CYRILLE HERBY

APPRENEZ À PROGRAMMER EN JAVA

LA PROGRAMMATION PROFESSIONNELLE À LA PORTÉE DE TOUS !

le Livre du Zéro
www.siteduzero.com

zCorrecteurs.fr

Cet ouvrage a bénéficié des relectures attentives des zCorrecteurs.

Simple IT 2011 - ISBN : 978-2-9535278-3-4

Avant-propos

Si vous lisez ces lignes, c'est que nous avons au moins deux choses en commun : l'informatique vous intéresse et vous avez envie d'apprendre à programmer. Enfin, quand je dis en commun, je voulais dire en commun avec moi au moment où je voulais apprendre la programmation.

Pour moi, tout a commencé sur un site maintenant très connu : le Site du Zéro. Étant débutant et cherchant à tout prix des cours adaptés à mon niveau, je suis naturellement tombé amoureux de ce site qui propose des cours d'informatique accessibles au plus grand nombre. Vous l'aurez sans doute remarqué, trouver un cours d'informatique simple et clair (sur les réseaux, les machines, la programmation...) est habituellement un vrai parcours du combattant.

Je ne me suis pas découragé et je me suis professionnalisé, via une formation diplômante, tout en suivant l'actualité de mon site préféré... Au sein de cette formation, j'ai pu voir divers aspects de mon futur métier, notamment la programmation dans les langages PHP, C#, JavaScript et, bien sûr, Java. Très vite, j'ai aimé travailler avec ce dernier, d'une part parce qu'il est agréable à manipuler, souple à utiliser en demandant toutefois de la rigueur (ce qui oblige à structurer ses programmes), et d'autre part parce qu'il existe de nombreuses ressources disponibles sur Internet (mais pas toujours très claires pour un débutant).

J'ai depuis obtenu mon diplôme et trouvé un emploi, mais je n'ai jamais oublié la difficulté des premiers temps. Comme le Site du Zéro permet d'écrire des tutoriels et de les partager avec la communauté, j'ai décidé d'employer les connaissances acquises durant ma formation et dans mon travail à rédiger un tutoriel permettant d'aborder mon langage de prédilection avec simplicité. J'ai donc pris mon courage à deux mains et j'ai commencé à écrire. Beaucoup de lecteurs se sont rapidement montrés intéressés, pour mon plus grand plaisir.

De ce fait, mon tutoriel a été mis en avant sur le site et, aujourd'hui, il est adapté dans la collection « Livre du Zéro ». Je suis heureux du chemin parcouru, heureux d'avoir pu aider tant de débutants et heureux de pouvoir vous aider à votre tour !

Et Java dans tout ça ?

Java est un langage de programmation très utilisé, notamment par un grand nombre de développeurs professionnels, ce qui en fait un langage incontournable actuellement.

Voici les caractéristiques de Java en quelques mots.

- Java est un langage de programmation moderne développé par Sun Microsystems, aujourd'hui racheté par Oracle. Il ne faut surtout pas le confondre avec JavaScript (langage de script utilisé sur les sites Web), car ils n'ont rien à voir.
- Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc.
- On peut faire de nombreux types de programmes avec Java :
  - des applications, sous forme de fenêtre ou de console ;
  - des applets, qui sont des programmes Java incorporés à des pages Web ;
  - des applications pour appareils mobiles, comme les smartphones, avec J2ME (Java 2 Micro Edition) ;
  - des sites web dynamiques, avec J2EE (Java 2 Enterprise Edition, maintenant JEE) ;
  - et bien d'autres : JMF (Java Media Framework), J3D pour la 3D...

Comme vous le voyez, Java permet de réaliser une très grande quantité d'applications différentes ! Mais... comment apprendre un langage si vaste qui offre tant de possibilités ? Heureusement, ce livre est là pour tout vous apprendre sur Java à partir de zéro.

Java est donc un langage de programmation, un langage dit compilé : il faut comprendre par là que ce que vous allez écrire n'est pas directement compréhensible et utilisable par votre ordinateur. Nous devrons donc passer par une étape de compilation (étape obscure où votre code source est entièrement transformé). En fait, on peut distinguer trois grandes phases dans la vie d'un code Java :

- la phase d'écriture du code source, en langage Java ;
- la phase de compilation de votre code ;
- la phase d'exécution.

Ces phases sont les mêmes pour la plupart des langages compilés (C, C++...). Par contre, ce qui fait la particularité de Java, c'est que le résultat de la compilation n'est pas directement utilisable par votre ordinateur.

Les langages mentionnés ci-dessus permettent de faire des programmes directement compréhensibles par votre machine après compilation, mais avec Java, c'est légèrement différent. En C++ par exemple, si vous voulez faire en sorte que votre programme soit exploitable sur une machine utilisant Windows et sur une machine utilisant Linux, vous allez devoir prendre en compte les spécificités de ces deux systèmes d'exploitation dans votre code source et compiler une version spéciale pour chacun d'eux.

Avec Java, c'est un programme appelé **la machine virtuelle** qui va se charger de retranscrire le résultat de la compilation en langage machine, interprétable par celle-ci. Vous n'avez pas à vous préoccuper des spécificités de la machine qui va exécuter votre programme : la machine virtuelle Java s'en charge pour vous !

Qu'allez-vous apprendre en lisant ce livre ?

Ce livre a été conçu en partant du principe que vous ne connaissez rien à la programmation. Voilà le plan en quatre parties que nous allons suivre tout au long de cet ouvrage.

1. **Les bases de Java** : nous verrons ici ce qu'est Java et comment il fonctionne. Nous créerons notre premier programme, en utilisant des variables, des opérateurs, des conditions, des boucles... Nous apprendrons les bases du langage, qui vous seront nécessaires par la suite.
2. **Java et la Programmation Orientée Objet** : après avoir dompté les bases du langage, vous allez devoir apprivoiser une notion capitale : l'objet. Vous apprendrez à encapsuler vos morceaux de code afin de les rendre modulables et réutilisables, mais il y aura du travail à fournir.
3. **Les interfaces graphiques** : là, nous verrons comment créer des interfaces graphiques et comment les rendre interactives. C'est vrai que jusqu'à présent, nous avons travaillé en mode console. Il faudra vous accrocher un peu car il y a beaucoup de composants utilisables, mais le jeu en vaut la chandelle ! Nous passerons en revue différents composants graphiques tels que les champs de texte, les cases à cocher, les tableaux, les arbres ainsi que quelques notions spécifiques comme le *drag'n drop*.
4. **Interactions avec les bases de données** : de nos jours, avec la course aux données, beaucoup de programmes doivent interagir avec ce qu'on appelle des *bases de données*. Dans cette partie, nous verrons comment s'y connecter, comment récupérer des informations et comment les exploiter.

Comment lire ce livre ?

Suivez l'ordre des chapitres

Lisez ce livre comme on lit un roman. Il a été conçu de cette façon.

Contrairement à beaucoup de livres techniques où il est courant de lire en diagonale et de sauter certains chapitres, ici il est très fortement recommandé de suivre l'ordre du cours, à moins que vous ne soyez déjà un peu expérimentés.

Pratiquez en même temps

Pratiquez régulièrement. N'attendez pas d'avoir fini la lecture de ce livre pour allumer votre ordinateur et faire vos propres essais.

Utilisez les codes web !

Afin de tirer parti du Site du Zéro dont est issu ce livre, celui-ci vous propose ce qu'on appelle des « codes web ». Ce sont des codes à six chiffres à entrer sur une page du Site du Zéro pour être automatiquement redirigé vers un site web sans avoir à en recopier l'adresse.

Pour utiliser les codes web, rendez-vous sur la page suivante [1] :

`http://www.siteduzero.com/codeweb.html`

Un formulaire vous invite à rentrer votre code web. Faites un premier essai avec le code ci-dessous :

▷ Tester le code web
Code web : 123456

Ces codes web ont deux intérêts :

- vous faire télécharger les codes source inclus dans ce livre, ce qui vous évitera d'avoir à recopier certains codes un peu longs ;
- vous rediriger vers les sites web présentés tout au long du cours.

Ce système de redirection nous permet de tenir à jour le livre que vous avez entre les mains sans que vous ayez besoin d'acheter systématiquement chaque nouvelle édition. Si un site web change d'adresse, nous modifierons la redirection mais le code web à utiliser restera le même. Si un site web disparaît, nous vous redirigerons vers une page du Site du Zéro expliquant ce qui s'est passé et vous proposant une alternative.

En clair, c'est un moyen de nous assurer de la pérennité de cet ouvrage sans que vous ayez à faire quoi que ce soit !

Ce livre est issu du Site du Zéro

Cet ouvrage reprend le cours Java présent sur le Site du Zéro dans une édition revue et corrigée, avec de nombreuses mises à jour.

Il reprend les éléments qui ont fait le succès des cours du site, c'est-à-dire leur approche progressive et pédagogique, le ton décontracté et léger, ainsi que les TP vous permettant de réellement pratiquer de façon autonome.

Ce livre s'adresse donc à toute personne désireuse d'apprendre les bases de la programmation en Java, que ce soit :
- par curiosité ;
- par intérêt personnel ;
- par besoin professionnel.

1. Vous pouvez aussi utiliser le formulaire de recherche du Site du Zéro, section « Code Web ».

Remerciements

Comme pour la plupart des ouvrages, beaucoup de personnes ont participé de près ou de loin à l'élaboration de ce livre et j'en profite donc pour les en remercier.

- Ma compagne, Manuela, qui me supporte et qui tolère mes heures passées à écrire les tutoriels pour le Site du Zéro. Un merci spécial à toi qui me prends dans tes bras lorsque ça ne va pas, qui m'embrasses lorsque je suis triste, qui me souris lorsque je te regarde, qui me donnes tant d'amour lorsque le temps est maussade : pour tout ça et plus encore, je t'aime ;
- Agnès HAASSER (Tûtie), Damien SMEETS (Karl Yeurl), Mickaël SALAMIN (micky), François GLORIEUX (Nox), Christophe TAFANI-DEREEPER, Romain CAMPILLO (Le Chapelier Toqué), Charles DUPRÉ (Barbatos), Maxence CORDIEZ (Ziame), Philippe LUTUN (ptipilou), zCorrecteurs m'ayant accompagné dans la correction de cet ouvrage ;
- Mathieu NEBRA (alias M@teo21), père fondateur du Site du Zéro, qui m'a fait confiance, soutenu dans mes démarches et qui m'a donné de précieux conseils ;
- Tous les Zéros qui m'ont apporté leur soutien et leurs remarques ;
- Toutes les personnes qui m'ont contacté pour me faire des suggestions et m'apporter leur expertise.

Merci aussi à toutes celles et ceux qui m'ont apporté leur soutien et qui me permettent d'apprendre toujours plus au quotidien, mes collègues de travail :

- Thomas, qui a toujours des questions sur des sujets totalement délirants ;
- Angelo, mon chef adoré, qui est un puits de science en informatique ;
- Olivier, la force zen, qui n'a pas son pareil pour aller droit au but ;
- Dylan, discret mais d'une compétence plus que certaine dans des domaines aussi divers que variés ;
- Jérôme, que j'ai martyrisé mais qui, j'en suis persuadé, a adoré... :-)

Sommaire

Chapitre 1

Installer les outils de développement

Difficulté :

L'un des principes phares de Java réside dans sa machine virtuelle : celle-ci assure à tous les développeurs Java qu'un programme sera utilisable avec tous les systèmes d'exploitation sur lesquels est installée une machine virtuelle Java. Lors de la phase de compilation de notre code source, celui-ci prend une forme intermédiaire appelée **byte code** : c'est le fameux code inintelligible pour votre machine, mais interprétable par la machine virtuelle Java. Cette dernière porte un nom : on parle plus communément de **JRE** (**J**ava **R**untime **E**nvironment). Plus besoin de se soucier des spécificités liées à tel ou tel OS (*Operating System*, soit système d'exploitation). Nous pourrons donc nous consacrer entièrement à notre programme.

Afin de nous simplifier la vie, nous allons utiliser un outil de développement, ou **IDE** (**I**ntegrated **D**evelopment **E**nvironment), pour nous aider à écrire nos futurs codes source... Nous allons donc avoir besoin de différentes choses afin de pouvoir créer des programmes Java : la première est ce fameux JRE !

Installer les outils nécessaires

JRE ou JDK

Téléchargez votre environnement Java sur le site d'Oracle.

▷ Télécharger JRE
Code web : 924260

Choisissez la dernière version stable (figure 1.1).

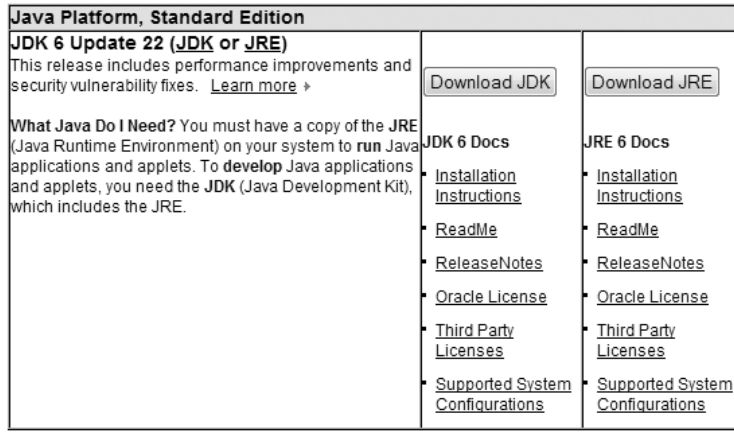

FIGURE 1.1 – Encart de téléchargement

Vous avez sans doute remarqué qu'on vous propose de télécharger soit le JRE, soit le **JDK**[1]. La différence entre ces deux environnements est écrite, mais pour les personnes fâchées avec l'anglais, sachez que le JRE contient tout le nécessaire pour faire en sorte que vos programmes Java puissent être exécutés sur votre ordinateur ; le JDK, en plus de contenir le JRE, contient tout le nécessaire pour développer, compiler...

L'IDE contenant déjà tout le nécessaire pour le développement et la compilation, nous n'avons besoin que du JRE. Une fois que vous avez cliqué sur « `Download JRE` », vous arrivez sur la page correspondante (figure 1.2).

Sélectionnez votre système d'exploitation et cochez la case : « `I agree to the Java SE Development Kit 6 License Agreement` ». Lorsque vous serez à l'écran correspondant (figure 1.3), sélectionnez celui de votre choix puis validez.

Je vous ai dit que Java permet de développer différents types d'applications : il y a donc des environnements permettant de créer des programmes pour différentes plates-formes.

– `J2SE`[2] : permet de développer des applications dites « client lourd », par exemple

1. **J**ava **D**evelopment **K**it.
2. Java 2 Standard Edition, celui qui nous intéresse dans cet ouvrage.

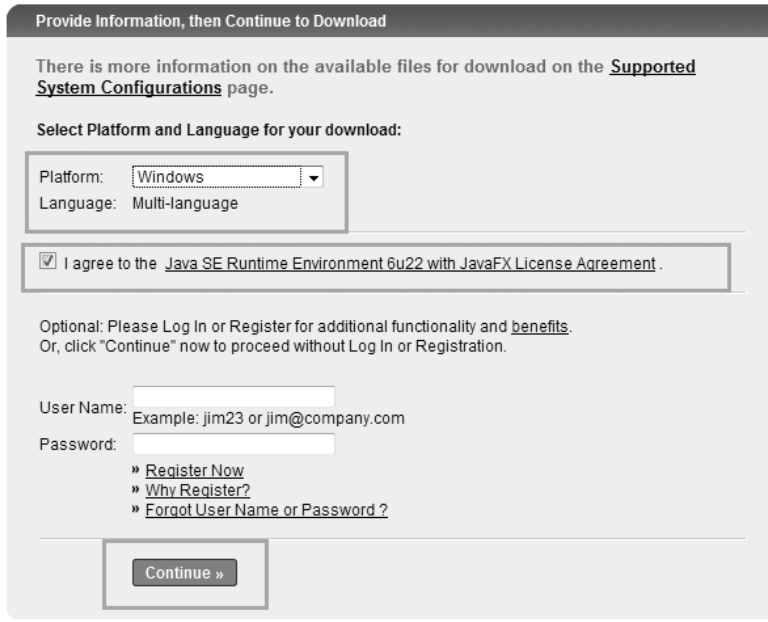

FIGURE 1.2 – Page de choix de votre système d'exploitation

FIGURE 1.3 – Choix de l'exécutable

Word, Excel, la suite OpenOffice.org... Toutes ces applications sont des « clients lourds ». C'est ce que nous allons faire dans ce livre.
- `J2EE`[3] : permet de développer des applications web en Java. On parle aussi de clients légers.
- `J2ME`[4] : permet de développer des applications pour appareils portables, comme des téléphones portables, des PDA...

Eclipse IDE

Avant toute chose, quelques mots sur le projet Eclipse.

Eclipse IDE est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP...). C'est en somme l'outil que nous allons utiliser pour programmer.

Eclipse IDE est lui-même principalement écrit en Java.

Je vous invite donc à télécharger Eclipse IDE.

▷ Télécharger Eclipse
Code web : 395144

Accédez à la page de téléchargement puis choisissez « `Eclipse IDE for Java Developers` », en choisissant la version d'Eclipse correspondant à votre OS[5] (figure 1.4).

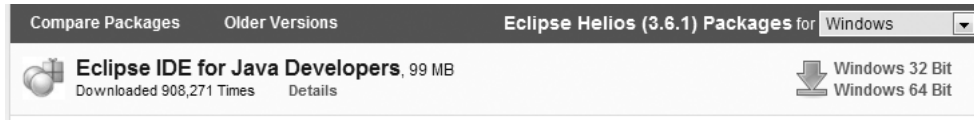

FIGURE 1.4 – Version d'Eclipse IDE

Sélectionnez maintenant le miroir que vous souhaitez utiliser pour obtenir Eclipse. Voilà, vous n'avez plus qu'à attendre la fin du téléchargement.

Pour ceux qui l'avaient deviné, Eclipse est le petit logiciel qui va nous permettre de développer nos applications ou nos applets, et aussi celui qui va compiler tout ça. Notre logiciel va donc permettre de traduire nos futurs programmes Java en langage **byte code**, compréhensible uniquement par votre JRE, fraîchement installé.

La spécificité d'Eclipse IDE vient du fait que son architecture est totalement développée autour de la notion de **plug-in**. Cela signifie que toutes ses fonctionnalités sont développées en tant que plug-ins. Pour faire court, si vous voulez ajouter des fonctionnalités à Eclipse, vous devez :

3. Java 2 Enterprise Edition.
4. Java 2 Micro Edition.
5. Operating System = système d'exploitation.

– télécharger le plug-in correspondant ;
– copier les fichiers spécifiés dans les répertoires spécifiés ;
– démarrer Eclipse, et ça y est !

Lorsque vous téléchargez un nouveau plug-in pour Eclipse, celui-ci se présente souvent comme un dossier contenant généralement deux sous-dossiers : un dossier « plugins » et un dossier « features ». Ces dossiers existent aussi dans le répertoire d'Eclipse. Il vous faut donc copier le contenu des dossiers de votre plug-in vers le dossier correspondant dans Eclipse (`plugins` dans `plugins`, et `features` dans `features`).

Vous devez maintenant avoir une archive contenant Eclipse. Décompressez-la où vous voulez, puis entrez dans ce dossier (figure 1.5). Cela fait, lancez Eclipse.

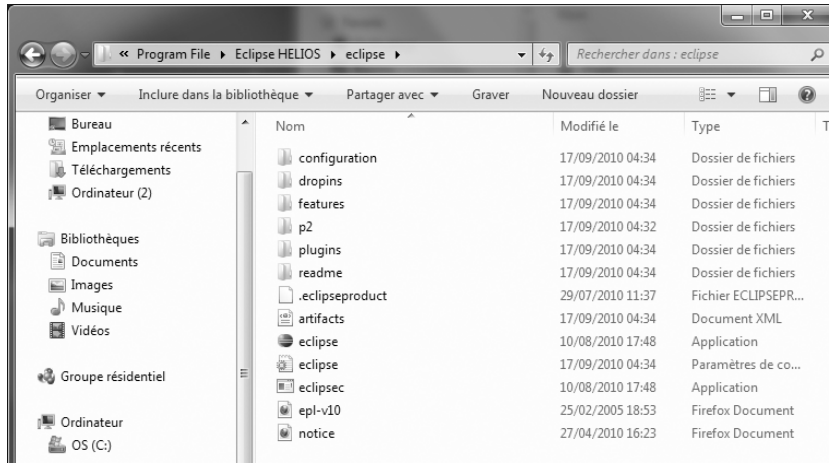

FIGURE 1.5 – Contenu du dossier Eclipse

Ici (figure 1.6), Eclipse vous demande dans quel dossier vous souhaitez enregistrer vos projets ; sachez que rien ne vous empêche de spécifier un autre dossier que celui proposé par défaut.

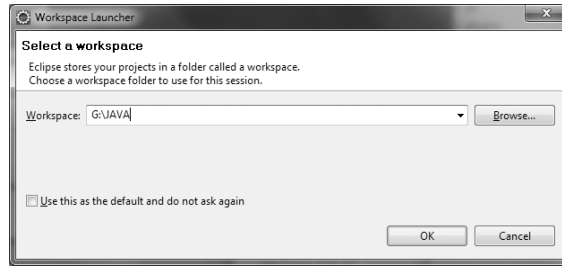

FIGURE 1.6 – Première fenêtre Eclipse

Une fois cette étape effectuée, vous arrivez sur la page d'accueil d'Eclipse. Si vous avez envie d'y jeter un œil, allez-y.

Présentation rapide de l'interface

Je vais maintenant vous faire faire un tour rapide de l'interface d'Eclipse.

Le menu « `File` » (figure 1.7)

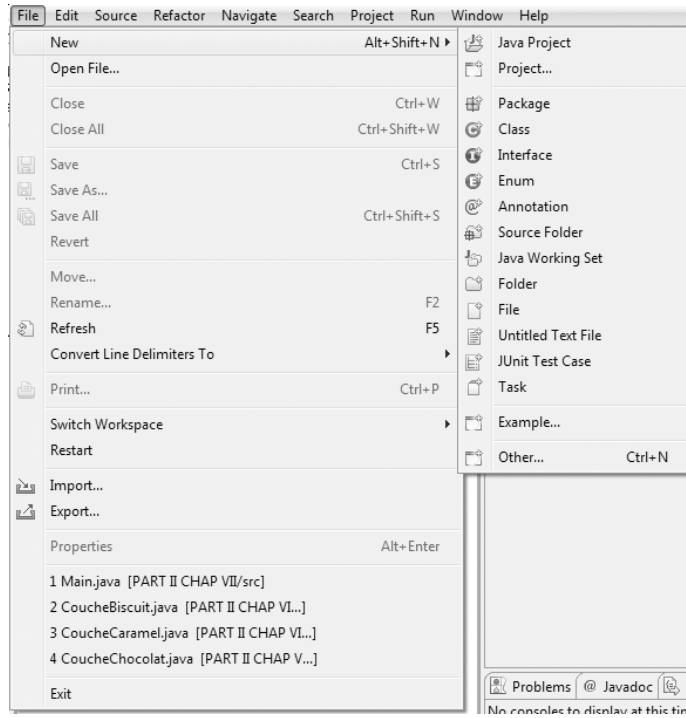

FIGURE 1.7 – Menu « File »

C'est ici que nous pourrons créer de nouveaux projets Java, les enregistrer et les exporter le cas échéant. Les raccourcis à retenir sont :

- ALT + SHIFT + N : nouveau projet ;
- CTRL + S : enregistrer le fichier où l'on est positionné ;
- CTRL + SHIFT + S : tout sauvegarder ;
- CTRL + W : fermer le fichier où l'on est positionné ;
- CTRL + SHIFT + W : fermer tous les fichiers ouverts.

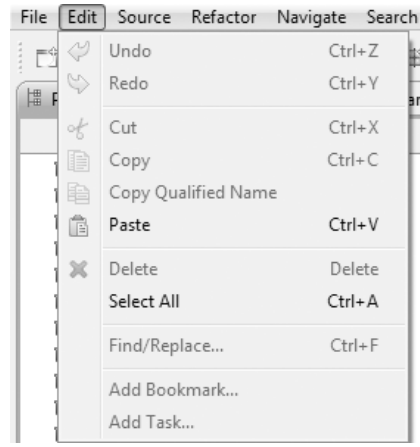

FIGURE 1.8 – Menu « Edit »

Le menu « `Edit` » (figure 1.8)

Dans ce menu, nous pourrons utiliser les commandes « `copier` », « `coller` », etc. Ici, les raccourcis à retenir sont :

- `CTRL + C` : copier la sélection ;
- `CTRL + X` : couper la sélection ;
- `CTRL + V` : coller la sélection ;
- `CTRL + A` : tout sélectionner ;
- `CTRL + F` : chercher-remplacer.

Le menu « `Window` » (figure 1.9)

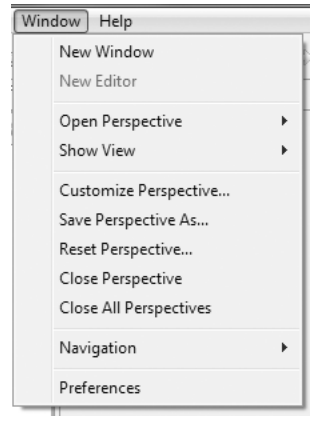

FIGURE 1.9 – Menu « Window »

Dans celui-ci, nous pourrons configurer Eclipse selon nos besoins.

La barre d'outils (figure 1.10)

FIGURE 1.10 – Barre d'outils

Nous avons dans l'ordre :

1. nouveau général. Cliquer sur ce bouton revient à faire « `Fichier / Nouveau` » ;
2. enregistrer. Revient à faire `CTRL + S` ;
3. imprimer ;
4. exécuter la classe ou le projet spécifié. Nous verrons ceci plus en détail ;
5. créer un nouveau projet. Revient à faire « `Fichier / Nouveau / Java Project` » ;
6. créer une nouvelle classe, c'est-à-dire en fait un nouveau fichier. Revient à faire « `Fichier / Nouveau / Classe` ».

Maintenant, je vais vous demander de créer un nouveau projet Java (figures 1.11 et 1.12).

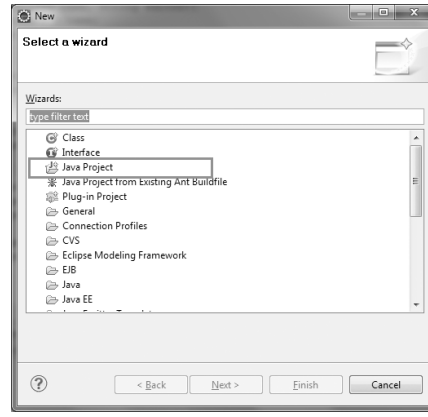

FIGURE 1.11 – Création de projet Java - étape 1

Renseignez le nom de votre projet comme je l'ai fait (encadré 1). Vous pouvez aussi voir où sera enregistré ce projet (encadré 2). Un peu plus compliqué, maintenant : vous avez donc un environnement Java sur votre machine, mais dans le cas où vous en auriez plusieurs, vous pouvez aussi spécifier à Eclipse quel JRE [6] utiliser pour ce projet.

Vous devriez avoir un nouveau projet dans la fenêtre de gauche (figure 1.13).

6. Vous pourrez changer ceci à tout moment dans Eclipse en allant dans « `Window / Preferences` », en dépliant l'arbre « `Java` » dans la fenêtre et en choisissant « `Installed JRE` ».

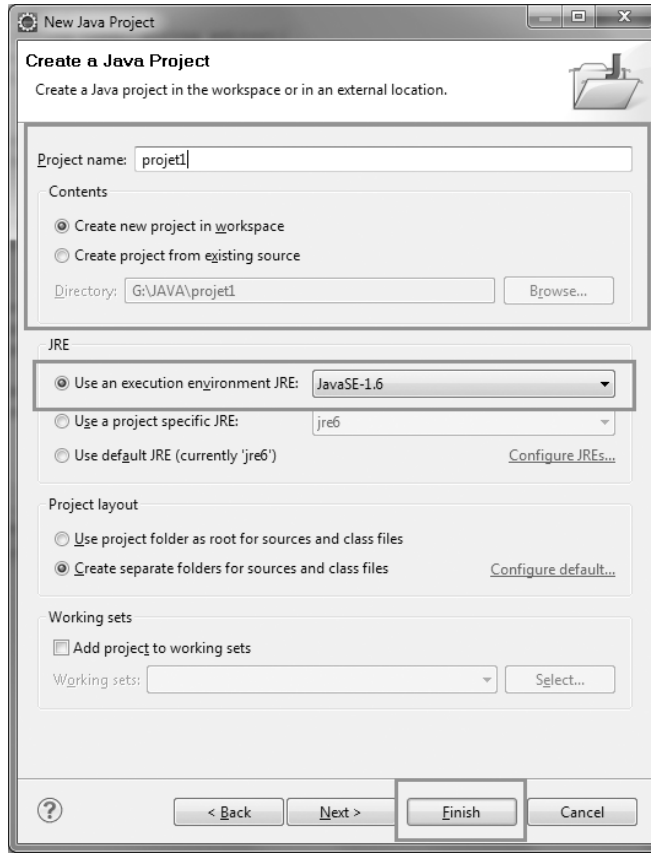

FIGURE 1.12 – Création de projet Java - étape 2

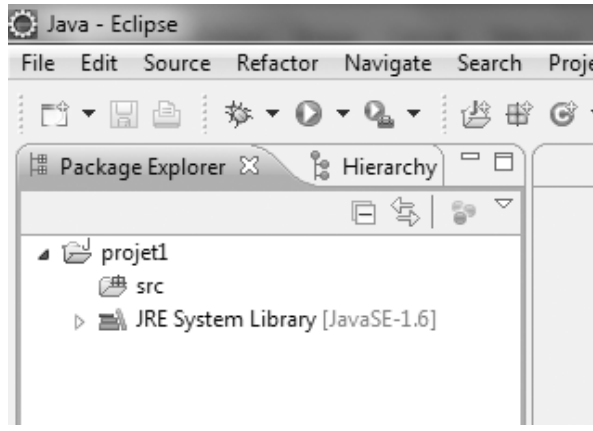

FIGURE 1.13 – Explorateur de projet

Pour boucler la boucle, ajoutons dès maintenant une nouvelle classe dans ce projet comme nous avons appris à le faire plus tôt.
Voici la fenêtre sur laquelle vous devriez tomber : figure 1.14.

Une classe est un ensemble de codes contenant plusieurs instructions que doit effectuer votre programme. Ne vous attardez pas trop sur ce terme, nous aurons l'occasion d'y revenir.

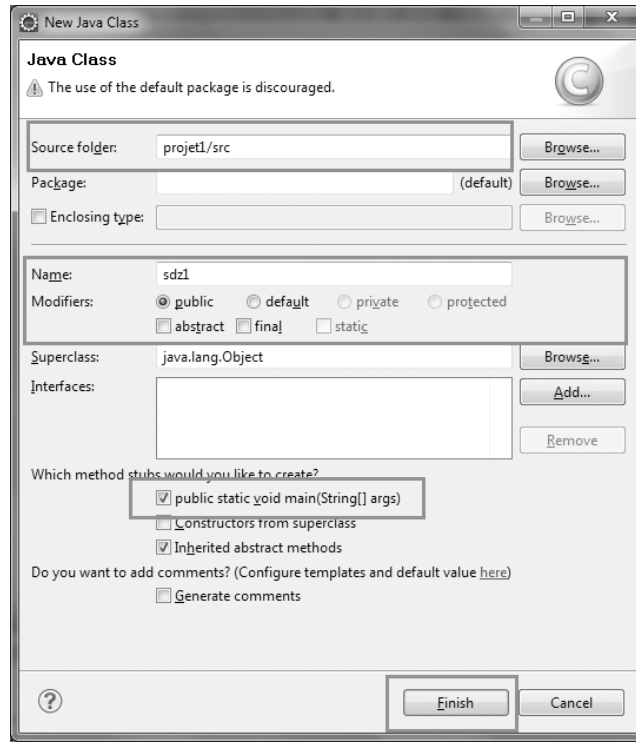

FIGURE 1.14 – Création d'une classe

Dans l'encadré 1, nous pouvons voir où seront enregistrés nos fichiers Java. Dans l'encadré 2, nommez votre classe Java ; moi, j'ai choisi **sdz1**. Dans l'encadré 3, Eclipse vous demande si cette classe a quelque chose de particulier. Eh bien oui ! Cochez « `public static void main(String[] args)` [7] », puis cliquez sur « `Finish` ».

Avant de commencer à coder, nous allons explorer l'espace de travail (figure 1.15).

Dans l'encadré de gauche, vous trouverez le dossier de votre projet ainsi que son contenu. Ici, vous pourrez gérer votre projet comme bon vous semble (ajout, suppression...).

Dans l'encadré positionné au centre, je pense que vous avez deviné : c'est ici que nous

7. Nous reviendrons plus tard sur ce point.

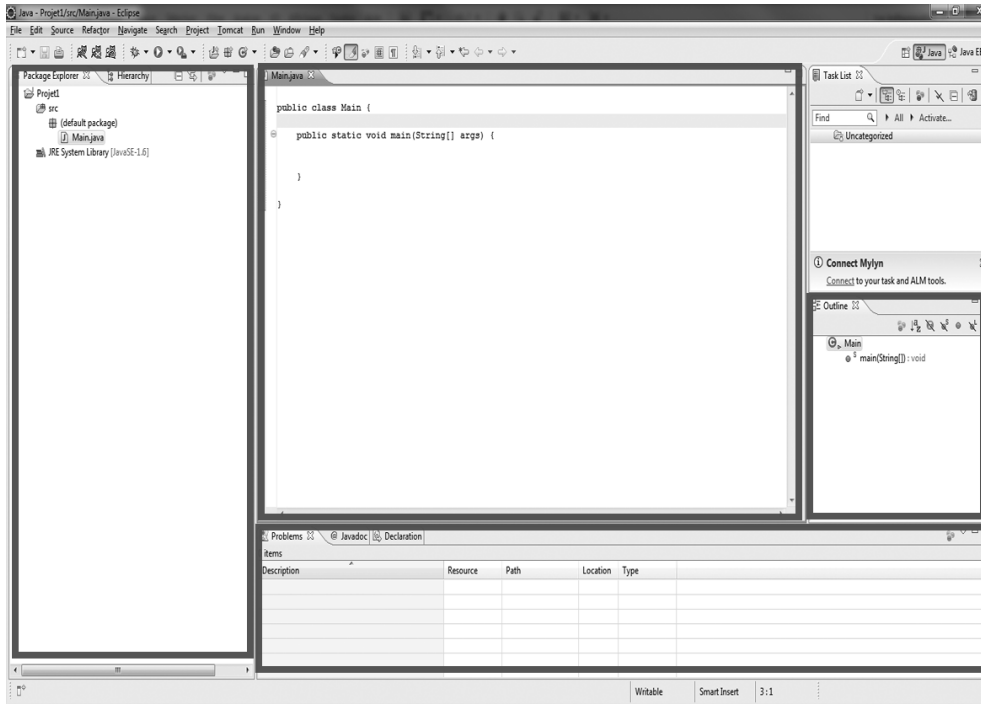

FIGURE 1.15 – Fenêtre principale

allons écrire nos codes source.

Dans l'encadré du bas, c'est là que vous verrez apparaître le contenu de vos programmes... ainsi que les erreurs éventuelles !

Et pour finir, c'est dans l'encadré de droite, dès que nous aurons appris à coder nos propres fonctions et nos objets, que la liste des méthodes et des variables sera affichée.

Votre premier programme

Comme je vous l'ai maintes fois répété, les programmes Java sont, avant d'être utilisés par la machine virtuelle, précompilés en byte code (par votre IDE ou à la main). Ce byte code n'est compréhensible que par une JVM, et c'est celle-ci qui va faire le lien entre ce code et votre machine.

Vous aviez sûrement remarqué que sur la page de téléchargement du JRE, plusieurs liens étaient disponibles :

– un lien pour Windows ;
– un lien pour Mac ;
– un lien pour Linux.

Ceci, car la machine virtuelle Java se présente différemment selon qu'on se trouve sous Mac, sous Linux ou encore sous Windows. Par contre, le byte code, lui, reste le même quel que soit l'environnement avec lequel a été développé et précompilé votre programme Java.

Conséquence directe : quel que soit l'OS sous lequel a été codé un programme Java, n'importe quelle machine pourra l'exécuter si elle dispose d'une JVM !

Tu n'arrêtes pas de nous rabâcher byte code par-ci, byte code par-là... Mais c'est quoi, au juste ?

Eh bien, un byte code[8] n'est rien d'autre qu'un code intermédiaire entre votre code Java et le code machine. Ce code particulier se trouve dans les fichiers précompilés de vos programmes ; en Java, un fichier source a pour extension `.java` et un fichier précompilé a l'extension `.class` : c'est dans ce dernier que vous trouverez du byte code. Je vous invite à examiner un fichier `.class` à la fin de cette partie (vous en aurez au moins un), mais je vous préviens, c'est illisible !

Par contre, vos fichiers `.java` sont de simples fichiers texte dont l'extension a été changée. Vous pouvez donc les ouvrir, les créer ou encore les mettre à jour avec le Bloc-notes de Windows, par exemple. Cela implique que, si vous le souhaitez, vous pouvez écrire des programmes Java avec le Bloc-notes ou encore avec Notepad++.

8. Il existe plusieurs types de byte code, mais nous parlons ici de celui créé par Java.

Reprenons. Vous devez savoir que **tous les programmes Java sont composés d'au moins une classe**.

Cette classe doit contenir une méthode appelée `main` : ce sera le point de démarrage de notre programme.

Une méthode est une suite d'instructions à exécuter. C'est un morceau de logique de notre programme. Une méthode contient :

- un en-tête : celui-ci va être en quelque sorte la carte d'identité de la méthode ;
- un corps : le contenu de la méthode, délimité par des accolades ;
- une valeur de retour : le résultat que la méthode va retourner.

Vous verrez un peu plus tard qu'un programme n'est qu'une multitude de classes qui s'utilisent l'une l'autre. Mais pour le moment, nous n'allons travailler qu'avec une seule classe.

Je vous avais demandé de créer un projet Java ; ouvrez-le (figure 1.16).

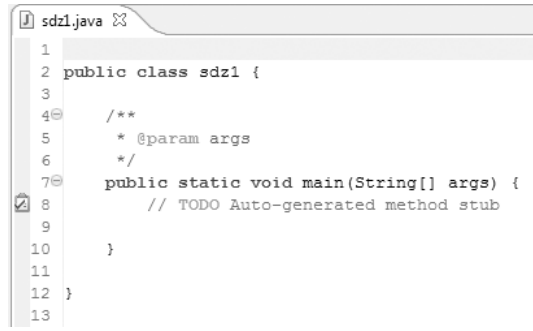

FIGURE 1.16 – Méthode principale

Vous voyez la fameuse classe dont je vous parlais ? Ici, elle s'appelle « sdz1 ». Vous pouvez voir que le mot `class` est précédé du mot `public`, dont nous verrons la signification lorsque nous programmerons des objets.

Pour le moment, ce que vous devez retenir, c'est que votre classe est définie par un mot clé (`class`), qu'elle a un nom (ici, sdz1) et que son contenu est délimité par des accolades ({}).

Nous écrirons nos codes sources entre la méthode `main`. La syntaxe de cette méthode est toujours la même :

```
public static void main(String[] args){
//Contenu de votre classe
}
```

Ce sera entre les accolades de la méthode `main` que nous écrirons nos codes source.

Excuse-nous, mais... pourquoi as-tu écrit « //Contenu de votre classe » et pas « Contenu de votre classe » ?

Bonne question ! Je vous ai dit plus haut que votre programme Java, avant de pouvoir être exécuté, doit être précompilé en byte code. Eh bien, la possibilité de forcer le compilateur à ignorer certaines instructions existe ! C'est ce qu'on appelle des **commentaires**, et deux syntaxes sont disponibles pour commenter son texte.

– Il y a les commentaires unilignes : introduits par les symboles //, ils mettent tout ce qui les suit en commentaire, du moment que le texte se trouve sur la même ligne que les //.

```
public static void main(String[] args){
//Un commentaire
//Un autre
//Encore un autre
Ceci n'est pas un commentaire !
}
```

– Il y a les commentaires multilignes : ils sont introduits par les symboles /* et se terminent par les symboles */.

```
public static void main(String[] args){

/*
Un commentaire
Un autre
Encore un autre
*/
Ceci n'est pas un commentaire !
}
```

D'accord, mais ça sert à quoi ?

C'est simple : au début, vous ne ferez que de très petits programmes. Mais dès que vous aurez pris de la bouteille, leurs tailles et le nombre de classes qui les composeront vont augmenter. Vous serez contents de trouver quelques lignes de commentaires au début de votre classe pour vous dire à quoi elle sert, ou encore des commentaires dans une méthode qui effectue des choses compliquées afin de savoir où vous en êtes dans vos traitements...

Il existe en fait une troisième syntaxe, mais elle a une utilité particulière. Elle permettra de générer une documentation pour votre programme : une Javadoc (Java Documenta-

tion). Je n'en parlerai que très peu, et pas dans ce chapitre. Nous verrons cela lorsque nous programmerons des objets, mais pour les curieux, je vous conseille le très bon cours de dworkin sur ce sujet disponible sur le Site du Zéro.

▷ Présentation de la Javadoc
Code web : 478278

À partir de maintenant et jusqu'à ce que nous programmions des interfaces graphiques, nous allons faire ce qu'on appelle des programmes procéduraux. Cela signifie que le programme s'exécutera de façon procédurale, c'est-à-dire qui s'effectue de haut en bas, une ligne après l'autre. Bien sûr, il y a des instructions qui permettent de répéter des morceaux de code, mais le programme en lui-même se terminera une fois parvenu à la fin du code. Cela vient en opposition à la programmation événementielle (ou graphique) qui, elle, est basée sur des événements (clic de souris, choix dans un menu...).

Hello World

Maintenant, essayons de taper le code suivant :

```
public static void main(String[] args){
    System.out.print("Hello World !");
}
```

N'oubliez surtout pas le " ; " à la fin de la ligne ! **Toutes les instructions en Java sont suivies d'un point-virgule.**

Une fois que vous avez saisi cette ligne de code dans votre méthode `main`, il vous faut lancer le programme. Si vous vous souvenez bien de la présentation faite précédemment, vous devez cliquer sur la flèche blanche dans un rond vert (figure 1.17).

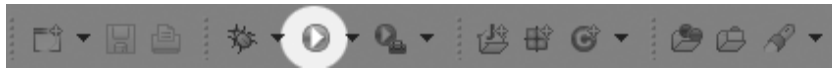

FIGURE 1.17 – Bouton de lancement du programme

Si vous regardez dans votre console, dans la fenêtre du bas sous Eclipse, vous devriez voir la figure 1.18.

Expliquons un peu cette ligne de code. Littéralement, elle signifie « la méthode `print()` va écrire **Hello World !** en utilisant l'objet `out` de la classe `System` ».

- `System` : ceci correspond à l'appel d'une classe qui se nomme « `System` ». C'est une classe utilitaire qui permet surtout d'utiliser l'entrée et la sortie standard, c'est-à-dire la saisie clavier et l'affichage à l'écran.
- `out` : objet de la classe `System` qui gère la sortie standard.

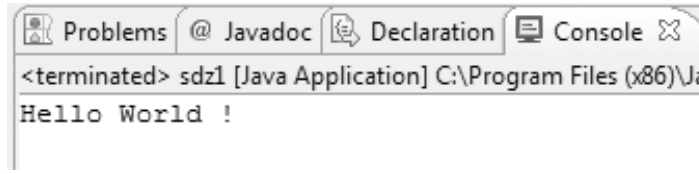

FIGURE 1.18 – Console d'Eclipse

– `print` : méthode qui écrit dans la console le texte passé en paramètre.

Si vous mettez plusieurs `System.out.print`, voici ce qui se passe. Prenons ce code :

```
System.out.print("Hello World !");
System.out.print("My name is");
System.out.print("Cysboy");
```

Lorsque vous l'exécutez, vous devriez voir des chaînes de caractères qui se suivent sans saut de ligne. Autrement dit, ceci s'affichera dans votre console :

```
Hello World !My name isCysboy
```

Je me doute que vous souhaiteriez insérer un retour à la ligne pour que votre texte soit plus lisible... Pour cela, vous avez plusieurs solutions :

– soit vous utilisez un caractère d'échappement, ici `\n` ;
– soit vous utilisez la méthode `println()` à la place de la méthode `print()`.

Donc, si nous reprenons notre code précédent et que nous appliquons cela, voici ce que ça donnerait :

```
System.out.print("Hello World ! \n");
System.out.println("My name is");
System.out.println("\nCysboy");
```

Le résultat :

```
Hello World !
My name is

Cysboy
```

Vous pouvez voir que :

– lorsque vous utilisez le caractère d'échappement `\n`, quelle que soit la méthode appelée, celle-ci ajoute immédiatement un retour à la ligne à son emplacement ;
– lorsque vous utilisez la méthode `println()`, celle-ci ajoute automatiquement un retour à la ligne à la fin de la chaîne passée en paramètre ;
– un caractère d'échappement peut être mis dans la méthode `println()`.

J'en profite au passage pour vous mentionner deux autres caractères d'échappement :

– `\r` va insérer un retour chariot, parfois utilisé aussi pour les retours à la ligne ;
– `\t` va faire une tabulation.

Vous avez sûrement remarqué que la chaîne de caractères que l'on affiche est entourée de "<chaîne>". En Java, les guillemets doubles[9] sont des délimiteurs de chaînes de caractères ! Si vous voulez afficher un guillemet double dans la sortie standard, vous devrez « l'échapper[10] » avec un \, ce qui donnerait : `System.out.println("Coucou mon \"chou\"! ");`.

Je vous propose maintenant de passer un peu de temps sur la compilation de vos programmes en ligne de commande. Cette section n'est pas obligatoire, loin de là, mais elle ne peut être qu'enrichissante.

Compilation en ligne de commande (Windows)

Bienvenue donc aux plus curieux ! Avant de vous apprendre à compiler et à exécuter un programme en ligne de commande, il va vous falloir le JDK (**J**ava SE **D**evelopment **K**it). C'est avec celui-ci que nous aurons de quoi compiler nos programmes. Le nécessaire à l'exécution des programmes est dans le JRE... mais il est également inclus dans le JDK.

Je vous invite donc à retourner sur le site d'Oracle et à télécharger ce dernier. Une fois cette opération effectuée, il est conseillé de mettre à jour votre variable d'environnement `%PATH%`.

Euh... quoi ?

Votre **variable d'environnement**. C'est grâce à elle que Windows trouve des exécutables sans qu'il soit nécessaire de lui spécifier le chemin d'accès complet. Vous — enfin, Windows — en a plusieurs, mais nous ne nous intéresserons qu'à une seule. En gros, cette variable contient le chemin d'accès à certains programmes.

Par exemple, si vous spécifiez le chemin d'accès à un programme X dans votre variable d'environnement et que, par un malheureux hasard, vous n'avez plus aucun raccourci vers X : vous l'avez définitivement perdu dans les méandres de votre PC. Eh bien vous pourrez le lancer en faisant « `Démarrer` → `Exécuter` » et en tapant la commande « `X.exe` » (en partant du principe que le nom de l'exécutable est `X.exe`).

D'accord, mais comment fait-on ? Et pourquoi doit-on faire ça pour le JDK ?

9. Il n'est pas rare de croiser le terme anglais *quote* pour désigner les guillemets droits. Cela fait en quelque sorte partie du jargon du programmeur.
10. Terme désignant le fait de désactiver : ici, désactiver la fonction du caractère « " ».

J'y arrive. Une fois votre JDK installé, ouvrez le répertoire **bin** de celui-ci, ainsi que celui de votre JRE. Nous allons nous attarder sur deux fichiers.

Dans le répertoire `bin` de votre JRE, vous devez avoir un fichier nommé **java.exe**. Fichier que vous retrouvez aussi dans le répertoire `bin` de votre JDK. C'est grâce à ce fichier que votre ordinateur peut lancer vos programmes par le biais de la JVM. Le deuxième ne se trouve que dans le répertoire `bin` de votre JDK, il s'agit de **javac.exe** [11]. C'est celui-ci qui va précompiler vos programmes Java en byte code.

Alors, pourquoi mettre à jour la variable d'environnement pour le JDK ? Eh bien, compiler et exécuter en ligne de commande revient à utiliser ces deux fichiers en leur précisant où se trouvent les fichiers à traiter. Cela veut dire que si l'on ne met pas à jour la variable d'environnement de Windows, il nous faudrait :

- ouvrir l'invite de commande ;
- se positionner dans le répertoire `bin` de notre JDK ;
- appeler la commande souhaitée ;
- préciser le chemin du fichier `.java` ;
- renseigner le nom du fichier.

Avec notre variable d'environnement mise à jour, nous n'aurons plus qu'à :

- nous positionner dans le dossier de notre programme ;
- appeler la commande ;
- renseigner le nom du fichier Java.

Allez dans le « `Panneau de configuration` » de votre PC ; de là, cliquez sur l'icône « `Système` » ; choisissez l'onglet « `Avancé` » et vous devriez voir en bas un bouton nommé « `Variables d'environnement` » : cliquez dessus. Une nouvelle fenêtre s'ouvre. Dans la partie inférieure intitulée « `Variables système` », cherchez la variable `Path`. Une fois sélectionnée, cliquez sur « `Modifier` ». Encore une fois, une fenêtre, plus petite celle-ci, s'ouvre devant vous. Elle contient le nom de la variable et sa valeur.

Ne changez pas son nom et n'effacez pas son contenu ! Nous allons juste ajouter un chemin d'accès.

Pour ce faire, allez jusqu'au bout de la valeur de la variable, ajoutez-y un point-virgule (`;`) s'il n'y en a pas, et ajoutez alors le chemin d'accès au répertoire `bin` de votre JDK, en terminant celui-ci par un point-virgule !

Chez moi, ça donne ceci : « `C:\Sun\SDK\jdk\bin` ».

Auparavant, ma variable d'environnement contenait, avant mon ajout :

```
%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;
```

Et maintenant :

```
%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;C:\Sun\SDK\jdk\bin;
```

11. **Java c**ompiler.

Validez les changements : vous êtes maintenant prêts à compiler en ligne de commande.

Pour bien faire, allez dans le répertoire de votre premier programme et effacez le `.class`. Ensuite, faites « `Démarrer > Exécuter` [12] » et tapez « `cmd` ».

Pour rappel, dans l'invite de commande, on se déplace de dossier en dossier grâce à l'instruction `cd`. `cd <nom du dossier enfant>` : pour aller dans un dossier contenu dans celui dans lequel nous nous trouvons. `cd ..` : pour remonter d'un dossier dans la hiérarchie.

Par exemple, lorsque j'ouvre la console, je me trouve dans le dossier `C:\toto\titi` et mon application se trouve dans le dossier `C:\sdz`, je fais donc :

```
cd ..
cd ..
cd sdz
```

Après de la première instruction, je me retrouve dans le dossier `C:\toto`. Grâce à la deuxième instruction, j'arrive à la racine de mon disque. Via la troisième instruction, je me retrouve dans le dossier `C:\sdz`. Nous sommes maintenant dans le dossier contenant notre fichier Java !

Cela dit, nous pouvions condenser cela en :

```
cd ../../sdz
```

Maintenant, vous pouvez créer votre fichier `.class` en exécutant la commande suivante :

```
javac <nomDeFichier.java>
```

Si, dans votre dossier, vous avez un fichier `test.java`, compilez-le en faisant : `javac test.java`. Et si vous n'avez aucun message d'erreur, vous pouvez vérifier que le fichier `test.class` est présent en utilisant l'instruction `dir` qui liste le contenu d'un répertoire.

Cette étape franchie, vous pouvez lancer votre programme Java en faisant ce qui suit :

```
java <nomFichierClassSansExtension>
```

Ce qui nous donne : `java test`. Et normalement, le résultat de votre programme Java s'affiche sous vos yeux ébahis !

Attention : il ne faut pas mettre l'extension du fichier pour le lancer, mais il faut la mettre pour le compiler.

Donc voilà : vous avez compilé et exécuté un programme Java en ligne de commande... Vous avez pu voir qu'il n'y a rien de vraiment compliqué et, qui sait, vous en aurez peut-être besoin un jour.

12. Ou encore touche Windows + R.

En résumé

- La JVM est le cœur de Java.
- Elle fait fonctionner vos programmes Java, précompilés en byte code.
- Les fichiers contenant le code source de vos programmes Java ont l'extension `.java`.
- Les fichiers précompilés correspondant à vos codes source Java ont l'extension `.class`.
- Le byte code est un code intermédiaire entre celui de votre programme et celui que votre machine peut comprendre.
- Un programme Java, codé sous Windows, peut être précompilé sous Mac et enfin exécuté sous Linux.
- Votre machine NE PEUT PAS comprendre le byte code, elle a besoin de la JVM.
- Tous les programmes Java sont composés d'au moins une classe.
- Le point de départ de tout programme Java est la méthode `public static void main(String[] args)`.
- On peut afficher des messages dans la console grâce à ces instructions :
  - `System.out.println`, qui affiche un message avec un saut de ligne à la fin ;
  - `System.out.print`, qui affiche un message sans saut de ligne.