

Traducteur : Jean-Pierre VIDAL

25.04.2001 - Version 0.2 :

- Mise en forme du code html (armel).

19.08.2000 - Version 0.1 :

- Dernière mise à jour de la version française

# Préface

J'ai suggéré à mon frère Todd, qui est en train de migrer du hardware vers le software, que la prochaine grande révolution serait l'ingénierie génétique.

Nous créerons bientôt des organismes dédiés à la fabrication de nourriture, de carburant, de plastique ; ils digéreront la pollution et, de manière générale, nous permettront de maîtriser la manipulation du monde réel, ceci pour un coût minime comparé à celui d'aujourd'hui. J'ai prétendu que la révolution informatique serait vraiment peu de chose au regard de cela.

Puis j'ai réalisé que j'étais en train de commettre une erreur triviale chez les auteurs de science-fiction : oublier le propos de la technologie (ce qui est évidemment très facile à faire en science-fiction). Un écrivain expérimenté sait qu'on ne raconte jamais une histoire à propos de choses, mais de gens. La génétique aura un très grand impact sur nos vies, mais je ne suis pas persuadé qu'elle éclipsera la révolution informatique (qui d'ailleurs rend possible la révolution génétique) — ou au moins la révolution de l'information. L'information, c'est essentiellement se parler les uns les autres : bien entendu, les voitures, les chaussures, et surtout les maladies génétiques sont importantes, mais en définitive ce ne sont que des faux-semblants. La vraie question est notre relation au monde. Ainsi en est-il de la communication.

Ce livre est un cas. Une majorité d'amis pensa que j'étais soit vraiment hardi soit légèrement dérangé pour mettre tout cela sur le Web. « Pourquoi quelqu'un voudrait-il l'acheter ? » me disaient-ils. Si j'avais eu un tempérament plus conservateur, je ne m'y serais pas pris de cette manière ; en réalité je ne voulais pas écrire un livre supplémentaire de style traditionnel sur les ordinateurs. Je ne savais ce qui en aurait résulté si je n'avais pas agi ainsi, mais en tout cas ce fut la meilleure chose que j'ai jamais réalisée avec un livre.

Tout d'abord, chacun commença à m'envoyer des correctifs. Ce fut un processus très amusant, parce que mes amis avaient fureté dans chaque coin et recoin et repéré les erreurs techniques aussi bien que grammaticales, ce qui me permit d'éliminer les fautes de toutes sortes que j'aurais laissé passer sans cela. Dans ce travail, ils ont été tout simplement formidables, commençant très souvent par me dire « bon, je ne dis pas ça pour critiquer... » pour me mettre ensuite sous le nez un ensemble d'erreurs que je n'aurais jamais été capable de trouver par moi-même. Je crois que ce fut une espèce de travail de groupe, et cela a réellement ajouté quelque chose de spécial à ce livre.

Mais ensuite, j'ai commencé à entendre ceci : « OK, très bien, c'est bien gentil vous avez fait une version électronique, mais moi j'aurais préféré une version complète imprimée chez un vrai éditeur ». Alors j'ai beaucoup travaillé afin que chacun puisse l'imprimer dans un format adéquat, mais cela n'a pas suffi à résorber la demande d'un livre « publié ». La plupart des gens n'ont pas envie de lire le livre à l'écran dans son intégralité, et l'idée de transporter un paquet de feuilles volantes, même impeccablement imprimées, ne leur conviendrait pas davantage (de plus, je pense que ce n'est pas forcément économique en terme de toner). Après tout il semblerait que la révolution informatique ne risque pas de mettre les éditeurs au chômage. Un étudiant suggéra toutefois que cela pourrait servir de modèle pour l'édition du futur : les livres seraient d'abord publiés sur le

Web, et, à condition qu'ils rencontrent un certain intérêt, on les coucherait sur papier. Actuellement, une grande majorité de livres représentent des désastres financiers, et cette nouvelle approche pourrait peut-être rendre l'industrie de l'édition plus rentable.

Ce livre a été par ailleurs une expérience enrichissante pour moi. Ma première approche de Java fut « juste un nouveau langage de programmation », ce qu'il est de fait par ailleurs. Mais, le temps passant, et au fur et à mesure que je l'étudiais plus en profondeur, je commençais à m'apercevoir que son propos fondamental était différent de celui de tous les langages que j'avais connu auparavant.

Programmer, c'est gérer la complexité : complexité du problème que l'on veut résoudre, superposée à la complexité de la machine sur laquelle il va être résolu. Bien des projets de programmation ont avorté à cause de cette complexité. Je voudrais maintenant dire que, de tous les langages de programmation que je connaisse, aucun n'a été conçu pour gérer la complexité du développement et de la maintenance de programmes [1]. Bien entendu, dans la conception des langages, beaucoup de décisions ont été prises en gardant la complexité présente à l'esprit, mais, dans chaque exemple et à partir d'un certain moment, d'autres problèmes ont surgi qui furent considérés comme essentiels et par suite intégrés à la mixture. Fatalement, ces nouveaux problèmes furent de ceux qui envoyèrent finalement les programmeurs « droit dans le mur » avec ce langage. Par exemple, C++ se devait de posséder une compatibilité ascendante avec C (afin de favoriser la migration des programmeurs C), ainsi qu'une certaine efficacité. Ces deux buts étaient très utiles et ont grandement participé au succès de C++, mais dans le même temps ils ont généré une complexité supplémentaire qui a eu pour résultat d'empêcher certains projets d'arriver à terme (bien entendu, vous pouvez toujours vous en prendre à la responsabilité des programmeurs et/ou de leur encadrement, mais, si un langage pouvait vous aider à repérer vos erreurs, pourquoi ne le ferait-il pas ?). Autre exemple, Visual Basic (VB) était lié à BASIC, lequel n'était pas vraiment conçu comme un langage extensible, et par suite toutes les extensions superposées à VB se sont traduites par une syntaxe vraiment horrible et impossible à maintenir. Perl a une compatibilité ascendante avec Awk, Sed, Grep ainsi que d'autres outils Unix qu'il était censé remplacer, le résultat est qu'il est souvent accusé de produire du « code-à-écrire-seulement » (c'est à dire qu'on est incapable de se relire quelques mois plus tard). D'un autre côté, C++, VB, Perl, et d'autres langages comme Smalltalk, de par leur conception, ont abordé le problème de la complexité, et par là se révèlent remarquablement efficaces dans la résolution de certains types de problèmes.

Ce qui m'a le plus frappé alors que je commençais à comprendre Java est quelque chose qui s'apparente à l'incroyable finalité de diminuer la complexité *pour le programmeur*. Un peu comme si l'on disait « rien n'est important, mis à part réduire le temps et la difficulté pour produire un code robuste ». Dans les premières versions de Java, cette finalité s'est traduite par un code qui ne tournait pas très vite (bien que l'on ait fait de nombreuses promesses concernant la vitesse de Java) mais il n'empêche que cela a réduit le temps de développement de manière stupéfiante : la moitié, ou moins, du temps nécessaire à la création d'un programme équivalent en C++. Ce seul résultat pourrait déjà se traduire par une incroyable économie de temps et d'argent, mais Java ne s'arrête pas là. Il s'attache à encapsuler toutes les tâches complexes qui ont pris de l'importance, comme le multithreading et la programmation réseau, au moyen de fonctionnalités du langage ou de bibliothèques rendant parfois ces tâches triviales. Et pour finir, il traite quelques problèmes d'une réelle complexité : programmes multi-plate-forme, changements dynamiques de code, sans oublier la sécurité, chacun d'entre eux pouvant s'adapter à votre gamme de difficultés, depuis un « obstacle » jusqu'à un « point bloquant ». Ainsi, malgré les problèmes de performance que nous avons vus, les promesses de Java sont énormes : il est capable de faire de nous des programmeurs encore plus productifs.

Le Web est l'un des lieux où cela se révèle le plus. La programmation réseau a toujours été difficile, et Java la rend facile (et les concepteurs du langage Java travaillent à la rendre encore plus facile). La programmation réseau, c'est ce qui fait que nous parlons entre nous de manière plus pertinente et meilleur marché que nous ne l'avons jamais fait avec le téléphone (l'email à lui seul a révolutionné bien des entreprises). Alors que nous nous parlons de plus en plus, des choses sensationnelles commencent à émerger, peut-être plus incroyables que celles promises par l'ingénierie génétique.

Dans tous les cas — en créant des programmes, en travaillant en groupe pour créer des programmes, en concevant des interfaces utilisateur permettant aux programmes de dialoguer avec l'utilisateur, en faisant tourner des programmes sur différents types de machine, en écrivant facilement des programmes qui communiquent au travers de l'Internet — Java accroît la bande passante de la communication entre les gens. Je pense que le but de la révolution de la communication n'est certainement pas de transmettre de grandes quantités de bits ; la vraie révolution sera là lorsque nous serons tous capables de nous parler plus facilement : de personne à personne, mais aussi en groupe, et, pourquoi pas, sur la planète entière. J'ai entendu dire que la prochaine révolution verra l'émergence d'une espèce d'esprit global associant un grand nombre de personnes à un grand nombre d'interconnexions. Il se peut que Java soit, ou pas, l'outil de cette révolution, mais en tout cas cette possibilité m'a fait comprendre qu'il n'était pas insensé de tenter d'enseigner ce langage.

## Préface à la 2ème édition

Les lecteurs de la première édition de ce livre ont fait énormément de commentaires élogieux à son propos, ce qui m'a été très agréable. Toutefois de temps à autres l'un ou l'autre s'est plaint, et l'une des critiques récurrentes est « le livre est trop gros ». Dans mon esprit, je dois dire que si « trop de pages » est votre seule plainte, c'est une faible critique (on se souvient de l'empereur d'Autriche se plaignant du travail de Mozart : « trop de notes ! », mais n'allez pas penser que je cherche d'une manière ou d'une autre à me comparer à Mozart). De plus, je peux seulement supposer qu'une telle demande provient d'une personne qui en est encore à prendre conscience de l'étendue du langage Java lui-même, et qui n'a pas encore vu ou lu les autres livres traitant du sujet — par exemple, ma référence favorite, *Core Java* de Cay Horstmann & Gary Cornell (Prentice-Hall), qui a tellement grossi qu'on a dû le scinder en deux volumes. Malgré cela, une des choses que j'ai essayé de faire dans cette édition a été d'éliminer les parties obsolètes, ou tout au moins non essentielles. Je n'ai pas eu de scrupules en faisant cela car l'original existe toujours sur le site Web ainsi que sur le CD ROM qui accompagne ce livre, sous la forme de la première édition du livre, librement téléchargeable (<http://www.BruceEckel.com>). Si c'est l'ancienne version qui vous intéresse, elle existe encore, et ceci est un merveilleux soulagement pour un auteur. Par exemple, vous pouvez remarquer que le dernier chapitre de l'édition originale, « Projects », a disparu ; deux des projets ont intégré d'autres chapitres, et le reste n'avait plus d'intérêt. Pareillement, le chapitre « Design Patterns », devenu trop gros, a fait l'objet d'un livre séparé (également téléchargeable sur le site Web). Ainsi, logiquement, le livre devrait être plus mince.

Mais, hélas, il n'en sera pas ainsi.

Le plus gros problème est le continuel développement du langage Java lui-même, et en particulier l'extension de l'API qui nous promet de nous procurer une interface standard pour tout ce que nous pourrions imaginer (et je ne serais pas surpris de voir paraître une API « JCafetiere » pour couronner le tout). Le propos de ce livre n'est certainement pas de parler de ces API, d'autres auteurs se chargeront de cette tâche, mais certaines questions ne peuvent être ignorées. Parmi les plus importantes, Java « côté serveur » (principalement les Servlets et les Java Server Pages, ou *JSP*), qui représentent réellement une excellente solution au problème du World Wide Web, pour lequel nous avons découvert que les divers navigateurs Web ne sont pas suffisamment consistants pour traiter la programmation côté client. En outre, un problème reste entier, celui de créer facilement des applications qui se connectent à des bases de données, qui supervisent des transactions, qui gèrent la sécurité, etc., ce que traitent les Enterprise Java Beans (EJBs). Ces sujets se retrouvent dans le chapitre anciennement nommé « Programmation Réseau », appelé maintenant « Informatique Distribuée », un sujet qui est en passe de devenir essentiel. Ce chapitre a également grossi afin d'inclure une vue d'ensemble de Jini (prononcez « djini », ce n'est pas un acronyme, mais un nom), qui est une technologie de pointe permettant de concevoir différemment les interconnexions entre applications. Et, bien entendu, le livre a évolué pour utiliser tout au long des exemples la bibliothèque de composants graphiques Swing (GUI, Graphics User Interface, Interface Graphique Utilisateur, NdT). Ici aussi, si vous vous intéressez aux anciennes versions Java 1.0/1.1, vous les trouverez dans le livre que vous pouvez télécharger gratuitement à <http://www.BruceEckel.com> (et qui est

également inclus dans le CD ROM fourni avec cette nouvelle édition ; vous en saurez davantage à ce sujet un peu plus tard).

Outre les quelques nouvelles fonctionnalités du langage Java 2 et les corrections effectuées dans tout le livre, un autre changement majeur est le chapitre sur les collections (chapitre 9), qui met maintenant l'accent sur les collections Java 2 utilisées tout au long du livre. J'ai également amélioré ce chapitre pour traiter plus en profondeur certaines facettes des collections, entre autres expliquer comment fonctionne une fonction de hashing (afin que vous sachiez comment en créer une convenablement). Il y a eu d'autres changements, tels que la réécriture du Chapitre 1, la suppression de quelques appendices ainsi que d'autres parties qui ne me paraissent plus indispensables pour le livre imprimé, mais ce fut le plus gros des suppressions. D'une manière générale, j'ai essayé de tout revoir, d'enlever de cette 2<sup>e</sup> édition tout ce qui n'était plus indispensable (mais que l'on peut trouver sous la forme électronique de la première édition), de traiter les modifications, et d'améliorer tout ce qui pouvait l'être. Comme le langage continue d'évoluer — mais toutefois pas à la même allure vertigineuse qu'auparavant — il ne fait pas de doute que de nouvelles éditions de ce livre verront le jour.

Je dois m'excuser auprès de ceux qui persistent dans leur critique à propos de la taille du livre. Que vous me croyiez ou non, j'ai travaillé dur pour le rendre plus mince. Malgré sa taille, je pense qu'il existe assez d'alternatives pour vous satisfaire. D'une part, le livre existe sous forme électronique (sur le site Web, ainsi que sur le CD ROM accompagnant ce livre), ainsi lorsque vous prenez votre ordinateur portable vous pouvez également emporter le livre sans supplément de poids. Si vous êtes réellement partisan de la minceur, il existe des versions Palm Pilot (quelqu'un m'a dit qu'il lirait le livre au lit sur l'écran rétro-éclairé de son Palm afin de ne pas déranger sa femme. Je ne peux qu'espérer que cela l'aidera à glisser dans les bras de Morphée). Si vous le préférez sur papier, je connais des personnes qui impriment un chapitre à la fois et l'emmènent dans leur attaché-case afin de le lire dans le train.

## Java 2

Alors que j'écris ceci, la sortie de la version 1.3 du *Java Development Kit* (JDK) de Sun est imminente, et les modifications proposées pour le JDK 1.4 ont été publiées. Bien que ces numéros de version soient encore dans les « uns », la manière standard de se référer à une version du JDK 1.2 ou supérieur est de l'appeler « Java 2 ». Ceci souligne les modifications significatives entre le « vieux Java » — qui possède beaucoup de verrues, ce que je critiquais dans la première version de ce livre — et la nouvelle version du langage, améliorée et plus moderne, comportant bien moins de verrues et beaucoup de compléments, ainsi qu'une conception agréable.

Ce livre est écrit pour Java 2. J'ai la grande chance de dominer l'ancien langage et de n'écrire que pour le nouveau langage amélioré, parce que l'ancienne information existe encore dans la 1<sup>re</sup> édition sur le Web et sur le CD ROM (ce qui représente votre source d'information si vous utilisez une version antérieure à Java 2). D'autre part, et parce que n'importe qui peut librement télécharger le JDK depuis [java.sun.com](http://java.sun.com), cela signifie qu'en écrivant sur Java 2 je n'oblige personne à une contrainte budgétaire élevée en lui imposant une mise à jour.

Il y a toutefois une nuance. JDK 1.3 possède quelques améliorations que j'aimerais réellement utiliser, mais la version de Java actuellement fournie pour Linux est le JDK 1.2.2. Le système Linux (voir <http://www.Linux.org>) est un développement très important en conjonction avec Java, parce qu'il est en train de devenir rapidement la plus importante plate-forme serveur — rapide, fiable, robuste, sécurisée, bien maintenue, et gratuite, une vraie révolution dans l'histoire de l'informatique (je ne me souviens pas avoir jamais rencontré l'ensemble de ces fonctionnalités dans aucun outil auparavant). Et Java a trouvé une très importante niche dans la programmation côté serveur sous la forme des *Servlets*, une technologie qui est une énorme amélioration de la programmation CGI traditionnelle (ceci est décrit dans le chapitre « Informatique Distribuée »).

Aussi, malgré le fait que j'aimerais n'utiliser que les toutes nouvelles fonctionnalités de Java, il est essentiel que

l'ensemble puisse être compilé sous Linux, et donc que lorsque vous installerez le code source et que vous le compilerez sous cet OS (avec le dernier JDK) vous puissiez constater que l'ensemble peut être compilé. Toutefois, vous verrez que le texte est parsemé çà et là de notes à propos des fonctionnalités du JDK 1.3.

## Le CD ROM

Un autre bonus de cette édition est le CD ROM que vous trouverez à la fin du livre. J'ai naguère rejeté cette idée, pensant que mettre quelques Ko de code source sur cet énorme CD n'était pas justifié, et préféré à cela que les gens le téléchargent depuis mon site Web. Cependant, vous allez voir tout de suite que ce CD ROM représente autre chose.

Le CD contient le code source que l'on trouve dans le livre, mais il contient aussi l'intégralité du livre, sous différents formats électroniques. Mon préféré est le format HTML, parce qu'il est rapide et complètement indexé — vous cliquez simplement sur une entrée de l'index ou de la table des matières et vous vous retrouvez immédiatement à l'endroit voulu dans le livre.

Toutefois la plus grande partie des 300 Mo du CD consiste en un ensemble multimédia complet nommé *Thinking in C : Foundations for C++ & Java*. A l'origine, j'avais délégué à Chuck Allison la création de ce « séminaire sur CD ROM » en tant que produit à part entière, puis j'ai décidé de l'inclure dans les secondes éditions de *Thinking in C++* et de *Thinking in Java* après avoir vécu, lors d'un séminaire, l'arrivée de personnes dépourvues de connaissances suffisantes en langage C. Leur propos était apparemment « je suis un bon programmeur et je n'ai *pas envie* d'apprendre le C, mais plutôt C++ ou Java, c'est pourquoi je compte passer rapidement sur C pour aller directement à C++/Java ». Peu après leur arrivée au séminaire, ils prennent conscience que le prérequis de la connaissance de la syntaxe du C se trouve là pour d'excellentes raisons. En incluant le CD ROM dans le livre, nous nous assurons que chaque participant à un séminaire a une préparation suffisante.

Le CD permet également d'élargir l'audience du livre. Même si le chapitre 3 (Contrôle du flux de programme) traite de parties fondamentales de Java provenant du langage C, le CD est une introduction en douceur, et à l'inverse du livre suppose de la part de l'étudiant une moindre connaissance de la programmation. Le CD étant inclus dans le livre, j'espère que davantage de personnes intégreront le cercle de la programmation Java.

---

[1] j'ai enlevé ceci de la 2<sup>ème</sup> édition : je pense que le langage Python est très proche de faire exactement cela. Voir <http://www.Python.org>.