pas de décrire en détail le fonctionnement de l'ordinateur et de ses composants mais d'en donner une image simplifiée.

Pour tenter de comprendre comment travaille l'ordinateur et, surtout, comment il se programme, nous allons schématiser à l'extrême ses mécanismes de fonctionnement.

Un ordinateur est composé de deux parties distinctes, la **mémoire centrale** et l'**unité centrale**.

La mémoire centrale permet de mémoriser toutes les informations nécessaires à l'exécution d'un programme. Ces informations correspondent à des **données** ou à des ordres à exécuter (**instructions**). Les ordres placés en mémoire sont effectués par l'unité centrale, la partie active de l'ordinateur.

Lorsqu'un ordinateur exécute un programme, son travail consiste en grande partie à gérer la mémoire, soit pour y lire une instruction, soit pour y stocker une information. En ce sens, nous pouvons voir l'ordinateur comme un robot qui sait agir en fonction des ordres qui lui sont fournis. Ces actions, en nombre limité, sont décrites ci-après.

Déposer ou lire une information dans une case mémoire

La mémoire est formée d'éléments, ou cases, qui possèdent chacune un numéro (une adresse). Chaque case mémoire est en quelque sorte une boîte aux lettres pouvant contenir une information (une lettre). Pour y déposer cette information, l'ordinateur (le facteur) doit connaître l'adresse de la boîte. Lorsque le robot place une information dans une case mémoire, il mémorise l'adresse où se situe celle-ci afin de retrouver l'information en temps nécessaire.

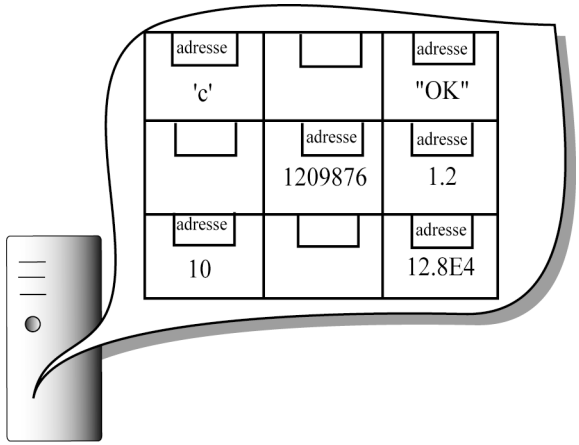

Figure I-1 *La mémoire de l'ordinateur est composée de cases possédant une adresse et pouvant contenir à tout moment une valeur.*

Le robot sait déposer une information dans une case, mais il ne sait pas la retirer (au sens de prendre un courrier déposé dans une boîte aux lettres). Lorsque le robot prend l'information déposée dans une case mémoire, il ne fait que la lire. En aucun cas il ne la retire ni ne l'efface. L'information lue reste toujours dans la case mémoire.

Remarque

Pour effacer une information d'une case mémoire, il est nécessaire de placer une nouvelle information dans cette même case. Ainsi, la nouvelle donnée remplace l'ancienne, et l'information précédente est détruite.

Exécuter des opérations simples telles que l'addition ou la soustraction

Le robot lit et exécute les opérations dans l'ordre où elles lui sont fournies. Pour faire une addition, il va chercher les valeurs à additionner dans les cases mémoire appropriées (stockées, par exemple, aux adresses `a` et `b)` et réalise ensuite l'opération demandée. Il enregistre alors le résultat de cette opération dans une case d'adresse `c`. De telles opérations sont décrites à l'aide d'ordres, appelés aussi **instructions**.

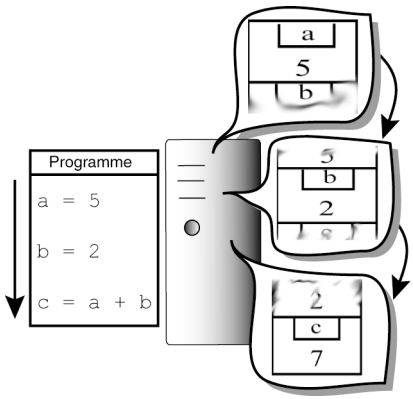

Figure I-2 *Le programme exécute les instructions dans l'ordre de leur apparition.*

Comparer des valeurs

Le robot est capable de comparer deux valeurs entre elles pour déterminer si l'une d'entre elles est plus grande, plus petite, égale ou différente de l'autre valeur. Grâce à la comparaison, le robot est capable de tester une condition et d'exécuter un ordre plutôt qu'un autre, en fonction du résultat du test.

La réalisation d'une comparaison ou d'un test fait que le robot ne peut plus exécuter les instructions dans leur ordre d'apparition. En effet, suivant le résultat du test, il doit rompre l'ordre de la marche à suivre, en sautant une ou plusieurs instructions. C'est pourquoi il existe des instructions particulières dites de **branchement**. Grâce à ce type d'instructions, le robot est à même non seulement de sauter des ordres mais aussi de revenir à un ensemble d'opérations afin de les répéter.

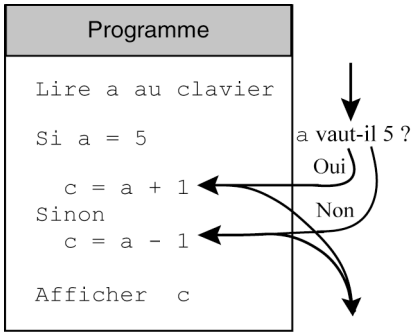

Figure I-3 *Suivant le résultat du test, l'ordinateur exécute l'une ou l'autre instruction en sautant celle qu'il ne doit pas exécuter.*

Communiquer une information élémentaire

Un programme est essentiellement un outil qui traite l'information. Cette information est transmise à l'ordinateur par l'utilisateur. L'information est saisie par l'intermédiaire du clavier ou de la souris. Cette transmission de données à l'ordinateur est appelée communication d'entrée (*input* en anglais). On parle aussi de **saisie** ou encore de lecture de données.

Après traitement, le programme fournit un résultat à l'utilisateur, soit par l'intermédiaire de l'écran, soit sous forme de fichiers, que l'on peut ensuite imprimer. Il s'agit alors de communication de sortie (*output*) ou encore d'**affichage** ou d'**écriture** de données.

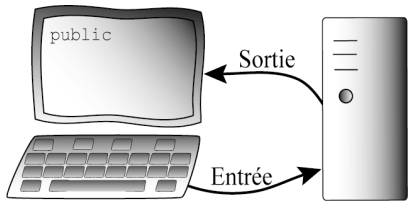

Figure I-4 *La saisie au clavier d'une valeur correspond à une opération d'entrée, et l'affichage d'un résultat à une opération de sortie.*

Coder l'information

De par la nature de ses composants électroniques, le robot ne perçoit que deux états : composant allumé et composant éteint. De cette perception découle le langage binaire, qui utilise par convention les deux symboles 0 (éteint) et 1 (allumé).

Ne connaissant que le 0 et le 1, l'ordinateur utilise un code pour représenter une information aussi simple qu'un nombre entier ou un caractère. Ce code est un programme, qui différencie chaque type d'information et transforme une information (donnée numérique ou alphabétique) en valeurs binaires. À l'inverse, ce programme sait aussi transformer un nombre binaire en valeur numérique ou alphabétique. Il existe autant de codes que de types d'informations. Cette différenciation du codage (en fonction de ce qui doit être représenté) introduit le concept de **type** de données.

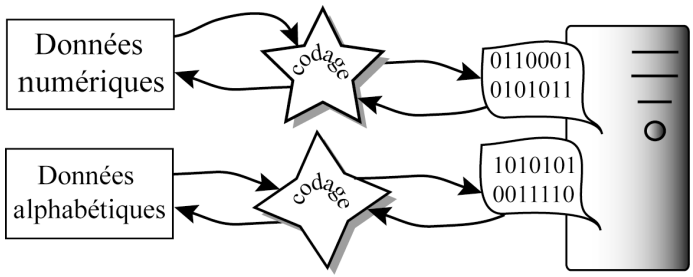

Figure I-5 *Toute information est codée en binaire. Il existe autant de codes que de types d'informations.*

Remarque Toute information fournie à l'ordinateur est, au bout du compte, codée en binaire. L'information peut être un simple nombre ou une instruction de programme.

Exemple

Pour additionner deux nombres, l'ordinateur fait appel aux trois éléments qui lui sont nécessaires pour réaliser cette opération. Ces éléments sont les suivants :

- Le code binaire représentant l'opération d'addition (par exemple 0101).
- L'adresse de la case mémoire où est stocké le premier nombre (par exemple 011101).
- L'adresse de la case mémoire où se trouve la deuxième valeur (par exemple 010101).

Pour finir, l'instruction d'addition de ces deux nombres s'écrit en assemblant les trois codes binaires (soit, dans notre exemple, 0101011101010101).

Remarque Le code binaire associé à chaque code d'opération (addition, test, etc.) n'est pas nécessairement identique d'un ordinateur à un autre. Ce code binaire est déterminé par le constructeur de l'ordinateur. De ce fait, une instruction telle que l'addition de deux nombres n'a pas le même code binaire d'une machine à une autre. Il existe donc, pour un même programme, un code binaire qui diffère suivant le type d'ordinateur utilisé.

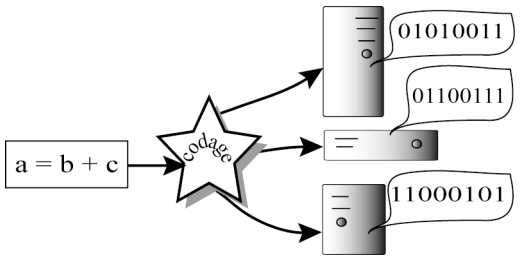

Figure I-6 *Pour un même programme, le code binaire diffère en fonction de l'ordinateur utilisé.*

L'ordinateur n'est qu'un exécutant

En pratique, le robot est très habile à réaliser l'ensemble des tâches énoncées ci-dessus. Il les exécute beaucoup plus rapidement qu'un être humain.

En revanche, le robot n'est pas doué d'intelligence. Il n'est ni capable de choisir une action plutôt qu'une autre, ni apte à exécuter de lui-même l'ensemble de ces actions. Pour qu'il puisse exécuter une instruction, il faut qu'un être humain détermine l'instruction la plus appropriée et lui donne l'ordre de l'exécuter.

Le robot est un exécutant capable de comprendre des ordres. Compte tenu de ses capacités limitées, les ordres ne peuvent pas lui être donnés dans le langage naturel propre à l'être humain.

En effet, le robot ne comprend pas le sens des ordres qu'il exécute mais seulement leur forme. Chaque ordre doit être écrit avec des mots particuliers et une forme, ou syntaxe, préétablie. L'ensemble de ces mots constitue un langage informatique. Les langages C, C++, Pascal, Basic, Fortran, Cobol et Java sont des langages de programmation, constitués de mots et d'ordres dont la syntaxe diffère selon le langage.

Pour écrire un programme, il est nécessaire de connaître un de ces langages, de façon à traduire un algorithme en un programme composé d'ordres.

Un premier programme en Java, ou comment parler à un ordinateur

Pour créer une application, nous allons avoir à décrire une liste ordonnée d'opérations dans un langage compréhensible par l'ordinateur. La contrainte est de taille et se porte essentiellement sur la façon de définir et de représenter les objets nécessaires à la résolution du problème en fonction du langage de l'ordinateur.

Pour bien comprendre la difficulté du travail à accomplir, regardons comment faire calculer à un ordinateur la circonférence d'un cercle de rayon quelconque.

Calcul de la circonférence d'un cercle

L'exercice consiste à calculer le périmètre d'un cercle de rayon quelconque. Nous supposons que l'utilisateur emploie le clavier pour transmettre au programme la valeur du rayon.

Définition des objets manipulés

Pour calculer la circonférence du cercle, l'ordinateur a besoin de stocker dans ses cases mémoire la valeur du rayon ainsi que celle du périmètre. Les objets à manipuler sont deux valeurs numériques appartenant à l'ensemble des réels $\mathbb{R}$. Nous appelons `lePerimetre` la valeur correspondant au périmètre et `unRayon` la valeur du rayon.

La liste des opérations

La circonférence d'un cercle est calculée à partir de la formule :

$$\texttt{LePerimetre} = 2 \times \pi \times \texttt{unRayon}.$$

La valeur du rayon est fournie par l'utilisateur à l'aide du clavier. Elle n'est donc pas connue au moment de l'écriture du programme. En conséquence, il est nécessaire d'écrire l'ordre (instruction) de saisie au clavier avant de calculer la circonférence.

La liste des opérations est la suivante :

```
1. Réserver deux cases mémoire pour y stocker les valeurs
   ➥correspondant au rayon (unRayon) et au périmètre (lePerimetre).
2. Demander à l'utilisateur de saisir la valeur du rayon au clavier
    ➥et la placer dans la case mémoire associée.
3. Connaissant la valeur du rayon, calculer la circonférence.
4. Afficher le résultat.
```

La valeur du rayon puis, après calcul, celle de la circonférence sont les données principales de ce programme. L'ordinateur doit les stocker en mémoire pour les utiliser.

L'opération `1` consiste à donner un nom aux cases mémoire qui vont servir à stocker ces données. Lors de cette opération, appelée **déclaration de variables**, l'ordinateur réserve une case mémoire pour chaque nom de variable défini. Ici, ces variables ont pour nom `P` et `R`. Au cours de cette réservation d'emplacements mémoire, l'ordinateur associe le nom de la variable et l'adresse réelle de la case mémoire.

Remarque

Pour le programmeur, le nom et l'adresse d'une case ne font qu'un, car il ne manipule les variables que par leur nom, alors que l'ordinateur travaille avec leur adresse. En donnant un nom à une case, l'être humain sait facilement identifier les objets qu'il manipule, alors qu'il lui serait pénible de manipuler les adresses binaires correspondantes. Inversement, en associant un nom à une adresse codée en binaire, l'ordinateur peut véritablement manipuler ces objets.

L'opération `2` permet de saisir au clavier la valeur du rayon. Pour que l'utilisateur non initié sache à quoi correspond la valeur saisie, il est nécessaire, avant de procéder à cette saisie, d'afficher un message explicatif à l'écran. L'opération `2` se décompose en deux instructions élémentaires, à savoir :

```
Afficher un message demandant à l'utilisateur du programme de saisir
➥une valeur pour le rayon.
Une fois la valeur saisie par l'utilisateur, la placer dans sa case
➥mémoire.
```

Les opérations `3` et `4` sont des actions élémentaires directement traduisibles en langage informatique.

La traduction en Java

Une application, ou programme, ne s'écrit pas en une seule fois. Nous verrons à la lecture de cet ouvrage que programmer c'est toujours décomposer une difficulté en différentes tâches plus aisées à réaliser. Cette décomposition s'applique aussi bien pour construire un algorithme que pour l'écriture du programme lui-même.

D'une manière générale, la meilleure façon de procéder pour fabriquer un programme revient à écrire une première ébauche et à la tester. De ces tests, il ressort des fautes à corriger et, surtout, de nouvelles idées. Le programme final consiste en l'assemblage de toutes ces corrections et de ces améliorations.

Pour traduire la marche à suivre définie précédemment selon les règles de syntaxe du langage Java, nous allons utiliser cette même démarche. Nous nous intéresserons, dans un premier temps, à la traduction du cœur du programme (opérations `1` à `4` décrites à la section précédente).

Nous verrons pour finir comment insérer l'ensemble de ces instructions dans une structure de programme Java.

- L'opération `1` consiste à déclarer les variables utilisées pour le calcul de la circonférence. Cette opération se traduit par l'instruction :

```
double unRayon, lePerimetre ;
```

  Par cette instruction, le programme demande à l'ordinateur de réserver deux cases mémoire, nommées `unRayon` et `lePerimetre`, pour y stocker les valeurs du rayon et de la circonférence. Le mot réservé `double` permet de préciser que les valeurs numériques sont réelles « avec une double précision », c'est-à-dire avec une précision pouvant aller jusqu'à 17 chiffres après la virgule.

Pour en savoir plus Pour plus d'informations sur la définition des types de variables, reportez-vous au chapitre 1, « Stocker une information ».

- L'opération `2` s'effectue en deux temps :
  1. Afficher un message demandant à l'utilisateur de saisir une valeur. Cette opération se traduit par l'instruction :

```
System.out.print("Valeur du rayon : ") ;
```

     `System.out.print()` est ce que l'on appelle un programme, ou une fonction, prédéfini par le langage Java. Ce programme permet d'écrire à l'écran le message spécifié à l'intérieur des parenthèses. Le message affiché est ici un fragment de texte, appelé, dans le jargon informatique, une chaîne de caractères. Pour que l'ordinateur comprenne que la chaîne de caractères n'est pas un nom de variable mais un texte à afficher, il faut placer entre guillemets (`" "`) tous les caractères composant la chaîne.
  2. Saisir et stocker la valeur demandée en mémoire. Pour ce faire, nous devons écrire les instructions suivantes :

```
Scanner lectureClavier = new Scanner(System.in);
unRayon = lectureClavier.nextDouble();
```

     `Scanner` est un outil (une classe) proposé à partir de la version 1.5 de Java qui permet à l'utilisateur de communiquer une valeur numérique au programme par l'intermédiaire du clavier. Cet outil utilise des fonctions (par exemple `nextDouble()`) qui donnent l'ordre à l'ordinateur d'attendre la saisie d'une valeur (de double précision, pour notre exemple). La saisie est effective lorsque l'utilisateur valide sa réponse en appuyant sur la touche `Entrée` du clavier.

  Une fois la valeur saisie, celle-ci est placée dans la case mémoire nommée `unRayon` grâce au signe =.

Pour en savoir plus Pour plus de précisions sur les deux méthodes `System.out.print()` et `lecture Clavier.nextDouble()` reportez-vous au chapitre 2, « Communiquer une information ». Pour le signe égal (=), voir le chapitre 1, « Stocker une information ». La notion de classe et l'opérateur `new` sont étudiés au chapitre 7, « Les classes et les objets ».

- L'opération `3` permet de calculer la valeur de la circonférence. Elle se traduit de la façon suivante :

```
lePerimetre = 2 * Math.PI * unRayon ;
```

  Le signe `*` est le symbole qui caractérise l'opération de multiplication. `Math.PI` est le terme qui représente la valeur numérique du nombre π avec une précision de 17 chiffres après la virgule. Le mot-clé `Math` désigne la boîte à outils composée de fonctions mathématiques accompagnant le langage Java. Cette bibliothèque contient, outre des constantes telles que `p`, des fonctions standards, comme `sqrt()` (racine carrée) ou `sin()` (sinus). Une fois les opérations de multiplication effectuées, la valeur calculée est placée dans la variable `lePerimetre` grâce au signe `=`.

- La dernière opération (`4`) de notre programme a pour rôle d'afficher le résultat du calcul précédent. Cet affichage est réalisé grâce à l'instruction :

```
System.out.print("Le cercle de rayon " + unRayon +
                 " a pour perimetre : " + lePerimetre);
```

  Ce deuxième appel à la fonction `System.out.print()` est plus complexe que le premier. Il mélange l'affichage de chaînes de caractères (texte entre guillemets) et de contenu de variables.

  Si les caractères `R` et `P` ne sont pas placés entre guillemets, c'est pour que l'ordinateur les interprète non pas comme des caractères à afficher mais comme les variables qui ont été déclarées en début de programme. De ce fait, il affiche le contenu des variables et non les lettres `R` et `P`.

  Les signes `+`, qui apparaissent dans l'expression `"Le cercle de rayon " + unRayon + " a pour perimetre : " + lePerimetre`, indiquent que chaque élément du message doit être affiché en le collant aux autres : d'abord la chaîne de caractères `"Le cercle de rayon "`, puis la valeur de `unRayon`, puis la chaîne `"a pour périmètre :"` et, pour finir, la valeur de `lePerimetre`.

En résumé, la partie centrale du programme contient les cinq instructions suivantes :

```
double unRayon, lePerimetre ;
Scanner lectureClavier = new Scanner(System.in);
System.out.print("Valeur du rayon : ") ;
```

```
unRayon = lectureClavier.nextDouble();
lePerimetre = 2 * Math.PI * unRayon ;
System.out.print("Le cercle de rayon " + unRayon +
                 " a pour perimetre : " + lePerimetre );
```

Pour améliorer la lisibilité du programme, il est possible d'insérer dans le programme, des commentaires, comme suit :

```
// 1. Déclarer les variables
double unRayon, lePerimetre ;
Scanner lectureClavier = new Scanner(System.in);
// 2.a Afficher le message "Valeur du rayon : " à l'écran
System.out.print("Valeur du rayon : ") ;
// 2.b Lire au clavier une valeur, placer cette valeur dans la
   ➥variable unRayon
unRayon = lectureClavier.nextDouble();
// 3. Calculer la circonférence en utilisant la formule consacrée
lePerimetre = 2 * Math.PI * unRayon ;
// 4. Afficher le résultat
System.out.print("Le cercle de rayon " + unRayon +
                 " a pour perimetre : " + lePerimetre );
```

Les lignes du programme qui débutent par les signes `//` sont considérées par l'ordinateur non pas comme des ordres à exécuter mais comme des lignes de commentaire. Elles permettent d'expliquer en langage naturel ce que réalise l'instruction associée.

Écrites de la sorte, ces instructions constituent le cœur de notre programme. Elles ne peuvent cependant pas encore être interprétées correctement par l'ordinateur. En effet, celui-ci exécute les instructions d'un programme dans l'ordre de leur arrivée. Une application doit donc être constituée d'une instruction qui caractérise le début du programme. Pour ce faire, nous devons écrire notre programme ainsi :

```
public static void main(String [] arg)
{
  // 1. Déclarer les variables
  double unRayon, lePerimetre ;
  Scanner lectureClavier = new Scanner(System.in);
  // 2.a Afficher le message "Valeur du rayon : " à l'écran
  System.out.print("Valeur du rayon : ") ;
  // 2.b Lire au clavier une valeur, placer cette valeur
  // dans la variable unRayon
  unRayon = lectureClavier.nextDouble();
  // 3. Calculer la circonférence en utilisant la formule consacrée
  lePerimetre = 2 * Math.PI * unRayon ;
```

```
  // 4. Afficher le résultat
  System.out.print("Le cercle de rayon " + unRayon +
                   " a pour perimetre : " + lePerimetre );
} // Fin de la fonction main()
```

La ligne `public static void main(String [] arg)` est l'instruction qui permet d'indiquer à l'ordinateur le début du programme. Ce début est identifié par ce que l'on appelle la fonction `main()`, c'est-à-dire la fonction principale du programme. De cette façon, lorsque le programme est exécuté, l'ordinateur recherche le mot-clé `main`. Une fois ce mot-clé trouvé, l'ordinateur exécute une à une chaque instruction constituant la fonction.

Pour en savoir plus Les autres mots-clés, tels que `public`, `static` ou `void`, déterminent certaines caractéristiques de la fonction `main()`. Ces mots-clés, obligatoirement placés et écrits dans cet ordre, sont expliqués au fur et à mesure de leur apparition dans le livre et plus particulièrement à la section « Quelques techniques utiles » du chapitre 9, « Collectionner un nombre fixe d'objets ».

Pour finir, nous devons insérer la fonction `main()` dans ce qui est appelé une classe Java. En programmation objet, un programme n'est exécutable que s'il est défini à l'intérieur d'une classe. Une classe est une entité interprétée par l'ordinateur comme étant une unité de programme, qu'il peut exécuter dès qu'un utilisateur le souhaite.

Aucun programme ne peut être écrit en dehors d'une classe. Nous devons donc placer la fonction `main()` à l'intérieur d'une classe définie par l'instruction `public class Cercle {}`, comme suit :

```
public class Cercle
{
 public static void main(String [] arg)
 {
   // 1. Déclarer les variables
   double unRayon, lePerimetre ;
   Scanner lectureClavier = new Scanner(System.in);
   // 2.a Afficher le message "Valeur du rayon : " à l'écran
   System.out.print("Valeur du rayon : ") ;
   // 2.b Lire au clavier une valeur, placer cette valeur dans
   // la variable unRayon
   unRayon = lectureClavier.nextDouble();
   // 3. Calculer la circonférence en utilisant la formule consacrée
   lePerimetre = 2 * Math.PI * unRayon ;
```

```
    // 4. Afficher le résultat
    System.out.print("Le cercle de rayon " + unRayon +
                     " a pour perimetre : " + lePerimetre );
  }
} // Fin de la classe Cercle
```

Nous obtenons ainsi le programme dans son intégralité. La ligne `public class Cercle` permet de définir une classe. Puisque notre programme effectue des opérations sur un cercle, nous avons choisi d'appeler cette classe `Cercle`. Nous aurions pu lui donner un tout autre nom, comme `Rond` ou `Exemple`. Ainsi définie, la classe `Cercle` devient un programme à part entière.

Pour finir, il convient de débuter le programme par l'instruction :

```
import java.util.*;
```

Cette instruction est obligatoire lorsqu'on utilise la classe `Scanner`. Elle indique au compilateur qu'il doit charger les classes (et notamment la classe `Scanner`) enregistrées dans la boîte à outils (*package*) `java.util` avant de commencer la phase de compilation. Si vous omettez cette instruction, le compilateur Java signale une erreur du type `cannot find symbol class Scanner`.

Remarque

Les classes standards de Java sont regroupées par paquetage (en anglais *package*). Par exemple toutes les classes relatives au traitement de texte son regroupées dans le paquetage `java.text`, le paquetage `java.awt` fournit quant à lui toutes les classes correspondant à la gestion des interfaces graphiques.

Pour en savoir plus

Pour voir le résultat de l'exécution de ce programme, reportez-vous à la section « Exemple sur plate-forme Unix », ci-après.

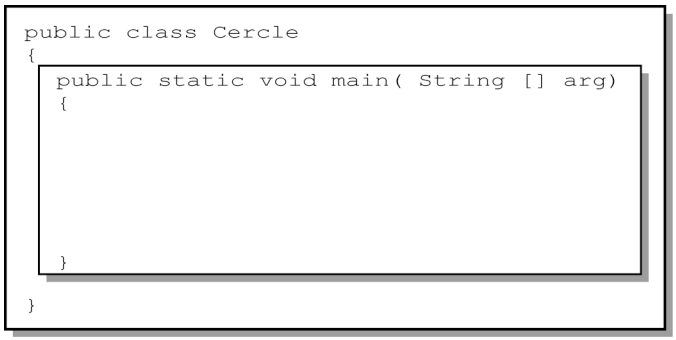

Figure I-7 *Un programme Java est constitué de deux blocs encastrés. Le premier bloc représente la classe associée au programme, tandis que le second détermine la fonction principale.*

En observant la figure 7, nous constatons que ce programme, de même que tous ceux à venir, est constitué de deux blocs encastrés définis par les deux lignes `public class Cercle{}` et `public static void main(String [] arg){}`.

Ces deux blocs constituent la charpente principale et nécessaire à tout programme écrit avec le langage Java. Cet exemple montre en outre que les mots réservés par le langage Java sont nombreux et variés et qu'ils constituent une partie du langage Java.

Si la syntaxe, c'est-à-dire la forme, de ces instructions peut paraître étrange de prime abord, nous verrons à la lecture de cet ouvrage que leur emploi obéit à des règles strictes. En apprenant ces règles et en les appliquant, vous pourrez vous initier aux techniques de construction d'un programme, qui reviennent à décomposer un problème en actions élémentaires puis à traduire celles-ci à l'aide du langage Java.

Question

Dans la classe `Cercle`, quelle instruction faut-il modifier pour calculer non plus le périmètre, mais la surface d'un cercle ?

Réponse

La surface d'un cercle est obtenue par la formule : `Surface = π × Rayon × Rayon.`

Il suffit donc de modifier l'instruction de déclaration :

```
// Déclaration des variables
  double unRayon, lePerimetre ;
```

en :

```
// Déclaration des variables
  double unRayon, laSurface ;
```

Puis de remplacer l'instruction :

```
// Calculer la circonférence en utilisant la formule consacrée
  lePerimetre = 2*Math.PI*unRayon ;
```

en :

```
// Calculer la surface en utilisant la formule consacrée
  laSurface = Math.PI*unRayon*unRayon ;
```

Et pour finir remplacer l'instruction :

```
   // . Afficher le résultat
   System.out.print("Le cercle de rayon " + unRayon +
                    " a pour perimetre : " + lePerimetre );
```

en :

```
   // . Afficher le résultat
   System.out.print("Le cercle de rayon " + unRayon +
                    " a pour surface : " + laSurface );
```