Cyrille Herby

APPRENEZ À PROGRAMMER EN

JAVA

3e édition

OPENCLASSROOMS

EYROLLES

APPRENEZ À PROGRAMMER EN

JAVA

3e édition

Vous aimeriez apprendre à programmer en Java, mais vous débutez dans la programmation ? Pas de panique ! Vous tenez dans vos mains un livre conçu pour les débutants qui souhaitent se former à Java, le langage de programmation incontournable des professionnels ! De quoi permettre d'apprendre pas à pas à développer vos premiers programmes.

QU'ALLEZ-VOUS APPRENDRE ?

Bien commencer en Java

- Installer les outils de développement
- Les variables et les opérateurs
- Lire les entrées clavier
- Les conditions
- Les boucles

Java orienté objet

- Votre première classe
- L'héritage
- Les classes abstraites et les interfaces
- Java 8 ou la révolution des interfaces
- Les exceptions
- Les collections d'objets
- La généricité en Java
- Les flux d'entrées sorties
- Java et la réflexivité
- Classes anonymes, interfaces fonctionnelles, lambdas et références de méthode
- Manipuler vos données avec les streams
- La nouvelle API de gestion des dates de Java 8
- Une JVM modulaire avec Java 9

Java et la programmation événementielle

- Votre première fenêtre
- Positionner des boutons
- Interagir avec des boutons
- Les champs de formulaire
- Les menus et boîtes de dialogue
- Conteneurs, sliders et barres de progression
- Les arbres et leur structure
- Les interfaces de tableaux
- Mieux structurer son code : le pattern MVC
- Le drag'n drop
- Mieux gérer les interactions avec les composants

Initiation à Java FX

- Introduction et installation des outils
- Lier un modèle à votre vue
- Interagir avec vos composants
- Java FX a du style !

Interaction avec les bases de données

- JDBC : la porte d'accès aux bases de données
- Fouiller dans sa base de données
- Limiter le nombre de connexions

À PROPOS DE L'AUTEUR

Auditeur en sécurité, Cyrille Herby travaille sur des projets Java depuis plusieurs années. Il a notamment été administrateur système et réseau, puis responsable de la sécurité des systèmes d'information au sein du groupe Cordon Electronics. Il a débuté dans la programmation en découvrant OpenClassrooms il y a plusieurs années déjà. Désormais, il y rédige à son tour des cours pour montrer qu'il est possible de s'initier à la programmation et ses concepts sans avoir un dictionnaire à la main… et surtout, sans migraine !

L'ESPRIT D'OPENCLASSROOMS

Des cours ouverts, riches et vivants, conçus pour tous les niveaux et accessibles à tous gratuitement sur notre plate-forme d'e-éducation : www.openclassrooms.com. Vous y vivrez une véritable expérience communautaire de l'apprentissage, permettant à chacun d'apprendre avec le soutien et l'aide des autres étudiants sur les forums. Vous profiterez des cours disponibles partout, tout le temps.

www.editions-eyrolles.com

APPRENEZ À PROGRAMMER EN

JAVA

DANS LA MÊME COLLECTION

É. Lalitte. – **Apprenez le fonctionnement des réseaux TCP/IP.**
N° 67477, 3e édition, 2018, 304 pages.

M. Nebra. – **Concevez votre site web avec PHP et MySQL.**
N° 67475, 3e édition, 2017, 392 pages.

M. Nebra. – **Réalisez votre site web avec HTML 5 et CSS 3.**
N° 67476, 2e édition, 2017, 362 pages.

V. Thuillier. – **Programmez en orienté objet en PHP.**
N° 14472, 2e édition, 2017, 474 pages.

J. Pardanaud, S. de la Marck. – **Découvrez le langage JavaScript.**
N° 14399, 2017, 478 pages.

A. Bacco. – **Développez votre site web avec le framework Symfony3.**
N° 14403, 2016, 536 pages.

M. Chavelli. – **Découvrez le framework PHP Laravel.**
N° 14398, 2016, 336 pages.

R. De Visscher. – **Découvrez le langage Swift.**
N° 14397, 2016, 128 pages.

M. Lorant. – **Développez votre site web avec le framework Django.**
N° 21626, 2015, 285 pages.

M. Nebra, M. Schaller. – **Programmez avec le langage C++.**
N° 21622, 2015, 674 pages.

SUR LE MÊME THÈME

C. Delannoy. – **Programmer en Java.**
N° 67536, 10e édition, 2017, 984 pages.

A. Tasso. – **Le livre de Java premier langage.**
N° 67486, 12e édition, 2017, 616 pages.

Cyrille Herby

APPRENEZ À PROGRAMMER EN

JAVA

3e édition

EYROLLES

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

ISBN : 978-2-212-67521-4

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 (maintenant OpenClassrooms). 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 permettait 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 mon 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 !

Structure de l'ouvrage

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 avant 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 pour cela. Contrairement à beaucoup de livres techniques où il est courant de lire en diagonale et de sauter certains chapitres, il est ici fortement recommandé de suivre l'ordre du livre, à moins que vous ne soyez déjà, au moins un peu, expérimenté.

Pratiquez en même temps

Pratiquez régulièrement. N'attendez pas d'avoir fini de lire ce livre pour allumer votre ordinateur.

Les compléments web

Pour télécharger le code source des exemples de cet ouvrage, veuillez-vous rendre à cette adresse : https://www.editions-eyrolles.com/dl/0067521.

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 OpenClassrooms. Un merci spécial à toi qui me prends dans tes bras lorsque ça ne va pas, qui m'embrasse 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.

Mes deux filles Jaana et Madie qui me donnent tant de joie et de bonheur au quotidien.

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) et les relecteurs m'ayant accompagné dans la correction de cet ouvrage.

Mathieu Nebra (alias M@teo21), père fondateur d'OpenClassrooms, qui m'a fait confiance, soutenu dans mes démarches et qui m'a donné de précieux conseils.

Jessica Mautref, Marine Gallois et Laurène Gibaud qui m'ont accompagné chez OpenClassrooms pendant un bon nombre d'années et qui sont des perles de gentillesse et de compétences.

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… :-)

Table des matières

Première partie

Bien commencer en Java

1 Installer les outils de développement

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.

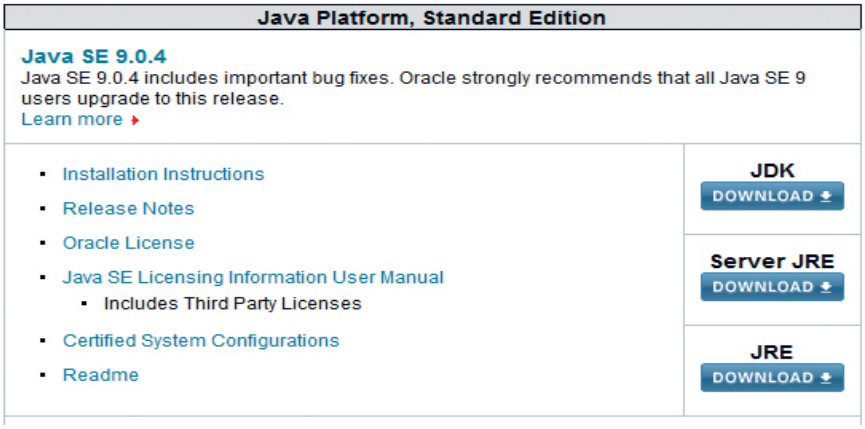
Afin de nous simplifier la vie, nous allons utiliser un outil de développement, ou **IDE** *(**I**ntegrated **D**evelopment **E**nvironment)*, pour nous aider à écrire nos futurs codes sources… Nous allons donc avoir besoin de différentes choses afin de pouvoir créer des programmes Java : la première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

Installer les outils nécessaires

JRE ou JDK

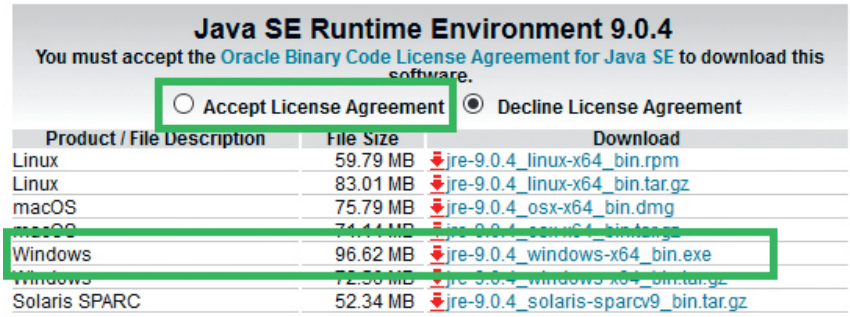
Commencez par télécharger l'environnement Java sur le site d'Oracle, comme le montre la figure page suivante. Choisissez la dernière version stable.

Vous avez sans doute remarqué qu'on vous propose de télécharger soit le JRE, soit le JDK *(**J**ava **D**evelopment **K**it)*. La différence entre ces deux environnements est indiquée sur le site d'Oracle, mais pour les personnes fâchées avec l'anglais, sachez que le JRE contient tout le nécessaire pour 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…



Encart de téléchargement

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 représentée à la figure suivante.



Choix du système d'exploitation

Cochez la case ***Accept License Agreement*** puis cliquez sur le lien correspondant à votre système d'exploitation (***x86*** pour un système 32 bits et ***x64*** pour un système 64 bits). Une fenêtre de téléchargement doit alors apparaître.

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** (*Java 2 Standard Edition*, celui qui nous intéresse dans cet ouvrage) : il permet de développer des applications dites « client lourd », par exemple Microsoft Word ou Excel, la suite OpenOffice.org… C'est ce que nous allons faire dans cet ouvrage.

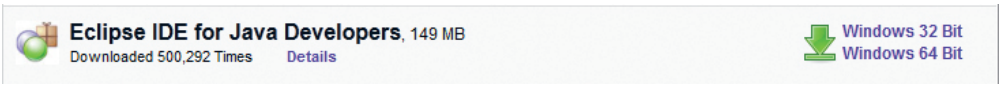
- **J2EE** *(Java 2 Enterprise Edition)* : il permet de développer des applications web en Java. On parle aussi de « clients légers ».
- **J2ME** *(Java 2 Micro Edition)* : il permet de développer des applications pour appareils mobiles, comme des téléphones portables, des PDA…

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 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 sur le site *https://www.eclipse.org/*. Une fois la page de téléchargement ouverte, choisissez ***Eclipse IDE for Java Developers***, en choisissant la version d'Eclipse correspondant à votre OS, comme indiqué à la figure suivante.



Version Windows 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.

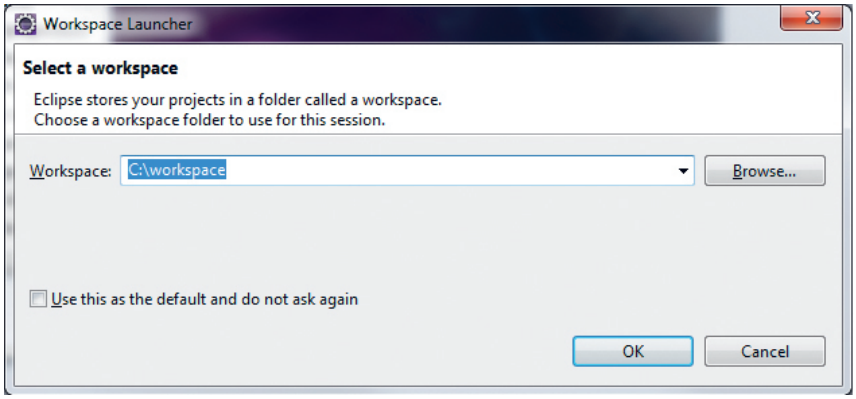
La spécificité d'Eclipse IDE vient du fait que son architecture est totalement développée autour de la notion de plug-in. Cela signifie que toutes ses fonctionnalités sont développées en tant que plug-ins. Pour faire court, si vous voulez ajouter des fonctionnalités à Eclipse, vous devez :

- télécharger le plug-in correspondant ;
- copier les fichiers dans les répertoires spécifiés ;
- démarrer Eclipse.

Et c'est tout !

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, entrez dans ce dossier et lancez Eclipse. Au démarrage, comme le montre la figure suivante, 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. 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 !



Vous devez indiquer où enregistrer vos projets.

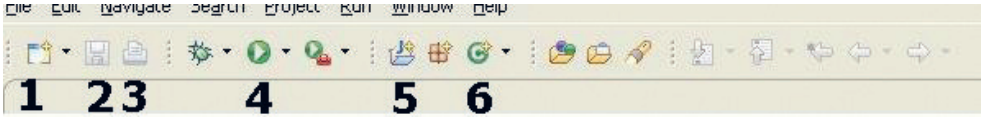
Présentation rapide de l'interface

Je vais maintenant vous présenter rapidement l'interface d'Eclipse. Voici les principaux menus :

- ***File*** : c'est ici que nous pourrons créer de nouveaux projets Java, les enregistrer et les exporter le cas échéant.
  Les raccourcis clavier 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.
- ***Edit*** : dans ce menu, nous pourrons utiliser les commandes ***Copier***, ***Coller***, etc.
- ***Window*** : ce menu nous permet de configurer Eclipse selon nos besoins.

La barre d'outils

La barre d'outils ressemble à celle de la figure ci-dessous.

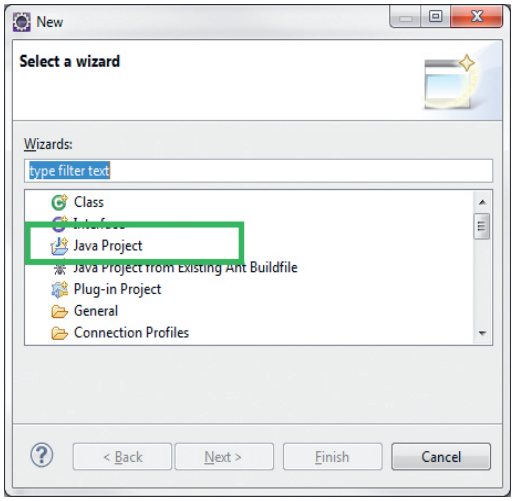


La barre d'outils d'Eclipse

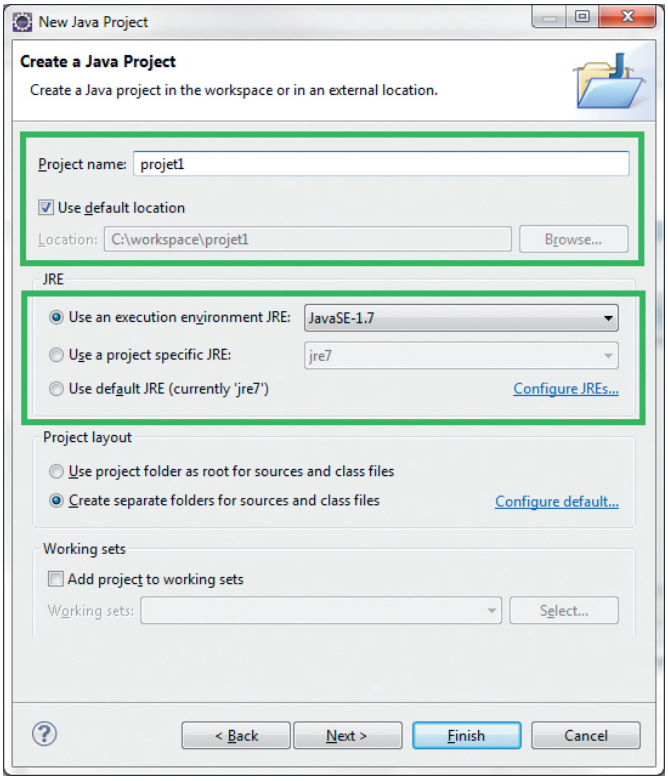
Voici les icônes de la barre d'outils, de gauche à droite :

1. ***Nouveau général*** : cliquer sur cette icône revient à sélectionner le menu ***Fichier>Nouveau*** ;
2. ***Enregistrer*** : cette icône à la même action que le raccourci ***Ctrl+S*** ;
3. ***Imprimer*** : ai-je besoin de préciser l'action de cette icône ?
4. ***Exécuter la classe ou le projet spécifié*** : nous verrons ceci plus en détail par la suite ;
5. ***Créer un nouveau projet*** : cliquer sur cette icône revient à sélectionner le menu ***Fichier>Nouveau>Java Project*** ;
6. ***Créer une nouvelle classe*** : cette icône permet de créer un nouveau fichier. Cela revient à sélectionner le menu ***Fichier>Nouveau>Classe***.

Maintenant, je vais vous demander de créer un nouveau projet Java, comme indiqué dans les figures suivantes.

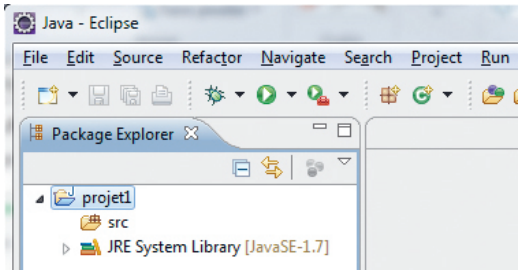


Création de projet Java – étape 1



Création de projet Java – étape 2

Renseignez le nom de votre projet comme je l'ai fait dans le premier encadré de la deuxième figure. Vous pouvez aussi voir où sera enregistré ce projet. Un peu plus compliqué maintenant : vous avez un seul environnement Java sur votre machine, mais si vous en aviez plusieurs, vous pourriez aussi spécifier à Eclipse quel JRE utiliser pour ce projet, comme sur le second encadré. Vous pourrez changer ceci à tout moment dans Eclipse via le menu ***Window>Preferences***, en dépliant l'arbre ***Java*** dans la fenêtre et en choisissant ***Installed JRE***.

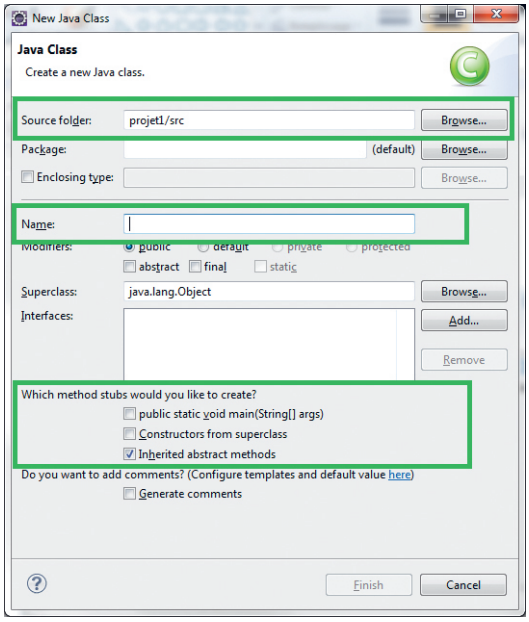


Explorateur de projets

Vous devriez avoir un nouveau projet dans la fenêtre de gauche, comme représenté à la figure suivante.

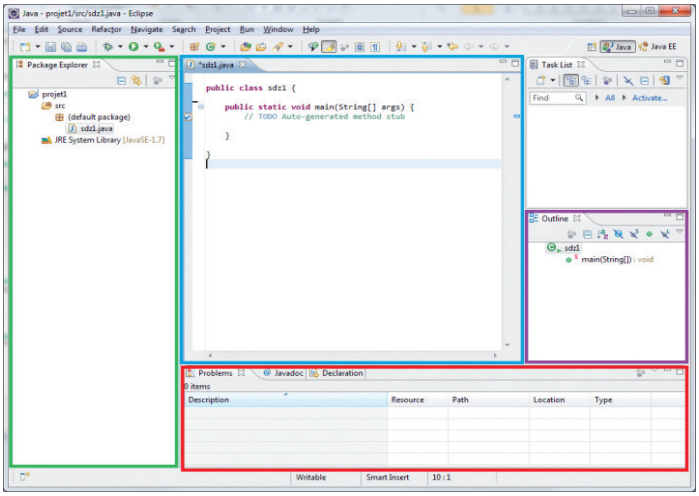
Pour boucler la boucle, ajoutons dès maintenant une nouvelle **classe** dans ce projet comme nous avons appris à le faire précédemment via la barre d'outils. La figure suivante représente la fenêtre que vous devriez obtenir.

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.



Création d'une classe

Dans le premier encadré de la figure, nous pouvons voir où seront enregistrés nos fichiers Java (paramètre ***Source folder***). Nommez votre classe Java grâce au champ ***Name***, ici j'ai choisi `sdz1`. Dans le troisième encadré, Eclipse vous demande si cette classe a quelque chose de particulier. Eh bien oui ! Cochez l'option ***public static void main(String[] args)*** (nous reviendrons plus tard sur ce point), puis cliquez sur le bouton ***Finish***. La fenêtre principale d'Eclipse s'ouvre alors comme à la figure page suivante.



Fenêtre principale d'Eclipse

Avant de commencer à coder, nous allons explorer l'espace de travail. Dans l'encadré de gauche (en vert sur la figure), vous trouverez le dossier de votre projet ainsi que son contenu. Vous pourrez ici gérer votre projet comme bon vous semble (ajout, suppression de données…). Pour l'encadré central (en bleu sur la figure), je pense que vous avez deviné : c'est ici que nous allons écrire nos codes sources. C'est dans l'encadré situé en bas (en rouge sur la figure) que vous verrez apparaître le contenu de vos programmes… ainsi que les erreurs éventuelles ! Et pour finir, c'est dans l'encadré de droite (en violet sur la figure), 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 indiqué précédemment, 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 le 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.

En effet, la machine virtuelle Java se présente différemment selon que l'on se trouve sous Mac OS, Linux ou encore Windows. En revanche, le byte code reste quant à lui le même quel que soit l'environnement avec lequel a été développé et précompilé votre programme Java. Par conséquent, 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 répéter byte code par-ci, byte code par-là... Mais c'est quoi, au juste ?

Eh bien, un byte code (il existe plusieurs types de byte codes, mais nous parlons ici de celui créé par Java) 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 porte 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 !

En revanche, 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++.

Reprenons. Vous devez savoir que **tous les programmes Java sont composés d'au moins une classe**. Elle doit contenir une méthode appelée `main` : ce sera le point de démarrage de notre programme.

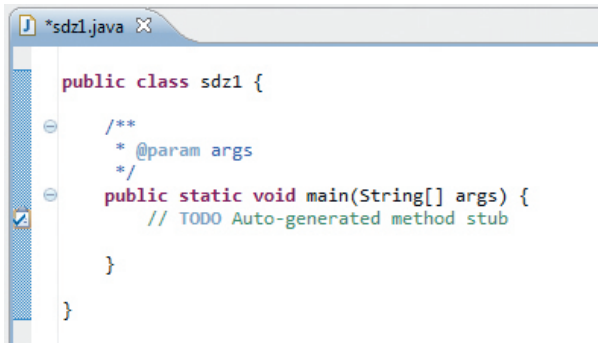
Une méthode est une suite d'instructions à exécuter. C'est un morceau de logique de notre programme. Une méthode contient :

- 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 les unes les autres. Mais pour le moment, nous n'allons travailler qu'avec une seule classe.

Je vous avais demandé précédemment de créer un projet Java, ouvrez-le. Vous voyez la fameuse classe dont je vous parlais ? Ici, elle s'appelle `sdz1` (figure suivante). 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 les accolades de la méthode `main`.



Méthode principale

La syntaxe de cette méthode est toujours la même :

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

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

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

- les commentaires unilignes : ils sont introduits par les caractères `//` et mettent tout ce qui les suit en commentaire, du moment que le texte se trouve sur la même ligne ;
- les commentaires multilignes : ils sont introduits par les caractères `/*` et se terminent par `*/`.

```
public static void main(String[] args){
   // Un commentaire
   // Un autre
   // Encore un autre

   /*
   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 content 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 (on l'appelle « Javadoc » pour *Java Documentation*). Nous aborderons ce sujet lorsque nous programmerons des objets mais pour les curieux, je vous conseille le très bon cours de dworkin sur ce sujet, disponible sur OpenClassrooms (*https://openclassrooms.com/courses/presentation-de-la-javadoc*).

À partir de maintenant, et jusqu'à ce que nous programmions des interfaces graphiques, nous allons faire ce que l'on appelle des « programmes procéduraux ». Cela signifie que le programme s'exécutera de façon procédurale, c'est-à-dire 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 est quant à elle basée sur des événements (clic de souris, choix dans un menu…).

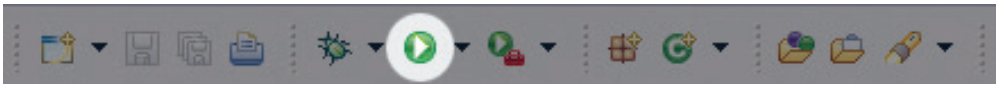
Hello World

Maintenant, essayons de taper le code suivant :

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

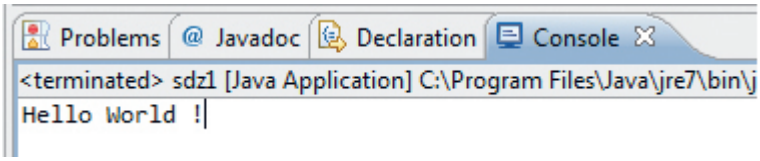
N'oubliez surtout pas le point-virgule (`;`) à la fin de la ligne ! Toutes les instructions en Java sont suivies d'un point-virgule.

Une fois que vous avez saisi cette ligne de code dans votre méthode `main`, il vous faut lancer le programme. Si vous vous souvenez de la présentation de la barre d'outils faite précédemment, vous devez cliquer sur l'icône représentant une flèche blanche dans un rond vert (figure suivante).



Bouton de lancement du programme

Si vous regardez dans votre console, dans la fenêtre du bas sous Eclipse, vous devriez voir quelque chose ressemblant à la figure suivante.



Console d'Eclipse

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` ». Avant que vous arrachiez les cheveux, voici quelques précisions :

- `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` : il s'agit d'un objet de la classe System qui gère la sortie standard.
- `print` : cette méthode écrit dans la console le texte passé en paramètre (entre les parenthèses).

Prenons le code suivant :

```
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 ces solutions, nous obtenons :

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

Avec pour 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 indiquer deux autres caractères d'échappement :

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

Vous avez sûrement remarqué que la chaîne de caractères que l'on affiche est entourée par des guillemets doubles ". En Java, les guillemets doubles sont des délimiteurs de chaînes de caractères. Si vous voulez afficher un guillemet double dans la sortie standard, vous devrez « l'échapper » avec le signe `\`, ce qui donnerait : `"Coucou mon \"chou\" !"`. 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.

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

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 sources 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 OS 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 aux instructions suivantes :
  - `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.