Chapitre 3
Introduction à la plate-forme Java

1. Introduction

Au cœur des ordinateurs, des listes d'instructions sont exécutées par les microprocesseurs. Ces listes contiennent des codes binaires représentant des calculs et des actions à faire sur des registres, de la mémoire ou encore sur des périphériques. Sauf cas très rare, lorsqu'un développeur écrit un programme, il ne construit pas directement cette liste d'instructions aussi appelée code de bas niveau ou langage assembleur. Heureusement pour lui, il passe par un langage de programmation avec lequel il va décrire un enchaînement d'actions de « haut niveau ». Par exemple, si le programme a besoin d'ouvrir le fichier config.txt alors le développeur utilisera la fonction `fopen("config.txt")` de son langage favori pour le faire. Derrière cette ligne se cachera toute une liste d'instructions « machine » que le microprocesseur exécutera quand l'application sera exécutée. Les langages de hauts niveaux simplifient donc grandement l'écriture mais, au final, ce sont toujours des instructions machine qui seront déroulées.

Le programme source – dans notre exemple, celui qui contient `fopen("config.txt")` – est un fichier texte que le développeur construit avec un éditeur de texte basique comme Notepad, Notepad++ ou bien encore l'éditeur intégré à son environnement de développement (IDE), présenté plus loin. Ce fichier texte tout seul ne peut pas être exécuté par la machine. Il doit être converti directement ou indirectement en langage « machine » pour devenir exécutable.

Cette translation peut s'opérer de trois façons différentes :

– **La compilation directe** : le fichier est converti en un contenu binaire directement exécutable par le système d'exploitation. C'est le cas de la plupart des programmes écrits en C et C++ (la plupart, car il est possible d'écrire des programmes .NET en C++, mais c'est une autre histoire).

– **L'interprétation** : le fichier est converti par un interpréteur « à la volée ». C'est le cas des scripts PHP dans les pages web. Le navigateur internet du client demande une page à un serveur web. Le serveur web détecte que la page demandée contient un script PHP car son contenu est dynamique (exemple : liste de clients, commandes en cours, etc.). Il la passe alors à son analyseur qui va l'interpréter et composer dynamiquement la page HTML. Cette méthode est très pratique dans le cas du Web car les scripts sont écrits au format texte. En cas de modifications, pas besoin de recompiler quoi que ce soit, il suffit de mettre à jour le fichier dans l'arborescence du serveur et le changement est immédiatement actif. Bien sûr, l'ensemble (décodage du PHP, son exécution et réencodage en HTML) peut s'avérer lent.

– **La compilation indirecte** : c'est le cas de Java et de C# et c'est un mélange des deux précédents... En fait, le fichier source est compilé dans un format binaire intermédiaire qui va être exécuté très rapidement par une machine virtuelle. Pourquoi cette machine virtuelle entre le programme et le système d'exploitation ? Cette machine virtuelle sert à plusieurs choses : tout d'abord elle va vérifier que le programme source ne fait pas de « bêtises » pendant son exécution en allant écrire, par exemple, dans la mémoire des autres applications, et ainsi rendre le système d'exploitation instable. Elle va également s'occuper de la gestion de la mémoire, de la sécurité, des droits, etc. La machine virtuelle est l'amie des systèmes d'exploitation et l'amie des développeurs. D'un côté, elle assure une exécution machine la plus fiable possible, et de l'autre, elle fédère un code source (pratiquement) unique pour toutes les plates-formes.

Microsoft Windows propose .NET et sa machine virtuelle. .NET se programme à partir de nombreux langages (même si le C# a été conçu pour lui). Ce puissant framework était au départ très lié aux systèmes d'exploitation Microsoft (PC, serveurs, Web, tablettes, téléphones) mais est maintenant multiplate-forme et open source. Cet effort d'ouverture de Microsoft est relativement récent alors que, depuis le départ, Java a pu s'exécuter sur la plupart des systèmes d'exploitation dès qu'ils proposaient une machine virtuelle appropriée.

Remarque

Une plate-forme Java désigne donc un environnement d'exécution pour un système d'exploitation sur une machine donnée et l'environnement de développement associé.

2. Environnement d'exécution

Les applications écrites en Java ne communiquent jamais directement avec le système d'exploitation. D'ailleurs, le résultat d'une compilation Java (fichiers d'extension *.class*) n'est pas directement exécutable par le système d'exploitation. Cette première mouture, appelée *Byte Code*, contient des instructions pour la machine virtuelle Java qui va « convertir à la volée » le code intermédiaire en instructions compatibles avec l'environnement réel d'exécution. On parle de *Just In Time Compiler* (ou *JIT Compiler*).

Chaque environnement réel d'exécution (Windows, Linux, macOS...) a sa propre machine virtuelle Java, mais le *Byte Code* généré après la compilation du programme source est le même et est relativement portable entre les différentes machines virtuelles Java. L'adverbe « relativement » est ajouté ici pour préciser que la conception d'un programme « portable » requiert quelques précautions. Imaginons que vous souhaitiez réaliser un programme qui devra être fonctionnel à la fois sur PC, tablette et téléphone. Il est évident que les caractéristiques de ces trois dispositifs diffèrent totalement, et il faudra donc que le programme s'adapte automatiquement à l'environnement d'exécution. Il pourra, par exemple, lui demander la résolution du dispositif d'affichage pour adapter sa présentation en conséquence.

Remarque

Revers de la médaille : le résultat de la compilation peut être facilement désassemblé, c'est-à-dire converti du Byte Code en lignes de programmes source Java.

Cela est problématique quand, par exemple, vos concurrents parviennent à reconstituer votre code source à partir du produit compilé que vous commercialisez... Il est possible de rendre le résultat de cette action moins lisible en effectuant un traitement d'obscurcissement (obfuscation) avec des outils spécialisés tels que ProGuard (open source). Sachez que le problème est le même pour nos collègues développeurs C#...

3. Une librairie très complète

La plate-forme Java propose une très large collection de classes sur lesquelles s'appuient les applications. On parle d'API Java. Ces classes simplifient considérablement la gestion d'objets courants (chaînes de caractères, valeurs décimales, collections, etc.), mais aussi la gestion de fichiers, des interfaces graphiques classiques, les API web, l'accès aux bases de données, les communications réseau, la sécurité, les diagnostics, etc. La liste des classes est très conséquente et, même si elles sont proposées de façon hiérarchique, le problème est souvent de savoir s'y repérer !

Ces classes sont, la plupart du temps, extensibles. Cela veut dire qu'il est possible de les étendre pour profiter de leurs comportements de base puis d'y ajouter des spécificités métier.

Pour organiser ces classes, la plate-forme utilise le concept des packages qui regroupent les classes par finalités (services de même nature). Par exemple, le package `java.io` contient une « boîte à outils » pour gérer les fichiers.

Il est naturellement possible de créer ses propres packages. Le package Java correspond au package de l'UML. Le découpage effectué veille à réduire les dépendances entre les packages. Pour qu'un programme puisse utiliser un package, ce dernier doit être référencé dans le projet et son utilisation doit être déclarée en haut du fichier source concerné grâce à la directive `import`, par exemple : `import java.io.*;`.

4. Des outils de développement performants

La plate-forme Java propose un compilateur (`javac.exe`) que l'on peut utiliser directement en ligne de commandes après avoir saisi le code du programme dans un éditeur de texte. La procédure est tout à fait possible mais pas très productive... Évidemment, le développeur recherche l'utilisation d'un environnement de développement totalement intégré, l'assistant pendant la rédaction du code, la réalisation de l'interface graphique, la mise au point et le déploiement de son application. Ce programme est un IDE (*Integrated Development Environment*). Il intègre au moins un éditeur de code source, des outils automatisant les compilations et un débogueur.

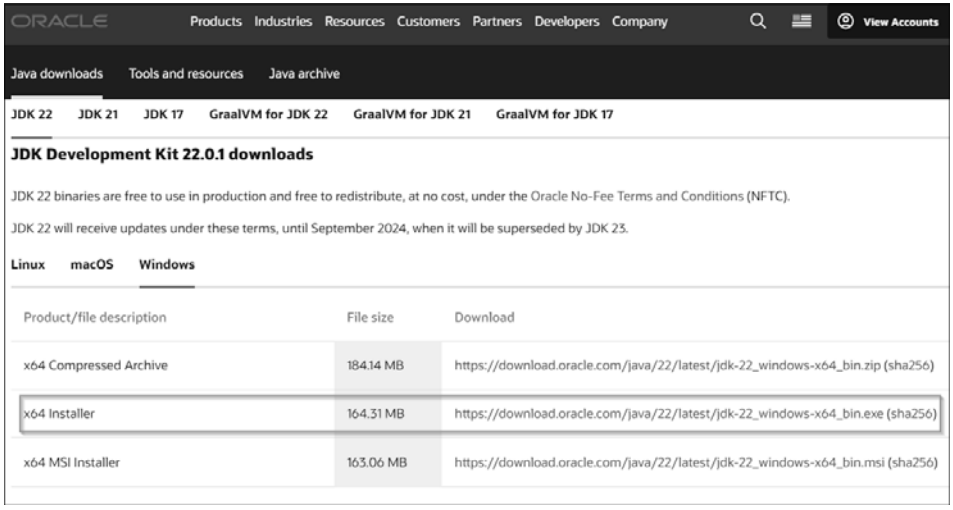
Il existe plusieurs IDE pour Java avec parmi eux trois grandes références : IntelliJ IDEA de JetBrains, Eclipse de la fondation Eclipse et NetBeans créé à l'initiative de Sun Microsystems et maintenu et distribué par Oracle et par Apache Software Foundation pour sa version Apache NetBeans. Les démonstrations et exercices de ce livre sont basés sur l'utilisation d'IntelliJ IDEA car c'est l'IDE qui nous a semblé le plus simple pour commencer à développer.

5. Téléchargement et installation du JDK

- Pour développer en Java nous aurons besoin d'un « kit de développement » appelé JDK sur lequel nous reviendrons très vite… Pour l'instant, rendez-vous sur le site Oracle à cette adresse : https://www.oracle.com/java/technologies/downloads/#jdk22-windows pour télécharger la version du JDK 22 avec son programme d'installation.

Remarque

Cette version 22 ne sera peut-être pas la dernière version lorsque vous lirez ces lignes.

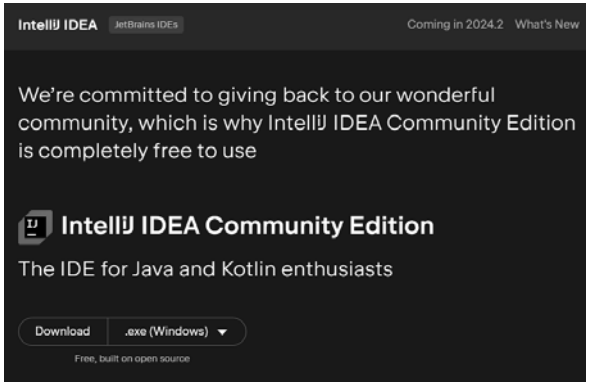


- Exécutez le programme téléchargé : jdk-22_windows-x64_bin.exe en gardant les options par défaut proposées dans les différentes boîtes de dialogue.

6. Téléchargement et installation d'IntelliJ IDEA

- Rendez-vous maintenant sur le site officiel de JETBRAINS (https://www.jetbrains.com/idea/download/?section=windows) pour télécharger la version Community Edition de cet IDE.

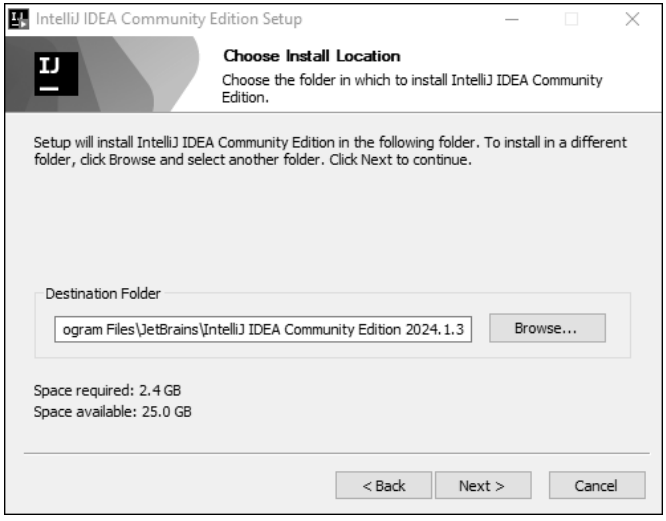
- Faites défiler l'écran jusqu'à l'affichage suivant :



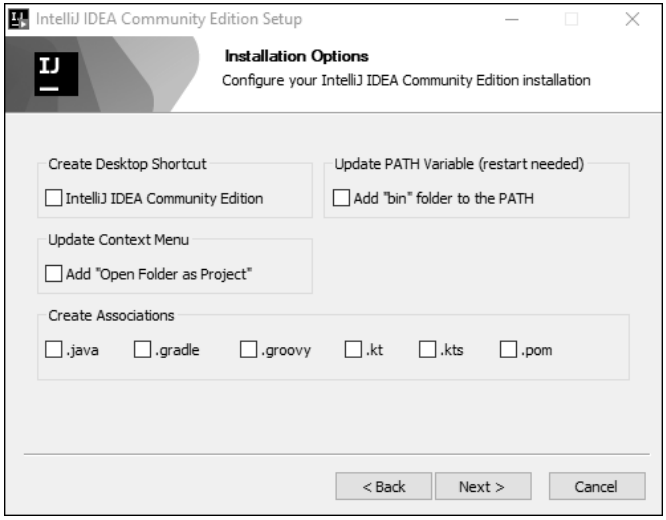
- Cliquez sur le bouton **Download**.

- Une fois le téléchargement terminé, double cliquez sur le fichier reçu (ideaIC-2024.1.3.exe à l'écriture de cet ouvrage).
- Autorisez l'application à apporter des modifications.

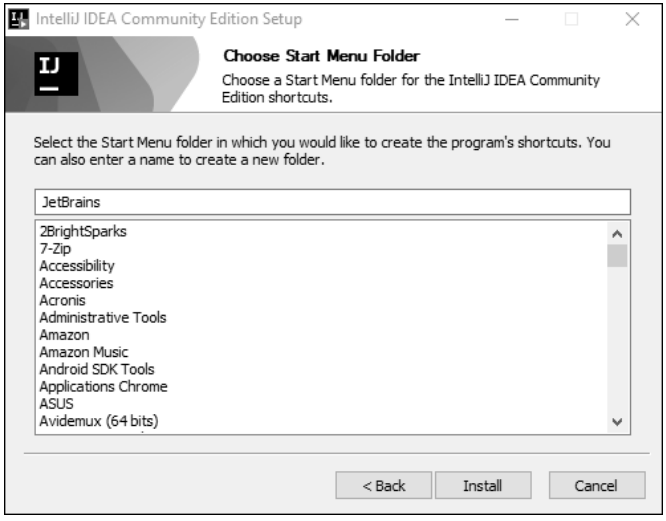
▶Cliquez sur le bouton **Next**.



▶Conservez les valeurs proposées par défaut et cliquez sur le bouton **Next**.



▶Conservez les valeurs proposées par défaut et cliquez sur le bouton **Next**.



▶Conservez les valeurs proposées par défaut et cliquez sur le bouton **Install**.